Algorithm for Mobile Security Platform Employment in an Unknown Indoor Environment

Author
Abid Ali
Regn Number
00000148441

Supervisor
Dr Waqar Shahid

DEPARTMENT OF MECHATRONICS ENGINEERING
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY
ISLAMABAD

JUL, 2019

Algorithm for Mobile Security Platform Employment in an Unknown Indoor Environment

Author

Abid Ali

Regn Number

00000148441

A thesis submitted in partial fulfillment of the requirements for the degree of

MS Mechatronics Engineering

Thesis Supervisor:

Dr Waqar Shahid

Thesis Supervisor's Signature:____

DEPARTMENT OF MECHATRONICS ENGINEERING

COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING

NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY, ISLAMABAD

JUL, 2019

Declaration

I certify that this research work titled "*Algorithm for Mobile Security Platform Employment in an Unknown Indoor Environment"* is my own work. The work has not been presented elsewhere for assessment. The material that has been used from other sources it has been properly acknowledged / referred.

Signature of Student

Abid Ali

00000148441

Language Correctness Certificate

This thesis has been read by an English expert and is free of typing, syntax, semantic, grammatical and spelling mistakes. Thesis is also according to the format given by the university.

Signature of Student

Abid Ali

Registration Number

00000148441

Signature of Supervisor

Dr Waqar Shahid

Copyright Statement

Acknowledgements

I am thankful to my Creator Allah Subhana-Watala to have guided me throughout this work at every step and for every new thought which You setup in my mind to improve it. Indeed I could have done nothing without Your priceless help and guidance. Whosoever helped me throughout the course of my thesis, whether my parents or any other individual was Your will, so indeed none be worthy of praise but You.

I am particularly thankful to my parents who brought me up to what I am today and to my wife for her continuous support.

I would also like to express special thanks to my supervisor Dr. Waqar Shahid for his help throughout my thesis and also for Image Processing and Machine Design courses which he has taught me.

I would like to express my special gratitude to Dr. Kunwar Faraz Ahmed for his round the clock availability to guide and support me for completion of my thesis.

I would also like to acknowledge the support and guidance of Dr. Umar Shahbaz Khan for being on my thesis guidance and evaluation committee.

Finally, I would like to express my gratitude to all the individuals who have rendered valuable assistance to my study.

Dedicated to my parents who could have not done more for me,

my siblings whose prayers always carried me along,

my beloved wife whose support helped me in completing thesis.

Abstract

Designing and fabrication of mobile security platform is a relatively new development in security conscious world. These mobile security platforms are being developed keeping in view their employment environment. In general, these platforms perform two main tasks, first is the basic task of mapping the environment in which it will operate as well as self localization and thereafter navigation in that particular environment, second is the specific security task of surveillance. Mapping and localization has become one of the mainstream research areas in mobile robotics and is generally performed using simultaneous localization & mapping (SLAM) algorithms. These SLAM algorithm are broadly classified in three categories, first is based on calculation methods like Kalman Filter or the particle filter, hybrid comprising of both filters and graph-based, second is based on sensors like vision, range measurement devices and odometry, third is based on structure like online SLAM and full SLAM. Surveillance task is performed by assessment algorithms supported by different sensors like acoustic, vibration, passive infrared, microwave, optical, ultrasonic and vision.

This work focuses on mapping and localization part of a mobile security platform which has to be employed in an unknown indoor environment. In this context, an in-depth study has been carried out on SLAM algorithms developed to-date along with different sensors available. Visual SLAM algorithm is tested with latest ZED stereo camera in an unknown indoor environment for its use in mobile security platform.

Key Words: *Mobile Security Platform, SLAM, vSLAM, Stereo Camera.*

Table of Contents

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

The research work presented in this dissertation is about selecting a SLAM algorithm and testing it with a sensor in an unknown indoor environment for its further use in Mobile Security Platform. The thesis has been organized into chapters. Chapter 1 develops a background about different categories of mobile security platforms, SLAM algorithms and sensors available. Chapter 2 presents literature review of relevant researches as regards to this thesis. Chapter 3 presents comparative study and analysis as regards to this thesis. Chapter 4 is about selection of suitable SLAM algorithm/ sensor and its testing. Chapter 5 which is also the last chapter, concludes this thesis.

1.1 Background, Scope and Motivation

Post 9/11 world has witnessed the rise of a global phenomenon of security consciousness. Pakistan, having a geo-strategic importance and being the only Islamic country having nuclear arsenal, also have security related concerns regarding its sensitive installations. These installations are manned/ monitored round the clock owing to sensitive nature and prevalent security situation. Prolonged and isolated duties at these sensitive installations put a heavy toll on employed manpower. Technological advancement in recent past has addressed the security related concerns with automated surveillance incorporating intelligent robots. Dedicated man-based monitoring system can be strengthened with use of automated surveillance incorporating intelligent robots. Robots can be used both in passive and active surveillance role, latter involve the complexities.

During the last fifteen years, mobile platforms have been developed worldwide incorporating different techniques for performing the security related tasks. A mobile security platform is a mobile robot designed to navigate autonomously in an unidentified environment for performing surveillance tasks. It generally performs three tasks, 1st task is mapping the environment in which it will operate along with self localization, 2nd task is of surveillance i.e. people/ object detection, 3rd task is to communicate the information acquired through a viable media. Fusion of these tasks in a single mobile security platform poses a considerable challenge. Furthermore, basic task of mapping and localization is accomplished using techniques like SLAM, odometry, vSLAM, VO. Similarly, specific task of surveillance is performed using techniques like template matching, color based, part/ shape based etc. Performance analysis of these techniques used to perform basic task of a mobile security platform in an indoor environment is still a challengeable task.

1.2 Evolution of Mobile Security Platform

History of developing mobile robots dates back to mid 20th century. Various mobile robots have been developed to-date for security related application in indoor and outdoor environments.

1.2.1 Mobile Detection Assessment Response System (MDARS) Program

Aim of MDARS program was to develop a robotic system that would be able to look over storage areas. Research prototypes developed under MDARS program were named as ROBART.

ROBART I (1980–1982) was able to detect an intruder only.

ROBART II (1982–1992) was able to detect along with assessment aim of eliminating the nuisance alarm. Sensors incorporated in ROBART II were acoustic, vibration, passive infrared, microwave, optical, ultrasonic, and video.

ROBART III (1992-2010) focused on generating response by passive IR, microwave, LIDAR and vision sensors.

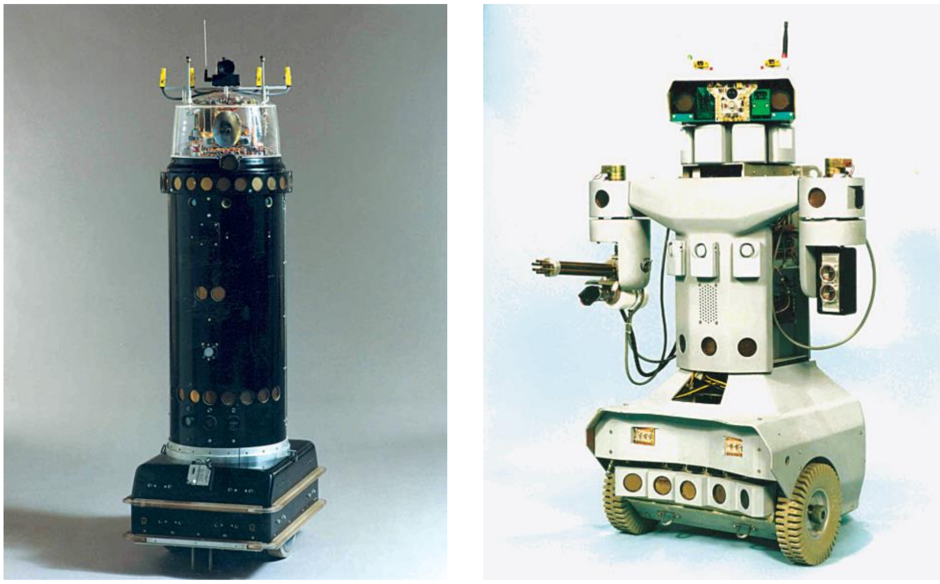

Figure 1.1 ROBART II and ROBART III

Most advanced robotic security system under this program was MDARS Exterior platform. Sensors incorporated in this design include FMCW radar and FLIR camera.

Figure 1.2 MDARS Exterior Platform

1.2.2 Team of Robotic Agents for Surveillance (2000)

This robotic system was designed for security and surveillance tasks. It consisted of two types of robotic agents; ranger and scouts. Ranger was a heavy duty platform used to transport scouts which were small mobile sensor platforms. Ranger communicated with scouts which observed the area of interest.

Figure 1.3 Team of Robotic Agents

1.2.3 RoboGaurd (2001)

It was a semi-autonomous mobile security device having cameras and sensors which navigated in the area and sent video streams to human watch guards.

Figure 1.4 RoboGaurd

1.2.4 Robotic Security Guard (2004)

It was designed for surveillance of indoor environments. This robot was able to keep watch over a given area, identify persons and forward sensory details.

Figure 1.5 Robotic Security Guard

1.2.5 Mobile Autonomous Robotic Vehicle for Indoor Navigation (MARVIN) (2004)

MARVIN was planned to act as a security robot in indoor environments. Speech recognition and speech synthesis software was embedded in the robot and could show emotional states.

Figure 1.6 MARVIN

1.2.6 Airport Night Surveillance Expert Robot (ANSER) (2005)

It consisted of a UGV having a GPS module for patrolling at airports. It needed supervision of human. It was used for localization and navigation purpose.

Figure 1.7 ANSER

1.2.7 Multisensory Robotic Platform (2010)

It was developed to navigate in the indoor environment autonomously and perform surveillance tasks. Sensors incorporated in the robotic platform (PeopleBot) included monocular camera, a laser scanner, and an RFID device.

Figure 1.8 Multisensory Robotic Platform

1.3 Indoor Mobile Robots

Mobile robots are available commercially for indoor application. Some of them are listed below:-

1.3.1 Khepera IV

The Khepera IV robot is the latest version of this series designed for flat surfaces and generally used for research in the field of Navigation, Artificial Intelligence, Control, and Real-Time Programming. Sensors incorporated in Khepera IV are encoders, inertial measurement unit

(IMU), infra red (IR) proximity sensors, ambient light sensors, ultrasonic sensors. It offers integration with GNU C/C++ compilers.

Figure 1.9 Khepra IV

1.3.2 Ridgeback Robot

It is an omnidirectional robot designed for carrying high loads in a constrained environment. Sensors incorporated in Ridgeback Robot are inertial measurement unit (IMU), laser range finders, and light detection and ranging (LiDAR). It offers integration with ROS and Gazebo.

Figure 1.10 Ridgeback Omnidirectional Robot

1.3.3 Boxer Mobile Robot

It is intended for robotic solutions and generally used for research. Sensors incorporated in Boxer Mobile Robot are encoders, front-facing stereo camera, light detection and ranging (LiDAR) and sound detection and ranging (SONAR). It offers integration with ROS.

Figure 1.11 Boxer Mobile Robot

1.3.4 TurtleBot3 Robot

It is a robot designed for robotic solutions. Sensor incorporated in TurtleBot3 is inertial measurement unit (IMU) having gyroscope, accelerometer and magnetometer. It offers integration with ROS.

Figure 1.12 TurtleBot3 Robot

1.3.5 Mobile Base Apollo

It is a middle sized robotic base robot designed to move autonomously in public places and can execute mapping function and autonomous navigation. Sensors incorporated in Mobile Base Apollo are depth camera, ultrasonic sensors and LiDAR. It offers integration with ROS and provided with SDK to develop third party applications including iOS / Android.

Figure 1.13 Mobile Base Apollo

1.4 SLAM

Initial work on SLAM was carried out by Smith, Self and Cheeseman. Based on their work, Hugh Durrant-Whyte and John J. Leonard developed SLAM. In SLAM, map of unknown environment is built by the robot and navigation is carried out using map simultaneously.

SLAM consists of a number of steps; Landmark extraction, data association, state estimation, state update and landmark update.

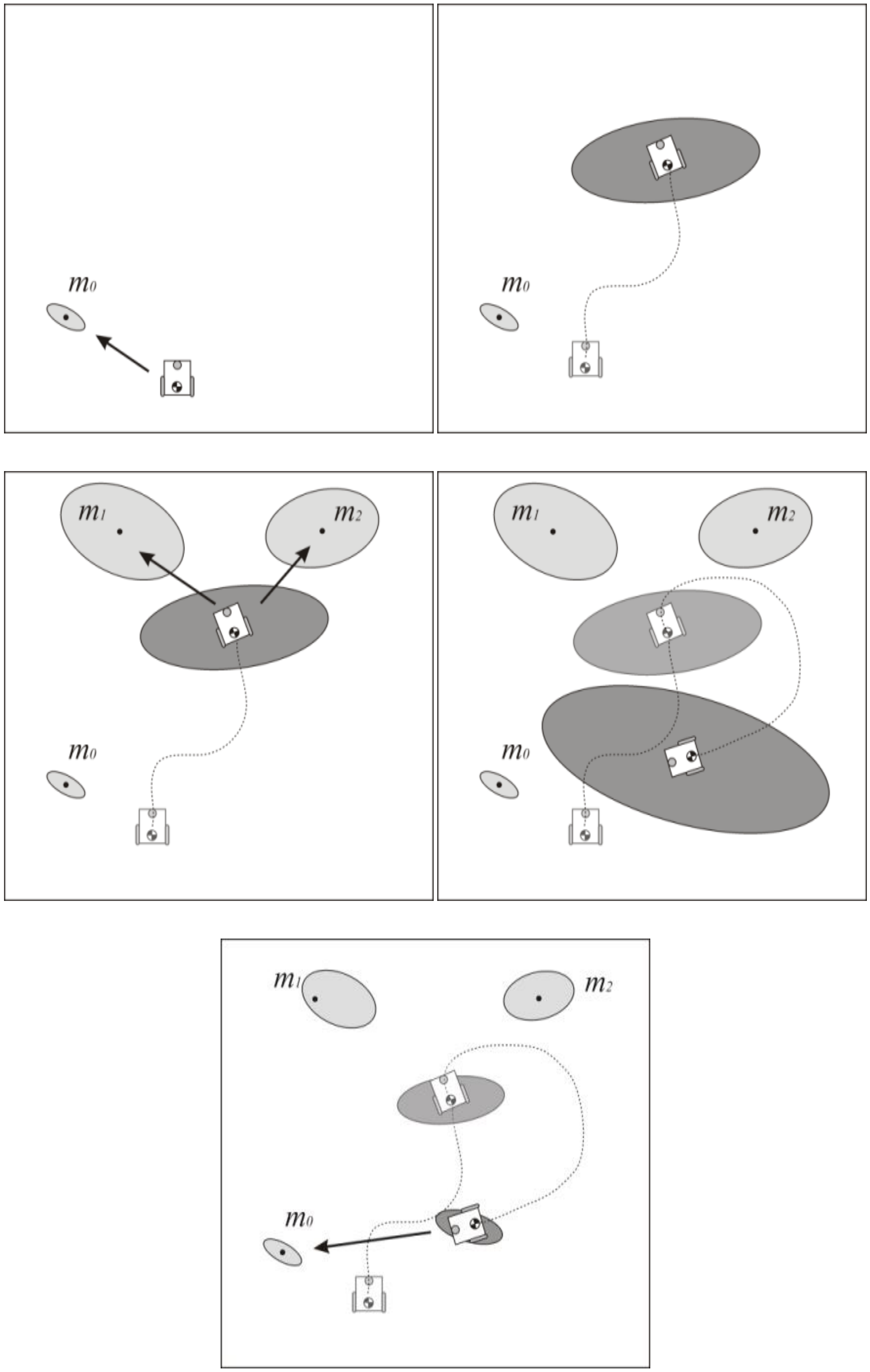

Figure 1.14 SLAM Process

The architecture of a SLAM system is given below: -

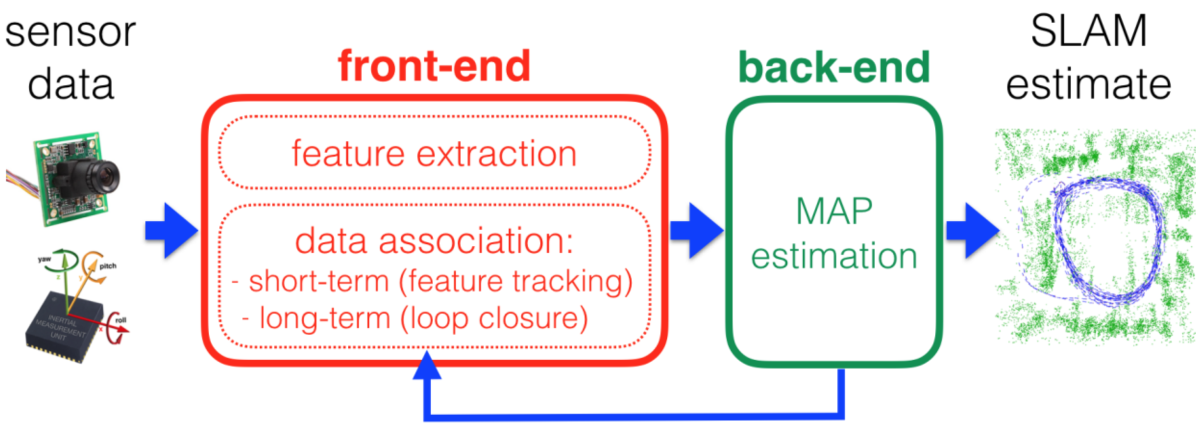

Figure 1.15 SLAM Architecture

1.4.1 Classes of SLAM

1.4.1.1 Feature based SLAM

In this SLAM, landmarks and environment model is known which used to estimate the robot path and the map [1]. There are four different techniques to the state estimation [2]: extended Kalman filter (EKF), information extended filter (IEF), particles filter (PF), and optimization techniques (graph based) [3].

1.4.1.2 Pose based SLAM

In this SLAM, landmark positions are ignored and estimation is carried out of robot state trajectory only. Path estimation is carried out by optimization techniques [4], information [5], and particles filtering methods [6].

1.4.1.3 Appearance based SLAM

Metric information and the landmark positions are not used in this SLAM; instead only visual images are utilized to recognize the place. It is very common that these appearance techniques are used complementary to any metric SLAM method to detect loop closures [7].

1.4.2 SLAM Challenges

SLAM is mainly popular for indoor applications due to non usage of GPS application and its major challenges include: representation of the map, the correspondence problem, uncertainty management, computational complexity and consistency. Detail of SLAM challenges for long term autonomy is as under:-

- **Robustness**. SLAM system failure can be of two types i.e. algorithmic and hardware related. Algorithmic failure may occur during data association or due to perceptual aliasing in which same sensor signature is obtained from different sensors. SLAM systems (in particular, the data association modules) require extensive parameter tuning in order to work correctly for a given scenario. Basic assumption that the world is static is used in most SLAM methods whereas in real the world has dynamic behavior. Hardware related failure is associated with sensor failures.
- **Scalability**. For long term application, the size of the factor graph grows with the passage of time which requires high memory and more computational time. Therefore, SLAM application in long term remains a challenge.

1.5 Visual SLAM

SLAM with only visual input is termed as vSLAM. vSLAM has higher technical difficulty in comparison to other range sensors. Tracking and mapping (TAM) term is used in vSLAM in which tracking is done with every frame and mapping is carried out at a certain time interval.

1.5.1 Basic Elements of vSLAM

vSLAM has three basic modules; Initialization, Tracking, Mapping and two additional modules; Relocalization and Global Optimization. Detail of each is as under:-

- **Initialization**. Global coordinate system is defined in this module and a certain portion of area is constructed as map in the global coordinate system.
- **Tracking**. In this module, camera pose of the image is estimated with respect to map by tracking of constructed map in the image.
- **Mapping**. In this module, camera view unknown regions and initial map is expanded.
- **Relocalization**. This module caters for tracking failure due to any disturbance by computing camera pose again with respect to map.

- **Global Map Optimization**. This is performed to suppress the accumulative estimation error in the map. Loop closing is a technique to acquire the reference information and loop detection is done for obtaining geometrically consistent map.

1.5.2 vSLAM Challenges

vSLAM faces some problems in practical situations which are described below:-.

- **Purely Rotational Motion**. Purely rotational motion creates problem when mapping with monocular camera based vSLAM, however, RGB-D vSLAM can handle this issue.
- **Map Initialization**. Accurate estimation in vSLAM depends on map initialization. Baseline should be wide in order to obtain an accurate initial map. However, ideal camera motion is difficult in practical scenarios.
- **Intrinsic Camera Parameters Estimation**. Most vSLAM algorithms assume known intrinsic camera parameters. This requires calibration of camera before vSLAM applications.
- **Rolling Shutter Distortion**. Camera pose estimation is difficult to rolling shutter distortion.
- **Scale Ambiguity**. Absolute scale information is required with monocular vSLAM applications. Human body parts are used in general to determine the absolute scale based on assumption of small size of body parts.

1.6 Sensors

Sensors are classified as exteroceptive and proprioceptive. Exteroceptive sensors include IR sensors, ultrasonic sensors, LiDAR, SONAR and camera (computer vision) sensors. Infrared and ultrasonic distance sensors are small and cheap, however, are prone to cross-talk between sensors. LIDAR sensors are accurate, however, are expensive. SONAR has accuracy issues. One advantage of vision based systems is their ability to register 3D information, however, have issues with regards to accuracy, calibration and the nature of the measurements. Proprioceptive sensors include encoders, accelerometers, gyroscopes which have cumulative error issues due to inherent noise.

In the last ten years, there is a clear tendency for using vision as the sensor for solving SLAM problem. The main reason is that cameras are able to estimate range with the introduction of depth cameras.

CHAPTER 2 : LITERATURE REVIEW

2.1 SLAM

Durrant Whyte and Bailey in their two surveys [8, 9] reviewed the first 20 years of the SLAM problem and this is termed as the classical age (1986-2004). Main probabilistic formulations for SLAM, including approaches based on Extended Kalman Filters, Rao-Blackwellised Particle Filters, and maximum likelihood estimation were introduced in this age. It delineated the basic challenges connected to efficiency and robust data association. Subsequent period is termed as algorithmic-analysis age (2004-2015), and is partially covered by Dissanayake et al. in [10]. Fundamental properties of SLAM, including observability, convergence, and consistency were studied in this age. Key role of sparsity towards efficient SLAM solvers was also understood, and the main open-source SLAM libraries were developed. Latest period is termed as robust-perception age (2016-to-date), which is characterized by key requirements of robust performance, high level understanding, resource awareness and task driven perception.

Year	Topic	Reference
2006	Probabilistic approaches and data association	Durrant Whyte and Bailey [8,9]
2008	Filtering approaches	Aulinas et al. [11]
2011	SLAM back-end	Grisetti et al. [12]
2011	Observability, consistency and convergence	Dissanayake et al. [10]
2012	Visual Odometry	Scaramuzza and Fraundofer [13, 14]
2016	Multi Robot SLAM	Saeedi et al. [15]
2016	Visual place recognition	Lowry et al. [16]
2016	SLAM in the Handbook of Robotics	Stachiness et at. [17]
2016	Theoretical Aspects	Haung and Dissanayake [18]

Table 2.1 Surveying the Surveys and Tutorials

2.2 vSLAM

2.2.1 Feature Based Methods

2.2.1.1 MonoSLAM (2003)

In MonoSLAM, camera motion and 3D structure of an unknown environment are simultaneously estimated using an extended Kalman filter (EKF). MonoSLAM computational cost increases in proportion to the size of an environment; hence, it is difficult to achieve real-time computation in large environments.

2.2.1.2 Parallel Tracking and Mapping PTAM (2007)

PTAM method separated the tracking and the mapping into different threads on CPU which enabled handling of thousands of feature points in the map. Closed-loop detection and pose-graph optimization is used before bundle adjustment in order to overcome local minimum problem in bundle adjustment.

2.2.1.3 ORB SLAM (2015)

It includes multi-threaded tracking, mapping, and closed-loop detection, and the map is optimized using pose-graph optimization and bundle adjustment. ORB-SLAM can use monocular, stereo and RGB-D cameras.

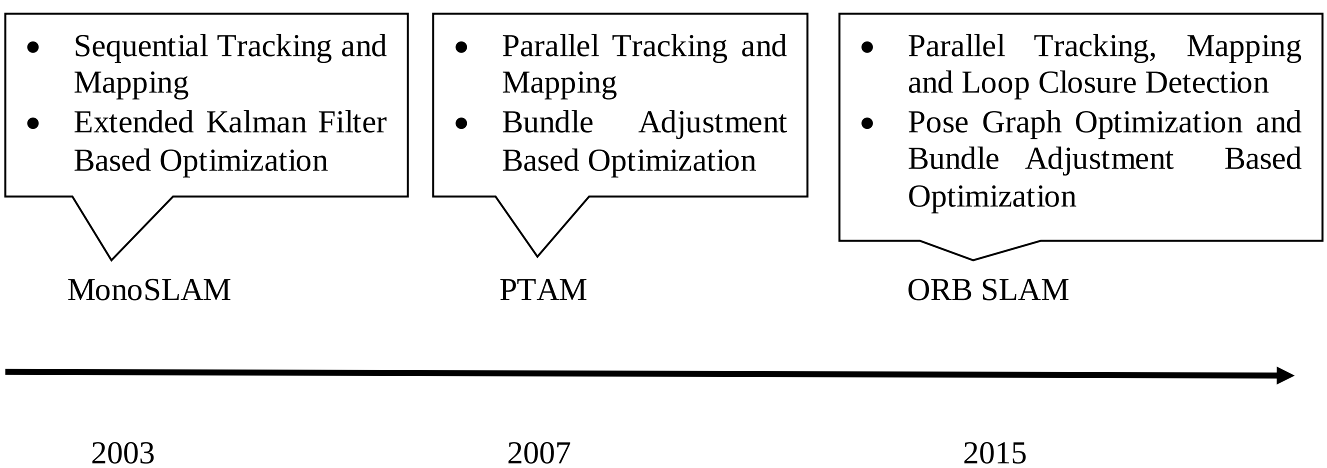

Figure 2.1 Feature Based Methods

2.2.2 Direct Methods

2.2.2.1 Direct Tracking and Mapping DTAM (2011)

- Stereo measurement performs map initialization.
- Estimation of camera motion is done using synthetic view generation from the reconstructed map.
- Estimation of depth information is done and thereafter optimization carried out.

2.2.2.2 Large Scale Direct SLAM (2014)

- Initial depth values for each pixel are random values.
- Estimation of camera motion is done using synthetic view generation from the reconstructed map.
- Only high intensity areas are reconstructed.
- Geometrically consistent map is obtained by 7 DoF pose-graph optimization.

2.2.2.3 Semi Direct Visual Odometry SVO and Direct Sparse Odometry DSO (2015)

In SVO, tracking is done by feature point matching and the mapping is done by the direct method. DSO is a fully direct method and focus on local geometric consistency only, hence, is not a vSLAM rather VO.

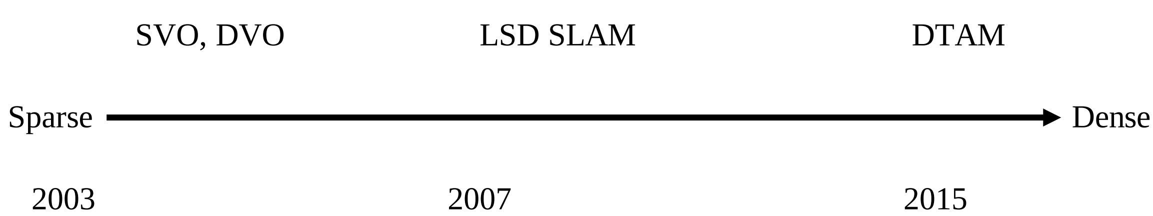

Figure 2.2 Direct Methods

	Method	Map Density	Global Optimization	Loop Closure
MonoSLAM	Feature	Sparse	No	No
PTAM	Feature	Sparse	Yes	No
ORB SLAM	Feature	Sparse	Yes	Yes
DTAM	Direct	Dense	No	No
LSDSLAM	Direct	Semi Dense	Yes	Yes
SVO	Semi Direct	Sparse	No	No
DSO	Direct	Sparse	No	No
KinectFusion	RGB-D	Dense	No	No
Dense Visual SLAM	RGB-D	Dense	Yes	Yes
ElasticFusion	RGB-D	Dense	Yes	Yes
SLAM++	RGB-D	Dense	Yes	Yes

Table 2.2 vSLAM Algorithms

Author	Type of Sensing Device	Core of the Solution
Davison (2003)	Monocular camera	MonoSLAM (EKF)
Nister et al.(2004)	Stereo or monocular cameras	Visual odometry
Saez and Escolano (2006)	Stereo camera	Global Entropy Minimization Algorithm
Mouragnon et al.(2006)	Monocular camera	Visual odometry + Local bundle adjustment
Klein and Murray (2007)	Monocular camera	Parallel Tracking and Mapping (Visual odometry + Bundle Adjustment)
Ho and Newman (2007)	Monocular camera and laser	Delayed state formulation
Clemente et al.(2007)	Monocular camera	Hierarchical map + EKF

Lemaire et al.(2007)	Stereo or monocular cameras	EKF
Milford (2008)	Monocular camera	RatSLAM (models of the rodent hippocampus)
Scaramuzza and Siegwart (2008)	Omnidirectional camera	Visual odometry
Eade and Drummond (2008)	Monocular camera	GraphSLAM
Paz et al. (2008)	Stereo camera	Conditionally independent divide and conquer (EKF)
Angeli et al. (2008)	Monocular camera	EKF
Cummins and Newman (2008)	Monocular camera mounted on a *pan-tilt*	Fast Appearance Based Mapping (FAB-MAP)
Pinies and Tardos (2008)	Monocular camera	Conditionally independent local maps (EKF)
Konolige et al. (2009)	Stereo camera + IMU	Visual odometry + sparse bundle adjustment
Williams (2009)	Monocular camera	Hierarchical map + EKF + Visual odometry
Kaess and Dellaert (2010)	Multi-camera rig	Expectation maximization + Standard bundle adjustment
Botterill et al. (2010)	Monocular camera	Odometria visual + Bag of words
Mei et al. (2010)	Stereo camera	Visual odometry + Relative bundle adjustment + FAB-MAP

Table 2.3 vSLAM Algorithms [19]

CHAPTER 3: COMPARATIVE STUDY

Various SLAM and vSLAM algorithms were reviewed in Chapter 2. This chapter presents a comparative study of various SLAM algorithms.

3.1 Comparative Study I – Evaluation of the Modern Visual SLAM Methods [20]

In this study of 2015, comparative analysis of four recent vSLAM algorithms (ORB SLAM, Open RatSLAM, LSD-SLAM and L-SLAM) was carried out. Comparison was done on virtual server of Amazon EC2 with Intel Xeon CPU 2.4GHz, 8 GB RAM and sensor used was monocular camera. Algorithms were tested using subset of TUM RGB benchmark dataset [21].

3.1.1 SLAM Algorithms Overview

Table 3.1 gives the overview of SLAM algorithms compared in this study.

Algorithm	Core Estimated Values	Used Information	Main Contribution	Assumptions
ORB-SLAM	• Camera transformation (by feature matching error minimization); • 3d feature position;	RGB image	• Usage of ORB as environment features;	• Robot movement between 2 consecutive frames is relatively small.
RatSLAM	• Robot position and orientation (by Pose Cell Network); • Visual odometry (optionally);	• RGB image; • Odometry;	• First biological-inspired SLAM;	• Excitatory links weight matrix has Gaussian distribution.
LSD-SLAM	• Camera transformation (by photometric error minimization); • Inverse depth map (by pixelwise Kalman filter);	RGB image	• Tracking structure of environment; • Monocular camera is the only sensor;	• Inverse depth is Gaussian; • Noises are Gaussian;

Table 3.1 vSLAM Algorithms Overview [20]

3.1.2 Results and Conclusions

Dataset	Median (m)			Mean (m)			Standard Deviation (m)		
	LSD	ORB	Rat	LSD	ORB	Rat	LSD	ORB	Rat
fr1_desk	0.75 (7%)	0.05	1.21	0.75	0.05	1.12	X	0.02	0.22
fr1_room	0.05 (-)	0.19 (87%)	0.97 (93%)	0.05	0.18	0.95	0	0.12	0.06
fr1_xyz	0.17 (93%)	0.04	0.22	0.26	0.05	0.22	0.18	0.03	0.01
fr2_desk	0.43 (7%)	0.74	2.56 (93%)	0.43	0.76	2.56	X	0.09	0.01
fr2_pioneer_slam2	0.55 (7%)	0.24	1.62	0.55	0.62	1.62	X	0.62	0.006
fr3_large_cabinet	0.80 (20%)	1.75 (40%)	1.88	0.71	1.75	1.88	0.14	0.07	0.001
fr3_long_office_house hold	0.54 (-)	1.11	1.38	0.54	1.10	1.26	0.22	0.02	0.25

Table 3.2 RMSE Comparison in the TUM RGB-D Benchmark [20]

- Stable tracking was not obtained during the testing of vSLAM algorithms with TUM RGB-D dataset and trajectory error was high.
- Robust results were not generated. Preprocessing is required in ORB SLAM, slow processing in RatSLAM, tweaking done on LSD-SLAM (non deterministic behavior).
- RMSE values do not support any of vSLAM algorithms for practical application.

3.2 Comparative Study II - Comparative Analysis of ROS-based Monocular SLAM Methods for Indoor Navigation [22]

In this study of 2016, comparative analysis of four recent vSLAM algorithms (ORB SLAM [22], REgularized MOnocular Depth Estimation REMODE SLAM [23], LSD-SLAM [24] and Dense Piecewise Planar Tracking and Mapping DPPTAM [25]) was carried out. Unmanned Ground Vehicle (UGV) having a wide-angle full HD webcam and a USB WideCam F1005 was used in an indoor office environment having light colored walls.

Vision System Configuration of UGV prototype		Genius WideCam F100 Camera	
Parameter	**Configuration**	**Parameter**	**Configuration**
Processor	Intel Core^{TM} i3-4160 CPU @ 3.60GHz x 4	Image Sensor	1080p Full HD pixel CMOS
GPU	GeForce GT 740M	Video resolution	VGA/720P HD/1080p FHD
RAM	8 GB	Interface	USB 2.0
Camera	Genius WideCam F100	Image Resolution	12MP, 1920x1080, 128x720, 640x480
OS	Linux	Frame rates	up to 30fps
ROS	Jade Turtle	Lens	120 degrees
Driver	ROS usb_cam webcam	Shutter	Rolling shutter

Table 3.3 Characteristics of UGV Hardware [22]

3.2.1 SLAM Algorithms Overview

Table 3.4 gives the overview of four SLAM algorithms compared in this study.

Parameter	ORB-SLAM	REMODE	LSD-SLAM	DPPTAM
Type of method	Feature-based	Feature-based	Direct	Direct
CUDA-enabled	No	Yes	No	No
Separate odometry module	No (Built-in)	Yes (SVO)	No (Built-in)	No (Built-in)
Camera trajectory module	Yes	No	Yes	No
Visualization	Built-in	ROS/RViz	Built-in	ROS/RViz
Noise level	Low	Middle	High	Low
Visual odometry quality	Good	Good	Poor	Poor

Table 3.4 vSLAM Algorithms Overview [22]

3.2.2 Observations and Conclusions

- Only ORB SLAM was able to reconstruct closed-loop UGV trajectory.
- Results for indoor environment are approximately same for feature-based and direct vSLAM methods.
- Corners and other features were detected by almost all vSLAM methods.
- vSLAM methods tested showed poor results in detection of light colored walls of indoor environment.
- Robustness of vSLAM ROS based monocular methods is questionable.

3.3 Comparative Study III - Comparison of ROS based Visual SLAM Methods in Homogeneous Indoor Environment [27]

In this study of 2017, comparative analysis of five SLAM algorithms (Hector SLAM, monocular ORB SLAM, monocular Dense Piecewise Planar Tracking and Mapping DPPTAM, stereo ZEDfu, Real-Time Appearance-Based Mapping RTAB-Map with Kinect 2.0 Depth Sensor) was carried out. Unmanned ground vehicle (UGV) equipped with computing system having following specifications and sensors were used in a tapped indoor office environment having light colored walls.

Parameters	Configuration
UGV hardware	
Processor	Intel Core i3-416 CPU @ 3.60GHz x 4
GPU	GeForce GT 740M
RAM	8 GB
Sensors	
LIDAR	HOKUYO UTM-30LX[a]
Camera	Basler acA2000-50gc GigE[b]
Stereo camera	Stereolabs ZED cameraa[c]
RGB-D sensor	Microsoft Kinect 2.0[d]
Software	
OS	Ubuntu 14.04
ROS	Indigo Igloo

Table 3.5 Configuration of UGV Hardware[27]

3.3.1 SLAM Algorithms Overview and Results

Table 3.6 gives the overview of SLAM algorithms compared in this study.

Parameter	**ORB-SLAM**	**DPPTAM**	**ZEDfu**	**RTAM-Map**
Visualization	Built-in	ROS/RViz	Built-in	Built-in
CUDA	No	No	Yes	No
Odometry	No	No	Yes	Yes
Trajectory	Yes	Yes	No	No
Noise level	Low	Low	Low	Middle
3D map quality	Low	Average	Good	Good
3D map type	Sparse	Dense	Dense	Dense
Odometry quality (Max. deviation)	Good (0.43m)	Poor (4.26m)	Good (0.32m)	Good (0.67m)

Table 3.6 SLAM Algorithms Overview [27]

Trajectory deviation of different SLAM methods used in this study is as under:-

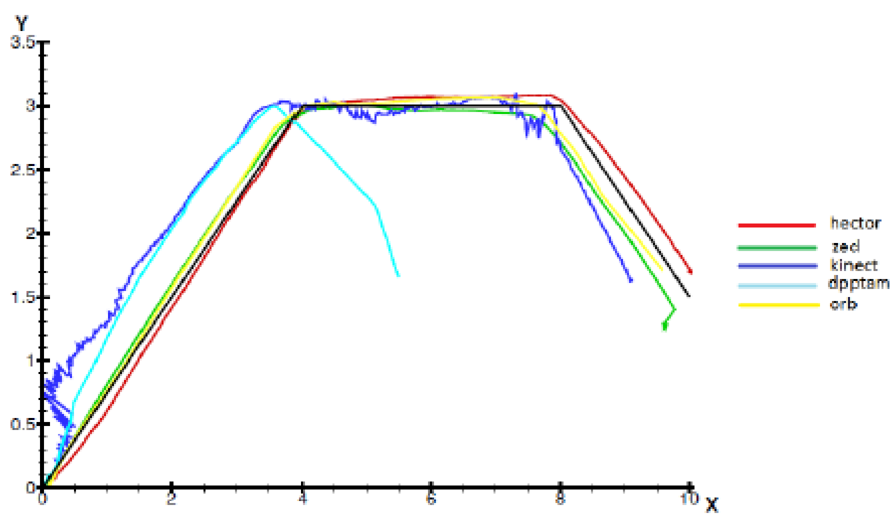

Figure 3.1 Comparison of Trajectories (in meters) of different SLAM Methods [27]

SLAM method	Sensors	Average deviation, m	Maximal deviation, m
Hector SLAM	2D LIDAR	0.11	0.18
ORB-SLAM	Monocular Camera	0.19	0.43
DPPTAM	Monocular Camera	2.05	4.26
ZEDfu	Stereo ZED camera	0.14	0.32
RTAB-Map	Kinect 2.0 depth sensor	0.42	0.67

Table 3.7 Trajectory Deviations of different SLAM Methods [27]

3.3.2 Observations and Recommendations

- Estimation of absolute scale of the map cannot be done by monocular SLAM algorithms; hence, localization is not possible. Size of the map objects can be verified by ground truth or by estimating the displacement value of camera.
- Robustness of ORB SLAM is greater than DPPTAM in indoor environments.
- ORB SLAM is recommended for applications requiring high performance.
- DPPTAM is recommended for building a dense area map of obstacles and the environment having fewer features provided hardware has high performance power.
- Stereo cameras and RGB-D sensors have similar resource consumption.
- Stereo camera should be used for building a map with high depth.
- RBD-D sensor is recommended for use in colorless walls or mirrors.

3.4 Comparative Study IV - Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment [28]

In this study of 2018, comparative analysis of following eleven SLAM algorithms with different sensors was carried out.

Year	System	Sensor
2007	GMapping	2D Lidar
2007	Parallel Tracking and Mapping (PTAM)	mono
2011	Hector SLAM	2D Lidar
2014	Semi-direct Visual Odometry (SVO)	mono
2014	Large Scale Direct monocular SLAM (LSD SLAM)	mono
2014	Real-Time Appearance-Based Mapping (RTAB map)	stereo
2015	ORB SLAM	mono, stereo
2015	Dense Piecewise Parallel Tracking and Mapping (DPPTAM)	Mono
2016	Direct Sparse Odometry (DSO)	Mono
2016	Cartographer	2D Lidar
2017	Stereo Parallel Tracking and Mapping (S-PTAM)	stereo

Table 3.8 SLAM Systems (ROS based) [28]

Hardware configuration of Labcar platform was as under:-

Parameters	Configuration
Chassis	**4WD Traxxas # 74076**
Hardware	**Jetson TX 1**
Processor **GPU** **RAM**	**Quad ARM A57** **NVIDIA Maxwell** **4 GB**
Sensors	
Lidar **Camera** **Stereo camera**	**Hokuyo UTM-30LX** **Basler acA1300-200uc** **ZED camera**
Software	
Jetpack **OS** **ROS**	**3.1** **Ubuntu 16.04** **Kinetic Kame**

Table 3.9 Configuration of Labcar Platform [28]

Hardware configuration of ground station was as under:-

Parameters	Configuration
Processor **GPU** RAM	**Intel Core i7 6500U** **NVIDIA GeForce GTX 950M** 12 GB
Software	Configuration
OS ROS	**Ubuntu 16.04** Kinetic Kame

Table 3.10 Configuration of Ground Station [28]

3.4.1 Results

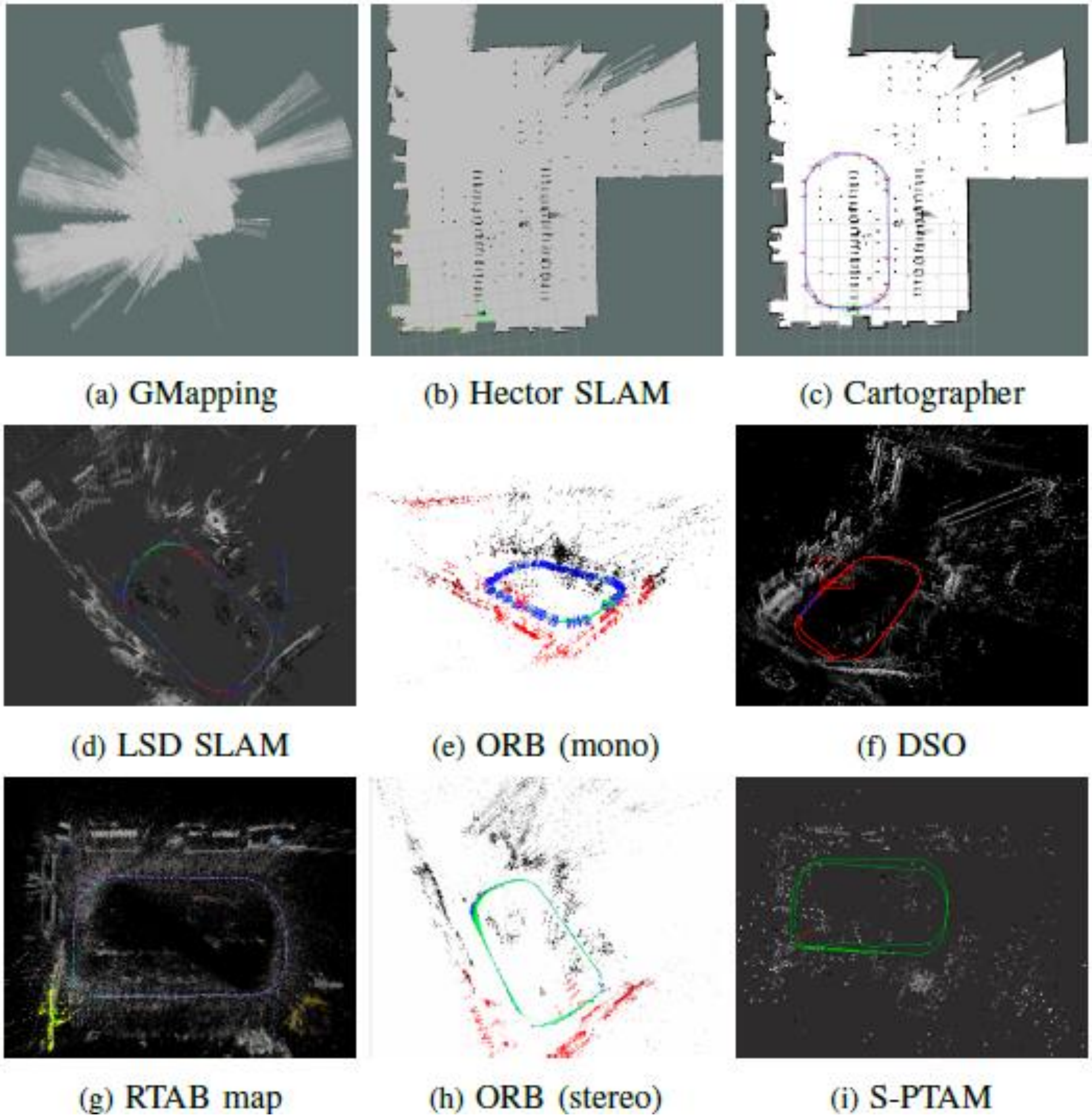

Figure 3.2 Maps generated by Various SLAM Methods [28]

Absolute trajectory error of various SLAM systems based on Hector SLAM trajectory is as under:-

System	**RMSE** (m)	**Mean** (m)	**Median** (m)	**Std.** (m)	**Min** (m)	**Max** (m)
Cartographer	0.024	0.017	0.013	0.021	0.001	0.07
LSD SLAM	0.301	0.277	0.262	0.117	0.08	0.553
ORB SLAM (mono)	0.166	0.159	0.164	0.047	0.047	0.257
DSO	0.459	0.403	0.419	0.219	0.007	0.764
ZEDfu	0.726	0.631	0.692	0.358	0.002	1.323
RTAB map	0.163	0.138	0.110	0.085	0.004	0.349
ORB SLAM (stereo)	0.190	0.151	0.102	0.115	0.004	0.414
S-PTAM (no loop cl.)	0.338	0.268	0.244	0.206	0.001	0.768
S-PTAM (loop cl.)	0.295	0.257	0.242	0.145	0.006	1.119

Table 3.11 Absolute Trajectory Error of Various SLAM Systems [28]

3.4.2 Discussion

- UGV localization and mapping was accurate in Hector SLAM and Cartographer. Gmapping was inaccurate. Cartographer is more robust to environmental changes.
- Monocular PTAM, SVO, DPPTAM failed in tracking and could not handle scale ambiguity.
- Monocular LSD SLAM, ORB SLAM, DSO with additional scale recovery module can be used for localization.
- Stereo ZEDfu, RTAB-Map, ORB SLAM, S-PTAM provide metric information.

- RTAB-Map had better results for localization, however, tracking is lost close to indistinct walls
- Stereo ORB SLAM is the most robust system.

CHAPTER 4 : ORB SLAM 2 AND ZED STEREO CAMERA

Based on the comparative studies of Chapter 3, SLAM algorithm selected for use in mobile security platform is stereo ORB SLAM 2 and sensor selected is ZED stereo Camera.

4.1 ORB SLAM 2 Overview

ORB SLAM 2 is an improved version of ORB SLAM developed in 2015. Algorithmic structure of ORB SLAM is based on three threads, tracking, local mapping, loop closing and sensor used in monocular camera.

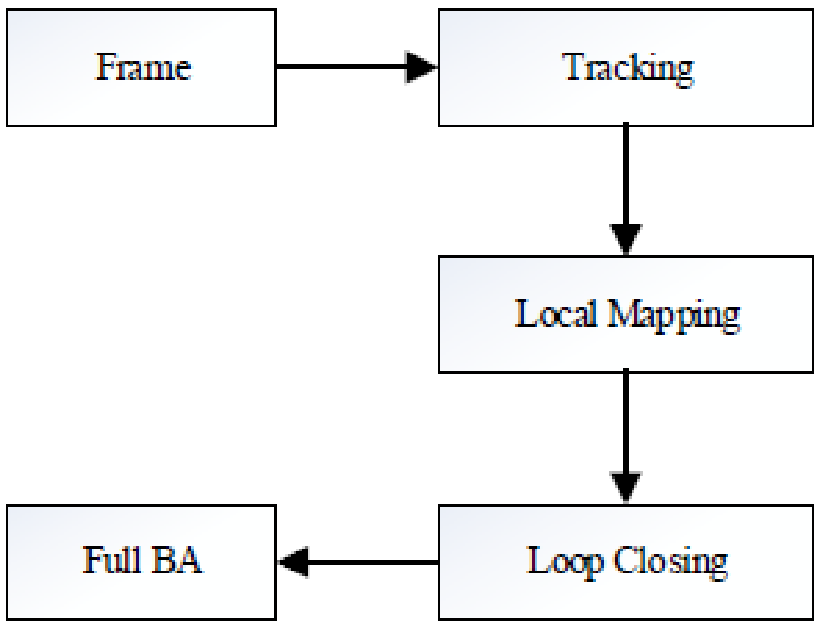

Figure 4.1 ORB-SLAM Overview

In tracking process, first task performed is of extracting ORB features from the image frame. Feature extraction method used is “Feature from Accelerated Segment Test” named as FAST in which edge features are found from the input image. After extracting features, descriptor is created using a method of “Binary Robust Independent Elementary Features” named as BRIEF. Localization of the camera is carried out by using each frame. Thereafter keyframes are selected which are used to build the map. Most useful frames are used in ORB SLAM based on ‘survival of the fittest’ approach. Feature matching between chosen keyframe and previous keyframe gives the initial pose estimation on which optimization is carried out. .

Relocalization is done in case tracking is lost. Successful tracking step gives as camera pose estimation and an initial set of feature matches.

Keyframes and map points after tracking process are used to build the map. Keyframes are placed in placed in a covisibility graph. Culling of keyframes is done to avoid unbounded growth of the graph. Similarly map points are also removed if they are observed by too few keyframes at a time.

Loop closing is done in final thread to check whether this location has been visited or not. Loop closing is essential as it allows the system to update beliefs about the location and determine the drift accumulated while the loop was traversed.

Sim3 is solved by the RANSAC (Random Sample Consensus) framework, and then Sim3 is optimized by re-matching and g2o (General Graphic Optimization) to correct the pose of the current keyframe.

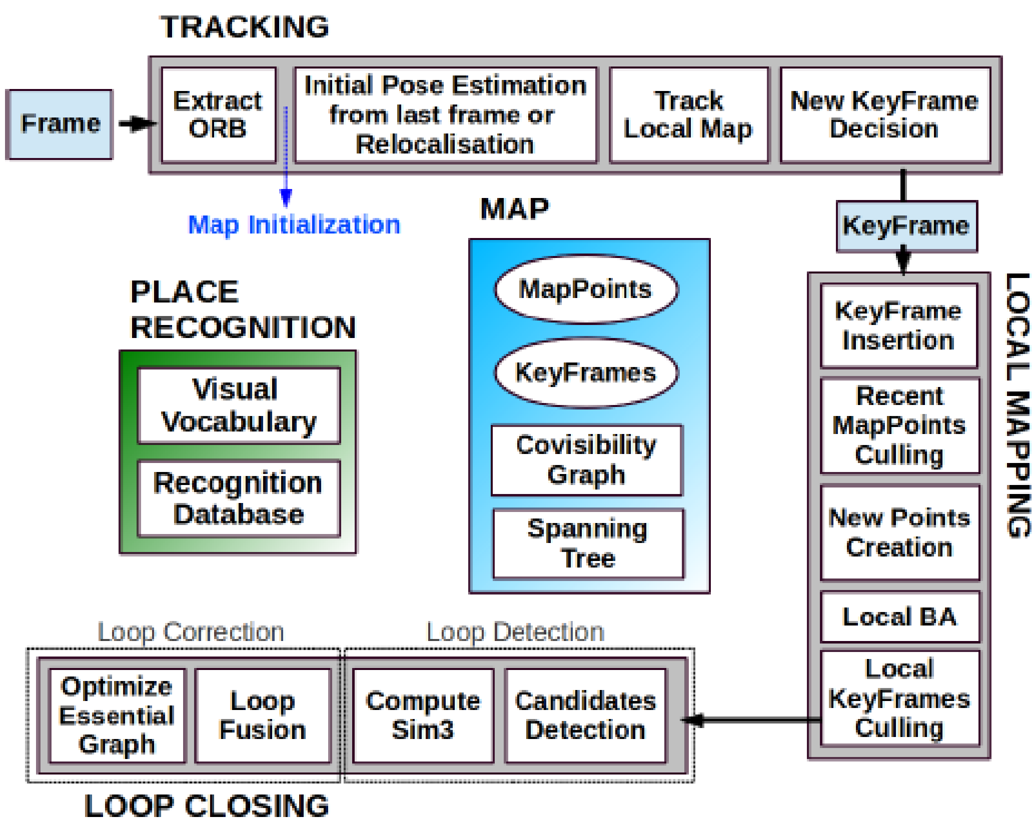

Figure 4.2 ORB SLAM [29]

Algorithmic structure of ORB SLAM 2 is also based on three threads, tracking, local mapping, loop closing, however, in loop closing, bundle adjustment is performed. Furthermore, ORB SLAM 2 can be used with different sensors lile monoculara camera, stereo camera and RGB-D camera.

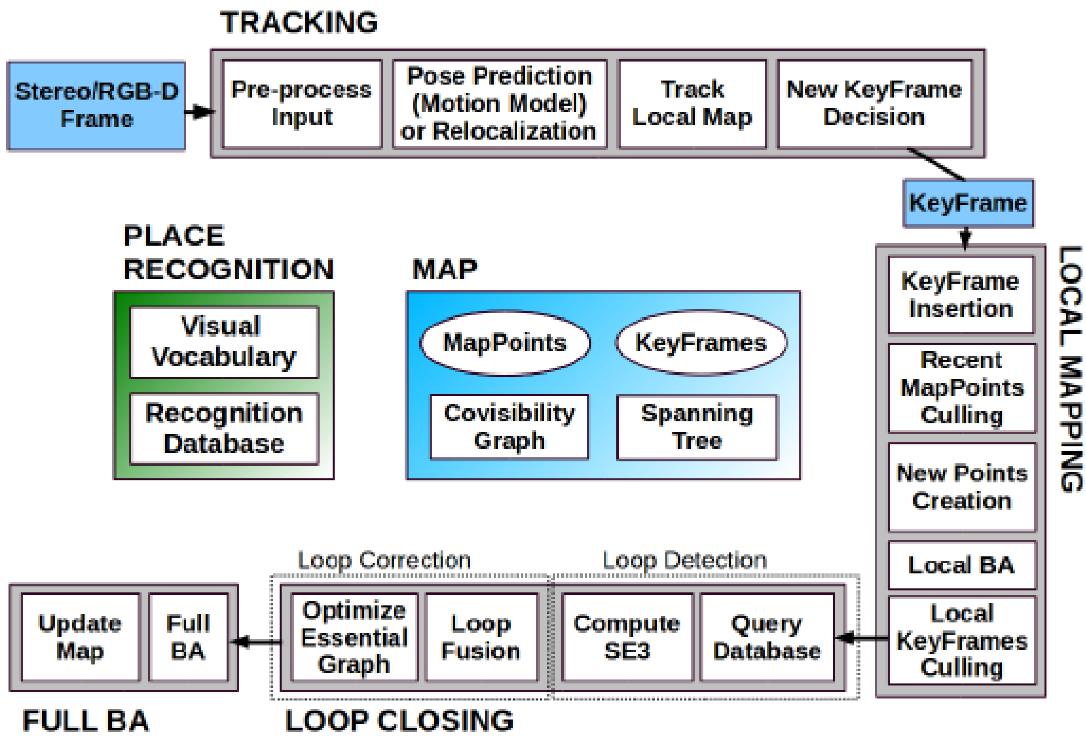

Figure 4.3 ORB SLAM 2 [30]

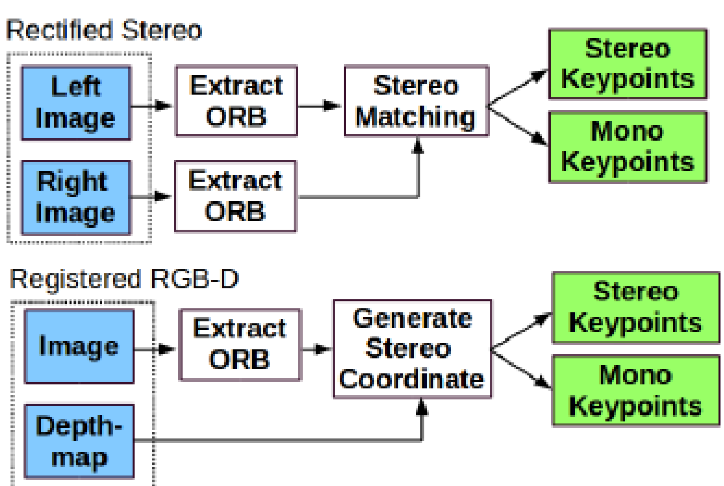

Figure 4.4 ORB-SLAM 2 Input Pre-processing [30]

4.2 ZED Camera Overview

The ZED is a camera with dual lenses. It captures high-definition 3D video with a wide field of view and outputs two synchronized left and right video streams.

Depth perception is the ability to determine distances between objects and see the world in three dimensions. Up until now, depth sensors have been limited to perceiving depth at short range and indoors, restricting their application to gesture control and body tracking. Using stereo vision, the ZED is the first universal depth sensor. The camera works indoors and outdoors, contrary to active sensors such as structured-light or time of flight.

Using computer vision and stereo SLAM technology, the ZED also understands its position and orientation in space, offering full 6DOF positional tracking. In VR/AR, this means you can now walk around freely and the camera will track your movements anywhere. If you're into robotics, you can now reliably determine your robot's position, orientation, and velocity and make it navigate autonomously to the coordinates of your choice on a map. You can access 6DOF motion tracking data through the ZED SDK or its plugins: Unity, ROS…

Spatial mapping is the ability to capture a digital model of a scene or an object in the physical world. By merging the real world with the virtual world, it is possible to create convincing mixed reality experiences or robots that understand their environment. The ZED continuously scans its environment to reconstruct a 3D map of the real-world. It refines its understanding of the world by combining new depth and position data over time. Spatial mapping is available either through the ZEDfu application or the ZED SDK.

Main specifications of ZED camera are as under:-

- It has a dual camera of 4MP each.
- It can capture 1080p HD video at 30 FPS.
- It has a field of view of 90° (H) x 60° (V) x 110° (D) max.
- It has a stereo baseline of 120 mm.
- It has a depth range of 0.5 - 20 m (1.64 to 65 ft) and depth format is 32 bit.
- It has Electronic Synchronized Rolling Shutter.

- It has the technology of Real-time depth-based visual odometry and SLAM.
- SDK System Requirements
  - Dual-core 2,3GHz or faster processor
  - 4 GB RAM or more
  - Nvidia GPU with compute capability > 3.0

Figure 4.3 ZED Stereo Camera

4.3 Configuration of Hardware and Software for ORB 2 SLAM Application

- Hardware
  - Intel (R) Core TM i5-4210 CPU @ 1.70GHz 2.40 GHz
  - RAM 4 GB
  - 64 Bit Operating System
  - NVIDIA GEFORCE graphic card
  - ZED Stereo Camera
- Software
  - Ubuntu 12.04, 14.04 and 16.04

- Pre-requisites
  - C ++11 or C++0x Compiler
  - Pangolin
  - OpenCV
  - Eigen3
  - DBOW and g2o
- ZED camera ROS wrapper
- ORB SLAM2

4.4 Implemntation

ORB SLAM 2 algorithm was implemented with Ubuntu as under:-

- In first case, ORB SLAM 2 was implemented with commercially available datasets i.e KITTI dataset and EuRoC. Examples used in this regard were of stereo camera.
- ORB SLAM 2 was tested in ROS envionment by running stereo node.
- In order to proces own sequences, ZED camera was installed. ZED installation guide is at Appendix A and B.
- ORB SLAM 2 was integrated with ZED camera in order to implement SLAM algorithm with state of the art stereo camera.

4.5 Discussion

Implementation of ORB SLAM 2 with ZED camera wass carried out successfully in static environment; however, issues were observed when implementing in dynamic environment. Observations with regards to implentation of ORB SLAM 2 with ZED camera in stereo mode are listed below:-

- ZED camera failed in calculating depth of points more than 3m, however, settings was changed to not pick points more than 3 m while calculating depth.

- ZED camera was not able to pick some features of locations having shadows. Efforts were made to capture the scene with varying light conditions without any significant improvement.
- ORB SLAM 2 reading from text format vocabulary (also containing invalid data) slows the process thus becomes time consuming. Text format vocabulary may be converted to binary format.
- Each time ORB SLAM 2 is launched, there is a lengthy process due to its inability to save and load map. Methodology is required to be devised for saving and loading map.
- Offline visualization of maps and mapping trajectories is not provided in ORB SLAM 2 which needs to be provided in its improved version.

CHAPTER 5: CONCLUSION AND FUTURE WORK

This work focused on selection of SLAM algorithm for mobile security platform employment in an unknown indoor environment. SLAM algorithms has changed and improved since its initial work and focus in recent past has shifted to visual SLAM algorithms. Detailed review of different SLAM algorithms was carried out in this thesis. Furthermore comparative studies of vSLAM algorithms of last five years was also studies and analyzed to select a robust SLAM algorithm. Similarly sensors were analyzed keeping in view the indoor application of our thesis work. Selection of ORB SLAM 2 algorithm and ZED stereo camera was the outcome of the study.

Efforts were made for practical implementation of our thesis work. SLAM was successfully implemented in static environment; however certain issues were observed with regards to ZED stereo camera hardware during implementation phase. Further experimentation with ZED stereo camera will elaborate its hardware related issues in detail. Similarly issues with regards to ORB SLAM 2 were observed which can be addressed in future work. Further work in this regard should be change of text format vocabulary to binary based vocabulary, saving/ loading of the map already made using algorithm, and offline visualization of the maps. ORB SLAM 2 with ZED stereo camera lacks robustness in dynamic environment.

While concluding, it is pertinent to mention that this work is a baseline for selecting a SLAM algorithm to be used in mobile security platform. Certain improvements in the algorithm selected will help in its effective utilization for the proposed work.

APPENDIX A

ZED Installation Guide

On Linux, download the ZED SDK for Linux and launch the .run file from a terminal.

chmod +x ZED_SDK_Linux_*.run

./ZED_SDK_Linux_*.run

After the installation, download CUDA 9 or CUDA 10 (depending on the selected SDK installer) from NVIDIA website and install it on your system.

Restart you computer to complete the installation.

Recommended specifications of the operating system are as under:-

	Minimum	**Recommended**	**Embedded**
Processor	Dual-core 2,3GHz	Quad-core 2,7GHz or faster	Jetson TX1, TX2, Xavier
RAM	4GB	8GB	8GB
Graphics Card	NVIDIA GPU*	GTX1060 or higher	TX1, TX2, Xavier
USB port	USB 3.0	USB 3.0	USB 3.0
Operating System	**Windows 7, 8.1, 10, Ubuntu 16.04, 18.04**		L4T

On Linux, compiling an application with the ZED SDK requires a toolchain with GCC (5, 6) and CMake (3.5.0 minimum). To install both, type:

sudo apt-get install build-essential cmake

APPENDIX B

Getting Started with ROS

The ZED ROS wrapper lets you use the ZED stereo cameras with ROS. It provides access to the following data:

- Left and right rectified/unrectified images
- Depth map
- Colored 3D point cloud
- Visual odometry: Position and orientation of the camera
- Pose tracking: Position and orientation of the camera fixed and fused with IMU data (ZED-M only)

Installation

- **Pre-requisites**
  - Ubuntu 16.04
  - ZED SDK and its CUDA dependency
  - ROS Kinetic

Note: if you are using a ZED-M and want to visualize the IMU information using RVIZ, you will also need to install the RVIZ IMU plugin:

$ sudo apt install ros-kinetic-rviz-imu-plugin

- **Build the Package**

  zed_ros_wrapper is a catkin package. It depends on the following ROS packages:

  - tf2_ros
  - tf2_geometry_msgs
  - nav_msgs
  - roscpp
  - rosconsole
  - sensor_msgs

- stereo_msgs
- image_transport
- dynamic_reconfigure
- nodelet
- diagnostic_updater
- urdf
- message_generation
- roslint
- robot_state_publisher
- message_runtime

To install **zed_ros_wrapper**, open a bash terminal, clone the package from Github and build it:

$ cd ~/catkin_ws/src/ #use your current catkin folder

$ git clone https://github.com/stereolabs/zed-ros-wrapper.git

$ cd ..

$ catkin_make -DCMAKE_BUILD_TYPE=Release

$ echo source $(pwd)/devel/setup.bash >> ~/.bashrc

$ source ~/.bashrc

- **Starting the ZED node**

The ZED is available in ROS as a node that publishes its data to topics.

Open a terminal and use roslaunch to start the ZED node:

$ roslaunch zed_wrapper zed.launch

If you are using a ZED-M camera:

$ roslaunch zed_wrapper zedm.launch

- **Displaying ZED data**
  - **Using RVIZ**

RVIZ is a useful visualization tool in ROS. Using RVIZ, you can visualize the ZED left and right images, depth, point cloud, and 3D trajectory.

Launch the ZED wrapper along with RVIZ using the following command:

$ roslaunch zed_display_rviz display_zed.launch

If you are using a ZED-M camera, you can visualize additional information about IMU data using the following command:

$ roslaunch zed_display_rviz display_zedm.launch

- **Dispalying Images**

The ZED node publishes both original and stereo rectified (aligned) left and right images. In RVIZ, select a topic you and use the image preview mode. Here the list of the available image topics:

/zed/zed_node/rgb/image_rect_color: Color rectified image (left sensor by default)

/zed/zed_node/rgb/camera_info: Color camera calibration data

/zed/zed_node/rgb_raw/image_raw_color: Color unrectified image (left sensor by default)

/zed/zed_node/rgb_raw/camera_info: Unrectified color camera calibration data

/zed/zed_node/right/image_rect_color: Right camera rectified image

/zed/zed_node/right/camera_info: Right sensor calibration data

/zed/zed_node/right_raw/image_raw_color: Right camera unrectified image

/zed/zed_node/right_raw/camera_info: Unrectified right sensor calibration data

/zed/zed_node/confidence/confidence_image: Confidence map as image

- **Displaying Depth**

The depth map can be displayed in RVIZ with the following topic:

/zed/zed_node/depth/depth_registered: 32-bit depth values in meters. RVIZ will normalize the depth map on 8-bit and display it as a grayscale depth image.

- **Dispalying Disparity**

The Disparity Image is available by subscribing to the **/zed/zed_node/disparity/disparity_image** topics.

Launch the Disparity Viewer to visualize it:

$ rosrun image_view disparity_view image:=/zed/zed_node/disparity/disparity_image

- **Dispalying The Point Cloud**

A 3D colored point cloud can be displayed in RVIZ with the **zed/point_cloud/cloud_registered** topic.

Add it in RVIZ with **point_cloud** -> **cloud** -> **PointCloud2**. Note that displaying point clouds slows down RVIZ, so open a new instance if you want to display other topics.

- **Dispalying Position and Path**

The ZED position and orientation in space over time is published to the following topics:

/zed/zed_node/odom: Odometry pose referred to odometry frame (only visual odometry is applied for ZED, visual-inertial for ZED-M)

/zed/zed_node/pose: Camera pose referred to Map frame (complete data fusion algorithm is applied)

/zed/zed_node/pose_with_covariance: Camera pose referred to Map frame with covariance

/zed/zed_node/path_odom: The sequence of camera odometry poses in Map frame

/zed/zed_node/path_map: The sequence of camera poses in Map frame

Launching with Recorded SVO Video

With the ZED, you can record and play back stereo video using the .svo file format. To record a sequence, open the ZED Explorer app and click on the **REC** button.

To launch the ROS wrapper with an SVO file, set an **svo_file** path launch parameter in the command line when starting the package:

ZED:

roslaunch zed_wrapper zed.launch svo_file:=/path/to/file.svo

ZED-M:

roslaunch zed_wrapper zedm.launch svo_file:=/path/to/file.svo

Dynamic Reconfigure

You can dynamically change many configuration parameters during the execution of the ZED node:

- **confidence_threshold**: Sets a threshold that filters the values of the depth or the point cloud. With a *confidence threshold* set to 100, all depth values will be written in the depth and the point cloud. This is set to 80 by default, which removes the least accurate values.
- **auto_exposure**: Enables/disables automatic gain and exposure
- **exposure**: Sets camera exposure only if *auto_exposure* is false
- **gain**: Set camera gain only if *auto_exposure* is false
- **mat_resize_factor**: Sets the scale factor of the output images and depth map. Note that the camera will acquire data at the dimension set by the *resolution* parameter; images are resized before being sent to the user
- **max_depth**: Sets the maximum depth range

You can set the parameters using the command *dynparam set*, e.g.:

$ rosrun dynamic_reconfigure dynparam set /zed/zed_node confidence 80

…or you can use the GUI provided by the rqt stack:

$ rosrun rqt_reconfigure rqt_reconfigure

The ZED Node

To start a ZED ROS node you can use the command line

$ roslaunch zed_wrapped zed.launch

or

$ roslaunch zed_wrapped zedm.launch

if you own a ZED-M camera.

The ZED node publishes data to the following topics:

- **Left camera**

/zed/zed_node/rgb/image_rect_color: Color rectified image (left RGB image by default)

/zed/zed_node/rgb_raw/image_raw_color: Color unrectified image (left RGB image by default)

/zed/zed_node/rgb/camera_info: Color camera calibration data

/zed/zed_node/rgb_raw/camera_info: Color unrectified camera calibration data

/zed/zed_node/left/image_rect_color: Left camera color rectified image

/zed/zed_node/left_raw/image_raw_color: Left camera color unrectified image

/zed/zed_node/left/camera_info: Left camera calibration data

/zed/zed_node/left_raw/camera_info: Left unrectified camera calibration data

- **Right camera**

/zed/zed_node/right/image_rect_color: Color rectified right image

/zed/zed_node/right_raw/image_raw_color: Color unrectified right image

/zed/zed_node/right/camera_info: Right camera calibration data

/zed/zed_node/right_raw/camera_info: Right unrectified camera calibration data

- **Stereo Pair**

/zed/zed_node/stereo/image_rect_color: stereo rectified pair images side-by-side

/zed/zed_node/stereo_raw/image_raw_color: stereo unrectified pair images side-by-side

Note: to retrieve the camera parameters you can subscribe to the topics /zed/zed_node/left/camera_info, /zed/zed_node/right/camera_info, /zed/zed_node/left_raw/camera_infoand/zed/zed_node/right_raw/camera_info`

- **Depth and Point Cloud**

/zed/zed_node/depth/depth_registered: Depth map image registered on left image (32-bit float in meters by default)

/zed/zed_node/depth/camera_info: Depth camera calibration data

/zed/zed_node/point_cloud/cloud_registered: Registered color point cloud

/zed/zed_node/confidence/confidence_image: Confidence image

/zed/zed_node/confidence/confidence_map: Confidence image (floating point values)

/zed/zed_node/disparity/disparity_image: Disparity image

- **Tracking**

/zed/zed_node/odom: Absolute 3D position and orientation relative to the Odometry frame (pure visual odometry for ZED, visual-inertial for ZED-M)

/zed/zed_node/pose: Absolute 3D position and orientation relative to the Map frame (Sensor Fusion algorithm + SLAM)

/zed/zed_node/pose_with_covariance: Camera pose referred to Map frame with covariance

/zed/zed_node/path_odom: Sequence of camera odometry poses in Map frame

/zed/zed_node/path_map: Sequence of camera poses in Map frame

- **Mapping**

/zed/zed_node/point_cloud/fused_cloud_registered: Fused color point cloud. Note: published only if mapping is enabled, see mapping/mapping_enabled parameter

- **Inertial Data**

/zed/zed_node/imu/data: Accelerometer, gyroscope, and orientation data in Earth frame

/zed/zed_node/imu/data_raw: Accelerometer and gyroscope data in Earth frame

- **REFERENCES**

[1] S. Thrun, W. Burgard, and D. Fox, "Probabilistic Robotics," Kharagpur, Sri Lanka, 2005.

[2] G. Dissanayake, S.B. Williams, H. Durrant-Whyte, and T. Bailey, "Map Management for Efficient Simultaneous Localization and Mapping (SLAM)", Journal of Autonomous Robots, Vol. 12, no. 3, pp. 267-86, 2002.

[3] S. Thrun, and M. Montemerlo, "The GraphSLAM algorithm with applications to large-scale mapping of urban structures", International Journal on Robotics Research, Vol. 25, no. 5, pp. 403-29, 2006.

[4] J. Folkesson, and H. Christensen, "Closing the Loop With Graphical SLAM", IEEE Transactions on Robotics, Vol. 23, no. 4, pp. 731-41, 2007.

[5] R.M. Eustice, H. Singh, and J.J. Leonard, "Exactly Sparse Delayed-State Filters for View-Based SLAM", IEEE Transactions on Robotics, Vol. 22, no. 6, pp. 1100-14, 2006.

[6] G. Grisetti, C. Stachniss, and W. Burgard, "Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters", IEEE Transactions on Robotics, Vol. 23, no. 1, pp. 34-46, 2007.

[7] K.L. Ho, and P. Newman, "Loop closure detection in SLAM by combining visual and spatial appearance", Robotics and Autonomous Systems, Vol. 54, no. 9, pp. 740-9, 2006.

[8] T. Bailey and H. F. Durrant-Whyte. Simultaneous Localisation and Mapping (SLAM): Part II. Robotics and Autonomous Systems (RAS), 13(3):108–117, 2006.

[9] H. F. Durrant-Whyte and T. Bailey. Simultaneous Localisation and Mapping (SLAM): Part I. IEEE Robotics and Automation Magazine, 13(2):99–110, 2006.

[10] G. Dissanayake, S. Huang, Z. Wang, and R. Ranasinghe. A review of recent evelopments in Simultaneous Localization and Mapping. In International Conference on Industrial and Information Systems, pages 477–482. IEEE, 2011.

[11] J. Aulinas, Y. Petillot, J. Salvi, and X. Llad´o. The SLAM Problem: A Survey. In Proceedings of the International Conference of the Catalan Association for Artificial Intelligence, pages 363–371. IOS Press, 2008.

[12] G. Grisetti, R. K¨ummerle, C. Stachniss, and W. Burgard. A Tutorial on Graph-based SLAM. IEEE Intelligent Transportation Systems Magazine, 2(4):31–43, 2010.

[13] F. Fraundorfer and D. Scaramuzza. Visual Odometry. Part II: Matching, Robustness, Optimization, and Applications. IEEE Robotics and Automation Magazine, 19(2):78–90, 2012.

[14] D. Scaramuzza and F. Fraundorfer. Visual Odometry [Tutorial]. Part I: The First 30 Years and Fundamentals. IEEE Robotics and Automation Magazine, 18(4):80–92, 2011.

[15] S. Saeedi, M. Trentini, M. Seto, and H. Li. Multiple-Robot Simultaneous Localization and Mapping: A Review. Journal of Field Robotics (JFR), 33(1):3–46, 2016.

[16] S. Lowry, N. S¨underhauf, P. Newman, J. J. Leonard, D. Cox, P. Corke, and M. J. Milford. Visual Place Recognition: A Survey. IEEE Transactions on Robotics (TRO), 32(1):1–19, 2016.

[17] C. Stachniss, S. Thrun, and J. J. Leonard. Simultaneous Localization and Mapping. In B. Siciliano and O. Khatib, editors, Springer Handbook of Robotics, chapter 46, pages 1153–1176. Springer, 2nd edition, 2016.

[18] S. Huang and G. Dissanayake. A Critique of Current Developments in Simultaneous Localization and Mapping. International Journal of Advanced Robotic Systems, 13(5):1–13, 2016.

[19] Jorge Fuentes-Pacheco · José Ruiz-Ascencio Juan Manuel Rendón-Mancha, "Visual simultaneous localization and mapping: a survey" 2012.

[20] Arthur Huletski ; Dmitriy Kartashov ; Kirill Krinkin, "Evaluation of the Modern Visual SLAM Methods" , November 2015.

[21] J. Sturm., N. Engelhard, F. Endres, W. Burgard, D.Cremers, "A Benchmark for the Evaluation of RGB-D SLAM Systems", Proc of the International Conference on Intelligent Robot System (IROS), 2012.

[22] Alexander Buyval, Ilya Afanasyev, Evgeni Magid, Comparative Analysis of ROS-based Monocular SLAM Methods for Indoor Navigation, November 2016

[23] R. Mur-Artal, J.M.M. Montiel and J.D. Tardos "ORB-SLAM: A Versatile and Accurate Monocular SLAM System", In: IEEE Trans. on Robotics, 31(5), 1147-1163, (2015)

[24] M. Pizzoli, C. Forster and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time", In: IEEE Int. Conf. on Robotics and Automation, 2609-2616, (2014)

[25] J. Engel, "LSD-SLAM: Large-Scale Direct Monocular SLAM", In: 13th European Conf. on Computer Vision, Zurich, Switzerland, 834-849, (2014)

[26] A. Concha and J. Civera, "DPPTAM: Dense Piecewise Planar Tracking and Mapping from a Monocular Sequence", In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 5686-5693, (2015)

[27] Ilmir Z. Ibragimov, Ilya M. Afanasyev, “Comparison of ROS based Visual SLAM Methods in Homogeneous Indoor Environment”, October 2017

[28] Maksim Filipenko, Ilya Afanasyev, “Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment”, September 2018

[29] Ra´ul Mur-Artal*, J. M. M. Montiel, Member, IEEE, and Juan D. Tard´os, “ORB-SLAM: a Versatile and Accurate Monocular SLAM System”, 2015

[30] Ra´ul Mur-Artal and Juan D. Tard´os, “ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras”, October 2016

Completion Certificate

It is certified that the thesis titled **"Algorithm for Mobile Security Platform Employment in an Unknown Indoor Environment"** submitted by registration no. 00000148441, AO Abid Ali of MS-86 Mechatronics Engineering is completed in all respects as per the requirements of Main Office, NUST (Exam branch).

Supervisor: _____________

Dr. Waqar Shahid

Date: ____ Jul, 2019