Wrocław University
of Science and Technology

Improving Visual Simultaneous Localization and Mapping robustness.

Author:
Marcin OCHMAN

Supervisor:
prof dr hab. inż. Ewaryst RAFAJŁOWICZ

Faculty of Information and Communication Technology

September 10, 2024

"Science means constantly walking a tightrope between blind faith and curiosity; between expertise and creativity; between bias and openness; between experience and epiphany; between ambition and passion; and between arrogance and conviction – in short, between and old today and a new tomorrow."

Henrich Rohrer

Abstract

Improving Visual Simultaneous Localization and Mapping robustness.

by Marcin OCHMAN

Keywords: Image Processing, SLAM, VSLAM, Localization, Mapping, RGBD camera, RGB camera

This thesis presents new advancements in the field of Visual Simultaneous Localization and Mapping (VSLAM), focusing on enhancing the robustness and flexibility of VSLAM systems. The motivation behind this research stems from the growing significance of computer vision and its applications. The thesis offers several key contributions.

At the beginning of this work, both SLAM and VSLAM systems are discussed in detail. Forty years of research on localization and mapping have been compiled and analyzed, providing a comprehensive overview of the development and current state of VSLAM systems. A table of various SLAM systems has been created, serving as a guide for researchers, facilitating the effective comparison of these systems and aiding in the identification of their strengths and weaknesses. The thesis also identifies and addresses key challenges, such as outlier errors, dynamic environments, map maintenance, long-term operation, and others. These challenges are discussed and illustrated in detail to highlight their impact on the robustness of VSLAM.

It also redefines the concept of robustness in VSLAM, proposing a multifaceted approach that considers algorithmic robustness, software resilience, and computational efficiency. Each of these aspects is thoroughly analyzed and addressed in subsequent chapters of the thesis.

The concept of Modular SLAM, an extensible architecture designed to overcome the limitations of existing SLAM systems, is introduced. It supports rapid prototyping, allowing researchers to explore new solutions efficiently. Modular SLAM combines advancements in system architecture and VSLAM to build a modular and flexible solution.

Additionally, two novel methods are introduced: VSLAM SuperPoint, a deep learning-based feature detection technique that leverages image sequence data to improve keypoint repeatability, and VSLAM RANSAC, an enhanced pose estimation method utilizing historical data to increase robustness.

Finally, directions for future research are outlined, highlighting the potential for further innovations and applications of Modular SLAM.

Streszczenie

Metody zwiększające odporność procesu lokalizacji i mapowania z wykorzystaniem systemów wizyjnych.

Marcin OCHMAN

Słowa kluczowe: Przetwarzanie obrazów, SLAM, VSLAM, Lokalizacja, Mapowanie, Kamera RGBD, Kamera RGB

Niniejsza rozprawa przedstawia nowe rozwiązania w dziedzinie Lokalizacji i Mapowania Wizualnego (VSLAM), koncentrując się na zwiększeniu odporności i elastyczności systemów VSLAM. Motywacją do przeprowadzenia badań było rosnące znaczenie wizji komputerowej oraz jej zastosowań. W pracy zaprezentowano szereg istotnych osiągnięć.

Na początku rozprawy szczegółowo omówiono zarówno systemy SLAM, jak i VSLAM. Czterdzieści lat badań nad lokalizacją i mapowaniem zostało zebranych i przedstawionych w przystępny sposób, dostarczając szerokiej analizy rozwoju oraz aktualnego stanu systemów VSLAM. Opracowano także tabelę różnych systemów SLAM, która służy jako przewodnik dla badaczy, umożliwiając efektywne porównanie tych systemów oraz ułatwiając identyfikację ich mocnych i słabych stron. Praca identyfikuje i opisuje wyzwania, takie jak błędy grube, dynamiczne środowisko, utrzymanie mapy, długoterminowe działanie i inne. Wyzwania te są szczegółowo omówione i zilustrowane, aby podkreślić ich wpływ na odporność systemów VSLAM.

Rozprawa redefiniuje pojęcie odporności w kontekście VSLAM, proponując wieloaspektowe podejście, które obejmuje odporność algorytmiczną, oprogramowania oraz czas przetwarzania. Każdy z tych aspektów został szczegółowo przeanalizowany i zaadresowany w dalszej części pracy.

Wprowadzono koncepcję Modular SLAM o rozszerzalnej architekturze, zaprojektowaną w celu przezwyciężenia istniejących ograniczeń systemów SLAM. Modular SLAM wspiera szybkie prototypowanie, umożliwiając naukowcom efektywne eksplorowanie nowych rozwiązań. System ten wykorzystuje zalety różnych zasad budowy oprogramowania oraz wzorców projektowych, tworząc modułowe i elastyczne rozwiązanie.

Dodatkowo, w pracy zaprezentowano dwie nowatorskie metody: VSLAM SuperPoint, technikę detekcji cech opartą na uczeniu głębokim, która wykorzystuje sekwencyjność obrazów do poprawy powtarzalności wykrywania punktów charakterystycznych, oraz VSLAM RANSAC, usprawnioną metodę estymacji pozycji, która korzysta z danych historycznych w celu zwiększenia odporności systemu.

Na zakończenie, zarysowano kierunki przyszłych badań, wskazując na potencjał dalszych rozwiązań i zastosowań Modular SLAM.

Acknowledgements

I would like to express my deepest appreciation to those who have made the completion of this PhD thesis possible. Foremost, I am profoundly grateful to my mother, father and family, whose support and belief in my capabilities have been the backbone of my academic journey. Your love, encouragement, and sacrifices have been instrumental in reaching this milestone.

A special acknowledgment must be reserved for my supervisor, whose wisdom, guidance, and encouragement have been invaluable. Your unwavering commitment to academic excellence and your belief in my potential have significantly enriched this journey.

Lastly, but by no means least, I want to express my sincere gratitude to my fiancée. Your patience, understanding, and love have been a constant refuge during the demanding periods of this research. Your belief in my work and your support have made this journey easier and more fulfilling.

Contents

Abstract **v**

PL Abstract **vii**

Acknowledgements **ix**

PhD thesis introduction in Polish **1**

1 Introduction **7**
1.1 Thesis Scope and Motivation . . . 8
1.2 Thesis Contribution . . . 8
1.3 Thesis structure . . . 10
1.4 Author contribution . . . 11

2 Visual Simultaneous Localization and Mapping Overview **13**
2.1 Fundamentals of SLAM problem. Probabilistic approach . . . 15
2.2 SLAM categorization . . . 18
2.2.1 Sensors type . . . 19
Ultrasonic sensor . . . 19
Sonar . . . 20
LIDAR . . . 20
Camera . . . 21
RGB-D camera . . . 21
Inertial measurement unit . . . 22
Sensor fusion . . . 23
Sensors comparison and choice . . . 24
2.2.2 Operating environment . . . 25
2.2.3 Backend type . . . 28
Filter-based backend . . . 29
Optimization-based backend . . . 35
2.2.4 Map representation . . . 39
2.3 SLAM Related work . . . 42
2.4 Visual Odometry, Visual SLAM and Structure from Motion . . . 50
2.5 Visual SLAM . . . 57
2.5.1 Visual SLAM problem formulation . . . 58
2.5.2 State and observation representation . . . 61

2.6 Visual SLAM components 63
2.6.1 Sensors data collection 65
2.6.2 Frontend 66
Feature detection 67
Feature matching and data association 69
Keyframe creation 72
2.6.3 Backend 74
Map optimization 74
Map maintenance 74
2.6.4 Loop Closure 75
2.6.5 Semantic VSLAM 77
2.7 VSLAM related work 81

3 VSLAM challenges and robustness **89**
3.1 Keypoints distribution on image 90
3.2 Mapping in dynamic environments 93
3.3 Long-Term operation 94
3.4 Loop Closure 98
3.5 Real-time performance 102
3.6 Outliers handling 106
3.7 Map maintenance 108
3.8 Keyframe creation 108
3.9 Introduction to Robustness 111
3.9.1 Software robustness 113
Simplicity 114
Modularity 115
Reliability 116
Portability 117
Maintainability 117
3.9.2 Algorithms robustness 117
3.9.3 Execution efficiency robustness 118
3.9.4 VSLAM datasets 119
3.10 VSLAM evaluation 123

4 Concept of Modular SLAM **133**
4.1 Motivation 134
4.2 Modular SLAM Overview 136
4.3 System design 136
4.3.1 Design principles 137
Simplicity 137
Modularity 139
Reliability 139
Portability 140
Scalability 141

4.3.2 Design patterns 141
4.3.3 Visualization 148
4.3.4 Evaluation 150
4.4 Practical Application of Modular SLAM 152

5 New VSLAM robust methods 157
5.1 Keypoints detections 158
5.1.1 SuperPoint for VSLAM 163
Dataset generation 164
Model overview 170
Training stage 172
Evaluation 174
5.2 VSLAM RANSAC 177
5.2.1 Evaluation 180

6 Conclusions 187
6.1 Future Work 189

A SLAM projects 193

B Additional resources 199

Bibliography 201

List of Figures

2.1 Simultaneous Localization and Mapping 17
2.2 Typical SLAM architecture 18
2.3 Commonly used sensors in SLAM system. 22
2.4 Examples of structured and unstructured environments. . . . 27
2.5 Examples of indoor, outdoor and underwater environment. . 28
2.6 Timeline of backend types research. 29
2.7 Single iteration of Bayesian filter. 31
2.8 Comparison of Dynamic Bayesian Network with keyframe-based approach. 37
2.9 Example of SLAM factor graph. 38
2.10 Relationships between level of abstraction and type of the map 42
2.11 Timeline of SLAM. 47
2.12 Visual odometry problem. 52
2.13 Flow of the basic VO algorithms. 53
2.14 Structure from motion problem. 54
2.15 Flow of the basic SfM algorithms. 55
2.16 VSLAM correspondences. 59
2.17 Visual SLAM problem with two keyframes and current frame marked with dashed line 60
2.18 Example of VSLAM. 64
2.19 Visual SLAM components 65
2.20 Data synchronization challenge from various sensors 66
2.21 Mobile platform equipped with various sensors. 66
2.22 Comparison of various keypoint detectors 70
2.23 Result of matching procedure. 71
2.24 Various types of data correspondences [You+17]. 71
2.25 Comparison of sparse and dense VSLAM map. 73
2.26 Example of Loop Closure 78
2.27 Kimera VSLAM Semantic Mapping. 80
2.28 Timeline of VSLAM with the most important VSLAM systems. 82

3.1 Comparison of keypoint detection techniques. 91
3.2 Comparison of two systems with and without keypoints distribution technique with the ground-truth. 92
3.3 Example of scene containing moving objects – humans walking in the laboratory. 94

3.4 Visualization of a section of a city mapped using data from the KITTI dataset. 95
3.5 Different conditions of the same scene. 97
3.6 Comparison of APE for systems with and without drift compensation. 99
3.7 Examples of very similar environments but they represent various locations – perceptual aliasing 100
3.8 StellaVSLAM and loop detection misses 102
3.9 Comparison of tracking and mapping times. 104
3.10 Relationship between tracking time and number of local landmarks. 105
3.11 Examples of outlier sources in VSLAM 107
3.12 Example of map maintenance. 109
3.13 Tracked landmarks over time. 111
3.14 Diagram illustrating the various aspects of VSLAM robustness examined in this thesis. 112
3.15 Diagram illustrating the various aspects of software robustness in VSLAM examined in this thesis. 115
3.16 Diagram illustrating the various aspects of algorithmic robustness in VSLAM examined in this thesis. 118
3.17 Diagram illustrating the various aspects of execution efficiency robustness in VSLAM examined in this thesis. 120
3.18 Examples of VSLAM datasets including TUM, KITTI, TartainAir and Bonn. 124
3.19 Comparative Analysis of VSLAM System Performances. . . . 127
3.20 Comparison of VSLAM's APE distribution. 128
3.21 Trajectories plot in 3D. 130
3.22 Trajectories plot in 2D. 130
3.23 Trajectory and temporal position relationships. 131
3.24 Trajectory and temporal angles relationships. 131

4.1 The fundamental components of Modular SLAM and its responsibilities. 137
4.2 UML Diagram of SLAM System Components. 138
4.3 Builder pattern 143
4.4 Observer pattern for handling parameters change 145
4.5 An example of GUI for parameters handling. 146
4.6 Visitor Pattern in VSLAM 147
4.7 Strategy pattern for pose estimation in Modular SLAM 149
4.8 Two aspects of Modular SLAM visualization: real-time operations and final plots. 151
4.9 Strategy pattern for storing trajectory data in Modular SLAM 152

5.1 Detected keypoints on consecutive frames from a VSLAM sequence 159
5.2 Illustration of a sequence from the TartanAir dataset. 160
5.3 Tracked landmarks over time. 162
5.4 Top-down view illustrating the concept of occlusion. 167
5.5 Visualisation of training dataset preparation. 168
5.6 SuperPoint architecture 171
5.7 Determining keypoints from heatmap. 171
5.8 VSLAM SuperPoint architecture 172
5.9 Changing details of heatmap over the training. 175
5.10 Results of landmarks′ observations analysis using StellaVSLAM and KITTI dataset. 181

6.1 Roadmap of future studies. 191

List of Tables

2.1 Comparison of commonly used exteroceptive sensors in SLAM. 26
2.2 The SLAM surveys and overview articles excluding papers dedicated to VSLAM specifically. 48
2.3 Comparison of three interconnected 3D problems utilizing computer vision. 57
2.4 Comparison of descriptors [GHT11; TS18] 69
2.5 Overview of key surveys and papers in VSLAM and VO. . . . 87

3.1 StellaVSLAM performance using distributed keypoints detector compared to classic keypoint detector. 91
3.2 Mapped area of KITTI sequences 96
3.3 List of most popular Visual SLAM datasets. 121
3.4 Comparative Analysis of VSLAM System Performances. . . . 126

5.1 Tracked landmarks in various TUM sequences 164
5.2 Comparison of various detectors and VSLAM SuperPoint. . . 177
5.3 Evaluation results of RANSAC and VSLAM RANSAC – inliers ratio. 183
5.4 Evaluation results of RANSAC and VSLAM RANSAC – absolute position error. 184

A.1 The most significant VSLAM systems I 194
A.2 The most significant VSLAM systems II 195
A.3 The most significant VSLAM systems III 196
A.4 The most significant VSLAM systems IV 197

B.1 List of resources developed for this thesis. 199
B.2 List of Docker images. 200

Listings

4.1 Creating VSLAM system using Modular SLAM 153
4.2 Strategy pattern in Modular SLAM for pose estimation 154
4.3 Combination of strategy and observer pattern for trajectory saving. 155
4.4 Handling parameters in Modular SLAM 155
4.5 Visualisation in Modular SLAM 156
5.1 Algorithm of generating samples for VSLAM SuperPoint. . . . 169
5.2 Algorithm for transforming keypoints from previous frame to current frame. 170
5.3 RANSAC pseudocode 179
5.4 VSLAM RANSAC pseudocode 182

List of Abbreviations

AI	**A**rtificial **I**ntelligence
AR	**A**ugmented **R**eality
BR	**B**undle **A**djustment
CLM	**C**oncurrent **M**apping and **L**ocalization
EKF	**C**ompressed **E**xtended **K**alman **F**ilter
EKF	**E**xtended **K**alman **F**ilter
LIDAR	**Li**ght **D**etection **A**nd **R**anging
SLAM	**S**imultanous **L**ocalization **A**nd **M**apping
VO	**V**isual **O**dometry
VI	**V**isual-**I**ntertial
VR	**V**irtual **R**eality
XR	E**X**tended **R**eality
SONAR	**SO**und **NA**vigation **R**anging
VSLAM	**V**isual **S**imultanous **L**ocalization **A**nd **M**apping

Wstęp

Zakres rozprawy

Wizja komputerowa jest niezwykle istotną dziedziną badań. Naukowcy szacują, że nawet 80% percepcji człowieka zależy od wzroku [RP10], co czyni go najważniejszym zmysłem. Z tego powodu badacze dostrzegają ogromny potencjał, jaki niesie ze sobą wizja. W ostatnich latach społeczność zajmująca się wizją komputerową poczyniła znaczące postępy w zakresie sztucznej inteligencji w takich dziedzinach jak rolnictwo [KP18], opieka zdrowotna [Jia+17] czy robotyka [Mou21].

Dodatkowo, kamery, które są stosunkowo tanie w porównaniu do innych typów, takich jak czujniki ultradźwiękowe czy LIDAR, oferują znacznie większą ilość informacji o otoczeniu. Dlatego też niektóre duże firmy technologiczne rezygnują z innych sensorów na rzecz systemów opartych wyłącznie na wizji [Tes23].

Rosnące zainteresowanie i znaczenie systemów wizyjnych skłoniły autora do podjęcia badań nad nimi. Początek prac nad Modular SLAM wynikał z potrzeby rozwiązania wyzwań i ograniczeń, które istnieją w dotychczasowych systemach SLAM, takich jak brak elastyczności i możliwości rozbudowy, co utrudnia dostosowanie systemów do różnych konfiguracji sensorów, środowisk czy specyficznych wymagań. Motywacją do opracowania Modular SLAM było przezwyciężenie tych ograniczeń poprzez dostarczenie wszechstronnej i modułowej biblioteki do badań i rozwoju SLAM. Dzięki elastycznej architekturze i bogatemu zestawowi narzędzi i funkcjonalności, Modular SLAM umożliwia szybkie prototypowanie. W rezultacie, szerokie grono osób zaangażowanych w VSLAM, w tym badacze, praktycy oraz początkujący, mogą w krótszym czasie opracowywać nowe algorytmy. Uważa się, że Modular SLAM może stać się ważnym narzędziem wielu badaczy SLAM. Ponadto, niniejsza rozprawa koncentruje się na zwiększeniu odporności systemów VSLAM. Dzięki Modular SLAM łatwiej jest testować i porównywać proponowane metody z innymi rozwiązaniami.

Wkład pracy

Najważniejszymi osiągnięciami tej rozprawy jest szereg rozwiązań mających na celu poprawę odporności systemów VSLAM. Nie tylko wypełniają istniejące luki w literaturze, ale także otwierają nowe możliwości badań i praktycznych wdrożeń w VSLAM. Osiągnięcia te obejmują:

- **Kompleksowy przegląd literatury z ostatnich czterech dekad**, który dostarcza szeroką analizę rozwoju i obecnego stanu VSLAM. Przegląd ten podkreśla kluczowe kamienie milowe oraz najbardziej znaczące algorytmy i podejścia. Dodatkowo, stworzono rozszerzoną tabelę różnych systemów SLAM, która służy jako przewodnik dla badaczy umożliwiając efektywne porównanie tych systemów, ułatwiając identyfikację ich mocnych i słabych stron.

- **Identyfikacja kluczowych wyzwań**, które są istotne dla wydajności i niezawodności systemów VSLAM. Niniejsza rozprawa dostarcza szczegółowy opis i analizę wyzwań, takich jak błędy grube, dynamiczne środowisko, utrzymywanie mapy, długoterminowe działanie i inne. Te wyzwania są szczegółowo omówione i zilustrowane, aby podkreślić ich wpływ na odporność VSLAM.

- **Redefinicja odporności**, z uwzględnieniem różnorodnych interpretacji tego pojęcia w środowisku naukowym. W rozprawie zaproponowana jest nowa, bardziej wszechstronna definicja odporności VSLAM, która obejmuje kilka kluczowych aspektów, takich jak odporność algorytmiczna, odporność oprogramowania i czas przetwarzania. Ta definicja różni się od tradycyjnego podejścia w literaturze, które często koncentruje się tylko na pierwszym, podstawowym aspekcie. Ponadto, każdy z tych elementów jest szczegółowo opisany, z wyjaśnieniem, jak przyczyniają się do rozwoju niezawodnego systemu VSLAM.

- **Modular SLAM** stanowi znaczący wkład, będąc fundamentem przeprowadzonych badań. Koncepcja Modular SLAM wprowadza modułową architekturę umożliwiającą elastyczność i rozszczerzalność systemu SLAM. Dostarcza wszechstronnych narzędzi do szybkiego prototypowania i oceny algorytmów SLAM, umożliwiając eksplorację i implementację różnych strategii dla poszczególnych komponentów systemu. Dzięki swojej adaptacyjnej konstrukcji i bogatej funkcjonalności, Modular SLAM otwiera nowe możliwości eksperymentowania oraz dokonywania szybszych postępów w dziedzinie SLAM, stanowiąc cenne narzędzie dla badaczy i praktyków.

- **VSLAM SuperPoint** wprowadza nową metodę detekcji punktów charakterystycznych (ang. *feature detection*) na obrazie za pomocą uczenia głębokiego. Tradycyjne systemy VSLAM zazwyczaj wykrywają punkty kluczowe (ang. *landmark*), wykorzystając aktualny obraz. Jednakże, w VSLAM przetwarza się sekwencje kolejnych obrazów, dlatego istnieje możliwość wykorzystania informacji o pozycjach wcześniej wykrytych punktów kluczowych. W rozprawie nie tylko zaproponowano udoskonaloną architekturę modelu zdolną do generowania mapy cieplnej wcześniej wykrytych punktów kluczowych, ale także wprowadza zmodyfikowaną funkcję straty. Dodatkowo, zamiast polegać na macierzy homografii, opracowano nową metodę generowania pozycji punktów kluczowych opartą na sekwencjach VSLAM, rzeczywistych ruchach kamery oraz mapy głębi.

- **VSLAM RANSAC** jest ulepszonym podejściem, które wykorzystuje dane historyczne w systemie VSLAM do dokładniejszego oszacowania pozycji kamery. RANSAC radzi sobie z wartościami odstającymi poprzez losowe próbkowanie i iteracyjne testowanie części wybranych punktów w celu oszacowania bieżącej pozycji. Jednakże w VSLAM można użyć dwóch dodatkowych miar do bardziej efektywnego wyboru punktów i poprawy szybkości estymacji. Reliable RANSAC wykorzystuje miarę niezawodności punktu orientacyjnego i odległość deskryptorów dopasowanych punktów kluczowych.

Struktura Rozprawy

Pozostałe rozdziały rozprawy wyglądają następująco:

Rozdział 2 – jest wprowadzeniem do zagadnień SLAM oraz VSLAM. Formułuje problem oraz definiuje kilka podstawowych pojęć używanych w rozprawie. Opisuje również niemal cztery dekady badań dotyczących lokalizacji i mapowania oraz architekturę SLAM.

Rozdział 3 – zagłębia się w wyzwania związane z VSLAM, koncentrując się na kwestiach i potencjalnych błędach w jego implementacji. Złożoności wynikające z dynamicznych, nieznanych środowisk rzeczywistych oraz ich wpływ na dokładność mapowania i lokalizacji są dokładnie zbadane. Stara się odpowiedzieć na pytanie „Czy SLAM jest rozwiązany?”, które również zadał Frese, Wagner, and Röfer w [FWR10]. Omówiono również wady zastosowania kamer w systemach VSLAM, z uwzględnieniem ich wpływu na ogólną wydajność systemu. Problemy związane z obliczeniami i skalowalnością, szczególnie istotne w kontekście dużych i długoterminowych VSLAM, zostały również omówione szczegółowo.

Rozdział 4 – przedstawia przegląd koncepcji Modular SLAM, która stanowi fundament badań prowadzonych w tej rozprawie. Rozdział rozpoczyna się szczegółowym opisem architektury, omawiając jej modułową konstrukcję oraz kluczowe elementy składające się na jej ramy. Ponadto, omówiono zalety, jakie oferuje Modular SLAM, podkreślając jego elastyczność, rozszerzalność oraz dostosowanie do różnych zastosowań VSLAM.

Rozdział 5 – opisuje opracowane metody zwiększenia odporności systemów VSLAM poprzez wprowadzenie dwóch kluczowych innowacji. Pierwszą z nich jest VSLAM SuperPoint, nowatorska metoda detekcji cech oparta na uczeniu głębokim, która poprawia detekcję punktów kluczowych poprzez wykorzystanie sekwencyjnych danych obrazowych, co prowadzi do dokładniejszego i bardziej niezawodnego śledzenia punktów kluczowych. Drugą jest VSLAM RANSAC, udoskonalone podejście do szacowania pozycji, które wykorzystuje dane historyczne, takie jak niezawodność punktów orientacyjnych i odległości deskryptorów, aby podejmować bardziej świadome i efektywne decyzje, zmniejszając tym samym wpływ wartości odstających i zwiększając ogólną odporność systemu.

Rozdział 6 – przedstawia przegląd otrzymanych rezultatów i podkreśla znaczenie badań. Dodatkowo, w rozdziale zaproponowano potencjalne przyszłe kierunki i obszary dalszych badań, wskazując możliwe drogi rozwoju badań i eksploracji nowych możliwości. Na końcu przedstawia kompleksową i ostateczną perspektywę na badania, podsumowując ich wpływ i wyznacza drogę dla przyszłych prac i postępów w tej dziedzinie.

Wkład autora

W trakcie studiów doktoranckich autor opublikował kilka artykułów związanych z wizją komputerową. Poniżej przedstawiono listę wszystkich przygotowanych artykułów naukowych:

1. Marcin Ochman, Magda Skoczeń, Damian Krata, Marcin Panek, Krystian Spyra, and Andrzej Pawłowski. “RGB-D Odometry for Autonomous Lawn Mowing”. In: *Artificial Intelligence and Soft Computing*. Springer International Publishing, 2021, pp. 81–90. [Link]

2. Magda Skoczeń, Marcin Ochman, Krystian Spyra, Maciej Nikodem, Damian Krata, Marcin Panek, and Andrzej Pawłowski. “Obstacle Detection System for Agricultural Mobile Robot Application Using RGB-D Cameras”. In: *Sensors* 21.16 (Aug. 2021), p. 5292. [Link]

3. Wojciech Macherzyński, Marcin Ochman, Zbigniew Kulas, Krzysztof Dudek, Mateusz Didyk, and Dawid Sroczyński. “The Use of Thermovision for Leak Detection in the Automotive Sector”. In: *Pomiary Automatyka Robotyka* 25.3 (Sept. 2021), pp. 79–85. [Link]

4. D. Krata, M. Ochman, M. Panek, M. Skoczen, K. Spyra, Z. Kulas, D. Sroczynski, and A. Pawlowski. “Adaptive Smith Predictor Control Scheme

for a Nonlinear Hydraulic System". In: *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sept. 2021. [Link]

5. Marcin Ochman. "Hybrid approach to road detection in front of the vehicle". In: *IFAC-PapersOnLine* 52.8 (2019), pp. 245–250. [Link]

Wiedza oraz umiejętności zdobyte na przestrzeni lat, które zostały wykorzystane w rozprawie doktorskiej oraz w niektórych publikacjach naukowych, jest również wynikiem prac nad grantami badawczymi finansowanymi przez Narodowe Centrum Badań i Rozwoju. Autor brał udział w trzech grantach wymienionych poniżej:

1. **Connected Worker** / *POIR.01.01.01-00-1229/20*
   „Technologia AR wykorzystująca przestrzenne mapowanie i śledzenie otoczenia w czasie rzeczywistym oraz przetwarzanie brzegowe lub chmurowe w celu zapewnienia efektywności komunikacji i współpracy w obszarach działania sieci 3G/4G/5G"

2. **Kosiarka Autonomiczna** *POIR.01.01.01-00-1069/18*
   „Zaprojektowanie i zwalidowanie w warunkach rzeczywistych autonomicznej kosiarki do trawy do użytku profesjonalnego, w tym opracowanie nowatorskich, dedykowanych algorytmów sterowania"

3. **Leak Detector** / *POIR.01.01.01-00-0773/17*
   „Opracowanie i weryfikacja w warunkach rzeczywistych innowacyjnego detektora nieszczelności opartego o metodę termografii, dedykowanego w szczególności branży automotive"

Jednym z istotnych osiągnięć autora było opracowanie i uzyskanie polskiego patentu:

1. Wojciech Macherzyśnki, Marcin Ochman, Mateusz Didyk, Krzysztof Dudek, Zbigniew Kulas, and Dawid Sroczyński. *Method of leak detection, especially in closed-volume systems*. Pat.242047. Oct. 2022

Chapter 1

Introduction

Overview

This chapter aims to answer several questions regarding the thesis: *"What is it about?"*, *"What problems does it solve?"* and *"What is its main contribution?"*. It highlights author's motivation and goals. Moreover, it is a brief summary of its content, conducted research, developed methods and achievements.

At the beginning it focuses on the role of computer vision with reference to localization and mapping. It explains why progress made on Visual Simultaneous Localization and Mapping may be important for other scientists and engineers. Second, it introduces thesis scope and issues raised in this work. Then, there is a section focusing on main contributions of the thesis. Finally, to ease navigation through this work document structure is presented and briefly described.

In addition, this thesis is a result of many years of PhD studies and research. That is why author's research papers, grants and contributions were listed.

"Scientific research is one of the most exciting and rewarding of occupations."
— Frederick Sanger

1.1 Thesis Scope and Motivation

Computer vision is a very important field of study due to several reasons. Researchers suppose that up to 80% of human's perception and cognition are dependent on sight [RP10], becoming the most important sense. That is why scientists are aware of the possibilities brought by vision. In recent years, computer vision community made a big progress with artificial intelligence in different disciplines including agriculture [KP18], healthcare [Jia+17] or robotics [Mou21].

Moreover, vision sensors, which are relatively inexpensive compared to other types such as ultrasonics or LIDARs, offer a significantly greater amount of information about the environment in comparison to the aforementioned sensors. That's why some big technological companies abandon other sensors in favour of vision-only systems [Tes23].

The steady rise in popularity, interest and importance of vision systems caused the author to start working with them as well. The beginning of Modular SLAM framework stems from the need to address the challenges and limitations present in existing SLAM systems including the lack of flexibility and extensibility, making it difficult to adapt the systems to different sensor configurations, environments, or specific requirements. The motivation behind the development of the Modular SLAM is to overcome these limitations by providing a versatile and modular framework for SLAM research and development. By offering a flexible architecture and a comprehensive set of tools and functionalities, Modular SLAM offers fast prototyping. As a consequence, wide range of people involved in VSLAM including advanced researchers, practitioners as well as beginners may explore novel algorithms in significantly shorter time. It is believed that Modular SLAM will become important tool of many SLAM researchers. Furthermore, this thesis focuses on robustness of VSLAM systems. By providing Modular SLAM framework, it is easier to test and compare proposed methods to other solutions.

1.2 Thesis Contribution

The most significant contributions of this thesis include a series of advancements aimed at improving the robustness of VSLAM systems. These contributions not only fill existing gaps in the literature but also pave the way for future research and practical implementations in VSLAM. These contributions are:

- **A comprehensive review of research conducted over the past four decades**, providing an in-depth analysis of the evolution and current state of VSLAM. This review highlights key milestones, the most significant algorithms and approaches. Additionally, an extended table of various SLAM systems was created, serving as a reliable guide for researchers to compare these systems effectively and facilitating the identification of strengths and weaknesses across different approaches.

- **Identification of key challenges**, which are critical to the performance and reliability of VSLAM systems. This thesis provides a detailed description and analysis of several significant challenges, including outlier handling, dynamic environments, map maintenance, long-term operation and more. These challenges are thoroughly illustrated and discussed to highlight their impact on VSLAM robustness.

- **Redefining robustness**, recognizing that the understanding of robustness can vary across the scientific community. In this thesis, a new, more comprehensive definition of VSLAM robustness is proposed. It is characterized as a combination of several key aspects, including algorithmic robustness, system robustness, and processing time. This definition differs from the traditional focus in the literature, which often emphasizes only one primary aspect. Additionally, each of these components is described in detail, explaining how they collectively contribute to the development of a reliable VSLAM system.

- **Modular SLAM** stands as a significant contribution to this PhD thesis, serving as a fundamental component of the research conducted. Developed as part of the study, this concept embodies a modular architecture that enables flexible and extensible SLAM capabilities. It provides a versatile framework for rapid prototyping and evaluation of SLAM algorithms, facilitating the exploration and implementation of various frontend, backend, and mapping strategies. With its adaptable design and comprehensive functionality, the Modular SLAM framework opens new avenues for experimentation, innovation, and advancements in the field of SLAM, serving as a valuable tool for researchers and practitioners.

- **VSLAM SuperPoint** introduces a novel method for feature detection using deep learning. Traditional VSLAM systems typically detect keypoints using only the current image. However, since VSLAM processes sequences of consecutive images, there is an opportunity to incorporate additional information about the positions of previously detected keypoints. VSLAM SuperPoint extends deep learning-based feature detection by leveraging this information. This thesis not only proposes an enhanced model architecture capable of generating a heatmap of previously detected keypoints but also introduces a modified loss function/

Additionally, a new method for generating ground truth keypoint positions is developed, which is based on VSLAM sequences and real-world camera movement, rather than relying on homography as it was done by the authors of SuperPoint [DMR18].

- **VSLAM RANSAC** is an enhanced approach that leverages historical data within the VSLAM system for more accurate pose estimation. Traditionally, RANSAC addresses outliers by randomly sampling and iteratively testing various data points to estimate the current pose. However, in VSLAM, two additional metrics can be used to select data points more effectively and improve estimation speed. VSLAM RANSAC utilizes the reliability metric of a landmark and the descriptor distance of keypoint matching to make more informed and efficient data point selections.

1.3 Thesis structure

The remaining chapters of the thesis is organized as follows.

Chapter 2 – is an introduction to Simultaneous Localization and Mapping (SLAM) and Visual SLAM (VSLAM). It formulates the problem and defines several basic terms used in the thesis. It also describes almost four decades of research regarding localization and mapping and architecture of SLAM.

Chapter 3 – delves into the challenges in the field of VSLAM, focusing on issues and potential errors inherent in its implementation. The complexities posed by real-world dynamic, unknown environments, and their implications for accurate mapping and localization, are thoroughly explored. It tries to give an answer to the question "Is SLAM solved", which was also asked by Frese, Wagner, and Röfer in [FWR10]. Computational challenges and scalability issues, particularly relevant in the context of large-scale and long-term VSLAM, are also examined in detail.

Chapter 4 – provides an overview of the Modular SLAM concept, which serves as the foundation for the research conducted in this thesis. The chapter begins by presenting a detailed description of the architecture of the framework, outlining its modular design and the key components that constitute its framework. Furthermore, the advantages offered by the Modular SLAM are discussed, emphasizing its flexibility, extensibility, and adaptability to diverse VSLAM applications.

Chapter 5 – describes developed methods to increase robustness of VSLAM systems by introducing two key innovations. The first is VSLAM SuperPoint, a novel deep learning-based feature detection method that enhances keypoint detection by utilizing sequential image data, leading to more accurate and reliable keypoint tracking. The second is VSLAM RANSAC, an improved approach to pose estimation that leverages historical data, such as the reliability of landmarks and descriptor distances, to make more informed and efficient decisions, thereby reducing the impact of outliers and improving the overall robustness of the system.

Chapter 6 – presents an overview of the main outcomes and highlights the significance of the study. Additionally, the chapter explores potential future directions and areas for further investigation, outlining possible avenues for expanding on the research and exploring new opportunities. Finally, it provides a comprehensive and conclusive perspective on the study, encapsulating its impact and setting the stage for future work and advancements in the domain.

1.4 Author contribution

During PhD studies author published several papers related to computer vision. The following list shows all prepared research articles:

1. Marcin Ochman, Magda Skoczeń, Damian Krata, Marcin Panek, Krystian Spyra, and Andrzej Pawłowski. "RGB-D Odometry for Autonomous Lawn Mowing". In: *Artificial Intelligence and Soft Computing*. Springer International Publishing, 2021, pp. 81–90. [Link]

2. Magda Skoczeń, Marcin Ochman, Krystian Spyra, Maciej Nikodem, Damian Krata, Marcin Panek, and Andrzej Pawłowski. "Obstacle Detection System for Agricultural Mobile Robot Application Using RGB-D Cameras". In: *Sensors* 21.16 (Aug. 2021), p. 5292. [Link]

3. Wojciech Macherzyński, Marcin Ochman, Zbigniew Kulas, Krzysztof Dudek, Mateusz Didyk, and Dawid Sroczyński. "The Use of Thermovision for Leak Detection in the Automotive Sector". In: *Pomiary Automatyka Robotyka* 25.3 (Sept. 2021), pp. 79–85. [Link]

4. D. Krata, M. Ochman, M. Panek, M. Skoczen, K. Spyra, Z. Kulas, D. Sroczynski, and A. Pawlowski. "Adaptive Smith Predictor Control Scheme for a Nonlinear Hydraulic System". In: *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sept. 2021. [Link]

5. Marcin Ochman. "Hybrid approach to road detection in front of the vehicle". In: *IFAC-PapersOnLine* 52.8 (2019), pp. 245–250. [Link]

The knowledge gained over the years, which has been utilized in the thesis and some scientific publications, is the result of work on research grants funded by the National Centre for Research and Development The author participated in three grants listed below:

1. **Connected Worker** / *POIR.01.01.01-00-1229/20*
2. **Autonomous Mower** *POIR.01.01.01-00-1069/18*
3. **Leak Detector** / *POIR.01.01.01-00-0773/17*

Finally, one notable contribution of the author's research during his PhD studies is successful filing and granting of a Polish patent:

1. Wojciech Macherzyśnki, Marcin Ochman, Mateusz Didyk, Krzysztof Dudek, Zbigniew Kulas, and Dawid Sroczyński. *Method of leak detection, especially in closed-volume systems*. Pat.242047. Oct. 2022

Chapter 2

Visual Simultaneous Localization and Mapping Overview

Overview

Simultaneous Localization and Mapping (SLAM) and its variant i.e. Visual SLAM (VSLAM) are the main subjects of the thesis. Therefore, it is crucial to fully understand considered problems, terminology, challenges, as well as methods developed by scientists over decades.

This chapter is an introduction to VSLAM. It contains mathematical formulation of the problem and defines basic terms used in this dissertation. One may also find detailed architecture description of representative VSLAM algorithm. Despite of the belief that VSLAM problem is solved, it was possible to list at least several issues which requires more research to develop better techniques in terms of efficiency, accuracy and robustness. Furthermore, fast-growing industry making use of computer vision finds novel applications in new environments leading to higher requirements and bigger challenges which are still growing. Hence, a possible future of Visual SLAM algorithms is taken into consideration. The chapter provides also detailed and thorough literature review.

To understand a science, it is necessary to know its history.
– Auguste Comte

Humans visiting new indoor and outdoor places such as rooms, corridors, streets or squares can still know where they are. They can immediately collect information about unfamiliar area, make reasoning about surroundings and their own position relative to the objects nearby. In other words, localizing and mapping is a natural, constantly working process occurring in their minds.

It is so common, that probably most do not recognize it as a crucial part of living. However, it is very challenging and intriguing when it comes to implement mentioned functional component in autonomous systems such as mobile robots or autonomous cars. One of their key feature is ability to continuously localize in its environment. That is why Visual Simultaneous Localization and Mapping (VSLAM) has become an important research topic in robotics [Cad+16].

Despite the frequent usage of VSLAM in that particular domain [Bar+22; You+17], it should be noted that not only robotics applies techniques developed by VSLAM community. There are many areas where SLAM based on computer vision finds its usage. For instance, Augmented Reality (AR) and Virtual Reality (VR) also take advantage of VSLAM algorithms [Jin+19; TC20; The+22]. That is why this thesis do not focus only on robotics. It is believed that VSLAM may find much broader application including AR, VR and Mixed Reality (MR) collectively named as eXtended Reality (XR) [BCL15], 3D reconstruction and mapping [LCL19], autonomous navigation [Aua+10] and every system requiring automated perception of environment such system operates in [Li+22b].

As a consequence, all definitions and terms used in this document will not refer exclusively to autonomous systems or mobile robots. Instead, terminology is selected carefully to be applicable across a wide range of disciplines. That is why device conducting SLAM will be refereed to as *mobile platform*.

As stated before, localization and mapping is a natural process for every person. Nevertheless, it is crucial to understand and be able to define these fundamental terms. SLAM answers two basic questions about current state, destination and path i.e. *"Where am I?"* and *"What is my environment?"* [Bou21]. Mobile platform scans its environment related to its position. *Localization* is the task of position estimation while environment is known. On the contrary, collecting and building map based on the position is called *mapping*. In real world, neither map or localization are known. Position and map have to be estimated based on the data which comes from the same sensors. The problem of simultaneous estimation of both is called *Simultaneous Localization and Mapping (SLAM)*. In other words, the trajectory of platform and position of landmarks are estimated online without a priori knowledge [DRN96]. In general, localization and mapping seem to be dependent processes. To successfully complete localization there must be a map which algorithm may relate to. On the other hand, to build a reliable map accurate position is required. In literature, one may find that these two operations are often compared to the chicken and egg problem [KJS16]. Although, solution of chicken and egg problem may be considered questionable, it is proven that simultaneous localization and mapping is solvable and converges to the correct state [HD16].

The following section presents brief historical overview of SLAM and formulates general problem referring to classical age of SLAM dated to 1986-2004 [Cad+16].

2.1 Fundamentals of SLAM problem. Probabilistic approach

SLAM's origin reaches back to late 1986 when the IEEE Robotics and Automation Conference took place in San Francisco, California. In the second half of the decade researchers agreed that fully autonomous systems placed in the unknown environment is required to have algorithms for consistent map estimation and localization. It was beginning of probabilistic methods introduced into robotics and Artificial Intelligence (AI) [DB06].

Theoretical work by Smith [SC86] and Durrant-Whyte [Dur88] were key papers establishing basic relationships between landmarks and geometric uncertaintities. The main contribution of these papers was proof of increasing correlation between landmarks as a result of collecting more observations. Four years later Smith published article [SSC90] which showed that all landmarks of the map are correlated to each other due to correlation between landmark and platform state. In other words, to estimate platform pose large state vector would be required leading to high computational complexity. Concurrently, researchers were working on navigation methods using various sensors including vision or sonar. The term *Simultaneous Localization and Mapping*, as well as SLAM's architecture was introduced in 1996 by Durrant-Whyte in survey paper [DRN96]. In literature, SLAM was also known as Concurrent Mapping and Localization (CML) [TBF05].

According to [Aul+08], by 2008 probabilistic methods have become principal in SLAM community. At that time SLAM was mainly applicable by roboticists which were trying to model environment and platfom pose with uncertaintities and sensors affected by noise. From author's perspective, a comprehensive understanding of the classical era of SLAM, including both the problem formulation and historical background, is essential for conducting more advanced research referring to Visual SLAM. Before probabilistic SLAM is defined, basic terms need to be introduced which will be referenced throughout the remainder of the thesis.

While *localization* deduces the location based on specific *map* which is a set of given landmarks, *mapping* is a procedure of collecting and placing landmarks on the map. Landmark term has been already used several times in this chapter and it appears to have an intuitive meaning. By definition, *landmark* is a typical, representative and motionless element of the scene. It will be shown in Chapter 3 that two adjectives i.e. *representative* and *motionless* play a key role in terms of SLAM robustness.

Figure 2.1 demonstrates SLAM problem. Mobile platform is placed into an unknown environment containing L landmarks $l_j, j \in \{1, 2, ..., L\}$. Platform's state at time $n \in \mathbb{Z}^+$ is represented by $\mathbf{x}_n$. The detailed discussion of platform and landmarks state representations will be covered in Section 2.5.2. During environment exploration in consecutive time points $\{n, n+1, ...\}$, platform observes landmarks. Due to noises of the sensors which platform is equipped with and other issues discussed in Chapter 3, measurement $\mathbf{z}_{n,j}$ of the observed landmark's state l_j at time n differs.

Depending on type of SLAM, the goal of the algorithm may be formulated twofold. Both definitions are similar and the distinction between them is subtle. Researchers distinguish Online SLAM and Full SLAM. The prior is responsible for estimating only current state $\mathbf{x}_n$ along with the map. On the contrary, Full SLAM estimates all poses of the platform's trajectory [TBF05].

Let $\mathcal{X}$ be an unknown variable being estimated. Typically, it contains platform's trajectory as well as landmarks poses. In that case, Full SLAM is involved. Online SLAM is performed when $\mathcal{X}$ includes only current pose and set of landmarks, Set $Z = \{\mathbf{z}_{n,j} : n \in \mathbb{Z}^+, j \in \{1, \ldots, L\}\}$ is a set of all observations gathered during scene exploration. Next, let $\mathbf{u}_n$ be an action at time n. The system state is altered by the *action* [Mor15]. Finally, Full SLAM may be defined as a problem of calculating posterior (Equation 2.1).

$$p(\mathcal{X}_{\text{online}} | Z, \mathbf{u}_{1:n}) = p(\mathbf{x}_{1:n}, l_{1:L} | Z, \mathbf{u}_{1:n}) \tag{2.1}$$

The online version is a simplified form of Full SLAM and may be expressed by Equation 2.2.

$$p(\mathcal{X}_{\text{full}} | Z, \mathbf{u}_{1:n}) = p(x_n, l_{1:L} | Z, \mathbf{u}_{1:n}) \tag{2.2}$$

Using Bayes formula, it is possible to construct SLAM problem recursively. It is worth noting that such form requires two models to be defined i.e. state transition model and observation model. The former is described by the probability of observation z_n given platform state and landmarks position:

$$p(\mathbf{z}_n | \mathbf{x}_n, l_{1:L}) \tag{2.3}$$

The latter is a probability of next state $\mathbf{x}_n$ given previous state $\mathbf{x}_{n-1}$ and occured action $\mathbf{u}_n$:

$$p(\mathbf{x}_n | \mathbf{x}_{n-1}, \mathbf{u}_n) \tag{2.4}$$

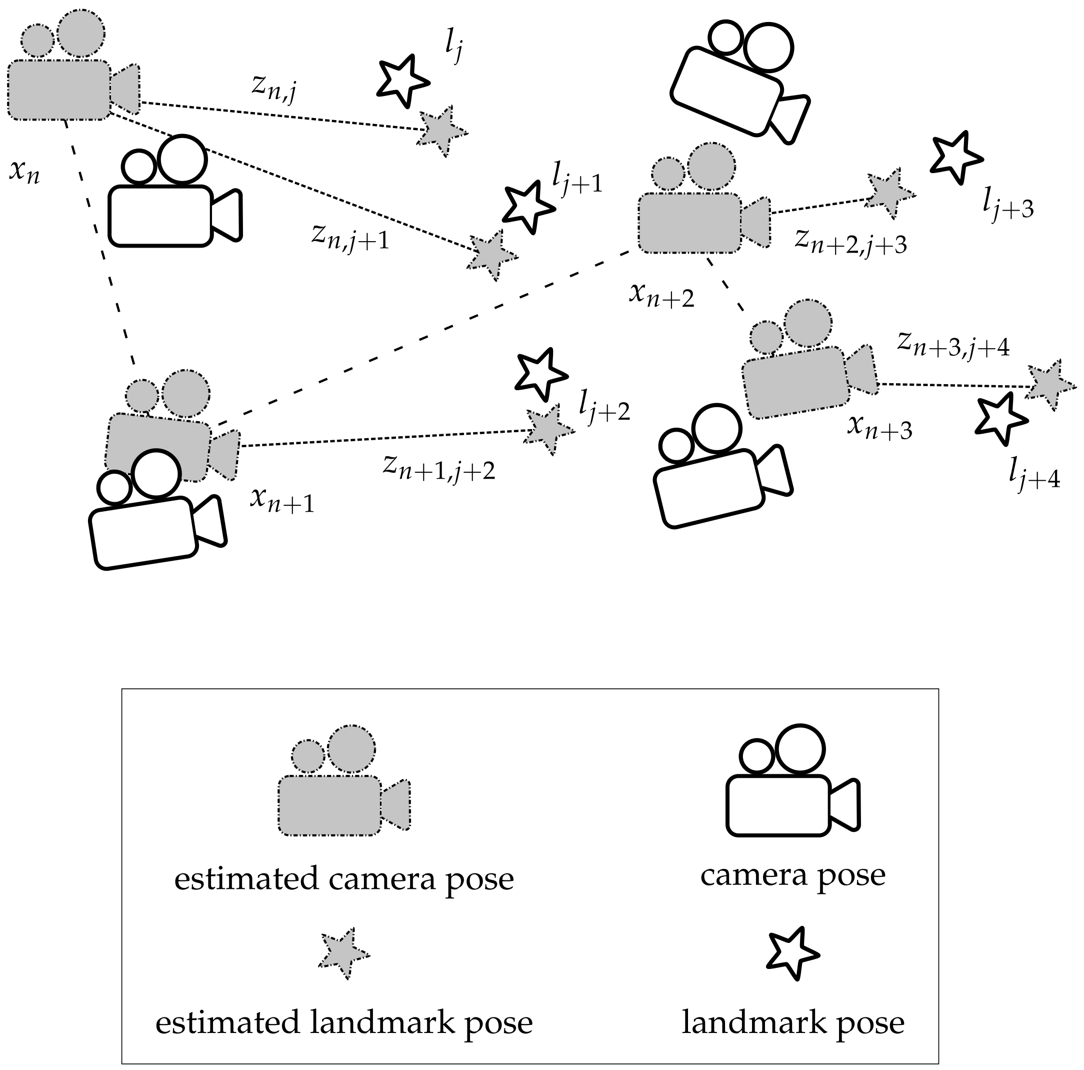


FIGURE 2.1: Simultaneous Localization and Mapping [Kaz+22]. Mobile platform (camera in the example), moves around the scene and observes landmarks $l_j, j \in 1, \ldots, L$ at specified moments of time $n, n+1, \ldots, n+3$. Observations are affected by sensor's noise and other issues discussed in Chapter 3, as a result, collected observations $\{\mathbf{z}_{n,j}\}, i \in 1, \ldots, L$ are not aligned with real values.

Determining an appropriate representation for the observation model (Equation 2.3) and motion model (Equation 2.4) is crucial for an efficient, iterative solution to the probabilistic SLAM problem.

Theoretical research regarding SLAM has also been made over the years. Convergence of the SLAM algorithm is an important topic. A survey article states [Dis+11] that under additional assumptions including observability and stationary features, uncertaintities related to features state decreases over time. However, there are still open problems regarding convergence, for instance, what time is required to build a map of the environment with given accuracy or is quality of the map sufficient.

The probabilistic SLAM's introduction was only the beginning of SLAM's wide research description. This formulation was related to core of the problem which is an estimation of the map and platform's state. By convention, this part is called *backend*. The backend will be discussed in detail in Section 2.2.3 and Section 2.6.3. Moreover, backend uses abstracted data acquired by other component called *frontend*. The frontend module is described in Section 2.6.2.

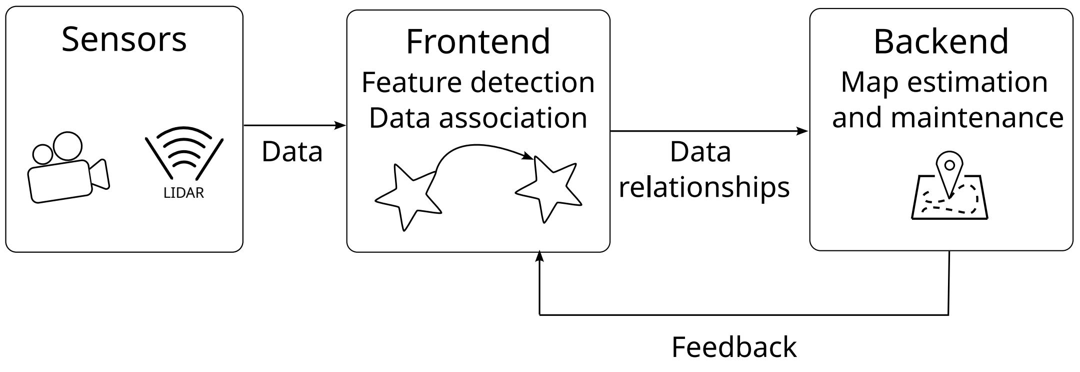


FIGURE 2.2: Typical SLAM architecture. It is composed of a data acquisition module, tasked with gathering and forwarding data to the frontend. Subsequently, the frontend identifies data correlations and conveys them to the backend, which is accountable for map estimation. The architecture is further enhanced by a feedback loop that relays any modifications from the backend to the frontend [Cad+16].

Many methods and improvements have been proposed for almost 40 years of research. The next two sections (Section 2.2 and Section 2.3) will give a brief description about SLAM categorization and related work focusing on innovative solutions. Section 2.5 is fully dedicated to VSLAM.

2.2 SLAM categorization

SLAM techniques can be categorized based on several key components including type of sensors used by a platform, frontend type, backend type and platform's target environment. The frontend is mainly responsible for observations determination and motion estimation and for that purpose can use various methodologies classified as feature-based, direct or semi-direct. The backend focuses on map estimation with loop closure and can use filter-based or graph-based approaches. Finally, operating environment plays important role for SLAM system. Depending on whether environment is arranged in predictable manner it can be called structured or unstructured.

All these categories determine the wide range of SLAM systems, adapted to various scenarios and operational requirements. The next section describes each categorization scheme in detail.

2.2.1 Sensors type

Every SLAM pipeline begins with data acquisition block, which is responsible for gathering and combining data from all sensors placed on the mobile platform [SK18]. These sensors, functioning as the perceptual eyes and ears of the platform, play a cardinal role in determining the effectiveness and reliability of the SLAM system. It is well know fact that every sensor has its own strengths and weaknesses . As a result, there is no sensor which can fulfill all requirements [Cen+20].

A *sensor* is a device which detects or measures certain inputs from the physical environment. Depending on type of sensors such qualities may be distance to object, light, speed, acceleration etc. They can be divided into two categories proprioceptive sensors and exteroceptive sensors [Mor15]. *Proprioceptive sensors* provide information for platform's internal state i.e. wheel encoders, accelerometers, and gyroscopes. Their main drawback is being prone to cumulative errors over time. To compensate errors it is required to equip the platform with *exteroceptive sensors*. They help to perceive and interpret the external environment, and capture data that allows the SLAM system to identify and localize landmarks, detect obstacles, and consequently, build a comprehensive map of the environment.

In the subsequent subsections, a comprehensive exploration and analysis of various sensor modalities used in SLAM including the most popular i.e. LiDARs, RGB cameras, RGB-D cameras, and acoustic sensors will be presented [Kha+22]. Each sensor's unique characteristics, capabilities, and the role they play in the intricate process of environment mapping and robot navigation will be discussed.

Ultrasonic sensor

Ultrasonic sensor emits high-frequence waves and detects their reflection. By measuring time of they offer distance measurements. These sensors provide advantages such as cost-effectiveness, simplicity in integration, and reliable performance in detecting objects and measuring distances within a defined range.

On the other hand, they generally offer lower spatial resolution, which can affect the accuracy and detail of the captured data. Additionally, they may struggle with complex environments containing multiple reflective surfaces, objects with irregular shapes. According to [Zaf+18], due to lack of medium, they cannot operate in vacuum.

Sonar

Sonar sensors provide valuable distance measurements by emitting sound waves and detecting their echoes as they bounce off objects in the environment. These sensors offer distinct advantages, including underwater navigation capabilities, reliable object detection, and suitability for applications in environments with low visibility. They are commonly used in uderwater environments.

Due to many similarities to ultrasonic sensors, sonars has analogous drawbacks. They tend to have lower spatial resolution compared to other sensors, which can affect the accuracy and detail of the captured data [Apa+22]. Additionally, sonar signals can experience interference and noise in cluttered environments, leading to potential inaccuracies in distance measurements. Moreover, the range of sonar sensors is generally limited, which can constrain their applicability in certain large-scale mapping scenarios.

LIDAR

A Light Detection and Ranging (*LiDAR*) sensor provides three-dimensional information about the environment by emitting laser beams and measuring the time taken for the light to return after reflecting objects. An example of LiDAR sensor is shown in Figure 2.3c. The primary advantages of LiDAR include high-resolution depth information, ability to function well in various lighting conditions, and capability to accurately measure distances over a relatively large range.

However, LiDAR sensors also present certain limitations. Firstly, they are generally more expensive and consume more power than cameras, which can be a significant consideration in cost and resource-sensitive applications. They also typically produce a large volume of data, demanding higher computational resources for processing. In addition, while LiDAR sensors provide excellent depth and distance information, they do not capture color or texture details like cameras do, which can limit their utility in certain applications.

Camera

A *monocular camera* is a single-lens camera that captures images from one viewpoint. It is equivalent of a human eye. The basic parameters of a monocular camera are focal length and resolution. The focal length determines the camera's zoom level and have influence on the area the camera can capture in a single frame. The resolution affects the detail level of the captured image.

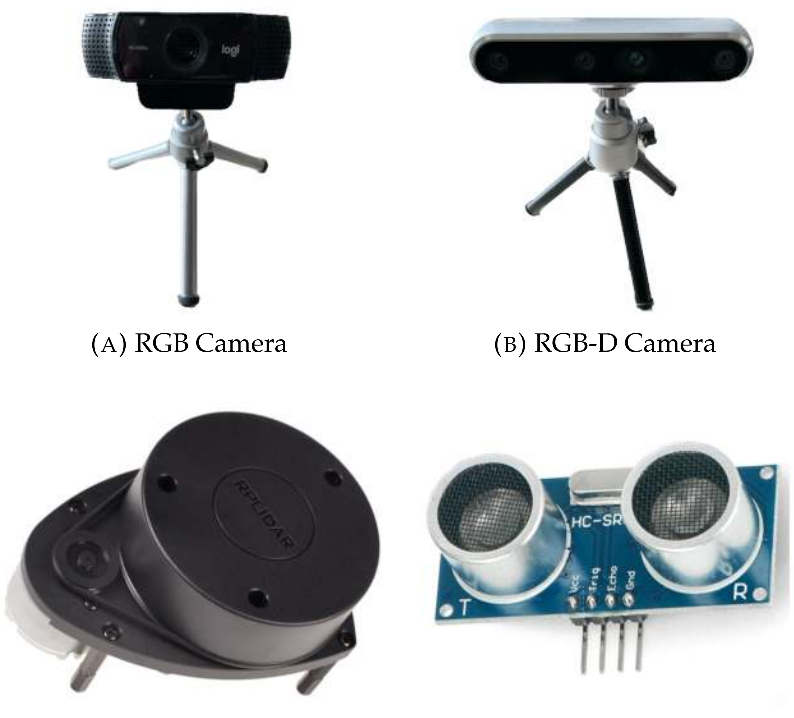
In SLAM systems, monocular cameras offer crucial visual insights into the environment. Their principal benefits include lightweight design, ease of integration, and a favorable balance between the cost and the information they provide.

Nevertheless, the use of monocular cameras is not without its limitations. A significant drawback is the scale ambiguity, which presents a challenge when deriving actual distances or dimensions from 2D images, due to the absence of depth information. This shortcoming can result in inaccuracies in the map building or localization processes. Moreover, creating an accurate map with monocular SLAM often necessitates more observations. Additional complications can arise in low-light scenarios or with moving objects, as these conditions may lead to motion blur and resultant errors. Lastly, the increase in data volume necessitates greater computing performance. In general, higher resolutions significantly influence the power consumption and costs associated with processing units. Figure 2.3a shows an example of RGB camera.

RGB-D camera

RGB-D camera is an extension of RGB one by combining color and depth data. An example of an RGB-D camera is depicted in Figure 2.3b. Its primary advantages include the provision of rich 3D information, enhanced object recognition, and relatively straightforward integration into various platforms. Given its ability to provide both color and depth data, it can overcome the scale ambiguity problem which has been mentioned discussing monocular cameras, leading to more accurate mapping and localization.

However, like all sensors, RGB-D cameras also present certain limitations. These cameras generally have a limited range and field of view, which can impact the completeness and accuracy of the captured data. They may struggle in outdoor environments or bright light due to interference with the infrared projector used for depth estimation [Cho+15]. Furthermore, they are usually more expensive and consume more power than monocular cameras, which can be a factor to consider when designing mobile or resource-limited platforms.



(A) RGB Camera (B) RGB-D Camera

(C) LIDAR (D) Ultrasonic sensor

FIGURE 2.3: Commonly used sensors in SLAM system.

Inertial measurement unit

An *Inertial Measurement Unit* (IMU) provides information about the motion and orientation of a mobile platform. It is proprioceptive sensors. As a consequence, it has to be used with other devices to be able to build a map. IMUs play a crucial role in complementing other sensors like cameras or LIDAR, especially when navigating in challenging conditions, such as underground or indoor environments. IMUs also offer valuable information for sensor fusion algorithms, allowing for more accurate and reliable localization and mapping.

IMUs also have certain limitations that must be considered. Over time, their measurements can drift due to integration errors, leading to positional inaccuracies in long-term SLAM operations. Moreover, IMUs are sensitive to external disturbances and vibrations, which can introduce noise and affect the quality of their measurements. To mitigate these limitations, IMUs are often used in combination with other sensors to compensate provided errors [Leu+14].

Sensor fusion

Based on the previous paragraphs, it can be concluded that each sensor has its strengths and weaknesses. Therefore, ideas have emerged to equip the platform with not just one, but multiple types of sensors. As a consequence, it is very common to encounter platforms that utilize two ore more various kinds of sensors. *Sensor fusion* is a process of combining data from various sensors to provide coherent information [KJD18]. For instance, to overcome the limitation of an RGB camera, which does not provide information about the distance from objects, additional sensors such as LIDAR are used leading to Visual-LIDAR approaches by combining two advantages of each sensor i.e. rich visual information with depth measurements [DV20]. Another example that frequently appears in scientific literature is the utilization of an RGB camera and an IMU sensor [Che+18]. Visual-Inertial (VI) approaches use Intertial Measurement Unit (IMU) to provide real scale [Qua+19]. By fusing data from different sensors, the system can compensate for individual sensor limitations and enhance the overall perception and understanding of the surroundings.

The utilization of multiple sensors introduces additional requirements for the data acquisition component. One significant challenge arises from the varying update frequencies of the sensors. For instance, while IMU sensor measurements arrive at a rate of 200Hz, the camera's framerate is usually limited to 30 or 60Hz. Consequently, the synchronization of data between these sensors plays a crucial role in ensuring accurate and coherent information. Similarly, in the case of Visual-LiDAR SLAM, there are additional challenges to address. Apart from the data synchronization aspect, the fusion of RGB images and LiDAR point clouds requires the association of data from these distinct sensor sources. This data association task is a complex task due to the diverse coordinate systems and characteristics of the sensors involved.

In general, choosing right data association algorithm is not an easy task. There are many options including nearest neighbor algorithm, probabilist approaches, Kalman filter, machine learning and many more [WWN20]. The following articles and books are recommended to deepen understanding of multisensor fusion, which is a critical concept in sensor data integration and the improvement of perception systems. A comprehensive overview of the field's fundamentals can be found in [LHL08] by Liggins, Hall, and Llinas. Practical applications in the automotive industry are explored in [GS22]. The fusion of data provided by many sensors is also discussed in [SK16]. It explores the challenges and advancements in sensor fusion algorithms, making it a valuable reference not only in the field of robotics. For those interested in advanced and state-of-the-art techniques, both articles [TLZ23] and [Men+20] explore the intersection of machine learning and data fusion.

Sensors comparison and choice

There are many possibilities when it comes to chose sensors for SLAM platform. Numerous sensors, which can be combined together give so many options. Thus, one can ask a simple question *"What sensors should be chosen?"*. While question seems to be not difficult, the answer is usually much more complicated.

The requirements of the SLAM application play a key role in determining the optimal sensor choice. When selecting the most suitable sensors for a SLAM system, several crucial factors must be taken into consideration [Che+18].

In Table 2.1 the previously described sensors have been compared with respect to various aspects that need to be taken into consideration when selecting suitable sensors for the SLAM system. Factors such as the desired level of accuracy, the complexity of the environment, real-time performance, and cost constraints must be carefully considered. By weighing these requirements against the strengths and weaknesses of each sensor, a well-informed decision can be made regarding the most suitable sensor or sensor fusion approach for the given SLAM application. Furthermore, numerous factors are intricately linked. To give an example, higher levels of detail require increased processing power and better hardware to handle larger data volumes, potentially leading to a higher energy demand. Moreover, a sensor's cost constitutes just one component of the overall SLAM system expenses. The price of the processing unit, and consequently the total cost, is also related to its computing capabilities i.e. faster CPUs cost more.

In general, choosing the best sensors for SLAM involves a thorough assessment of factors such as level of detail, system's update time, integration, and power consumption. Balancing these considerations and selecting sensors that align with the specific requirements of the SLAM application can significantly impact the success and its effectiveness.

The PhD thesis focuses on solutions that utilize cameras for a reason. The author believes that visual approaches are the best choice due to several compelling advantages they offer Visual SLAM leverages the wealth of information present in images, providing rich visual cues and detailed 2D or 3D representations of the environment [LWG18]. Cameras are widely available, cost-effective, and easily integrated into various platforms, making them accessible for a broad range of applications. The visual data captured by cameras allows for more accurate and robust feature extraction, which is crucial for reliable mapping and localization in complex and dynamic environments [Yu+18]. Additionally, the large amount of visual data available allows for the application of advanced computer vision and machine learning techniques, enhancing the system's perception and decision-making capabilities. Moreover, visual approaches offer the potential for more intuitive and human-like interaction with the environment. Cameras provide a natural and familiar way to perceive the world, which can facilitate human-robot interaction [Li+19] and collaborative tasks [ZN20]. VSLAM also benefits from the ability to perform tasks such as object recognition, tracking, and scene understanding [Fav23], which can further enhance the overall functionality and versatility of the system [Ros+21a]. By leveraging the rich visual information, VSLAM systems can adapt and learn from their surroundings, making them more adaptable and suitable for a wide range of real-world scenarios. Moreover, the generated map may include color information, enhancing its usability for human interpretation and utilization. This feature allows people to reuse the maps for various purposes and applications.

Overall, the author's belief in the superiority of visual approaches is rooted in their ability to harness the power of visual perception and leverage it to overcome challenges in SLAM, making them a promising choice for future advancements in the field.

2.2.2 Operating environment

Choosing sensors for SLAM mobile platform requires well specified operating environment. With the domain of SLAM algorithms, the operating environment plays a crucial role in shaping the efficacy and performance of these systems. Level of environment's complexity has big impact on possible assumptions which SLAM system can make.

Depending on capability of making simplifying assumptions operating environments may be classified into two categories i.e. *structured* and *unstructured* ones. The former refers to settings where there is a well-defined and recognizable layout, such as indoor spaces including offices, factories, or corridors. In structured environments, the presence of distinctive features like lines [ZZZ19] being part of walls, corridors and doors or known landmarks provide references for SLAM algorithms. Figure 2.4a shows an example of structured environments. The image depicts a well-structured environment featuring a road for cars. Clear and recognizable traffic signs, distinct road lanes, and evenly spaced traffic bollards contribute to its classification as a structured setting. These elements, spaced at consistent intervals, underline the organized layout of the environment, demonstrating the presence of discernible features

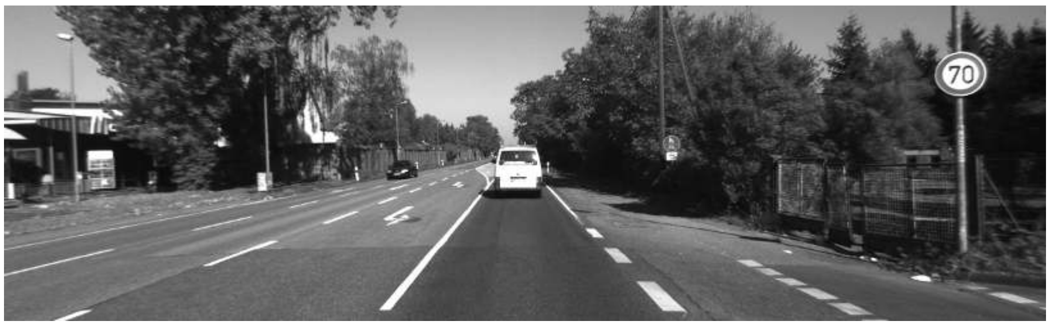
Conversely, the latter comprises settings that lack well-defined layouts or recognizable features, often encountered in outdoor landscapes, rugged terrains, or cluttered urban scenes. Figure 2.4b illustrates structured environment. The image captures a rugged mountainous landscape, exemplifying an unstructured environment characterized by its irregular topography, lack of discernible patterns, and absence of recognizable landmarks. The varied terrain and absence of predefined references underscore the complexities in

Sensor	***Environment***	***Depth Range m***	***Price €***	***Power W***
LIDAR	Indoor, Outdoor	0.1 - 185	500-5000	5-200
Sonar	Underwater	0.1 - 6000	500 - 5000	0.01-5
Ultrasonic	Indoor	0.1-10	2-500	0.1-1.5
Monocular Camera	Indoor, Outdoor	N/A	100-5000	0.01-10
RGB-D	Indoor, Outdoor	0.1-10	100-500	0.3-5

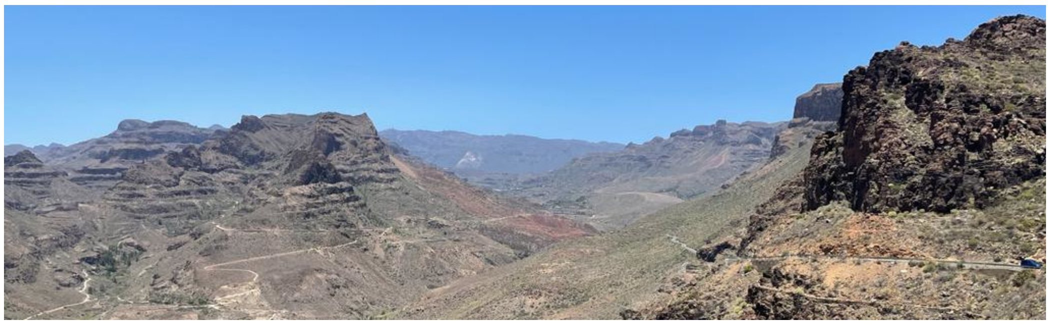
Sensor	***Update frequency Hz***	***Data bandwidth Mbps***	***Compute requirements***	***Notes***
LIDAR	10-30 Hz	1-100	●●	No color information
Sonar	0.1 - 10	1-10	●●	No color information; Range depends on wave frequency
Ultrasonic	1-50Hz	$\sim 10^{-6}$	●	No color information, Many sensors required
Monocular Camera	30-60 Hz	20-300	●●●	Scale ambiguity
RGB-D	30-60Hz	30-500	●●●	Color and 3D information

TABLE 2.1: Comparison of commonly used exteroceptive sensors in SLAM [Kaz+22; Li+22a; Cen+20].

accurately mapping and navigating through such intricate landscapes. According to [Gui+04] mapping large unstructured environments is the most challenging task. It arises from the fact that this is general problem. Being capable of mapping unstructured environments implies ability to map structured environments, although the reverse relationship does not hold. That is why researchers have been trying to develop methods which can handle any kind of environment for decades [Bai02; Gui+04].



(A) Structured environment [GLU12]



(B) Unstructured environment

FIGURE 2.4: Examples of structured and unstructured environments.

Another classification commonly encountered in scientific literature is the differentiation between indoor and outdoor [Tan+16; Kaz+22]. In many cases indoor spaces such as factories, offices and warehouses have well-defined structure including walls, doors, windows and corners. Figure 2.5a shows an example of indoor space. As a consequence, they are often identified with structured environments. Nevertheless, it is not difficult to give an example of structured environment which is not indoor i.e. well marked urban streets composed of road lanes, traffic lights and road signs.

On the contrary, there are outdoor environments like open roads, forests, or natural terrains which introduces another difficulties for SLAM systems. Figure 2.5b shows an example of city park which is an outdoor space. Changing weather conditions, varying lighting, and occlusions from natural elements like trees or terrain irregularities are only a few examples of challenges that can impact data quality and algorithm performance. Outdoor environments often feature a dynamic scene with vehicles, pedestrians, and wildlife constantly in motion which if are not handled properly may cause tracking errors and inconsistency in created map. To address all these challenges, outdoor SLAM systems may incorporate advanced techniques like loop closure detection, simultaneous multi-sensor fusion, and deep learning scene understanding.

Despite the fact that underwater environment falls into the category of an outdoor setting, it's worth distinguishing it as a separate type due to its significant differences from the terrestrial outdoor environment.

Unlike terrestrial outdoor settings, underwater environments lack clear visual cues and easily identifiable landmarks. The turbidity of water, fluctuating currents, and limited visibility hinder the effectiveness of traditional methods. Underwater SLAM systems must contend with the 3D nature of water bodies, including vertical movement and potential changes in water depth. The irregular topography of the seabed, coupled with underwater vegetation and marine life, contributes to the absence of consistent visual references, making it difficult to implement traditional feature-based SLAM approaches. Furthermore, the corrosive nature of water poses challenges to the durability of sensors and equipment. Finally, the presence of water significantly influences the accuracy of sensor measurements. Variations in the speed of wave propagation underwater and shifts in focal length can introduce considerable distortions to the data collected by sensors [Wan+23; Zha+22; Jia+19; Zha+19a].

(A) Indoor (B) Outdoor (C) Underwater [Zha+22]

FIGURE 2.5: Examples of indoor, outdoor and underwater environment.

2.2.3 Backend type

Section 2.2.1 and Section 2.2.2 mostly applied to the frontend part of a SLAM system. The frontend serves as a component that generates abstractions from sensor data, while the backend utilizes these abstractions to make inferences based on the abstracted information. Initially, it was mainly responsible for map estimation. In other words, backend processes the data collected from sensors and transformed by frontend, and refines the platforms's pose and the map over time. Based on the backend algorithm employed to accomplish the described task, it can be categorized into three distinct groups. First two types of backend are filter-based and optimization-based. Furthermore, due to recent advances in machine learning and scene reconstruction [SS23], a third category referred to as deep learning approaches has emerged. All three kinds of backend are discribed in next sections (Section 2.2.3 – Section 2.2.3).

According to Chen et al., contribution to the different kinds of backends may be visualised by timeline presented in Figure 2.6. Although, clear separation can be seen between three periods, it is worth noting that authors claim that this division does not have an absolute endpoint. Additionally, an attentive reader will notice that the Figure 2.6 is closely related to the timeline presented in Figure 2.11 regarding SLAM Related work.

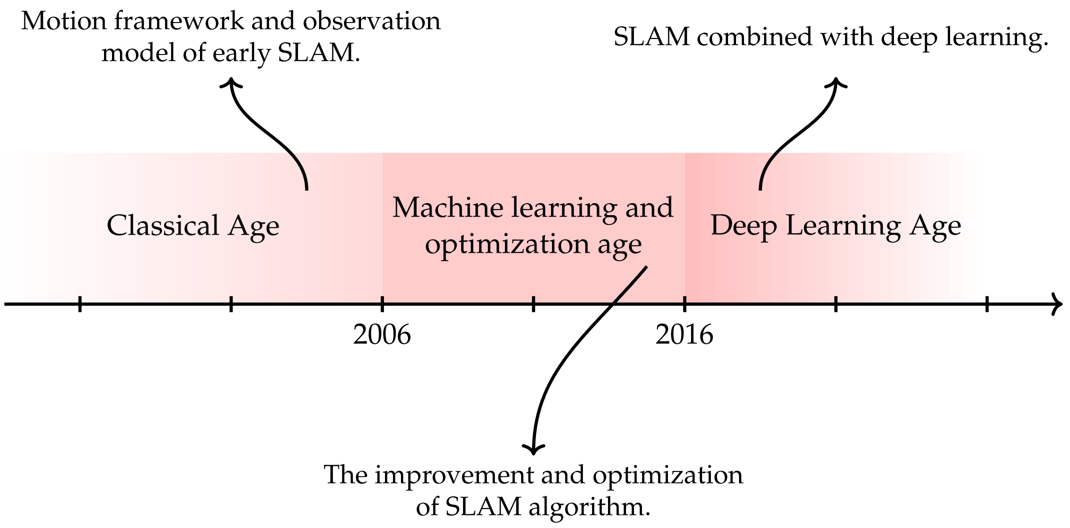

Motion framework and observation model of early SLAM.
SLAM combined with deep learning.
Classical Age
Machine learning and optimization age
Deep Learning Age
2006
2016
The improvement and optimization of SLAM algorithm.

FIGURE 2.6: Timeline depicting the evolution of research on different backend types [Che+22b].

Filter-based backend

Filter-based SLAM approaches formulates the SLAM as a state-estimation problem. In this context, state means both platform pose and map. The state undergoes continual refinement through a filtering procedure. Relying on actions and measurements, it makes ongoing estimations about state. As additional data becomes available, the estimation process continues, progressively enhancing and perfecting the state representation [Pal+22].

Let recall probabilistic SLAM formulation, specifically motion (in literature also referred to as transition) and observation model given by Equation 2.4 and Equation 2.3 respectively. With assumptions of landmarks' time-invariance and the platform's Markov motion model, SLAM can be recursively given by Equation 2.5 [WTT03], where m represents a map.

$$\begin{aligned} & p(\mathbf{x}_n, \mathbf{m} | \mathbf{z}_n, \mathbf{u}_n) = \\ & = \eta \underbrace{p(\mathbf{z}_n | \mathbf{x}_n, \mathbf{m})}_{\text{Observation model}} \int \underbrace{p(\mathbf{x}_n | \mathbf{x}_{n-1}, \mathbf{u}_n)}_{\text{Motion model}} \underbrace{p(\mathbf{x}_{n-1}, \mathbf{m} | \mathbf{z}_{n-1}, \mathbf{u}_{n-1})}_{\text{Previous posterior}} d\mathbf{x}_{n-1} \end{aligned} \tag{2.5}$$

From the Equation 2.5 defining the SLAM process recursively as a Bayesian filter, three algorithmic steps emerge [Kuz18]. Before diving into details of the algorithm, let introduce transition and observation model. The transition model given by Equation 2.6 predicts how the system state evolves over time based on control inputs and incorporates process noise. The observation model (Equation 2.7) relates the true state to sensor measurements and considers measurement noise. In the following equations, $\mathbf{X}_n$ is a state of SLAM systems combined of platform pose and a map.

$$\mathbf{X}_n = f(\mathbf{X}_{n-1}, \mathbf{u}_n) + \mathbf{w}_k \tag{2.6}$$

$$\mathbf{z}_{n,j} = h(\mathbf{y}_j, \mathbf{X}_n) + \mathbf{v}_{n,j} \tag{2.7}$$

These models are central to Bayesian filtering techniques, enabling the filter to estimate and update the state of a dynamic system using sensor data and dynamics models while accounting for uncertainty. The following list describes mentioned steps of filter's single iteration, while Figure 2.7 presents them in graphical form:

1. **Prediction** – in this step, the filter uses a mathematical model to predict how the state of the system i.e. platform's pose is expected to evolve from its current estimate. This is accomplished by propagating the state estimate forward in time using the system's motion model and control commands. The platform's current pose is predicted by the motion model given by Equation 2.6. Essentially, it predicts where the system will likely be in the next time step based on its current state and the applied controls.
2. **Observation** – the filter compares the predicted state, generated in the previous prediction step, with actual sensor measurements obtained from the environment. These measurements provide valuable information about the real-world state of the system, helping to correct any

discrepancies between the predicted and observed values. This step is expressed with the Equation 2.7.

3. **Update** – the final step of the filter is to update platform's pose and landmarks positions with formula given by Equation 2.5.

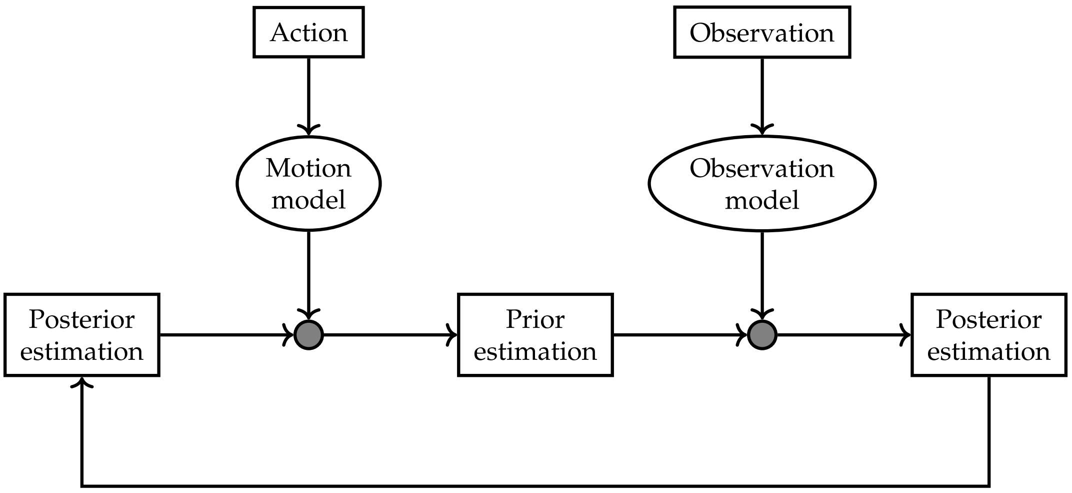


FIGURE 2.7: Single iteration of Bayesian filter based on recursive formula given by Equation 2.5.

Following an overview of Bayesian filtering, the subsequent sections will provide detailed descriptions of three distinct Bayesian filters, each offering a unique approach to state estimation. These filters, namely the Kalman Filter, the Extended Kalman Filter, and the Particle Filter has their strengths and limitations.

Kalman filter One of the most popular implementations of the Bayesian filter is technique proposed by Kalman in [Kal60], referred to in the literature as the Kalman filter. It relies on a set of assumptions that define its scope and functionality. The primary assumption is the linearity of the system dynamics. In other words, both transition model and observation model are linear. It means that function $f(\mathbf{X}_{n-1}, \mathbf{u}_n)$ from Equation 2.6 takes a linear form $f(\mathbf{X}_{n-1}, \mathbf{u}_n) = \mathbf{A}\mathbf{X}_{n-1} + \mathbf{B}\mathbf{u}_n$. Matrix A establishes the relationship between the state at time-step $n-1$ and the subsequent time-step excluding noise $\mathbf{w}_n$ and control $\mathbf{u}_n$. On the other hand matrix $\mathbf{B}$, links the action $\mathbf{u}_n$ with the present state. As a result, transition model is given by Equation 2.8.

$$\mathbf{X}_n = \mathbf{A}\mathbf{X}_{n-1} + \mathbf{B}\mathbf{u}_n + \mathbf{w}_n \tag{2.8}$$

Similarly, the function $h(\mathbf{y}_j, \mathbf{X}_n)$ becomes $h(\mathbf{y}_j, \mathbf{X}_n) = \mathbf{C}\mathbf{x}_{n-1}$ where matrix $\mathbf{C}$ links state $\mathbf{X}_{n-1}$ with observation $\mathbf{z}_n$. There is also additive noise $\mathbf{v}_n$. Equation 2.9 represents the observation model used in Kalman filter.

$$\mathbf{z}_{n,j} = \mathbf{C}\mathbf{X}_{n-1} + \mathbf{v}_{n,j} \tag{2.9}$$

Secondly, all uncertainties, including the state estimate and measurement noise, follow Gaussian distributions. In particular, $\mathbf{w}_n \sim \mathcal{N}(0, \sigma_w)$ and $\mathbf{v}_n \sim \mathcal{N}(0, \sigma_v)$ represent zero-mean Gaussian noise with standard deviations σ_w and σ_v for the transition and observation models, respectively.

By this assumptions, Kalman filter ensures Gaussianity of the estimates. It means that based on the posterior estimation at time $n-1$, which are parameters of Gaussian distribution i.e. mean $\boldsymbol{\mu}_{n-1}$ and covariance $\boldsymbol{\Sigma}_{n-1}$, the prior estimation $\bar{\boldsymbol{\mu}}_n, \bar{\boldsymbol{\Sigma}}_n$ can be calculated with Equation 2.10. With regard to the three mentioned steps of the Bayesian filter, this corresponds to the first step, which is prediction.

$$\begin{aligned} \bar{\boldsymbol{\mu}}_n &= \mathbf{A}\boldsymbol{\mu}_{n-1} + \mathbf{B}\mathbf{u}_n \\ \bar{\boldsymbol{\Sigma}}_n &= \mathbf{A}\Sigma_{n-1}\mathbf{A}^T + \sigma_v \end{aligned} \tag{2.10}$$

Then, observation takes place and a posterior estimation μ_n, Σ_n is determined with formula given by Equation 2.11, where $\mathbf{y}_n = \mathbf{z}_n - \mathbf{C}_n\bar{\boldsymbol{\mu}}_n$ is called innovation and $\mathbf{K}_n = \bar{\boldsymbol{\Sigma}}_n\mathbf{C}_n^T(\mathbf{C}_n\boldsymbol{\Sigma}_n\mathbf{C}_n^{\bar{T}} + \mathbf{Q})^{-1}$ is the Kalman gain. The posterior estimation refers to the third step of Bayesian filter.

$$\begin{aligned} \boldsymbol{\mu}_n &= \bar{\boldsymbol{\mu}}_n + \mathbf{K}_n\mathbf{y}_n \\ \boldsymbol{\Sigma}_n &= \mathbf{A}\boldsymbol{\mu}_{n-1} + \mathbf{B}\mathbf{u}_n \end{aligned} \tag{2.11}$$

Extended Kalman Filter Kalman filter is known for its simplicity, optimality and robustness [JU97]. However, one of its significant drawbacks is reliance on system linearity. Consequently, it cannot be applied to many real-world systems that are characterized by nonlinear functions $f(\cdot)$ and $g(\cdot)$ in their transition and observation models introduced in Equation 2.6 and Equation 2.7. Moreover, Gaussianity is not preserved in nonlinear systems. To address these issues, the Extended Kalman Filter (EKF) has been introduced, which involves the linearization of models using Taylor expansion. Similarly to the standard Kalman filter, first, a prediction is carried out, expressed by Equation 2.12.

$$\begin{aligned} \bar{\boldsymbol{\mu}}_n &= g(\mathbf{u}_n, \boldsymbol{\mu}_{n-1}) \\ \bar{\boldsymbol{\Sigma}}_n &= \mathbf{J}_n\boldsymbol{\Sigma}_{n-1}\mathbf{J}_n^T + \sigma_v \end{aligned} \tag{2.12}$$

$\mathbf{J}_n$ is Jacobian matrix of the motion model in a form given by Equation 2.13.

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_n^1} & \cdots & \frac{\partial f_1}{\partial x_n^S} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_P}{\partial x_n^1} & \cdots & \frac{\partial f_P}{\partial x_n^S} \end{bmatrix} \tag{2.13}$$

Subsequently, the system updates its state based on new observations, following Equation 2.14, where $\mathbf{K}_n = \mathbf{\Sigma}_n \mathbf{H}_n^T (\mathbf{H}\bar{\mathbf{\Sigma}}\mathbf{H}_n^T + \sigma_w)^{-1}$ is Kalman gain and the innovation is defined as $\mathbf{y}_n = \mathbf{z}_n - g(\bar{\boldsymbol{\mu}}_n)$. $\mathbf{H}_n$ is a Jacobian of observation model. For a more comprehensive understanding of this topic, mathematical derivations and further insights into the Extended Kalman Filter, interested readers are encouraged to refer to the works of Newman and Welch and Bishop [WB94; New99].

$$\begin{aligned} \boldsymbol{\mu}_n &= \bar{\boldsymbol{\mu}}_n + \mathbf{K}_n \mathbf{y}_n \\ \mathbf{\Sigma}_n &= (\mathbf{I} - \mathbf{K}_n \mathbf{H}_n)\bar{\mathbf{\Sigma}}_n \end{aligned} \tag{2.14}$$

After introducing Kalman filter and its variant for nonlinear systems, let's discuss their advantages, limitations and drawbacks. As it was mentioned earlier, Kalman filter is a parametric filter. Therefore, it is easily interpretable and does not require huge memory. Another advantage lies in its simplicity. Thus, it is computational efficient. In comparison to more complex nonlinear filters, the EKF is relatively low, making it suitable for real-time processing.

On the contrary, one of the major disadvantages of the filter is strong reliance on known and accurately specified models. Any mismatches between these models and the true system behavior can introduce bias and compromise estimation accuracy. Kalman filter assumes an absence of modeling errors. In practice, differences between mathematical models and the actual system can lead to estimation inaccuracies. While it handles moderately nonlinear systems, it may fail to converge in the face of highly nonlinear or non-Gaussian systems. The next concern related to SLAM is the quadratic cost associated with map size [Dis+01]. This can lead to significant computational overhead, especially in applications with large maps.

Despite the acknowledged constraints and the scientific community's emphasis on optimization-based approaches, the Extended Kalman Filter (EKF) is still employed in SLAM. An example of this can be found in [Ull+20].

Particle Filter Due to assumptions that the system's state and measurement noise follow Gaussian distributions fully characterized by their means and variances, KF and EKF are examples of parametric filters. In contrast to parametric filters there are also non-parametric ones, which do not make specific assumptions about the probability distribution and can represent a wide range of distributions using a set of N_s particles $\{\mathbf{X}_n^i, i = 0, \ldots, N_s\}$. As a result, particle filter solves one of the main disadvantages of KF and EKF which is requirement for Gaussianity.

A particle refers to a sample $\mathbf{X}_n^i$ which is a possible representation of the state sequence. Each particle has corresponding weight w_n^i. The weights are normalized i.e. $\sum_i w_n^i = 1$.

The fundamental idea behind particle filters in SLAM is to estimate the platforms's pose and map by propagating particles $\{\mathbf{X}_n^i, i = 0, \ldots, N_s\}$ through time. The higher weight w_n^i is, the closer to the true state is particular sample. Each particle represents potential trajectory of the platform and the corresponding map. The algorithm iteratively updates the particles based on sensor measurements and control inputs. Consequently, a discrete weighted approximation to the true posterior $p(\mathbf{X_n}|\mathbf{z}_n, \mathbf{u}_n)$ is obtained [Aru+02]. It is given by Equation 2.15, where $\delta(\cdot)$ is Dirac delta function.

$$p(\mathbf{X}_n|\mathbf{z}_n, \mathbf{u}_n) \approx \sum_{i=1}^{N_s} w_n^i \delta(\mathbf{X}_n - \mathbf{X}_n^i) \tag{2.15}$$

Particle filter, as an example of a Bayesian filter, retains its recursive structure. An additional step is resampling.

1. **Prediction** – each particle is moved forward in time using transition model given by Equation 2.6. This step simulates how the system's state evolves.

2. **Update** – particles are weighted based on their likelihood of producing the observed sensor measurements $p(\mathbf{z}_n|\mathbf{X}_n^i)$. Particles that better match the measurements receive higher weights.

3. **Resampling** – is a process of creating a new set of particles focusing on regions of high probability. It replicates particles based on their weights, with particles associated with higher weights being more likely to be duplicated. This process ensures that particles representing more probable states are given greater influence in the estimation, enhancing the filter's performance and preventing from the degeneracy [LBD15].

Particle filters have limitations that can be grouped into three key aspects. To begin, maintaining an adequate number of particles in the set is crucial. Small sample size can introduce bias into posterior estimations, potentially causing the filter to lose track of the true state, especially in high-dimensional spaces [DFG01]. On the other hand, big size of particle set has significant impact on filter's performance which can lead to high computational requirements. This is a trade-off between filter's accuracy and execution time. Secondly, the resampling procedure has to be chosen wisely. It depends on the specific requirements of the application, computational resources available, and the nature of the probability distribution being estimated. It often involves a trade-off between maintaining diversity and controlling computational costs while aiming for accurate state estimation. Too frequent resampling may also introduce variance [ETM21]. Finally, due to their nonparametric nature, particle filters demand more computational resources compared to KF or EKF. The latter recursively obtain next estimates with simple formulas(Equation 2.11 and Equation 2.14) for the state of the system, while the former needs update state of every particle. This might be a notable issue especially for large state spaces.

Optimization-based backend

In the previous sections, backend filtering approach was discussed that involve the maintenance of probability distribution functions. It employs a Bayesian Network (BN) to represent SLAM, which is a graphical model depicting stochastic processes in the form of a graph [Gri+10]. In a BN, each node corresponds to a random variable, and directed edges between nodes capture their dependencies [Ji19].

Figure 2.8a shows an example of SLAM problem modeled by BN. In this scenario, there are four poses denoted as $x_1, \ldots, x_4$, each corresponding to the platform's position at a specific time step n. At $n = 1$, the platform observes three landmarks, namely l_1, l_2, l_2, and these observations are modeled by measurements z_1, z_2, z_3. Subsequently, at $n = 2$, the platform detects two landmarks, l_2 and l_3, using measurements z_4 and z_5. Moving on to $n = 3$, three landmarks are observed via measurements z_6, z_7, z_8, while at $n = 4$, there is a single landmark, l_5, with observation z_9. The motion and observation models (Equation 2.4 and Equation 2.4) are represented by particular set of edges. The former is represented by edge connecting current pose x_n with previous pose x_{n-1}. The latter is described by two edges i.e. the current pose of platform x_n and map, which is represented by static landmark x_k.

In Figure 2.8b process of Bayesian filter inference is shown. In filtering approaches, the joint probability distribution typically involves the current state of the platform and the map. However, not all of these variables are required. For instance, past measurements and past poses are marginalized to keep graph relatively small. On the other hand, graph may become fully inter-connected. As a consequence, the computational cost of joint distribution propagation is highly correlated with number of variables and its performance especially for big maps may be poor. To preserve time constraints one have to limit map size. This is significant disadvantage of filtering approaches.

At the beginning of SLAM research, mainly because of noise, the problem was characterized in terms of probabilistic language. However, question arises whether other approaches exist. Indeed, the answer to this question is positive. Lu and Milios were first scientists who proposed a method which used spatial relations as a constraints to refine a map [LM97]. Therefore, an alternative to BN is to model SLAM as a graph which is constructed out of measurements. Every graph node represent a platform's pose or landmark, while edges are constraints.

Figure 2.8c is an example of optimization-based approach. There are several differences comparing to the filter-based methods. First, there are no node's related to measurements z_i. In graph-based approach measurements are represented as graph's edge. For instance, in Figure 2.8c, there are three measurements z_1, z_2, z_3 related to x_1 platform's pose and three landmarks l_1, l_2, l_3. In Figure 2.8c, measurements nodes are represented by edges from x_1 to l_1, l_2, l_3. Furthermore, graph contains less platform's pose nodes. It should be emphasized that number of nodes in graph should be maintained carefully. It means that adding all poses, landmarks and constraints would result in highly inefficient solution. To overcome that problem only specific poses and their corresponding set of sensor's measurements can be added to the graph. In comparison to filtering methods, only small subset of all poses are retained, while other poses are not marginalized but explicitly discarded. As a consequence, two poses in the described example, namely x_2 and x_4, were removed from the graph due to their insignificance.

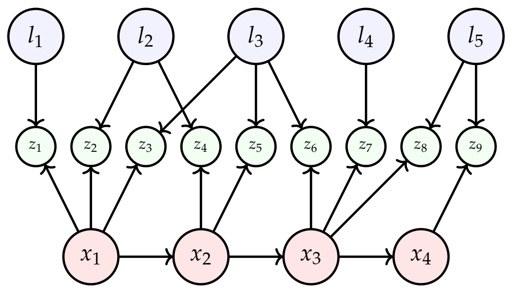
In such scenario, the SLAM backend can be structured as a two-step process. Initially, there is the construction of a graph. This step involves incorporating new nodes and edges into the graph, reflecting the spatial constraints of the scene and the platform's trajectory, relying on the past measurements. Subsequently, optimization is performed to refine the map and all poses, with the goal of maximizing function $f(\cdot)$ based on the collected measurements (Equation 2.16).

$$x^* = \operatorname{argmax} f(x, l, z) \tag{2.16}$$

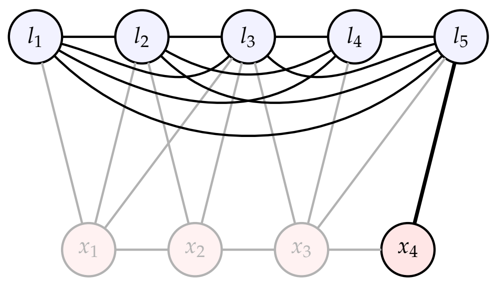
The question which may arise is what function $f(\cdot)$ is? In the research papers, typically optimization-based SLAM backend is formulated as a maximum a posteriori estimation (MAP). The idea behind MAP is straightforward. The $\mathcal{X}^*$ variable is seeked which maximizes posterior probability $p(\mathcal{X}|Z)$, where $p(Z|\mathcal{X})$ is the probability of measurements Z given $\mathcal{X}$, and $p(\mathcal{X})$ is a prior probability of $\mathcal{X}$. MAP may be formulated by Equation 2.17. It is also noteworthy that in linear Gaussian case, Kalman filtering and MAP give the same estimates.

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}}\, p(\mathcal{X}|Z) \tag{2.17}$$

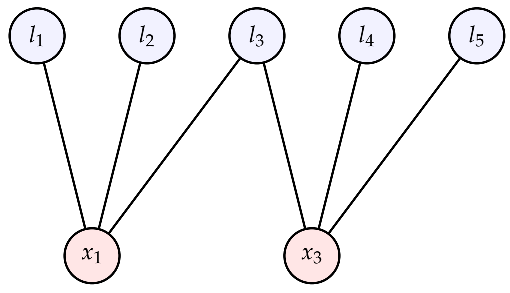
By assumption of independent measurements, MAP may be factorized into form shown in Equation 2.18.


l_1 l_2 l_3 l_4 l_5
z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8 z_9
x_1 x_2 x_3 x_4

(A) Dynamic Bayesian Network



(B) Inference in Bayesian filter



(C) Keyframe optimization-based approach

FIGURE 2.8: Comparison of Dynamic Bayesian Network with keyframe optimization-based approach [SMD12; Gri+10].

$$\mathcal{X}^* = \underset{\mathcal{X}}{\operatorname{argmax}}\ p(\mathcal{X}) \prod_{k=1}^{M} p(z_k|\mathcal{X}_k) \tag{2.18}$$

Optimization-based SLAM or alternatively, Graph SLAM utilizes factor graphs and factorization to represent complex models efficiently. A *factor graph* is a graphical representation of process which consist of variables and factors. The variables correspond to unknown quantities within the problem, while the factors correspond to functions that operate on specific subsets of these variables. The edges within the factor graph link factors with variables, signifying that a given factor relies on specific variables for its computations [KF09; Loe04]. An example of SLAM represented by a factor graph is presented in Figure 2.9. There are four poses represented by $x_1, \ldots, x_4$. Odometry constraints are modeled by factors $u_1, \ldots u_3$. For every odometry constraint there are two edges connecting consecutive poses. Furthermore, there are second type of nodes representing landmarks. In the example there are two landmarks l_1, l_2.

By assuming a Gaussian noise distribution, MAP estimation can be further reformulated as a least squares problem, as represented in Equation Equation 2.19. For a comprehensive understanding of the transformations from MAP to the least squares problem, additional insights can be gained from the following references: [Gri+10; Cad+16; Gao+17]. To clarify, h_k is a

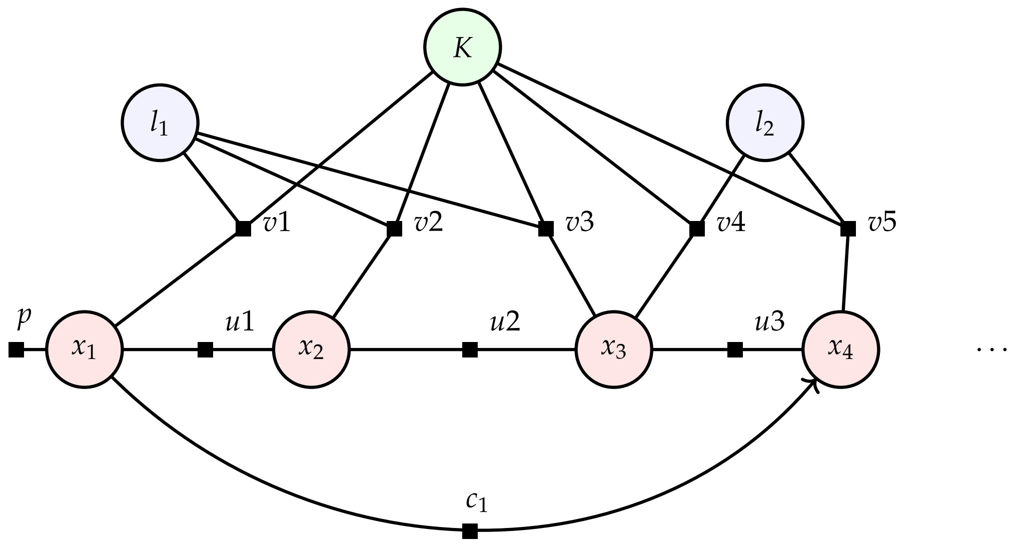


FIGURE 2.9: Example of factor graph of SLAM.

nonlinear function representing the observation model (Equation 2.7), and Ω_k is the information matrix, which is the inverse of a covariance matrix.

$$\operatorname{argmin} \sum_{k=0}^{M} (h_k(\mathcal{X}_k) - z_k)^T \Omega_k (h_k(\mathcal{X}_k) - z_k) \tag{2.19}$$

Those with experience in computer vision may notice that Equation 2.19 bears a resemblance to the bundle adjustment formulation. For a more detailed discussion and comparison between VSLAM and bundle adjustment, which represents a special case of the structure from motion problem, please refer to Section 2.4. However, it is worth mentioning two significant differences at this point. In bundle adjustment (BA), only cameras are employed, whereas SLAM has the flexibility to utilize various types of sensors. Additionally, SLAM operates as an incremental process, in contrast to BA, where estimation takes place after all the data has been gathered [Cad+16].

Recent research has shown that optimization-based approaches are better in terms of scalability, accuracy and performance. Strasdat, Montiel, and Davison in their work [SMD12] suggest that in most cases optimization-based approaches are better than filtering methods. Further scientific papers seem to support that statements. One may observe that in the 2010s, optimization-based methods gained the attention of researchers, leading to the majority of developed algorithms utilizing optimization [You+17].

2.2.4 Map representation

In order to develop SLAM system it is important to chose proper map representation. The generation and maintenance of maps are fundamental aspects in every mapping process. Maps not only allow proper environment reconstruction but, as will be explored further, also may help in navigation and understanding platform's surroundings.

In the preceding sections that focused on SLAM categorization, the attention was primarily directed towards SLAM as an estimation problem. Aspects related to both the frontend and backend of SLAM were discussed, inherently involving the mapping process. However, it is important to acknowledge that certain assumptions were made regarding the map representation. So far, the map was a set of landmarks which were represented as a point in R^3. While this definition of the map was convenient for mathematical formulation of the SLAM problem, one should be aware of other options which may be taken into considerations. There are at least several kinds of map. However, they can be grouped into two categories i.e. metric maps and topological maps.

Each of them will be described in the following paragraphs. The choice of map type depends on the specific requirements of the SLAM application, the available sensor data, and the desired level of accuracy and detail in representing the environment. Different SLAM algorithms may utilize different types of maps to perform tasks like localization, mapping, and path planning effectively.

Metric maps are types of maps which represent the environment in metrical manner. It means that they allow for measurements of distances and angles. They provide a quantitative representation of the environment, allowing localization, navigation and other activities requiring spatial information. There are several kinds of metric maps including occupancy grid, voxel map, point cloud maps, semantic maps and hybrid ones.

Occupancy grid maps represent the environment as a grid of cells, each associated with a value indicating its occupancy status. This value can be binary (occupied or not) or within a range, representing the likelihood of occupancy. In a formal sense, occupancy maps model the configuration space of a platform [Lat90]. They find common use in structured environments like factories and warehouses due to their simplicity, ease of construction, and map maintenance regardless of the environment's size. However, they come with certain limitations. One key consideration is selecting an appropriate cell size, which requires a trade-off between the accuracy of environmental representation and memory constraints, as larger maps demand more memory. Additionally, most implementations use a fixed cell size, limiting their adaptability when higher levels of detail are needed. Next, occupancy grids may struggle to scale when approaching map boundaries. Finally, it's important to note that occupancy grid maps typically represent 2D environments.

Voxel map is an extension of occupancy grid map to 3D environments. Instead of cells, environment is divided into a grid of volumetric elements called voxels. This representation is essential for applications involving autonomous vehicles, 3D scanning, and mapping of complex indoor or outdoor spaces. Due to the fact that voxel maps are a generalized form of 2D occupancy maps, they share advantages and disadvantageous. Again, their representation simplicity leads to easy creation and updates. However, the problem related to the resolution trade-off becomes even more apparent. Adding an extra dimension means that reducing the voxel size by a factor of 2 while keeping the size of the mapped environment constant results in an eightfold increase in memory requirements. It is clear that this relationship is cubic. Their drawbacks also include limited dynamic adaptability and scalability issues near the map boundaries.

Point Cloud Maps store information about the environment as a collection of 3D points. They provide a detailed and accurate representation of the world by capturing the spatial information of objects. High level of detail is especially useful in application where fine-grained environmental knowledge is crucial i.e. mobile systems operating in small, cluttered spaces. Another advantage is their sensor-agnostic nature. It means that they can be generated from various sensors like depth sensors, LIDARs or RGB-D cameras. On the contrary, relatively high level of detail is associated with data volume and computational complexity. Pointclouds may contain large number of points which can be hard to process in real-time applications. Moreover, such high levels of detail may not be necessary, particularly when dealing with distant objects.

Feature maps represents environment created by detecting and describing distinctive features. These features may be objects, edges, corners or other typical elements of the environment. In such approach these maps reduce required size of the memory compared to other solutions. As a consequence of decreased data volume, faster processing may be achieved. However, they may be less suitable for applications that require detailed environment representation or where feature-rich data is fundamental. Furthermore, they rely on feature detection and description algorithms. As a consequence, representation of places where such algorithms struggle to detect such features may be not possible.

Semantic maps offer a representation of the environment that incorporates semantically meaningful details about objects and relationships [Ros+20b]. For instance, in the context of autonomous vehicles, this process involves not only constructing a traditional map of the environment using point cloud data but also concurrently executing a classification algorithm. This algorithm categorizes points within the cloud, distinguishing between elements such as humans, roads, bicycles, and more. Subsequently, this information is integrated into the map. Semantic maps enhance environmental perception and provide valuable information for decision-making processes. However, creating and maintaining accurate semantic maps can be challenging, primarily due to the need for advanced perception algorithms. Consequently, achieving the requirements of these algorithms may require better hardware.

Hybrid map combines the advantages of several other map types to create a more versatile and comprehensive representation of the environment. The key idea is to provide a map that not only represents the spatial aspects of the environment but also captures semantic information about objects and their relationships or other information which may be beneficial for specific application [Gao+24]. For instance, a map can be a combination of a semantic map to retain information about objects and an voxel map to enable navigation algorithms. This fusion of information enables mobile platforms to perform complex tasks in diverse environments. The another advantage is improved situational awareness. Despite their versatility, one of their most significant issue is the complexity of construction and maintenance. Moreover, they can become computationally intensive especially when maps are large. As a consequence, their utilization may be limited to high-performance hardware only.

Topological map is a type of map which characterizes environment based on relationships between key locations or landmarks. Unlike metric maps focusing on spatial measurements, they abstract the mapped world into a network of nodes and edges emphasizing spatial relationships and topology of the world [GO15]. Due to high-level understanding of the environment they are particularly well-suited in applications where navigation and path planning tasks play an important role. They are efficient in terms of memory and computations requirements due to their compactness, especially comparing to metric maps. Additionally, topological maps are more robust in situations where changes occur slowly. On the other hand, they may struggle when high precision is needed. Furthermore, they may have issues when landmarks change their positions. Constructing topological maps may require manual effort to define key landmarks [CN01] and connectivity, which can be impractical for large or complex areas. However, in literature, one can find many academic papers describing methods which do not use manual map preparation. Finally, the main issue of the topological maps is perceptual aliasing where two locations seem to be the same for the platform's sensors [BSA13].

Choosing the appropriate map representation is a critical decision in the development of a Simultaneous Localization and Mapping system. Several factors must be carefully considered to ensure that the chosen representation aligns with the specific requirements of the application. These factors are the level of detail required to capture the environment accurately, the computational resources available for processing and storing map data, types of sensors being employed. It is worth noting that choosing type of the map have an impact of level of abstraction as it is illustrated in Figure 2.10. To summarize, selecting the right map representation plays a pivotal role in the success and efficiency of SLAM systems, directly impacting their performance, adaptability, and scalability to meet the demands of various real-world scenarios.

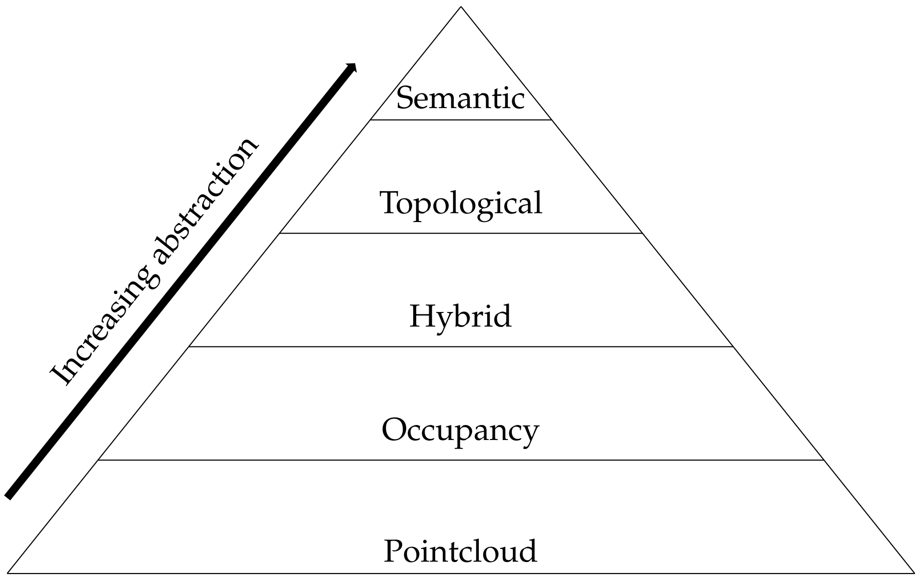


FIGURE 2.10: Relationships between level of abstraction and type of the map [BSA13].

2.3 SLAM Related work

Over 30 years of extensive research resulted in creation of many remarkable SLAM algorithms. In this section, the discussion is limited to SLAM implementations that do not incorporate camera technology. For details on Visual SLAM solutions, readers are directed to Section 2.7. The emphasis here is on exploring the most prominent SLAM algorithms, along with tools and libraries that facilitate work in this field.

At the beginning of the SLAM research most of the work was dedicated to problem formulation and solution by the EKF. There are several, modern implementations of proposed algorithms available. For instance, Sakai et al. implemented several robotics related algorithms in Python programming language for educative purposes in [Sak+18]. One of the algorithm is EKF-SLAM. Moreover, one may also find classic EKF-SLAM implementation in Matlab Navigation Toolbox [Inc22].

Clearly, researchers were aware of the limitations that EKF had. In the naive approach to SLAM, the complexity of the update increases quadratically with the number of landmarks [BD06], which became a big issue for real-time requirements. First applied technique is partitioned updates. Their main idea is to update a small, local part of the map at sensor-rate while global map is updated less often. In [GN01] Compressed EKF (CEKF) was proposed which significantly reduced computational requirements by maintaining information gained in local area and then transferring it to the global map in single iteration being equivalent of full SLAM computational cost. The other approach is to use submap or a local map. This approach is very similar to algorithms employing local map in graph-based approaches. However, the main difference between local submap in terms of EKF and local map in graph-based SLAM is that in the latter local map is used mostly for tracking and map is extended and maintained by local optimization algorithm locking positions and poses of the landmarks and keyframes which are not part of the local map.

In 2002, the FastSLAM algorithm was introduced, marking a significant advancement in reducing computational requirements for SLAM. This innovative algorithm employs both the Rao-Blackwellized particle filter for pose estimation and the Extended Kalman Filter (EKF) for mapping. The key distinction between FastSLAM and traditional EKF-based approaches lies in the dimensionality of the update equations: FastSLAM reduces it to only two dimensions, compared to $2L + 3$ in classic EKF methods, where L represents the number of landmarks. This effective separation of the platform's trajectory and landmark estimations leads to substantially lower computational complexity. FastSLAM is a foundation for further developments in SLAM including FastSLAM 2.0 [Mon+03]. FastSLAM 2.0 introduces significant improvements over FastSLAM 1.0, notably through the adoption of an efficient particle filtering technique, which enhances the accuracy of pose estimation by relying not only on the motion estimate but also on measurements. A significant contribution of this paper is a proof of convergence for linear SLAM problems when employing just a single particle. The authors have shown that comparing to the original version of FastSLAM, the next version improved accuracy by an order of magnitude.

One of the most popular and widely used SLAM algorithm is GMapping. It is designed for a range sensors including LIDAR or RADAR. Its popularity stems from being an open-source implementation and a part of the ROS ecosystem [Mac+22a]. It employs a Rao-Blackwellized particle filter to estimate the platform's pose and create an occupancy grid map simultaneously [GSB07]. One of its notable features are simplicity, real-time processing and stable, well-tested implementation. While GMapping is a powerful SLAM algorithm, it does have some limitations. It primarily works in 2D environments, making it not suitable for mapping complex 3D spaces. Additionally, it relies on range sensors, which may not capture certain environmental features effectively like shape of the objects. In the context of ROS libraries, OctoMap [Hor+13] is an impressive implementation of voxel maps. Its name is derived from its primary function of creating 3D occupancy grid maps with a focus on the octree data structure [Mea82]. This library has the ability to represent various environments without requiring additional assumptions. Unlike typical voxel maps, where the grid is simply divided into 3D voxels, OctoMap employs a tree structure to store data more efficiently. Consequently, only voxels that are occupied or unoccupied are stored, and information about unknown space is implicitly encoded in the map. In other words, voxels corresponding to the unknown space are not stored, resulting in an optimal map size. Furthermore, OctoMap offers a high degree of flexibility, including multi-resolution capabilities for adaptive level-of-detail representation. Lastly, it has ability to update a single map by multiple SLAM instances.

The last algorithm reviewed here that supports ROS is Cartographer, developed by Google, as documented in [Hes+16]. This algorithm employs a dual-structured technique, segmenting the SLAM process into local and global components. The local SLAM operates by processing incoming sensor data to create multiple local submaps, each providing a small but coherent representation of the environment. Subsequently, the global SLAM component optimizes and aligns these submaps to generate a unified global map. This bifurcated approach enables Cartographer to handle large volumes of sensor data efficiently, ensuring both high accuracy and minimal latency in map creation. Additionally, it is worth noting that two other solutions, AMCL and Marathon, are supported in the next version of ROS, namely ROS 2, as referenced in [ROS; Mac+20].

Another publicly available SLAM algorithm worth mentioning is tinySLAM, as presented by Steux and Hamzaoui in [SH10]. Contrary to the complexity commonly associated with SLAM algorithms discussed in this thesis, the creators of tinySLAM claim to have developed their system in fewer than 200 lines of C code. A key feature of tinySLAM is its simplicity. The developers aimed to design a SLAM algorithm that is straightforward, easy to understand, and delivers high performance. tinySLAM employs a laser sensor and platform odometry. However, this simplicity entails several compromises. For example, the accuracy of the map and the robustness of the algorithm are limited compared to other, more sophisticated and complex solutions. Georgia Tech Smoothing and Mapping (GTSAM) is an open-source library written in C++ programming language suited for SLAM [DC22]. It uses nonlinear optimization and graphical models i.e. factor graphs, to model and solve complex estimation problems efficiently. Its core strength lies in ability to perform smoothing over a set of measurements, allowing for more accurate state estimation than filtering methods. GTSAM is used by many other algorithms as a part of backend including StellaVSLAM [SSS19] and RTAB-Map[LM18]. There is also great tutorial available regarding GTSAM and factor graphs in [Del12]. In terms of graph optimization there is a framework known as g^2o. This open-source C++ framework is specifically designed for the optimization of nonlinear functions modeled as graphs [Kum+11]. Its architecture emphasizes modularity and extensibility, allowing users to effortlessly define their own representations of vertices and edges, corresponding to nodes and constraints, respectively. Utilizing the efficient Levenberg-Marquardt algorithm and robust data structures, g^2oachieves commendable performance in optimizing large-scale problems, even those involving hundreds or thousands of nodes and constraints. Its versatility is evident through its widespread applications in both robotics and computer vision. Additionally, g^2obenefits from an active community of users and developers who contribute to its ongoing enhancement. As of 2023, it remains under active development. Regarding optimization algorithms, it's also worth recalling the second library, known as Ceres [AMT22]. It is an open-source C++ library used for modeling and solving large nonlinear optimization problems. Its main features are easy usage, high performance and maturity. It is also used internally at Google. Similar to g^2oit is well-suited for optimization-based backend. There are many SLAM implementations where Ceres plays a major role including StellaVSLAM [SSS19] or Cartographer [Hes+16].

There is also a big part of research regarding LIDAR based SLAM algorithms. The term Lidar Odometry and Mapping (LOAM) was first introduced by Zhang and Singh in two related papers [ZS14; ZS16]. Their proposed method is well-known in SLAM community. It is able to achieve low-drift and preserve real-time performance. The main idea of the proposed approach was to divide SLAM processing into two tasks. First task is responsible for odometry while the second task performs fine matching and pointcloud registration. As can be seen in Section 2.5 modified solution is also used in VSLAM systems.

LEGO-LOAM which stands for Lightweight and Ground-Optimized Lidar Odometry and Mapping is efficient and effective in generating detailed 3D maps with reduced computational demands [SE18]. The authors claim that proposed algorithm may achieve real-time performance on embedded systems such as Nvidia Jetson. One of the key features of Lego-LOAM is its ability to separate ground points from non-ground points in the LIDAR data, allowing for more accurate and robust odometry in various terrains Proposed framework is divided into five modules i.e. segmentation, feature extraction, odometry, mapping and transform integration. While first four modules are straightforward the last one needs clarification. Transform integration module fuses results from odometry and mapping components. The improved loop closure method called SC-LEGO-LOAM was described in [KK18]. The original authors of LEGO LOAM, collaborating with additional team members, subsequently developed a method known as LIO-SAM. The proposed method not only make use of LIDAR but also IMU sensor. The integration of these two types of sensors allows LIO-SAM to effectively build accurate maps and precise localization [Sha+20]. The creators of LEGO-LOAM, in collaboration with an expanded team, subsequently introduced an advanced method called LIO-SAM. This approach leverages both LIDAR and IMU sensors, integrating their capabilities to enhance mapping accuracy and localization precision. The fusion of LIDAR's detailed environmental scanning with the rapid, motion-sensitive data from IMUs enables LIO-SAM to create highly accurate maps and maintain precise pose estimation in diverse settings. As a result, LIO-SAM represents a significant advancement in SLAM technology [Sha+20].

To illustrate the progression of the solutions discussed in relation to the time they were published, a timeline is provided, as shown in Figure 2.11. The journey to the current state of research has spanned many years, marked by gradual advancements and the emergence of new ideas that have enhanced both accuracy and performance. This timeline effectively highlights the evolutionary stages of SLAM development. Initially, the focus was on standard EKF-based solutions. This was followed by a period where particle filters gained popularity. Currently, graph-based SLAM represents the state-of-the-art in this field.

Up to this point, the most popular or breakthrough were discussed. However, SLAM community have been very active and many various ideas have arised over three decades of research. Therefore, it is equally important to focus on available surveys and overviews of the methods. In Table 2.2, a collection of papers focused on SLAM algorithms is presented. This compilation illustrates that, over the years, a significant number of articles have been published on this topic.

The two classic papers referring to probabilistic approaches are [DB06] and [BD06] written by Durrant-Whyte and Bailey. The former presents an

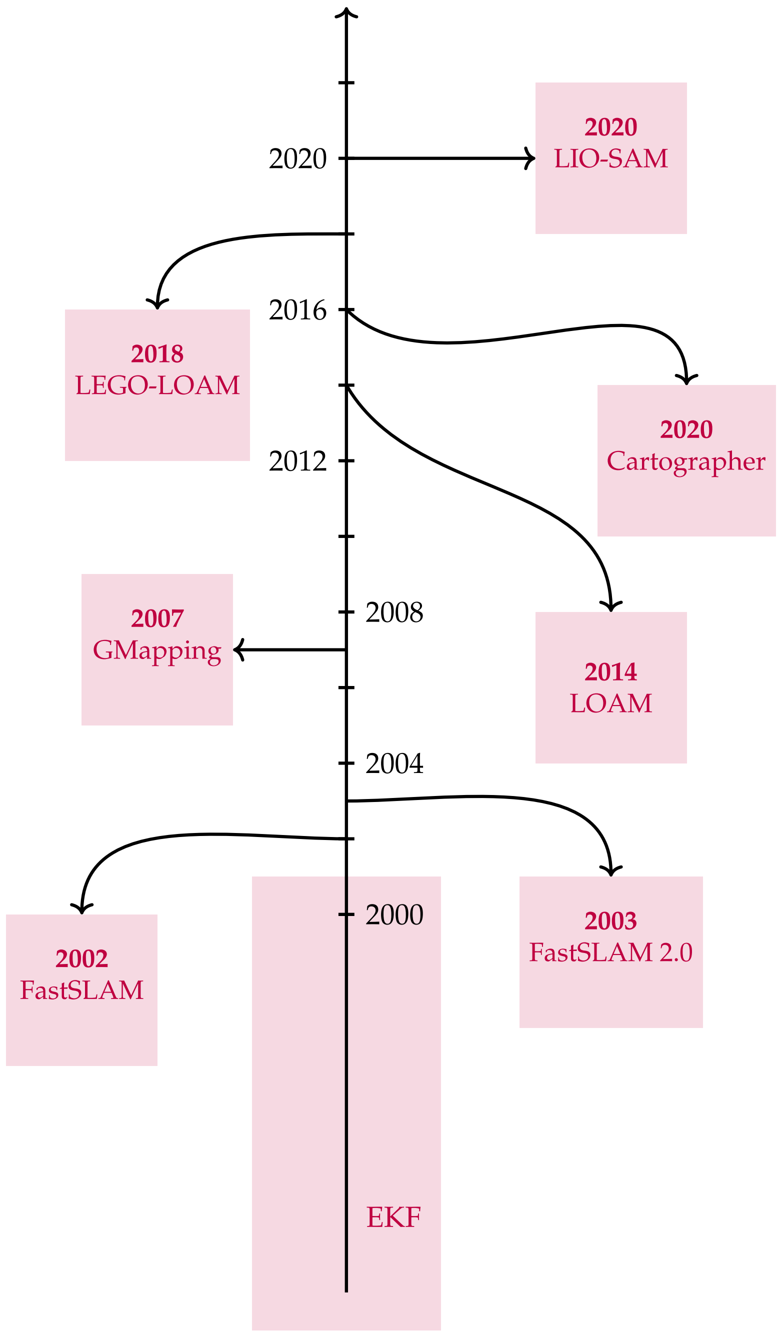


FIGURE 2.11: Timeline of SLAM.

Author(s)	***Reference***	***Year***	***Notes***
Durrant-Whyte and Bailey	[DB06]	2006	SLAM Tutorial I
Bailey and Durrant-Whyte	[BD06]	2006	SLAM Tutorial II
Aulinas et al.	[Aul+08]	2008	Methods review
Grisetti et al.	[Gri+10]	2010	Graph SLAM
Dissanayake et al.	[Dis+11]	2011	Methods review
Saeedi et al.	[Sae+15]	2015	Multi SLAM
Huang and Dissanayake	[HD16]	2016	Theoretical aspects
Takleh Omar Takleh et al.	[Tak+18]	2018	SLAM for Autonomous Vehicles
Jia, Yan, and Xu	[JYX19]	2019	Robot SLAM
Khan et al.	[Kha+21]	2021	LIDAR SLAM
Khan et al.	[Kha+22]	2022	Sensors used in SLAM

TABLE 2.2: The SLAM surveys and overview articles excluding papers dedicated to VSLAM specifically.

overview of the SLAM problem and describes fundamental implementations. The latter emphasizes three aspects of SLAM i.e. computational complexity, data association and the representation of environment. Two years later, in 2008, Aulinas et al. published another SLAM survey [Aul+08]. This paper primarily concentrates on filtering methods, encompassing EKF, CEKF, Information Filters, Particle Filters, and Expectation Maximization. Additionally, the authors address unresolved challenges, including large-scale SLAM, computational complexity, and the handling of dynamic objects in SLAM environments. They also highlight the significant potential of vision systems for future research. In contrast to filtering methods, Grisetti et al. in [Gri+10] published a comprehensive SLAM tutorial in 2010 focusing on graph-based SLAM. This article introduces readers to the graph-based SLAM methodology by defining the problem, illustrating the construction of the graph, formulating the optimization problem, and discussing the state-of-the-art solutions available at that time. The paper's primary goal is to present graph-based approaches in a manner that empowers readers to independently develop the proposed solutions from the ground up. In 2011, Dissanayake et al. provides another review of recent SLAM algorithms [Dis+11]. This concise survey primarily addresses the formulation of the SLAM problem, while also covering aspects related to observability, convergence, and consistency. Essentially, the paper focuses on outlining the fundamental properties of the SLAM problem and its associated challenges. Evidently, [HD16] emerges as one of the most distinctive articles among those mentioned so far. The article published in 2016, similarly to [Dis+11], but in a more comprehensive manner, delves into the observability of different SLAM formulations and assesses the convergence, accuracy, and consistency of various SLAM algorithms. Additionally, it highlights that, despite significant progress in solving SLAM, a complete theoretical understanding of this essential problem remains only partial.

While LIDAR in one of the most popular sensor used in SLAM [Kha+22] it is also worth mentioning paper [Kha+21] from 2021 prepared by Khan et al. The authors describe different kinds of available LIDARs and compare them to other sensor technologies. Afterwards, LIDAR SLAM is introduced and again is compared to SLAM algorithms which utilize other types of sensors.

Although it was noted at the beginning of this chapter that robotics is only one area where SLAM is applied, it is also worth mentioning articles related to mobile robots or autonomous vehicles. It is still wide source of knowledge. An interesting field of study is SLAM for multiple instances of mobile platforms. Saeedi et al. in [Sae+15] cover an introduction to the multi-SLAM, reviews existing solutions, and evaluates the pros and cons of these approaches. With the rising popularity of autonomous vehicles on public roads, a specific survey of SLAM in the context of autonomous vehicles was published. In a concise paper [Tak+18], Takleh Omar Takleh et al. explore the concept of the SLAM problem, its categorization, and its significance in modern applications. The paper particularly focuses on three types of SLAM: EKF SLAM, FastSLAM, and Graph SLAM. Similarly, with an additional section dedicated to deep learning, Jia, Yan, and Xu discusses SLAM for robots in [JYX19].

Finally, there is also survey regarding sensors used in SLAM [Kha+22]. Khan et al. conduct a detailed literature review of most common SLAM sensors like acoustic sensor, RADAR, camera, LIDAR and RGB-D camera. One may also find comparison of SLAM performance depending on utilized sensors using analytical hierarchy process and indicators including accuracy, range, cost and computational requirements.

While the last 30 years have seen remarkable advancements in SLAM research, as detailed in this section, the field still presents open challenges and opportunities for innovation, as noted by [Pit+11]. In the 2010s, a notable shift occurred in SLAM research, with many experts beginning to integrate cameras into their systems. This was driven by the potential of cameras to yield more accurate maps and enhance pose estimation. Transitioning from these developments, the subsequent sections focus specifically on vision-based solutions, delving into the realms of Visual Odometry and Visual SLAM.

2.4 Visual Odometry, Visual SLAM and Structure from Motion

The first significant paper in the field of Visual SLAM is the article by Davison in 2003 [Dav03]. This paper introduced a pioneering real-time SLAM system using a single camera, marking a groundbreaking development at the time. It signified a major departure from the traditional reliance on range sensors and established the groundwork for future advancements in Visual SLAM technology. A year later, the term 'Visual Odometry' (VO) was coined by Nister, Naroditsky, and Bergen in [NNB]. According to Scaramuzza and Fraundorfer VO involves estimating the 3-D motion of a camera based on a sequence of images capturing its environment [SF11]. There is a clear connection between VO and VSLAM systems. In this section, the primary differences between VO and VSLAM are outlined. Additionally, due to the notable similarities between VSLAM and the structure from motion (SfM) problem, a comparative analysis of these two areas is also provided.

At the beginning of the SLAM evolution, most solutions used range-based sensors. Cameras were not popular, due to several reasons including computational requirements and difficulty in finding correspondences. On the other hand, in well-known PhD thesis [Mor80], 23 years before VO term was introduced, researchers noticed that computer vision may be also worth regarding motion estimation of a mobile platform. Alongside the advancements in VO, there was parallel development of algorithms capable of reconstructing a 3D environment using images captured from a variety of cameras. These algorithms are known as structure from Motion (SfM). It seems that with the development of VSLAM, the two previously mentioned, independent fields of computer vision have become interconnected.

First, let formulate the VO problem. As it has been already stated, the VO is a process for estimating the motion of a camera in real-time by analyzing the changes in its pose over consecutive images.

Set of N cameras is moving through a scene and capturing frames at discrete time instants k. Each taken frame of the camera i is expressed by $I_{i,0:n} = \{I_{i,0}, ..., I_{i,n}\}$. Transformation of the rigid body setup of the cameras between $k-1$ and k frame is given by Equation 2.20,

$$\mathbf{T}_{k,k-1} = \begin{bmatrix} \mathbf{R}_{k,k-1} & \mathbf{t}_{k,k-1} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.20}$$

where $\mathbf{R}_{k,k-1} \in SO(3)$ is rotation matrix and $\mathbf{t}_{k,k-1} \in \mathbb{R}^3$ is the translation vector.

Camera poses $\mathbf{C}_k$ can be calculated using recursive formula given by Equation 2.21.

$$\mathbf{C}_k = \mathbf{T}_k \mathbf{C}_{k-1} \tag{2.21}$$

Nonrecursive solution of the platfom's pose may be expressed with Equation 2.22.

$$\mathbf{C}_k = \mathbf{C}_0 \prod_{l=1}^{k} \mathbf{T}_{l,l-1} \tag{2.22}$$

Equation 2.22 shows that leading task of VO is to calculate transformations of the successive frames. To find $T_{k,k-1}$ there are several necessary steps which are: detection of features, feature matching, motion estimation and local optimization. The whole process of VO was shown in figure 2.13.

The remaining step involves estimating the camera transformation $T_{k,k-1}$. This represents a typical optimization problem, where the goal is to minimize the squared error between the estimated transformation and the actual transformation across multiple keypoints. Depending on the type of data available, three different types of correspondences can be established, which guide the choice of the appropriate error function i.e. 2D to 2D, 3D to 3D, and 3D to 2D correspondences. Each type influences how the transformation is computed and optimized.

VO stands out with significant advantages in localization when compared to other techniques, as highlighted by [Aqe+16]. Notably, VO demonstrates superior accuracy, with a relative error margin of just 0.1%-2% as reported

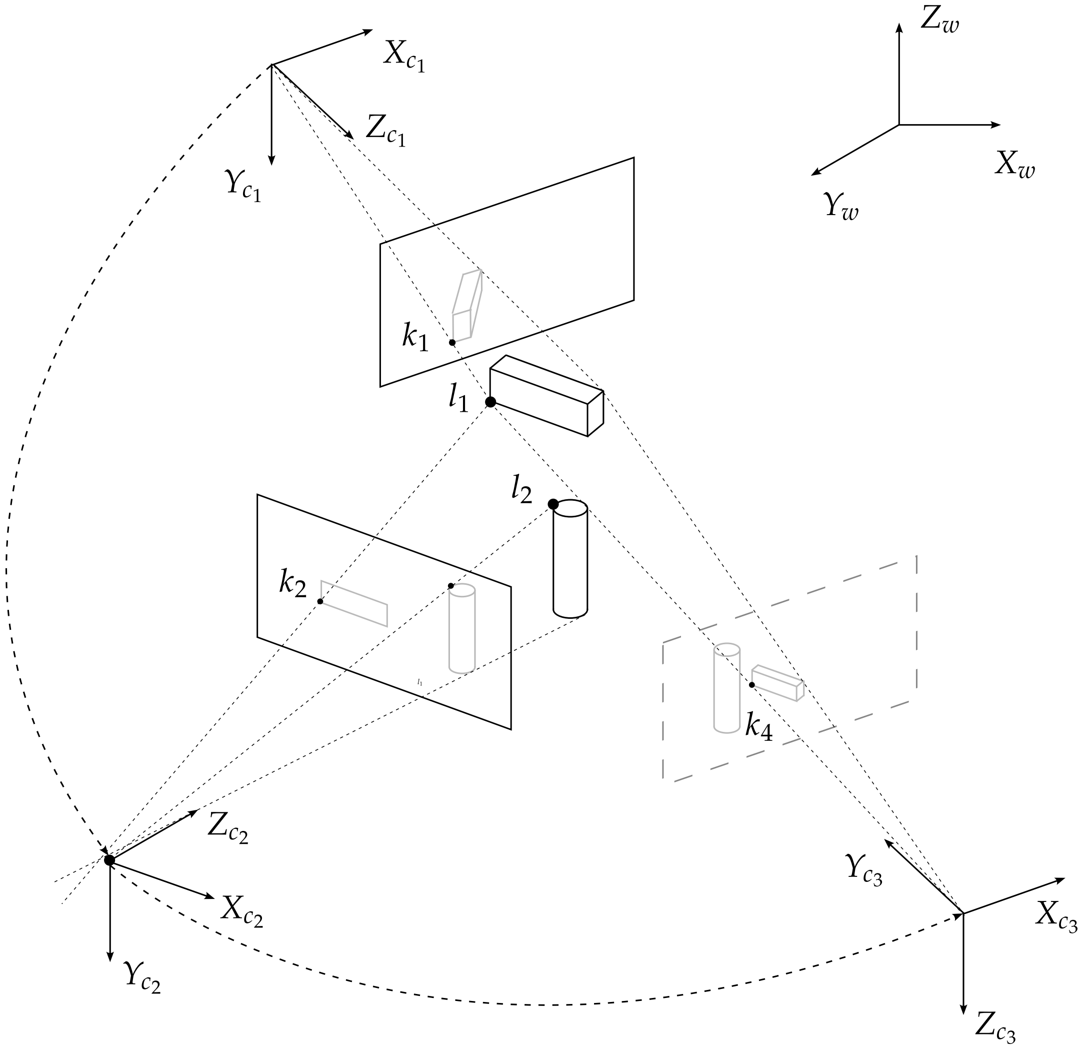


FIGURE 2.12: Visual odometry problem. Consecutive frames and transformations between them [SF11].

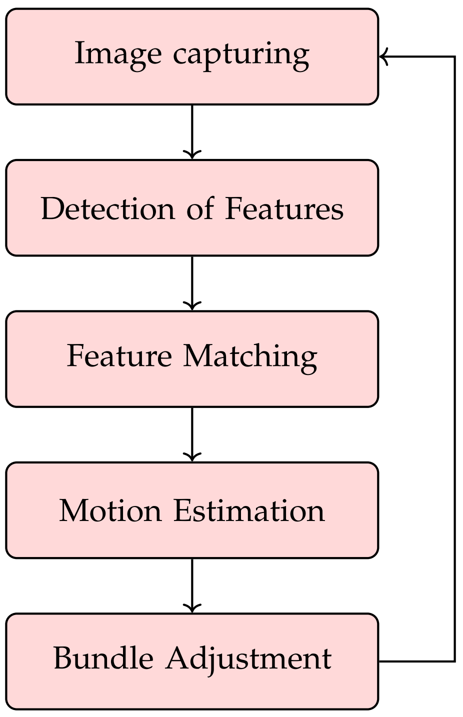


FIGURE 2.13: Flow of the basic VO algorithms [SF11].

in [SF11], surpassing other common methods. An important aspect to consider alongside its low error rates is cost-effectiveness. VO proves to be more economical than traditional techniques such as GPS or wheel odometry, particularly when accuracy is a priority. One of the key benefits of VO over GPS is its effectiveness in indoor environments, where GPS often loses signal and becomes unreliable. In contrast, wheel odometry, while useful, can accumulate significant drift over extended periods, affecting its long-term accuracy. VO, grounded in its fundamental principles, offers greater robustness against such drift. Additionally, the rich data captured in images used for VO can be leveraged for other computer vision tasks, enhancing scene understanding through obstacle or working area detection. Expanding on its advantages, VO's reliance on camera technology makes it highly adaptable for a wide range of vehicles. Its versatility, combined with the previously mentioned benefits, positions VO as a highly viable option for diverse localization applications, both in terms of functionality and cost [Moh+19].

Another problem closely related to VSLAM is Structure from Motion (SfM). As with the VO section, it is essential first to define what SfM entails. SfM is a computer vision problem that involves creating a 3D structure of a scene from a set of two-dimensional images [Özy+17]. Interestingly, its origins can be traced back to the influential work [Lon81] of Longuet-Higgins in 1981, a period close to the early developments of VO in 1980. Since then, numerous methods and improvements have been introduced in the field.

To conceptualize the SfM problem, reference is made to Figure 2.14. Similar to VO, a set of N_c cameras moves through a scene, capturing frames. The frames taken by camera i are denoted as $I_{i,0:N_i} = \{I_{i,0}, ..., I_{i,N_i}\}$, where N_i is the number of images captured by camera i. Each camera has its unique calibration parameters k_i, and each frame n taken with camera i has its specific pose $\mathbf{C}_{i,n}$. The goal of SfM is straightforward: to determine all camera poses $C = \{\mathbf{C}_{i,n} : i \in \{1, \ldots, N_i\}, n \in \{1, \ldots, N_i\}$, camera parameters $K = \{k_i : i \in \{1, \ldots, N_i\}\}\}$, and reconstruct the 3D structure of the scene, represented by a set of N_p points $P = \{\mathbf{p}_i : i \in \{1, \ldots, N_p\}\}$, corresponding to the keypoints detected in the captured frames.

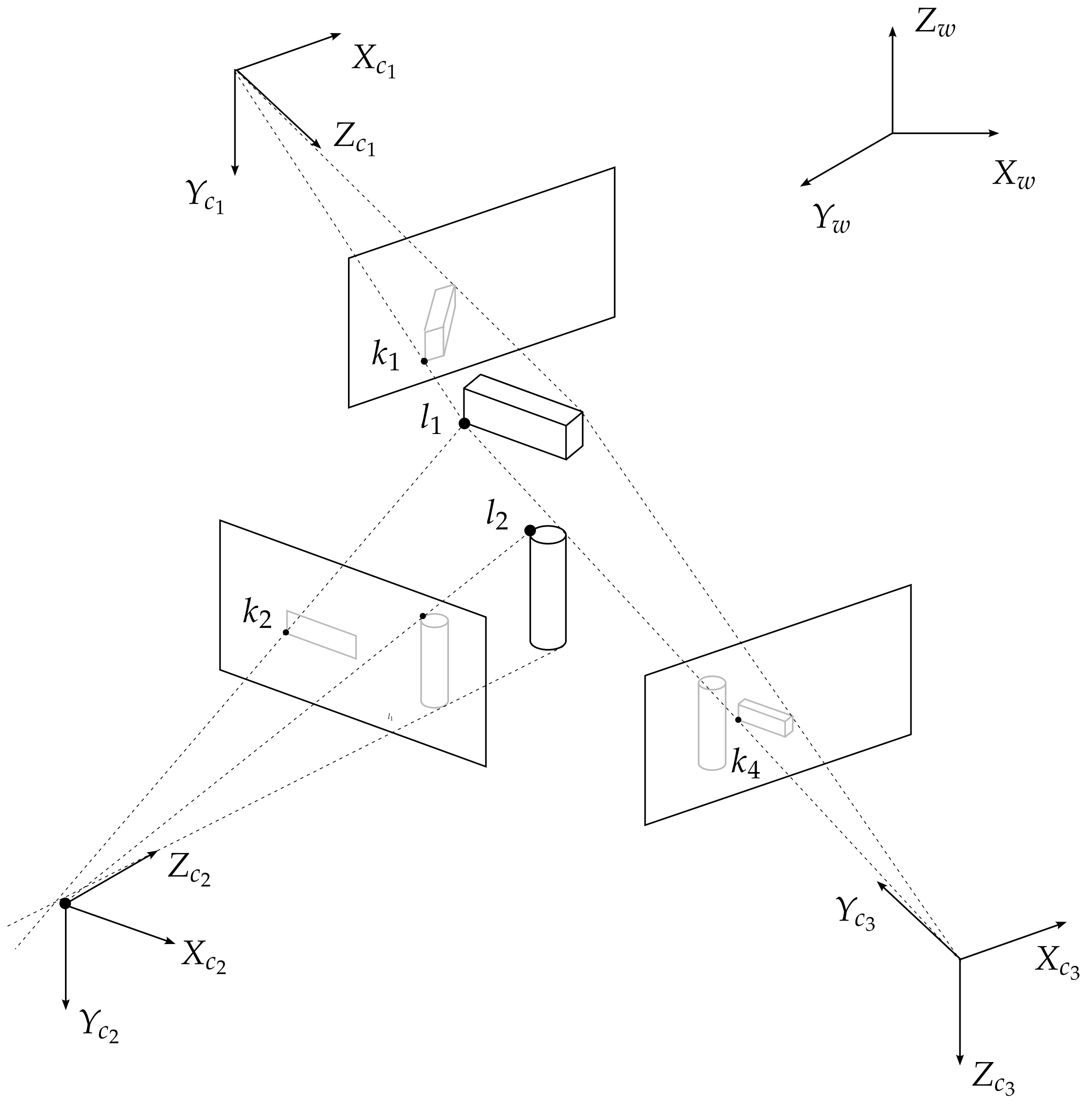


FIGURE 2.14: Structure from motion problem. The order of the captured frames is arbitrary.

Overall, as illustrated in Figure 2.15, SfM encompasses three core stages:

feature detection and matching, motion estimation, and 3D structure recovery. A widely-used technique for executing the latter two stages — motion estimation and 3D structure recovery—is Bundle Adjustment (BA). BA represents a substantial geometric parameter estimation challenge, involving the determination of 3D feature coordinates, camera poses, and camera calibration parameters [Tri+00]. BA is also a crucial stage in VSLAM's mapping process, that is why it is discussed further in Section 2.5.

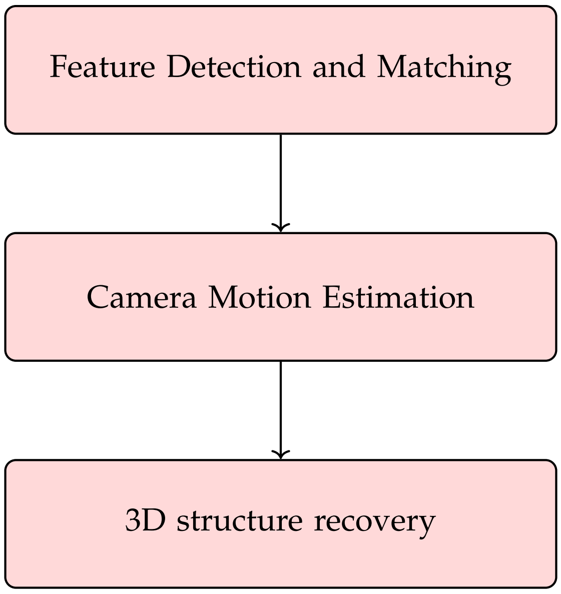


FIGURE 2.15: Flow of the basic SfM algorithms [Özy+17].

Following a brief introduction to three key topics – VO, Visual SLAM, and SfM – the subsequent paragraphs will provide an overview of these three areas. The discussion will particularly emphasize the differences in their complexity, the nature of their input and output data, and the constraints related to time.

Input and Output The first major difference between all three problems – VO, VSLAM, and SfM – lies in their input and output data. For VO and VSLAM, the input typically comprises either a single image or a set of images, particularly when the system is equipped with multiple cameras. This data is drawn from a continuous stream of images captured by the sensors as they traverse through an environment. In simpler terms, VO and VSLAM process consecutive frames iteratively. The input for SfM differs significantly. Rather than relying on the iterative processing of consecutive frames, SfM typically utilizes a collection of unordered images that capture various views of a scene. Distinct from VO and VSLAM, SfM does not necessitate a temporal sequence of images. It operates with static images, which can be sourced from different times, perspectives, and even various cameras. All these images are provided simultaneously, at once. It operates with static images, which can originate from various times, perspectives, and even different cameras. All images are provided at once.

The differences are even more clear comparing the output of three problems. VO's output is the incremental motion information, typically presented as a series of relative transformations or poses over time. VO does not aim to construct a complete map of the environment or maintain a global reference. In contrast, VSLAM provides not only the current pose of the platform but also constructs and maintains a map of the environment. To put it simply, as its name states, VSLAM is responsible for both localization and mapping. Lastly, SfM outputs the pose of each provided image, the overall map, and camera parameters.

Complexity Complexity is a broad concept, so it's important to first define how it is understood. In the context of comparing VO, VSLAM, and SfM, a suitable candidate for defining complexity appears to be the number of parameters that are estimated. Being armed with such definition distinct differences emerge. VO, focused primarily on estimating the camera's motion, typically involves fewer parameters. It calculates the camera's pose frame-by-frame without the need for a comprehensive map of the environment, thereby limiting its parameter set to the camera's position and orientation or transformation between them. In contrast, VSLAM encompasses a broader range of parameters, as it not only estimates the camera's pose like VO but also builds and updates a map of the environment. This requirement significantly increases the number of parameters in VSLAM, as it must keep track of both the camera's trajectory and the positions of numerous environmental landmarks. SfM, on the other hand, operates with a static set of images and aims to reconstruct a 3D model of the scene. This involves not just estimating the scene's 3D points and the camera's pose for each image, but also additional camera parameters, increasing the complexity significantly, especially in situations involving numerous images or intricate environments. To summarize, in terms of parameter estimation complexity, VO is generally the simplest, followed by VSLAM, with SfM being the most complex.

Time constraints As a final point of the distinction between compared problems, time constraints also should be take into considerations. Clearly, VO has real-time operation constraints, processing images sequentially and instantly to estimate the camera's motion. VSLAM, while also often operating in real-time, faces greater time constraints due to its dual objectives of tracking the camera's position and building a map of the environment. In contrast, SfM typically does not operate under strict real-time constraints. It is often used in post-processing scenarios, where a set of static images is analyzed to reconstruct a 3D model of the scene. This lack of real-time requirement allows SfM to focus on accuracy over speed, dealing with extensive computational processes that are impractical for on-the-fly execution. To conclude, while VO and VSLAM are designed for immediate or near-immediate response scenarios, SfM operates under more relaxed time constraints, prioritizing detailed and accurate reconstruction over speed. Consequently, VO faces the strictest time constraints, followed by VSLAM, while SfM allows for more flexibility in terms of time.

Table 2.3 provides a summary of the considerations described previously. At first glance, Visual Odometry (VO), Visual Simultaneous Localization and Mapping (VSLAM), and Structure from Motion (SfM) may appear similar, but they are fundamentally different problems. This is particularly evident when examining their complexity, input data and outputs, as well as the time constraints, where numerous distinctions among them become clear.

Property	*VO*	*VSLAM*	*SfM*
Single Input	Image	Image	Set of Images
Output	Current pose	Current pose, recent poses and map	Poses and Map
Time constraints	●●●	●●●	●
Overall complexity	●	●●●	●●●

TABLE 2.3: Comparison of three interconnected 3D problems utilizing computer vision.

2.5 Visual SLAM

Vision is a major sense not only for humans, but also for many animals. According to [RP10], people's activities including perception and acquiring knowledge are based up to 80% on eyesight. Capabilities of human perception and interpreting images created by the advanced vision system is the inspiration for scientists and engineers to develop systems with similar advantages. However, this is not a simple task and for many years researchers have been studied various vision based solutions. Localization and mapping problem is no exception. In previous sections, beginning of SLAM was described which focused on other types of sensors. Over time cameras became better in terms of resolution and image quality, they also got cheaper. Computer's performance also leveled up. These several factors caused that vision based solutions got attention of scientists. Although SLAM has already been introduced in previous sections discussing the SLAM problem (Section 2.1–Section 2.4), it is beneficial to revisit some terminology and definitions, this time focusing exclusively on the context of VSLAM.

One significant advantage of using cameras in VSLAM over other sensors, such as LiDAR or ultrasonic sensors, is the richness of the data they provide. Cameras capture detailed visual information that includes textures, colors, and patterns, enabling more nuanced and context-aware mapping and localization. This high-resolution data facilitates the identification and differentiation of various environmental features, which is crucial for accurate mapping and navigation in complex and dynamically changing environments. Moreover, cameras are highly versatile in terms of their adaptability to different lighting conditions and environments, ranging from indoor scenarios to outdoor landscapes.

Another notable benefit of employing cameras in VSLAM systems is their cost-effectiveness and accessibility compared to more other sensors. Cameras, particularly those used in consumer electronics, have undergone significant advancements in terms of quality and affordability, making VSLAM systems more scalable and economically viable for a wider range of applications. Additionally, the use of cameras allows for the implementation of advanced computer vision algorithms, such as deep learning techniques, which can further enhance the system's ability to understand and interpret complex visual scenes. This integration of sophisticated image processing methods not only improves the accuracy of localization and mapping but also opens up possibilities for additional functionalities like object recognition and scene understanding, thereby broadening the scope of VSLAM systems in various fields including autonomous vehicles, robotics, AR, and smart city infrastructure.

In the following sections, continuing the approach from earlier chapters on SLAM, the VSLAM problem is defined, its architecture and categorization are described. Additionally, key publications and research in the field of VSLAM are highlighted.

2.5.1 Visual SLAM problem formulation

The formulation of the VSLAM problem does not significantly differ from the standard SLAM formulation. In this scenario, a mobile platform is placed in an unknown environment populated with L landmarks, denoted as $l_j, j \in \{1, 2, ..., L\}$. The state of the platform at any given time point n is represented by x_n. A comprehensive discussion regarding the representations of the platform's and landmarks' states is forthcoming in Section 2.5.2. As

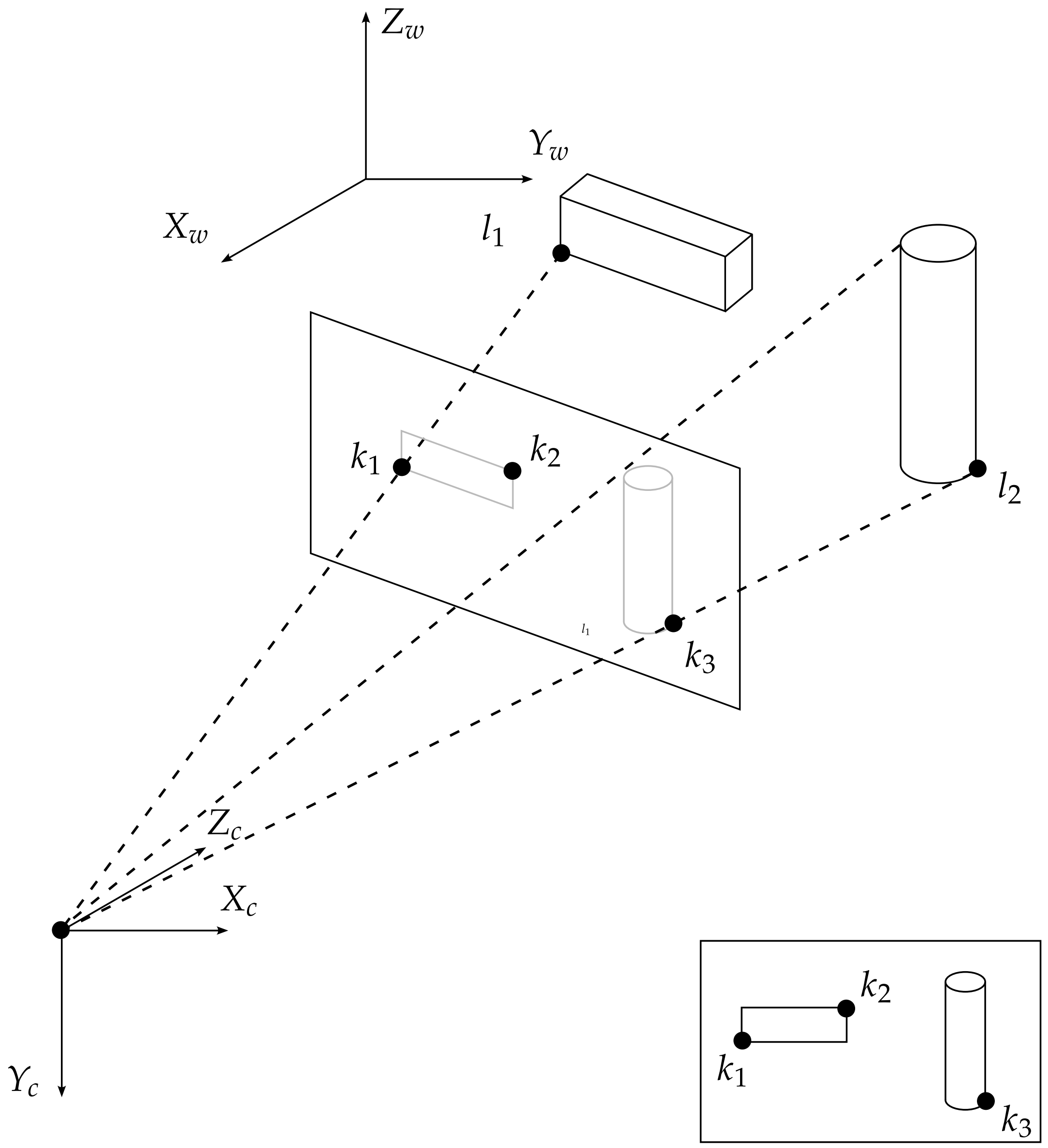


FIGURE 2.16: VSLAM correspondences between landmark, keypoints and world coordinate system, camera coordinate system. Landmark l_1 was observed current frame by detecting keypoint l_1. Respectively, landmark l_2 is also observed by detecting keypoint k_3. It is also worth noting

the platform navigates through the environment at successive time intervals $\{n, n+1, \ldots\}$, it observes these landmarks. However, due to sensor noise and other issues, which are elaborated in Chapter 3, the measurement $z_{n,j}$ of a landmark's state l_j at time n is approximate, not exact. The goal of the VSLAM problem is to estimate the current pose of the platform x_n as well as the set of all landmarks l, corresponding to the localization and mapping processes.

In Figure 2.17

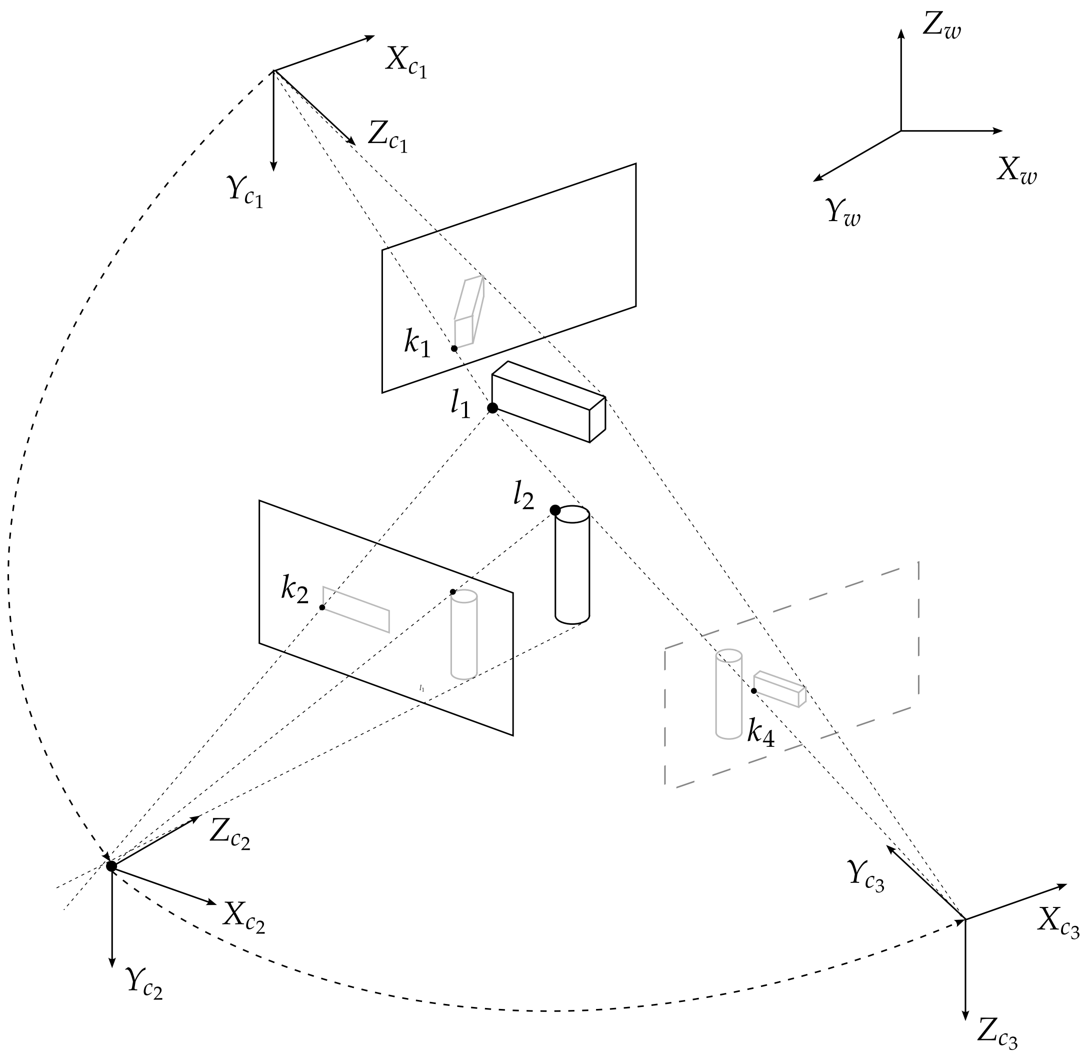


FIGURE 2.17: Visual SLAM problem with two keyframes and current frame marked with dashed line

2.5.2 State and observation representation

To allow deep understanding of VSLAM it is required to dive into details and discuss common representation. In previous section platform's pose x_n at given time point n and map being set of all landmarks l_j were introduced. Accurate and efficient representation is pivotal, as it directly influences the system's ability to accurately localize itself and build a map.

Firstly, let's define the platform's position. To describe a position in the Euclidean R^3 space, three variables are necessary which is given by Equation 2.23.

$$\mathbf{t} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathcal{R}^3 \tag{2.23}$$

In parallel, the state of a landmark can be defined similarly. Given that a landmark is a point in 3D Euclidean space, it is represented as described in Equation 2.24:

$$\mathbf{l}_j = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \in \mathcal{R}^3 \tag{2.24}$$

Continuing our exploration of state representation in VSLAM, let's now focus on the platform's orientation. Orientation in a three-dimensional space can be represented in various ways. This representation typically involves rotational components about the three principal axes. Intuitively, orientation can be simplified to three angles: yaw, pitch, and roll, commonly known as Euler angles. While Euler angles offer simplicity, they are fraught with disadvantages, including ambiguity and computational inconvenience in transformations.

Therefore, orientation is often represented using rotation matrices. The following form is utilized to express orientation in terms of rotation matrices:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \in SO(3) \tag{2.25}$$

The rotation matrix $\mathbf{R}$ is an element of the Special Orthogonal group $SO(3)$, possessing two essential properties. Firstly, its inverse is equal to its transpose. The definition of $SO(3)$ can be formally expressed as:

$$SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times 3} \mid \mathbf{R}^T\mathbf{R} = \mathbf{I}_3, \det(\mathbf{R}) = 1\} \tag{2.26}$$

In this definition, $\mathbf{I}_3$ represents the 3x3 identity matrix. The conditions $\mathbf{R}^T\mathbf{R} = \mathbf{I}_3$ and $\det(\mathbf{R}) = 1$ ensure that $\mathbf{R}$ maintains the properties of orthogonality and unit determinant, which are characteristic of matrices in $SO(3)$. The orthogonality property may be formulated by Equation 2.27.

$$\mathbf{R}^{-1} = \mathbf{R}^{T} = \begin{bmatrix} r_{11} & r_{21} & r_{31} \\ r_{12} & r_{22} & r_{32} \\ r_{13} & r_{23} & r_{33} \end{bmatrix} \tag{2.27}$$

Following the initial discussion on how orientation and position are represented within a platform, the transition to a combined representation can be made.

After the brief introduction of the orientation and position representation of the platform it is now possible to introduce a unified representation. The Special Euclidean group in three dimensions, denoted as $SE(3)$, serves this purpose by providing a comprehensive framework for modeling the rigid body transformations in a three-dimensional space. SE(3) is a mathematical group that combines the rotation of an object, described by the Special Orthogonal group SO(3), with its position in space, denoted by $\mathcal{R}^3$. Elements of $SE(3)$ are typically represented as 4x4 homogeneous transformation matrices, which consist of a 3x3 rotation matrix from $SO(3)$ that encodes the object's orientation, and a 3x1 translation vector from $SO(3)$. This is denoted by Equation 2.28

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \mid \mathbf{R} \in SO(3), \mathbf{t} \in \mathbb{R}^3 \right\} \tag{2.28}$$

Finally, the platform's pose $\mathbf{x}$ can be expressed as a $SE(3)$ (Equation 2.29). For an discussion on state representation, readers are directed to [Gao+17], and for an in-depth understanding of rotation representation, the work of Diebel is recommended [Die06].

$$\mathbf{C} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \in SE(3) \tag{2.29}$$

To fully define the VSLAM problem it is required to introduce several variables. Their meanings are defined as follows:

- $\mathbf{C}_n$ — This symbol represents the state of the n^{th} keyframe within the system, encapsulating both the orientation and the position of the keyframe at a given instant. It serves as a snapshot of the system's pose in space at the time associated with the keyframe.

- l_n — Denotes the n^{th} landmark in the environment. It is a fixed point in space that the VSLAM system uses as a reference to understand and navigate the surrounding area.

- $z_{n,k}$ — Represents the observation of the k^{th} landmark from the perspective of the n^{th} keyframe. It is the measurement data that correlates the observed landmark with a specific keyframe, contributing to the system's awareness of its environment.

- $C_{0:n} = \{\mathbf{C}_0, \mathbf{C}_1, \ldots, \mathbf{C}_K\}$ — This set contains all the states from the initial keyframe $\mathbf{C}_0$ up to the state of the K^{th} keyframe, offering a chronological sequence of the system's spatial orientation and position up to that point.

- $l = \{l_1, l_2, \ldots, l_L\}$ — Signifies the entire collection of landmarks within the environment that the VSLAM system has recognized and utilized. This set is foundational for the system to lock onto the physical world and maintain a coherent internal map.

- $Z_{0:n} = \{z_1, z_2, \ldots, z_n\}$ — Constitutes the aggregation of all observations made by the system from the initial observation z_1 through to the observation z_n. It embodies the cumulative sensory data that informs the VSLAM system's understanding of its spatial relationship with the landmarks.

Let consider the illustrative instance of a VSLAM process as depicted in Figure 2.18. The platform explores the very simple environment consisted of $L = 7$ landmarks. The set of all landmarks may be formulated as $l = \{l_1, l_2, \ldots, l_L\}$. During the process VSLAM system created $K = 13$ keyframes. For each keyframe n, there is a state C_n representing rotation and position. The set of keyframes is given by $C = \{C_1, C_1, \ldots, C_{13}\}$. Finally, there are observations marked as $z_{n,j}$ referring to keyframe n and landmark j. The set of all observations is given by $Z = \{z_{1,1}, z_{2,1}, \ldots z_{13,2}\}$.

It is worth noting that this particular example generally shares similarities with the scenario illustrated in Figure 2.1. However, there are major elements which are not visible in the latter. Namely, there is a distinction between typical frames and keyframes. Frames are primarily utilized for real-time pose estimation, allowing the system to estimate its current position and orientation. In contrast, keyframes carry additional significance. Not only do they contribute to the immediate pose estimation like frames, but they are also selected to be part of the map construction.

2.6 Visual SLAM components

The big picture of the VSLAM system does not differ significantly comparing to general scheme of SLAM systems. As a result, three basic components remain the same (see Figure 2.2).

As illustrated in Figure 2.19, the essential building blocks of VSLAM systems also encompass the frontend, backend, and loop closure mechanisms. Furthermore, VSLAM systems have the capability to augment their functionality by incorporating semantic information extracted from visual data. This feature is highlighted on the VSLAM architecture diagram by the optional

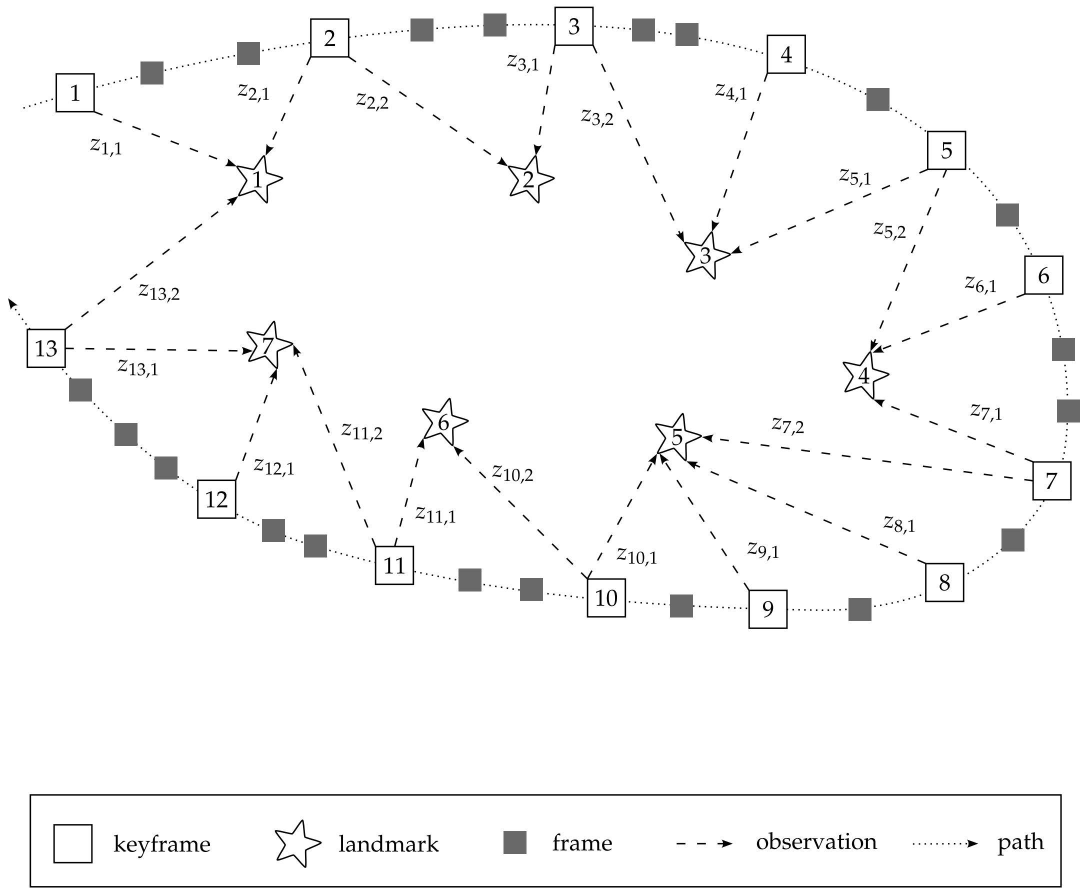


FIGURE 2.18: Example of VSLAM [Mor15]

semantic component. The subsequent sections, from Section 2.6.2 to Section 2.6.5, will provide an in-depth exposition of each individual component.

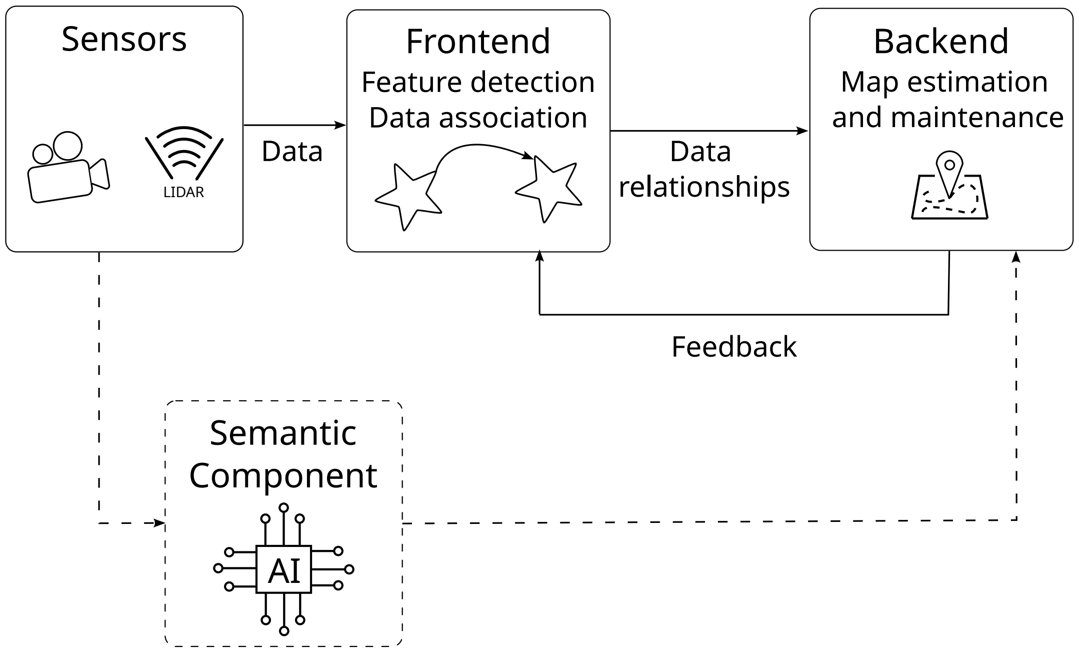


FIGURE 2.19: Visual SLAM components

2.6.1 Sensors data collection

The first task which every VSLAM system is responsible for is data collection. This component communicates with the available sensors of the platform and gathers the data. Despite the fact that initially it may seem that the component responsible for data collection is relatively easy to implement, attention should be paid to several things for which it is responsible.

First, the data collection component of a VSLAM system plays a crucial role in data synchronization. This task involves ensuring that the data from various sensors are temporally aligned, which is essential for accurate data fusion and interpretation. Time synchronization ensures that the timestamps of data from all sensors match up, allowing the system to construct a coherent representation of the environment. This is particularly pivotal when combining data from high-speed sensors or when dealing with systems that operate in dynamic environments, as any discrepancy in time can lead to errors in localization or mapping. For instance, consider a platform equipped with an RGB camera and a LIDAR sensor. The RGB camera captures frames 30 times per second, whereas the LIDAR sensor takes 0.1s to deliver 3D data. Clearly, the data rates between these two sensors vary significantly. The problem is shown in Figure 2.20.

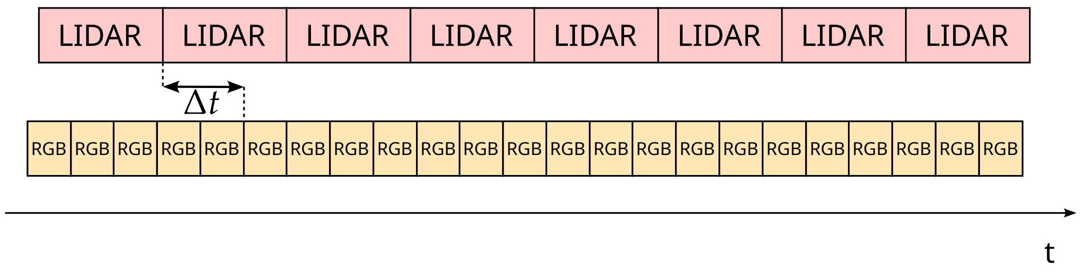


FIGURE 2.20: Data synchronization challenge from various sensors

Secondly, another critical function of the data collection component is the alignment of data from different sensors. This process, known as sensor fusion, involves aligning data spatially to ensure that the information from one sensor corresponds accurately with that from others. Visual data from cameras must be precisely aligned with inertial measurements from accelerometers and gyroscopes or depth information from LIDAR sensor. This alignment is essential for creating a unified and accurate 3D map of the environment but also for pose estimation. To give an example, Figure 2.21 presents a mobile platform equipped with a LIDAR and two cameras. Their distance between them is Δd. Due to various coordinate systems they operate it is required to align not only those two image but also LIDAR data which is above the two mentioned cameras.

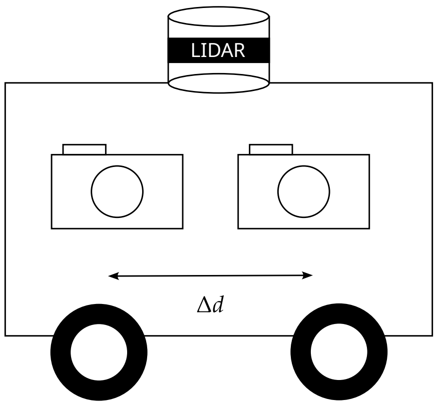


FIGURE 2.21: Mobile platform equipped with various sensors.

2.6.2 Frontend

As depicted in Figure 2.19, the frontend receives data from the data acquisition component. Its main task is to process this data in a manner that makes it compatible with the backend. This processing involves several critical tasks including data preprocessing, feature detection, feature tracking, data association, pose estimation, and triggering keyframe creation. The frontend effectively acts as the initial filter and organizer of incoming visual information, preparing it for the more complex analytical processes handled by the backend.

The frontend's role is pivotal as it directly influences the efficiency and accuracy of the VSLAM system. By efficiently detecting and tracking visual features, it can provide reliable input for constructing a preliminary map of the environment and for estimating the camera's motion relative to its surroundings. Additionally, the frontend is responsible for identifying key moments or positions—known as keyframes—which are essential for reducing computational load and improving the map's accuracy. These keyframes serve as reference points in the map optimization processes carried out by the backend.

In subsequent sections, each of these tasks will be examined in detail, with an exploration of their contributions to the robustness and reliability of the VSLAM system. The discussion will encompass the algorithms and techniques used in the frontend, how they are integrated into the VSLAM pipeline, and their effects on the system's overall performance. This detailed breakdown will not only highlight the functionalities of the frontend but also underscore its critical role within the context of VSLAM's operational hierarchy.

Feature detection

Feature detection is a process of finding features i.e. pattern of the image which is characteristic and easily identifiable. It must be different from its neighbor region concerning intensity, color and texture [FS12]. There are many algorithms which search image for the features. These may be points but there are also other types like lines or circles [Li+15]. Especially, there are numerous algorithms available that looks for keypoints. The keypoint is defined as a small patch which is highly recognizable and unique [KB17]. Computer vision community developed many feature detectors including SIFT [Low04], SURF [BTG06], CENSURE [AKB08], ORB [Rub+11], KAZE [ABD12] and its improved version A-KAZE [ANB13] or DAISY [TLF10].

SIFT developed by Lowe in 2004, detects and describes local features in images, renowned for its robustness to changes in scale, rotation, and illumination. These features are extensively utilized in object recognition, robotic mapping, and 3D modeling due to their stability and accuracy. In contrast, SURF, introduced by Bay, Tuytelaars, and Gool, accelerates the feature detection process compared to SIFT. It employs a Hessian matrix-based measure for detection and a distribution-based descriptor, making it suitable for real-time applications due to its efficiency and resistance to viewpoint and illumination changes.

BRISK is created by Leutenegger, Chli, and Siegwart. It is a feature detector and descriptor that is designed for speed without sacrificing robustness. BRISK uses a novel method of sampling points in a pattern around a central pixel and compares their intensity pairs to create a binary string that serves as the descriptor. This method allows BRISK to be scale-invariant and rotation-invariant, making it highly effective in diverse imaging conditions.

CENSURE, offers a scale and rotation-invariant feature detector known for its computational speed, particularly useful in stereo vision tasks in robotics and mobile vision systems. Following this, ORB (Oriented FAST and Rotated BRIEF) by Rublee et al. in 2011, combines the FAST keypoint detector with the BRIEF descriptor [Cal+10], providing a cost-effective, fast alternative to SIFT and SURF, ideal for real-time video analysis and mobile applications.

KAZE, distinguishes itself by employing non-linear scale spaces instead of the traditional Gaussian blurs, improving the accuracy and robustness in detecting features in images with natural noise and fine textures, particularly effective for high-resolution images and complex motion patterns. Its variant, A-KAZE, introduced in 2013 by Alcantarilla, Nuevo, and Bartoli speeds up the process by using fast explicit diffusion for scale-space construction, maintaining the effectiveness of KAZE while enhancing efficiency.

DAISY, introduced by [TLF10] in 2010, serves as a dense feature descriptor designed for efficient matching in wide-baseline stereo images, aiding significantly in high-speed and high-accuracy stereo matching tasks, commonly used in 3D scene reconstruction and visual SLAM applications.

SuperPoint [DMR18], introduced by DeTone, Malisiewicz, and Rabinovich, represents a significant advancement in feature detection by leveraging deep learning techniques. This detector is unique as it integrates both interest point detection and feature description into a single convolutional neural network. SuperPoint is trained on synthetic and real-world data to automatically learn and generalize salient and repeatable interest points across a wide variety of images. The strength of SuperPoint lies in its ability to operate effectively across different conditions and environments, demonstrating high robustness and precision in feature detection. It is especially proficient in handling scenarios with significant changes in viewpoint, scale, and illumination, which are challenging for traditional detectors. Moreover, its design enables it to perform both detection and descriptors calculation simultaneously, enhancing processing speeds and making it highly suitable for real-time applications.

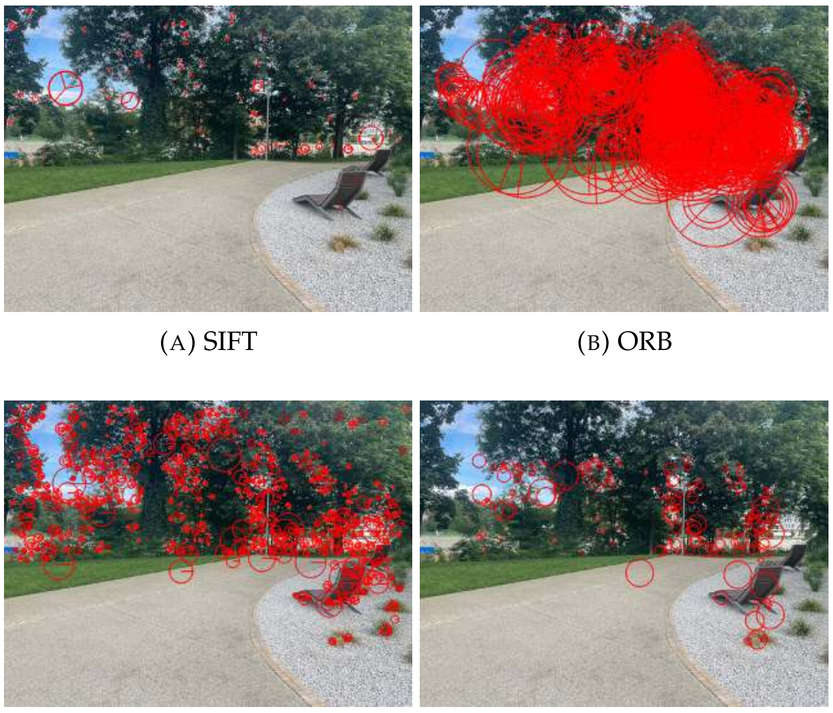
Each detected keypoint is paired with a corresponding descriptor. Several of the detectors previously discussed have the dual capability of identifying keypoints and generating descriptors for these features. A *Descriptor* mathematically characterizes a visual feature identified in an image. Typically, this descriptor d is expressed as a vector in $\mathbb{R}^{N_d}$, where N_d is the length of the descriptor. The primary function of a descriptor is to enable comparison with other descriptors. If the calculated distance $l = f(d_i, d_j)$ between two descriptors d_i and d_j approaches zero, it indicates that the descriptors are similar. On the other hand, a greater distance suggests that the descriptors relate to different keypoints. The closer the distance value l is to zero, the more precise the match between the keypoints. This process, known as feature matching, is essential for correlating features across various images and is thoroughly discussed in Section 2.6.2. A summary of various descriptors can be found in Table 2.4, and Figure 2.22b illustrates examples of keypoints detected by various detectors.

Descriptor	*Type*	*Notes*
ORB	Binary	Efficient, good for real-time applications.
SIFT	Gradient-based	Robust to scale and rotation changes, computationally intensive.
SURF	Hessian-based	Faster than SIFT, good for real-time applications, moderately invariant to scale and rotation.
CENSURE	Binary	Fast, suitable for stereo vision tasks, not scale or rotation invariant.
A-KAZE	Non-linear scale spaces	Offers robustness to noise and fine textures, computationally expensive.
KAZE	Non-linear scale spaces	Similar to A-KAZE but slower, robust to noise and texture.
DAISY	Gradient-based	Efficient for dense matching in stereo vision, not well-suited for large scale changes.
SuperPoint	Deep learning-based	Excels in diverse conditions, handles viewpoint and illumination changes well.

TABLE 2.4: Comparison of descriptors [GHT11; TS18]

Feature matching and data association

After obtaining features of the current image VO algorithm has to detect which features are still visible on the new obtained frame and which features of the previous frame relate to the new one. These steps are called *feature matching*. The quality of the feature matching in a great extent relies on the feature descriptor, which transforms found feature into a vector of numbers



(A) SIFT (B) ORB

(C) KAZE (D) STAR

FIGURE 2.22: Comparison of various keypoint detectors

representing compressed information about the region. Invariance of rotation, translation and scaling is profoundly demanded attribute. Figure 2.23 shows an example of matching ORB keypoints.

There are two classic feature matching methods: brute-force and k-nearest neighbors. The brute-force method is one of the simplest approaches, where each descriptor in one image is compared to every descriptor in another image using a distance metric. The pairs with the smallest distances are selected as matches. This method is straightforward but can be computationally intensive. In the k-nearest neighbors method, for each descriptor in the first set, the k closest descriptors in the second set are identified based on the chosen distance metric. Typically, k is set to 2. Additionally, Lowe's ratio test, as proposed by David Lowe in [Low04], is used. This test checks the ratio of distances between the closest and second-closest match. If the ratio is below a threshold (commonly 0.75), the match is considered good. This helps in reducing the number of false matches by eliminating those that are not significantly better than the second-best match.

In recent years, a novel approach has emerged within the field of feature matching, leveraging advancements in machine learning. As detailed in [Sar+20], a machine learning-based matcher was introduced, representing a significant shift from traditional algorithms. This matcher utilizes deep neural networks to learn optimal feature matching strategies directly from

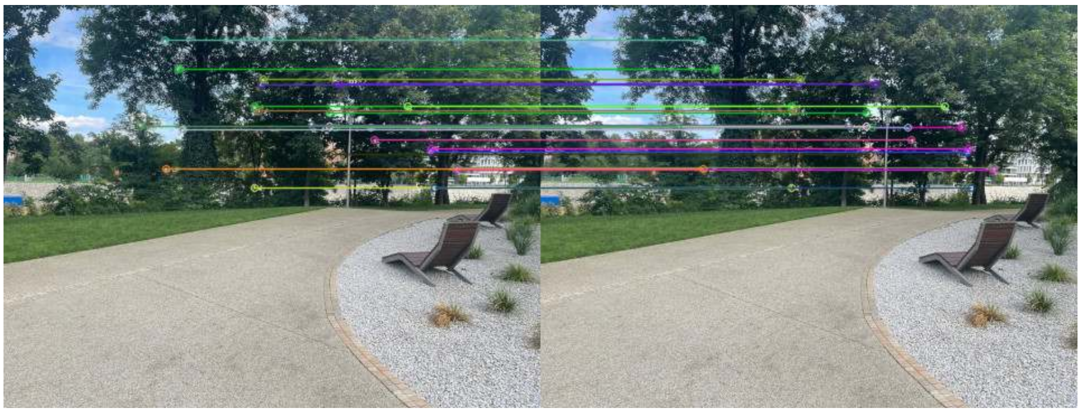


FIGURE 2.23: Result of matching procedure. Only several matches is drawn.

data. Unlike conventional methods that rely on handcrafted descriptors and heuristic matching rules, this machine learning approach can adapt to a wide range of visual conditions and dynamically improve through continuous learning. Such capabilities enable the matcher to handle complex variations in appearance and geometry that often pose challenges in traditional visual odometry and VSLAM systems.

It is also worth noting that not only is the keypoints matching approach possible. Figure 2.24 illustrates various potential correspondences that can be used in visual processing tasks. The most straightforward among these is the 2D-2D correspondences, which involve matching features between two 2D images. This method is typically used in stereo vision or when processing sequential frames in video. By establishing correspondences between two images, depth information can be inferred, which is essential for reconstructing scenes in 3D and estimating the relative motion between cameras or successive frames. Furthermore, 2D-3D correspondences are commonly uti-

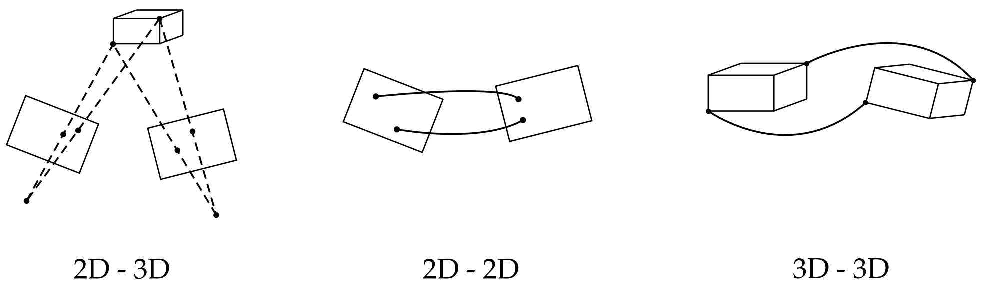


FIGURE 2.24: Various types of data correspondences [You+17].

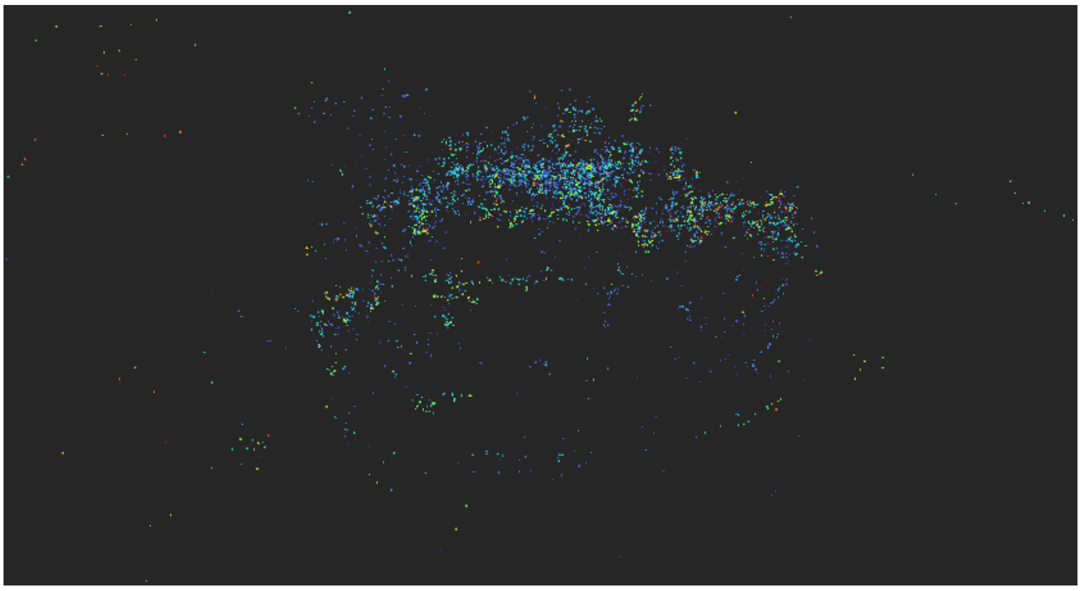
lized when 2D features detected in images are matched with a pre-existing 3D model or map. This approach allows each 2D feature in the image to be associated with a specific 3D point in the world coordinate system, enabling precise localization and camera pose estimation. Additionally, 3D-3D correspondences are crucial in applications like 3D reconstruction, where multiple scans or images of the same scene are taken from different viewpoints. Matching 3D points from different datasets or different times in a dynamically changing environment allows for the integration of various datasets into a comprehensive 3D model. This facilitates tasks such as complex scene analysis and virtual reality modeling, where accurate and detailed environmental representation is required.

In the context of data association, it is imperative to differentiate between sparse and dense VSLAM approaches. Sparse VSLAM selectively focuses on a limited number of significant features within the environment, such as keypoints or landmarks. These features are specifically chosen for their distinctiveness and their ease of tracking, simplifying the mapping process by prioritizing these critical elements. In contrast, dense VSLAM seeks to construct an exhaustive representation of the environment by processing extensive data at the pixel or voxel level, aiming to produce a detailed and continuous map. The distinction between these two methods becomes apparent when viewing the same scene: sparse VSLAM emphasizes only the key features, while dense VSLAM offers a comprehensive and finely detailed visualization of the entire environment, as depicted in Figure 2.25.

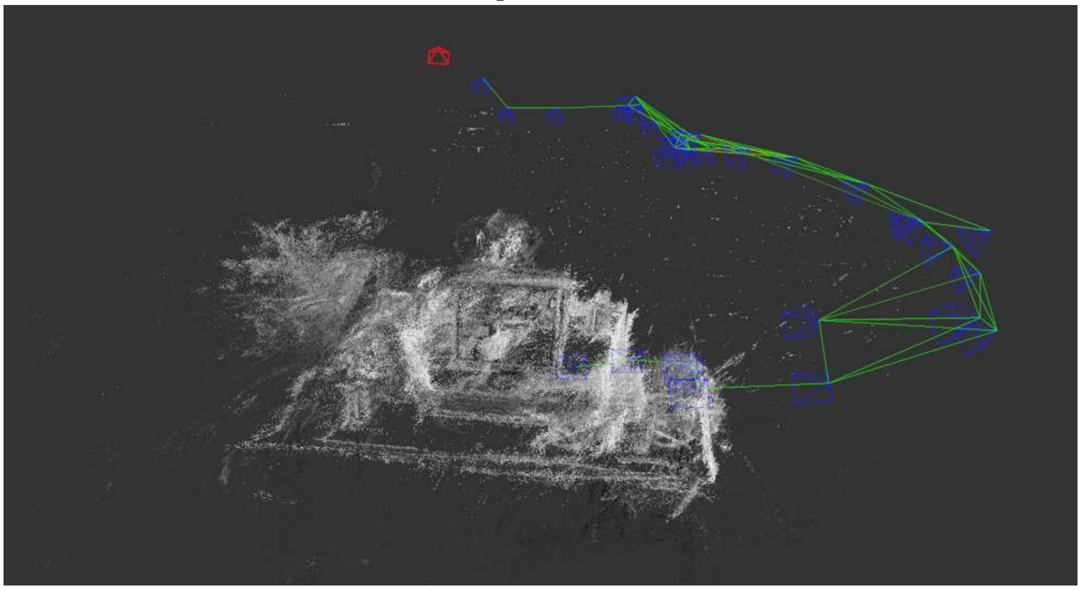
Keyframe creation

In graph-based VSLAM systems, keyframes are essential components that significantly influence both the localization and mapping processes. Serving as foundational elements of the map, keyframes not only help in reducing the size of data required for map representation but also play a crucial role in outlier rejection during the optimization process, as highlighted in [DLD23]. The strategic selection of keyframes is therefore critical to the overall performance of the system.

There exists a balance between achieving high accuracy and maintaining low compute times. If too few keyframes are selected, the map may lack necessary detail, which can impact on the accuracy of the localization module, potentially causing it to lose tracking. Conversely, an excessive number of keyframes can slow down the system significantly. Therefore, the process of choosing keyframes must be carefully managed to optimize both the efficiency and effectiveness of the VSLAM system. This involves determining the optimal intervals at which keyframes should be selected to provide sufficient coverage and detail of the environment while avoiding redundancy that could impair system performance. This trade-off is pivotal in designing systems that are both fast and reliable, ensuring robust localization and accurate mapping.



(A) Sparse SLAM



(B) Dense SLAM

FIGURE 2.25: Comparison of sparse and dense VSLAM map. The dense VSLAM map contains much more points and it is clear that scene shows a desk with a computer, while the sparse VSLAM map is not reach with the details and it is difficult to recognize this is the same scene.

The frontend of a VSLAM system is responsible for monitoring the quality of both the localization and mapping processes. Its key function is to assess when a new keyframe is needed. If specific conditions are fulfilled such as significant changes in the environment or inadequate data from current keyframes—the system triggers the creation of a new keyframe. This involves generating new data associations and landmarks, which are essential for updating and refining the map. By continuously monitoring and responding to these conditions, the frontend ensures that the system maintains high accuracy and efficiency in its mapping and localization tasks.

2.6.3 Backend

After frontend finishes its processing, the next step might be backend. In most cases backend task is not performed every frame. The backend complements the frontend by performing tasks critical to maintaining the consistency and accuracy of the map over time. In particular, backend is responsible for map optimization and its maintenance. As backend is very similar to the backend of typical SLAM, only brief overview of these two task is provided.

Map optimization

Map optimization is a crucial process in VSLAM systems, aimed at refining the accuracy and consistency of the 3D map generated from visual data. This involves the use of algorithms such as bundle adjustment, which seeks to minimize the reprojection error across all observed features by fine-tuning both camera poses and landmark positions. By iteratively adjusting these parameters, the map optimization process ensures that the virtual representation of the environment aligns closely with the actual world. This is especially important when the camera moves through extensive environments or revisits previously mapped areas, as it helps maintain the reliability of the SLAM system's navigational and interactive capabilities. The success of map optimization directly impacts the system's performance in tasks requiring high levels of precision, such as robotic navigation in complex, dynamic settings.

Map maintenance

Map maintenance, on the other hand, involves the ongoing management and updating of the map to keep it relevant and usable over time. This includes handling the addition and removal of features from the map to avoid clutter and redundancy, which can degrade system performance. Effective map maintenance also requires identifying and correcting any inconsistencies that arise from dynamic changes within the environment, such as moving objects or alterations in the layout of a space. Furthermore, the backend of a VSLAM system manages data association, ensuring that new observations are correctly matched to existing map elements, which is critical for accurate localization and robust map updates. Additionally, map maintenance strategies must deal with issues like scale drift and loop closures to ensure long-term consistency and usability of the map, supporting applications that depend on prolonged and repeated use of the SLAM system in evolving environments.

2.6.4 Loop Closure

In many scenarios, a platform exploring an environment will eventually return to a location it has previously visited. This event is known as a loop closure in the context of VSLAM systems. Detecting and properly handling loop closures is crucial for maintaining the robustness and accuracy of the system.

Loop closures are pivotal for correcting drift that accumulates over time as the platform navigates through the environment. Drift, or the gradual deviation from the true trajectory, is a common issue in VSLAM due to small errors in measurement and estimation that accumulates during continuous operation. When a loop closure is detected, it provides a unique opportunity to correct this drift by aligning the current position with a previously recorded position in the map. Furthermore, effective management of loop closures helps in optimizing the map's efficiency by preventing redundancy. Recognizing when the system has returned to a previously mapped area allows it to reconnect with and utilize existing map data instead of redundantly adding similar data i.e. keyframes and landmarks.. This not only saves computational resources but also simplifies the map, making it easier to manage. To ensure the system remains robust against errors and efficient in its operation, sophisticated algorithms are employed to detect loop closures reliably. These algorithms compare newly captured data with existing data in the map to identify potential revisits to locations. Upon confirmation of a loop closure, the system can then adjust the entire map by realigning it to minimize the total error across all measurements, thereby enhancing the overall accuracy and reliability of the VSLAM system.

Historically, researchers were aware of the loop closure importance, even in classic SLAM problem. Around year 2000, one may already find scientific articles about the loop closure and experimental results indicating better performance [GK99; New+02]. Over the years, loop closure has become an important part of the whole system, being an obligatory component to obtain state-of-the-art performance. For instance, ORB-SLAM, StellaVSLAM, LSD-SLAM, all these systems implement loop closure component [ESC14; MT17; SSS19].

Visual loop closure detection is a crucial aspect of Visual Simultaneous Localization and Mapping (VSLAM) as it employs visual data to recognize locations previously visited. This process is closely linked to another computer vision task known as visual place recognition [Low+16; TBG22]. Loop closure primarily consists of two essential steps to ensure the accuracy and continuity of the system's map. The first step involves searching for keyframe candidates that resemble the current frame, a process deeply intertwined with visual place recognition. In fact, many algorithms used for visual place recognition are also applicable here, highlighting the strong connection between these two tasks. For those interested in deeper insights, a tutorial paper on this subject is highly recommended [Sch+23]. Within the VSLAM community, where execution time is crucial, bag of words methods like DBoW2 or FBoW [GT12; MM20] have become popular due to their efficiency. Recently, due to popularity gain of machine learning methods, techniques utilizing deep learning are also developed and evaluated [ZWS21; AK21].

Selecting the correct keyframes is vital as it directly impacts the system's ability to accurately recognize a revisited location. After identifying potential keyframes, the system evaluates them to determine the best matches, focusing on visual similarity and geometric consistency.

Following the selection of the most suitable keyframes, the system performs data association, where it matches features from the current frame with features from each selected keyframe. This stage is crucial for verifying the presence of a loop closure. If a keyframe meets the predefined conditions of the loop closure algorithm, such as a sufficient number of matching features and geometric alignment, the system confirms a loop closure event. Once a loop closure is detected, new constraints are added to the VSLAM system, enabling it to adjust and optimize the overall map. This optimization often involves minimizing the cumulative error across the map, thereby refining the trajectory and enhancing the accuracy of the entire system's spatial understanding. This iterative process of detection, verification, and optimization is essential for maintaining the integrity and usability of the map in dynamic environments.

Visual loop closure detection is strictly related with VSLAM as it leverages visual data to identify previously visited locations. This process is closely associated with another computer vision task known as visual place recognition [TBG22]. In general, loop closure consists of two steps. First, keyframe candidates are searched which are similar to the current frame. After the best keyframes are selected, then for every keyframe data association takes place. If the selected keyframe met conditions of the algorithm then loop closure is detected, new constraints are added to the system and finally optimization of the map is performed. A good survey is also

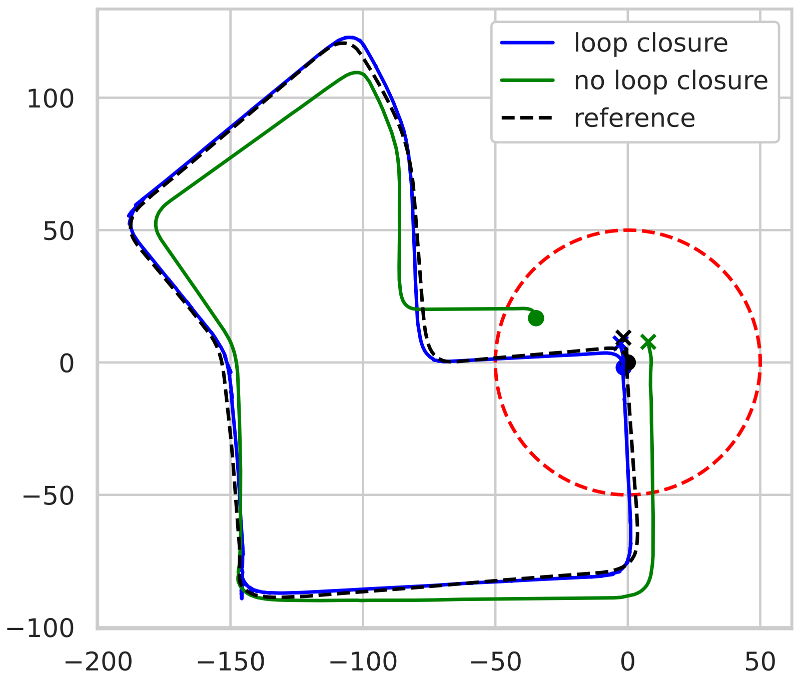
Loop closure can significantly enhance the quality of both the map and the trajectory in a VSLAM system, provided that loops are correctly detected.

However, there are scenarios where challenges arise. In some cases, loop closure might fail to be detected; this leads to the uncorrected drift accumulation, as illustrated in Figure 2.26a. This failure to identify a loop means that the system's estimate of its own trajectory deviates increasingly from the true trajectory. Conversely, incorrect loop closure detection—where a false loop is mistakenly confirmed—can also be problematic. Such false positives in loop detection can lead to erroneous corrections in the map, ultimately causing failures in the optimization process or wrong map estimation. These inaccuracies can disrupt the overall integrity and reliability of the system, making robust loop detection critical for successful VSLAM operation.

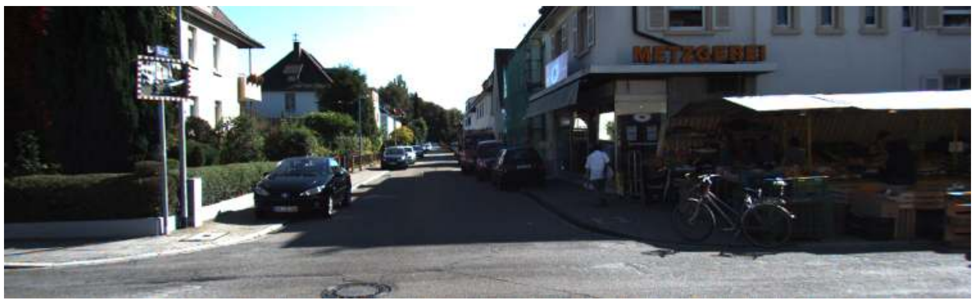
Figure 2.26 presents an example of loop closure in a VSLAM system using a sequence from the KITTI dataset. The first image 2.26a illustrates a comparison of two trajectories: one that incorporates loop closure correction (blue line) and one that does not (green line). The reference trajectory is indicated by the dashed line. The noticeable alignment of the blue trajectory with the reference in the area marked by a red circle indicates the effectiveness of loop closure in correcting drift that accumulates over the course of the sequence. The second 2.26b and third 2.26c images below depict the start and end frames of the sequence, respectively. These images provide a visual context to the trajectory paths shown in the first image, illustrating the urban environment in which the VSLAM system was tested. The presence of loop closure significantly enhances the accuracy of the trajectory mapping by ensuring that the end frame aligns closely with the starting frame, despite the complex urban setting and the lengthy route taken, as seen in the trajectory plot. This example effectively demonstrates how loop closure is crucial for maintaining consistent and accurate trajectory in navigation and mapping applications.

2.6.5 Semantic VSLAM

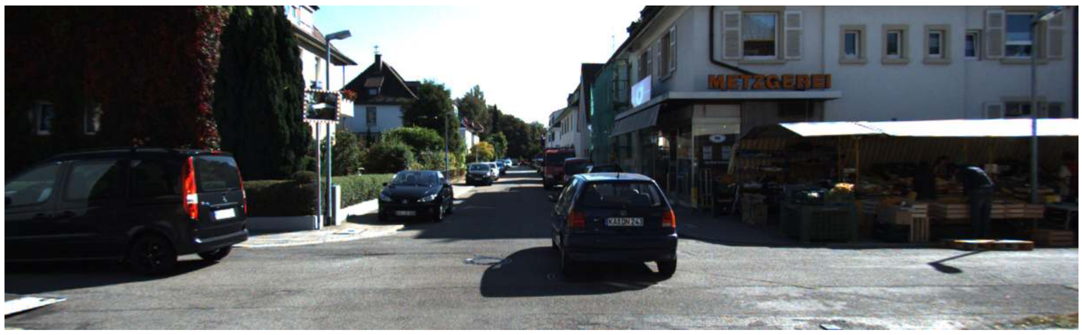
Cameras provide an abundance of information about the environment, enabling the implementation of not only VSLAM but also various scene understanding algorithms. In contemporary systems, it is increasingly common to integrate additional modules that analyze this data for semantic interpretation of the surroundings, adding a layer of contextual understanding to the environmental data. Such systems go beyond mapping and localization, offering a deeper insight into the environment's characteristics and features. This integration of semantic analysis represents a significant advancement in the field, as referenced in various studies [Xia+20]. To elaborate, semantic VSLAM is a variant of the VSLAM system that not only offers the current platform's position and environmental mapping but also enriches this data with additional semantic information. There are numerous semantic segmentation algorithms available [JK20].



(A)



(B)

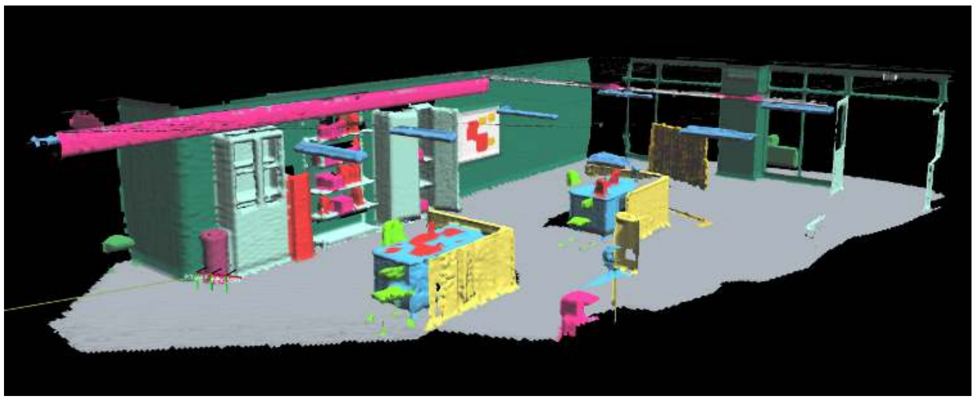


(C)

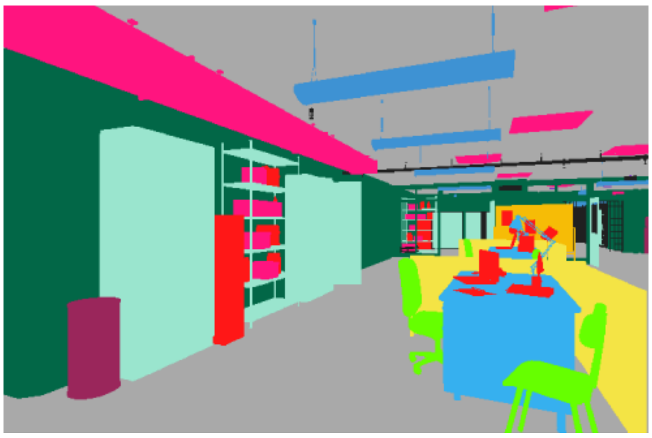
FIGURE 2.26: Example of Loop Closure

Semantic VSLAM offers significant advancements over traditional VSLAM by integrating semantic understanding into the mapping process [Fre23]. This integration allows the system to not only perceive and map the environment geometrically but also to understand and classify different elements within it. One major advantage of semantic VSLAM is improved navigation and interaction capabilities in complex environments. By recognizing and categorizing objects and areas, such as distinguishing between walkable surfaces and obstacles, semantic VSLAM enables autonomous systems, like robots and autonomous vehicles, to make more informed decisions and interact more naturally and safely with their surroundings. This level of understanding is crucial in environments where dynamic changes occur and where interaction with humans and other moving objects is frequent. Another advantage of semantic VSLAM is its robustness in localization accuracy. Traditional VSLAM systems are prone to errors due to dynamic objects and changes in the environment, which can lead to inaccuracies in the map and subsequently in the localization of the robot or vehicle. By identifying and categorizing objects, semantic VSLAM can differentiate between static and dynamic components of the environment, focusing localization efforts on stable landmarks that are less likely to change over time. This capability significantly enhances the system's ability to maintain accurate localization even in environments that undergo frequent changes or contain moving objects, such as busy urban scenes or industrial settings with moving machinery. Furthermore, semantic VSLAM enhances user experiences in augmented reality (AR) applications by enabling more contextually aware AR content. For instance, in an educational or tourist setting, an AR system equipped with semantic VSLAM could recognize and interpret various historical landmarks, art pieces, or natural features, providing users with real-time, interactive information overlays that enhance learning and engagement. This technology can adapt content delivery based on the specific elements within the user's view, such as detailing architectural features on buildings or offering narratives tied to specific locations or objects. By integrating semantic understanding, AR applications can become significantly more interactive and tailored to the user's immediate environment, thus improving the informational depth and engagement level of the AR experience.

Figure 2.27 illustrates the capabilities of the Kimera VSLAM [Ros+20b] system, an advanced approach that integrates VSLAM with semantic segmentation. In Figure 2.27a, a 3D reconstruction of an indoor environment where different objects and structural elements are distinctly color-coded is presented. This semantic segmentation allows for a clear distinction between various types of surfaces and objects, such as walls, furniture, and floor, enhancing the scene's comprehension for both humans and autonomous systems. Figure 2.27b shows image segmentation results.



(A)



(B)

FIGURE 2.27: Kimera VSLAM Semantic Mapping. (a) A 3D semantic reconstruction of an indoor environment showcasing distinct color-coded elements. (b) Detailed view of semantic segmentation within the same environment.

2.7 VSLAM related work

As it was already stated in Section 2.7, SLAM has a very long research tradition. Clearly, at the beginning of the VSLAM, many researchers based on the scientific achievements of the SLAM. In Figure 2.28 an illustrative timeline of the most important systems discussed in this section is presented.

According to [You+17] paper by Younes et al., one of the pionieering papers related to VSLAM is [Dav03]. This foundational paper in the field of Visual SLAM was published by Davison in 2003. This paper marked a substantial shift away from the conventional use of range sensors and laid the essential foundation for subsequent developments in Visual SLAM technology. Four years after, in 2007, Davison et al. introduced MonoSLAM. This system utilizes EKF to estimate the camera pose and map features from a sequence of images. Notably, MonoSLAM continues to operate within a probabilistic framework, a consistent theme in the evolution of VSLAM technologies. Further expanding on this concept, [Cza+20] unified various approaches to VSLAM, incorporating learned priors and addressing the multi-view optimization problem within the probabilistic framework. This approach underscores the ongoing relevance and adaptability of probabilistic methods in the advancement of VSLAM technologies.

In the same year as MonoSLAM release, PTAM was introduced [KM07]. Klein and Murray proposed a new VSLAM architecture, which is used even by the contemporary system. PTAM separates the camera tracking and map building processes into two parallel threads, allowing the system to maintain high accuracy and real-time performance even on standard computing hardware. This separation enables the tracking thread to operate at high frame rates, ensuring smooth camera motion tracking, while the mapping thread focuses on building a detailed, accurate map of the environment. PTAM also incorporates keyframe-based mapping and bundle adjustment, which optimizes the map's structure continuously as new information becomes available. Moreover, it has been demonstrated that VSLAM extends beyond robotics and can be successfully applied in AR applications. Further research, including [LZB16], also shows that VSLAM is still a good approach for AR.

In 2014, the innovative LSD-SLAM algorithm was introduced [ESC14], marked by its direct method approach that bypasses traditional keypoints in favor of using image intensities. LSD-SLAM is capable of creating detailed, semi-dense maps on a large scale without requiring explicit feature extraction or stereo vision. It tracks the camera's pose and builds a depth map by minimizing photometric error, a method that allows it to function effectively in environments with minimal texture, enhancing its versatility in visual SLAM applications. This photometric error minimization technique is also a fundamental component of DSO [EKC18]. Additionally, researchers from TUM,

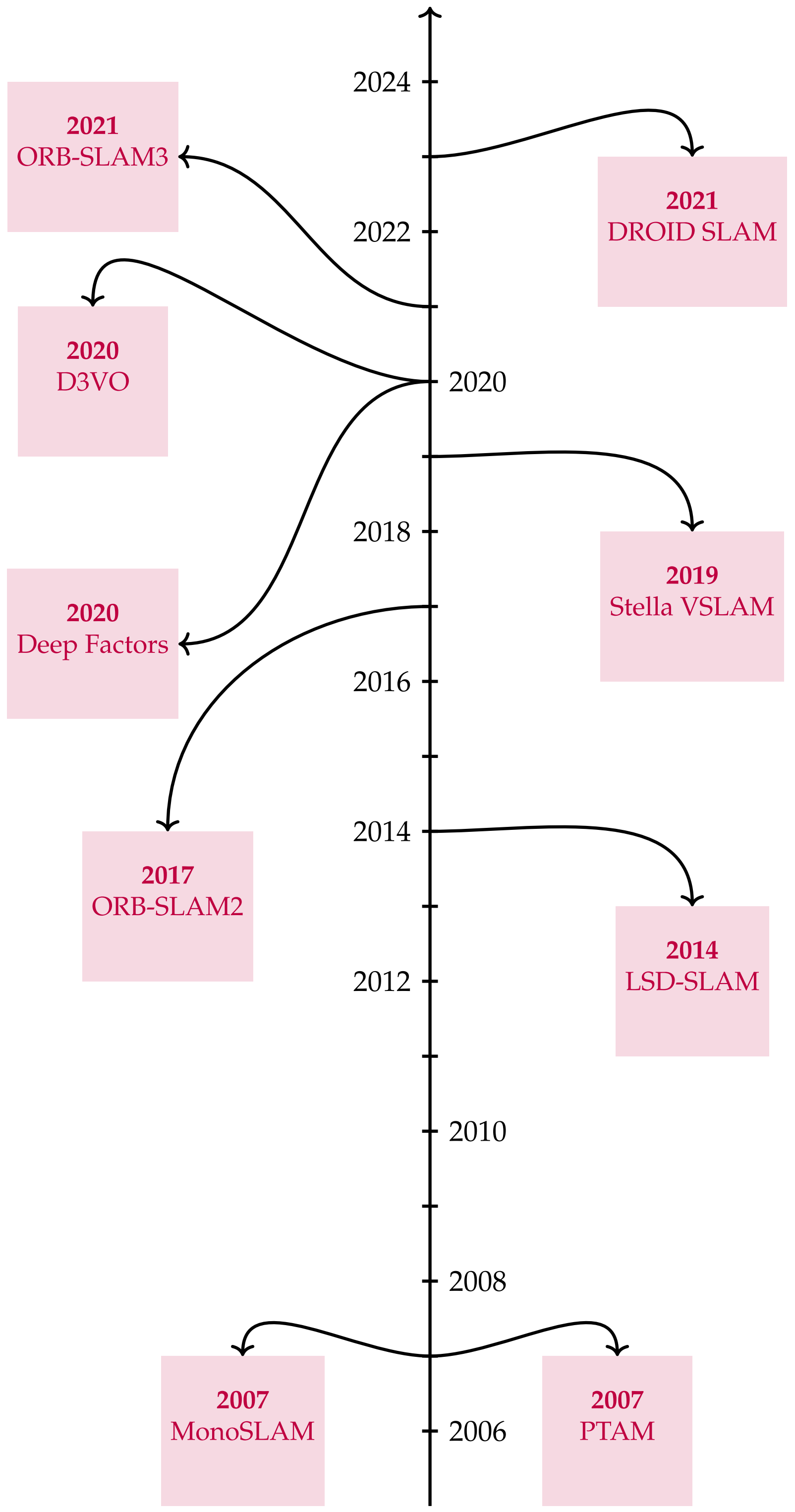


FIGURE 2.28: Timeline of VSLAM with the most important VSLAM systems.

who are highly active in the VSLAM community, have developed DM-VIO, which builds upon DSO by integrating deep learning to enhance motion estimation and 3D mapping through the combined use of visual and inertial data [SC22].

Interestingly, direct methods was not a direction which all researchers has followed. The great example is ORB-SLAM. Developed in three iterations i.e. ORB-SLAM, ORB-SLAM2, and ORB-SLAM3, represents a significant evolution in the field of Visual SLAM [MMT15; MT17; Cam+21]. These algorithms utilize ORB features, which are highly efficient for real-time operation. ORB-SLAM1 laid the groundwork by providing robust monocular SLAM, while ORB-SLAM2 expanded capabilities to include stereo and RGB-D cameras, enhancing its versatility and accuracy in 3D space reconstruction. The latest, ORB-SLAM3, further advances this technology by incorporating multi-map SLAM and the ability to handle visual-inertial datasets, significantly improving its robustness and utility in varied and challenging environments.

StellaVSLAM [SSS19] is a modern adaptation and extension of the well-established ORB-SLAM framework, designed to enhance and expand its capabilities. Like ORB-SLAM, StellaVSLAM employs ORB features for feature detection and description, maintaining the efficient and robust characteristics that is typical for ORB-SLAM family. StellaVSLAM builds on the foundational principles of ORB-SLAM but introduces several significant improvements. For instance, it offers better support for various camera models and configurations, such as fisheye and wide-angle lenses, which broadens its applicability in scenarios where expansive field-of-view capture is crucial. Moreover, StellaVSLAM offers a good code quality, which is crucial for the beginners to understand the VSLAM architecture and algorithms behind it.

Researchers have explored various innovative methods to refine the classical approach to SLAM. [SSP19] introduces BAD-SLAM, a novel implementation featuring a fast, direct bundle adjustment formulation within a real-time dense RGB-D SLAM algorithm. The paper also addresses the vulnerability of direct RGB-D SLAM systems to issues such as rolling shutter effects, sensor synchronization, and calibration errors, proposing a new benchmark that utilizes synchronized global shutter RGB and depth cameras. Additionally, [Gom+19] presented PL-SLAM, a specialized version of SLAM that improves upon traditional feature-based methods by integrating both point features and line segments into the mapping process, thereby enhancing the robustness and accuracy of the mapping in diverse environments.

Since 2018, there has been a significant shift into incorporating machine learning techniques into VSLAM systems. DynaSLAM is an enhancement of ORB-SLAM2 that incorporates dynamic object detection and background inpainting to address the limitations posed by the rigidity assumption in traditional SLAM algorithms. DynaSLAM excels in dynamic environments across monocular, stereo, and RGB-D configurations, utilizing multi-view geometry and deep learning to detect moving objects and inpaint occluded backgrounds, creating accurate static maps of scenes [Bes+18]. Another approach is presented by Fusion++ [Mcc+18]. Mccormac et al. proposed a method with an advanced 3D reconstruction framework that extends the capabilities of traditional scene reconstruction by incorporating semantic segmentation into the process. This integration allows Fusion++ to not only capture geometric details of the environment but also to understand and categorize different elements within the scene, such as distinguishing furniture from walls In [Yan+20] D3VO was proposed. It implemented deep depth, deep pose and deep uncertainty estimation for monocular VSLAM. It is not full machine learning solution, but it rather solves many subproblems of the VSLAM using deep learning. Yang et al. claim that D3VO robustly integrates predicted depth, pose, and uncertainty measurements into a direct visual odometry approach, enhancing both the front-end tracking and the back-end non-linear optimization processes. On the other hand, DROID-SLAM represents another approach [TD21]. Unlike D3VO, DROID-SLAM fully integrates deep learning into the SLAM pipeline. This integration allows DROID-SLAM to handle a wider range of complex environments and dynamic scenarios more effectively than methods like D3VO, which rely more heavily on geometry and traditional optimization techniques. Currently, DROID-SLAM is state-of-the-art system. However, its computations requirements are very high, especially when dealing with training.

In the literature, several methods also capitalize on deep learning to enhance SLAM capabilities. For instance, DeepSLAM represents a monocular SLAM paradigm that leverages unsupervised training with stereo imagery. This system integrates critical components for tasks such as pose estimation, loop detection, mapping, and graph optimization, allowing it to generate pose estimates, depth maps, and outlier rejection masks [LWG21]. Another deep learning-based approach, LIFT-SLAM, introduced by Bruno and Colombini, employs a novel method for extracting features from images, which can lead to a higher number of correct matches due to its enhanced feature extraction capabilities [BC21]. Additionally, research in VSLAM has been effectively applied to Augmented Reality (AR). A notable example is the work by Marchesi et al., who developed a system that integrates ORB-SLAM2 with the Fast-SCNN network [PLC19]. This integration facilitates the creation of a 3D map that is enriched with semantic information, enhancing the environmental understanding necessary for realistic AR applications [Mar+21].

Due to progress of machine learning in rendering, especially methods based on NERF [Mil+20], gaussian splatting [Ker+23] and tools like Nerf Studio[Tan+23], new VSLAM involving these algorithms and classic VSLAM systems like ORB-SLAM in mapping process were proposed. [Chu+22] and

[RLC23] use Nerf. In algorithms proposed in [Mat+23], authors employed gaussian splatting. These cutting-edge techniques enable the creation of continuous, high-fidelity 3D envrionments from standard camera inputs, facilitating more immersive and visually accurate mappings. By combining the precise localization capabilities of systems like ORB-SLAM with the dense, photorealistic rendering offered by NeRF or gaussian splatting, these hybrid VSLAM systems set new standards for accuracy and detail in real-time spatial mapping applications. However, the integration of machine learning rendering techniques into VSLAM systems also brings with it several disadvantages. One major concern is the potential degradation in performance, particularly in terms of computational efficiency. These technique are computationally intensive, requiring significant processing power and potentially leading to slower frame rates in real-time applications. This can be a substantial drawback in scenarios where speed and low latency are critical. Moreover, the usefulness of such sophisticated rendering methods in practical VSLAM applications can sometimes be questioned. While they offer enhanced visual fidelity, the complexity and resource demands of implementing these methods may not always justify the incremental improvements in map.

The literature on VO and VSLAM is extensive, as demonstrated by Table 2.5 which summarizes over twenty papers on these topics, providing a rich source of knowledge. While this table captures a significant portion of research, there are numerous additional surveys and overviews available that delve deeper into various aspects of these fields.

The first significant paper by [SF11] introduces the basics of VO, including fundamental algorithms and terminology, serving as an excellent starting point for newcomers. [Sae+15] explores the specific challenges of multi-agent SLAM, discussing existing solutions and their efficacy. [Car+15] evaluates methods for estimating 3D rotation, providing insights into their practical applications. For those new to the field, [YBH15] offers a broad introduction to robot localization and mapping, covering everything from simple techniques like wheel odometry to more complex VO and SLAM systems.

In [TUI17] progress in the VSLAM community from 2010 to 2016 is detailed, while [Cad+16] serves as both a tutorial and a critique, posing critical questions about the necessity and completion of SLAM. The incorporation of deep learning into VSLAM is reviewed by [Dua+19], highlighting the substantial advancements these models have contributed to computer vision. Challenges and future directions in VSLAM are the focus of [LWG18], and [SMT18] identifies key challenges in dynamic environments, offering a comprehensive taxonomy of current methods and discussing their practicality and robustness.

[DV20] reviews VSLAM algorithms that integrate camera and LIDAR technologies, while [Azz+20] concentrates on feature-based approaches. Modern VSLAM algorithms are compared by [MM21]. [Xia+20] provides a review of the latest developments in semantic VSLAM. In the context of mobile and embedded devices, [YI21] evaluates power consumption, a critical consideration due to the battery dependency of these platforms. The role of deep learning in VSLAM is further examined by [Li+22b], and the evolution of VSLAM technologies, including traditional and semantic approaches, is reviewed by [Che+22a]. Looking forward, [Tou+22] speculates on future advancements in VSLAM. [SSM23] provides an extensive review focused on long-term SLAM challenges for mobile robots, and the importance of keyframe selection is emphasized by [DLD23], who seek to catalog and evaluate various selection methods. Lastly, [DWW23] analyzes common methods, architectures, and techniques employed in VSLAM systems, providing a comprehensive overview of the field.

This overview has traced the history and development of both SLAM and VSLAM, examining key advancements and various algorithms that have shaped its evolution. The next chapter will transition into a focused discussion on Modular SLAM. It details the specific contributions of this thesis, exploring innovative approaches and enhancements that have been developed to further advance the capabilities of VSLAM systems.

Author(s)	*Topic*	Year
Scaramuzza and Fraundorfer [SF11]	Introduction to VO	2011
Saeedi et al. [Sae+15]	Multi-agent SLAM	2015
Carlone et al. [Car+15]	3D Rotation Estimation Methods	2015
Yousif, Bab-Hadiashar, and Hoseinnezhad [YBH15]	Fundamentals of Robot Localization and Mapping	2015
Cadena et al. [Cad+16]	SLAM: State, Challenges, and Future	2016
Taketomi, Uchiyama, and Ikeda [TUI17]	Progress in VSLAM 2010-2016	2017
Jamiruddin et al. [Jam+18]	RGB-D VSLAM	2018
Duan et al. [Dua+19]	Deep Learning Applications in VSLAM	2019
Li, Wang, and Gu [LWG18]	Challenges and Opportunities in VSLAM	2018
Saputra, Markham, and Trigoni [SMT18]	VSLAM in Dynamic Environments	2018
Debeunne and Vivet [DV20]	VSLAM Algorithms using Cameras and LIDARs	2020
Azzam et al. [Azz+20]	Feature-based VSLAM Approaches	2020
Xia et al. [Xia+20]	Recent Advances in VSLAM	2020
Merzlyakov and Macenski [MM21]	Comparison of Modern VSLAM Algorithms	2021
Li et al. [Li+22b]	Deep Learning in VSLAM	2022
Chen et al. [Che+22a]	Evolution of VSLAM Technologies	2022
Tourani et al. [Tou+22]	Trends in VSLAM	2022
Sousa, Sobreira, and Moreira [SSM23]	Long-term SLAM Problems	2023
Dias, Laureano, and Da Costa [DLD23]	Importance of Keyframe Selection in SLAM	2023
Dai, Wu, and Wang [DWW23]	Common Methods and Techniques in VSLAM	2023

TABLE 2.5: Overview of key surveys and papers in VSLAM and VO. It lists prominent authors and their contributions to the fields, focusing on specific aspects and advancements in VO and VSLAM.

Chapter 3

VSLAM challenges and robustness

Overview

This chapter evaluates the ongoing challenges in VSLAM and positions the research within this thesis in the broader VSLAM landscape. It addresses a range of issues such as the distribution of keypoints, mapping in environments that change over time, maintaining the map's accuracy, ensuring the system's real-time responsiveness, managing outliers, and more. This discussion is an introduction for the subsequent two chapters, which are focused on proposing solutions to these identified problems.

Next, a detailed exploration of the concept of robustness within the context of VSLAM is undertaken. It is initiated with a comprehensive definition of robustness, which is constructed through the integration of three critical aspects: methods of computer vision, software reliability, and execution performance. This approach, characterized by its multifaceted nature, improves understanding of robustness, thereby setting a foundation for an in-depth examination of the combined impact of these elements on the effectiveness and reliability of VSLAM systems and their further development.

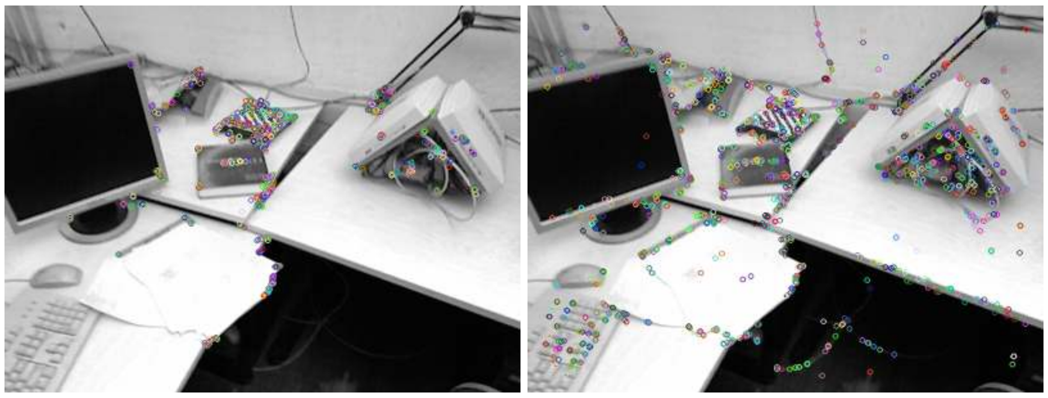
You cannot teach a man anything, you can only help him discover it in himself.
– Galileo Galilei

In the previous chapter, the fundamentals of the VSLAM problem were introduced, with not only a definition provided but also a historical overview of its development, including key breakthrough systems. This exploration has highlighted the various problems that have had to be navigated and the challenges that have emerged over time. This chapter aims to delve into several significant difficulties that must be considered when VSLAM systems are being designed including outliers handling, accumulative error in loop scenarios, map maintenance and more. Moreover, it provides a foundational context for the innovative methods and systems introduced in Chapter 5 and Chapter 4, setting the stage for a comprehensive understanding of their development and the rationale behind their design.

3.1 Keypoints distribution on image

One of the initial tasks performed by the VSLAM systems's frontend, which relies on keypoints, is to determine the position of these keypoints across the image. In other words, one of the frontend's resposibility is keypoint detection. As discussed in Chapter 2, there exists a wide array of keypoint detectors. However, it is noteworthy that these detectors generally do not consider the spatial distribution of keypoints during the detection process. The distribution of keypoints across the image could potentially enhance pose estimation accuracy. This is particularly evident in scenarios where a single part of the image contains a high density of details. To visualize how the output of two detectors may differ, two images with marked detected keypoints are presented in Figure 3.1. In Figure 3.1a keypoints are grouped into several parts. Those parts characterize with highly changing texture. It can be observed that several parts of the image contains significantly more keypoints than the others. On the other hand, Figure 3.1b presents results of the detector which is aware of distribution and it tries to spread keypoints over the image.

To ensure that the mentioned assumption is indeed true, tests were conducted that involved comparing the estimated trajectory in two cases. In the first case, the default detector for StellaVSLAM was used. In the second test, the source code of StellaVSLAM was modified in such a way as to use standard detector from OpenCV library [Ope24]. The comparative results presented in Table 3.1 demonstrate that the distribution of keypoints significantly enhances performance across all evaluated metrics. Both APE and RPE metrics indicate that the distributed keypoints method achieves reductions in RMSE, Mean, Median, and Sum of Squared Errors (SSE). Moreover, it leads to lower minimum and maximum values, underscoring the effectiveness of distributed keypoints in improving accuracy. Furthermore, Figure 3.2 illustrates the trajectories of two systems, one employing keypoint distribution and the other not, in comparison to a reference trajectory. It is evident



(A) Standard detector (B) Distribution-aware detector

FIGURE 3.1: Comparison of keypoint detection techniques: distributed vs. non-distributed approaches, with detected keypoints marked on the image.

that several segments of the trajectories deviate from the reference. These discrepancies are consistent with the findings from the APE and RPE metrics.

APE

System	*RMSE*	*Mean*	*Median*	*Std*	*Min*	*Max*	*SSE*
Distributed	0.01641	0.01386	0.01137	0.00879	0.00071	0.06394	0.16077
Standard	0.02521	0.02104	0.01714	0.01389	0.00272	0.07933	0.37942

RPE

System	*RMSE*	*Mean*	*Median*	*Std*	*Min*	*Max*	*SSE*
Distributed	0.00987	0.00793	0.00625	0.00588	0.00041	0.0429	0.0581
Standard	0.01139	0.00942	0.00749	0.00641	0.00070	0.0375	0.0773

TABLE 3.1: StellaVSLAM performance using distributed keypoints detector compared to classic keypoint detector.

Along with enhanced localization accuracy, distributed detectors offer additional benefits. In keyframe-based systems, the number of keyframes significantly impacts map size and execution time. A better distribution of keypoints across the image greatly influences landmark creation. A prevalent criterion for initiating new keyframe creation is tied to the number of landmarks tracked in the current frame. If this number falls below a specified threshold, then the creation of a new keyframe is triggered. Consequently,

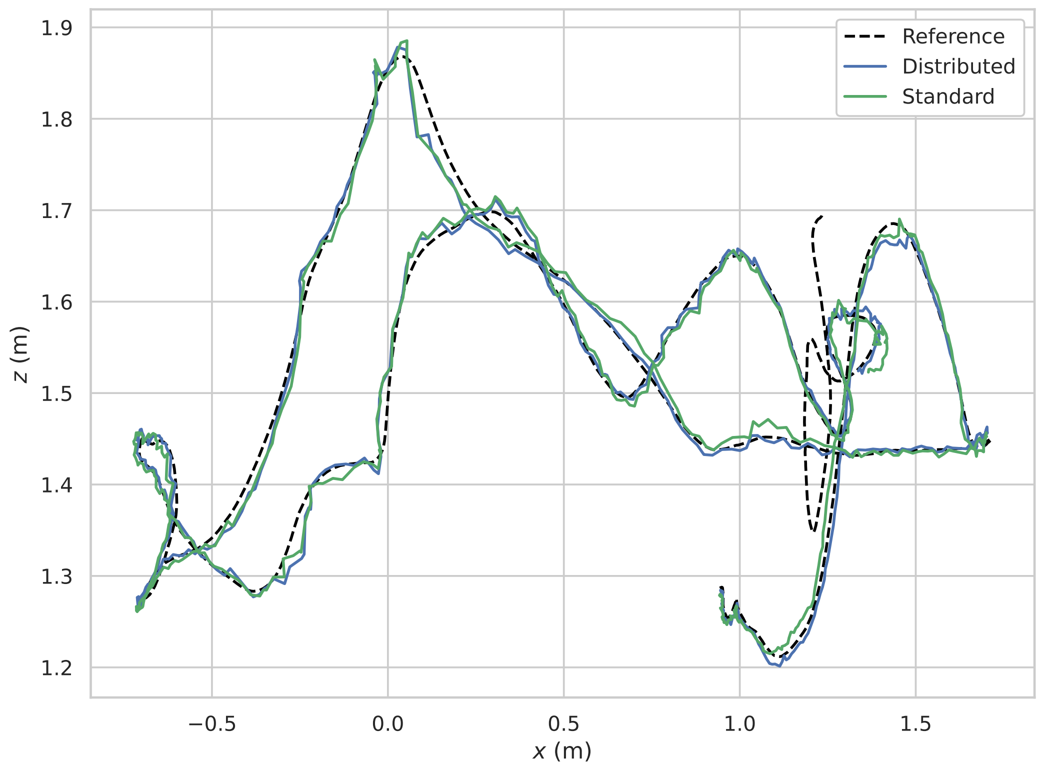


FIGURE 3.2: Comparison of two systems with and without keypoints distribution technique with the ground-truth.

if the keypoints—and, by extension, the landmarks—are well-distributed, it may lead to less frequent generation of new keyframes.

3.2 Mapping in dynamic environments

Dynamic environments are characterized by the presence of moving objects or environmental changes occurring within relatively short periods of time. This presents a challenge for VSLAM systems, which must adapt to these environmental changes. However, a core assumption underpinning every SLAM system is the constancy of the map over time. This assumption significantly affects the system's operational effectiveness, as it presupposes a static environment without dynamic changes or moving objects. Essentially, the principle that ensures the accurate functioning of SLAM systems inherently requires the absence of dynamic elements within the scene. What's interesting, according to Ballester et al., most of the SLAM systems, event the state-of-the-art methods still assume static environment [Bal+21].

Let recall the definition of the VSLAM problem. For each landmark, denoted as $\mathbf{l}_i$, there exists a set of measurements Z_i associated with that specific landmark $\mathbf{l}_i$, along with the keyframes during which the landmark was observed. The position of the landmark is represented by a three-dimensional vector in Euclidean space. The error term $e(\mathbf{l}_i, \mathbf{z}_{i,j})$ for $\mathbf{l}_i$, utilized in the optimization step of the backend, is detailed in Equation 3.1. This equation underscores the previously mentioned assumption i.e. the position of the landmark is considered invariant across all observations.

$$e(\mathbf{l}_i, \mathbf{z}_{i,j}) = \sum (\mathbf{l}_i - \mathbf{z}_{i,j})^2 \tag{3.1}$$

Figure 3.3 showcases a sequence from the Bonn dataset. This sequence captures a laboratory scene through an RGB-D camera, where two individuals are observed walking while the camera remains stationary. In scenarios like this, landmarks associated with moving objects can significantly impact camera pose estimation. A VSLAM system that is robust against moving objects must be capable of recognizing such movements, ensuring that the estimated camera pose remains unaffected and stable despite the dynamic elements in the environment.

There are multiple strategies to mitigate this challenge. The first method retains the original definition of the VSLAM system. Systems like StellaVSLAM or ORBSLAM 3 treat landmarks associated with moving objects as typical outliers. These systems remove landmarks if they detect that the observations of the landmarks are irrelevant. Secondly, a more sophisticated method incorporates moving object detectors within the VSLAM framework. This technique actively discriminates between static and dynamic landmarks. Then landmarks associated with moving objects are filtered out and only static landmarks are used for both localization and mapping. Lastly, the most sophisticated solution involves redefining the VSLAM problem itself to accommodate the modeling of movement within its framework. This approach aims to integrate dynamic object movement into the core algorithm, allowing for a more robust understanding and representation of the environment.

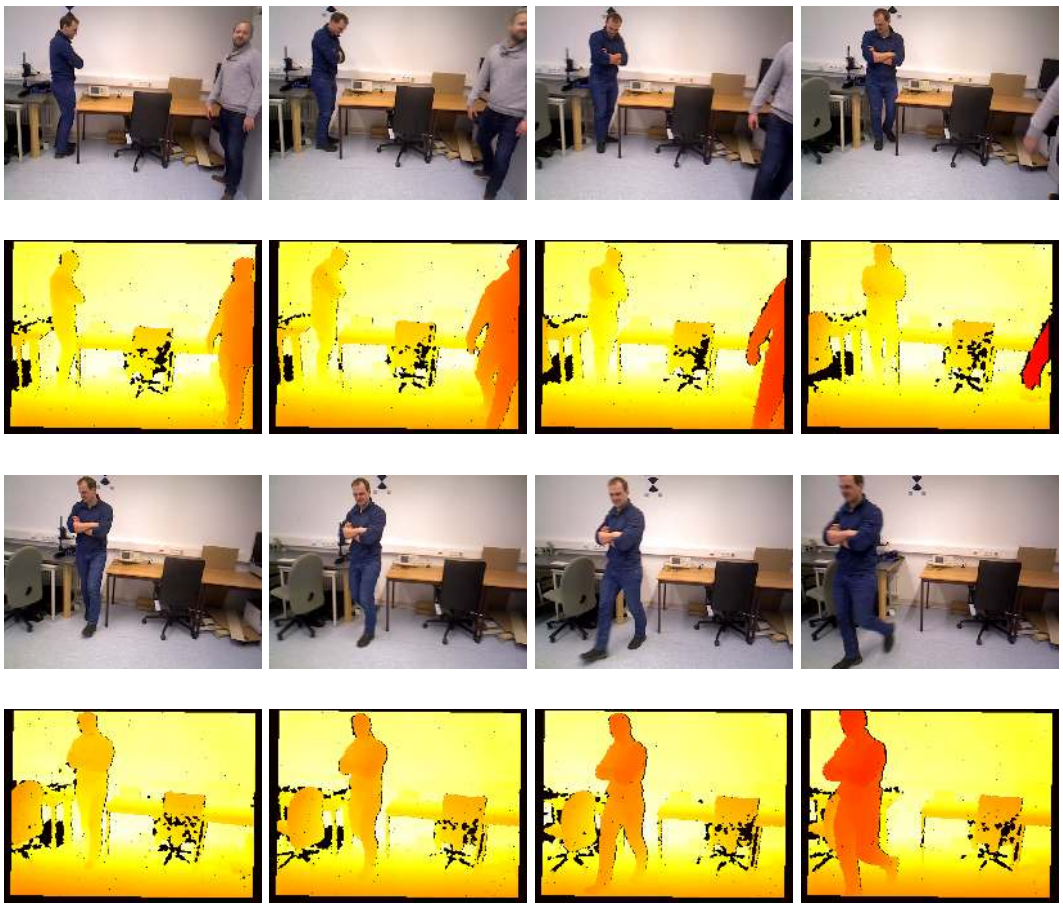


FIGURE 3.3: Example of scene containing moving objects – humans walking in the laboratory.

3.3 Long-Term operation

The primary objective of a VSLAM system is to achieve accurate localization and mapping. However, achieving these capabilities requires careful implementation, particularly with respect to the system's long-term operation. Key challenges to consider for sustained functionality include managing large map sizes and memory management efficiency, reducing the time required for relocalization, accommodating changes in the environment or drift accumulation.

Figure 3.4 visualizes a map generated using Stella VSLAM and data from a KITTI sequence. This map represents a relatively large area, covering approximately 0.25, km^2. It contains 50,087 landmarks and 704 keyframes. Storing all the landmarks, keyframes, keypoints, and observations requires approximately 201MB of disk space. While this file size is manageable for PCs and powerful embedded devices like Nvidia Jetson, it may pose challenges for less capable devices such as Raspberry Pi.

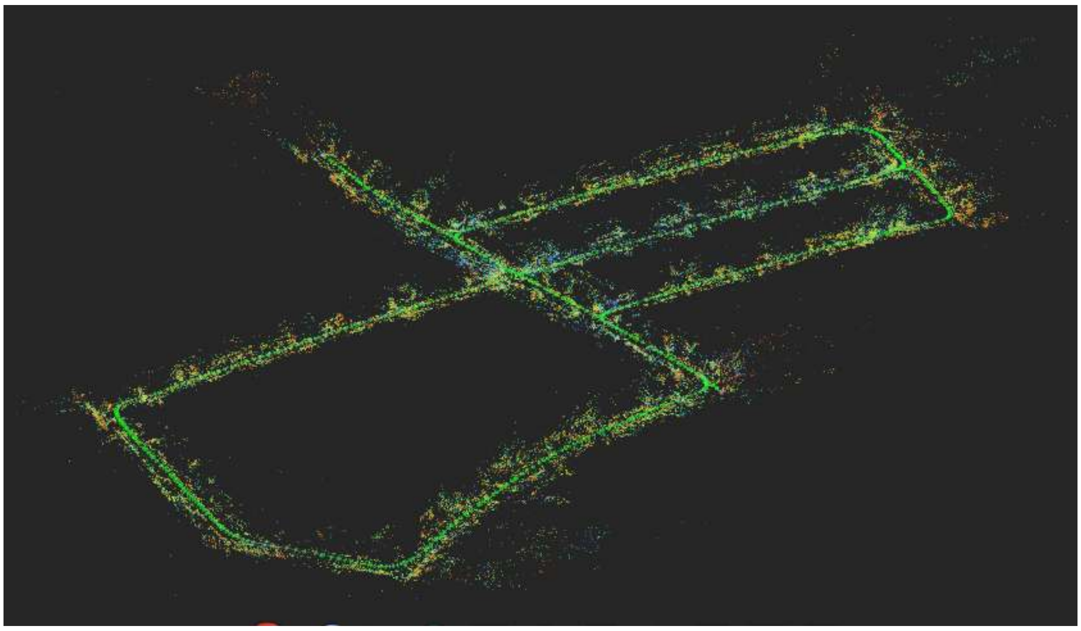


FIGURE 3.4: Visualization of a section of a city mapped using data from the KITTI dataset.

Furthermore, aside from disk space concerns, it's important to note that a significant portion of the required data is stored in RAM during the operation of the VSLAM system. Such constraints have a significant impact on the entire system. For example, a Raspberry Pi typically has only 512MB of available RAM. Rough calculations suggest that the device can store data for an area of approximately 0.5, km^2, assuming no other processes are running concurrently on the machine. Another issue related to long-term operation and hardware resources is the relocalization time. With a high number of keyframes, the relocalization process, typically a two-step procedure involving selecting keyframe candidates and estimating poses, takes considerably more time when dealing with large maps. On the other hand, usually relocalization is a component of the tracking system, which has real-time constraints.

The Table 3.2 provides a detailed comparison of various KITTI sequences by detailing the number of keyframes and landmarks created in StellaVSLAM, along with the mapped areas in square kilometers for each sequence.

A notable observation is that larger mapped areas tend to correspond with increased numbers of keyframes and landmarks. However, the KITTI sequences do not always require a high level of detail, unlike situations where cameras are positioned close to the objects being mapped before moving to other fragments of the environment. In such detailed scenarios, the count of keyframes and landmarks can escalate even further. This implies that mapping larger areas, especially those densely packed with landmarks, poses a significant challenge to real-time processing. It highlights the critical need for sophisticated algorithms and robust hardware designed to efficiently manage large datasets without compromising performance.

Sequence	***Keyframes***	***Landmarks***	***Area*** [$\mathbf{km^2}$]
00	1254	92060	0.27962
01	—	—	2.11580
02	1733	138189	0.56592
03	170	11887	0.09363
04	99	6317	0.00022
05	705	50196	0.20420
06	371	24189	0.01046
07	290	22439	0.04008
08	1371	101825	0.31546
09	628	46131	0.26395
10	401	28725	0.11871

TABLE 3.2: Mapped area of KITTI sequences

Long-term operation also encompasses variations in lighting conditions or scene appearance, which can result from changes in weather the time of day or season. It is crucial to ensure that the algorithm can effectively handle these diverse conditions, as they can significantly alter the visual appearance of the scene. Adapting to such changes ensures the robustness and reliability of the algorithm across different environments and scenarios. In Figure 3.5, four distinct images depict the same location under varying conditions. These visuals underscore how different times of the day and seasonal changes can drastically alter the appearance of a scene. For example, the absence of leaves on trees during winter significantly impacts visual features. This becomes particularly relevant when considering Figure 2.22, which highlights that many keypoints are detected on trees. Furthermore, the time of day plays a crucial role due to variations in lighting, such as changes in shadow intensity and overall luminosity, affecting the system's ability to accurately detect and track keypoints. The issue of lighting appears to be particularly relevant to dense systems that utilize photogrammetric error.

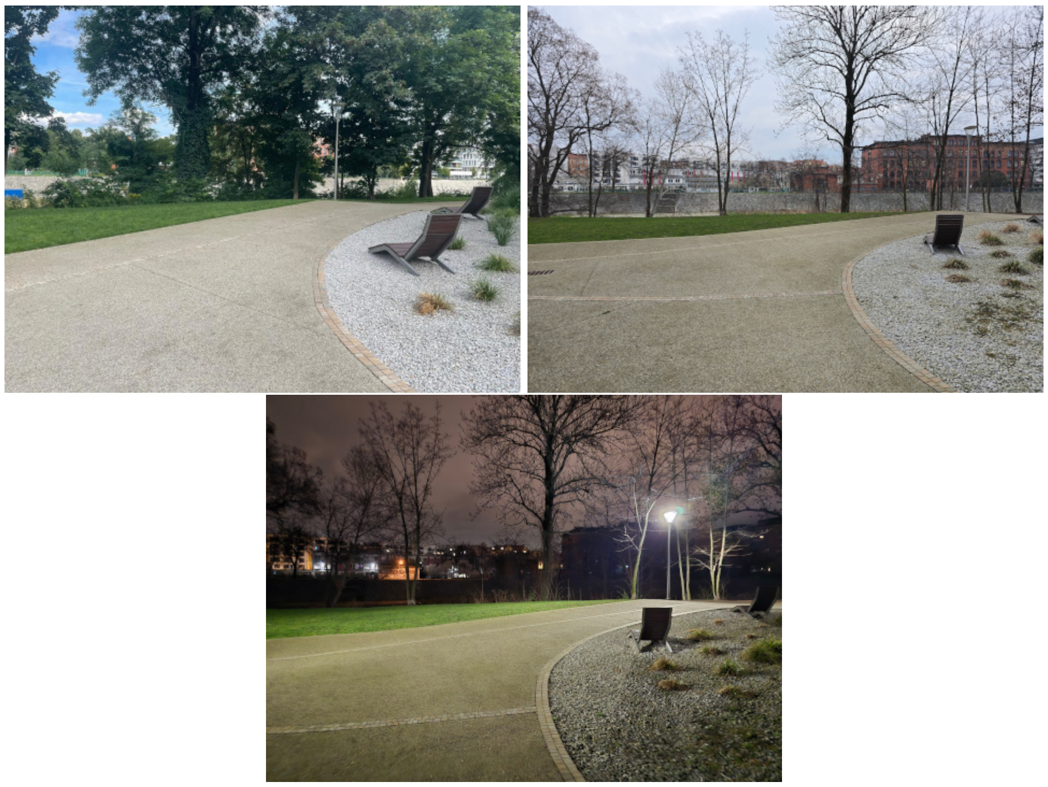


FIGURE 3.5: Different conditions of the same scene.

In the context of long-term operation, accumulative drift emerges as a critical challenge for VSLAM systems. Effectively compensating for this drift is essential for ensuring accurate performance over extended periods. Loop closure is a pivotal technique for error compensation, thoroughly discussed in Section 2.6.4 and Section 3.4. This strategy helps the system to recognize previously visited locations, thereby correcting the cumulative error that builds up over time. Illustrations of the APE on the trajectory are provided in Figure 3.6a and Figure 3.6b. The former demonstrates the error in scenarios where loop closure effectively mitigates drift, while the latter depicts scenarios lacking loop closure support, highlighting the significance of this mechanism in maintenance of long-term VSLAM accuracy. The accumulated drift becomes particularly noticeable towards the end of the trajectory. With drift compensation, the system can achieve localization accuracy, maintaining an

APE below 10 meters. Conversely, without compensation, the system's accuracy significantly declines, resulting in an APE of approximately 30 meters

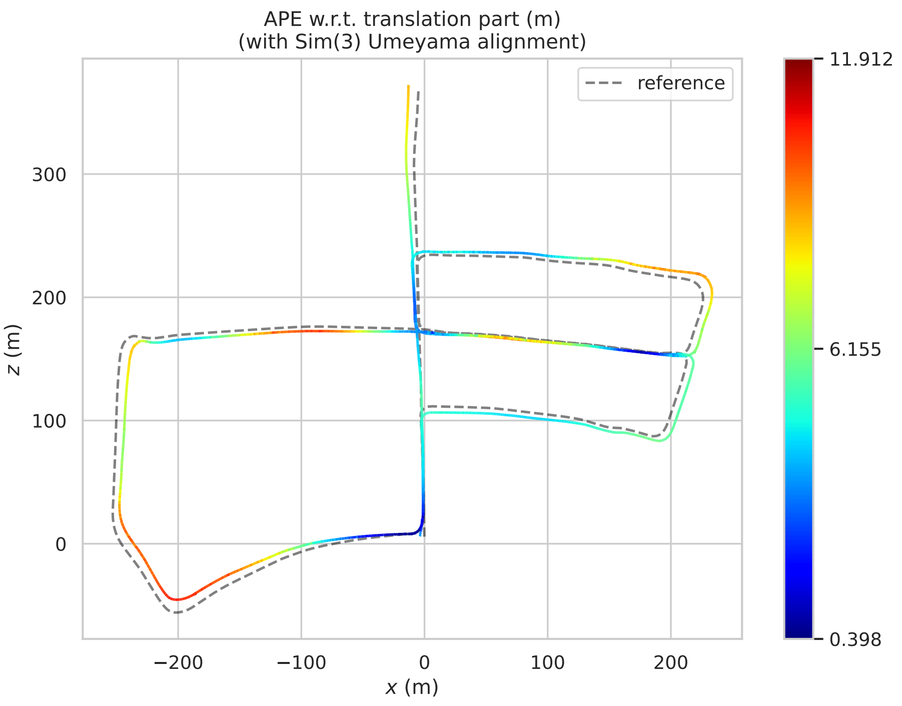
3.4 Loop Closure

Humans do not have problems with recognizing places. It is especially true when there are various lightning conditions or some environmental changes have taken place. To give an example, roadworks frequently occur in urban areas, altering parts of the environment temporarily. Similarly, as previously mentioned, the changing seasons, leading to the shedding of leaves, serve as another example Another example which was already given is change of seasons and consequently leaves fall. Despite these changes, people can still identify the specific location based on the remaining, unchanged features of the environment. Another example was already mentioned in Section 3.3. Due to lightning conditions the environment may change a lot. Figure 3.5 proves that.

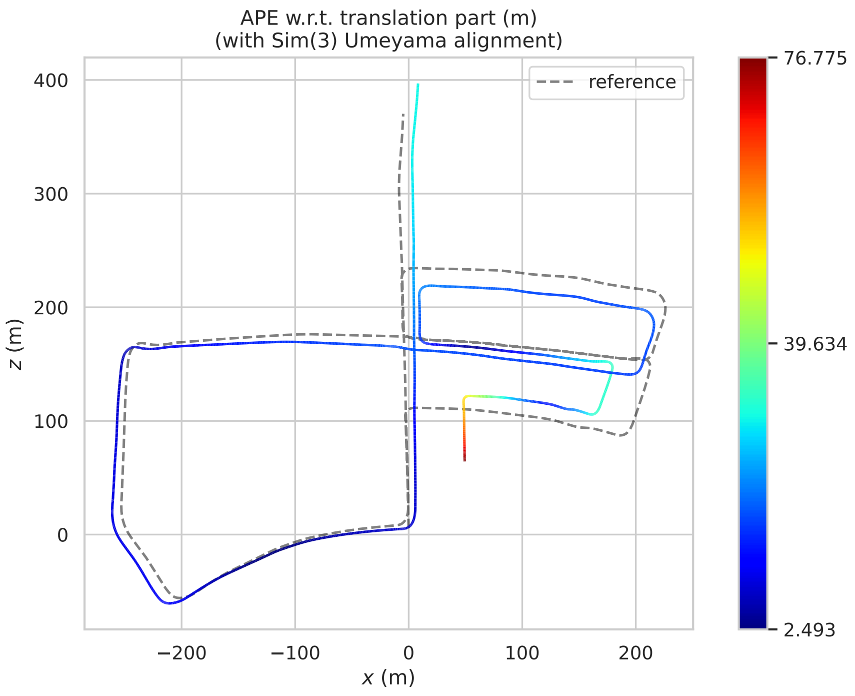
Loop closure is essential for VSLAM systems. The loop closure component, which was previously discussed in Section 2.6.4, is a key feature that sets VSLAM apart from standard VO systems. Therefore, it's crucial for this component to be both reliable and accurate. In the context of loop closure, these two qualities refer to several functions. Initially, loop detection is performed, which from a computer vision standpoint is a typical classification problem. The objective is to determine whether two images —- the current frame and those associated with previously created keyframes -— depict the same location. In this task there are at least several challenges which have to be taken into considerations.

One of the primary issues is the high similarity between different places, which may only differ in subtle details, such as floor numbers in a building. Distinguishing between such nuances requires the system to be highly discerning and sensitive to minor variations. Such scenario is shown in Figure 3.7. This is called *perceptual aliasing*. According to Lajoie et al., it is a phenomenon where different places generate a similar visual and leads to the wrong estimates of the VSLAM system. What is more, perceptual aliasing may be used as an adversarial attack for VSLAM system [Ikr+22].

Additionally, varying lighting conditions can drastically change the appearance of a scene, complicating the process of recognizing the same place across different times of day or weather conditions. Finally, the presence of moving objects can introduce discrepancies between two images of the same location, as these objects may appear in one image and not the other, leading to erroneous classification and wrong loop closures. Addressing these challenges is crucial for the robust performance of the loop detection component in VSLAM systems.

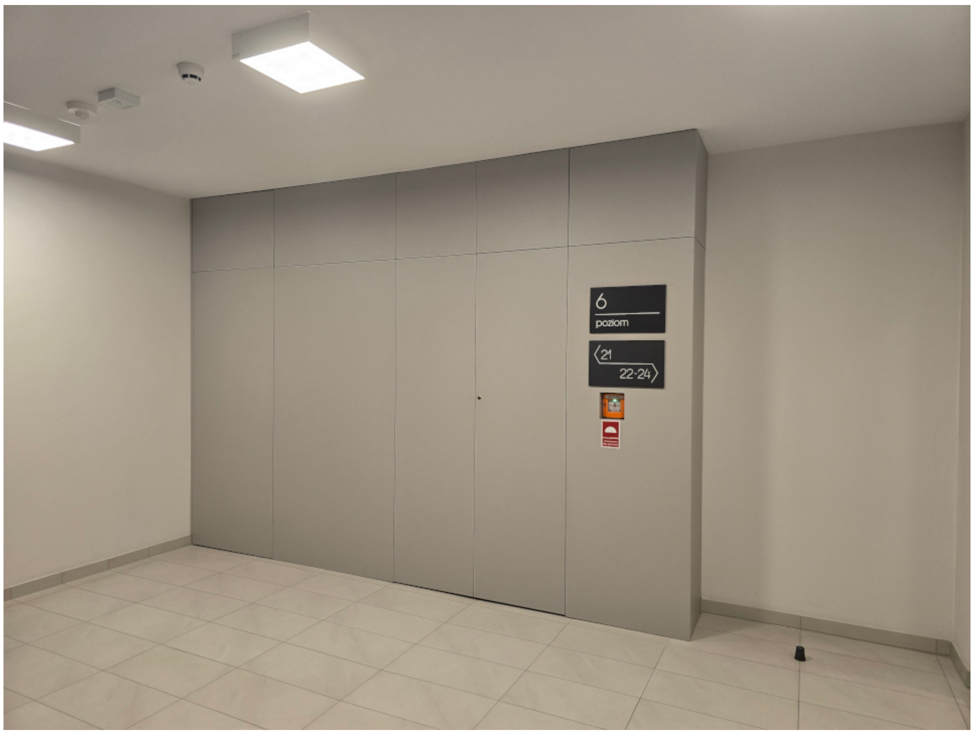


(A) With drift compensation.

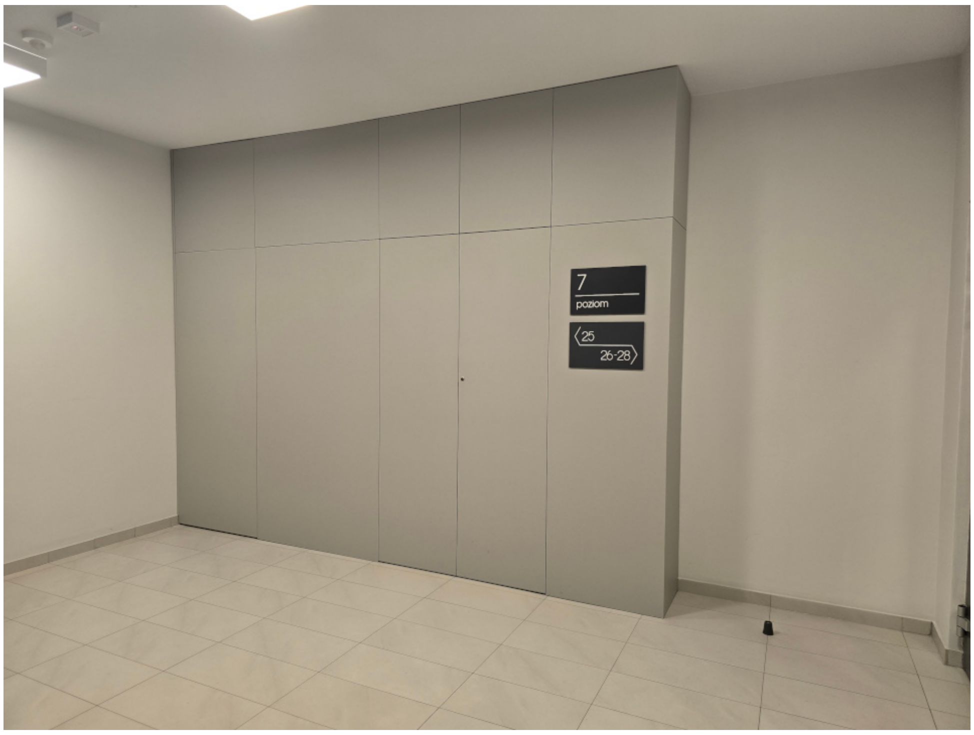


(B) Without drift compensation.

FIGURE 3.6: Comparison of APE for systems with and without drift compensation. The first plot (a) shows the APE for a system implementing drift compensation, while the second one (b) illustrates the APE for a system lacking this feature.



(A)



(B)

FIGURE 3.7: Examples of very similar environments but they represent various locations – perceptual aliasing

Within the framework of loop detection for VSLAM systems, the concepts of reliability and accuracy are pivotal to asses the system's performance. This evaluation is conducted through metrics such as classification accuracy, precision, and recall, which offer a nuanced view of the system's effectiveness. A critical aspect of this analysis involves understanding the system's error rates, encompassing both false positives and false negatives. False positives, or erroneous loop detections, directly impact precision by indicating loops where none exist, while false negatives, which are overlooked actual loops, affect recall by highlighting missed detection opportunities. Thus, a balanced assessment of these errors provides a more complete picture of the system. Keeping both, recall and precision metrics high is a big challenge.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.2}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.3}$$

Following the identification of a potential loop by the loop detection module, the process advances to the crucial stage of loop closure. This step is responsible for finding correspondences to validate the detected loop. It's important to recognize that the loop detection component might suggest multiple keyframe candidates for this purpose. As a consequence, this step not only validates but also chooses the best keyframe candidate. Finally, the keyframe with the best score is chosen and if the conditions are met than the new constraints resulting from loop are created and the map optimization is performed. In this stage, several critical issues need to be tackled. A primary concern is the execution time of the entire process. Given the complexity of this approach, it is essential to develop an efficient algorithm that can complete calculations quickly and update the map in a short period of time to provide updated map for localization module. Another significant challenge is managing the new constraints introduced by loop closure. It is vital to design the system in a way that prevents loop closure from being triggered repeatedly for the same location where a loop has already been identified. Addressing these scenarios within the algorithm can pose a considerable challenge.

To demonstrate the accuracy of the loop detection, a test was performed using the KITTI sequence 00 with the Stella VSLAM system. The Figure 3.8 depicts the system's generated map and trajectory, where the red circles mark the areas that should be identified as loops. These specific locations are places where the system revisits them. Ideally, they should trigger the loop closure mechanism to correct any trajectory drift. However, it was observed that out of all the marked potential loops, the system detected only two as actual loops. This suggests that improvements are possible in the system's ability to detect loop closures reliably.

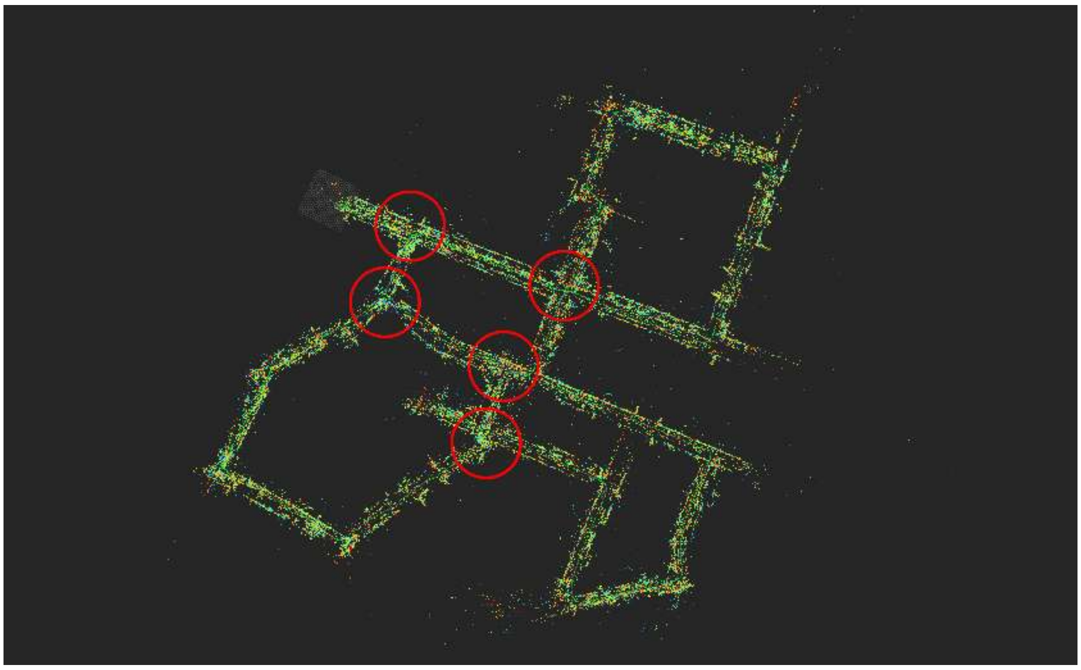


FIGURE 3.8: Map generated using KITTI 00 sequence and StellaVSLAM. The red circles highlight potential loop closure points within the trajectory. Of the marked locations, only two were successfully identified as loops by the system's loop detection algorithm, demonstrating areas for potential improvements in detection.

3.5 Real-time performance

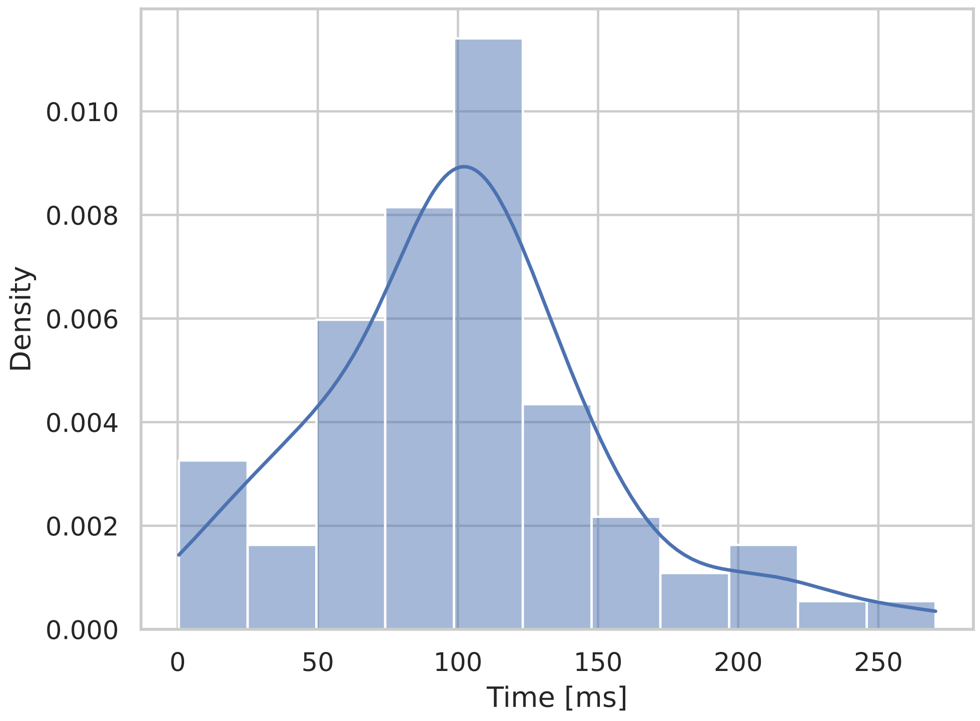
It has been already discussed in Section 2.4 that one of the main difference between SfM and VSLAM problem is the time constraints. In VSLAM, execution time of processing single frame is very limited. For instance, typical camera is able to provide 30 frames per second. It means that VSLAM system has only 33 milliseconds to use the image for localization and mapping purposes before the next frame arrives. Clearly, there are applications where higher throughput is required, and even more frames per second should be handled. Furthermore, resolution of the image has also vital impact on the performance. This is especially a challenge when high resolution images are considered such as FullHD or 4K.

These constraints are the reason why localization and mapping processes are typically separated into two various tasks. In VSLAM systems, the priority is often given to pose estimation over the immediate delivery of the current map. This is because accurate pose estimation is crucial for real-time navigation and interaction with the environment. Therefore, in many VSLAM approaches, pose estimation is integrated into the core pipeline of frame processing. Meanwhile, mapping including keyframe creation and map optimization is performed in parallel. This parallel processing allows for the system to maintain a rapid understanding of its position and orientation, while concurrently building and refining the map without compromising the speed and responsiveness of pose estimation.

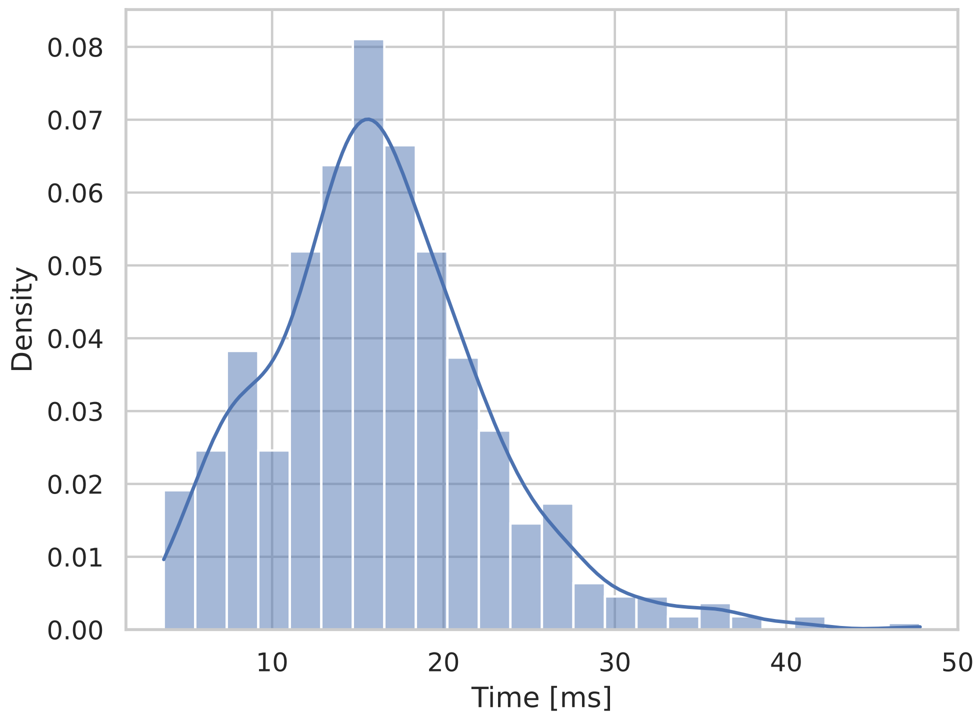
To have an intuition about localization and mapping tasks and their execution time tests were conducted. Stella VSLAM was modified to measure the time of localization and mapping processes. The tests were performed on the KITTI sequence and results are shown in Figure 3.9 and Figure 3.10. The histogram of mapping times shows a broader distribution with a longer tail, indicating that mapping operations typically require more time to complete, with some instances taking upwards of 250 milliseconds. This suggests that the mapping process is computationally intensive especially in comparison to the second plot which depicts localization times. The histogram has a much narrower distribution, concentrated primarily under 30 milliseconds. The peak of this distribution is quite sharp, indicating that localization times are generally consistent and remarkably faster than mapping times. This also reflects the system's prioritization of pose estimation, which is essential for real-time operation and a confirmation that the localization process is significantly faster than mapping.

Moreover, another evaluation was conducted to illustrate the relationship between the number of local landmarks, the number of tracked landmarks, and the tracking time. This is shown in Figure 3.10. It was observed that an increase in the number of local landmarks corresponds to an increase in tracking time, a result that aligns with expectations, given that tracking a greater number of landmarks requires additional computational effort and time. The data points are color-coded based on the number of landmarks currently being tracked. Again, a similar trend is visible: a higher number of currently tracked landmarks has a significant impact on the tracking time.

The conducted tests underscore the necessity for careful consideration of VSLAM performance and the conscious selection of algorithms. The evidence indicates that several critical factors significantly affect system performance. These factors include not only the size of the map consisting of keyframes, landmarks and observations. In addition, factors such as the resolution of the images processed by the system also play a significant role. The balance is crucial for the practical application of VSLAM in various scenarios and this is also a challenging task.



(A)



(B)

FIGURE 3.9: Comparison of tracking and mapping times.

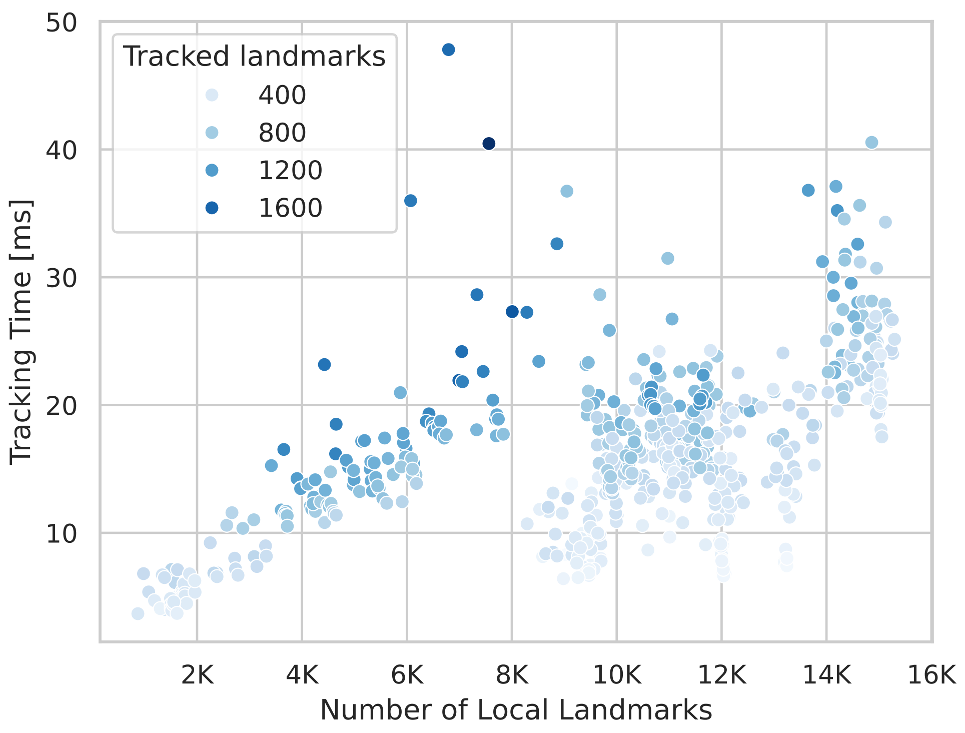


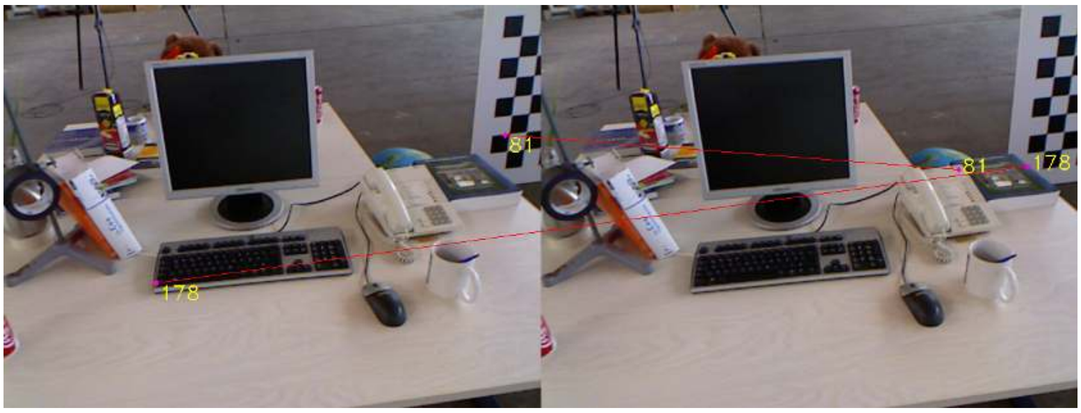
FIGURE 3.10: Relationship between tracking time and number of local landmarks.

3.6 Outliers handling

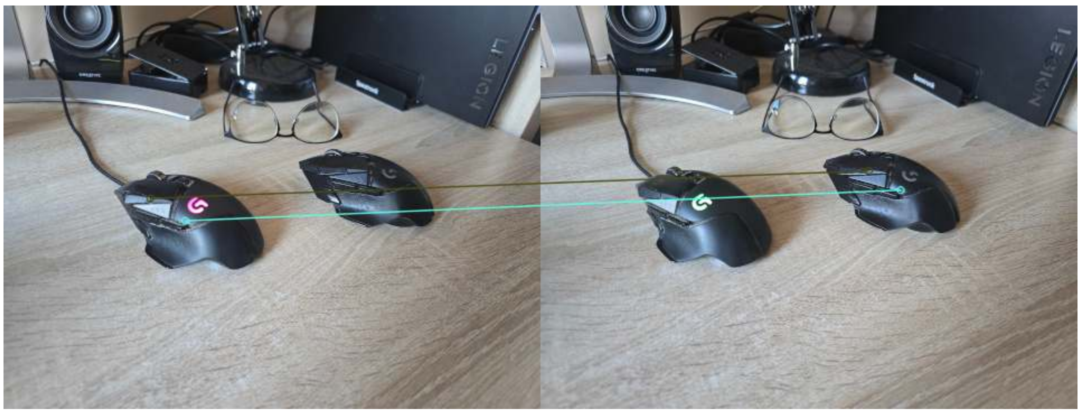
Every system is exposed to various errors, and VSLAM systems are no exception. The foundation of robustness in such systems lies in effectively managing these errors. Outliers in VSLAM may originate from multiple sources, including sensor noise, dynamic objects in the environment, and incorrect feature matching due to repetitive or similar textures. Sensor inaccuracies, especially in challenging lighting or weather conditions, can significantly distort the data, leading to errors in landmarks position estimation. Dynamic objects, such as moving vehicles or pedestrians, introduce changes in the scene that can mislead the tracking process. Furthermore, the algorithm's reliance on feature matching can struggle in environments with high levels of visual ambiguity, causing mismatches and, consequently, inaccuracies in pose estimation and map construction. Handling these outliers efficiently is essential for maintaining the integrity and accuracy of the VSLAM system, supporting the system's ability to explore complex environments reliably.

The example of the significant outliers which may have major influence on pose estimation and consequently also in mapping process are shown in Figure 3.11a. Despite the fact that Lowe's ratio was used to lower the number of bad correspondences [Low04]. This technique is expected to significantly reduce the number of false matches, improving the robustness of feature matching. However, the problem still occurs. Indeed, the image proves that there are still cases where not relevant keypoints are matched. Errors may also emerge from keypoint matches that seem accurate but actually belong to different instances of identical or very similar objects, representing a similar category of matching errors as previously described. This phenomenon is illustrated in Figure 3.11b. While numerous matches accurately identify components of computer mice, complications arise due to the presence of two almost identical mice on the desk. In this scenario, the matches mistakenly associate keypoints with the incorrect instances of the mice

Lastly, Figure 3.11c illustrates a scenario where a mouse has been moved yet is still recognized as being in its origin. This situation leads to errors concerning the 2D-3D correspondences. To clarify, each keypoint is matched with a specific landmark in the environment. When an object like the mouse is relocated, its actual position in the 3D space changes, rendering previous keypoint-to-landmark associations obsolete. Consequently, any observations based on these now-incorrect associations become irrelevant, highlighting the challenges VSLAM systems face with dynamic objects in the environment.



(A) Incorrect keypoint matching


(B) Similar objects in the scene

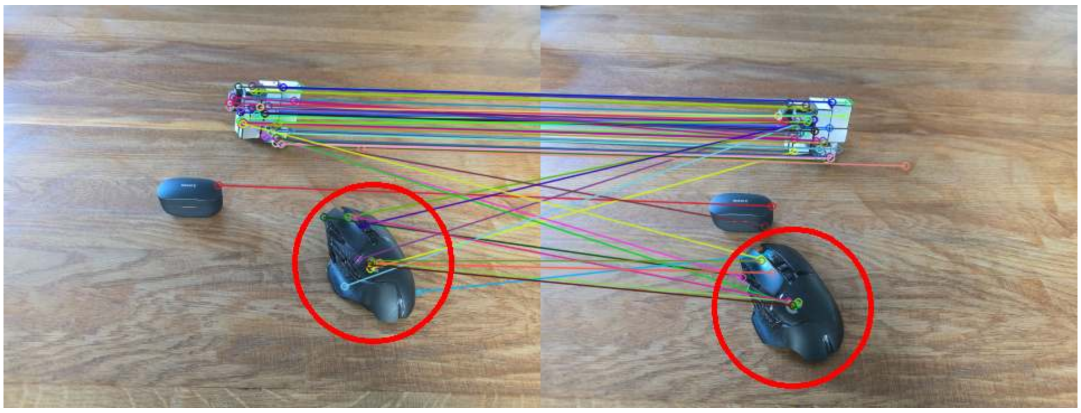

(C) Moved objects

FIGURE 3.11: Examples of outlier sources in VSLAM

3.7 Map maintenance

The map plays a crucial role in VSLAM system, serving as the basis for both localization and mapping tasks. As a consequence, map maintenance emerges as a integral process that every VSLAM system must carefully undertake to guarantee its consistent functionality.

First, let's clarify the concept of map maintenance. It refers to the continuous process of updating and refining the map to accurately represent the environment over time. This task involves addressing dynamic changes in the space, which may manifest as outlier landmarks during optimization processes. Effective map maintenance seeks for a balance between the precision of localization and mapping and the overall size of the map. Consequently, it involves not only the elimination of outliers but also the removal of accurate landmarks and keyframes that are supposed redundant, to optimize both performance and storage efficiency.

The Figure 3.12 illustrates the dynamic nature of map visualization and maintenance in the StellaVSLAM system over time. The Figure 3.12a presents the initial scene captured by the system. The remaining images sequentially demonstrate the progressive development of the map. These stages show the addition of new keyframes, indicated by the green shapes, and the establishment of interconnections, depicted as purple lines. As time passes, the map evolves. Some keyframes and their associated landmarks are deleted to optimize the map's structure and resource utilization, reflecting the ongoing map maintenance process. This frequent addition and deletion of keyframes help maintain an accurate and efficient map, highlighting the system's ability to adapt to changes within the environment and its own internal representations.

3.8 Keyframe creation

In keyframe-based VSLAM systems, keyframes are crucial for constructing the map, with landmarks being generated at the time of keyframe creation. These components are essential for the functionality of the system, underscoring the importance of an effective method for keyframe creation. It's also vital to have well-defined criteria for the introduction of new keyframes to ensure the system operates efficiently. The performance of the system, measured by execution time as well as localization and mapping accuracy, is greatly influenced by the map's size. As previously discussed in Section 3.5 and illustrated by Figure 3.10, the size of the map is a critical factor affecting real-time performance. Furthermore, it has been already shown in Section 3.5 that mapping is much more complex problem than localization.

Due to the reasons mentioned above, it is not feasible for researchers to force the system to generate a large number of keyframes in a short period of

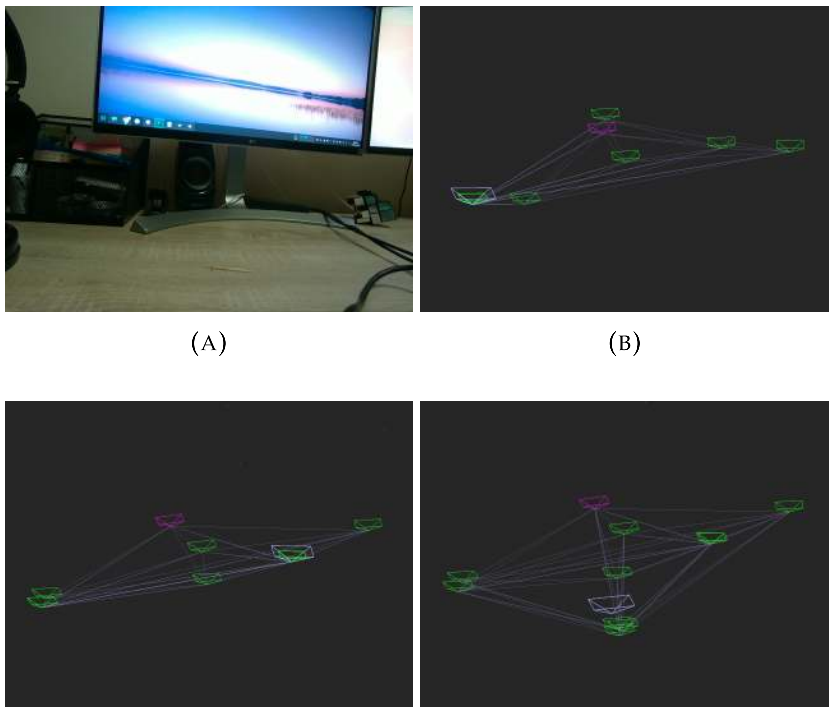

(A) (B) (C) (D)

FIGURE 3.12: Example of map maintenance where keyframes are dynamically added and removed during the mapping process.

time. In scenarios where an excessive number of keyframes are created too quickly, the mapping component which is tasked with creating keyframes and landmarks often queues these requests or, in some instances, ignore them entirely. It's important to remember that in VSLAM systems, the interaction between the localization and mapping components is tightly integrated. Consequently, ignored or postponed keyframe creation requests can lead to localization failures, necessitating a switch to relocalization mode.

In StellaVSLAM system evaluates multiple criteria to decide whether a new keyframe should be created, reflecting the system's adaptability to various conditions for accurate mapping and localization. Key factors like the time elapsed since the last keyframe insertion, the distance traveled, changes in the field of view, and the number of reliable landmarks are carefully assessed. For example, a new keyframe is considered necessary if the elapsed time or traveled distance exceeds predefined thresholds, or if there's a significant change in the scene observed by the current frame compared to the reference keyframe. Additionally, the system's current state, including the stability of tracking and the sufficiency of landmarks, plays a crucial role in this decision-making process. The system avoids keyframe insertion when tracking is unstable or almost all landmarks are reliably tracked, to prevent redundant data that could have impact on the system's efficiency. Furthermore, the presence of a sufficient number of keyframes and the avoidance of local bundle adjustment are also taken into account, balancing the need for new keyframes against the potential for overloading the system. This multi-faceted approach ensures that keyframe creation is both necessary for enhancing the map's accuracy and feasible within the system's operational constraints [SSS19].

On the other hand, such approach to keyframe insertion is not without its criticisms. One significant challenge lies in determining the relative importance of each condition considered by the system. With numerous factors influencing the decision to create a new keyframe the prioritization of these conditions can be complex. Furthermore, the reliance on thresholds to parameterize many of these conditions introduces an additional layer of complexity. These thresholds must be carefully chosen to ensure they accurately reflect the system's needs in various scenarios. However, finding the optimal settings can be challenging, as it requires a delicate balance between sensitivity to environmental changes and the system's operational efficiency. This complexity underscores the need for ongoing research and refinement to enhance the system's decision-making process in keyframe creation, aiming for an optimal balance of accuracy and resource management.

In Figure 3.13 number of tracked landmarks over time is shown. What is interesting about this relationship are peaks in many parts of this plot.

This is related to the mapping process and the reliability of the created landmarks. After mapping module creates the new landmarks, typically localization module is able to track high number of them. However, after short period of time they neither are detected or are not inliers anymore. Consequently, the mapping module is then required to eliminate these short-lived landmarks. This seems to be another challenge for the mapping module to create landmarks that contribute to the global map consistently over time.

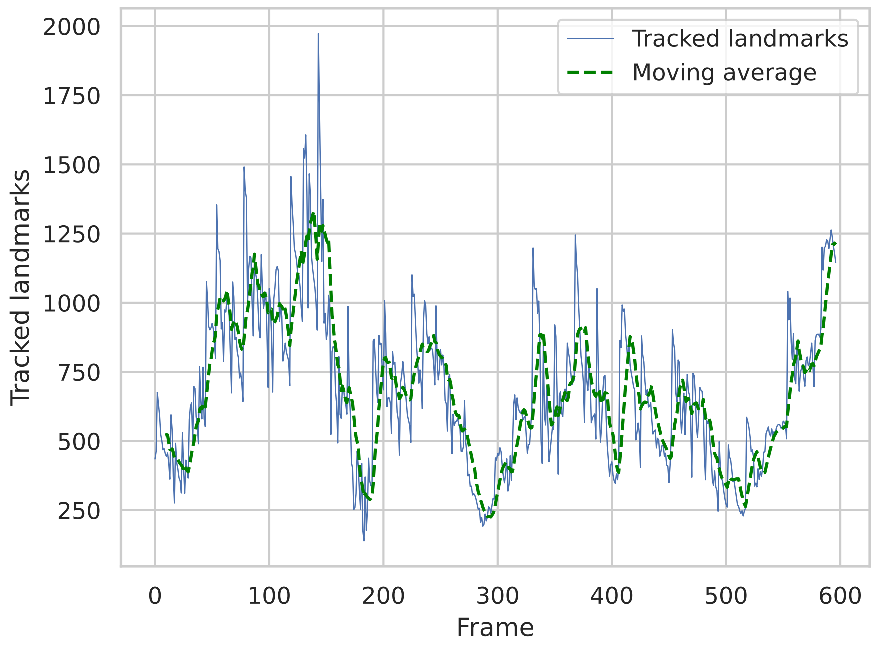


FIGURE 3.13: Tracked landmarks over time with visible peaks related to the finish of keyframe creation.

3.9 Introduction to Robustness

In the preceding chapter, numerous challenges faced by VSLAM systems were explored, revealing a range of issues that highlight the importance of robustness within these systems. The concept of robustness, particularly in the realm of computer vision, is considered essential for the effective functioning of VSLAM. However, it has been noted that the term *robustness* encompasses a broad spectrum of interpretations across various studies and implementations. As highlighted by Drenkow et al., the term robustness in the computer vision context is often regarded as an overloaded term, encompassing a wide array of criteria and expectations [Dre+21].

The broad range of meanings attributed to robustness contributes to the challenges in defining and measuring this concept within VSLAM systems. This situation underscores the necessity for a more refined understanding and methodical approach in tackling the notion of robustness. In light of the ambiguity surrounding this term, the following section will aim to provide a clear definition of robustness in the context of this thesis and VSLAM.

Robustness is a term full of ambiguity and is often overloaded with meanings. Its definition varies significantly across different fields. Especially for VSLAM which is a field where various For instance, what constitutes robustness for scientists working on mobile robots differs remarkably from the interpretation of robustness by those in the field of computer vision. Furthermore, researchers which are interested in AR technology may have completely different criteria of assessing VSLAM robustness than scientists of 3D reconstruction.

The broad spectrum of meanings attributed to robustness prompts a reevaluation of its definition. For the purposes of this thesis, robustness in VSLAM is characterized by three key aspects: algorithms, software, and performance. These aspects are illustrated in Figure 3.14.

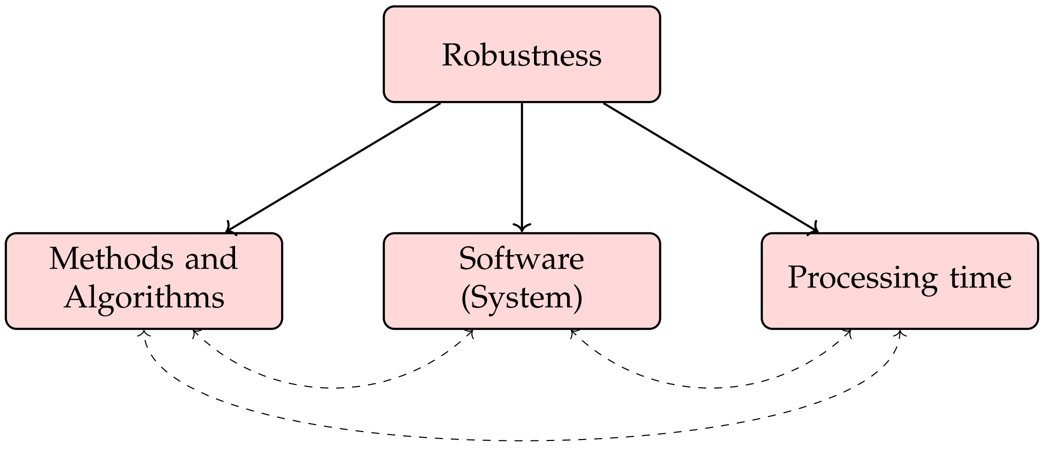


FIGURE 3.14: Diagram illustrating the various aspects of VSLAM robustness examined in this thesis.

The dashed connections at the bottom of the diagram represent the influence among the different components of VSLAM robustness: Methods and Algorithms, Software, and Processing Time. These interconnections suggest that changes in one area can impact the others. For instance, more sophisticated algorithms may improve algorithmic robustness but could also lead to increased processing time. Conversely, enhancements in software efficiency might reduce processing time and also allow for the implementation of more complex algorithms without compromising system performance. Similarly, optimizing processing time can contribute to software stability, making it more robust, which in turn can affect the choice and implementation of methods and algorithms. This interconnections highlight the need for a balanced approach in VSLAM system design, where each component is tuned not only for its direct contribution to robustness but also for its integration with the others.

3.9.1 Software robustness

SLAM research spans almost four decades. Over such a long period, the field has matured significantly, with numerous breakthroughs emerging. The development of VSLAM algorithms particularly intensified during the 2010s. Throughout these years, a multitude of research articles were published, proposing innovative approaches to the VSLAM problem ([Cad+16]). The fact that many new methods were developed around the year 2010 is not surprising, as this was a period of significant change in the approach to VSLAM. Researchers understood that keyframe-based solution is better in terms of scalability, lower computational power and improved robustness.

However, the increased focus on research within the field has not necessarily translated into the number of high quality tools and source code available for the developed systems. This discrepancy poses a significant challenge for newcomers to the field, who encounter a high barrier to entry. There are only a few handful libraries or tools, such as GTSAM [Del12], g2o [Kum+11], ATLAS [Bos+03] and [Gru17], available to assist with specific subproblems in VSLAM. Nonetheless, these tools do not offer an all-encompassing solution to the wide-ranging VSLAM challenge, thus restricting their applicability to more general uses.

In the existing literature, there are a few works that recognize the significance of software, including tools and libraries, in the research process. These contributions, while not numerous, underline the benefits of facilitating rapid experimentation and minimizing the need for redundant implementation of identical methods. For instance, the researchers in [Sem+22] emphasize the advantages of modularity and concurrency for enhancing research efficiency. Similarly, ProSLAM, a VSLAM system discussed in [SCG18], is noted for its emphasis on implementation simplicity. Unlike many studies that concentrate on the mathematical intricacies of VSLAM, Schlegel, Colosi, and Grisetti shift the focus towards the software aspects and the C++ code, highlighting the importance of software engineering principles in the development of VSLAM systems. Moreover, the significance of educational aspects cannot be overlooked. The researchers in [Xu+24] have emphasized that educational platforms play a crucial role in facilitating students' transition into the industrial sector. These platforms not only enhance their awareness but also contribute to the reduction of the time required for training.

ROS is a widely recognized framework within the VSLAM community, offering a comprehensive suite of tools, libraries, and programs primarily for robotics use [Mac+22a]. Despite its popularity, the employment of ROS in VSLAM, which has a broad range of applications across various disciplines, has drawn some criticisms. Firstly, leveraging robotics-focused software for VSLAM might not always align with optimal design principles due to the diverse requirements of VSLAM applications. Secondly, the intricate architecture of ROS can present a significant learning barrier, particularly for individuals new to robotics and VSLAM, potentially prolonging the initial development phase. The framework's general approach may also introduce processing inefficiencies, impacting the speed and responsiveness critical in real-time VSLAM tasks. Moreover, reliance on ROS could restrict the versatility and deployability of VSLAM systems. For instance, deploying ROS-based solutions on mobile devices, such as smartphones, might be challenging due to their operational constraints.

To summarize, the VSLAM community lacks a comprehensive framework specifically tailored to encompass the entire VSLAM research needs. In the context of this thesis, identifying these features is pivotal as they are considered integral to the software robustness aspect of VSLAM. Such a framework should ideally support a wide range of functionalities. Let try to identify what features such framework should have.

In this thesis, it is proposed that multiple key aspects contribute to the robustness of software. These aspects are shown in Figure 3.15, specifically highlighting simplicity, adaptability, modularity, reliability, portability, and maintainability. Subsequent sections will delve into each of these aspects, providing a comprehensive description of their significance and impact on software robustness.

Simplicity

Particularly, recent systems suffer from a lack of clarity because they introduce many components that are tightly coupled. Yet, it is feasible to create advanced and expansive systems that remain accessible and simple to comprehend. Simplicity ensures that even a beginner can grasp the overall concept of the system and, over time, acquire in-depth knowledge. This approach helps lower the barrier to entry for newcomers to the field.

The simplicity of software is the result of adhering to several popular rules found in the literature on design and software architecture. The first one is the KISS principle, which stands for 'Keep It Simple, Stupid'. It emphasizes the importance of keeping designs and systems as straightforward and uncomplicated as possible [Mar08]. This principle advocates for simplicity to enhance usability, maintainability, and effectiveness. By avoiding unnecessary complexity, the KISS principle aids in creating more reliable, efficient,

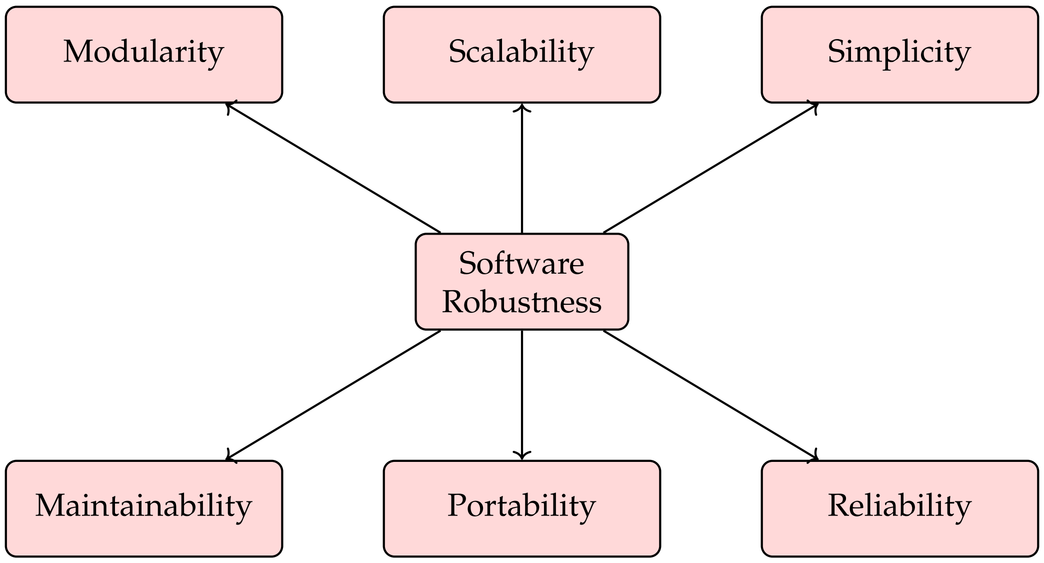


FIGURE 3.15: Diagram illustrating the various aspects of software robustness in VSLAM examined in this thesis.

and user-friendly solutions that are easier to understand, develop, and maintain. The next principle is DRY, an abbreviation for 'Don't Repeat Yourself'. It highlights the importance of avoiding duplication in code and aims to reduce redundancy, facilitate changes, and improve source code quality. Adherence to the DRY principle leads to more maintainable, clearer, testable, and error-free codebases [OG21]. Finally, there is the YAGNI principle. 'You Aren't Gonna Need It' encourages developers to focus on immediate requirements without getting sidetracked by features or capabilities that might be needed in the future.

Following these rules is essential to preserve simplicity and ensure a low barrier to entry for beginners.

Modularity

Modular architecture is one of the most desired features of software, especially in scientific domains where the need to conduct numerous experiments in a short period of time is pivotal. Modularity refers to designing software in discrete components, each responsible for a specific piece of the system's functionality. This approach allows individual modules to be developed, tested, and debugged independently of one another, facilitating parallel development and enhancing the speed at which improvements can be made.

In the context of VSLAM systems, modularity makes it possible to isolate different tasks such as feature detection, data association, and mapping into separate, interchangeable components. This not only accelerates the process of experimentation by allowing researchers to adjust or replace modules without affecting the entire system but also contributes to cleaner, more understandable code, which is crucial for collaborative work and ongoing research. Furthermore, modular systems are more adaptable and maintainable, since updates and improvements can individual components without necessitating a full-scale software changes. A well-designed modular architecture thus forms the backbone of scalable and robust scientific software, aligning with the needs of a fast-paced, evolving research environment [JH18].

The absence of modularity in VSLAM systems has directly led to the proposition of Modular SLAM described in Chapter 4. This concept emphasizes the need for rapid prototyping of new solutions that allow not only the tweaking of individual component parameters but also the swift exchange and evaluation of novel algorithms. This strategy is aimed at accelerating research and the development of new systems. Additionally, it offers an advantageous platform for new students to engage with, learn from, and experiment on pre-existing systems, thereby fostering an educational environment conducive to innovation and discovery in the field of VSLAM.

Reliability

Software reliability is a critical factor in the VSLAM systems. Long-term operation discussed in Section 3.3 without well tested and reliable software is not achievable. Achieving continuous and dependable performance over extended periods is dependent on the deployment of well-tested and reliable software. This necessity stems not only from the operational demands of VSLAM technologies but also from the foundational requirement for precise and trustworthy experimental outcomes.

Moreover, it is imperative for researchers to have confidence in the correctness of the algorithms they employ. The presence of software bugs can significantly distort experimental results, leading to incorrect conclusions and potentially derailing the development of new technologies. To mitigate these risks, a rigorous testing framework is essential—one that encompasses a variety of scenarios, from controlled environments to unpredictable real-world conditions. Such a framework ensures that software reliability is maintained across diverse operational contexts, bolstering the integrity of the research.

In addition to careful testing, the application of best practices in software development plays a vital role in enhancing software reliability. This includes the adoption of version control systems like git, continuous integration and deployment pipelines, and code review processes [Tsi24]. Implementing these practices facilitates early detection of errors, simplifies the debugging process, and contributes to the overall robustness of VSLAM systems. In the long run, the goal is to create a software foundation that not only supports the current state of VSLAM research but is also adaptable enough to accommodate future advancements in the field.

Portability

Modern software is designed to run on a wide array of devices that differ not only in their memory, storage capacity, and processor speed, but also in their underlying architecture. For example, in the field of VSLAM, systems are no longer confined to the conventional x86 architecture. There is a growing trend to deploy these systems on devices with different architectures, such as the Raspberry Pi, which utilizes ARM architecture. This shift necessitates a focus on portability – the ease with which software can be adapted, moved, and executed on various hardware platforms. Ensuring portability means that VSLAM applications can be more universally used, allowing researchers and practitioners to implement these sophisticated mapping and localization systems in a diverse range of environments and contexts.

Maintainability

In the scientific community, it is frequently observed that systems with open-sourced code eventually become unmaintainable. This predicament often arises because many researchers, moving forward to new projects, finish supporting their previous works. The implications of this practice are significant, leading to a landscape where once cutting-edge software quickly becomes obsolete, lacking critical updates or compatibility with evolving technologies. Such a trend not only slows down the progression of research but also diminishes the potential long-term impact of these projects like cooperation with the industry. Ensuring maintainability requires a commitment to ongoing support, adequate documentation, and a framework that encourages other contributors to adapt and build upon the existing codebase, thereby fostering a sustainable and collaborative environment for scientific innovation. Especially, the last point i.e. encourage of other users for further contribution by the quality and simplicity is crucial. In this thesis, it is believed that providing such framework can boost the maintainability of the VSLAM projects.

3.9.2 Algorithms robustness

Within the realm of VSLAM, a significant portion of the scientific literature emphasizes the concept of robustness, frequently incorporating it within the titles of research papers. This emphasis largely centers on the algorithmic dimensions of robustness. Specifically, algorithmic robustness refers to the capability of VSLAM systems to maintain high levels of performance in localization and mapping accuracy. This form of robustness is vital as it ensures the algorithms underpinning VSLAM systems can produce accurate and dependable outcomes, even when navigating through complex and unpredictable environments. As outlined in Chapter 3, there are numerous, various challenges that relates to algorithmic robustness. These challenges encompass dealing with dynamic changes within environments, preserving accuracy over extended periods i.e. long-term operation, compensating for drift, and accurately detecting loops, properly close them and robustness to the different weather and light conditions [JKK22]. Addressing these challenges is paramount for enhancing the resilience of VSLAM algorithms, thereby making them more adept at handling the complexities inherent in real-world applications. To summarize, Figure 3.16 shows the diagram of different aspects of algorithm robustness.

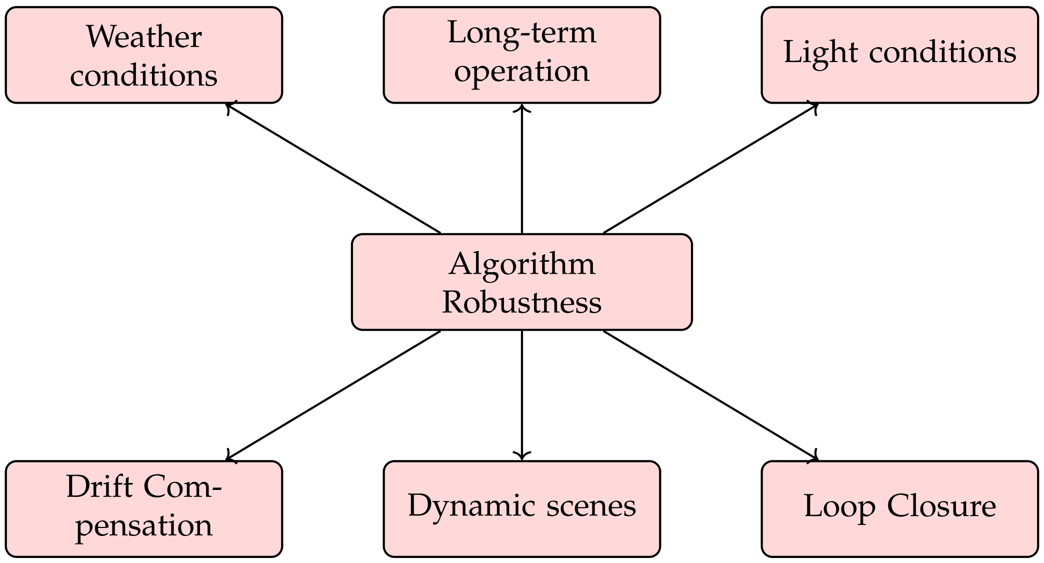


FIGURE 3.16: Diagram illustrating the various aspects of algorithmic robustness in VSLAM examined in this thesis.

3.9.3 Execution efficiency robustness

A key distinction between VSLAM and SfM is the requirement for real-time operation in VSLAM systems. This necessitates that performance is critically important throughout the VSLAM process. Typically, the processing of each frame should be completed within 30ms being a relatively tight span given the complexity of the computations involved. This strict time constraint underscores the challenge of balancing the demand for rapid processing with the need for accurate localization and mapping.

Scalability is of paramount importance; as environments grow in size and complexity, VSLAM systems must scale efficiently without compromising on the speed and accuracy of processing. Efficient scalability can be achieved by employing smart resource management strategies. These strategies ensure optimal use of available computational resources, such as intelligently allocating tasks across processors and managing memory usage to prevent bottlenecks.

Furthermore, the choice of algorithms and their parameters plays a significant role in maintaining robust execution efficiency. Selecting the right algorithms that balance speed and precision, and tuning their parameters for optimal performance, are essential steps in designing a VSLAM system suited for real-time applications. Efficient code optimizations are another cornerstone of robust VSLAM systems. Optimized code that can execute tasks rapidly and accurately is crucial, as it directly affects the frame processing rate and overall system responsiveness.

Concurrency, or the ability of the system to perform multiple operations in parallel, is also instrumental in achieving execution efficiency robustness. By leveraging multi-threading and modern multi-core processors, VSLAM systems can handle several tasks simultaneously, such as processing incoming sensor data while updating the map and determining the localization, all within the critical 30ms window. Utilizing technologies like high parallelization using GPU and CUDA [HKE22] or C++ high performance code [ASG20] may also benefit in the reduction of computing time.

Computational efficiency has a profound impact on energy savings, especially in resource-constrained environments where power consumption is a critical concern. Efficient algorithms and streamlined code not only speed up processing and reduce execution time but also minimize the energy required to perform computations. When a system operates with higher computational efficiency, it utilizes processor power more effectively, leading to a decrease in the overall energy demand.

In summary, each aspect of execution efficiency robustness from algorithm choice and parameter tuning to code optimization must be considered and balanced. This ensures that VSLAM systems not only work in smaller, controlled environments but also demonstrate the same level of precision and reliability when scaling to larger, more complex scenarios. It's these considerations that underpin the development of robust VSLAM systems capable of meeting the rigorous demands of real-time operation across a vast array of applications. Figure 3.17 presents all aspects of the execution time robustness discussed in this section.

3.9.4 VSLAM datasets

Evaluating VSLAM systems across a variety of scenarios and conditions is crucial for ensuring their effectiveness and robustness. The broad range of applications for VSLAM demands that these systems operate reliably under diverse environmental conditions and operational constraints. Factors such as lighting variations, the presence of dynamic objects, and scene complexity can significantly influence VSLAM performance. Testing these systems

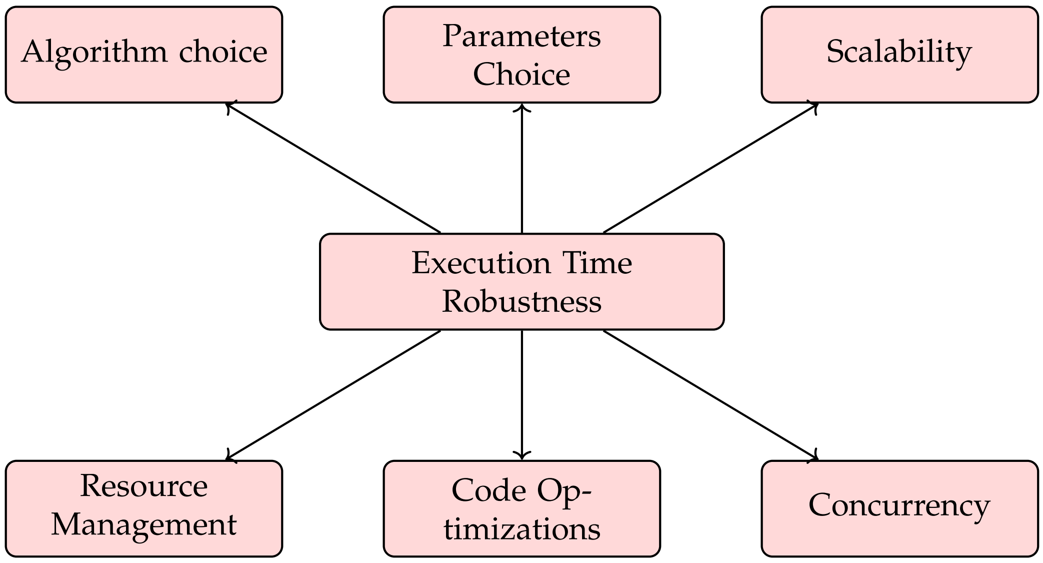


FIGURE 3.17: Diagram illustrating the various aspects of execution efficiency robustness in VSLAM examined in this thesis.

in multiple scenarios allows researchers and developers to indicate the specific strengths and weaknesses of their algorithms. This process aids in optimizing VSLAM algorithms for accuracy, efficiency, and reliability. Moreover, thorough evaluation ensures that VSLAM systems are equipped to manage the real-world challenges they will encounter in practical deployments. For these reasons, utilizing a variety of datasets that represent diverse environments with varying levels of difficulty is essential. Over the years, numerous datasets have been developed to support this process. Table 3.3 presents datasets which are the most used by the community.

The KITTI dataset [GLU12] is the most widely used in the VSLAM community. It features extensive driving scenarios captured with a range of sensors, including RGB, grayscale stereo cameras, and LIDAR. This dataset is especially valuable for testing VSLAM systems in automotive and outdoor contexts. Examples from the KITTI dataset are shown in the third and fourth rows of Figure 3.18.

Above the KITTI dataset, in the first and second rows of Figure 3.18, is the TUM dataset [Stu+12]. This dataset is ideal for evaluating VSLAM algorithms in indoor settings using RGB-D cameras. According to Sturm et al., the provided sequences encompass a wide array of scenes and camera motions. Additionally, the dataset includes simpler sequences featuring only translational or rotational movements, or those without loop closures, which are useful for straightforward debugging of algorithms. The TUM dataset contains 39 sequences depicting office and industrial environments.

Next, Burri et al. prepared dataset for Micro Aerial Vehicle (MAV). The EuRoC dataset is a set of 11 sequences with a various difficulty level from

Name	*Reference*	*Sensors*	*Notes*
KITTI	[GLU12]	Stereo cameras, LiDAR	Widely used for benchmarking algorithms in autonomous driving scenarios; focuses on outdoor environments
TUM	[Stu+12]	RGB-D, Stereo	Features indoor environments, suitable for evaluating depth-based SLAM techniques
EuRoC	[Bur+16]	Stereo, IMU	Contains challenging MAV flight sequences in indoor and outdoor setups
Bonn	[Pal+19]	RGB-D	Focused on navigation with dynamic elements
TartanAir	[Wan+20]	Synthetic RGB, Depth	Offers visually and dynamically diverse environments with adverse weather conditions for simulation
Hilti	[Zha+23]	LiDAR, Cameras	Targets applications in construction environments, includes high-resolution sensors for detailed mapping

TABLE 3.3: List of most popular Visual SLAM datasets.

slow flights with good visibility to fast, motion blurred and poor light conditions. It is worth underlining that ground-truth is very precise thanks to laser tracking system. Due to nature of MAV, captured images has different characteristics and is also a good test for the VSLAM algorithm. It consists not only of visual data but also IMU output was collected.

Following these resources, the EuRoC dataset [Bur+16] is uniquely tailored for Micro Aerial Vehicles (MAVs). It encompasses 11 sequences, ranging from slow flights with clear visibility to fast, motion-blurred flights under poor lighting conditions. The EuRoC dataset is distinguished by its highly accurate ground truth data, provided by a laser tracking system. The distinct dynamics of MAVs mean the captured images have unique characteristics, making this dataset a valuable tool for testing VSLAM algorithms. Furthermore, the dataset includes IMU outputs along with visual data.

The Bonn dataset, published by Palazzolo et al. in [Pal+19], focuses on the challenges posed by dynamic environments on the accuracy of VSLAM systems. This dataset is designed with a variety of scenes that include multiple dynamic objects, allowing researchers to test and evaluate the robustness of algorithms in handling dynamic changes.

In the development of the TartanAir dataset [Wan+20], the authors addressed the challenge of collecting accurate ground-truth data under specific conditions by using a simulation-based approach. Generated using Unreal Engine and AirSim [Sha+17], this dataset offers a diverse array of synthetic data featuring various lighting conditions, weather scenarios, and surface textures. An automated pipeline facilitates the evaluation of how different factors impact VSLAM performance, highlighting the ongoing complexities and challenges in solving VSLAM. Examples from TartanAir are depicted in the fifth and sixth rows of Figure 3.18.

Similarly, the creators of the Hilti-Oxford dataset aim to push the limits of VSLAM systems. However, their methodology differs markedly. Rather than relying on simulated data, this dataset was produced using a specially designed data collection device that includes LIDAR, five cameras, and an IMU, combined with a novel ground-truth collection technique that achieves millimeter-level accuracy. Like other datasets, the Hilti-Oxford dataset categorizes its sequences into three levels of difficulty, offering a structured approach to testing VSLAM systems under varied conditions.

To summarize, the VSLAM community recognizes the critical role of datasets in advancing the field. Numerous open-source datasets are available, facilitating the comparative evaluation of various VSLAM systems across different scenarios. It is important to highlight that the datasets mentioned previously—such as KITTI, TUM, EuRoC, TartanAir, and Bonn—are just a few examples of the resources available to researchers. The literature also references several other datasets, including GSLAM [Zha+19b] and VECtor [Gao+22].

These datasets not only offer diverse environmental conditions and challenges but also enable consistent and reliable benchmarking of performance, driving forward the development and refinement of VSLAM technologies. By providing a standardized basis for comparison, these datasets play an indispensable role in the evolution and enhancement of VSLAM systems.

The next section describes how these datasets can be used to evaluate and compare VSLAM systems. It will delve into the methodologies for conducting systematic tests using these datasets, including the metrics for measuring system accuracy, efficiency, and robustness in various environmental conditions.

3.10 VSLAM evaluation

After introduction to the most popular datasets in the VSLAM field of study, discussion about performance evaluation is needed. The evaluation plays a pivotal role in determining the effectiveness and applicability of different systems. These metrics serve as critical benchmarks that not only facilitate a comprehensive understanding of a VSLAM system's capabilities but also guide the development and refinement of more advanced algorithms.

To compare various VSLAM systems it is crucial to provide not only visual data but essentially quantitative metrics facilitating an accurate comparison of these systems. The evaluation of VSLAM systems through objective metrics is vital for several reasons. Fundamentally, it enables scientists and developers to assess both the accuracy and robustness of a system's capabilities for various conditions and scenarios. Such analysis may help in identifying advantages and disadvantages of specific VSLAM algorithm providing a guidance for further improvements. Secondly, quantitative evaluation plays a pivotal role in benchmarking. By applying a standardized set of metrics different VSLAM systems can be directly compared against each other. Benchmarking is crucial not only for academic research but also for practical applications, ensuring that the chosen VSLAM solution meets the specific requirements of real-world tasks, such as autonomous navigation, augmented reality, and robotics. Furthermore, performance evaluation highlights the trade-offs inherent in VSLAM systems. For instance, a system might prioritize real-time processing speed over absolute accuracy, making it suitable for applications requiring high responsiveness but tolerating minor localization errors. Conversely, another system might focus on generating highly accurate maps, benefiting applications where precision is paramount. Understanding these trade-offs through quantitative metrics allows for informed decision-making when selecting or designing VSLAM systems for particular applications. Additionally, the performance evaluation of VSLAM systems is critical for validating theoretical advancements. Quantitative metrics enable researchers to substantiate claims of improvements over previous methods,

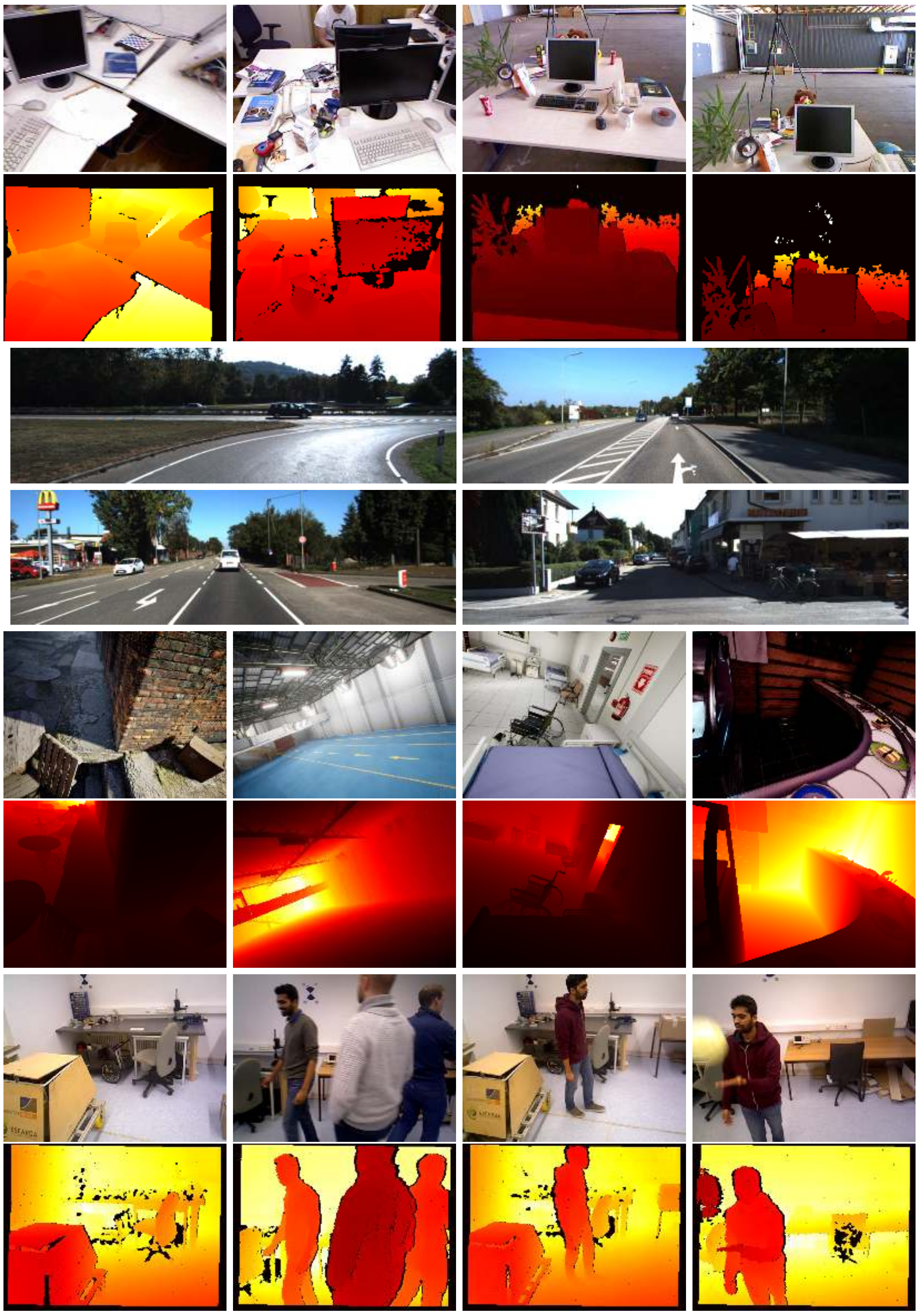


FIGURE 3.18: Examples of VSLAM datasets including TUM, KITTI, TartainAir and Bonn.

demonstrating how new algorithms enhance localization accuracy, reduce drift, or improve the system's robustness against environmental changes and dynamic obstacles.

Bearing all these considerations in mind, it becomes evident that there is a necessity for dedicated tools aimed at SLAM evaluation. Among these, one of the most popular and robust tool for this purpose is evo [Gru17]. Evo is a collection of libraries and tools for VO and SLAM evaluation. It supports various trajectories formats including TUM [Stu+12], KITTI [GLU12] and EuRoC [Bur+16]. It provides both metrics calculation as well as generating trajectory related plots. Additionally, it is noteworthy that evo can align trajectory points utilizing Umeyama's method [Ume91], with the capability to introduce scale correction.

In terms of metrics calculation, evo provides two major metrics i.e. Absolute Pose Error(APE) and Relative Pose Error(RPE). The former metric assesses the global consistency of the estimated SLAM trajectory, whereas the latter examines its local consistency.

Let assume that $\mathbf{C}_{\text{ref},n}$ is a reference pose at time n. Then, inverse composition operator $\ominus$ may be introduced. Its input is two poses and the results is the relative pose E_n given by Equation 3.4 [LM97], where $||\cdot||_F$ is a Frobenius norm and $\log_{SO(3)}(\cdot)$ is the inverse of $exp_{SO(3)}(\cdot)$.

$$E_n = \mathbf{C}_n \ominus C_{\text{ref},n} = \mathbf{C}_n^{-1}\mathbf{C}_{\text{ref},n} \tag{3.4}$$

Based on E_n, several variants of APE may be defined depending on chosen pose relations including translation part, rotation angle, rotation part and full transformation given by Equation 3.5–3.8.

$$\text{APE} = ||\text{trans}(E_n)|| \tag{3.5}$$

$$\text{APE} = |\text{angle}(\log_{SO(3)}(\text{rot}(E_n)))| \tag{3.6}$$

$$\text{APE} = ||\text{rot}(E_n) - I_{3\times3}||_F \tag{3.7}$$

$$\text{APE} = ||E_n - I_{4x4}||_F \tag{3.8}$$

By default, evo package calculates APE as a translation part. Similarly, in this thesis, it is assumed that APE is equivalent to Equation 3.5. Finally, several statistics may be formulated including APE_{RMSE}, $\overline{\text{APE}}$, APE_{std}, APE_{max}, APE_{min} and APE_{SSE} given by following equations.

$$\text{APE}_{\text{RMSE}} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\text{APE}_n^2} \tag{3.9}$$

$$\overline{\text{APE}} = \frac{1}{N}\sum_{n=1}^{N}\text{APE}_n \tag{3.10}$$

$$\text{APE}_{\text{std}} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N}(\text{APE}_n - \overline{\text{APE}})^2} \tag{3.11}$$

$$\text{APE}_{\text{max}} = \max(\text{APE}_n) \tag{3.12}$$

$$\text{APE}_{\text{min}} = \max(\text{APE}_n) \tag{3.13}$$

$$\text{APE}_{\text{SSE}} = \sum_{n=1}^{N}(\text{APE}_n - \overline{\text{APE}})^2 \tag{3.14}$$

To summarize the results, there are two prevalent methods. First, the aforementioned statistics can be compiled and displayed in tabular form. Alternatively, visualization through graphs, particularly histograms, serves as another effective approach. To give an example, an evaluation of StellaVSLAM and ORB-SLAM3 on rgbd_dataset_freiburg2_desk TUM sequence has been performed. The results are presented in Table 3.4. Figure 3.19 is an equivalent of the table mentioned. In particular, such histograms may be useful when many VSLAM systems are compared.

APE

System	*RMSE*	*Mean*	*Median*	*Std*	*Min*	*Max*	*SSE*
ORB-SLAM3	0.01076	0.00971	0.00933	0.00464	0.00070	0.02662	0.25178
StellaVSLAM	0.01094	0.01042	0.01006	0.00331	0.00204	0.02369	0.26391

RPE

System	*RMSE*	*Mean*	*Median*	*Std*	*Min*	*Max*	*SSE*
ORB-SLAM3	0.00361	0.00308	0.00276	0.00187	0.00010	0.03607	0.02828
StellaVSLAM	0.00294	0.00255	0.00227	0.00146	0.00016	0.01535	0.01905

TABLE 3.4: Comparative Analysis of VSLAM System Performances. This table presents a comprehensive evaluation of ORBSlam3 and StellaVSLAM systems, showcasing key statistical measures of Absolute Pose Error (APE) including the Root Mean Square Error (RMSE), Mean, Median, Standard Deviation (Std), Minimum (Min), Maximum (Max) values, and the Sum of Squared Errors (SSE).

Despite the apparent benefits of presenting numerical data, such figures may not always be adequate in certain analytical contexts. Particularly when the objective is to analyze the distribution of the APE metric, violin plots become essential. They provide a more comprehensive visual representation of the data's distribution. Figure 3.20 illustrates the APE distributions for

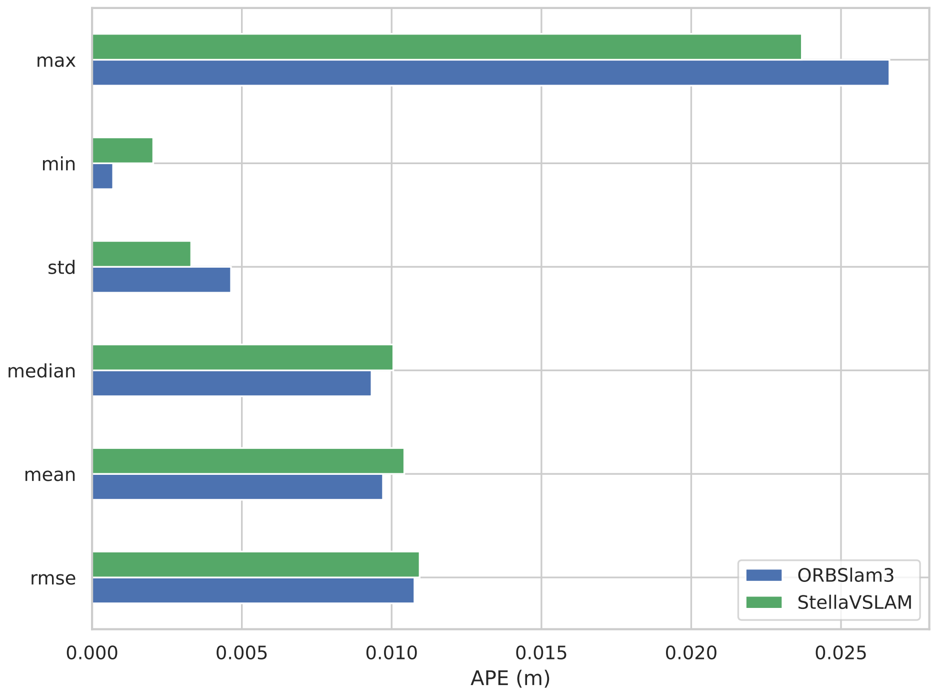


FIGURE 3.19: Comparative Analysis of VSLAM System Performances.

two evaluated systems, offering insight into the spread and tendency of the errors.

In similar manner, RPE may be introduced. Let define delta pose difference $E_{n,m}$ given by Equation 3.15.

$$E_{n,m} = \delta_{n,m} \ominus \delta_{\text{ref},n,m} = (\mathbf{C}_{\text{ref},n}\mathbf{C}_{\text{ref},m})^{-1}(\mathbf{C}_n\mathbf{C}_m) \tag{3.15}$$

Leveraging $E_{n,m}$, multiple variations of RPE can be established, contingent on the selected pose relations. Again, these include the translational component, rotational angle, rotational component, and the comprehensive transformation characterized by Equation 3.16–3.19.

$$\text{RPE} = ||\text{trans}(E_{n,m})|| \tag{3.16}$$

$$\text{RPE} = |\text{angle}(\log_{SO(3)}(\text{rot}(E_{n,m})))| \tag{3.17}$$

$$\text{RPE} = ||\text{rot}(E_{n,m}) - I_{3\times3}||_F \tag{3.18}$$

$$\text{RPE} = ||E_{n,m} - I_{4\times4}||_F \tag{3.19}$$

Lastly, a range of metrics including RPERMSE, $\overline{\text{RPE}}$, RPEstd, RPEmax, RPEmin, and RPESSE can be derived, as delineated by the equations that follow. Compared to APE, RPE appears to be more complex owing to the necessity of computing $E_{n,m}$ for each pair of $n, m \in \{1, \ldots, N\}$.

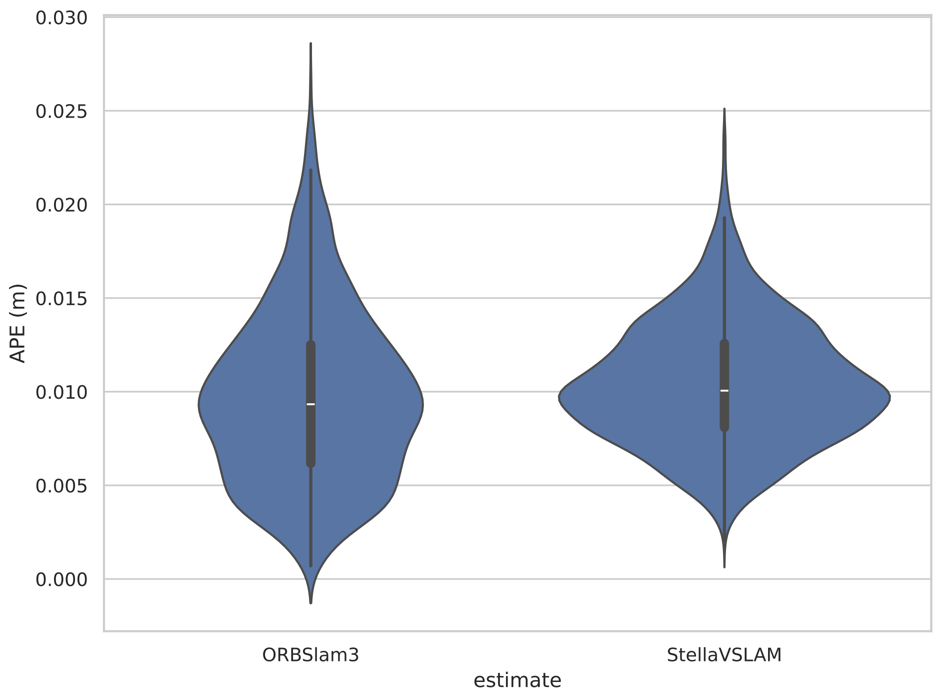


FIGURE 3.20: Comparison of VSLAM's APE distribution.

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N} \mathrm{RPE}_n^2} \tag{3.20}$$

$$\overline{\mathrm{RPE}} = \frac{1}{N}\sum_{n=1}^{N} \mathrm{RPE}_n \tag{3.21}$$

$$\mathrm{RPE}_{\mathrm{std}} = \sqrt{\frac{1}{N-1}\sum_{n=1}^{N} (\mathrm{RPE}_n - \overline{\mathrm{RPE}})^2} \tag{3.22}$$

$$\mathrm{RPE}_{\mathrm{max}} = \max(\mathrm{RPE}_n) \tag{3.23}$$

$$\mathrm{RPE}_{\mathrm{min}} = \max(\mathrm{RPE}_n) \tag{3.24}$$

$$\mathrm{RPE}_{\mathrm{SSE}} = \sum_{n=1}^{N} (\mathrm{RPE}_n - \overline{\mathrm{RPE}})^2 \tag{3.25}$$

The second way of evaluation provided by evo package is generating trajectory plots. It is a fundamental tool which offers a graphical representation of the trajectory taken by platform over time. There are numerous advantages of such plots. First, they provide an immediate visuals of the trajectory characteristics. Second, by comparing the plotted trajectory against a reference trajectory, one can assess the accuracy of a navigation system. Deviations from the reference trajectory are easily spotted and quantified. Moreover, such comparison may be considered not only in terms of reference trajectory, but also for other SLAM systems. This is especially useful in scenarios where several algorithms are being tested under the same conditions. The plots can help in understanding the dynamic behavior of the system, such as how quickly and accurately it corrects its course in response to environmental changes. Furthermore, they can be overlaid on maps, providing context for the trajectory and helping to identify environmental factors affecting performance.

To demonstrate the previously outlined benefits, trajectories based on data derived from the TUM rgbd_dataset_freiburg2_desk sequence were plotted for two evaluated VSLAM systems. Figure 3.21 depicts the estimated 3D trajectories of ORB-SLAM3 and StellaVSLAM in comparison to the reference trajectory. It may be observed that camera was moving around the environment. However, it is common that such graph does not show every detail of the trajectory, prompting the necessity for additional types of visualizations. Consequently, Figure 3.23 shows the same trajectory limited to xz plane. While the 3D plot suggested minimal variation in the z-axis, in reality, the z value varied within the range of $[1.2, 1.8]$. Two presented graphs are great tool for visualizing spatial relations. Although, time also plays very important role. Therefore, to encompass this aspect, two further plot types are introduced, showcasing xyz coordinates and yaw, pitch, roll angles as functions of time. These temporal relationships are shown in Figure 3.23 and Figure 3.24, offering a comprehensive view of the systems' performance over time.

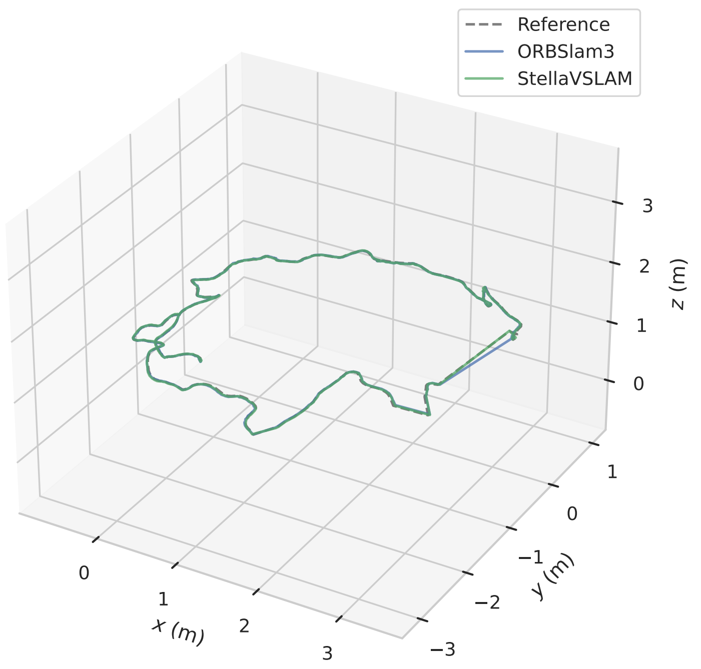


FIGURE 3.21: Trajectories plot in 3D.

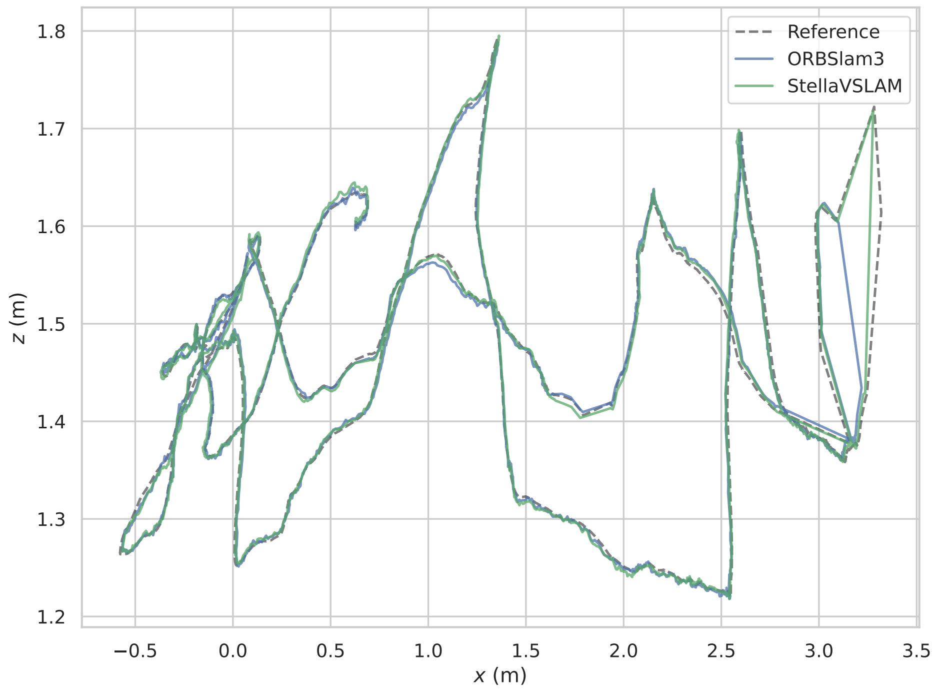


FIGURE 3.22: Trajectories plot in 2D.

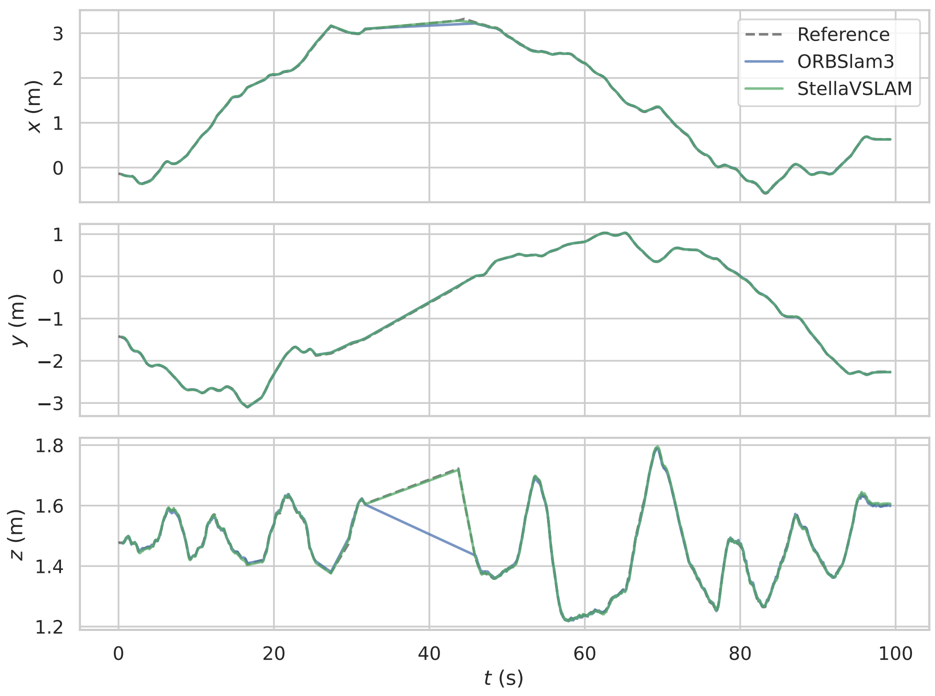


FIGURE 3.23: Trajectory and temporal position relationships.

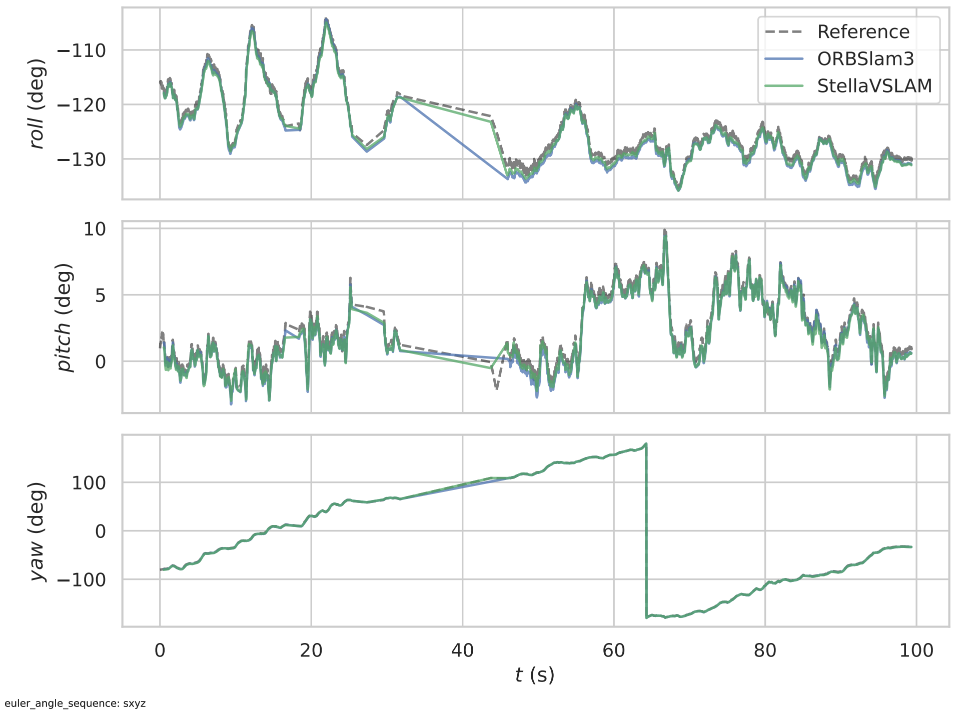


FIGURE 3.24: Trajectory and temporal angles relationships.

Chapter 4

Concept of Modular SLAM

Overview

This thesis introduces a major contribution: Modular SLAM. Modular SLAM is designed as a flexible solution that enables researchers to rapidly construct and assess the performance of Visual Simultaneous Localization and Mapping (VSLAM) systems. This chapter provides a detailed description of the Modular SLAM concept. It begins by outlining the primary goals and underlying assumptions of the system. The chapter further explores the architecture and main components of Modular SLAM, detailing how these elements contribute to its flexibility and extensibility, setting the stage for future enhancements and research within this framework.

Design and programming are human activities; forget that and all is lost.
– Bjarne Stroustrup

In the preceding chapters, referred to as Chapter 2 and Chapter 3, the historical context, problem formulation, and associated challenges were explored. This chapter represents a pivotal transition towards the principal contributions of this thesis. The concept of Modular SLAM is discussed in the following sections. In the Chapter 5, the methods improving VSLAM robustness are discussed.

4.1 Motivation

Visual SLAM is a complex and challenging problem, especially for novice researchers. Building a high-quality VSLAM system requires a wide range of skills and knowledge, including computer vision, 3D reconstruction, optimization algorithms, parallel computations and GPU programming, and code optimization. The threshold for entry into this field is high, and it requires knowledge of topics from various scientific fields and techniques. Unfortunately, the SLAM community lacks materials and programming libraries that can easily introduce a beginner to the complex studies. Many scientists have had to spend a significant amount of time getting familiarized with the subject. The best references for novice researchers are [Cha20], which provides a roadmap of the most popular SLAM systems, techniques, and skills that every SLAM scientist or developer should know, and [Gao+17], a book consisting of 14 lectures on VSLAM that are focused on practical aspects. Additionally, there are also several surveys and overview papers regarding SLAM mentioned in Chapter 2. However, most of them still remains difficult for beginners and they lack an overview of practical aspects.

Another issue prevalent in VSLAM development is the tendency for major VSLAM systems to independently implement standard algorithms, within their own distinct codebases. This practice of not reusing existing solutions can lead to several disadvantages. Although having separate, custom implementations might allow for optimizations specific to each system's requirements, it also results in duplicated effort, increases maintenance complexity, hinders compatibility between different systems. This fragmentation can slow down collaborative advancements and the integration of improvements across platforms. Common tools and algorithms that can be reused are crucial for enhancing researchers' productivity [RI19]. For instance, the AI research community has experienced a significant acceleration in progress by adopting common tools, such as PyTorch [Pas+19]. These frameworks allow for quick experimentation and easy sharing of results, underscoring the benefits of standardization and code reuse in advancing technology.

The research associated with VSLAM systems can be extended to various components, such as feature detection, outlier handling, mapping, localization, among others. Typically, researchers focus on applying their newly proposed algorithms to just one of these specific areas. Consequently, the ability to swiftly replace an old algorithm with a new version can significantly expedite research progress. Therefore, ensuring that VSLAM systems are modular is essential for conducting reliable and systematic research. This modularity facilitates easier updates and enhancements, allowing for more dynamic and responsive development in the field. Moreover, modularity also allows for implementing a dynamic behavior in the system by substituting components based on current conditions. This adaptability can be crucial for VSLAM systems operating in varied and unpredictable environments where conditions can change rapidly. For instance, different feature detection algorithms might be more effective in different lighting conditions or landscapes. By enabling the system to automatically switch between these algorithms depending on the environmental inputs, modularity not only enhances the system's robustness but also optimizes its performance in real-time.

In computer vision tasks, including VSLAM, visualization plays a crucial role. It not only demonstrates whether the algorithm is functioning correctly, but it also provides intuitive insights into how the algorithm processes and interprets the visual data. Effective visualization helps in identifying specific areas where the algorithm might be underperforming, such as inaccuracies in feature detection or errors in mapping. Additionally, it enables researchers and developers to observe the real-time performance of their systems, offering a clear, visual assessment of dynamic changes and interactions within the environment.

Furthermore, the evaluation of VSLAM systems plays a crucial role in the field. As it has been already discussed in Section 3.10, it facilitates an objective comparison between different systems, which is essential for identifying advancements and areas that require further improvement. Providing common functionality to export results and generate comparative analyses offers a significant advantage. This standardized approach to evaluation ensures that researchers can consistently measure and compare the effectiveness of various algorithms and configurations. It also allows for benchmarking against established standards, making it easier to track progress over time. Ultimately, this capability enhances transparency and fosters a more collaborative environment where insights and advancements can be shared effectively across the research community.

To summarize, simplicity and a low entry threshold, along with the reusability of common tools, the modularity of systems, and evaluation, are essential aspects of VSLAM research and development. These factors collectively enhance the efficiency, effectiveness, and accessibility of technological advancements within the field. However, it's evident that not all of these crucial aspects are fully addressed by the community, indicating room for improvements and solutions like Modular SLAM. Its goals, design, provided tools and functionality are discovered in the next several sections.

4.2 Modular SLAM Overview

The modularity and architecture of SLAM systems have been addressed in existing literature, as noted by works such as those by Blanco-Claraco and Colosi et al. [Bla19; Col+20]. However, each proposed system exhibits specific limitations. For instance, the MOLA framework, developed by Blanco-Claraco, although it supports camera inputs, predominantly emphasizes other sensors like LIDAR. Furthermore, while both methods have been credited with identifying general components of SLAM systems, Colosi et al. have described these components as "atomic", it will be demonstrated that such components can still be further subdivided to achieve even greater modularity. Modular SLAM provides an enhanced view of VSLAM systems. This framework is deeply analyzed, revealing several major parts that significantly differ from those discussed in the scientific literature. Modular SLAM focuses on further development, providing not only a default implementation of the VSLAM with several core algorithms but also maintaining high extensibility. It includes visualization and evaluation tools that facilitate the quick identification of issues. Moreover, its primary goal is to provide robust support for camera systems.

To summarize, this thesis introduces an expanded concept of Modular SLAM that not only builds upon these previous works but also places a strong focus on the application of cameras. By doing so, it aims to more fully integrate developments from the VSLAM community and capitalize on advances in software and systems modeling. This approach seeks to push the boundaries of modularity and adaptability in SLAM systems, catering specifically to the dynamic requirements of visual data processing.

4.3 System design

To effectively address the concerns highlighted in Section 4.1, careful system design is critical. An optimal VSLAM framework must incorporate three fundamental aspects essential for robust development and research. Firstly, the development aspect involves devising and optimizing core algorithms and methodologies of the VSLAM system, with a focus on enhancing accuracy and efficiency. Secondly, evaluation is crucial and involves rigorous testing to assess the system's performance, benchmarking against established metrics, and conducting comparative analyses with other VSLAM systems to ensure reliability across diverse scenarios. Lastly, visualization plays a key role in presenting spatial data and tracking information in a user-friendly manner, which aids in debugging, system refinement, and provides intuitive insights into the system's performance and behavior in real-world settings. These elements collectively form the backbone of a successful VSLAM framework, facilitating continuous improvement and adaptation in both development and research environments. These three fundamental blocks are depicted in Figure 4.1.

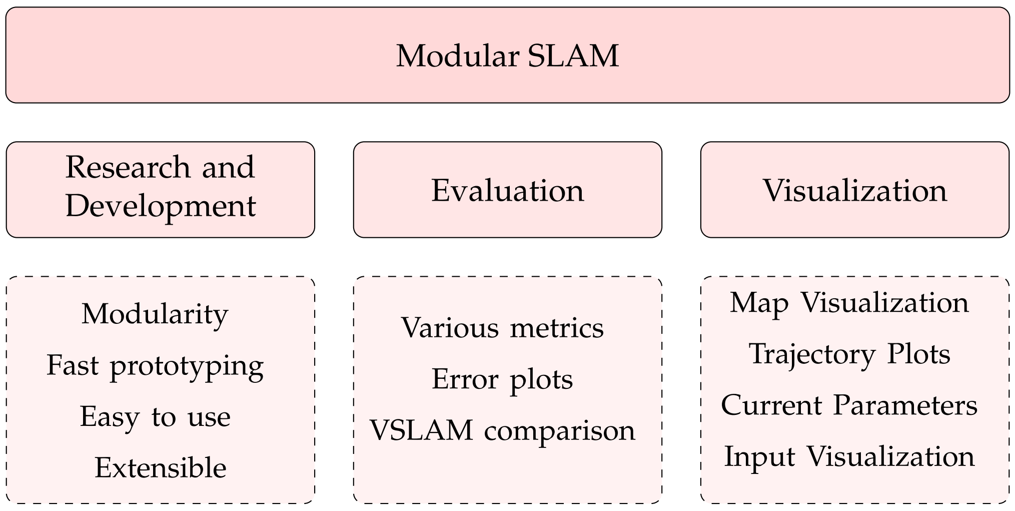


FIGURE 4.1: The fundamental components of Modular SLAM and its responsibilities.

4.3.1 Design principles

Following the discussion in Section 4.1, which explored the underlying motivations, several foundational principles have been established to guide the design of Modular SLAM. These principles aim to address the identified needs and challenges by emphasizing adaptability, extensibility, robust performance across different environments, ease of use, rapid prototyping, and adaptability. The key principles include simplicity, modularity, reliability, scalability, and portability. The attentive reader will notice that these issues have also been addressed in the chapter Section 3.9.1, where the robustness aspects were covered. As noted, these features are major In subsequent sections, each principle will be discussed further, outlining its significance in the overall design and the methods planned to achieve these aims. What's interesting, it will be shown that the way to achieve these features share common denominator.

Simplicity

In this thesis, it has been established that contemporary VSLAM systems require substantial effort from beginners to comprehend a multitude of intricate details. Additionally, this challenge is not exclusive to novices. Experienced researchers may also encounter difficulties in fully understanding these systems. Therefore, simplicity is paramount in the development process. The significance of this attribute is underscored in the literature, as exemplified by Steux and Hamzaoui, who demonstrated that a basic yet limited VSLAM system could be implemented in fewer than 200 lines of source code [SH10].

Nevertheless, it is essential to understand that simplicity does not compromise robustness, nor does it detract from the advancements or state-of-the-art nature of a system. There are strategies that effectively manage the complexity of systems, and it is widely recognized that selecting an appropriate system architecture can facilitate this management. In Modular SLAM, each component is designed with a clear and specific responsibility, utilizing a top-down approach to ensure coherent and efficient organization. The top-down approach prioritizes defining the overall system architecture first, then breaking it down into smaller, manageable subcomponents [BKC12]. Modular SLAM employs a well-established architecture comprising four basic components: the frontend, backend, data acquisition, and the semantic component. It is shown in Figure 4.2.

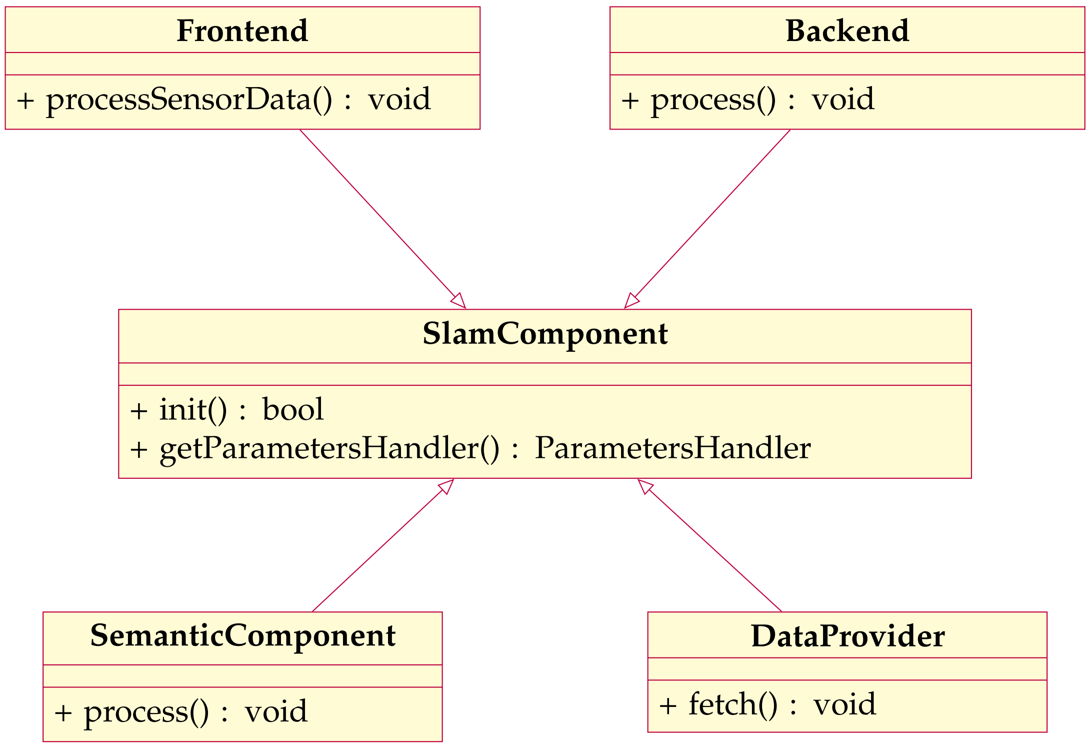


FIGURE 4.2: UML diagram of SLAM system components referring to Figure 2.19.

Each component within the system can be subdivided into more specialized subcomponents. To illustrate, consider evaluating the impact of various pose estimation algorithms on tracking performance. The top-down architecture simplifies this process significantly, focusing on two main areas. First, the responsibility for pose estimation is assigned solely to the frontend, ensuring that any modifications are confined to this component. Second, using dependency injection, the behavior of the frontend can be adapted without altering its core structure. To integrate a new pose estimation algorithm, one only needs to implement the designated interface. This method facilitates the easy swapping of algorithms and, importantly, does not require in-depth knowledge of Modular SLAM's intricate details. All changes are implemented during the VSLAM system's creation and build process. Here, the strategy and builder design patterns play a crucial role, as detailed in Section 4.3.2. In summary, the system's simplicity is maintained through two mechanisms: minimal changes are required to alter the VSLAM system, and these changes occur only during the system creation process, while the implementation of algorithms is managed on the user side. This design minimizes the need for users to fully understand the entire system, thereby facilitating quicker experimental setups.

Modularity

The name "Modular SLAM" reflects one of its cornerstone features: modularity. This design principle involves decomposing a system into distinct, interchangeable modules, with each module encapsulating a specific subset of the system's functionality. The primary aim of modularization is to simplify the management of system complexity by establishing clear and well-documented boundaries within the application [Bus+96].

In Modular SLAM, this objective is achieved through a structured approach similar to the simplicity discussed previously. The system utilizes a top-down architecture to ensure a clear separation of concerns: the backend is tasked with map estimation; data acquisition manages sensor inputs; the frontend employs various algorithms for data relationship analysis and pose tracking; and the semantic component addresses the interpretation of data in context. This architecture effectively segregates the system into well-defined modules. Within these modules, Modular SLAM employs design patterns such as strategy and visitor to facilitate the implementation of various algorithms, enhancing the system's adaptability and maintainability.

Reliability

Modular SLAM's goal is to be a reliable framework for all users. The reliability in that context is threefold. Firstly, as an open-source platform, Modular SLAM ensures transparency and community involvement, which significantly enhance its reliability. Open-source code allows developers from around the world to scrutinize, test, and improve the system's algorithms, thereby increasing the robustness and stability of the framework. This communal approach to development helps to quickly identify and rectify bugs, fosters innovation through collective problem-solving, and accelerates the evolution of the system with contributions from diverse technological cultures.

Secondly, Modular SLAM is designed to enable researchers and developers to experiment with new approaches within specific components of the system while relying on the stability and established functionality of other parts. This modular structure facilitates flexibility and extensibility, allowing users to modify or replace individual modules without affecting the entire system. Such an approach reduces the risk associated with introducing new algorithms or technologies, as the core system retains its operational integrity, thus maintaining a stable platform for continuous experimental development.

Lastly, the reliability of Modular SLAM is bolstered by its well-documented and established architecture. Good documentation is crucial as it provides a detailed understanding of the system's design and operational mechanics, which is essential for both new and experienced users. It ensures that developers can easily navigate the system, understand how different modules interact, and implement changes or enhancements correctly. Moreover, a well-structured and consistently maintained architecture ensures that despite the system's complexity and the depth of its capabilities, it remains accessible and maintainable. This organized and transparent structure significantly contributes to the overall reliability of the Modular SLAM framework by ensuring that it can be effectively used, maintained, and evolved over time.

Portability

The capability to deploy VSLAM algorithms across a diverse array of platforms is essential, given their broad applications in mobile devices, wearables, robotics, and more. Ensuring that Modular SLAM can operate on a wide range of devices is critical for reaching a broader user base. This adaptability not only enhances the system's accessibility but also maximizes its potential impact across various technological domains.

To achieve compatibility with numerous platforms, Modular SLAM employs two key strategies. First, the core of Modular SLAM is written in C++, a standard and inherently multiplatform language that underpins its architecture. This foundation allows the system to abstract platform-specific features into dedicated components, maintaining consistent core functionality while adapting to different operating environments efficiently.

Secondly, to extend its reach and usability, Modular SLAM includes bindings to other programming languages, notably Python, which is widely favored in the scientific community for its simplicity and powerful computational capabilities [Bar21]. By offering Python bindings, Modular SLAM ensures that researchers and developers accustomed to Python can easily integrate and utilize the system in their projects, leveraging its robust capabilities in a familiar programming environment.

Scalability

To ensure that the pursuit of portability does not compromise efficiency or scalability, it is crucial to optimally utilize the specific features of the hardware. Modular SLAM's architecture allows for the interchange of algorithms to exploit dedicated hardware peripherals effectively. For example, systems equipped with GPUs might employ algorithms optimized for parallel processing, such as those utilizing CUDA for intensive computation tasks, or mobile devices could use NEON technology for optimized vector processing. This flexibility ensures that Modular SLAM not only adapts to different hardware environments but also maximizes the potential of each platform to enhance performance and processing speed. Such strategic use of hardware-specific capabilities ensures that scalability and efficiency are maintained across diverse deployment scenarios.

Through the use of abstraction, Modular SLAM decouples the system's core functionalities from the hardware-specific implementations. This separation means that as new or more capable hardware becomes available, the system can be adapted without extensive modifications to the core logic. Abstraction layers within Modular SLAM serve as intermediaries, translating general commands into hardware-specific actions, thus allowing the system to operate across a diverse range of devices and platforms.

4.3.2 Design patterns

The integration of design patterns, as originally proposed in [Gam95] by Gamma, into the VSLAM system represents a significant contribution of this thesis. Although the general architecture of VSLAM systems has been extensively discussed in existing literature [Cad+16; SSM23], the novel application of design patterns specifically to enhance the robustness of VSLAM systems marks an important advancement from both a research and engineering perspective. This thesis argues that selecting an appropriate architectural framework, complemented by several design patterns, is essential for addressing and resolving issues related to software robustness. The detailed discussion presented in this section outlines how these design patterns contribute to the theoretical framework. Also, it will be briefly presented in Section 4.4 how it improves practical implementation. By systematically applying these patterns, the VSLAM system achieves enhanced stability and reliability, demonstrating the profound impact of architectural choices on the overall efficacy and robustness of the system.

In [Gam95], the authors categorized design patterns into several distinct groups, specifically creational, structural, and behavioral. Each category serves a unique function in design, facilitating solutions tailored to common development challenges and enhancing modularity and flexibility. Additionally, SOLID principles are supported by desing patterns [Mar00; Mar17; Rot17]. Let start with the creational ones to see how the process of VSLAM system build may be designed.

As previously mentioned, the overall architecture of a VSLAM system is composed of several key components: the data acquisition component, the frontend, the backend, and data semantics (refer to Figure 4.2). This structured approach is critical in efficiently managing the complex functionalities of VSLAM systems. The objectives of organizing the VSLAM system in this manner are dual-purpose. Firstly, the construction of the VSLAM system must adhere to the established general architecture. This alignment ensures that all components are integrated systematically, facilitating seamless interaction and data flow between the various modules. This systematic integration is crucial for maintaining the operational integrity and performance of the VSLAM system under different conditions. Secondly, it is imperative to provide researchers and developers with the flexibility to introduce custom and innovative features into the system. This flexibility is essential for supporting ongoing development and experimentation within the VSLAM field. Enabling researchers to implement new, possibly unforeseen features without constraints encourages innovation and allows the VSLAM system to evolve in response to emerging needs and technologies. Thus, while the general architecture provides a necessary framework for stability and consistency, the design must also accommodate the dynamic addition of new functionalities to foster advancement in VSLAM research and applications. This balance between structure and flexibility is key to developing VSLAM systems. Thus, it seems that builder pattern is an appropriate choice.

According to [FR20], the **builder pattern** encapsulates the process of constructing an object, enabling it to be assembled in a step-by-step manner. The UML diagram of the builder pattern proposed for the VSLAM is shown in Figure 4.3. In the illustrated Builder pattern for a VSLAM system, each component plays a critical role in the construction process. The `VSLAMSystemDirector` serves as the orchestrator, responsible for managing the sequence of operations to build the VSLAM system. It delegates the tasks of constructing the system's components to the `VSLAMBuilder`, thus ensuring that the assembly process adheres to a specific methodology and order. The `VSLAMBuilder` is an abstract interface that outlines the necessary methods for adding different components to the VSLAM system, such as the frontend, backend, and data acquisition modules. It also includes a method to finalize and retrieve the completed system. This interface is essential as it establishes a uniform blueprint for building a VSLAM system, which concrete builders can implement according to specific requirements. The `ConcreteVSLAMBuilder` is a specific implementation of the `VSLAMBuilder` interface. It defines how each component of the VSLAM system is constructed and integrated, allowing for the creation of a system with tailored features and functionalities. This concrete builder handles the practical aspects of component creation and assembly, ensuring that all parts are correctly configured and work cohesively. Finally, the `VSLAMSystem` represents the end product, comprising various components like the frontend, backend, and data acquisition modules. Each component is crucial for the system's overall functionality and is integrated during the construction process facilitated by the `ConcreteVSLAMBuilder` under the direction of the `VSLAMSystemDirector`. This setup not only simplifies the construction of complex systems but also ensures that modifications to the system's configuration can be made with minimal disruption to the overall architecture, thereby supporting simplicity and scalability.

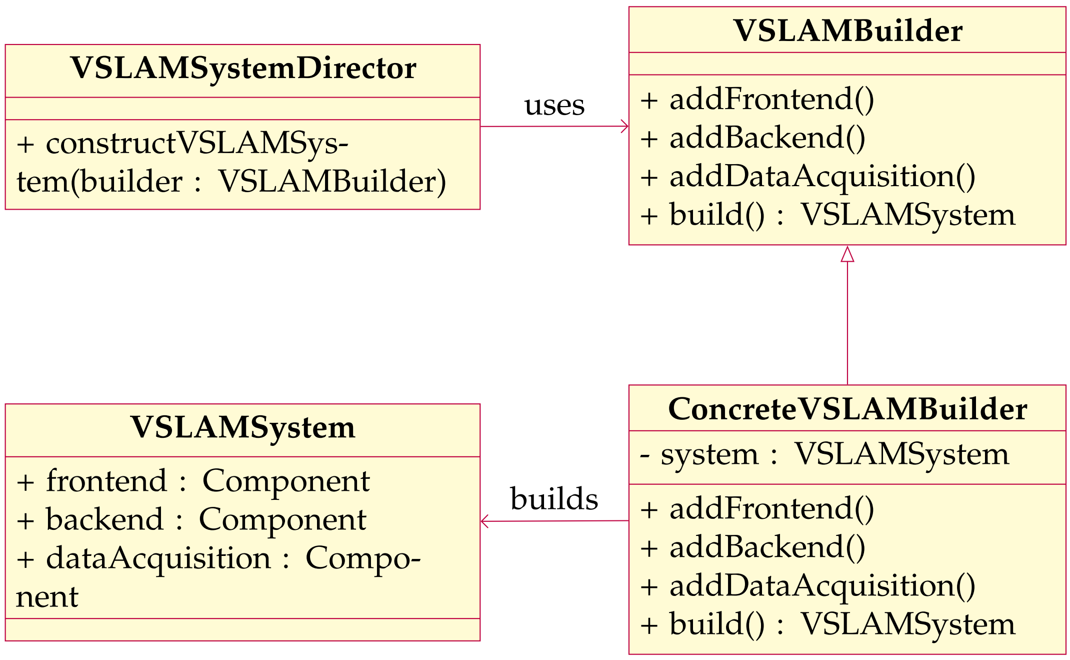


FIGURE 4.3: Builder pattern

The complexity of VSLAM systems is influenced by various factors, with one significant contributor being the extensive range of parameters associated with each component of the system. For instance, within the frontend component, parameters such as the maximum number of keypoints, edge detection thresholds, and scale factors all play crucial roles in feature detection processes. Similarly, in pose estimation, parameters like the threshold for RANSAC estimation errors critically impact the accuracy and reliability of the pose data. Furthermore, during keyframe creation, parameters such as the minimum number of tracked landmarks and the quality of tracking are vital for maintaining the integrity of the map. Each of these parameters must be carefully calibrated to optimize the system's performance, adding layers of complexity to the design and operation of VSLAM systems.

Moreover, it is common for one parameter within a VSLAM system to significantly influence another. Take the mapping process as an example: the frequency of keyframe creation directly affects the size of the map. The more frequent the creation of keyframes, the larger the map becomes. Consequently, the loop detection algorithm, which activates each time a new keyframe is added, must be optimized to handle this increased load. This is crucial because maintaining real-time operation could be compromised by the larger map size. To address this, adjustments might be needed, such as tuning the maximum number of keyframe candidates tested during loop detection to ensure efficient processing while preserving system performance.

To effectively manage VSLAM parameters, Modular SLAM introduces the concept of a `ParametersHandler`. This mechanism allows each component to register its specific parameters, facilitating centralized control and accessibility. However, a critical aspect of this system is how interconnected parameters impact each other. Especially, when the one parameters changes its value. To address this, Modular SLAM employs the **observer pattern**. It is a behavioral design pattern which has the ability to notify several objects about the occurred event with any modifications of the notifiers and notified components [Gam95]. In the scenario of handling parameters, the pattern enables a system where changes to one parameter can be observed and responded to by other dependent parameters or components, ensuring that updates are handled cohesively across the system. This integration not only simplifies parameter management but also enhances the system's adaptability and responsiveness to changes within its operating environment.

The UML diagram illustrating the observer pattern can be found in Figure 4.4. In this diagram, the `ParametersHandler` is defined as an interface that encapsulates the fundamental functionalities required for managing parameters. This interface plays a crucial role in registering new parameters and adding observers. These observers are then notified through the `handleOnChange` method whenever there is a change in the value of a parameter. Essentially, once a parameter's value is updated, the `handleOnChange` method is invoked for all observers linked to that specific parameter, ensuring that all related components remain synchronized with the latest parameter values.

Another significant advantage of using the observer pattern is its capability to integrate seamlessly with external modules that set parameters externally. This integration is particularly valuable for GUI applications, where users can manually adjust system parameters. Such functionality is especially beneficial for researchers, as it allows them to quickly modify parameters and immediately observe the effects on system performance. This rapid iteration can significantly speed up experimental processes and fine-tuning. Furthermore, the Observer pattern facilitates integration with sophisticated

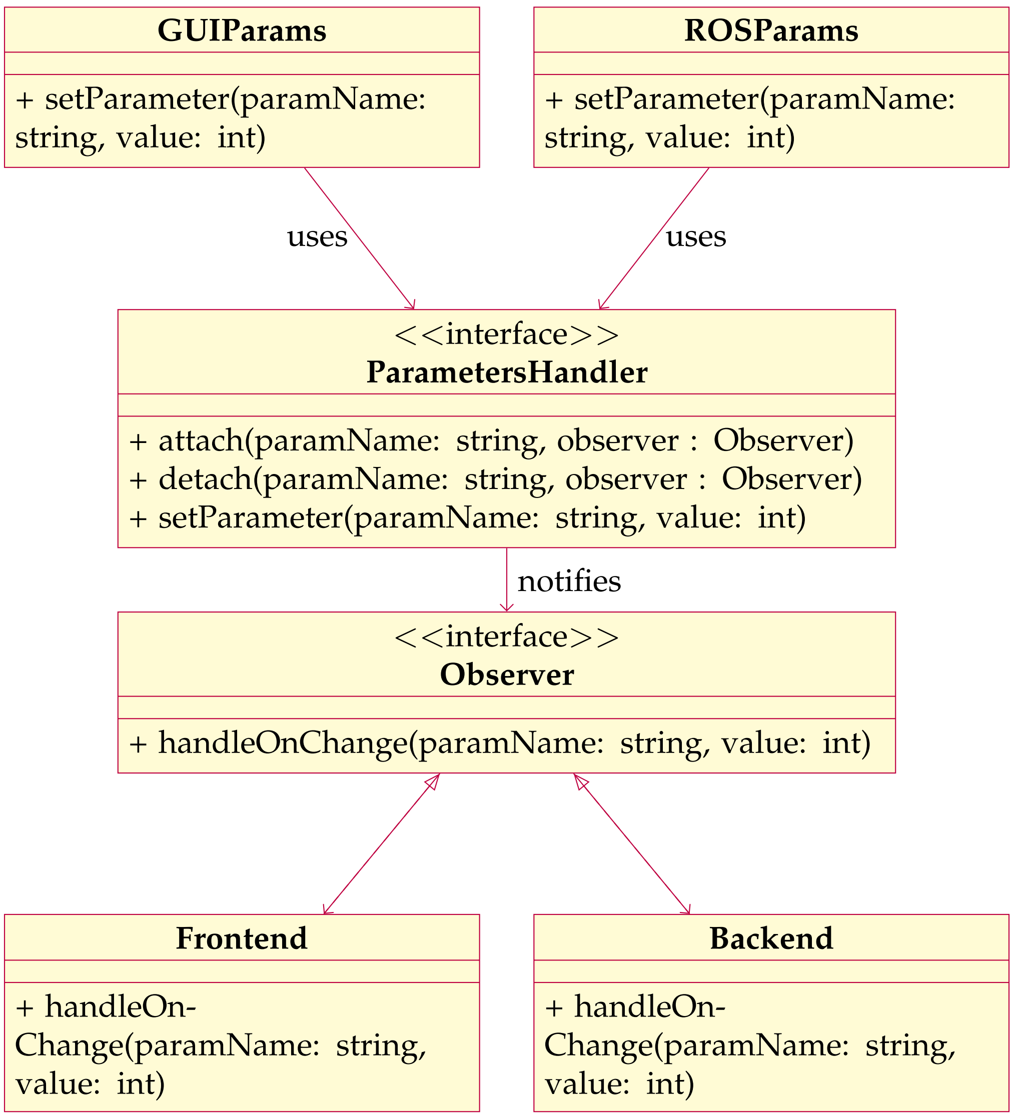

GUIParams
+ setParameter(paramName: string, value: int)
ROSParams
+ setParameter(paramName: string, value: int)
uses
uses
<<interface>>
ParametersHandler
+ attach(paramName: string, observer : Observer)
+ detach(paramName: string, observer : Observer)
+ setParameter(paramName: string, value: int)
notifies
<<interface>>
Observer
+ handleOnChange(paramName: string, value: int)
Frontend
+ handleOnChange(paramName: string, value: int)
Backend
+ handleOnChange(paramName: string, value: int)

FIGURE 4.4: Observer pattern for handling parameters change

parameter handling systems like ROS dynamic reconfigure. This tool enables real-time adjustments of parameters within the ROS environment, providing a flexible and responsive interface for managing system behaviors at a high level. The implementation of such a solution using a GUI with the `ParametersHandler` is shown in Figure 4.5.

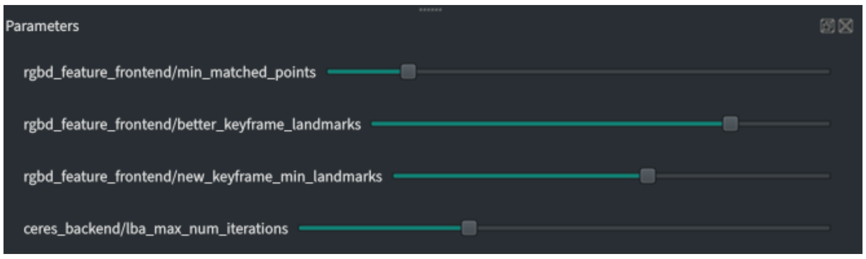

Parameters
rgbd_feature_frontend/min_matched_points
rgbd_feature_frontend/better_keyframe_landmarks
rgbd_feature_frontend/new_keyframe_min_landmarks
ceres_backend/lba_max_num_iterations

FIGURE 4.5: An example of GUI for parameters handling.

In the context of VSLAM, a map consist of landmarks and keyframes, together with their relationships. Due to the principle of modularity, these maps can be internally represented in various forms tailored to the specific requirements and architectural design of the system. From an informatics perspective, maps can be structured similarly to graphs and may be represented using data structures such as lists or matrices. However, different algorithms need to interact with these maps to locate specific landmarks or keyframes. For example, bundle adjustment is often applied only to a local map, which represents a subset of the entire map. This necessitates an efficient method for querying and manipulating the necessary elements within these maps. In scenarios where dynamic access and operation on various parts of the map are crucial, the **visitor pattern** provides an effective solution. This pattern enables external operations to be performed on elements of an object structure without changing the classes on which it operates, thereby facilitating efficient interaction with complex map structures. As noted by [Bus+96], the visitor pattern enables the incorporation of new functionalities into a system without altering its existing structure.

In Figure 4.6, the scenario of querying elements of the map is shown, emphasizing the interaction between the map structure and visitor objects in the VSLAM system. The `BasicMap` class implements the `MapVisitable` interface, indicating its capability to interact with different types of visitors by accepting a `MapVisitor` through its `accept` method. This allows the map to be flexible and extensible in terms of the operations that can be performed on its elements. The `CeresVisitor` class is a specific implementation of the `MapVisitor` interface, equipped with three distinct functions: `visitLandmark`, `visitKeyframe`, and `visitObservation`. Each function is designed to handle different types of elements within the VSLAM map—landmarks, keyframes, and observations respectively, providing tailored operations for optimization or data extraction. The diagram shows the utilization relationship between the `MapVisitable` and `MapVisitor`, with arrows indicating the direction of interaction. This setup exemplifies the visitor design pattern, where operations can be added to objects without altering their structures, thereby promoting loose coupling and enhancing the system's adaptability to future changes or extensions.

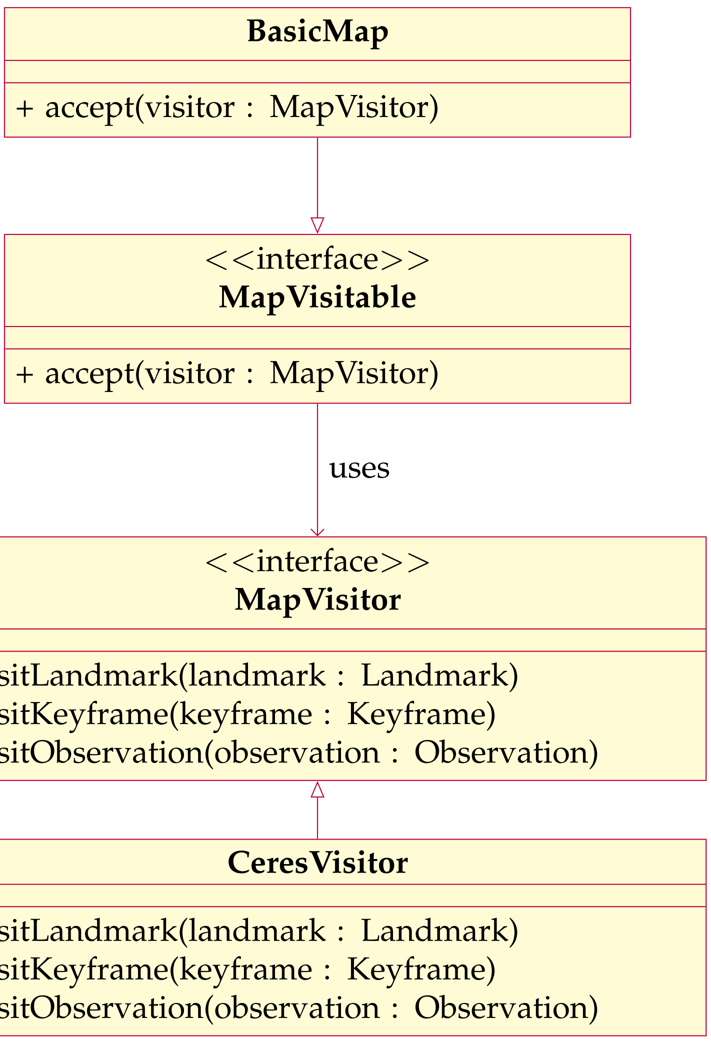

BasicMap
+ accept(visitor : MapVisitor)
<<interface>>
MapVisitable
+ accept(visitor : MapVisitor)
uses
<<interface>>
MapVisitor
+ visitLandmark(landmark : Landmark)
+ visitKeyframe(keyframe : Keyframe)
+ visitObservation(observation : Observation)
CeresVisitor
+ visitLandmark(landmark : Landmark)
+ visitKeyframe(keyframe : Keyframe)
+ visitObservation(observation : Observation)

FIGURE 4.6: Visitor Pattern in VSLAM

From the general perspective of the Modular SLAM architecture, the most important design pattern is the strategy pattern. This pattern is crucial because it allows for the encapsulation of a family of algorithms, making it possible to interchange them seamlessly within the system. The strategy pattern is particularly pivotal for ensuring modularity, as it provides the flexibility to switch between different algorithms or behaviors based on varying requirements without altering the core architecture of the system. The adoption of the strategy pattern in Modular SLAM also enables the system to adapt dynamically to different environments or objectives by simply switching out the algorithmic strategies at runtime. For instance, in scenarios where precision is prioritized over speed, a more detailed but computationally intensive algorithm might be employed. Conversely, in time-critical applications, faster algorithms that trade off some accuracy for speed could be utilized. This capability not only enhances the system's versatility but also significantly boosts its efficiency and effectiveness in real-world applications.

The strategy pattern is used in Modular SLAM in many various areas. To give an example, let consider the pose estimation problem. In literature, there are many methods described to estimate the movement of the platform. That is why, it is useful to provide not only one single algorithm in the system, but rather specify the interface for the algorithm and decide which algorithm may be used in the process of VSLAM creation or in the runtime as it was stated.

The UML diagram of the strategy pattern for pose estimation used in Modular SLAM, as shown in Figure 4.7, illustrates how the `Frontend` component effectively utilizes the `PoseEstimationStrategy` interface to dynamically estimate poses based on sensor data. This modular approach allows the `Frontend` to switch between different pose estimation strategies without altering its underlying implementation, thereby enhancing both flexibility and functionality. In this setup, the `Frontend` class acts as a context in the strategy pattern, maintaining a reference to the `PoseEstimationStrategy` interface. This setup facilitates the easy switching of algorithms depending on the scenario's requirements or the specific characteristics of the input data. For example, while the `RansacPoseEstimation` might be suitable for environments with a high degree of visual noise, the `ReliableRansacPoseEstimation` could be optimized for scenarios where accuracy and reliability are paramount. The `Frontend` can switch between these strategies seamlessly during runtime, which is particularly beneficial in dynamic environments typical of SLAM applications. This strategy integration not only supports optimal performance adaptation but also encapsulates the pose estimation logic, keeping the system's architecture clean and maintainable.

4.3.3 Visualization

While working on the VSLAM system, visualization is critical for providing insights into the system's performance and current state. Effective visualization serves not only to display the final trajectory of the system but also to clarify the real-time operational status and data processing stages.

Visualization in Modular SLAM encompasses several key aspects, starting with the raw input from sensors. This initial visualization is fundamental as it allows for the assessment and verification of the data quality that the system processes, which is crucial for tuning and preprocessing steps. As

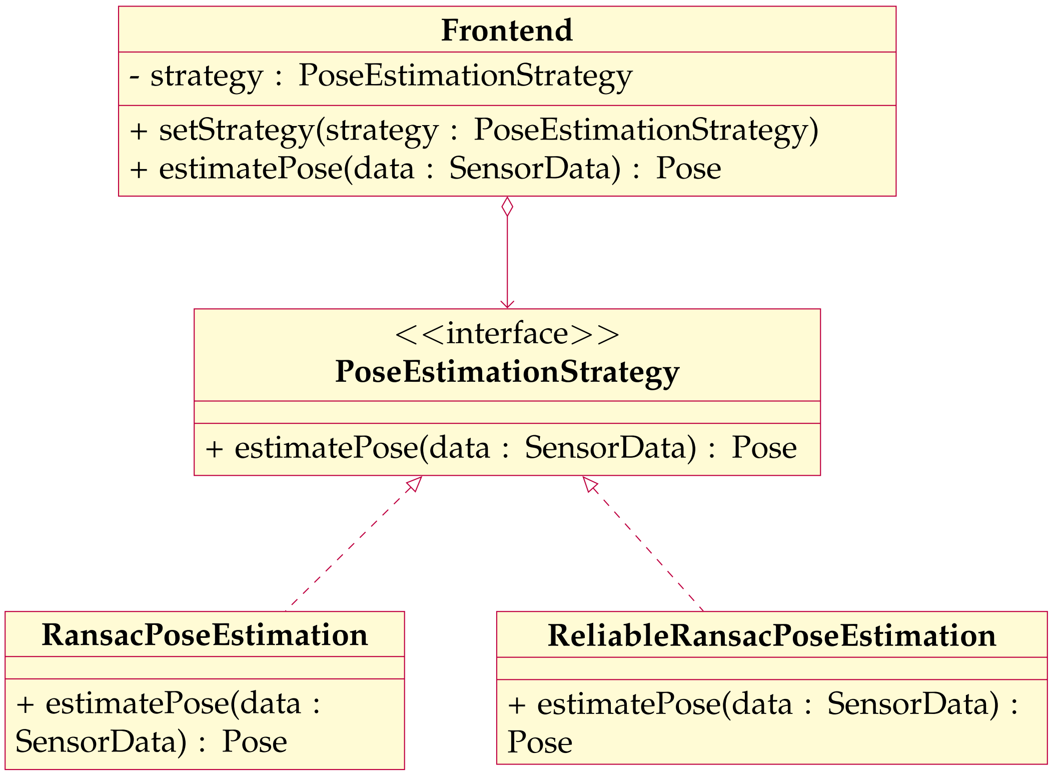

Frontend
- strategy : PoseEstimationStrategy
+ setStrategy(strategy : PoseEstimationStrategy)
+ estimatePose(data : SensorData) : Pose
<<interface>>
PoseEstimationStrategy
+ estimatePose(data : SensorData) : Pose
RansacPoseEstimation
+ estimatePose(data : SensorData) : Pose
ReliableRansacPoseEstimation
+ estimatePose(data : SensorData) : Pose

FIGURE 4.7: Strategy pattern for pose estimation in Modular SLAM

the system progresses, visualization extends to displaying the spatial relationships and features identified, such as landmarks and keyframes in 3D or positions of the keypoints on the original image. This not only helps in tracking the map construction over time but also provides essential feedback on the accuracy of localization and mapping efforts.

Additionally, the final trajectory visualization offers a detailed depiction of the system's movement through the environment, clearly illustrating the path taken based on the data collected. This aspect of visualization is significantly enhanced by using evo [Gru17], which provides comprehensive capabilities for trajectory analysis and comparison with ground truth. Such comparisons are essential for validating the performance of the system. Importantly, Modular SLAM is designed with versatility in mind, capable of conducting tests on established VSLAM systems like ORB-SLAM3 or StellaVSLAM. It can not only generate but also compare trajectories from these systems, establishing a robust framework for assessing and benchmarking various VSLAM algorithms. This ability to compare trajectories offers valuable insights into the effectiveness of different algorithms under similar conditions. Figure 4.8b showcases an example of a trajectory generated using Modular SLAM and the scripts provided, highlighting the practical application of these capabilities in real-world scenarios.

The visualization interface of Modular SLAM includes also a display of the current system settings, showcasing the parameters set for operation. This feature is crucial as it allows users to see and adjust settings in real-time, ensuring optimal system performance under varying conditions. Alongside parameter settings, the interface also presents a range of metrics related to real-time performance. These metrics include the time taken for pose estimation, mapping durations, and other pertinent data that provide insights into the system's efficiency. Displaying these metrics helps operators and researchers to monitor the system's performance closely and make informed decisions about potential adjustments or optimizations needed to improve accuracy and processing speed.

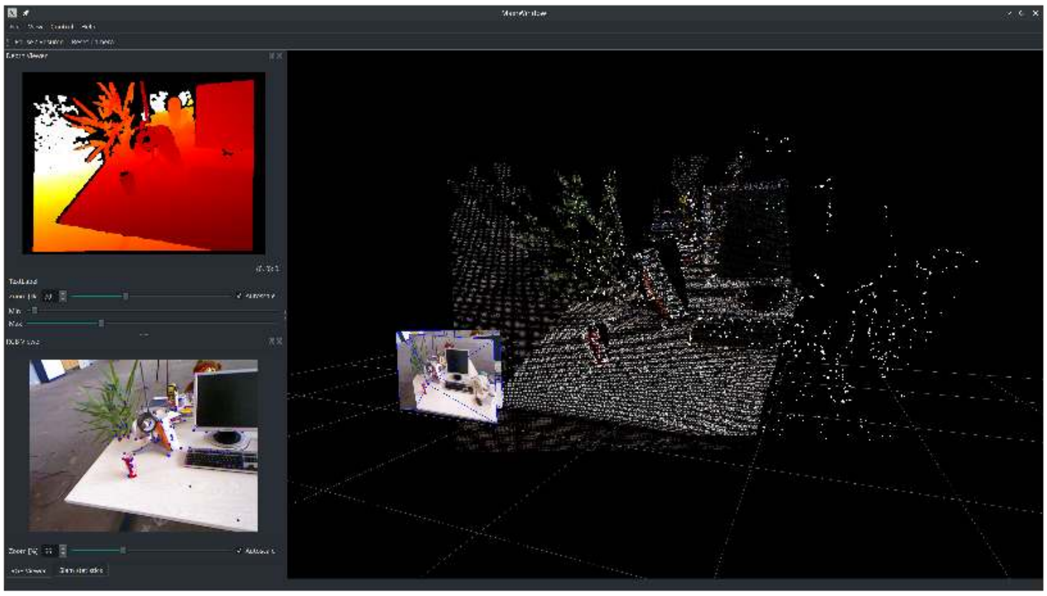
Integrating these visual tools within Modular SLAM ensures a multifaceted approach to system analysis and development. Effective visualization aids in immediate troubleshooting and enhancements and supports strategic development for future iterations of the system. The integrated viewer, as shown in Figure 4.8a, exemplifies how such tools are embedded within the Modular SLAM framework, facilitating both real-time adjustments and thorough post-operation analysis. This comprehensive approach not only enables detailed monitoring and modification of the system but also enriches research and practical applications.

4.3.4 Evaluation

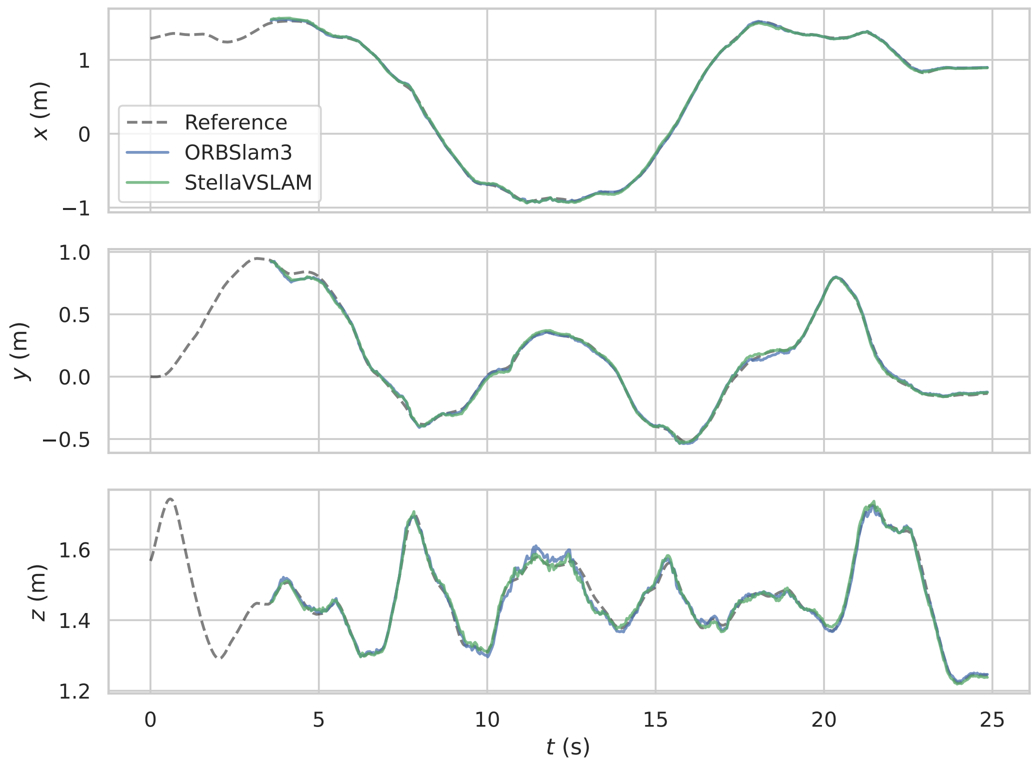
Evaluation is a critical component of Modular SLAM, playing a pivotal role due to its emphasis on iterative research and development. Modular SLAM incorporates several features specifically designed to facilitate thorough evaluation processes.

At the forefront, Modular SLAM integrates tools that enable comprehensive analysis of the platform's entire trajectory. This functionality is largely supported by the evo library [Gru17], but Modular SLAM extends these capabilities with significant enhancements. Not only does Modular SLAM provide the ability to calculate basic metrics, but it also allows for comparative analyses using other VSLAM systems. This feature is particularly valuable as it enables users to run benchmarks with other well-known systems such as StellaVSLAM, ORBSLAM, or LSD SLAM on the same dataset and machine. By doing so, researchers can directly compare the performance of the developed system against established alternatives, offering a robust framework for evaluating accuracy, efficiency, and other critical performance metrics.

Secondly, Modular SLAM includes a component specifically designed for storing trajectories, which can be utilized by various analysis tools within the community. In the field, there are primarily two recognized formats for representing the platform's trajectory: the KITTI format and the TUM format. Researchers can specify their preferred format effortlessly by selecting the



(A) Real-time visualization with Modular SLAM.



(B) Visualization with Modular SLAM and evo comparing ORB-SLAM and StellaVSLAM for TUM *rgbd_dataset_freiburg1_desk2* sequence.

FIGURE 4.8: Two aspects of Modular SLAM visualization: real-time operations and final plots.

appropriate concrete strategy for storing trajectories. This flexibility is made possible again through the implementation of the strategy pattern in Modular SLAM, allowing for easy adaptation to the diverse needs of different research projects or operational environments.

The strategy pattern's application is evident in how trajectory data management is handled within Modular SLAM. By abstracting the trajectory dumping process into a strategy interface, `TrajectoryDumper`, Modular SLAM permits seamless interchange between different storage strategies. Whether it is the `KittiLocalizationDumper` or the `TumLocalizationDumper`, each implements the `TrajectoryDumper` interface, ensuring that they can be used interchangeably without affecting the rest of the system's architecture. This approach not only simplifies the configuration and extension of the system but also enhances its robustness by encapsulating the functionality needed for trajectory storage within well-defined classes, as illustrated in the provided UML diagram in Figure 4.9.

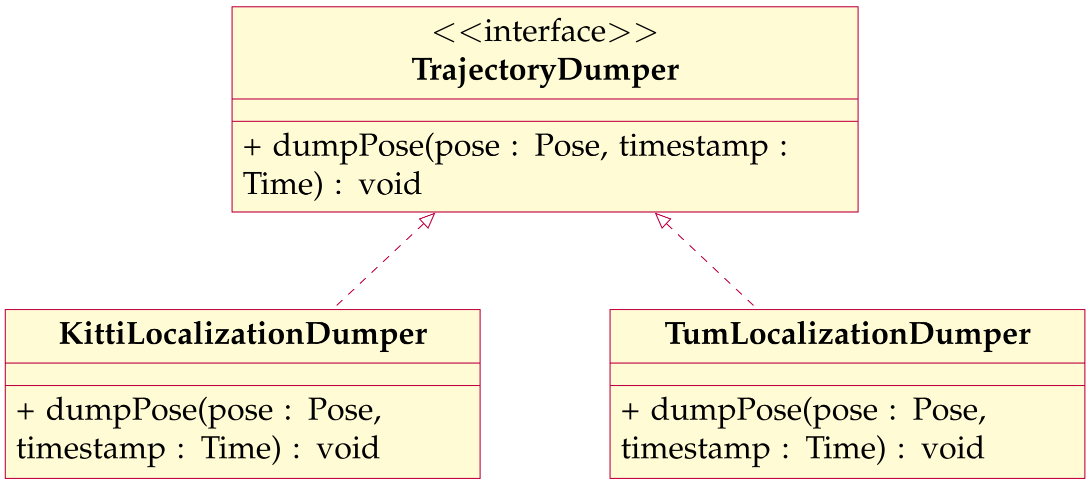

<<interface>>
TrajectoryDumper
+ dumpPose(pose : Pose, timestamp : Time) : void
KittiLocalizationDumper
+ dumpPose(pose : Pose, timestamp : Time) : void
TumLocalizationDumper
+ dumpPose(pose : Pose, timestamp : Time) : void

FIGURE 4.9: Strategy pattern for storing trajectory data in Modular SLAM

4.4 Practical Application of Modular SLAM

While a detailed exploration of the implementation lies beyond the primary focus of this thesis, it is essential to illustrate how the design patterns and principles discussed here find their practical application within the Modular SLAM framework. This section aims to briefly demonstrate the benefits of such approach by showing how they can be effectively implemented to enhance the functionality and flexibility of Modular SLAM system.

For a comprehensive understanding of the practical applications of the Modular SLAM, including detailed insights into its API and illustrative, extended C++ examples, it is strongly encouraged to consult its repository and

Appendix B. It is designed to bridge the gap between theoretical concepts discussed in the main text and their real-world implementations. It serves as a valuable resource for developers and researchers interested in applying the Modular SLAM to their projects, offering a hands-on approach that elucidates the system's functionality and adaptability. It not only enhances understanding but also provides practical tools and code snippets that can be directly utilized or adapted for various applications. By exploring these supplementary materials, a deeper understanding of the system's capabilities and the design considerations that underpin its architecture will be achieved.

To demonstrate how careful design significantly contributes to simplicity, both from a theoretical and practical perspective, let examine the creation process of a SLAM system. As previously discussed, this process is structured and streamlined through the use of the builder pattern. As illustrated in Listing 4.1, setting up the VSLAM system involves configuring several main components, a method that emphasizes clarity and ease of integration. The builder pattern facilitates a modular setup where components like the parameter handler, data provider, frontend, backend, and the map are methodically assembled. The Listing 4.1 shows how the `slamBuilder` object sequentially adds components, ensuring that each is correctly configured before it moves to the next. This not only streamlines the construction process but also allows for custom actions to be registered easily, such as setting the most recent frame or updating observations once the frontend processing completes.

```cpp
// ...
slamBuilder.addParameterHandler(std::make_shared<mslam::
    BasicParameterHandler>())
    .addDataProvider(dataProvider)
    .addFrontend(frontend)
    .addBackend(backend)
    .addMap(map)
    .registerDataFetchedAction([slamThread](std::shared_ptr<mslam::
    RgbdFrame> frame)
                                  { slamThread->setRecentFrame(frame);
    })
    .registerFrontendFinishedAction([slamThread](const
    FrontendOutputType& output) -> void
                                        { slamThread->
    setRecentObservations(output.landmarkObservations); });

auto slam = slamBuilder.build();
```

LISTING 4.1: Creating VSLAM system using Modular SLAM

Next, it was highlighted that strategy pattern is crucial for Modular SLAM functioning. This pattern is fundamental to achieving modularity and adaptability within the system, allowing for flexible adjustments and improvements in real-time. As demonstrated in Listing 4.2, the strategy pattern facilitates the use of various strategies for critical tasks including pose estimation, which can be configured and modified during runtime without disrupting the overall system's operations. The provided listing illustrates how different pose estimation strategies can be integrated into the system. For example, an instance of `mslam::OpenCvRansacPnp` is initially used for pose estimation and is attached to a frontend processing unit.

```cpp
auto poseEstimation = std::make_shared<mslam::OpenCvRansacPnp>();
auto frontend = std::make_shared<mslam::RgbdFeatureFrontend>(
      poseEstimation, /* ... */);

// or

frontend.setPoseEstimation(poseEstimation);
```

LISTING 4.2: Strategy pattern in Modular SLAM for pose estimation

The strategy pattern's utility is further exemplified in its application for saving trajectories in Modular SLAM systems. As detailed in the code snippet shown in Listing 4.3, the system can output trajectory data in two prominent formats: TUM and KITTI. Depending on the operational requirements or the preferred data analysis tools, the system can dynamically select either the `KittiLocalizationDumper` or the `TumLocalizationDumper` to handle the trajectory data. The listing further integrates the observer pattern by registering actions to be performed once the frontend processing finishes. Each action involves using the chosen strategy to write out the trajectory data, demonstrating a seamless combination of strategy and observer patterns. This dual-pattern approach not only simplifies the code by decoupling format-specific processing from the main SLAM logic but also ensures that the system's behavior can be modified without extensive changes.

The observer pattern plays a crucial role within the parameters handler framework, particularly in notifying relevant components about changes to specific parameters. The first part of the Listing 4.3 illustrates how several parameters of the frontend are registered. These parameters include the minimum number of matched keypoints required to maintain tracking without forcing relocalization and the minimum number of landmarks needed to trigger the creation of a new keyframe. When these parameters are set, other components can use the `registerParameter` method to subscribe to notifications of changes. This registration enables the components to execute specific actions in response to parameter modifications, effectively adapting their behavior based on the updated configuration.

```cpp
if(args.output->format == TrajectoryFileFormat::KITTI)
{
    auto dumper = std::make_shared<KittiLocalizationDumper>(args.
   output->trajectoryFilePath);
    slamBuilder.registerFrontendFinishedAction(
        [dumper](const auto& frontendOutput)
        {
            auto& dumperRef = *dumper;
            dumperRef(frontendOutput);
        });
}
else
{
    auto dumper = std::make_shared<TumLocalizationDumper>(args.
   output->trajectoryFilePath);
    slamBuilder.registerFrontendFinishedAction(
        [dumper](const auto& frontendOutput)
        {
            auto& dumperRef = *dumper;
            dumperRef(frontendOutput);
        });
}

mslam::SlamBuilder builder;
mslam::VSLAM vslam = builder.build();
```

LISTING 4.3: Combination of strategy and observer pattern for trajectory saving.

```cpp
// registering parameters of frontend
constexpr auto make_param = std::make_pair<ParameterDefinition,
   ParameterValue>;

const ParamsDefinitionContainer params = {
    make_param({"rgbd_feature_frontend/min_matched_points",
   ParameterType::Number, {}, {0, 100000, 1}}, 10.f),
    make_param({"rgbd_feature_frontend/new_keyframe_min_landmarks",
    ParameterType::Number, {}, {0, 10000, 1}},
               30.f)};

for(const auto& [definition, value] : params)
{
    parametersHandler->registerParameter(definition, value);
}

// registering observers
parametersHandler->registerObserver("rgbd_feature_frontend/
   min_matched_points", [](){std::cout <<"Parameter changed"});
```

LISTING 4.4: Handling parameters in Modular SLAM

Finally, the visualization aspect of the VSLAM system is thoroughly addressed. To enable real-time visualization of the VSLAM system, one simply needs to instantiate a `ViewerMainWindow` and a `SlamThread` to run the algorithm in the background. Once the SLAM system is activated, everything operates automatically, and the visualization appears as shown in Figure 4.8a.

```
mslam::ViewerMainWindow mainWindow;
SlamThread* slamThread = new SlamThread(&mainWindow);

auto slam = buildSlam(args, slamThread);
slamThread->setSlam(std::move(slam));
slamThread->start();
```

LISTING 4.5: Visualisation in Modular SLAM

The concept of Modular SLAM is not only powerful but also intentionally designed for ongoing development and maintenance by the scientific open-source community. It is anticipated that as this community continues to contribute, Modular SLAM will evolve to become even more robust and effective. Future enhancements and versions are expected to expand its capabilities and applications. This concludes the discussion on the concept of Modular SLAM. The next chapter will shift focus to additional contributions of this thesis, specifically exploring robust methods that further enhance system performance and reliability. It is worth noting that these methods are also integral components of the Modular SLAM framework.

Chapter 5

New VSLAM robust methods

Overview

This chapter provides an overview of the advanced computer vision algorithms implemented in ModularSLAM, specifically focusing on two robust methods: VSLAM SuperPoint and VSLAM RANSAC. These methods aim to significantly improve the accuracy and reliability of VSLAM in challenging environments marked by dynamic changes, variable lighting conditions, and significant occlusions.

VSLAM SuperPoint utilizes a deep learning approach to detect and describe features through a convolutional neural network, offering superior performance over traditional feature detection methods. Following this, the chapter explores VSLAM RANSAC, an improved variant of the traditional RANSAC algorithm that adapts sampling strategies and consensus thresholds dynamically to ensure efficient and accurate model estimation. Together, these methods contribute to the robustness of ModularSLAM, enabling better performance in real-world applications.

"A computer would deserve to be called intelligent if it could deceive a human into believing that it was human."
– Alan Turing

In this chapter, novel methodologies designed to enhance the robustness of feature-based VSLAM systems are introduced. The first part of the chapter is focused on addressing the challenges associated with keypoint detection, a critical component in VSLAM. Traditional approaches to keypoint detection have often struggled with robustness in dynamic and complex environments. To overcome these limitations, a new method that leverages AI and temporal information to improve the accuracy and reliability of keypoint detection is proposed. This approach not only enhances the detection process but also contributes significantly to the overall stability and performance of VSLAM systems. Through analysis and experimentation, it is demonstrated how these advancements can be integrated into existing frameworks to achieve superior robustness in various real-world scenarios. Moreover, each method is integrated with Modular SLAM.

5.1 Keypoints detections

In feature-based methods, keypoints play a crucial role in the entire VSLAM system. As discussed in Section 2.6.2, keypoints are not only used in pose estimation but are also fundamental building blocks of the map. Additionally, keypoint detection is the first task performed by the frontend, which is why it has a significant impact on both localization and mapping. Figure 5.1 shows several examples of detected keypoints in two consecutive frames. A brief analysis reveals several important observations. First, despite the initial appearance that two consecutive frames are almost identical, there are keypoints that are not detected in the subsequent frame. Furthermore, the distribution of keypoints is not always even across the images. In some cases, keypoints are concentrated in specific areas of the image, rather than being evenly distributed. These observations highlight the challenges and variability in keypoint detection, which can affect the robustness and accuracy of the VSLAM system.

A good feature detector is essential in VSLAM application. Let's describe the features of a good keypoint detector. One of the primary features of a good detector is repeatability. Repeatability measures the ability of the detector to consistently identify the same features across different frames or images. High repeatability ensures that these features can be reliably matched, which is crucial for applications requiring precise tracking and mapping.

Another critical attribute is robustness to changes in lighting conditions. A good feature detector should perform well regardless of variations in illumination, such as shadows, highlights, or overall brightness changes. This robustness is particularly important in real-world applications where lighting conditions can vary significantly, such as outdoor environments with

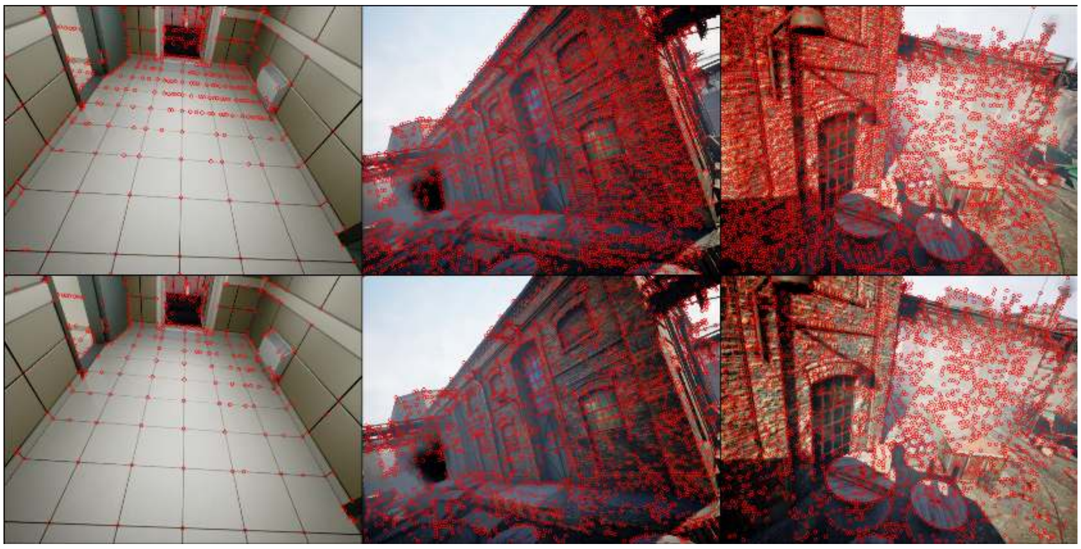


FIGURE 5.1: Detected keypoints on consecutive frames from a VSLAM sequence. Each red dot represents a keypoint detected in the image. The first row depicts the keypoints detected in the first frame, while the second row shows the keypoints detected in the subsequent frame. The keypoint detection algorithm used in this example is SuperPoint.

fluctuating sunlight or indoor settings with artificial lighting. By maintaining performance across different lighting scenarios, the detector can ensure consistent feature detection, which is vital for reliable visual processing.

In addition to repeatability and lighting robustness, a good feature detector should be invariant to changes in view angle. This means that the detector should be able to recognize the same feature even when observed from different perspectives. This invariance is crucial for applications like VSLAM, where the camera's viewpoint changes continuously as it moves through the environment. Furthermore, a good detector should be computationally efficient, allowing real-time processing on various hardware platforms, from powerful servers to mobile devices. Other desirable attributes include scale invariance, which ensures consistent detection across different scales, and robustness to noise, which helps maintain performance in noisy or cluttered environments. These features collectively contribute to a detector's overall effectiveness and reliability in diverse real-world scenarios.

In VSLAM systems, keypoint detection is executed for every camera frame to identify distinctive features in the environment. Consequently, if the system is not in relocalization state, it retains information about previously detected keypoints. However, the majority of widely-used feature-based VSLAM algorithms do not leverage this historical keypoint data. Instead, they treat each frame independently, often re-detecting keypoints from scratch. This approach can lead to inefficiencies and missed opportunities for improving the robustness and accuracy of the mapping and localization processes.

By integrating the information from previously detected keypoints, VSLAM systems could potentially enhance their performance, reduce computational overhead, and improve the consistency of keypoint tracking across frames.

In Figure 5.2 the gascola P001 sequence from the TartanAir dataset is illustrated. Each frame in this sequence displays keypoints that were initially detected in the first frame. The subsequent positions of these keypoints are computed based on their 3D coordinates and the ground truth camera poses. This particular sequence spans 32 frames, and it is evident that most of the keypoints detected in the first frame remain visible in the last frame. Nonetheless, due to disadvantages of both feature detector and specifically detected keypoints descriptors and feature matching algorithm the number of reoccurred keypoints is significantly lower.

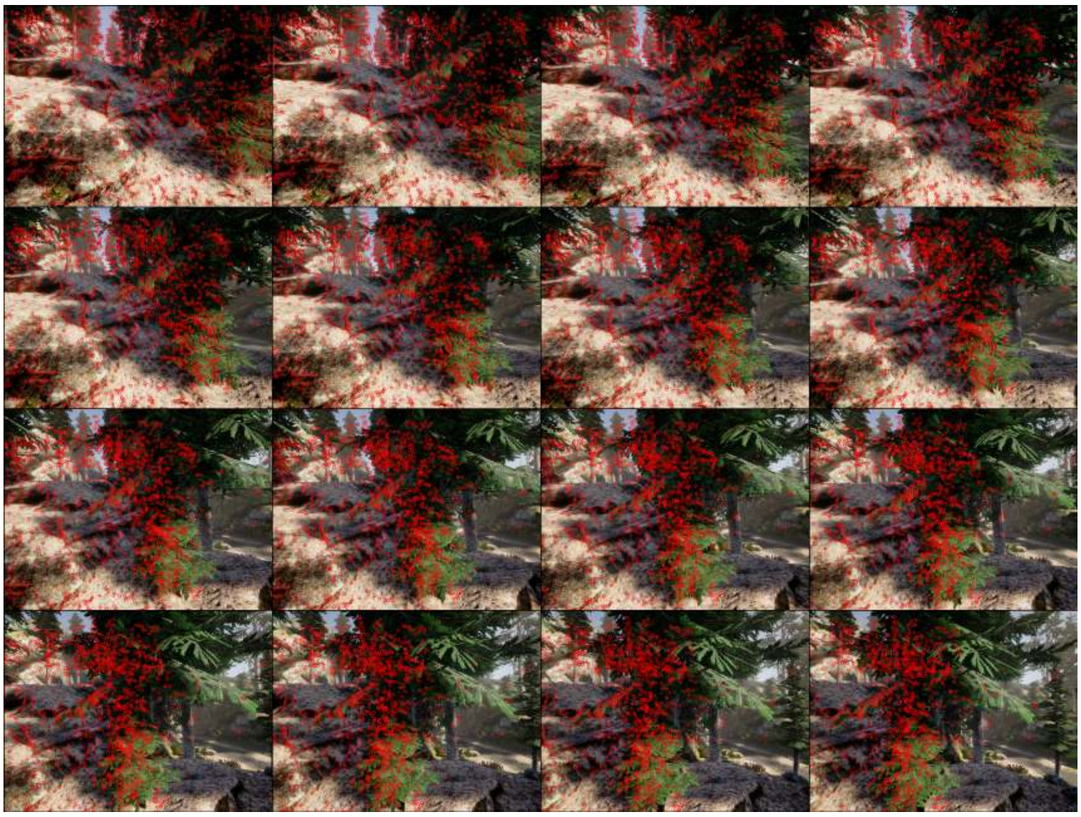


FIGURE 5.2: Illustration of a sequence from the TartanAir dataset, specifically the gascola P001 sequence. Keypoints shown on the following images are the keypoints detected in the first frame.

In Section 3.8, and particularly in Figure 3.13, an example of changing tracked landmarks was presented. This section continues and expands upon the topic of the number of tracked landmarks over time. Figure 5.3a presents the number of tracked landmarks in a another sequence. Additionally, the moving average $\overline{n}_i^{(l)}$ is expressed by Equation 5.1, where w represents the window length over which the average is computed.

$$\overline{n}_i^{(l)} = \frac{1}{w} \sum_{j=i-\left\lfloor \frac{w}{2} \right\rfloor}^{i+\left\lfloor \frac{w}{2} \right\rfloor} x_j \tag{5.1}$$

Figure 5.3b shows the detrended sequences of tracked landmarks. The detrended number of landmarks is given by Equation 5.2. The idea is straightforward. To identify local changes in the number of tracked landmarks, the weighted mean $\overline{n}_i^{(l)}$ is subtracted from the current number of tracked landmarks $n_i^{(l)}$.

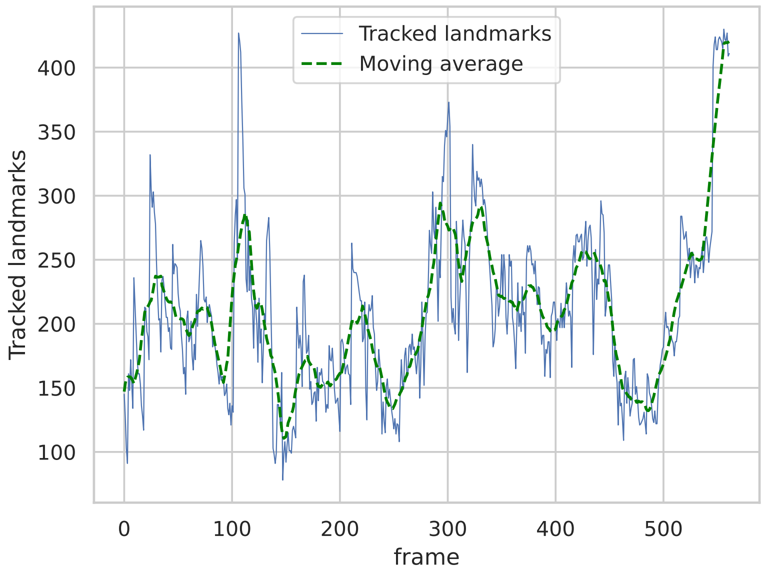
$$\tilde{n}_i^{(l)} = n_i^{(l)} - \overline{n}_i^{(l)} \tag{5.2}$$

In fact, changes in the number of tracked landmarks are expected, especially when the system is exploring new environments and creating map. However, if keypoints are being accurately detected and matched, the peaks visible on the plot should be less frequent and the slope more subtle. Based on this observation, several metrics can be defined, including standard deviation $\tilde{n}_\sigma^{(l)}$ and maximum value $\tilde{n}_{\max}^{(l)}$, given by Equation 5.3 and Equation 5.4 respectively.

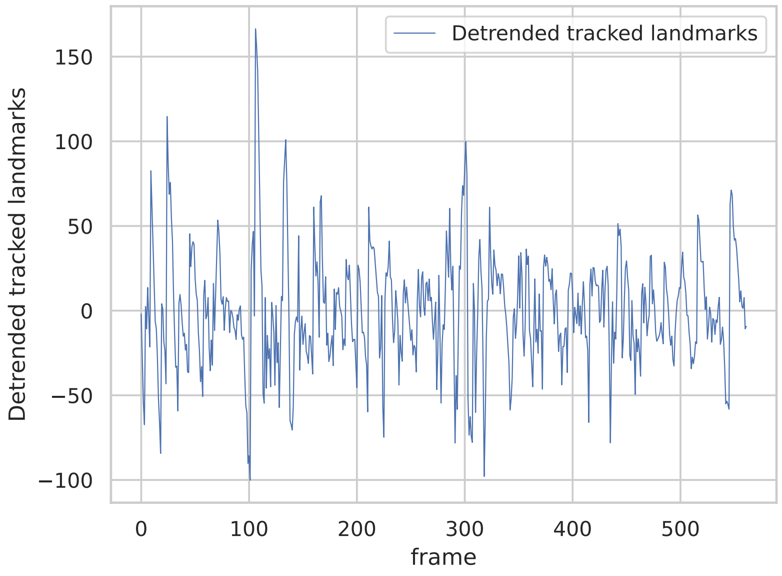
$$\tilde{n}_\sigma^{(l)} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(x_i - \frac{1}{w} \sum_{j=i-\left\lfloor \frac{w}{2} \right\rfloor}^{i+\left\lfloor \frac{w}{2} \right\rfloor} x_j \right)^2} \tag{5.3}$$

$$\tilde{n}_{\max}^{(l)} = \max\{\tilde{n}_0^{(l)}, \tilde{n}_1^{(l)}, \ldots\} \tag{5.4}$$

The standard deviation $\tilde{n}_\sigma^{(l)}$ and the maximum value $\tilde{n}_{\max}^{(l)}$ may be important metrics for evaluating the stability and variability of tracked landmarks. The standard deviation $\tilde{n}_\sigma^{(l)}$ measures the dispersion of the detrended number of tracked landmarks around the mean. A lower standard deviation indicates that the number of tracked landmarks is more consistent over time, suggesting robust keypoint detection and tracking. Conversely, a higher standard deviation points to greater variability, which may indicate issues with keypoint detection or changes in the environment. The maximum value $\tilde{n}_{\max}^{(l)}$, on the other hand, captures the largest deviation from the mean number of tracked landmarks. This metric highlights the most significant changes or anomalies in the number of tracked landmarks, providing insights into moments where the system may face challenges in maintaining consistent tracking. Together, these metrics help in assessing the performance and reliability of VSLAM feature-based systems.



(A) Tracked landmarks over time with visible peaks related to the finish of keyframe creation.



(B) Detrended tracked landmarks over time.

FIGURE 5.3: Tracked landmarks over time.

Table 5.1 presents the results for various sequences from the TUM RGB-D dataset, focusing on the number of tracked landmarks and their statistical measures for StellaVSLAM. The metrics include the mean number of tracked landmarks $n^{(l)}$, the standard deviation of tracked landmarks $n_\sigma^{(l)}$, the standard deviation of the detrended sequence $\tilde{n}_\sigma^{(l)}$, and the maximum value of the detrended sequence $\tilde{n}_{\max}^{(l)}$. Upon examining the data, it is evident that the sequences vary significantly in terms of every metric.

For instance, the *rgbd_dataset_freiburg2_xyz* sequence has the highest mean value at 558.692, indicating a dense tracking scenario, whereas *rgbd_dataset_freiburg1_desk2* has the lowest mean at 121.727, suggesting significantly fewer tracked landmarks. The standard deviation of the tracked landmarks $n_\sigma^{(l)}$ also shows considerable variation across sequences. Higher values, such as 209.272 for 'rgbd_dataset_freiburg2_rpy', indicate greater variability in the number of tracked landmarks, reflecting dynamic changes or challenges in the tracking process. Conversely, sequences like *rgbd_dataset_freiburg1_desk2* with a standard deviation of 66.319 suggest more consistent tracking. The detrended standard deviation $\tilde{n}_\sigma^{(l)}$ provides insights into local fluctuations, with values ranging from 13.498 in 'rgbd_dataset_freiburg2_rpy' to 51.085 in *rgbd_dataset_freiburg1_xyz*. Finally, the maximum value of the detrended sequence $\tilde{n}_{\max}^{(l)}$ shows the most significant deviations from the weighted mean. The highest peak is observed in *rgbd_dataset_freiburg1_xyz* at 235.250, signaling potential moments of significant change in the tracking environment. These metrics together reveal the dynamic nature of different sequences and the stability of tracked landmarks within them. Sequences with high variability and significant peaks may indicate more challenging tracking conditions, while those with lower variability and smoother changes suggest more stable environments.

5.1.1 SuperPoint for VSLAM

In VSLAM systems, images are sequentially processed, which is a fundamental aspect of how these systems operate. Despite this, the sequential nature of image processing is not typically leveraged to enhance robustness. The traditional approach in feature-based VSLAM systems begins with feature detection, followed by feature matching, and pose estimation based on these matches. Utilizing historical data, particularly the positional information of previously detected keypoints, could significantly improve the system's performance. It is anticipated that such information would prolong the tracking duration of individual landmarks, especially under challenging conditions.

SuperPoint, one of the most effective feature detectors identified in the literature, was proposed by DeTone, Malisiewicz, and Rabinovich [DMR18].

Sequence	$n^{(l)}$	$n_{\sigma}^{(l)}$	$\tilde{n}_{\sigma}^{(l)}$	$\tilde{n}_{\max}^{(l)}$
rgbd_dataset_freiburg2_desk	216.757	73.264	19.078	81.100
rgbd_dataset_freiburg2_rpy	389.744	209.272	13.498	65.900
rgbd_dataset_freiburg2_xyz	558.692	118.481	15.196	93.350
rgbd_dataset_freiburg1_rpy	246.823	142.775	36.566	112.300
rgbd_dataset_freiburg1_desk	209.011	67.249	34.105	166.450
rgbd_dataset_freiburg1_xyz	506.607	178.438	51.085	235.250
rgbd_dataset_freiburg1_desk2	121.727	66.319	25.768	108.350

TABLE 5.1: Tracked landmarks in various TUM sequences

This detector excels through its use of a deep learning framework to efficiently detect and describe keypoints, demonstrating considerable robustness across diverse lighting and environmental conditions. Such capabilities render it highly suitable for applications that demand both high accuracy and reliability in real-time feature tracking.

By integrating the positions of previously detected keypoints with a modified SuperPoint model, a novel feature detection algorithm can be developed suited for VO and VSLAM systems. Subsequent sections will detail the generation of datasets, provide an overview of the model, and discuss its evaluation.

Dataset generation

During the training stage of the VSLAM SuperPoint model, dataset generation plays a critical role. Each training sample is composed of two consecutive frames from the base sequences, including TartanAir. The reference keypoint positions are detected by the original SuperPoint model in the initial frame. However, the generation of ground-truth keypoint positions is not based on a single detection. Similar to the approach in the SuperPoint paper, homography adaptation was used. The difference is that a much more powerful model is employed as the base detector.

These keypoints are then tracked, and their positions are calculated in the subsequent frame. This process ensures that the model learns to accurately detect and track keypoints across frames, enhancing its robustness and performance. The quality and diversity of the generated dataset significantly impact the model's ability to generalize to different scenarios, making careful dataset preparation essential for effective training. Since sample preparation involves several steps, the following paragraphs will describe this process in detail.

The first step is to retrieve the list of all images and their corresponding poses. In other words, every camera frame has its own pose $\mathbf{C}_n$ represented as a translation $\mathbf{t}_n$ vector and quaternion $\mathbf{q}_n = q_0 + q_1 i + q_2 j + q_2 k$, which is shown in Equation 5.5.

$$\mathbf{C}_n = \begin{bmatrix} \mathbf{t}_n \\ \mathbf{q}_n \end{bmatrix} = \begin{bmatrix} x & y & z & q_0 & q_1 & q_1 & q_1 \end{bmatrix}^T \tag{5.5}$$

For further transformations, it is necessary to convert the rotation representation from a quaternion to a more convenient rotation matrix. This can be achieved using the Rodriguez formula, as given by Equation 5.6. The quaternion to rotation matrix conversion allows for more straightforward manipulation.

$$\mathbf{R}_n = \begin{bmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & 1 - 2(q_1^2 + q_3^2) & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & 1 - 2(q_1^2 + q_2^2) \end{bmatrix} \tag{5.6}$$

Given matrix $\mathbf{R}_i$ and translation vector $\mathbf{t}_i$, a transformation $\mathbf{T}_i$ can be defined, which represents the transformation from global coordinate system to the i-th pose. It is given by Equation 5.7.

$$\mathbf{T}_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0} & 1 \end{bmatrix} \tag{5.7}$$

In the next step, it is necessary to calculate the 3D positions of the landmarks associated with the detected keypoints. According to the pinhole camera model, obtaining the image position of a 3D point in the camera coordinate system requires multiplying the intrinsic matrix $\mathbf{K}$ by the 3D coordinates of the point. The intrinsic matrix $\mathbf{K}$ which is shown in Equation 5.8 encapsulates the camera's internal parameters, including the focal lengths f_x and f_y, and the optical center coordinates c_x and c_y.

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{5.8}$$

These parameters are determined through a camera calibration procedure described in [Zha00; KB17]. During this calibration process, the matrix $\mathbf{K}$ is estimated, which allows for accurate projection of 3D points onto the 2D image plane. The formula used for this transformation is given by Equation 5.9, where $\mathbf{p}$ represents the 2D coordinates of the point on the image, and $\mathbf{l}$ is a 3D point in the camera coordinate system, where $s \in \mathbb{R}$.

$$\mathbf{p} = s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{l} = \mathbf{K} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{5.9}$$

The transformation given by Equation 5.9 can be explicitly expressed, assuming that the distance of the point to the camera is not equal to zero i.e. $z_c \neq 0$. The relationship is described by the following equations:

$$\begin{bmatrix} u \\ \\ v \end{bmatrix} = \begin{bmatrix} \frac{x_c}{z_c} f_x + c_x \\ \\ \frac{y_c}{z_c} f_y + c_y \end{bmatrix} \tag{5.10}$$

However, the inverse process is required. To be more precise, $\mathbf{l}$ has to be calculated. Thus, based on Equation 5.10 the inverse process may be expressed with Equation 5.11.

$$\mathbf{l} = \begin{bmatrix} x_c \\ \\ y_c \\ \\ z_c \end{bmatrix} = \begin{bmatrix} (u - c_x) z_c f_x^{-1} \\ \\ (v - c_y) z_c f_y^{-1} \\ \\ z_c \end{bmatrix} \tag{5.11}$$

Since the 3D point $\mathbf{l}$ is initially represented in the coordinate system of the i-th camera, it is necessary to transform this point into the coordinate system of the j-th camera for further processing. This transformation is achieved by combining the transformation matrices of both camera poses. Specifically, the transformation matrix of the i-th camera is multiplied by the inverse of the transformation matrix of the j-th camera. This transformation ensures that the 3D point is accurately referenced in the new coordinate system, as shown in Equation 5.12.

$$\mathbf{T}_{i,j} = \mathbf{T}_i \mathbf{T}_j^{-1} \tag{5.12}$$

By combining all transformations which were discussed previously, the keypoint's position $\mathbf{p}_j$ on the j-th image is given by Equation 5.13.

$$\mathbf{p}_j = \begin{bmatrix} \mathbf{K} & \mathbf{0} \end{bmatrix} \mathbf{T_{i,j}} \begin{bmatrix} \mathbf{K}^{-1} & \mathbf{0} \end{bmatrix} \mathbf{p}_i = \begin{bmatrix} \mathbf{K} & \mathbf{0} \end{bmatrix} \mathbf{T_{i,j}} \begin{bmatrix} \mathbf{K}^{-1} & \mathbf{0} \end{bmatrix} \mathbf{p}_i \tag{5.13}$$

However, there is still one significant problem that needs to be solved. The procedure of finding the position of a previously seen point is not valid due to the fact that occlusion may occur. Occlusion happens when an object obstructs the line of sight between the camera and the point of interest, making it impossible to detect the point from a certain viewpoint. This phenomenon can significantly impact the accuracy and reliability of visual tracking systems, such as those used in VSLAM, where maintaining continuous visibility of landmarks is crucial for precise mapping and localization. Figure 5.4 illustrates the concept of occlusion from a top-down view. Two rectangular objects of different sizes are positioned in the scene. In the first camera position, the line of sight to a specific point is unobstructed, allowing the camera to detect the point clearly. This is depicted by the dashed lines connecting the first camera to the point. In the second camera position, the larger rectangular object blocks the line of sight, causing occlusion. The dashed lines from the second camera to the point intersect another object, indicating that the point cannot be seen from this position. This visual representation underscores the challenges posed by occlusion in maintaining continuous observation of landmarks and highlights the need for robust algorithms to handle such scenarios effectively.

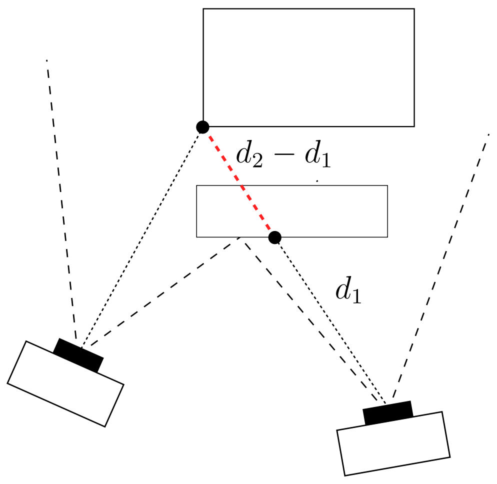
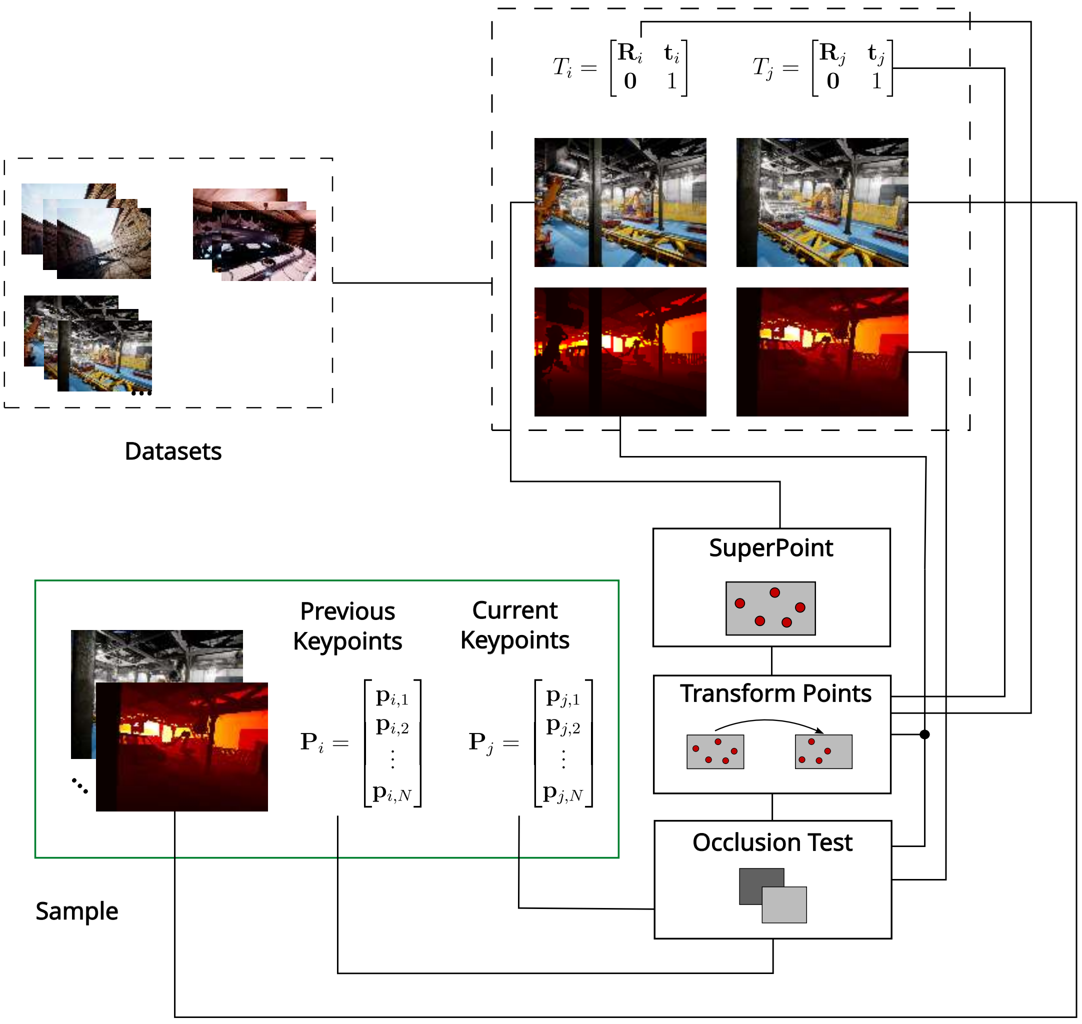


FIGURE 5.4: Top-down view illustrating the concept of occlusion.

In feature detection, occlusion presents a significant problem because the descriptor of the point can change drastically when it becomes associated with a different object. To address issues related with occlusion, a straightforward method is employed. As illustrated in Figure 5.4, occlusion is detected by comparing the distance between the camera and the point with the distance observed by the depth sensor. If the difference between these distances exceeds a certain threshold ϵ, occlusion is identified. The occlusion detection function is defined as follows:

$$f(d_1, d_2) = \begin{cases} 0, & \text{if } |d_2 - d_1| \leq \epsilon \\ 1, & \text{if } |d_2 - d_1| > \epsilon \end{cases} \tag{5.14}$$

In this equation, d_2 represents the distance from the camera to the point as



$T_i = \begin{bmatrix} \mathbf{R}_i & \mathbf{t}_i \\ \mathbf{0} & 1 \end{bmatrix}$
$T_j = \begin{bmatrix} \mathbf{R}_j & \mathbf{t}_j \\ \mathbf{0} & 1 \end{bmatrix}$
Datasets
SuperPoint
Previous Keypoints
Current Keypoints
$\mathbf{P}_i = \begin{bmatrix} \mathbf{p}_{i,1} \\ \mathbf{p}_{i,2} \\ \vdots \\ \mathbf{p}_{i,N} \end{bmatrix}$
$\mathbf{P}_j = \begin{bmatrix} \mathbf{p}_{j,1} \\ \mathbf{p}_{j,2} \\ \vdots \\ \mathbf{p}_{j,N} \end{bmatrix}$
Transform Points
Occlusion Test
Sample

FIGURE 5.5: Visualisation of training dataset preparation.

measured by the camera's intrinsic parameters, and d_1 is the observed distance from the depth sensor. The function $f(d_2, d_1)$ outputs 0 when the absolute difference between d_2 and d_1 is within the tolerance ϵ, indicating no occlusion. Conversely, it outputs 1 when the difference exceeds ϵ, indicating that occlusion is likely occurring.

To summarize the entire process, the diagram in Figure 5.5 illustrates all key steps, including feature detection, point transformation, and occlusion testing. It is important to note that the training samples comprise not only the images but also the keypoints detected on previous image and visible in the current one. As detailed in Section 5.1.1, the training stage allows for the calculation of various metrics related to feature detection. These metrics evaluate the effectiveness and accuracy of the feature detection process, essential for optimizing the performance of the VSLAM system. While the diagram effectively visualizes the entire process, supplementing it with code listings provides a more detailed description of the proposed method. Listing 5.1 details the initial steps of the dataset generation process. These steps include selecting pairs of images from the sequence, preparing transformation matrices, detecting keypoints, and processing their transformations. Given the complexity of the keypoint transformation procedure, Listing 5.2 specifically illustrates the steps involved in mapping the keypoints detected on the previous image to their new positions on the current image as well as the occlusion test. This detailed approach enhances understanding and provides a practical insight into the implementation of the method.

```
algorithm GenerateSample(images, depthMaps, poses,
                         index, step)
  $\mathbf{q}_i$, $\mathbf{t}_i$ = poses[index]
  j = i - step
  $\mathbf{q}_j$, $\mathbf{t}_j$ = poses[j]

  $\mathbf{R}_i$ = ToRotationMatrix($\mathbf{q}_i$)
  $\mathbf{R}_j$ = ToRotationMatrix($\mathbf{q}_j$)
  $\mathbf{T}_i$ = ToTransformationMatrix($\mathbf{R}$, $\mathbf{q}_i$)
  $\mathbf{T}_j$ = ToTransformationMatrix($\mathbf{R}$, $\mathbf{q}_j$)

  currentDepthMap = depthMaps[i]
  previousDepthMap = depthMaps[j]
  keypoints = DetectKeypoints(images[j])
  transformedKeypoints = TransformKeypoints(keypoints,
                                            previousDepthMap,
                                            currentDepthMap,
                                            $\mathbf{T}_i$, $\mathbf{T}_j$)
  return images[i], transformedKeypoints
end algorithm
```

LISTING 5.1: Algorithm of generating samples for VSLAM SuperPoint.

The proposed method can be viewed as an alternative to the approach presented in [DMR18]. In that work, the authors use pairs of images as training samples: one original image and another warped by a homography transformation. The loss function during training is designed to ensure consistent feature detection across these two images. However, the approach introduced in this thesis takes a different direction. VSLAM SuperPoint is primarily focused on feature detection in sequences that reflect real-world conditions.

One key distinction is that the homography-based approach does not account for real-world phenomena such as occlusions, which are common in VSLAM scenarios. By contrast, the method proposed here is specifically designed to handle these complexities, making it better suited for robust feature

```
algorithm TransformKeypoints(keypoints, previousDepthMap,
   currentDepthMap, $\mathbf{T_i}$, $\mathbf{T_j}$)

  $\mathbf{T}_{i,j}$ = $\mathbf{T}_i \mathbf{T}_j^{-1}$
  previousDepths = currentDepthMap[keypoints]
  depths = previousDepthMap[keypoints]

  transformation = $\begin{bmatrix}\mathbf{K} & \mathbf{0}\end{bmatrix} \mathbf{T_{i,j}} \begin{bmatrix}\mathbf{K}^{-1} & \mathbf{0}\end{bmatrix}$
  transformedKeypoints = transformation * keypoints
  mask = TestOcclusion(depth, transformedKeypoints[:, 2])

 return transformedKeypoints[mask]
end algorithm
```

LISTING 5.2: Algorithm for transforming keypoints from previous frame to current frame.

detection in VSLAM applications where environmental factors like occlusions play a significant role. As demonstrated in Section 5.1.1, this approach results in a loss function that is better aligned with the challenges faced in practical VSLAM systems.

Model overview

As stated in the introduction of VSLAM SuperPoint, the core concept involves enhancing the SuperPoint model by expanding its capabilities. Instead of solely processing single images, the updated model incorporates knowledge of keypoints detected in previous images.

The SuperPoint architecture, designed by DeTone, Malisiewicz, and Rabinovich, is straightforward yet effective, catering specifically to real-time applications that demand both speed and power. As illustrated in Figure 5.6, the architecture comprises three primary components. Initially, there is a shared encoder that reduces the image dimensionality. Subsequently, the encoder's output is directed along two parallel paths: the interest point decoder and the descriptor decoder. The interest point decoder generates a heatmap to pinpoint keypoint locations, while the descriptor decoder produces the corresponding descriptors [DMR18].

To illustrate how keypoints are identified using the SuperPoint model, an inference was performed, and the results are showcased in Figure 5.7. The process starts with a heatmap that indicates potential positions for the keypoints, displayed in Figure 5.7a. This is followed by preprocessing steps including non-maximum suppression to refine these positions, removal of keypoints close to image borders, and extraction of the top-k keypoints if

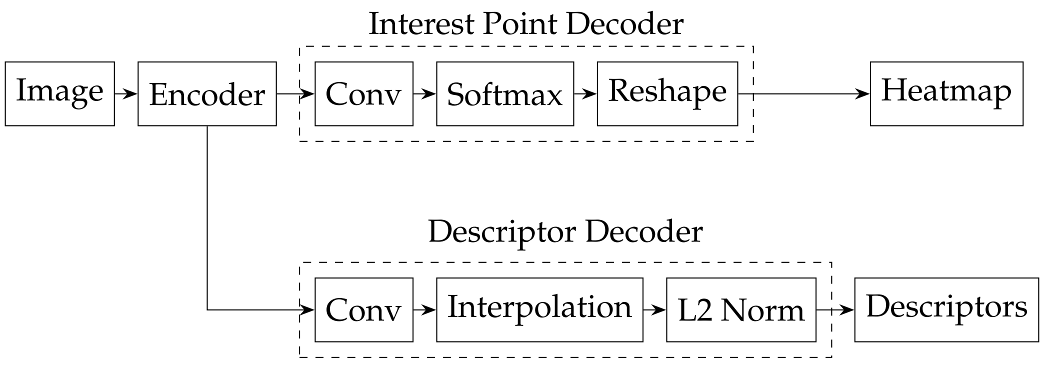
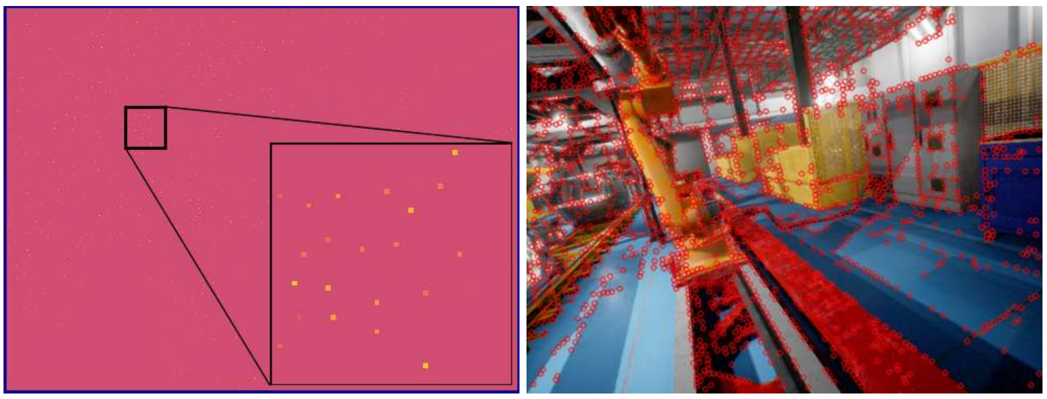


FIGURE 5.6: SuperPoint architecture

specified. The final keypoints are then drawn on the original image, as illustrated in Figure 5.7.



(A) Heatmap generated by the SuperPoint model. (B) Final keypoints marked on the original image.

FIGURE 5.7: Determining keypoints from heatmap.

The VSLAM SuperPoint represents a refined iteration of the original SuperPoint model, maintaining its core functionality and concept. It still processes a current image to output heatmaps and point descriptors. However, the VSLAM SuperPoint incorporates major architectural modifications to utilize historical data from previously detected keypoints. The most notable change is in the model's input, which now includes two distinct elements i.e. image and heatmap representing positions of previously detected keypoints. This change has an impact on the other part of the model. Original SuperPoint does not process the previous heatmap. That is why it is also required to provide the way to integrate that data into the pipeline of the model. There are numerous ways to accomplish that goal. However, it would be worth to use the learned weights of the original model to reduce the number of required training samples and as a consequence the time of learning stage.

To enhance the interest point decoder, modifications were made to the original architecture. After the initial interest point detector produces a heatmap $\hat{H} \in \mathcal{R}^{W \times H}$, additional layers were introduced. Specifically, to integrate information from both current and previously detected keypoints, two heatmaps are stacked together and fed into the subsequent layer. That layer is a 3D convolutional layer, whose primary role is to refine the final heatmap. This refinement ensures that the knowledge of previously detected keypoints enhances their detection in the current frames. The complete, refined architecture of the model is depicted in Figure 5.8.

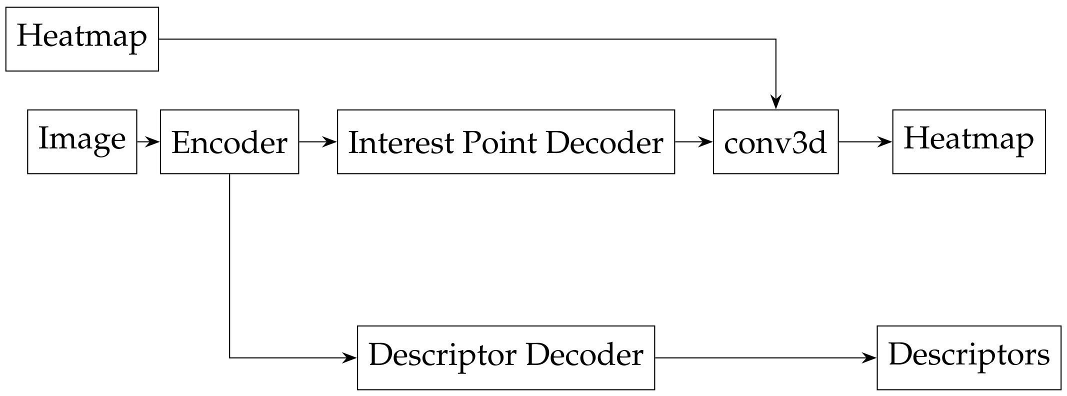


FIGURE 5.8: VSLAM SuperPoint architecture

It is also important to highlight that providing previous heatmap is not always possible. In scenarios such as initialization, relocalization, or other situations that disrupt sequential data continuity, the heatmap is unavailable. Consequently, it is essential for the model to be capable of accurately detecting keypoints even in the absence of prior heatmap data. In these cases, the model's input heatmap is filled entirely with ones, signaling that every position could potentially contain a keypoint. This approach uses the heatmap as a guide to direct the search for keypoints previously detected, thereby enhancing the model's tracking capability over longer sequences.

Training stage

In the training stage of AI models, several critical aspects must be addressed: the dataset, the model, the loss function, and the optimization algorithm. The first twodataset and model—have already been discussed in the previous sections. Now, let's turn the attention to the remaining two: the loss function and the optimization algorithm.

The primary goal of VSLAM SuperPoint is to enhance the tracking ability of landmarks. However, this is not the only objective. A good feature detector must meet several criteria. As discussed in Section 3.1, the distribution of keypoints across the image significantly impacts the accuracy of pose estimation. Additionally, the quality of descriptors plays a crucial role; consistent or similar descriptor values for a given keypoint across consecutive frames are essential for reliable feature matching, which in turn affects pose estimation accuracy. These considerations lead to a multi-objective loss function $\mathcal{L}$ as defined in Equation 5.15. The loss function is composed of two components, each representing a different aspect of the overall objective. The terms are weighted by λ_d and λ_p. The first component correspond to the loss proposed in the original SuperPoint paper, while the detector loss is modified version to align with the additional layers.

$$\mathcal{L} = \lambda_d \mathcal{L}_d + \lambda_p \mathcal{L}_p \tag{5.15}$$

In this equation, $\mathcal{L}_d$ represents the loss associated with descriptors. It is important that the descriptor of a keypoint remains consistent across two images. This descriptor loss is defined in Equation 5.16.

$$\mathcal{L}_d = \frac{1}{(H_c W_c)^2} \sum_{h=1}^{H_c} \sum_{w=1}^{W_c} \sum_{h'=1}^{H_c} \sum_{w'=1}^{W_c} l_d(\mathbf{d}_{hw}, \mathbf{d}'_{h'w'}; s_{hwh'w'}) \tag{5.16}$$

where,

$$\begin{aligned} l_d(\mathbf{d}, \mathbf{d}', s) = \lambda_s \cdot s \cdot \max(0, m_p - \mathbf{d}^T \mathbf{d}') + \\ (1 - s) \cdot \max(0, \mathbf{d}^T \mathbf{d}' - m_n) \end{aligned} \tag{5.17}$$

In this equation, H_c and W_c denote the height and width of the feature map, respectively, which define the resolution of the grid over which the descriptors are computed. The total number of descriptors in each feature map is therefore $H_c \times W_c$. The summation is performed over all pairs of descriptors from two feature maps: the original and the corresponding one from the next frame.

The function $l_d(\mathbf{d}_{hw}, \mathbf{d}'_{h'w'}; s_{hwh'w'})$ quantifies the difference between a pair of descriptors $\mathbf{d}_{hw}$ and $\mathbf{d}'_{h'w'}$ located at positions (h, w) and (h', w') in the respective feature maps. This difference is weighted by the matching score $s_{hwh'w'}$, which indicates the likelihood that these descriptors correspond to the same keypoint in both images. λ_s is a weighting factor that balances the influence of the positive and negative terms, s is the matching score between the descriptors $\mathbf{d}$ and $\mathbf{d}'$. m_p is a margin that defines the desired similarity for a positive match (i.e., when the descriptors should match) andm_n is a margin for a negative match. The loss function l_d comprises two hinge loss terms. The first term, $\max(0, m_p - \mathbf{d}^T \mathbf{d}')$, penalizes the model when the similarity between matching descriptors falls below the margin m_p. The second term, $\max(0, \mathbf{d}^T \mathbf{d}' - m_n)$, penalizes the model when the similarity between non-matching descriptors exceeds the margin m_n. The overall descriptor loss $\mathcal{L}_d$ is the average of these individual losses over all descriptor pairs in the feature maps.

Next, $\mathcal{L}_p$ is the loss related to the interest point detector, as defined in

Equation 5.18. It is a binary cross-entropy, where $H_{i,j}$ is a groundtruth of (i, j) pixel and $\hat{H}_{i,j}$ is a corresponding value of the detector model's output.

$$\mathcal{L}_p = -\frac{1}{HW}\sum_{i=1}^{H}\sum_{j=1}^{W}\left[H_{ij}\log(\hat{H}_{ij}) + (1 - H_{ij})\log(1 - \hat{H}_{ij})\right] \tag{5.18}$$

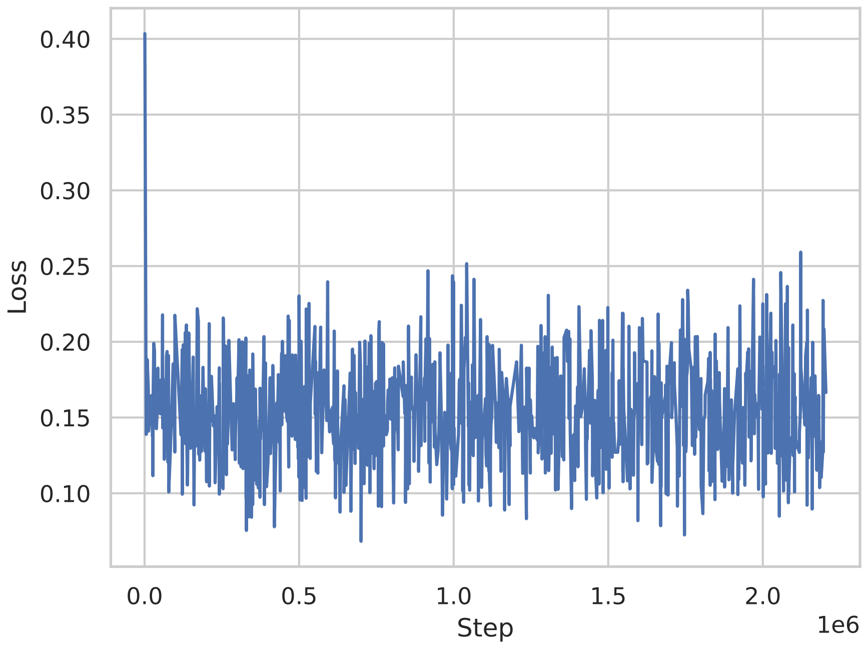
The model and training process were implemented using the PyTorch Lightning framework [FT19], leveraging a loss function and a dataset generated through the proposed method. Notably, instead of training the model entirely from scratch, transfer learning techniques were employed [Yan+20] to enhance the efficiency and effectiveness of the training process.

The outcomes of the training are depicted in Figure 5.9. In particular, Figure 5.9a shows the training loss curve, while Figure 5.9 demonstrates how the detected keypoints evolved throughout the training process. It illustrates the progression of the detector's learning process across various training steps. In the early stages, specifically at training step 600, the heatmap generated by the detector is diffused and unfocused, indicating that the model has not yet learned to localize keypoints precisely. As training progresses, particularly by step 1700, the heatmap begins to show more concentrated areas, but it still lacks precision. By training step 4200, the heatmap demonstrates a noticeable improvement, with keypoints becoming more distinct and localized. Finally, at training step 9800, the heatmap is much more precise, with the detector accurately focusing on specific keypoints, reflecting the model's learned ability to detect features with high accuracy. This progression highlights the refinement of the model's feature detection capabilities as it undergoes training.

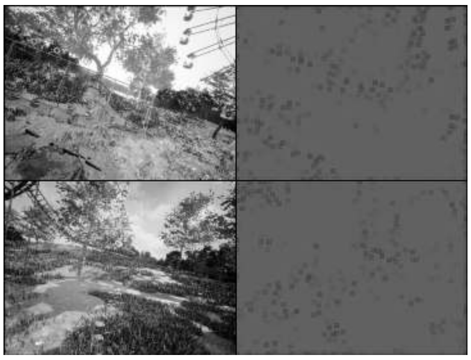
Evaluation

The evaluation process focuses on two key aspects: repeatability and tracking length. First, the repeatability r of the keypoints was measured. In this context, repeatability is defined as the ratio of the number of keypoints successfully tracked between two consecutive images. This metric is crucial as it reflects the stability and reliability of the keypoint detection process across frames. Repeatability is calculated using Equation 5.19, where N_{matches} represents the number of keypoints detected on both consecutive frames, and $N^{(1,2)}_{\text{keypoints}}$ denotes the number of keypoints detected on the first frame that are still visible and trackable in the subsequent frame.

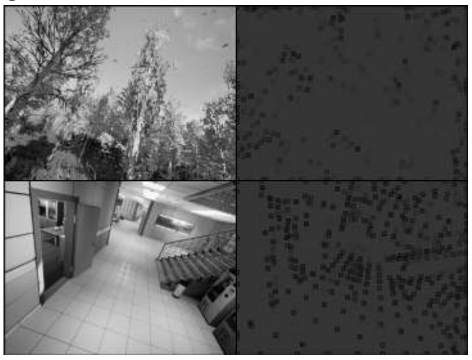
$$r = \frac{N_{\text{matches}}}{N^{(1,2)}_{\text{keypoints}}} \tag{5.19}$$



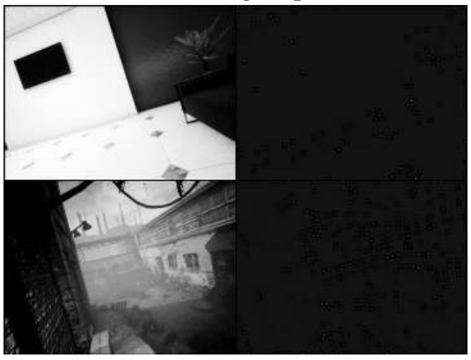
(A) Training loss



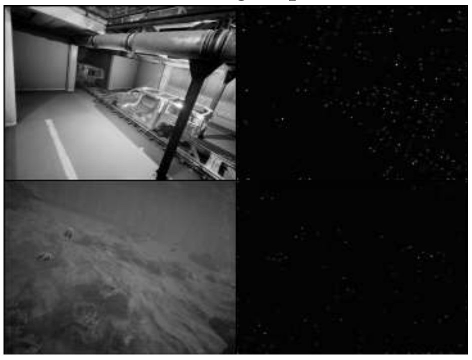
(B) Training step 600



(C) Training step 1700



(D) Training step 4200



(E) Training step 9800

FIGURE 5.9: Changing details of heatmap over the training.

To obtain a comprehensive evaluation, the repeatability r_i was measured for each i-th pair of consecutive frames in the dataset. The results were then analyzed by calculating the average, standard deviation, and minimum values of repeatability across all N_f frame pairs, as given by Equation 5.20, Equation 5.21, and Equation 5.22, respectively.

$$\bar{r} = \frac{1}{N_f} \sum_{i=1}^{N} r_i \tag{5.20}$$

$$\sigma_r = \sqrt{\frac{1}{N_f} \sum_{i=1}^{N} (r_i - \bar{r})^2} \tag{5.21}$$

$$r_{\min} = \min(r_1, r_2, \ldots, r_{N_f}) \tag{5.22}$$

Next, the tracking length was analyzed to evaluate the persistence of keypoints across consecutive frames within the dataset. Given a set of N_s sequences, each consisting of N_w consecutive frames, the tracking process was initiated for N_k keypoints detected in the first frame of each sequence. For each initial keypoint, the tracking length $L_{k,s}$ is defined as the maximum index of consecutive frames where the keypoint could be successfully tracked, up to the first instance where tracking is lost. This means that if tracking is lost at the 5th frame, $L_{k,s}$ is assigned the value 5, even if the keypoint is successfully tracked again in subsequent frames. The tracking length for a single keypoint is calculated as follows:

$$L_{k,s} = \max \{ j \mid r_i > 0 \text{ for all } i \in \{1, 2, \ldots, j\} \} \tag{5.23}$$

where r_i indicates whether the keypoint is successfully tracked between the i-th and $(i+1)$-th frames (i.e., $r_i = 1$ if the keypoint is tracked, and $r_i = 0$ otherwise).

To evaluate the overall performance, the average tracking length across all keypoints and sequences is calculated as:

$$\overline{L} = \frac{1}{N_s \cdot N_k} \sum_{s=1}^{N_s} \sum_{k=1}^{N_k} L_{k,s} \tag{5.24}$$

where $\overline{L}$ represents the average tracking length across all sequences and keypoints. The standard deviation of the tracking length, which measures the variability of the tracking performance, is given by:

$$\sigma_L = \sqrt{\frac{1}{N_s \cdot N_k} \sum_{s=1}^{N_s} \sum_{k=1}^{N_k} (L_{k,s} - \overline{L})^2} \tag{5.25}$$

These metrics provide a comprehensive evaluation of keypoint tracking performance across the test data. By analyzing these values, the robustness of the VSLAM SuperPoint model in maintaining accurate and consistent tracking across different sequences is better understood.

All results were collected in Table 5.2, which presents the performance on a test dataset that is part of the TartanAir dataset [Wan+20] which is considered as a challenging one. The results clearly show that VSLAM SuperPoint outperforms traditional detectors commonly used in popular VSLAM systems like ORB-SLAM3 [Cam+21] and Stella VSLAM [SSS19] in terms of the repeatability.

The deep learning-based detector outperforms traditional methods across nearly every metric, demonstrating superior repeatability and tracking length. These results strongly indicate that incorporating deep learning techniques into feature detection can significantly enhance the robustness of VSLAM systems. Furthermore, the improvements seen with VSLAM SuperPoint, compared to the standard SuperPoint model, suggest that the proposed training process and architectural modifications have further strengthened the system's robustness.

Detector	$\bar{r}$	σ_r	$\overline{L}$	σ_L
BRISK	57.18	9.71	0.41	0.49
ORB	54.94	9.84	0.46	0.51
SIFT	61.71	9.07	0.53	0.50
SuperPoint	74.33	10.0	0.68	0.47
VSLAM SuperPoint	**74.39**	10.5	**0.74**	0.44

TABLE 5.2: Comparison of various detectors and VSLAM SuperPoint.

5.2 VSLAM RANSAC

VSLAM SuperPoint has shown that incorporating historical data alongside the current frame significantly enhances the performance of the VSLAM system. This observation underscores the potential of utilizing historical insights to improve the accuracy of pose estimation. RANSAC, a widely recognized algorithm for handling outliers in tasks such as pose estimation, has evolved through numerous variants over the years, each designed to refine its efficacy [Mar+22]. In this section, a new variant of the RANSAC algorithm is proposed, specifically tailored for VSLAM systems. This novel approach leverages additional metrics collected during the operation to further enhance the pose estimation process.

At the beginning, let recall the RANSAC algorithm. It is a robust method for estimation of mathematical model's parameters based on a data which main contain outliers. An outlier is a data point that significantly deviates from the other observations in a dataset. In the context of VSLAM, outliers may be caused by incorrect measurements or errors in previous tasks like feature matching. RANSAC was introduced by Fischler and Bolles in [FB81]. It operates by repeatedly selecting a random subset of the original data. These subsets are used to estimate the model parameters, and then a consensus set is determined by identifying the data points that fit well with the model estimated from the random subset. The process iterates, each time potentially increasing the size of the consensus set. The best model is considered to be the one which corresponds to the largest consensus set. The idea of RANSAC is presented in Listing 5.3. At the beginning, initialization of variables occurs (lines 3-5). Then, the algorithm iterates `maxIterations` times (Line 7). In each iteration, random points are selected (Line 8) and an attempt is made to fit the model using these points (Line 9). Subsequently, the algorithm checks (Lines 12-14) which points fit the model. If the number of points that fit the model exceeds the threshold `minInliers` (Line 16), the model fitting procedure is performed again (Line 17). Finally, the overall error is calculated (Line 18), and if it is the lowest so far, the best model is updated and preserved (Lines 23-24).

During the mapping process, various landmarks are created and added to the map. However, these landmarks are expected to vary in reliability. For the purposes of this discussion, *reliability* is intuitively understood as the characteristic of landmarks that consistently perform well. A precise definition of reliability, along with the formula for calculating it, will be provided later in this section. Given this variation in reliability, landmarks with higher reliability are anticipated to have a more positive impact on the pose estimation procedure than those with lower reliability. Based on this simple yet powerful idea, the VSLAM RANSAC is proposed. This concept is inspired by the PROSAC methodology originally proposed by Chum and Matas.

The PROSAC algorithm is an adaptation of the traditional RANSAC, designed in such way to enhance the efficiency of model fitting in scenarios with significant inlier noise. By incorporating a prioritization scheme that leverages the ranking, PROSAC systematically selects the most promising data points for hypothesis testing [CM05]. Initially, PROSAC was proposed for the feature matching task. However, the scheme may be extended to other various tasks such as robust model fitting, object detection, motion estimation, and any application requiring outlier rejection or robust parameter

```
algorithm RANSAC(data, model, numSamples, maxIterations,
   threshold, minInliers)

  bestModel = null
  bestConsensusSet = null
  lowestError = ∞

  for iteration in range(1, maxIterations)
    candidateInliers = SelectRandomPoints(numSamples, data)
    candidateModel = FitModel(model, candidateInliers)
    consensusSet = candidateInliers

    for point in data:
      if IsInlier(point, candidateModel, threshold)
        AddPoint(consensusSet, point)

    if Size(consensusSet) > minInliers
      refinedModel = FitModel(consensusSet)
      currentError = CalculateError(refinedModel,
                                    consensusSet)

      if currentError < lowestError
        bestModel = refinedModel
        bestConsensusSet = consensusSet
        lowestError = currentError
  end for

  return bestModel, bestConsensusSet
end algorithm
```

LISTING 5.3: RANSAC pseudocode

estimation in the presence of noise and outliers. This progressive approach not only accelerates the convergence towards optimal model parameters by focusing on likely inliers first but also reduces computational overhead compared to conventional RANSAC. As a result, PROSAC offers a robust solution for applications requiring high precision in the presence of extensive outlier data, making it particularly valuable in fields like computer vision.

The prioritization scheme necessitates the introduction of a measure by which data points can be sorted. Consequently, this is a good point to introduce and define the reliability metric. The reliability metric s_l for a given landmark l can be defined as the ratio of the number of observations N_o of a landmark l to the total expected number of observations N_e that should take place. It is given by Equation 5.26.

$$s_l = \frac{N_o}{N_e} \tag{5.26}$$

It has previously been noted that the reliability metric can vary among different landmarks within a map. To substantiate this observation, an experiment was conducted using the StellaVSLAM framework and the KITTI dataset. For each sequence in the dataset, a map was constructed, and the observations of landmarks were analyzed. The results are depicted in Figure 5.10. Specifically, the reliability distribution of the landmarks is illustrated in Figure 5.10a, and a second plot in Figure 5.10b provides insight into the number of observations per landmark for each sequence. The analysis confirms the earlier discussion regarding the variability in the reliability of landmarks. The first part of the figure shows a varied distribution of landmark reliability across multiple sequences, indicating that some landmarks consistently offer higher reliability than others. This variability can significantly impact the performance and accuracy of the VSLAM system. Furthermore, the second part of the figure reveals that the number of observations per landmark varies significantly, ranging from 2 to 100. This variation underscores the need for techniques in handling landmarks with differing number of observations to optimize the mapping and navigation processes.

VSLAM RANSAC takes the idea of PROSAC algorithm, where the quality metric is calculated as the sum of two key components: the feature matching quality metric $s_K(\mathbf{d}_1, \mathbf{d}_2)$ between two descriptors $\mathbf{d}_1, \mathbf{d}_2$ and the landmark's l reliability $s_L(l)$. This combined quality metric, as shown in Equation 5.27, ensures that both the precision of feature matches and the reliability of landmarks are taken into account during the selection process.

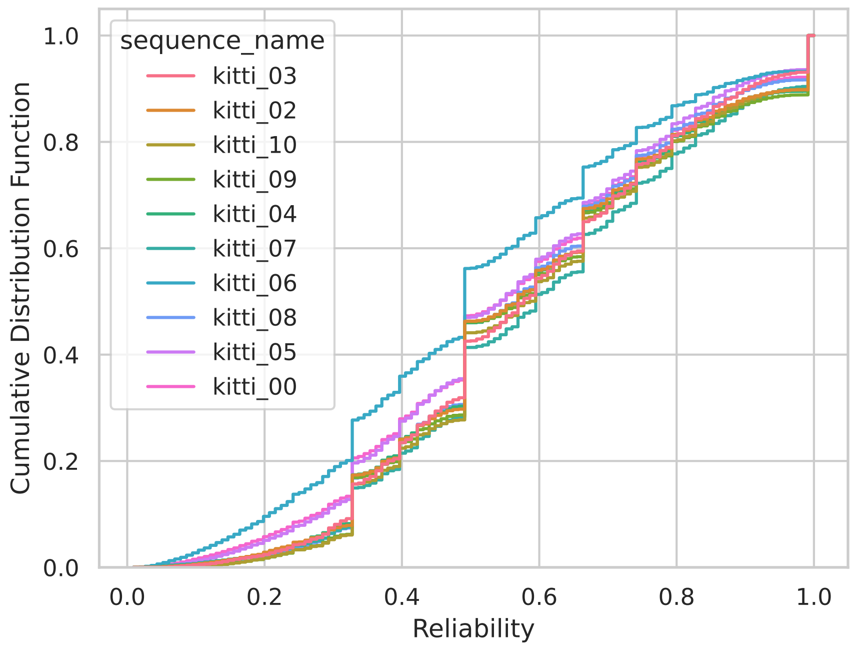
$$s(\mathbf{d}_1, \mathbf{d}_2, l) = s_L(l) + s_K(\mathbf{d}_1, \mathbf{d}_2) \tag{5.27}$$

The explicit form of $s_K(\mathbf{d}_1, \mathbf{d}_2)$ is not provided here, as it may vary depending on the type of descriptors used. Different descriptors may require different definitions of this metric. Next, the informed guess is performed. Based on the quality metrics the probabilities of each observation is calculated. With the provided probabilities a random subset is chosen. The rest of the algorithm is the same as RANSAC.

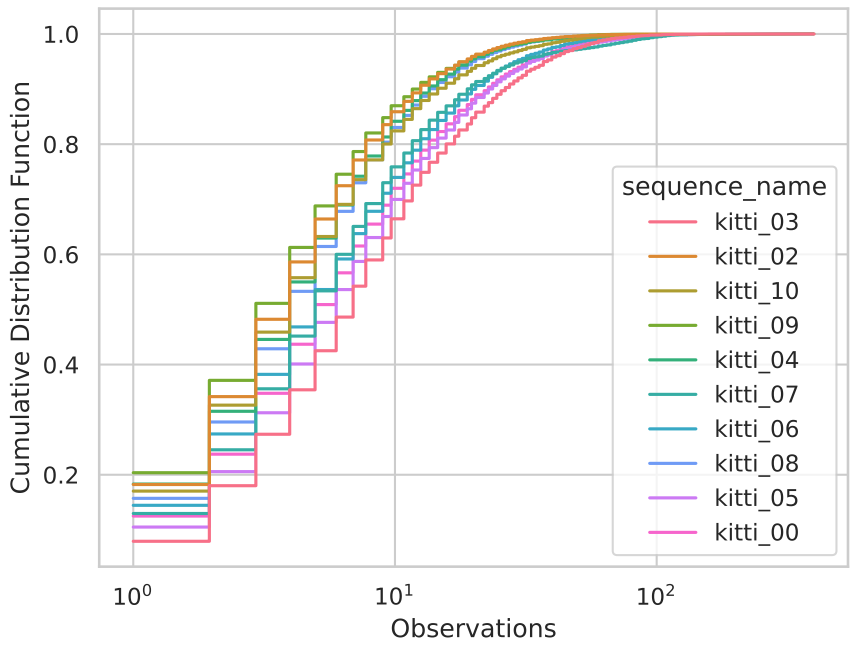
This approach enhances the robustness of the RANSAC method by prioritizing feature matches and reliable landmarks during the model estimation process, giving them a higher likelihood of being selected. This ensures that the model is built on more accurate and trustworthy data. The detailed implementation of this proposed algorithm can be found in Listing 5.4.

5.2.1 Evaluation

The evaluation focused primarily on two critical metrics: the ratio of inliers and the localization accuracy of each method compared to classic RANSAC.



(A) Distribution of the number of landmark's observations in StellaVSLAM.



(B)

FIGURE 5.10: Results of landmarks' observations analysis using StellaVSLAM and KITTI dataset.

```
algorithm VSLAM_RANSAC(qualityMetric, numInitialPoints,
   maxIterations, threshold, data)

  bestModel = null
  bestConsensusSet = null
  lowestError = ∞

  for iteration in range(1, maxIterations)
    # following line differs from RANSAC
    candidateInliers = SelectRandomPoints(numInitialPoints,
                                          data,
                                          qualityMetric)
    candidateModel = FitModel(candidateInliers, modelType)
    consensusSet = candidateInliers

    for point in data:
        if IsInlier(point, candidateModel, threshold)
          AddPoint(consensusSet, point)

      if Size(consensusSet) > minInliers
        refinedModel = FitModel(consensusSet)
        currentError = CalculateError(refinedModel,
                                      consensusSet)

        if currentError < lowestError
          bestModel = refinedModel
          bestConsensusSet = consensusSet
          lowestError = currentError
    end for

  return bestModel, bestConsensusSet

end algorithm
```

LISTING 5.4: VSLAM RANSAC pseudocode

The analysis of inliers ratio provided insights into the outliers rejection. Meanwhile, the second aspect concentrated on assessing and comparing the localization accuracy, which measures how effectively each method estimates the camera's position and orientation over time. These comparative analyses are essential for determining the effectiveness and reliability of VSLAM RANSAC, particularly in terms of maintaining consistent performance under challenging environmental conditions.

Before the results are presented, let define the inliers ratio r_i. Inliers ratio is a number of inliers points N_{inliers} divided by the number of all points used for estimation N_{all}. It is given by Equation 5.28.

$$r_{\text{inliers}} = \frac{N_{\text{inliers}}}{N_{\text{all}}} \tag{5.28}$$

The evaluations were carried out using sequences from the TUM dataset, which is well-regarded for testing the performance of visual SLAM systems. Given the real-time performance requirements of VSLAM, both the traditional RANSAC and the modified VSLAM RANSAC methods were assessed under identical experimental conditions. Specifically, both algorithms were configured to operate with the same parameters: a maximum of 10 iterations and a reprojection error threshold set at 3 pixels. The detailed results from these evaluations are summarized in Table 5.3.

Sequence	*RANSAC*		*VSLAM RANSAC*	
	$\bar{r}_{\text{inliers}}$	$\sigma_{r_{\text{inliers}}}$	$\bar{r}_{\text{inliers}}$	$\sigma_{r_{\text{inliers}}}$
freiburg1_desk	65.857	12.354	**67.781**	12.564
freiburg1_desk2	65.278	13.915	**66.719**	12.254
freiburg1_rpy	72.616	8.995	**76.424**	9.453
freiburg1_xyz	77.267	7.867	**78.064**	7.591
freiburg2_desk	73.906	9.022	**75.086**	8.031
freiburg2_rpy	**84.918**	4.453	82.971	7.080
freiburg2_xyz	80.635	4.602	**80.924**	4.281

TABLE 5.3: Evaluation results of RANSAC and VSLAM RANSAC – inliers ratio.

The experiments demonstrated that pose estimation within VSLAM systems is highly affected by outliers. Notably, in nearly all tested sequences, both the RANSAC and VSLAM RANSAC methods reached their maximum iteration limit without terminating early, despite the confidence ratio being set relatively low at 0.8. This persistent reaching of iteration limits underscores the challenging nature of accurately estimating pose in environments with significant outlier data. However, the modifications incorporated into VSLAM RANSAC yielded a higher ratio of inliers across most sequences compared to traditional RANSAC. Additionally, there was a noticeable reduction in the standard deviation of inlier ratios when using VSLAM RANSAC, indicating a more consistent performance across different test scenarios. This suggests that the enhancements made to the RANSAC algorithm

Sequence	***RANSAC***		***VSLAM RANSAC***	
	$\overline{APE}$	σ_{APE}	$\overline{\text{APE}}$	σ_{APE}
freiburg1_desk	0.027	0.018	**0.024**	0.015
freiburg1_desk2	0.091	0.057	**0.037**	0.021
freiburg1_rpy	0.0230	0.012	**0.0225**	0.016
freiburg1_xyz	0.0161	0.010	**0.0159**	0.011
freiburg2_desk	0.031	0.020	**0.030**	0.023
freiburg2_rpy	**0.015**	0.008	0.026	0.015
freiburg2_xyz	0.016	0.010	**0.011**	0.007

TABLE 5.4: Evaluation results of RANSAC and VSLAM RANSAC – absolute position error.

in the context of VSLAM improve its robustness and reliability in handling outliers.

The next phase of experimentation involved measuring the APE, as detailed in Section 3.10. The results, summarized in Table 5.4, reveal a positive impact on pose estimation accuracy in most of the sequences when using VSLAM RANSAC compared to the classic RANSAC approach. This demonstrates enhanced performance in terms of accuracy, although the improvements are not as pronounced as those observed in the inlier ratio. It is suspected that other aspects of the VSLAM system, such as local map tracking and loop closure mechanisms, may significantly influence overall performance.

It is important to highlight that the *freiburg2_rpy* sequence, marked in red, was singled out for a specific reason. At first, this sequence shows that the RANSAC algorithm performs better in terms of the number of inliers and the mean APE. However, a deeper analysis reveals that this is due to the fact that the number of frames tracked by RANSAC is significantly lower compared to VSLAM RANSAC. This reduced number of tracked frames results in artificially higher inlier counts and lower APE. This observation underscores the importance of carefully interpreting VSLAM evaluation results, as multiple factors must be considered. Furthermore, this finding suggests potential future research in developing new metrics to more accurately assess tracking performance. Finally, it is also evident that VSLAM RANSAC still preserves its robustness, consistently maintaining accurate tracking and producing reliable estimates even in situations where traditional RANSAC fails.

One of the primary goals of VSLAM RANSAC was to reduce the maximum number of iterations compared to the classic RANSAC. If the number of iterations can be reduced while maintaining the same or even better accuracy, VSLAM RANSAC not only enhances algorithmic robustness but also improves computational efficiency covering two aspects of robustness. In fact, tests were conducted where the classic RANSAC was executed with various maximum iteration limits, specifically 15, 20, 25, and 30 iterations. The results demonstrated that classic RANSAC required roughly twice the number of iterations to achieve similar accuracy to VSLAM RANSAC, highlighting the efficiency and robustness of the proposed approach.

This chapter concludes the discussion on new robust methods, a core segment of this thesis. Various techniques aimed at enhancing the reliability and accuracy of VSLAM systems have been explored, focusing on how these methodologies can mitigate common challenges associated with SLAM technology. The introduction of VSLAM SuperPoint and VSLAM RANSAC has been showcased, exemplifying significant strides made in this area and demonstrating the potential and complexity of integrating robust methods into existing frameworks. As this chapter is brought to a close, the foundation is established for the final chapter of the thesis, which will provide a comprehensive summary of the entire study.

Chapter 6

Conclusions

Overview

This chapter provides a comprehensive summary of the research conducted throughout the thesis, emphasizing the significant advancements made in improving the robustness of VSLAM systems. It revisits the primary objectives and proposed methods, discussing how each has been addressed by the findings and developments presented in the preceding chapters. It also identifies potential directions for future research, inspired by the unresolved issues and emerging trends in the field. It proposes several specific areas where further investigations could yield significant improvements in VSLAM robustness, especially using Modular SLAM. Finally, the chapter concludes with reflections on the broader implications of this research for the fields such as autonomous systems or augmented reality, suggesting how the enhanced VSLAM capabilities could be integrated into various practical applications.

"Somewhere, something incredible is waiting to be known."
– Carl Sagan

Throughout this thesis, the realms of Modular SLAM have been extensively explored, and new, robust methods that serve as essential components of the VSLAM pipeline have been introduced. As this work is brought to an end, it is relevant to reflect on the significant achievements that has been covered. At the beginning, a comprehensive introduction to the challenges and developments in SLAM and VSLAM was provided, surveying almost four decades of research in this dynamic field. This foundation set the stage for the subsequent examination and application of robustness-enhancing strategies within VSLAM systems. The methodologies proposed and examined aim to mitigate the vulnerabilities of traditional approaches, emphasizing the importance of adaptability and resilience in real-world applications. Furthermore, the challenges was described and initial analysis of the problems has been conducted

Furthermore, the concept of Modular SLAM was proposed, marking a significant contribution to this PhD thesis as a fundamental component of the research. Specifically developed for this study, the Modular SLAM employs a modular architecture that enhances SLAM capabilities by offering flexibility and extensibility. This versatile framework supports rapid prototyping and comprehensive evaluation of various SLAM algorithms, thus facilitating the exploration and implementation of diverse strategies across frontend, backend, and mapping. By enhancing the adaptability and functionality of SLAM systems, the Modular SLAM framework fosters avenues for experimentation, innovation, and advancement in the field, establishing itself as an invaluable resource for researchers and practitioners. Notably, it has been shown that software architecture and design patterns may be also beneficial for the researchers. Moreover, Modular SLAM integrates disciplines such as computer vision, 3D reconstruction, and software design, adopting best practices to achieve robustness.

In terms of robustness, the scope of this concept was significantly expanded in this work. Robustness is not only understood as algorithmic resilience but as a broader term encompassing multiple aspects. This thesis proposes a model of VSLAM robustness consisting of three components: algorithmic robustness, software robustness, and processing time efficiency, each critical to the overall performance and reliability of VSLAM systems. In this thesis, each aspect of robustness is addressed: Modular SLAM enhances software robustness, VSLAM SuperPoint and VSLAM RANSAC contribute to algorithmic robustness, and VSLAM RANSAC also improves performance robustness, typically delivering better results in less number of iterations.

VSLAM SuperPoint represents a significant advancement in keypoint detection by utilizing deep learning to achieve robust detection and accurate matching. Unlike traditional VSLAM systems, which depend on single images to detect keypoints, VSLAM SuperPoint capitalizes on the sequential processing of images inherent in VSLAM. This innovative method integrates additional data from previously detected keypoints, thereby enhancing the detector's ability to create comprehensive keypoint heatmaps. This enhancement is crucial as it allows the detector to leverage historical data effectively. The findings from this study indicate that the repeatability rates of traditional detectors such as ORB or SIFT are notably inferior to those achieved with deep learning-based methods, demonstrating that VSLAM systems can significantly benefit from incorporating these advanced approaches.

Furthermore, this thesis introduces VSLAM RANSAC, a refined variant of the classic RANSAC algorithm that incorporates an informed selection process for subset points based on the reliability of landmarks. This novel approach has been proven to enhance the performance of VSLAM systems in pose estimation tasks, surpassing the results of traditional RANSAC implementations. By using a reliability measure for landmarks, VSLAM RANSAC optimizes the selection of data points for model estimation, resulting in more accurate and robust pose estimation.

Both VSLAM SuperPoint and VSLAM RANSAC exemplify how historical data can be effectively used to increase the accuracy and robustness of VSLAM systems. The research presented in this thesis not only supports but also solidifies the proposition that integrating sophisticated data handling and analysis techniques can lead to significant improvements in the performance of VSLAM technologies. This thesis, therefore, confirms the critical role of advanced data utilization strategies in enhancing the capabilities of VSLAM systems.

Clearly, all these contributions represent a considerable step in the ongoing research within the VSLAM domain. While significant advancements have been made, numerous challenges and open problems in VSLAM remain. These unresolved issues highlight the dynamic and evolving nature of the field, underscoring the need for continued exploration and innovation. Consequently, this leads to the next section of the conclusions, where proposals for further research areas are outlined. These proposed directions aim to address the current limitations and expand upon the work presented in this thesis, offering a pathway for future advancements in the robustness and efficiency of VSLAM systems.

6.1 Future Work

This thesis has introduced several methodologies aimed at enhancing the robustness VSLAM systems. However, achieving complete robustness remains an ongoing and crucial challenge. Despite considerable advancements in the field, the complex and dynamic nature of real-world environments highlights the need for further research. Future studies should prioritize practical applications and real-world testing, as unique challenges often emerge only under such conditions. At the outset of this thesis, a fundamental question was posed: "Is SLAM solved?" The answer is still no. Numerous issues persist that demand comprehensive solutions and innovative approaches.

The Modular SLAM framework developed in this work represents a concept that is anticipated to undergo further refinement and enhancement. This framework offers numerous opportunities for augmentation through the integration of new algorithms and functionalities. For instance, incorporating other pose estimation algorithms could improve the accuracy and robustness of VSLAM systems. While this thesis has primarily focused on feature-based systems, it is evident that direct methods, which typically generate more detailed maps, are gaining popularity and will likely continue to do so. Additionally, extending the framework to include support for Python, language that has gained substantial popularity among researchers, would significantly enhance its accessibility and utility. Moreover, as time execution is crucial in VSLAM systems Modular SLAM research should also focus on the code optimization to make it possible to run on the smartphones i.e. to serve as a core technology for AR applications.

Furthermore, the exploration of collaborative and distributed VSLAM systems was not covered in this thesis. In today's era of interconnected devices, there exists potential to improve global mapping significantly. For example, autonomous vehicles could share real-time data to refine and enhance the accuracy of collective mapping efforts.

Another unresolved challenge is dynamic objects handling within VSLAM systems. Current methodologies largely focus on excluding dynamic objects from maps. However, recognizing and incorporating dynamic objects could be beneficial in scenarios such as autonomous navigation where interaction with moving objects is inevitable.

One of the most important paths for ongoing and future research is the application of AI algorithms, which can significantly enhance the capabilities of VSLAM systems. Integrating AI with traditional geometric-based approaches offers a robust synthesis that leverages the strengths of both methodologies. AI algorithms, particularly those based on machine learning and deep learning, can process complex visual data at high speeds and improve the accuracy of feature detection, data association, and environmental mapping. These algorithms are particularly adept at handling the high variability and unpredictability found in real-world environments, where traditional methods may fail.

The summary of potential research areas in VSLAM, as categorized in the provided mindmap, demonstrates that VSLAM remains a rich field for scientific inquiry. There are numerous areas that require further attention from researchers, as illustrated in Figure 6.1. This mindmap of future studies underscores the vast scope for advancements and the ongoing relevance of VSLAM research in technological progress.

Finally, it is important to note that VSLAM can be utilized in a variety

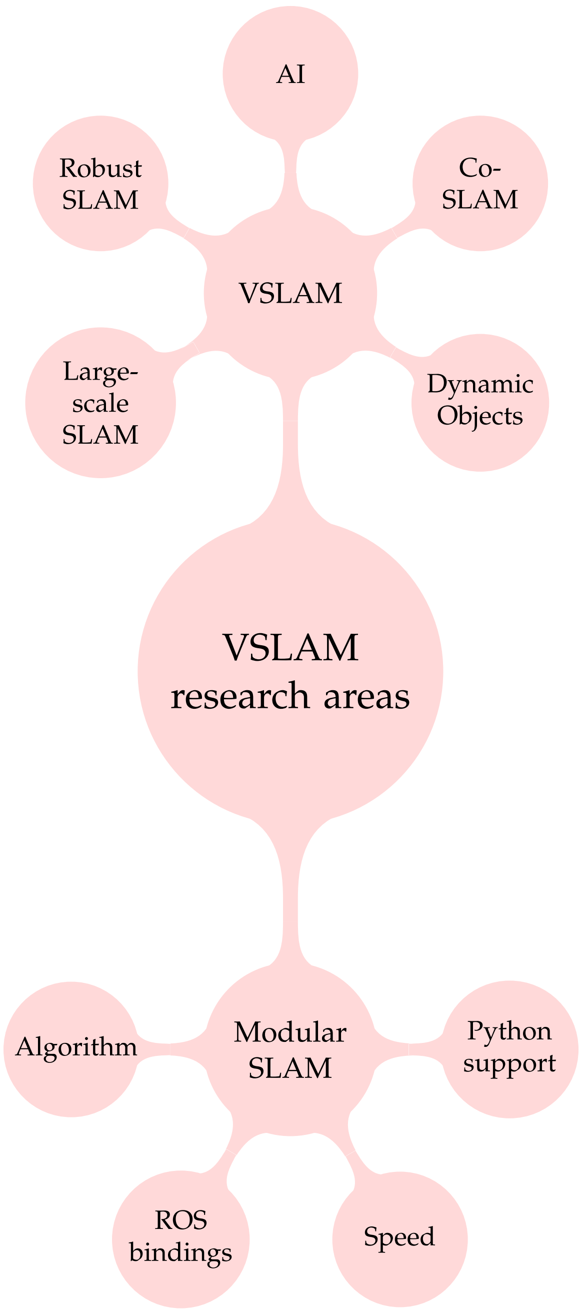


FIGURE 6.1: Potential directions for future research and development.

of applications. One of the most compelling uses is in AR, where VSLAM's precise tracking capabilities allow for the addition and tracking of markers on the map, which can then be displayed as additional content for users. Another significant application is autonomous navigation, where VSLAM algorithms help localize and build an understanding of the environment.

These two use-cases open up possibilities for employing VSLAM across various disciplines. For instance, in construction management, VSLAM can be used for precise site reconstructions. It may also play a crucial role in security and surveillance, logistics and warehousing, agriculture, and many other fields. This broad applicability underscores the versatility and importance of VSLAM technology.

Hopefully, Modular SLAM will continue to evolve, becoming an essential tool not only for researchers but also for practitioners who contribute to the advancement of VSLAM technology.

Appendix A

SLAM projects

Over the years, researchers have developed numerous solutions to address the challenges posed by SLAM. Visual SLAM, in particular, has seen significant advancements, resulting in a variety of systems that have been instrumental in advancing the field. One of the contributions of this thesis is to provide a comprehensive summary of the most valuable and influential VSLAM systems.

The tables in this appendix (see Table A.1–Table A.4) present an overview of these systems, highlighting their creation date, current status, sensor compatibility, type of SLAM approach, and additional notes that provide context to their significance. This summary serves as a resource for understanding the evolution and current state of VSLAM technologies.

Name	*Created*	*Status*	*Sensors*	*Type*	*Notes*
DTAM [New+11]	2011	Inactive	Monoc-ular	Direct	Dense Tracking and Mapping, focused on real-time dense 3D reconstruction.
Kinect-Fusion [NLD11]	2011	Inactive	RGB-D	Dense	Real-time dense mapping using Kinect sensor, highly influential in RGB-D SLAM.
DVO-SLAM [KSC13]	2013	Inactive	RGB-D	Direct	Dense visual odometry, uses direct image alignment.
RGBDSLAMv2 [End+14]	2014	Inactive	RGB-D	Feature-based	Extended version of the original RGBDSLAM with improved performance.

TABLE A.1: The most significant VSLAM systems I

Name	***Created***	***Status***	***Sensors***	***Type***	***Notes***
LSD-SLAM [ESC14]	2014	Inactive	Monocular	Direct	Large-Scale Direct Monocular SLAM, semi-dense mapping.
SVO [FPS14]	2014	Inactive	Monocular	Direct	Semi-Direct Visual Odometry, balances speed and accuracy, often used in drones.
ORB-SLAM [MMT15]	2015	Inactive	Monocular	Feature-based	Widely used, accurate, real-time performance.
ElasticFusion [Whe+16]	2016	Inactive	RGB-D	Dense	Real-time dense visual SLAM, emphasis on surface reconstruction.

TABLE A.2: The most significant VSLAM systems II

Name	*Created*	*Status*	*Sensors*	*Type*	*Notes*
ORB-SLAM2 [MT17]	2017	Active	Monocular, Stereo, RGB-D	Feature-based	Improved version of ORB-SLAM, supports multiple sensor configurations.
ProSLAM [SCG18]	2018	Inactive	Monocular, Stereo	Feature-based	Lightweight SLAM system optimized for efficiency, simple and effective.
Kimera [Ros+19; Ros+20b; Ros+20a; Ros+21b]	2019	Active	Monocular, Stereo, RGB-D	Feature-based	Combines SLAM with 3D mesh generation, semantic labeling, and planning.
RTabMap [LM18; LM17; LM14; LM13; LM11]	2011	Active	RGB-D, Stereo	Feature-based	Real-Time Appearance-Based Mapping, loop closure detection.

TABLE A.3: The most significant VSLAM systems III

Name	*Created*	*Status*	*Sensors*	*Type*	*Notes*
StellaVSLAM [SSS19]	2019	Active	Monocular, Stereo, RGB-D	Feature-based	Fork of OpenVSLAM with additional features and improvements.
ORB-SLAM3 [Cam+21]	2021	Active	Monocular, Stereo, RGB-D	Feature-based	Introduces support for multi-map and inertial data integration.
DROID-SLAM [TD21]	2021	Inactive	Monocular, Stereo	Direct	Deep-learning-based SLAM, known for its robustness in diverse environments.
Nerf-SLAM [RLC22]	2022	Active	Monocular	Neural	Neural Radiance Fields for SLAM, combines NeRF with SLAM for photorealistic scene reconstruction.

TABLE A.4: The most significant VSLAM systems IV

Appendix B

Additional resources

This appendix serves as a comprehensive repository of resources accompanying the main body of the thesis. Among these resources, readers will find links to GitHub repositories hosting the relevant source code, Jupyter notebooks [Jup24] that detail the experiments conducted with thorough descriptions, and Docker images [Mer14] that encapsulate the computational environments used throughout the research. The inclusion of these materials underscores the commitment to transparency and the advancement of knowledge within the field, allowing researchers to build upon the foundational work presented in this thesis. Table B.1 compiles a comprehensive list of resources utilized throughout this thesis, providing URLs for their locations as well as descriptions of each asset.

Resource	*Path*	*Description*
Modular SLAM	https://github.com/marcin-ochman/modular-slam	Source code of modular-slam library containing C++ implementation, Dockerfiles of other VSLAM systems and evaluation scripts
Modular SLAM Docs	https://github.com/marcin-ochman/modular-slam	Documentation of modular-slam library including classes and functions overview, architecture and user manual.
Docker Images	https://github.com/marcin-ochman/modular-slam	All Docker images of the systems evaluated in this thesis including modular-slam, Stella VSLAM, ORB-SLAM3 and LSD-SLAM.
Jupyter Notebook	https://github.com/marcin-ochman/phd-resources.git	All experiments were conducted using Jupyter Notebook to make them reproducible. They contain all commands used in the research along with short description.

TABLE B.1: List of resources developed for this thesis.

A significant contribution of this thesis and Modular SLAM is the development of Docker images for several popular VSLAM systems. Given that most of these systems do not offer Dockerfiles, creating these images presented numerous challenges. The decision to utilize Docker was motivated by the need to guarantee the reliable operation of these algorithms and to facilitate their distribution among researchers wishing to benchmark them against their own developments. Table B.2 presents a list of these Docker images, each accompanied by a brief description and a location in Modular SLAM repository.

Docker Image	***Path***	***Description***
modular-slam/lsd-slam	utils/tools/py/3rdparty_slam/lsd_slam	LSD-SLAM [ESC14]
modular-slam/stella-vslam	utils/tools/py/3rdparty/slam/stella_vslam	StellaVSLAM [SSS19]
modular-slam/orb-slam-3	utils/tools/py/3rdparty_slam/orb_slam_3	ORB-SLAM3[Cam+21]

TABLE B.2: List of Docker images.

Bibliography

[ABD12] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J. Davison. “KAZE Features”. In: *Computer Vision – ECCV 2012*. Springer Berlin Heidelberg, 2012, pp. 214–227. [Link].

[AK21] Saba Arshad and Gon-Woo Kim. “Role of Deep Learning in Loop Closure Detection for Visual and Lidar SLAM: A Survey”. In: *Sensors* 21.4 (Feb. 2021), p. 1243. [Link].

[AKB08] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. “CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching”. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2008, pp. 102–115. [Link].

[AMT22] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. *Ceres Solver*. Version 2.1. Mar. 2022.

[ANB13] Pablo Alcantarilla, Jesus Nuevo, and Adrien Bartoli. “Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces”. In: *Procedings of the British Machine Vision Conference 2013*. British Machine Vision Association, 2013. [Link].

[Apa+22] Joaquin Aparicio et al. “A Survey on Acoustic Positioning Systems for Location-Based Services”. In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022), pp. 1–36. [Link].

[Aqe+16] Mohammad O. A. Aqel et al. “Review of visual odometry: types, approaches, challenges, and applications”. In: *SpringerPlus* 5.1 (Oct. 2016). ISSN: 2193-1801. [Link].

[Aru+02] M.S. Arulampalam et al. “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking”. In: *IEEE Transactions on Signal Processing* 50.2 (2002), pp. 174–188. [Link].

[ASG20] Bjorn Andrist, Viktor Sehr, and Ben Garney. *C++ High Performance*. en. 2nd ed. Birmingham, England: Packt Publishing, Dec. 2020.

[Aua+10] Fernando A Auat Cheein et al. “SLAM algorithm applied to robotics assistance for navigation in unknown environments”. In: *Journal of NeuroEngineering and Rehabilitation* 7.1 (Feb. 2010). ISSN: 1743-0003. [Link].

[Aul+08] Josep Aulinas et al. "The SLAM problem: a survey". In: vol. 184. Jan. 2008, pp. 363–371. [Link].

[Azz+20] Rana Azzam et al. "Feature-based visual simultaneous localization and mapping: a survey". In: *SN Applied Sciences* 2.2 (Jan. 2020). ISSN: 2523-3971. [Link].

[Bai02] Tim Bailey. "Mobile Robot Localisation and Mapping in Extensive Outdoor Environments". In: 2002.

[Bal+21] Irene Ballester et al. "DOT: Dynamic Object Tracking for Visual SLAM". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2021. [Link].

[Bar+22] Andréa Macario Barros et al. "A Comprehensive Survey of Visual SLAM Algorithms". In: *Robotics* 11.1 (Feb. 2022), p. 24. [Link].

[Bar21] Lorena A. Barba. "The Python/Jupyter Ecosystem: Today's Problem-Solving Environment for Computational Science". In: *Computing in Science & Engineering* 23.3 (May 2021), pp. 5–9. ISSN: 1558-366X. [Link].

[BC21] Hudson Martins Silva Bruno and Esther Luna Colombini. "LIFT-SLAM: A deep-learning feature-based monocular visual SLAM method". In: *Neurocomputing* 455 (Sept. 2021), pp. 97–110. [Link].

[BCL15] Mark Billinghurst, Adrian Clark, and Gun Lee. "A Survey of Augmented Reality". In: *Foundations and Trends® in Human–Computer Interaction* 8.2-3 (2015), pp. 73–272. [Link].

[BD06] T. Bailey and H. Durrant-Whyte. "Simultaneous localization and mapping (SLAM): part II". In: *IEEE Robotics & Automation Magazine* 13.3 (Sept. 2006), pp. 108–117. ISSN: 1070-9932. [Link].

[Bes+18] Berta Bescos et al. "DynaSLAM: Tracking, Mapping, and Inpainting in Dynamic Scenes". In: *IEEE Robotics and Automation Letters* 3.4 (Oct. 2018), pp. 4076–4083. ISSN: 2377-3774. [Link].

[BKC12] Len Bass, Rick Kazman, and Paul Clements. *Software Architecture in Practice*. en. 3rd ed. SEI series in software engineering. Boston, MA: Addison-Wesley Educational, Sept. 2012.

[Bla19] Jose Luis Blanco-Claraco. "A Modular Optimization Framework for Localization and Mapping". In: *Proceedings of Robotics: Science and Systems*. FreiburgimBreisgau, Germany, June 2019. [Link].

[Bos+03] M. Bosse et al. "An Atlas framework for scalable mapping". In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 2. 2003, 1899–1906 vol.2. [Link].

[Bou21] Youssef Bouaziz. “Visual SLAM with automatic map update in dynamic environments”. PhD thesis. July 2021.

[BSA13] Jaime Boal, Álvaro Sánchez-Miralles, and Álvaro Arranz. “Topological simultaneous localization and mapping: a survey”. In: *Robotica* 32.5 (Dec. 2013), pp. 803–821. [Link].

[BTG06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. “SURF: Speeded Up Robust Features”. In: *Computer Vision – ECCV 2006*. Springer Berlin Heidelberg, 2006, pp. 404–417. [Link].

[Bur+16] Michael Burri et al. “The EuRoC micro aerial vehicle datasets”. In: *The International Journal of Robotics Research* 35.10 (Jan. 2016), pp. 1157–1163. [Link].

[Bus+96] Frank Buschmann et al. *Pattern-oriented software architecture*. en. Wiley Software Patterns Series. Nashville, TN: John Wiley & Sons, July 1996.

[Cad+16] Cesar Cadena et al. “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age”. In: *IEEE Transactions on Robotics* 32.6 (Dec. 2016), pp. 1309–1332. [Link].

[Cal+10] Michael Calonder et al. “BRIEF: Binary Robust Independent Elementary Features”. In: *Computer Vision – ECCV 2010*. Springer Berlin Heidelberg, 2010, pp. 778–792. [Link].

[Cam+21] Carlos Campos et al. “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual–Inertial, and Multimap SLAM”. In: *IEEE Transactions on Robotics* 37.6 (Dec. 2021), pp. 1874–1890. [Link].

[Car+15] Luca Carlone et al. “Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization”. In: *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2015. [Link].

[Cen+20] Linga Reddy Cenkeramaddi et al. “A Survey on Sensors for Autonomous Systems”. In: *2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. IEEE, Nov. 2020. [Link].

[Cha20] Hyunggi Chang. *Visual SLAM Roadmap*. . 2020.

[Che+18] Chang Chen et al. “A Review of Visual-Inertial Simultaneous Localization and Mapping from Filtering-Based and Optimization-Based Perspectives”. In: *Robotics* 7.3 (Aug. 2018), p. 45. [Link].

[Che+22a] Weifeng Chen et al. “An Overview on Visual SLAM: From Tradition to Semantic”. In: *Remote Sensing* 14.13 (June 2022), p. 3010. ISSN: 2072-4292. [Link].

[Che+22b] Weifeng Chen et al. “SLAM Overview: From Single Sensor to Heterogeneous Fusion”. In: *Remote Sensing* 14.23 (Nov. 2022), p. 6033. [Link].

[Cho+15] T.J. Chong et al. “Sensor Technologies and Simultaneous Localization and Mapping (SLAM)”. In: *Procedia Computer Science* 76 (2015), pp. 174–179. [Link].

[Chu+22] Chi-Ming Chung et al. *Orbeez-SLAM: A Real-time Monocular Visual SLAM with ORB Features and NeRF-realized Mapping*. 2022. [Link].

[CM05] O. Chum and J. Matas. “Matching with PROSAC - progressive sample consensus”. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 220–226 vol. 1. [Link].

[CN01] H. Choset and K. Nagatani. “Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization”. In: *IEEE Transactions on Robotics and Automation* 17.2 (Apr. 2001), pp. 125–137. ISSN: 1042-296X. [Link].

[Col+20] Mirco Colosi et al. “Plug-and-Play SLAM: A Unified SLAM Architecture for Modularity and Ease of Use”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2020. [Link].

[Cza+20] Jan Czarnowski et al. “DeepFactors: Real-Time Probabilistic Dense Monocular SLAM”. In: *IEEE Robotics and Automation Letters* 5.2 (Apr. 2020), pp. 721–728. [Link].

[Dav+07] Andrew J. Davison et al. “MonoSLAM: Real-Time Single Camera SLAM”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (June 2007), pp. 1052–1067. [Link].

[Dav03] Davison. “Real-time simultaneous localisation and mapping with a single camera”. In: *Proceedings Ninth IEEE International Conference on Computer Vision*. IEEE, 2003. [Link].

[DB06] H. Durrant-Whyte and T. Bailey. “Simultaneous localization and mapping: part I”. In: *IEEE Robotics & Automation Magazine* 13.2 (June 2006), pp. 99–110. [Link].

[DC22] Frank Dellaert and GTSAM Contributors. *borglab/gtsam*. Version 4.2a8. May 2022. [Link].

[Del12] Frank Dellaert. “Factor Graphs and GTSAM: A Hands-on Introduction”. In: 2012.

[DFG01] Arnaud Doucet, Nando de Freitas, and Neil Gordon, eds. *Sequential Monte Carlo methods in practice*. en. 2001st ed. Information Science and Statistics. New York, NY: Springer, June 2001.

[Die06] James Diebel. “Representing Attitude : Euler Angles , Unit Quaternions , and Rotation Vectors”. In: 2006.

[Dis+01] M.W.M.G. Dissanayake et al. “A solution to the simultaneous localization and map building (SLAM) problem”. In: *IEEE Transactions on Robotics and Automation* 17.3 (June 2001), pp. 229–241. [Link].

[Dis+11] Gamini Dissanayake et al. “A review of recent developments in Simultaneous Localization and Mapping”. In: *2011 6th International Conference on Industrial and Information Systems*. IEEE, Aug. 2011. [Link].

[DLD23] Nigel Joseph Bandeira Dias, Gustavo Teodoro Laureano, and Ronaldo Martins Da Costa. “Keyframe Selection for Visual Localization and Mapping Tasks: A Systematic Literature Review”. In: *Robotics* 12.3 (June 2023), p. 88. ISSN: 2218-6581. [Link].

[DMR18] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. “SuperPoint: Self-Supervised Interest Point Detection and Description”. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, June 2018. [Link].

[Dre+21] Nathan Drenkow et al. *A Systematic Review of Robustness in Deep Learning for Computer Vision: Mind the gap?* 2021. [Link].

[DRN96] Hugh Durrant-Whyte, David Rye, and Eduardo Nebot. “Localization of Autonomous Guided Vehicles”. In: *Robotics Research*. Ed. by Georges Giralt and Gerhard Hirzinger. London: Springer London, 1996, pp. 613–625. ISBN: 978-1-4471-0765-1.

[Dua+19] Chao Duan et al. “Deep Learning for Visual SLAM in Transportation Robotics: A review”. In: *Transportation Safety and Environment* 1.3 (Dec. 2019), pp. 177–184. [Link].

[Dur88] H.F. Durrant-Whyte. “Uncertain geometry in robotics”. In: *IEEE Journal on Robotics and Automation* 4.1 (1988), pp. 23–31. [Link].

[DV20] César Debeunne and Damien Vivet. “A Review of Visual-LiDAR Fusion based Simultaneous Localization and Mapping”. In: *Sensors* 20.7 (Apr. 2020), p. 2068. [Link].

[DWW23] Yong Dai, Jiaxin Wu, and Duo Wang. “A Review of Common Techniques for Visual Simultaneous Localization and Mapping”. In: *Journal of Robotics* 2023 (Feb. 2023). Ed. by Keigo Watanabe, pp. 1–21. ISSN: 1687-9600. [Link].

[EKC18] Jakob Engel, Vladlen Koltun, and Daniel Cremers. “Direct Sparse Odometry”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40.3 (Mar. 2018), pp. 611–625. [Link].

[End+14] Felix Endres et al. “3-D Mapping With an RGB-D Camera”. In: *IEEE Transactions on Robotics* 30.1 (Feb. 2014), pp. 177–187. [Link].

[ESC14] Jakob Engel, Thomas Schöps, and Daniel Cremers. “LSD-SLAM: Large-Scale Direct Monocular SLAM”. In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 834–849. [Link].

[ETM21] Jos Elfring, Elena Torta, and René van de Molengraft. “Particle Filters: A Hands-On Tutorial”. In: *Sensors* 21.2 (Jan. 2021), p. 438. [Link].

[Fav23] Margarita N. Favorskaya. “Deep Learning for Visual SLAM: The State-of-the-Art and Future Trends”. In: *Electronics* 12.9 (Apr. 2023), p. 2006. [Link].

[FB81] Martin A. Fischler and Robert C. Bolles. “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”. In: *Communications of the ACM* 24.6 (June 1981), pp. 381–395. ISSN: 1557-7317. [Link].

[FPS14] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. “SVO: Fast semi-direct monocular visual odometry”. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2014. [Link].

[FR20] Eric Freeman and Elisabeth Robson. *Head first design patterns*. en. 2nd ed. Sebastopol, CA: O’Reilly Media, Dec. 2020.

[Fre23] Luigi Freda. *PLVS: A SLAM System with Points, Lines, Volumetric Mapping, and 3D Incremental Segmentation*. 2023. eprint: `arXiv:2309.10896`.

[FS12] Friedrich Fraundorfer and Davide Scaramuzza. “Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications”. In: *IEEE Robotics & Automation Magazine* 19.2 (June 2012), pp. 78–90. [Link].

[FT19] William Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Version 1.4. Mar. 2019. [Link].

[FWR10] Udo Frese, René Wagner, and Thomas Röfer. "A SLAM Overview from a User's Perspective". In: *KI - Künstliche Intelligenz* 24.3 (Sept. 2010), pp. 191–198. ISSN: 1610-1987. [Link].

[Gam95] Erich Gamma. *Design patterns : elements of reusable object-oriented software*. Reading, Mass: Addison-Wesley, 1995. ISBN: 978-0201633610.

[Gao+17] Xiang Gao et al. *14 Lectures on Visual SLAM: From Theory to Practice*. Publishing House of Electronics Industry, 2017.

[Gao+22] Ling Gao et al. "VECtor: A Versatile Event-Centric Benchmark for Multi-Sensor SLAM". In: *IEEE Robotics and Automation Letters* 7.3 (July 2022), pp. 8217–8224. [Link].

[Gao+24] Hang Gao et al. "USV Path Planning in a Hybrid Map Using a Genetic Algorithm with a Feedback Mechanism". In: *Journal of Marine Science and Engineering* 12.6 (June 2024), p. 939. ISSN: 2077-1312. [Link].

[GHT11] Steffen Gauglitz, Tobias Höllerer, and Matthew Turk. "Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking". In: *International Journal of Computer Vision* 94.3 (Mar. 2011), pp. 335–360. ISSN: 1573-1405. [Link].

[GK99] J.-S. Gutmann and K. Konolige. "Incremental mapping of large cyclic environments". In: *Proceedings 1999 IEEE International Symposium on Computational Intelligence in Robotics and Automation. CIRA'99 (Cat. No.99EX375)*. IEEE, 1999. [Link].

[GLU12] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for autonomous driving? The KITTI vision benchmark suite". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2012. [Link].

[GN01] J.E. Guivant and E.M. Nebot. "Optimization of the simultaneous localization and map-building algorithm for real-time implementation". In: *IEEE Transactions on Robotics and Automation* 17.3 (June 2001), pp. 242–257. ISSN: 1042-296X. [Link].

[GO15] Emilio Garcia-Fidalgo and Alberto Ortiz. "Vision-based topological mapping and localization methods: A survey". In: *Robotics and Autonomous Systems* 64 (Feb. 2015), pp. 1–20. ISSN: 0921-8890. [Link].

[Gom+19] Ruben Gomez-Ojeda et al. "PL-SLAM: A Stereo SLAM System Through the Combination of Points and Line Segments". In: *IEEE Transactions on Robotics* 35.3 (June 2019), pp. 734–746. ISSN: 1941-0468. [Link].

[Gri+10] G Grisetti et al. “A Tutorial on Graph-Based SLAM”. In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43. [Link].

[Gru17] Michael Grupp. *evo: Python package for the evaluation of odometry and SLAM*. `https://github.com/MichaelGrupp/evo`. 2017.

[GS22] Sakshi Gupta and Itu Snigdh. “Multi-sensor fusion in autonomous heavy vehicles”. In: *Autonomous and Connected Heavy Vehicle Technology*. Elsevier, 2022, pp. 375–389. [Link].

[GSB07] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. “Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters”. In: *IEEE Transactions on Robotics* 23.1 (Feb. 2007), pp. 34–46. ISSN: 1552-3098. [Link].

[GT12] Dorian Gálvez-López and J. D. Tardós. “Bags of Binary Words for Fast Place Recognition in Image Sequences”. In: *IEEE Transactions on Robotics* 28.5 (Oct. 2012), pp. 1188–1197. ISSN: 1552-3098. [Link].

[Gui+04] Jose Guivant et al. “Navigation and Mapping in Large Unstructured Environments”. In: *The International Journal of Robotics Research* 23.4-5 (Apr. 2004), pp. 449–472. [Link].

[HD16] Shoudong Huang and Gamini Dissanayake. “A critique of current developments in simultaneous localization and mapping”. In: *International Journal of Advanced Robotic Systems* 13.5 (Sept. 2016), p. 172988141666948. [Link].

[Hes+16] Wolfgang Hess et al. “Real-time loop closure in 2D LIDAR SLAM”. In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016. [Link].

[HKE22] Wen-Mei W Hwu, David B Kirk, and Izzat El Hajj. *Programming massively parallel processors*. en. 4th ed. London, England: Morgan Kaufmann, Sept. 2022.

[Hor+13] Armin Hornung et al. “OctoMap: an efficient probabilistic 3D mapping framework based on octrees”. In: *Autonomous Robots* 34.3 (Feb. 2013), pp. 189–206. ISSN: 1573-7527. [Link].

[Ikr+22] Muhammad Haris Ikram et al. “Perceptual Aliasing++: Adversarial Attack for Visual SLAM Front-End and Back-End”. In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022), pp. 4670–4677. ISSN: 2377-3774. [Link].

[Inc22] The MathWorks Inc. *Navigation Toolbox version: 9.4 (R2023b)*. Natick, Massachusetts, United States, 2022.

[Jam+18] Redhwan Jamiruddin et al. *RGB-Depth SLAM Review*. 2018. [Link].

[JH18] Arne Johanson and Wilhelm Hasselbring. “Software Engineering for Computational Science: Past, Present, Future”. In: *Computing in Science & Engineering* 20.2 (Mar. 2018), pp. 90–109. ISSN: 1558-366X. [Link].

[Ji19] Qiang Ji. *Probabilistic graphical models for computer vision*. en. San Diego, CA: Academic Press, Dec. 2019.

[Jia+17] Fei Jiang et al. “Artificial intelligence in healthcare: past, present and future”. In: *Stroke Vasc Neurol* 2.4 (June 2017), pp. 230–243. [Link].

[Jia+19] Min Jiang et al. “A Survey of Underwater Acoustic SLAM System”. In: *Intelligent Robotics and Applications*. Springer International Publishing, 2019, pp. 159–170. [Link].

[Jin+19] Li Jinyu et al. “Survey and evaluation of monocular visual-inertial SLAM algorithms for augmented reality”. In: *Virtual Reality & Intelligent Hardware* 1.4 (Aug. 2019), pp. 386–410. [Link].

[JK20] Zeeshan Javed and Gon-Woo Kim. “A Comparative Study of Recent Real Time Semantic Segmentation Algorithms for Visual Semantic SLAM”. In: *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, Feb. 2020. [Link].

[JKK22] Youngseok Jang, Changhyeon Kim, and H. Jin Kim. “A Survey on Vision-based Navigation Systems Robust to Illumination Changes”. In: *2022 International Conference on Electronics, Information, and Communication (ICEIC)*. IEEE, Feb. 2022. [Link].

[JU97] Simon J. Julier and Jeffrey K. Uhlmann. “New extension of the Kalman filter to nonlinear systems”. In: *SPIE Proceedings*. Ed. by Ivan Kadar. SPIE, July 1997. [Link].

[Jup24] JupyterLab. *JupyterLab*. 2024. URL: https://github.com/jupyterlab/jupyterlab (visited on 02/18/2024).

[JYX19] Yujiao Jia, Xinying Yan, and Yihan Xu. “A Survey of simultaneous localization and mapping for robot”. In: *2019 IEEE 4th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*. IEEE, Dec. 2019. [Link].

[Kal60] R. E. Kalman. “A New Approach to Linear Filtering and Prediction Problems”. In: *Journal of Basic Engineering* 82.1 (Mar. 1960), pp. 35–45. [Link].

[Kaz+22] Iman Abaspur Kazerouni et al. “A survey of state-of-the-art on visual SLAM”. In: *Expert Systems with Applications* 205 (Nov. 2022), p. 117734. [Link].

[KB17] Adrian Kaehler and Gary Bradski. *Learning OpenCV 3: Computer Vision in C++ with the OpenCV Library*. O'Reilly Media, 2017. ISBN: 1491937998.

[Ker+23] Bernhard Kerbl et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering". In: *ACM Transactions on Graphics* 42.4 (July 2023), pp. 1–14. ISSN: 1557-7368. [Link].

[KF09] Daphne Koller and Nir Friedman. *Probabilistic graphical models*. en. Adaptive Computation and Machine Learning series. London, England: MIT Press, July 2009.

[Kha+21] Misha Urooj Khan et al. "A Comparative Survey of LiDAR-SLAM and LiDAR based Sensor Technologies". In: *2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC)*. IEEE, July 2021. [Link].

[Kha+22] Muhammad Shahzad Alam Khan et al. "Investigation of Widely Used SLAM Sensors Using Analytical Hierarchy Process". In: *Journal of Sensors* 2022 (Jan. 2022). Ed. by Qiang Wu, pp. 1–15. [Link].

[KJD18] Jelena Kocic, Nenad Jovicic, and Vujo Drndarevic. "Sensors and Sensor Fusion in Autonomous Vehicles". In: *2018 26th Telecommunications Forum (TELFOR)*. IEEE, Nov. 2018. [Link].

[KJS16] Dwi Kumiawan, Agung Nugroho Jati, and Unang Sunarya. "A study of 2D indoor localization and mapping using Fast-SLAM 2.0". In: *2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*. IEEE, Sept. 2016. [Link].

[KK18] Giseop Kim and Ayoung Kim. "Scan Context: Egocentric Spatial Descriptor for Place Recognition Within 3D Point Cloud Map". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2018. [Link].

[KM07] Georg Klein and David Murray. "Parallel Tracking and Mapping for Small AR Workspaces". In: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE, Nov. 2007. [Link].

[KP18] Andreas Kamilaris and Francesc X. Prenafeta-Boldú. "Deep learning in agriculture: A survey". In: *Computers and Electronics in Agriculture* 147 (Apr. 2018), pp. 70–90. [Link].

[Kra+21] D. Krata et al. "Adaptive Smith Predictor Control Scheme for a Nonlinear Hydraulic System". In: *2021 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, Sept. 2021. [Link].

[KSC13] Christian Kerl, Jurgen Sturm, and Daniel Cremers. "Dense visual SLAM for RGB-D cameras". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Nov. 2013. [Link].

[Kum+11] Rainer Kummerle et al. "G2o: A general framework for graph optimization". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, May 2011. [Link].

[Kuz18] Maxim Kuzmin. "Review. Classification and Comparison of the Existing SLAM Methods for Groups of Robots". In: *2018 22nd Conference of Open Innovations Association (FRUCT)*. IEEE, May 2018. [Link].

[Laj+19] Pierre-Yves Lajoie et al. "Modeling Perceptual Aliasing in SLAM via Discrete–Continuous Graphical Models". In: *IEEE Robotics and Automation Letters* 4.2 (Apr. 2019), pp. 1232–1239. ISSN: 2377-3774. [Link].

[Lat90] Jean-Claude Latombe. *Robot motion planning*. en. 1991st ed. The Springer International Series in Engineering and Computer Science. New York, NY: Springer, Dec. 1990.

[LBD15] Tiancheng Li, Miodrag Bolic, and Petar M. Djuric. "Resampling Methods for Particle Filtering: Classification, implementation, and strategies". In: *IEEE Signal Processing Magazine* 32.3 (May 2015), pp. 70–86. [Link].

[LCL19] Tristan Laidlow, Jan Czarnowski, and Stefan Leutenegger. "DeepFusion: Real-Time Dense 3D Reconstruction for Monocular SLAM using Single-View Depth and Gradient Predictions". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2019. [Link].

[LCS11] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. "BRISK: Binary Robust invariant scalable keypoints". In: *2011 International Conference on Computer Vision*. IEEE, Nov. 2011. [Link].

[Leu+14] Stefan Leutenegger et al. "Keyframe-based visual–inertial odometry using nonlinear optimization". In: *The International Journal of Robotics Research* 34.3 (Dec. 2014), pp. 314–334. [Link].

[LHL08] Martin Liggins, David Hall, and James Llinas, eds. *Handbook of multisensor data fusion*. 2nd ed. Electrical Engineering & Applied Signal Processing Series. Boca Raton, FL: CRC Press, Sept. 2008.

[Li+15] Yali Li et al. "A survey of recent advances in visual feature detection". In: *Neurocomputing* 149 (Feb. 2015), pp. 736–751. ISSN: 0925-2312. [Link].

[Li+19] Jianlong Li et al. "Development of a Human–Robot Hybrid Intelligent System Based on Brain Teleoperation and Deep Learning SLAM". In: *IEEE Transactions on Automation Science and Engineering* 16.4 (Oct. 2019), pp. 1664–1674. [Link].

[Li+22a] Nanxi Li et al. "A Progress Review on Solid-State LiDAR and Nanophotonics-Based LiDAR Sensors". In: *Laser & Photonics Reviews* 16.11 (Aug. 2022). [Link].

[Li+22b] Shaopeng Li et al. "Overview of deep learning application on visual SLAM". In: *Displays* 74 (Sept. 2022), p. 102298. [Link].

[LM11] M. Labbe and F. Michaud. "Memory management for real-time appearance-based loop closure detection". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Sept. 2011. [Link].

[LM13] Mathieu Labbe and Francois Michaud. "Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation". In: *IEEE Transactions on Robotics* 29.3 (June 2013), pp. 734–745. [Link].

[LM14] Mathieu Labbe and Francois Michaud. "Online global loop closure detection for large-scale multi-session graph-based SLAM". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Sept. 2014. [Link].

[LM17] Mathieu Labbé and François Michaud. "Long-term online multi-session graph-based SPLAM with memory management". In: *Autonomous Robots* 42.6 (Nov. 2017), pp. 1133–1150. [Link].

[LM18] Mathieu Labbé and François Michaud. "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation". In: *Journal of Field Robotics* 36.2 (Oct. 2018), pp. 416–446. [Link].

[LM97] F. Lu and E. Milios. "Globally Consistent Range Scan Alignment for Environment Mapping". In: *Autonomous Robots* 4.4 (1997), pp. 333–349. [Link].

[Loe04] H. Loeliger. "An Introduction to factor graphs". In: *IEEE Signal Processing Magazine* 21.1 (Jan. 2004), pp. 28–41. [Link].

[Lon81] H. C. Longuet-Higgins. "A computer algorithm for reconstructing a scene from two projections". In: *Nature* 293.5828 (Sept. 1981), pp. 133–135. ISSN: 1476-4687. [Link].

[Low+16] Stephanie Lowry et al. "Visual Place Recognition: A Survey". In: *IEEE Transactions on Robotics* 32.1 (Feb. 2016), pp. 1–19. [Link].

[Low04] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110. [Link].

[LWG18] Ruihao Li, Sen Wang, and Dongbing Gu. "Ongoing Evolution of Visual SLAM from Geometry to Deep Learning: Challenges and Opportunities". In: *Cognitive Computation* 10.6 (Sept. 2018), pp. 875–889. [Link].

[LWG21] Ruihao Li, Sen Wang, and Dongbing Gu. "DeepSLAM: A Robust Monocular SLAM System With Unsupervised Deep Learning". In: *IEEE Transactions on Industrial Electronics* 68.4 (Apr. 2021), pp. 3577–3587. [Link].

[LZB16] Haomin Liu, Guofeng Zhang, and Hujun Bao. "Robust Keyframe-based Monocular SLAM for Augmented Reality". In: *2016 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, Sept. 2016. [Link].

[Mac+20] Steve Macenski et al. "The Marathon 2: A Navigation System". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2020. [Link].

[Mac+21] Wojciech Macherzyński et al. "The Use of Thermovision for Leak Detection in the Automotive Sector". In: *Pomiary Automatyka Robotyka* 25.3 (Sept. 2021), pp. 79–85. [Link].

[Mac+22a] Steven Macenski et al. "Robot Operating System 2: Design, architecture, and uses in the wild". In: *Science Robotics* 7.66 (2022), eabm6074. [Link].

[Mac+22b] Wojciech Macherzyśnki et al. *Method of leak detection, especially in closed-volume systems*. Pat.242047. Oct. 2022.

[Mar+21] Giulia Marchesi et al. "EnvSLAM: Combining SLAM Systems and Neural Networks to Improve the Environment Fusion in AR Applications". In: *ISPRS International Journal of Geo-Information* 10.11 (Nov. 2021), p. 772. [Link].

[Mar+22] José María Martínez-Otzeta et al. "RANSAC for Robotic Applications: A Survey". In: *Sensors* 23.1 (Dec. 2022), p. 327. ISSN: 1424-8220. [Link].

[Mar00] Robert C Martin. "Design principles and design patterns". In: *Object Mentor* 1.34 (2000), p. 597.

[Mar08] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. English. Paperback. Pearson, Aug. 1, 2008, p. 464. ISBN: 978-0132350884.

[Mar17] Robert Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design (Robert C. Martin Series)*. English. Paperback. Pearson, Sept. 10, 2017, p. 432. ISBN: 978-9352865123.

[Mat+23] Hidenobu Matsuki et al. *Gaussian Splatting SLAM*. 2023. [Link].

[Mcc+18] John Mccormac et al. "Fusion++: Volumetric Object-Level SLAM". In: *2018 International Conference on 3D Vision (3DV)*. IEEE, Sept. 2018. [Link].

[Mea82] Donald Meagher. "Geometric modeling using octree encoding". In: *Computer Graphics and Image Processing* 19.2 (June 1982), pp. 129–147. ISSN: 0146-664X. [Link].

[Men+20] Tong Meng et al. "A survey on machine learning for data fusion". In: *Information Fusion* 57 (May 2020), pp. 115–129. [Link].

[Mer14] Dirk Merkel. "Docker: lightweight linux containers for consistent development and deployment". In: *Linux journal* 2014.239 (2014), p. 2.

[Mil+20] Ben Mildenhall et al. "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis". In: *Computer Vision – ECCV 2020*. Springer International Publishing, 2020, pp. 405–421. [Link].

[MM20] Rafael Muñoz-Salinas and R. Medina-Carnicer. "UcoSLAM: Simultaneous localization and mapping by fusion of keypoints and squared planar markers". In: *Pattern Recognition* 101 (May 2020), p. 107193. ISSN: 0031-3203. [Link].

[MM21] Alexey Merzlyakov and Steve Macenski. "A Comparison of Modern General-Purpose Visual SLAM Approaches". In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sept. 2021. [Link].

[MMT15] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. "ORB-SLAM: A Versatile and Accurate Monocular SLAM System". In: *IEEE Transactions on Robotics* 31.5 (Oct. 2015), pp. 1147–1163. [Link].

[Moh+19] Sherif A. S. Mohamed et al. "A Survey on Odometry for Autonomous Navigation Systems". In: *IEEE Access* 7 (2019), pp. 97466–97486. ISSN: 2169-3536. [Link].

[Mon+02] Michael Montemerlo et al. "FastSLAM: A factored solution to the simultaneous localization and mapping problem". In: 2002, pp. 593–598.

[Mon+03] Michael Montemerlo et al. “FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges”. In: Cited by: 869. 2003, pp. 1151–1156.

[Mor15] Francisco Angel Moreno. “Stereo Visual SLAM for Mobile Robots Navigation”. PhD thesis. University of Malaha, 2015.

[Mor80] Hans Peter Moravec. “Obstacle avoidance and navigation in the real world by a seeing robot rover”. PhD thesis. Stanford University, 1980.

[Mou21] Radouan Ait Mouha. “Deep Learning for Robotics”. In: *Journal of Data Analysis and Information Processing* 09.02 (2021), pp. 63–76. [Link].

[MT17] Raul Mur-Artal and Juan D Tardós. “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras”. In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262.

[New+02] P. Newman et al. “Explore and return: experimental validation of real-time concurrent mapping and localization”. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*. IEEE, 2002. [Link].

[New+11] Richard A. Newcombe et al. “KinectFusion: Real-time dense surface mapping and tracking”. In: *2011 10th IEEE International Symposium on Mixed and Augmented Reality*. IEEE, Oct. 2011. [Link].

[New99] Paul Newman. “On the Structure and Solution of the Simultaneous Localisation and Map Building Problem”. In: 1999.

[NLD11] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. “DTAM: Dense tracking and mapping in real-time”. In: *2011 International Conference on Computer Vision*. IEEE, Nov. 2011. [Link].

[NNB] D. Nister, O. Naroditsky, and J. Bergen. “Visual odometry”. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* IEEE. [Link].

[Och+21] Marcin Ochman et al. “RGB-D Odometry for Autonomous Lawn Mowing”. In: *Artificial Intelligence and Soft Computing*. Springer International Publishing, 2021, pp. 81–90. [Link].

[Och19] Marcin Ochman. “Hybrid approach to road detection in front of the vehicle”. In: *IFAC-PapersOnLine* 52.8 (2019), pp. 245–250. [Link].

[OG21] Adrian Ostrowski and Piotr Gaczkowski. *Software Architecture with C++: Design modern systems using effective architecture concepts, design patterns, and techniques with C++20*. English. Paperback. Packt Publishing, Apr. 23, 2021, p. 540. ISBN: 978-1838554590.

[Ope24] OpenCV. *Strona OpenCV*. 2024. URL: `http://opencv.org` (visited on 02/18/2024).

[Özy+17] Onur Özyeşil et al. "A survey of structure from motion." In: *Acta Numerica* 26 (May 2017), pp. 305–364. ISSN: 1474-0508. [Link].

[Pal+19] Emanuele Palazzolo et al. "ReFusion: 3D Reconstruction in Dynamic Environments for RGB-D Cameras Exploiting Residuals". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Nov. 2019. [Link].

[Pal+22] Sabita Pal et al. "Evolution of Simultaneous Localization and Mapping Framework for Autonomous Robotics—A Comprehensive Review". In: *Journal of Autonomous Vehicles and Systems* 2.2 (Apr. 2022). [Link].

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[Pit+11] Benjamin Pitzer et al. "Towards perceptual shared autonomy for robotic mobile manipulation". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE, May 2011. [Link].

[PLC19] Rudra P K Poudel, Stephan Liwicki, and Roberto Cipolla. *Fast-SCNN: Fast Semantic Segmentation Network*. 2019. [Link].

[Qua+19] Meixiang Quan et al. "Accurate Monocular Visual-Inertial SLAM Using a Map-Assisted EKF Approach". In: *IEEE Access* 7 (2019), pp. 34289–34300. [Link].

[RI19] European Commission. Directorate General for Research and Innovation. *Recognising the importance of software in research: Research Software Engineers (RSEs), a UK example*. Publications Office, 2019. [Link].

[RLC22] Antoni Rosinol, John J. Leonard, and Luca Carlone. *NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields*. 2022. [Link].

[RLC23] Antoni Rosinol, John J. Leonard, and Luca Carlone. "NeRF-SLAM: Real-Time Dense Monocular SLAM with Neural Radiance Fields". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2023. [Link].

[ROS] ROS 2 Community. *ROS 2 AMCL Library*. Accessed on 10.12.2023. URL: `https://github.com/ros-planning/navigation2/tree/main/nav2_amcl`.

[Ros+19] Antoni Rosinol et al. "Incremental Visual-Inertial 3D Mesh Generation with Structural Regularities". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, May 2019. [Link].

[Ros+20a] Antoni Rosinol et al. "3D Dynamic Scene Graphs: Actionable Spatial Perception with Places, Objects, and Humans". In: *Robotics: Science and Systems XVI*. Robotics: Science and Systems Foundation, July 2020. [Link].

[Ros+20b] Antoni Rosinol et al. "Kimera: an Open-Source Library for Real-Time Metric-Semantic Localization and Mapping". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2020. [Link].

[Ros+21a] David M. Rosen et al. "Advances in Inference and Representation for Simultaneous Localization and Mapping". In: *Annual Review of Control, Robotics, and Autonomous Systems* 4.1 (May 2021), pp. 215–242. [Link].

[Ros+21b] Antoni Rosinol et al. "Kimera: From SLAM to spatial perception with 3D dynamic scene graphs". In: *The International Journal of Robotics Research* 40.12-14 (Dec. 2021), pp. 1510–1546. [Link].

[Rot17] Stephan Roth. *Clean C++: Sustainable Software Development Patterns and Best Practices with C++ 17*. English. Paperback. Apress, Sept. 29, 2017, p. 308. ISBN: 978-1484227923.

[RP10] David L. Ripley and Thomas Politzer. "Vision Disturbance after TBI". In: *Neurorehabilitation* 27.3 (2010), pp. 215–216. ISSN: 1053-8135. [Link].

[Rub+11] Ethan Rublee et al. "ORB: An efficient alternative to SIFT or SURF". In: *2011 International Conference on Computer Vision*. IEEE, Nov. 2011. [Link].

[Sae+15] Sajad Saeedi et al. "Multiple-Robot Simultaneous Localization and Mapping: A Review". In: *Journal of Field Robotics* 33.1 (July 2015), pp. 3–46. [Link].

[Sak+18] Atsushi Sakai et al. *PythonRobotics: a Python code collection of robotics algorithms*. 2018. [Link].

[Sar+20] Paul-Edouard Sarlin et al. “SuperGlue: Learning Feature Matching With Graph Neural Networks”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. [Link].

[SC22] Lukas von Stumberg and Daniel Cremers. “DM-VIO: Delayed Marginalization Visual-Inertial Odometry”. In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022), pp. 1408–1415. [Link].

[SC86] Randall C. Smith and Peter C. Cheeseman. “On the Representation and Estimation of Spatial Uncertainty”. In: *The International Journal of Robotics Research* 5 (1986), pp. 56–68.

[SCG18] Dominik Schlegel, Mirco Colosi, and Giorgio Grisetti. “ProSLAM: Graph SLAM from a Programmer’s Perspective”. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2018. [Link].

[Sch+23] Stefan Schubert et al. *Visual Place Recognition: A Tutorial*. 2023. [Link].

[SE18] Tixiao Shan and Brendan Englot. “LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2018. [Link].

[Sem+22] Sofiya Semenova et al. “A modular, extensible framework for modern visual SLAM systems”. In: *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*. MobiSys ’22. ACM, June 2022. [Link].

[SF11] Davide Scaramuzza and Friedrich Fraundorfer. “Visual Odometry [Tutorial]”. In: *IEEE Robotics & Automation Magazine* 18.4 (Dec. 2011), pp. 80–92. [Link].

[SH10] Bruno Steux and Oussama El Hamzaoui. “tinySLAM: A SLAM algorithm in less than 200 lines C-language program”. In: *2010 11th International Conference on Control Automation Robotics & Vision*. IEEE, Dec. 2010. [Link].

[Sha+17] Shital Shah et al. “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles”. In: *Springer Proceedings in Advanced Robotics*. Springer International Publishing, Nov. 2017, pp. 621–635. ISBN: 9783319673615. [Link].

[Sha+20] Tixiao Shan et al. “LIO-SAM: Tightly-coupled Lidar Inertial Odometry via Smoothing and Mapping”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2020. [Link].

[SK16] Bruno Siciliano and Oussama Khatib, eds. *Springer Handbook of Robotics*. Springer International Publishing, 2016. [Link].

[SK18] Muhammad Sualeh and Gon-Woo Kim. “Simultaneous Localization and Mapping in the Epoch of Semantics: A Survey”. In: *International Journal of Control, Automation and Systems* 17.3 (Dec. 2018), pp. 729–742. [Link].

[Sko+21] Magda Skoczeń et al. “Obstacle Detection System for Agricultural Mobile Robot Application Using RGB-D Cameras”. In: *Sensors* 21.16 (Aug. 2021), p. 5292. [Link].

[SMD12] Hauke Strasdat, J.M.M. Montiel, and Andrew J. Davison. “Visual SLAM: Why filter?” In: *Image and Vision Computing* 30.2 (Feb. 2012), pp. 65–77. [Link].

[SMT18] Muhamad Risqi U. Saputra, Andrew Markham, and Niki Trigoni. “Visual SLAM and Structure from Motion in Dynamic Environments”. In: *ACM Computing Surveys* 51.2 (Feb. 2018), pp. 1–36. [Link].

[SS23] Taha Samavati and Mohsen Soryani. “Deep learning-based 3D reconstruction: a survey”. In: *Artificial Intelligence Review* 56.9 (Jan. 2023), pp. 9175–9219. [Link].

[SSC90] Randall Smith, Matthew Self, and Peter Cheeseman. “Estimating Uncertain Spatial Relationships in Robotics”. In: *Autonomous Robot Vehicles*. Springer New York, 1990, pp. 167–193. [Link].

[SSM23] Ricardo B. Sousa, Héber M. Sobreira, and António Paulo Moreira. “A systematic literature review on long-term localization and mapping for mobile robots”. In: *Journal of Field Robotics* 40.5 (Apr. 2023), pp. 1245–1322. [Link].

[SSP19] Thomas Schops, Torsten Sattler, and Marc Pollefeys. “BAD SLAM: Bundle Adjusted Direct RGB-D SLAM”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2019. [Link].

[SSS19] Shinya Sumikura, Mikiya Shibuya, and Ken Sakurada. “OpenVSLAM”. In: *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, Oct. 2019. [Link].

[Stu+12] Jrgen Sturm et al. "A benchmark for the evaluation of RGB-D SLAM systems". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, Oct. 2012. [Link].

[Tak+18] Talha Takleh Omar Takleh et al. "A Brief Survey on SLAM Methods in Autonomous Vehicle". In: *International Journal of Engineering & Technology* 7.4.27 (Nov. 2018), p. 38. ISSN: 2227-524X. [Link].

[Tan+16] Shengjun Tang et al. "Enhanced RGB-D Mapping Method for Detailed 3D Indoor and Outdoor Modeling". In: *Sensors* 16.10 (Sept. 2016), p. 1589. [Link].

[Tan+23] Matthew Tancik et al. "Nerfstudio: A Modular Framework for Neural Radiance Field Development". In: *ACM SIGGRAPH 2023 Conference Proceedings*. SIGGRAPH '23. 2023.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. en. Intelligent Robotics and Autonomous Agents series. London, England: MIT Press, Aug. 2005.

[TBG22] Konstantinos A. Tsintotas, Loukas Bampis, and Antonios Gasteratos. "The Revisiting Problem in Simultaneous Localization and Mapping: A Survey on Visual Loop Closure Detection". In: *IEEE Transactions on Intelligent Transportation Systems* 23.11 (Nov. 2022), pp. 19929–19953. [Link].

[TC20] Baihui Tang and Sanxing Cao. "A Review of VSLAM Technology Applied in Augmented Reality". In: *IOP Conference Series: Materials Science and Engineering* 782.4 (Mar. 2020), p. 042014. [Link].

[TD21] Zachary Teed and Jia Deng. "DROID-SLAM: Deep Visual SLAM for Monocular, Stereo, and RGB-D Cameras". In: vol. 20. 2021, pp. 16558–16569.

[Tes23] Tesla, Inc. *Tesla Vision Update: Replacing Ultrasonic Sensors with Tesla Vision*. 2023. URL: `http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm` (visited on 06/25/2023).

[The+22] Charalambos Theodorou et al. "Visual SLAM algorithms and their application for AR, mapping, localization and wayfinding". In: *Array* 15 (Sept. 2022), p. 100222. [Link].

[TLF10] E. Tola, V. Lepetit, and P. Fua. "DAISY: An Efficient Dense Descriptor Applied to Wide-Baseline Stereo". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.5 (May 2010), pp. 815–830. [Link].

[TLZ23] Qin Tang, Jing Liang, and Fangqi Zhu. “A comparative review on multi-modal sensors fusion based on deep learning”. In: *Signal Processing* 213 (Dec. 2023), p. 109165. [Link].

[Tou+22] Ali Tourani et al. “Visual SLAM: What Are the Current Trends and What to Expect?” In: *Sensors* 22.23 (Nov. 2022), p. 9297. [Link].

[Tri+00] Bill Triggs et al. “Bundle Adjustment — A Modern Synthesis”. In: *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2000, pp. 298–372. ISBN: 9783540444800. [Link].

[TS18] Shaharyar Ahmed Khan Tareen and Zahra Saleem. “A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK”. In: *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. IEEE, Mar. 2018. [Link].

[Tsi24] Mariot Tsitoara. *Beginning Git and GitHub: Version Control, Project Management and Teamwork for the New Developer*. English. Paperback. Apress, Mar. 15, 2024, p. 330. ISBN: 979-8868802140.

[TUI17] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. “Visual SLAM algorithms: a survey from 2010 to 2016”. In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (June 2017). [Link].

[Ull+20] Inam Ullah et al. “Simultaneous Localization and Mapping Based on Kalman Filter and Extended Kalman Filter”. In: *Wireless Communications and Mobile Computing* 2020 (June 2020), pp. 1–12. [Link].

[Ume91] S. Umeyama. “Least-squares estimation of transformation parameters between two point patterns”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.4 (Apr. 1991), pp. 376–380. ISSN: 0162-8828. [Link].

[Wan+20] Wenshan Wang et al. “TartanAir: A Dataset to Push the Limits of Visual SLAM”. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2020. [Link].

[Wan+23] Xiaotian Wang et al. “An Overview of Key SLAM Technologies for Underwater Scenes”. In: *Remote Sensing* 15.10 (May 2023), p. 2496. [Link].

[WB94] Greg Welch and Gary Bishop. “An Introduction to the Kalman Filter”. In: 1994.

[Whe+16] Thomas Whelan et al. “ElasticFusion: Real-time dense SLAM and light source estimation”. In: *The International Journal of Robotics Research* 35.14 (Sept. 2016), pp. 1697–1716. [Link].

[WTT03] Chieh-Chih Wang, C. Thorpe, and S. Thrun. “Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas”. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 1. 2003, 842–849 vol.1. [Link].

[WWN20] Zhangjing Wang, Yu Wu, and Qingqing Niu. “Multi-Sensor Fusion in Automated Driving: A Survey”. In: *IEEE Access* 8 (2020), pp. 2847–2868. [Link].

[Xia+20] Linlin Xia et al. “A survey of image semantics-based visual simultaneous localization and mapping: Application-oriented solutions to autonomous navigation of mobile robots”. In: *International Journal of Advanced Robotic Systems* 17.3 (2020), p. 1729881420919185. [Link]. eprint: `https://doi.org/10.1177/1729881420919185`.

[Xu+24] Yifeng Xu et al. “The Research and Development of an Educational SLAM AVG Based on Modular Design Concept”. In: *Lecture Notes in Networks and Systems*. Springer Nature Singapore, 2024, pp. 529–553. ISBN: 9789819984985. [Link].

[Yan+20] Nan Yang et al. “D3VO: Deep Depth, Deep Pose and Deep Uncertainty for Monocular Visual Odometry”. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2020. [Link].

[YBH15] Khalid Yousif, Alireza Bab-Hadiashar, and Reza Hoseinnezhad. “An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics”. In: *Intelligent Industrial Systems* 1.4 (Nov. 2015), pp. 289–311. [Link].

[YI21] Omer Faruk Yanik and Hakki Alparslan Ilgin. “Comparison of Power Consumption of Modern SLAM Methods on Various Datasets”. In: *2021 International Conference on Technological Advancements and Innovations (ICTAI)*. IEEE, Nov. 2021. [Link].

[You+17] Georges Younes et al. “Keyframe-based monocular SLAM: design, survey, and future directions”. In: *Robotics and Autonomous Systems* 98 (Dec. 2017), pp. 67–88. [Link].

[Yu+18] Chao Yu et al. “DS-SLAM: A Semantic Visual SLAM towards Dynamic Environments”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2018. [Link].

[Zaf+18] Mubariz Zaffar et al. “Sensors, SLAM and Long-term Autonomy: A Review”. In: *2018 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. IEEE, Aug. 2018. [Link].

[Zha+19a] WenLong Zhao et al. "Review of SLAM Techniques For Autonomous Underwater Vehicles". In: *Proceedings of the 2019 International Conference on Robotics, Intelligent Control and Artificial Intelligence*. ACM, Sept. 2019. [Link].

[Zha+19b] Yong Zhao et al. "GSLAM: A General SLAM Framework and Benchmark". In: *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, Oct. 2019. [Link].

[Zha+22] Song Zhang et al. "Visual SLAM for underwater vehicles: A survey". In: *Computer Science Review* 46 (Nov. 2022), p. 100510. [Link].

[Zha+23] Lintong Zhang et al. "Hilti-Oxford Dataset: A Millimeter-Accurate Benchmark for Simultaneous Localization and Mapping". In: *IEEE Robotics and Automation Letters* 8.1 (2023), pp. 408–415. [Link].

[Zha00] Z. Zhang. "A flexible new technique for camera calibration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334. ISSN: 0162-8828. [Link].

[ZN20] Tianwei Zhang and Yoshihiko Nakamura. "Humanoid Robot RGB-D SLAM in the Dynamic Human Environment". In: *International Journal of Humanoid Robotics* 17.02 (Feb. 2020), p. 2050009. [Link].

[ZS14] Ji Zhang and Sanjiv Singh. "LOAM: Lidar Odometry and Mapping in Real-time". In: *Robotics: Science and Systems X*. RSS2014. Robotics: Science and Systems Foundation, July 2014. [Link].

[ZS16] Ji Zhang and Sanjiv Singh. "Low-drift and real-time lidar odometry and mapping". In: *Autonomous Robots* 41.2 (Feb. 2016), pp. 401–416. ISSN: 1573-7527. [Link].

[ZWS21] Xiwu Zhang, Lei Wang, and Yan Su. "Visual place recognition: A survey from deep learning perspective". In: *Pattern Recognition* 113 (May 2021), p. 107760. [Link].

[ZZZ19] Jiyuan Zhang, Gang Zeng, and Hongbin Zha. "Structure-aware SLAM with planes and lines in man-made environment". In: *Pattern Recognition Letters* 127 (Nov. 2019), pp. 181–190. [Link].