Universidad Politécnica de Madrid
Escuela Técnica Superior De Ingenieros Industriales

**Positioning and Mapping for Aerial Robots using
on-board Perception for Autonomous Missions**

Ph.D. Thesis

Hriday Bavle
M.Sc. in Aerospace Engineering

2019

Escuela Técnica Superior De Ingenieros Industriales
Universidad Politécnica de Madrid

Positioning and Mapping for Aerial Robots using on-board Perception for Autonomous Missions

A thesis submitted for the degree of Doctor of Philosophy in
Automation and Robotics

Author:

Hriday Bavle

M.Sc. in Aerospace Engineering

Director:

Pascual Campoy Cervera

Prof. Dr. in Automation and Robotics

Co-Director:

Paloma de La Puente

Prof. Dr. in Automation and Robotics

Titulo:
**Positioning and Mapping for Aerial Robots using
on-board Perception for Autonomous Missions**

Autor:
Hriday Bavle
M.Sc. in Aerospace Engineering

Tribunal nombrado por el Mgco. y Excmo. Sr Rector de la Universidad Politécnica de Madrid el día.....de.....de 2019

Tribunal

Presidente :

Vocal :

Vocal :

Vocal :

Secretario :

Suplente :

Suplente :

Realizado el acto de lectura y defensa de la tesis el día de de ...
calificación de la Tesis.....

El Presidente:

Los Vocales:

El Secretario:

This Thesis is dedicated to,

My Father Milind, for teaching me the value of education.

My Mother Minnie, for teaching me the value of organization.

My Sister Hruti, for her endless love and support.

My love Agnieszka, for loving me and believing in me.

Hriday Bavle

Acknowledgements

I would sincerely like to express my gratitude towards all the people who have helped and supported me all along during the execution of this thesis.

First and foremost I would like to thank Prof. Pascual Campoy for giving me this opportunity to research under his guidance in the Computer Vision and Aerial Robotics (CVAR) group, of the Universidad Politécnica de Madrid. Thanks to his open nature towards research, he gave me the opportunity to research in my field of interest, allowing me to explore and learn a lot in the field of robotics. Thanks to his guidance and support in my research I have been able to achieve several publications in well known journals and conferences. He gave me many opportunities to participate in interesting research projects to explore the real world challenges. Last but not the least, thanks to his dedication towards the group, he has been able to provide me funding during the entire evolution of my thesis.

I would like to thank my advisor Prof. Paloma de la Puente, for listening to me and guiding me throughout the execution of my thesis. Her innovative ideas and support have made possible the development of many of the proposed algorithms in this thesis, as well as execution of the autonomous flights.

This thesis would not be complete without the support of the entire CVAR group all along during this thesis. Dr. Jose Luis Sanchez-Lopez, who's constant innovative ideas and spirit of never giving up, improved the overall outcome of this thesis. Carlos Sampedro, who's spirit of always achieving the best, motivated me to give my best in all the research projects. Alejandro Rodriguez-Ramos, who's great qualities of organization and team work, made us achieve the best results in some of the most difficult projects and competitions. Adrian Carrio, who's open nature and continued support in all the research projects, made the execution of all the projects easy and fun. Ramon Suarez-Fernandez, who gave me his unconditional support and ideas in all the research projects. Zorana Milosevic, for being there all along during this thesis and giving her amazing friendly support. Liang Lu, who added a lot more fun into this thesis and was always there to support me. Francisco Javier Rodriguez, for his openness to new ideas and energetic support in all the research projects. Adrian Alvarez, for his constant disposition and making all the research projects more fun and interesting. Prof. Martin Molina, for all his ideas and help in the creation and improvement of the Aerostack framework, which made all the autonomous flights possible. Prof. Sergio Dominguez for his patience and support during this time. Stephan Manthe, Miguel Fernandez, Ignacio Gil Moreno, Alberto Camporredondo and all the other graduate and under-graduate who researched in the group, during this thesis and helped in many aspects towards the achievement of this thesis. I would also like to thank all the other past members of the CVAR group, Dr. Jesus Pestana, Dr. Aneesh Chauhan, Prof. Carol Martinez, Dr. Changhong Fu, Dr. Jean-Francois Collumeau for their support during this thesis.

I would like to give my special thanks to all the professors and staff of the Centre

for Automation and Robotic (CAR). Special thanks to Prof. Manuel Ferre, Prof. Roque Saltaren, Prof. Fernando Matia for their patience and collaboration. Juan Cely, for this help and support. Angel Luis Martinez for all his help, innovation and professionalism in the construction of all the hardware components. Special thanks to Carlos Sanchez Calderon, Rosa Ortiz and Paloma Espin for their amazing support during this thesis and for taking care of all the bureaucracy work so that I could focus on the research.

I would also like to thank the Aerospace Controls Lab, in the Massachusetts Institute of Technology for my research stay. Special thanks to Prof. Jonathan How for providing me with the opportunity to work in his group. Thanks to all the group members of ACL for making the research stay more productive and fun. Thanks to Gayle Sherman and Britton Bradley for their administrative help and support during this research stay.

Lastly I would like to thanks my family for their support during the thesis. I am specially grateful to my father Milind Bavle, because of his strong will power I was able to peruse and finish this PhD thesis. Thanks to my mother Minnie Bavle and my sister Hruti Bavle for their continued love and support. To my girlfriend Agnieszka Madejowska and her mother Barbara Madejowska for being their as my family and giving me all their love and support.

Hriday Bavle

Resumen

Los robots aéreos autónomos han ganado una gran visibilidad debido a su capacidad de lograr misiones más rápida, así como para alcanzar altitudes más altas sin requerir estructuras adicionales. Aunque, limitaciones en los ambientes interiores, como la falta de posicionamiento preciso y errores en la percepción del entorno debido a sus condiciones dinámicos puede provocar una finalización incompleta de la misión. Para abordar algunos de estos inconvenientes, esta tesis se enfoca en resolver el posicionamiento en interiores y el mapeo de robots aéreos, así como probar y validar sus misiones autónomas en varios escenarios complejos de la vida real.

La primera parte de la tesis consta de cuatro módulos desarrollados para lograr una localización rápida y robusta a bordo los robots aéreos. El primer módulo presenta un estimador de altura basado en EKF de código abierto, capaz de fusionar de forma modular numerosos sensores individuales de estimación de altura junto con su sesgo, para estimar la altura real del robot aéreo, así como estimar la altura de los objetos terrestres en presencia de superficies irregulares. En el segundo módulo, se presenta un algoritmo de odometría semántica visual de código abierto basado en un filtro de partículas, que es capaz de estimar la pose del robot aéreo, combinando información semántica y de odometría visual, para superar los inconvenientes de la acumulación de errores de los algoritmos de odometría visual en presencia de características insuficientes o patrones repetitivos en los entornos interiores. El tercer módulo de esta tesis se centra en un algoritmo de código abierto, rápido y robusto de SLAM geométrico, segmentando y mapeando superficies planas horizontales usando sensores de nubes de puntos y estimando la altura del robot aéreo en presencia de obstáculos estáticos y dinámicos, superando varias limitaciones que se enfrentan al usar sensores de estimación de altura convencionales. El cuarto módulo de esta tesis se centra en un algoritmo de SLAM semántico visual basado en methodo de grafos para robots aéreos, que crea un mapa semántico escaso del entorno utilizando las superficies planas extraídas de los objetos semánticos y pueden realizar cierres de bucle robustos utilizando esta información semántica de alto nivel, superando errores de cierre de bucle debido a características de bajo nivel o requisito de altos recursos computacionales.

La segunda parte de la tesis presenta la extensión de nuestro marco de software de código abierto Aerostack para robots aéreos, integrando y validando los componentes presentados en la primera parte junto con los otros componentes desarrollados para lograr vuelos autónomos de robots aéreos en varios escenarios de la vida real. El módulo uno presenta la investigación sobre la inspección autónoma de una caldera térmica usando el Aerostack, así como los resultados de la misión de inspección de vuelo real dentro de la caldera. El módulo dos presenta la investigación y los resultados obtenidos usando el Aerostack, para inspecciones visuales autónomas de un avión dentro de un hangar de aviones. El módulo tres presenta la investigación y los resultados obtenidos utilizando el Aerostack a bordo de una plataforma robótica aérea personalizada, para la ejecución autónoma de la misión de la competición internacional de vehículos micro-aéreos.

Abstract

Autonomous aerial robots have gained high visibility due to their ability to achieve faster mission completion as well as reaching higher altitudes without requiring additional structures. Although, limitations in the indoor environments such as the lack of accurate positioning and errors in the perception of the environment due to its dynamically changing conditions can result in mission failures or incomplete mission completion. In order to address some these drawbacks, this thesis focuses towards solving indoor positioning and mapping for aerial robots as well as testing and validating their complete autonomous missions in several real-life challenging scenarios.

The first part of this thesis consists of four modules developed for achieving fast and robust localization on-board the aerial robots. The first module presents an open-source, fast and robust EKF based flight altitude estimator for the aerial robots, able to modularly fuse numerous individual altitude estimation sensors along with their bias, for estimating the true flight altitude of the aerial robot as well as estimating the altitude of the ground objects in presence of uneven ground surfaces. In the second module, an open-source, robust particle filter based visual semantic odometry algorithm is presented, which is capable of estimating the drift free pose of the aerial robot combining semantic and visual odometry information in loosely coupled fashion, in order to overcome the drawbacks of error accumulation of the visual odometry algorithms in presence of insufficient characteristic features or repetitive patterns. The third module of this thesis focuses on an open-source, fast and robust geometric SLAM, segmenting and mapping horizontal planar surfaces using point cloud sensors and estimating the true flight altitude of the aerial robot in presence of static as well as dynamic obstacles, overcoming several limitations faced when using conventional altitude estimation sensors. The fourth module of this thesis focuses on an open-source, robust and lightweight graph based visual semantic SLAM framework for aerial robots, which creates a sparse semantic map of the environment using the planar surfaces extracted from the semantic objects and is able to perform robust loop closures using this high-level semantic information, overcoming loop closure errors due to low-level features or requirement of high computational resources.

Second part of the thesis presents the extension of our open-source Aerostack software framework for aerial robots, integrating and validating the components presented in part one along with the other developed components in order to achieve complete autonomous flights of heterogeneous aerial robots in several real-life scenarios. Module one presents the research for the autonomous inspection of a thermal power plant using the Aerostack framework as well as the real flight inspection mission inside the power plant. Module two presents the research and obtained results using the Aerostack framework, for autonomous visual inspections of an airplane inside a real airplane hanger. Module three presents the research and obtained results using the Aerostack framework on-board a custom aerial robotic platform, for the accomplishment the challenging autonomous mission of the International Micro-Aerial Vehicles (IMAV) competition.

Acronyms

The following acronyms are the most widely used acronyms during this thesis:

CVAR: Computer Vision and Aerial Robotics Group

UPM: Universidad Politecnica de Madrid

CAR: Centre of Automation and Robotics

KF: Kalman Filter

EKF: Extended Kalman Filter

UKF: Unscented Kalman Filter

MCL: Monte-Carlo Localization

MSF: Multi-Sensor Fusion

PF: Particle Filter

SLAM: Simultaneous Localization and Mapping

V-SLAM: Visual-Simultaneous Localization and Mapping

VO: Visual Odometry

VIO: Visual Inertial Odometry

VSO: Visual Semantic Odometry

ROVIO: RObust Visual Inertial Odometry

VINS: Visual Inertial Navigation System

GPS: Global Positioning System

IMU: Inertial Measurement Unit

LIDAR: LIght Detection And Ranging

FOV: Field Of View

RGB: Red Green Blue

RGB-D Red Green Blue-Depth

PnP: Perspective-n-Point

KLT: Kanade-Lucas-Tomasi feature tracker

RANSAC: RANdom SAmple Consensus

CPU: Central Processing Unit

GPU: Graphical Processing Unit

FPS: Frames Per Second

DOF: Degrees Of Freedom

FAST: Features from Accelerated Segment Test

PID: Proportional-Integral-Derivative

RMSE: Root Mean Square Error

ATE: Absolute Trajectory Error

XML: Extensible Markup Language

GUI: Graphical User Interface

IMAV: International Mirco-Aerial Vehicle competition

SITL: Software In The Loop

IARC: International Aerial Robotics Competition

Contents

Acknowledgements	VII
Resumen	IX
Abstract	XI
Acronyms	XIII
Contents	XVIII
List of Figures	XXII
List of Tables	XXIII
 I INTRODUCTION	 1
1 Introduction	3
1.1 Introduction to Aerial Robotics	3
1.2 Problem Statement	6
1.2.1 Pose Estimation	6
1.2.2 Simultaneous Localization and Mapping	7
1.2.3 Autonomous Missions	7
1.3 Thesis Objectives and Contributions	8
1.4 Methodology	10
1.5 Thesis Outline	11
 2 State of the Art	 13
2.1 Pose Estimation	13
2.1.1 Flight Altitude Estimation	13
2.1.2 Visual Pose Estimation	15
2.2 Visual SLAM	19
2.2.1 Geometric SLAM	20
2.2.2 Geometric and Semantic SLAM	23
2.3 Autonomous Flights	24
2.3.1 Industrial Applications	24

2.3.2	International Competitions	25
II	POSE ESTIMATION	29
3	Altitude Estimation using Conventional Sensors	31
3.1	Introduction	31
3.2	Proposed Multi-Sensor Fusion State Estimator	32
3.2.1	State and process model	33
3.2.2	Prediction Stage	34
3.2.3	Measurements and measurement model	34
3.2.4	Update Stage	34
3.3	Components of the Proposed State Estimator	35
3.3.1	Aerial Robot	35
3.3.2	Ground object	36
3.3.3	IMU sensor: accelerometer, gyro and magnetometer	36
3.3.4	Barometer sensor	38
3.3.5	Distance sensor	39
3.4	Experiments and Results	39
3.4.1	Experimental Setup	39
3.4.2	Results	40
3.5	Discussions	42
3.6	Conclusions	44
4	Pose Estimation using Semantic Stereo Visual Odometry	47
4.1	Introduction	47
4.2	Stereo Visual Odometry	48
4.3	Semantic Detection and Segmentation	49
4.3.1	Semantic Detection	49
4.3.2	Semantic Segmentation	49
4.4	Particle Filter based Localization	50
4.4.1	Prediction	50
4.4.2	Update	51
4.5	Experiments and Results	53
4.5.1	System Setup	53
4.5.2	Results	53
4.6	Discussions	56
4.7	Conclusions	57
III	GEOMETRY AND OBJECT BASED SLAM	59
5	Altitude Estimation and Planar Mapping using Point Cloud Sensors	61
5.1	Introduction	61
5.2	3D Planar Clustering and Horizontal Plane Segmentation	62

5.2.1	Normal Based Clustering	63
5.2.2	Horizontal Plane Segmentation	64
5.3	Horizontal Plane-Based Localization and Mapping	65
5.3.1	Process Model	65
5.3.2	Prediction Stage	65
5.3.3	Measurement Model	66
5.3.4	Data Association and Update Stage	66
5.4	Experiments and Results	67
5.4.1	Standard Datasets	67
5.4.2	Real Flights	69
5.5	Discussion	74
5.5.1	Standard Datasets	74
5.5.2	Real Flights	74
5.5.3	Limitations	78
5.6	Conclusions	78
6	Pose and Map Estimation using Visual Semantic SLAM	81
6.1	Introduction	81
6.2	Semantics based Plane Extraction	83
6.2.1	Semantic Object Detection	83
6.2.2	Semantic Object Segmentation	83
6.3	Graph SLAM	85
6.3.1	VO/VIO Odometry	85
6.3.2	Graph Construction	86
6.3.3	Data Association	87
6.3.4	Graph Optimization	88
6.4	Experiments and Results	89
6.4.1	Standard Dataset	89
6.4.2	Field Experiments	91
6.5	Discussions	101
6.5.1	Standard Datasets	101
6.5.2	Field Experiments	101
6.6	Conclusions	104
IV	FIELD TESTS AND VALIDATION	107
7	Validation Framework	109
7.1	Introduction	109
7.2	Aerostack System Architecture	111
7.2.1	Aerostack Components Specifications	111
7.2.2	Multi layered Model Organization	112
7.3	Aerostack Subsystem Architecture	113
7.3.1	Interface Layer	113
7.3.2	Functional Layer	114

7.3.3	Executive Layer	119
7.4	Conclusions	121
8	Field Tests and Applications	123
8.1	Autonomous Thermal Power Plant Inspection	124
8.1.1	Introduction	124
8.1.2	Hardware Architecture	125
8.1.3	Software Architecture	126
8.1.4	Experiments and Results	131
8.1.5	Conclusions	138
8.2	Autonomous Airplane Inspection	139
8.2.1	Introduction	139
8.2.2	Hardware Architecture	139
8.2.3	Software Architecture	141
8.2.4	Experiments and Results	145
8.2.5	Conclusions	150
8.3	International Micro Aerial Vehicles Competition (IMAV) 2016	151
8.3.1	Introduction	151
8.3.2	Hardware Configuration	151
8.3.3	Software Architecture	154
8.3.4	Experiments and Results	159
8.3.5	Conclusions	163
9	Conclusions and Future Work	165
9.1	Conclusions	165
9.2	Future Work	167
V	Appendices	169
A	Scientific Dissemination	171
A.1	Publications	171
A.1.1	Journals	171
A.1.2	Conferences	172
A.2	Industrial Projects and International Competitions	173
A.2.1	Industrial Projects	173
A.2.2	International Competitions and Public Demonstrations	173
A.3	Video Demonstrations	174

List of Figures

Chapter. 2	13
2.1 A layout of the different Visual Pose Estimation Techniques.	16
2.2 A Layout of the different Visual SLAM Techniques.	20
Chapter. 3	31
3.1 The aerial platform used validating the proposed state estimator describing its frame of references.	32
3.2 The inputs and outputs of the proposed state estimator.	33
3.3 Results obtained while performing take-off, navigation and land tasks during the execution of a real flight.	41
3.4 Navigation phase of the same flight represented in Fig. 3.3, when the aerial robot is flying over several obstacles.	42
3.5 Results obtained during the execution of a complete flight, where take-off and land tasks are performed over an elevated ground surface (platform). .	43
Chapter. 4	47
4.1 The mini-aerial robotic platform used to validate the proposed approach, including its relevant frame of references.	48
4.2 Results obtained from autonomous real flight experiments of the aerial robot in a cluttered indoor environment during the optitrack evaluation experiment.	54
4.3 Results obtained from autonomous real flight experiments of the aerial robot in a cluttered indoor environment during the long trajectory evaluation experiment.	55
Chapter. 5	61
5.1 The custom aerial robotic framework used in the real experiments.	62
5.2 Chair as a ground obstacle used in the real flight experiments with all the extracted normals.	63
5.3 Indoors Environment used in IROS 2011 dataset with increasing complexity of ground obstacles.	69
5.4 Comparison of our algorithm with the MoCap ground truth data from the IROS 2011 dataset for slow and medium camera speed scenario.	70

5.5	Comparison of our algorithm with the MoCap ground truth data from the IROS 2011 dataset for high camera speed scenario.	71
5.6	Real flights performed in an indoor environment with different configuration of static as well as dynamic ground obstacles.	72
5.7	Real flight performed in static environment at different velocities of the aerial robot.	75
5.8	Real flight performed in dynamic environment at different velocities of the aerial robot.	76
Chapter. 6		81
6.1	The aerial robotic platform used for validating the presented visual semantic slam approach with the frame of references.	82
6.2	The structure of the graph created, using the VIO and the semantic landmarks.	86
6.3	The trajectory of the camera estimated by the proposed algorithm when comparing it to the ground truth estimated trajectory for the RGB-D TUM dataset.	92
6.4	The freiburg3 long office household sequence presenting the generated 3D point cloud map of the environment with the estimated trajectory.	94
6.5	The results obtained during the execution of the Long Hall experiment. . .	95
6.6	The results obtained during the execution of the Long Corridor experiment. .	96
6.7	The results obtained during the execution of the Repetitive Trajectory experiment in presence of several semantic objects.	97
6.8	The results obtained during the execution of the Random Trajectory experiment in presence of several semantic objects.	98
6.9	The results obtained during the execution of the experiment on-board an aerial robot flying at 2 m/s	99
Chapter. 7		109
7.1	The multi-layered aerostack architecture with all its layers and their corresponding components which can be used for autonomous flights of a single aerial robotic agents or n heterogeneous agents.	111
7.2	General description of the Feature Extraction System, and its relationship with the rest of the components of Aerostack.	114
7.3	General description of the localization and mapping system, and its relationship with the rest of the components of Aerostack.	116
7.4	General description of the Motor System, and its relationship with the rest of the components of Aerostack.	117
7.5	General description of the motion planning system, and its relationship with the rest of the components of Aerostack.	119
7.6	General description of the Executive layer, and its relationship with the rest of the components of Aerostack.	120

Chapter. 8	123
8.1 Images of the exterior as well as the interior view of the thermal power plant.	124
8.2 The DJI Matrice 100 aerial robot used for the thermal power plant inspection with all its components.	125
8.3 The software architecture of the components used during the inspection mission. With the circular boxes representing the main components integrated.	126
8.4 The aerial robot inside the thermal power plant, with all the frame of references.	127
8.5 The 2D map of the thermal power plant created during the real flight experiment.	131
8.6 The aerial robot entering inside the thermal power plant through the entrance of width 1.13 m.	132
8.7 The trajectory of the aerial robot inside the thermal power plant during the simulation experiment.	133
8.8 The real trajectory of the aerial robot inside the thermal power plant, with a true scaled (m) image of the plant.	135
8.9 Images of a section of the burner taken during the execution of the autonomous inspection mission.	136
8.10 The Matrice-210 aerial robotic platform used during the autonomous airplane inspection mission with its on-board sensors.	140
8.11 The Aerostack architecture with the main software components used in order to perform the airplane inspection mission.	141
8.12 The graphical user interface used for generating the user commanded mission points for the aerial robot with respect to an initial point in front the airplane.	142
8.13 The 3D map of the airplane created during the execution of the first inspection mission.	147
8.14 The 3D map of the airplane created during the execution of the second inspection mission.	148
8.15 The indoor mission area of the IMAV 2016 competition, illustrating the dimensions of the operating zone and the mission zone.	152
8.16 The entrance to the IMAV 2016 house using the door and the window. . .	152
8.17 The aerial robot used in the IMAV 2016 competition performing the release of the red item into the red bucket.	153
8.18 Custom aerial robot designed for IMAV 2016 competition with its hardware components.	153
8.19 The Aerostack framework with it software components used during the execution of the IMAV 2016 mission.	155
8.20 The aerial robotic platform with the frame of references.	157
8.21 The performed IMAV 2016 experiment with the orange arrows showing the movement of the aerial robot and the triangles representing the commanded mission points.	160

8.22 The 3D map created during the exploration mission in the IMAV 2016 competition.	161
---	-----

List of Tables

Chapter 4	47
4.1 Absolute Trajectory Error for the compared algorithms.	56
Chapter 5	61
5.1 Individual RMS errors obtained during the execution of IROS2011 kintect dataset experiments.	69
Chapter 6	81
6.1 Absolute Trajectory Error (ATE) m of the compared algorithms for the TUM dataset.	93
Chapter 8	123
8.1 The commanded mission points in m and deg to the aerial robot during the simulated autonomous mission.	134
8.2 The commanded mission points in m and deg to the aerial robot during the autonomous mission.	134
8.3 The commanded mission points in m and deg to the aerial robot during the autonomous mission.	146
8.4 The commanded mission points in <i>m</i> and <i>deg</i> to the aerial robot during the autonomous mission.	149

Part I

INTRODUCTION

Chapter 1

Introduction

1.1 Introduction to Aerial Robotics

Recent years have witnessed an increase in research and development of intelligent and autonomous robotics systems, efficiently achieving high-level complex missions aiding and minimizing the human effort. These robotic systems can be divided into three main branches, namely (1) Aerial Robotics, (2) Ground Robotics and (3) Under-Water Robotics. Multiple on-going solutions are being attempted using aerial robots for several commercial applications due to their capability of faster mission completion. This characteristic allows them to efficiently and rapidly achieve missions in numerous applications. In industrial inspections, aerial robots can perform inspections of tall and complex structures, reducing the time and cost of mounting and employing heavy and complex machinery such as scaffoldings, cranes, etc. In precision agriculture, aerial robots can help increase crop production as large plantations can be efficiently and quickly inspected, which can help track and monitor the growth of the plants at constant and regular time intervals. Surveillance missions can also be completed rapidly from higher altitudes. Aerial robots can also be used in several humanitarian missions such as disaster relief as well as package deliveries as they can potentially accomplish such missions in a very short time interval.

The aerial robotic platforms can be mainly divided into two subtypes; (1) Fixed wing aerial robots (2) Multi-rotor aerial robots. The fixed wing aerial robots can cover larger area in shorter time intervals and have a larger battery life, but they suffer two main limitations; first, they cannot hold position at a particular point (hover) and second, they cannot be flown in constrained indoor environments. These limitations can be solved using the multi-rotor aerial robots, efficiently performing missions flying or holding their positions in constrained indoor environments. Since the main contribution of this thesis is towards achieving autonomous multi-rotor aerial robots in indoor missions, only the multi-rotor aerial robots are discussed further. For convenience, in the rest of content of this thesis multi-aerial robots will be referred as aerial robots.

Modern aerial robots increased in demand in the late 2000s due to the introduction of

lightweight and cheaper sensors such as Inertial Measurement Units (IMU), Global Positioning System (GPS) and the on-board flight controllers. Numerous enterprises, as well as research institutes have provided diverse aerial robotic platforms as well as standalone autopilot solutions. DJI¹ which is currently the market leader in manufacturing aerial robots, has produced distinct platforms such as; (1) DJI Phantom series, (2) DJI Mavic series and (3) DJI Inspire series; which can be mounted with RGB or thermal cameras for professional aerial photography, inspections or similar applications. (4) DJI Matrice 100 platform has an open-source software and is widely used for research purposes, with the possibility of mounting additional payloads for autonomous flights. (5) DJI Matrice 200 series is similar to the Matrice 100 platforms, but with increased efficiency and flight performance, meeting the industrial standards. Parrot² manufactures aerial robots which are widely used for research applications due to their lightweight hardware and open-source software. Parrot AR drone and the Parrot Bebop are some of the noticeable and widely used platforms manufactured by Parrot. Pixhawk³, which originated as a research project in ETH Zurich, is a well known open-source hardware as well as software autopilot system for aerial robots. The open-source nature of the project has gained wide attention and is actively being used by research organizations and companies for several applications.

To achieve manual as well as autonomous flights, aerial robots require a combination of on-board sensors. The sensors used on-board the aerial robots can be divided into the following types:

(1) *Standard Sensors*: These sensors are critical for a stable flight of the aerial robots. The sensors include the Inertial Measurement Unit (IMU) consisting of an accelerometer measuring the linear accelerations and a gyroscope measuring the angular velocities of the aerial robot. The IMU sensor is also used for estimating the attitude of the aerial robot at high speeds, for robust and fast attitude control of the aerial robot. The IMU sensor is the most critical on-board sensor, in case of failure of an IMU sensor, the aerial robot cannot continue its flight. Hence many of the platforms nowadays have additional IMU sensors for redundancy. Magnetometer is another important sensor which is present on-board the aerial robot for estimating its correct heading. Barometer is used for measuring the flight altitude of the aerial robot for stable altitude control of the aerial robot. Since these sensors are small and lightweight they are generally incorporated within the autopilot of an aerial robotic platform and are not required to be mounted using additional structures on-board the aerial robots (unless required for a particular application).

(2) *Advanced Sensors*: Although the standard sensors are required for a basic manual flight of the aerial robot, they cannot be used in a standalone manner for autonomous flight as they alone cannot provide any perception of the environment. To provide intelligence to the aerial robot for them to perceive the environment, several additional sensors can be mounted on-board. RGB based cameras are most commonly used sensors for the perception of the environment as they are lightweight and provide abundant information about the environment in the form of images. RGB-D sensors are also commonly used sensors on-board the aerial robots, as they are lightweight and provide additional infor-

¹<https://store.dji.com/>

²<https://www.parrot.com>

³<http://pixhawk.org/>

mation of the environment as compared to the RGB cameras in the form of depth images. The commonly used RGB-D sensors on-board the aerial robots, are the Intel RealSense family⁴ and Microsoft Kinect⁵, which are lightweight and power-efficient. Laser-based sensors are also used on-board the aerial robots, as they provide high precision measurements. 1D laser sensors such as Lightware⁶ are used for measuring the altitude of the aerial robots with high precision of around 0.01 m. 2D laser sensors such as the Hokuyo sensors⁷ provide high precision measurements along the horizontal axis of the aerial robot capable of measuring maximum distances in the range of 5 m to 80 m. 3D laser sensors in recent years have also gained attention for aerial robotic flights due to the reduction in their size and power consumption. Velodyne⁸ and Ouster⁹ are the leading laser manufacturers providing lightweight 3D laser sensor which can be mounted on-board the aerial robots.

Autonomous flights using the aerial robots also require on-board computers for processing all the on-board sensor measurements and for processing all the algorithms related to perception, localization, control etc. Some of the latest and widely used on-board computers include Intel-NUC¹⁰ mini computers which can be configurable for Intel i3, i5 or i7 processors as well as the on-board ROM and RAM. The Nvidia family¹¹ of mini-computers namely the TX1, TX2 as well the Xavier, which consist of powerful ARM-based processors along with a Graphical Processing Unit (GPU) support. The lightweight Aaeon Upboards¹² consisting of configurable Intel processors but without GPU support. The DJI manifold¹³ mini-computers based on the Nvidia jetson TX1 and TX2 but with hardware modifications made by the DJI, mainly mounted on-board the DJI platforms.

This thesis demonstrates research work towards improving the self-awareness and intelligence of the aerial robots in indoors, presenting positioning and mapping algorithms, using a configuration of different on-board sensors. It also demonstrates the achievements of complete autonomous flights of the aerial robots solving different challenging real-life applications. The main aerial robotic platforms developed and utilized during the evolution of this thesis include; (1) A custom aerial robotic platform based on the Pixhawk autopilot, (2) the Parrot Bebop 2 platforms and the (3) DJI M100 and (4) M210 platforms. All of them weigh within a range of 0.7 kgs to 4 kgs, with a diagonal size range of 0.35 m to 1.2 m and are tested at flight altitudes ranging from 1.5 m to 50 m.

⁴<https://www.intelrealsense.com/>

⁵<https://developer.microsoft.com/en-us/windows/kinect>

⁶<https://lightware.co.za/>

⁷<https://www.hokuyo-aut.jp/search/index.php?cate01=1>

⁸<https://velodynelidar.com/>

⁹<https://www.ouster.io>

¹⁰<https://www.intel.com/content/www/us/en/products/boards-kits/nuc/mini-pcs.html>

¹¹<https://www.nvidia.com>

¹²<https://up-board.org/>

¹³<https://www.dji.com/es/manifold>

1.2 Problem Statement

Perception and localization algorithms are the core algorithms for an autonomous mission as they provide self-awareness to the aerial robot as well as awareness regarding its environment. However, the sensors on-board the aerial robots, as well as the indoors environments present several challenges, for the proper performance of these algorithms. Also to achieve efficient and intelligent autonomous aerial robots for solving real-life applications, requires an efficient framework for integration of the perception and localization algorithms with other multi-disciplinary techniques such as control, navigation, planning, etc. Hence the problem statement of this thesis is divided into the following three parts:

1.2.1 Pose Estimation

Sensors on-board the aerial robot when used individually in complex and unstructured indoor environments can accumulate error in their measurements, due to several factors that can affect the environments, such as changes in the illumination, changes in the 3D structure of the obstacles, etc as the aerial robot performs the flight. These induced errors if not taken into account can cause the aerial robot to perform the mission incorrectly or a complete mission failure. Combining the measurements of the available on-board sensors also called as **Sensor Fusion**, makes it possible to overcome these limitations, to robustly estimate the state of the aerial robot. During an autonomous indoor flight estimating the true **Flight Altitude**, of the aerial robot, is a critical necessity, as a wrong estimation or failure of a sensor measuring the altitude can cause the aerial robot to directly crash to the surface below it. Several sensors can be used indoors for measuring the flight altitude of the aerial robot, namely accelerometers, barometers, and distance sensors. Accelerometer measurements are affected due to vibrations of the aerial robotic flight. Barometer measurements are affected due to the pressure changes caused during an indoor flight as the aerial robot experiences turbulences when flying in tight spaces. Measurements from the distance sensors are affected in the presence of non-flat ground surfaces. Redundancy is thus required when estimating the true flight altitude of the aerial robot.

Autonomous missions performed using smaller sized aerial robots have the advantages of navigating in tight spaces in the indoor environments as well are less harmful in the presence of failures in terms of both hardware and software components. Hence lighter and smaller sized aerial robots are preferred choices for autonomous flights in unknown and unstructured indoor environments. Lightweight sensors such as RGB-D cameras can provide more detailed information of the environment in terms of RGB images along with the depth information. Visual Odometry (VO) or Visual Inertial Odometry (VIO) algorithms based on measurements of the RGB-D sensors are preferred choices to run on-board the aerial robots. Several VO/VIO algorithms presented in the literature use low-level characteristic features like points, lines, etc extracted from these sensors. These VO/VIO estimations are sensitive in the presence of illumination changes, lack of characteristic features in the environment or repetitive patterns. This causes pose estimation of the aerial robot to diverge from its true value. On the other hand, indoor environments are

characterized by several high level features such as chairs, tables, tv-monitors etc known as **Semantic Objects**, which can be used along with the VO/VIO algorithms in order to achieve more robust **Semantic Visual Odometry** algorithm capable of correcting the errors induced in the low-level visual odometry algorithms.

1.2.2 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) algorithms are generally more robust as compared to localization techniques as they create a map of the environment simultaneously localizing the robot in it. However, many SLAM algorithms create a dense map of the environment and require high computational resources, which are not available on-board the aerial robots. Utilizing abundant and concrete information from the environment and creating sparse maps of the indoor environment can reduce the computational complexity of such SLAM based algorithms. Most of the indoor environments have abundant information in terms of planar surfaces. One of the most common planar surface during an aerial robotic mission is the ground plane, below the robot. This ground plane along with other available horizontal planar surfaces can be utilized for **Horizontal Planar SLAM** creating a sparse planar map of the environment and simultaneously estimating the true flight altitude of the aerial robot.

Several sparse or semi-dense SLAM algorithms have been tested on-board the aerial robots, which depend on low-level characteristic features of the environment. The performance of such algorithms can degrade in the presence of changing environmental conditions such as repetitive patterns, bad or changing illuminations, etc. **Semantic Visual SLAM** algorithms can add robustness to the data associations and thus the loop closures, incorporating high-level sparse planar information extracted from the semantic objects, at the same time being computationally favorable for the computers on-board the aerial robots.

1.2.3 Autonomous Missions

To successfully solve real-life applications using autonomous aerial robots, several multidisciplinary works need to be rigorously tested and validated for achieving an intelligent and safe mission execution. The work includes (1) **Hardware Architecture Design**; based on the mission tasks and their criticality, appropriate design of the aerial robotic platform with the suitable combination of the on-board sensors is required. The hardware design also depends on the mission task requirements, characteristics of the environment as well as criticality of the autonomous mission. (2) **Software Architecture Design**; The software modules are selected based on the chosen hardware modules as well as the mission tasks to be performed. The software module design and testing depend on several other factors such as computational restrictions, temporal restrictions as well as the criticality of the mission. Software modules for perception and localization are required to be implemented with fail case scenarios and tested in different challenging environments, as a single wrong estimation by the localization module can induce a catastrophic system failure causing property damage or human injury. Software modules for autonomous

control also need to be tuned and tested for the particular setup of the platform for a smooth and safe flight. Software modules consisting of high-level task execution layers such as mission planners, supervision system, etc which are required to be examined for the successful execution of the autonomous mission. Lastly, the entire developed system in terms of hardware as well as the software components needs to be correctly tested in different indoor environments, ensuring proper working of all the components, and efficient communication between them, before the final mission execution in **Industrial Inspections** or **International Competitions**. For this purpose, efficient architectural solutions are required which can combine and reuse all the software components based on the selected hardware components, minimizing the development time of the software modules as well as facilitating their re-usability and integration.

1.3 Thesis Objectives and Contributions

The main goal of this thesis is to develop autonomous aerial robotic platforms that can perform missions in unknown and unstructured indoor spaces, which can be used for several industrial applications as well as challenging international competitions. To perform autonomous flights, it is very important to first analyze the global objective to be achieved using the aerial robots, based on which the aerial robotic design or platform needs to be considered along with the configuration of the on-board sensors. Each on-board sensor needs to be validated and integrated into the aerial robotic software framework and based on the sensor measurements, positioning and mapping algorithms are developed for the aerial robot to better understand the environment around it. Positioning and mapping algorithms thus form the core functionality of the autonomous aerial robot, over which several other algorithms can be developed such as control, path planning, mission planning, etc. Hence, this thesis focuses on improving the state of the art of positioning and mapping on-board the aerial robots enabling their autonomous flights and validating the real autonomous flights to solve real-life applications. The objectives of this thesis are given below:

- Development and testing of a novel flight altitude estimator for the aerial robots, fusing the measurements of the available on-board sensors, in presence of several unstructured and complex ground obstacles, in indoor scenarios. The flight altitude estimator is considered with the following specifications:
 - Modularity in terms of the number of sensors used. It should be able to fuse measurements from n number of sensors, including accelerometers, barometers and distance sensors.
 - Auto-calibration of the sensors to improve the accuracy, by means of sensor bias estimations.
 - Synchronous prediction stage whenever the estimated state is required by the altitude controller.
 - Asynchronous update stage whenever a measurement is received by the state estimator.

- Development and testing of a novel semantic visual odometry algorithm estimating the 6D pose of the aerial robot, fusing the visual odometry measurements with the semantic detections available in the indoor environment. The main objectives of this algorithm are:
 - Improving the long-term drift present in the odometry based algorithms.
 - Modularity in terms of the visual odometry algorithms used. Several visual odometry algorithms can be integrated and tested.
 - Lightweight in terms of the computational resources.
 - High-frequency estimation of the state of the aerial robot.
 - Accurate estimation of the relative position of complex-shaped semantic objects.
- Development and testing of novel simultaneous localization and mapping algorithm based on horizontal planar surfaces for estimating the flight altitude of the aerial robot. The main objectives of this work are:
 - Utilize RGB-D perception which provides higher quality information instead of conventional laser altimeter, to accurately map planar surfaces.
 - High-frequency planar segmentation and mapping in order to achieve fast and robust flight altitude estimation.
 - Able to provide an accurate estimate even in the presence of dynamic obstacles.
 - Employing low computational resources.
- Development and testing of a novel visual semantic simultaneous localization and mapping framework on-board the aerial robot with the following objectives:
 - Adding robustness in the form of high-level semantic features, to conventional low-level feature-based SLAM solutions.
 - Modularity in terms of the front end visual odometry algorithms used. Several visual odometry algorithms can be integrated with the framework.
 - Accurate estimation of the relative position of complex-shaped semantic objects, using all the available planar surfaces within the semantic object.
 - Modularity in terms of the object detectors used. The framework can integrate several state of the art object-based detectors.
 - Lightweight in terms of processing power, in order to run real-time on-board the aerial robot.
- Extension of our open-source software framework called Aerostack, integrating the above mentioned algorithms along with several other state of the art methods, in order to enable autonomous flights of the aerial robots in indoors.

- Integration, validation and rigorous testing of several aerial robotic platforms with distinct configuration on-board sensors and state of the art algorithms into the Aerostack framework in order safely and intelligently perform autonomous inspection missions in real and challenging industrial environments and international competitions.

1.4 Methodology

The methodology used during the evolution of this thesis follows a top down engineering approach which comprises of moving from high level abstractions and logical designs of the system to a physical implementation of the system and its testing and validation. Following are the main steps adopted:

- *Problem Proposal:* The first step of the entire process consists of a proposal regarding an application to be solved. This proposal can be provided by an enterprise requiring a robotics solution or can be self proposed scientific challenge or a proposal from the robotics community in the form of an international competition.
- *Problem Statement:* Based on the proposal provided, the problem statement is stated in order to analyze all the tasks the aerial robotic system needs to performs. It should also analyze all the requirements the system should satisfy as well as overview its current limitations
- *Literature Review:* This steps comprises of reviewing all the current scientific approaches available for solving the problem in hand. This step helps to analyze the advantages and drawbacks of the current approaches.
- *Development:* Based on the problem statement and the review of the current state of the start techniques, design and development of both the software as well as the hardware components of the aerial robotic system is carried out.
- *Evaluation:* Strong evaluation of the entire system is performed first the simulation stage and the in the real world environments in the form of real flight experiments. Based on the evaluation of the system, the development phase can be re-iterated in order to consider changes in the hardware or the software design. Further evaluation is carried out based on the changes implemented in the design.
- *Field Tests:* Depending on the problem proposal, if the proposal consists an application solving an industry related problem, the field tests are carried out in the final test environment required by the industry. In case of problem proposal by an international competition, the the final field tests are carried out the the environment constructed by the organizers in order to validate all the presented approaches.

1.5 Thesis Outline

The research work performed during the evolution of this thesis is organized into three parts and 10 chapters as follows:

- Part. I consists of two chapters which give a general introduction of the work done in this thesis and the related work as follows:
 - Chapter. 1 introduces the field of aerial robotics, providing the problem statement along with the objectives and the contributions of this thesis.
 - Chapter. 2 gives an in depth review of the current work in the literature of positioning and mapping of aerial robotics and autonomous missions performed using the aerial robots.
- Part. II explains the positioning algorithms developed for the aerial robots and is subdivided into two chapters as follows:
 - Chapter. 3 presents the flight altitude estimator developed fusing conventional altitude estimation sensors, for accurate estimation of the flight altitude of the aerial robots in presence of ground obstacles.
 - Chapter. 4 presents a particle filter based semantic visual odometry algorithm developed for aerial robots.
- Part. III is subdivided into two chapters each presenting the work done on simultaneous localization and mapping algorithms running on-board the aerial robots
 - Chapter. 5 is a horizontal planar SLAM algorithm developed for fast and robust flight altitude estimation of the aerial robots using point cloud sensors.
 - Chapter. 6 presents the visual semantic SLAM algorithm for fast and robust pose estimation of the aerial robot.
- Part. IV describes the results obtained in the field tests performed and is divided into three chapters.
 - Chapter. 7 presents briefly our Aerostack framework and integration of the software components to perform the autonomous flights.
 - Chapter. 8 demonstrates the results obtained using autonomous aerial robots in a real thermal power plant inspection and airplane inspection scenarios. It also demonstrates the results obtained of an autonomous flight in an international aerial robotics competition.
 - Chapter. 9 finally concludes the thesis and also presents the future direction of the research work.

Chapter 2

State of the Art

In the recent years, there has been an active research in the field of aerial robotics in order to solve a final goal i.e to achieve autonomous and intelligent aerial robots. In the literature, numerous researches have been presented in the form of control, localization, planning etc, techniques. This chapter presents the state of the art research in the field of localization and mapping applied to aerial robotics, and subsequently presents the research performed using aerial robots in several industrial applications as well as international competitions.

2.1 Pose Estimation

This section presents the pose estimation algorithms presented in the research community. Sect. 2.1.1 presents the research related to the flight altitude estimation of the aerial robot, which is considered one of the critical parameters of the state of the aerial robot, required for an autonomous flight. Sect. 2.1.2 presents the visual pose estimation algorithms researched in order to estimate the 6 DoF pose of the aerial robot.

2.1.1 Flight Altitude Estimation

Several flight altitude estimators for aerial robots have been presented in the literature. Based on the techniques and sensors used, they can be divided into two subsections as follows:

2.1.1.1 Filtering Based Techniques

This section presents the current research for estimating the flight altitude of the aerial robots using several conventional sensors such as laser altimeters, accelerometers, barometers etc. Grzonka et al., presents an autonomous aerial robotic platform for navigation in indoor environments. They estimate the altitude using a multilevel-SLAM technique for mapping the different levels of the ground objects using the measurements from laser

beams deflected downwards. Using 1D laser beam information pointing downwards can have lot of noise in the measurements when flying on top of complex structured obstacles, resulting in wrong mapping of the ground obstacles. Also the map of the ground obstacles is created depending on the x - y -axis position of the aerial robot and hence altitude estimation is valid only in presence of static ground obstacles and would fail in presence of dynamic ones.

Dryanovski et al., 2013 proposes a simple altitude estimation of a aerial robot deflecting several laser beams downwards from a horizontal 2D laser sensor, creating a histogram of a certain bin size. After detecting a peak of the histogram (a ground obstacle), a bin is created and an average of the readings belonging to that bin is computed, thus estimating the flight altitude and the height of the ground obstacle present. This approach does not consider the noise in the measurements of a laser neither performs a sensor fusion technique fusing the laser measurements with other sensor measurements.

Szafranski et al., 2013 provides a solution for estimating the aerial robot altitude fusing the altitude derived from a pressure sensor with an ultrasound sensor. The authors pre-filter the readings coming from both sensors and use a filter to fuse the sensor measurements, based on their assigned weights. The assigned weight for the ultrasound is higher in lower altitudes due to higher accuracy and less for the pressure sensor and vice a versa. At higher altitudes greater than 2–3 m the ultrasound sensor provides noisy measurements of altitude and the filter mostly depends on the pre-filtered altitude provided by the pressure sensor. The authors do not estimate any bias which might be present in the altitude measured by the pressure sensor and they do not compare the estimated altitude with a ground truth system.

Wang et al., 2014 presents an autonomous aerial robotic navigation solution for partially known indoors environments, using two 2D laser sensors and an IMU. For the estimation of the altitude, authors use the a 2D laser sensor mounted orthogonally with respect to the aerial robot. They filter the laser measurements by line segments, where the line segments are sorted based on the perpendicular distance between the ground plane and the center of the laser sensor. The longest line segment is thus assumed to be the flight altitude of the aerial robot. In presence of ground obstacles, this method only works if the laser sensor projects some beams on to the ground plane. Kumar et al., 2017 presents an aerial robotic system with integration of two 2D laser sensors and IMU in industrial environments for real-time pipeline classification. The authors use similar approach for flight altitude estimation presented by Wang et al., 2014, thus suffering from the same limitations.

2.1.1.2 Vision Based Techniques

Research also has been carried out estimating the altitude of aerial robot using vision-based systems. Cherian et al., 2009 evaluates a machine learning approach for estimating the altitude of an aerial robot using top-down images from a single on-board camera. The algorithm learns to estimate the altitude by mapping textured information in an image to a possible altitude value, thus it always requires a textured ground surface. It also provides higher errors in presence of ground obstacles.

Eynard et al., 2010 proposes a flight altitude estimation of an aerial robot using mixed stereoscopic vision system consisting of a combination of an omni directional camera with a perspective camera. The authors propose a plane sweeping approach to find a homography between the two views and with the sensor being calibrated and using an estimation of the attitude, estimate the flight altitude of the aerial robot along with estimating the ground plane. This technique can estimate the accurate flight altitude of an aerial robot only when the ground plane is present in the image, the more ground obstacles present the less accurate is the estimation of the altitude. The algorithm performs poorly in the presence of low texture or homogeneous planes. Mondragon et al., 2010 presents a real-time 3D pose estimation of an aerial robot, along with its altitude, with a single downward looking camera, using planar object tracking. Even though this method uses a single perspective camera for the pose estimation, it requires the use of artificial markers in the environment.

Natraj et al., 2011 evaluates a vision-based system consisting of a fisheye camera and a laser circle projector mounted on a fixed baseline. They provide a theoretical study of a camera modeled as a sphere and to obtain the attitude and the altitude of an aerial robot estimating a conic on this sphere. This approach assumes that the laser circle projector always projects on to the ground plane, hence does not provide a solution in presence of ground obstacles. Lange et al., 2012 proposes an autonomous corridor flight of an aerial robot using an RGB-D camera to estimate the distances from the walls of the corridors. The authors use a SRF10 sonar sensor for measuring the altitude of the aerial robot and 3D planar information from an RGB-D camera only for keeping the aerial robot aligned in the center of the corridor. The authors do not provide any solution for altitude estimation of the aerial robot in presence of ground obstacles, performing the experiments in corridors without ground obstacles.

Boutteau et al., 2018 proposes a system based on circular laser source and an RGB camera for estimating the attitude and altitude of a aerial robot. The authors develop a geometric modeling to estimate the pose at the scale of the system and relative to the plane onto which the pattern is projected, unable to estimate the correct altitude in presence of ground obstacles. Chirarattananon, 2017 proposes an altitude estimation of aerial robots combining the direct optic flow measurements from a single camera with the measurements from an IMU, but the author does not evaluate the performance of the algorithm in presence of ground obstacles.

2.1.2 Visual Pose Estimation

In indoor environments due to the absence of the GPS signals providing position information, the 6D pose estimation of the aerial robot is performed using either vision based sensors or laser based sensors. This obtained pose estimation can either be used directly as a feedback to the control system of the aerial robot or can be integrated with other sensor measurements such as IMU, using filtering based or non-linear optimization techniques, before being used as feedback to the control system, adding robustness to the pose estimation of the aerial robot. As the research presented in this thesis for pose estimation is based on vision techniques, Subsect. 2.1.2.1 presents the state of the art related to standalone geometric based vision based odometry techniques, where the

Figure 2.1: A layout of the different Visual Pose Estimation Techniques.

geometric techniques utilized low-level characteristic features such as points, lines, pixel intensities etc. Subsect. 2.1.2.2 and Subsect. 2.1.2.3 present geometric visual odometry techniques integrating additional sensor information using filtering based and non-linear optimization techniques respectively and Subsect. 2.1.2.4 present visual odometry techniques combining low-level geometric and high-level semantic features. Fig. 2.1 presents the layout of the entire vision based pose estimation techniques explained in this section. The three subsections are explained in detail below:

2.1.2.1 Standalone Geometric Visual Odometry

Visual odometry techniques are utilized for numerous aerial robotic research due to the lightweight RGB and RGB-D cameras. The first work on the visual odometry (VO) approaches was presented by Moravec, 1980 in order to estimate the ego-motion of a vehicle using visual input only. Several works since then have been carried out in order to robustly estimate the state of the robot. As explained in Scaramuzza and Fraundorfer, 2011, the VO techniques are based on three types namely, feature based, appearance based and hybrid techniques. Feature based techniques are based on characteristic or salient features found in the environment, whereas appearance based approaches are based on intensity changes in the images, with hybrid approaches being a combination of the two. These VO techniques can be further classified into three type based on the type of sensor utilized namely: (1) Monocular VO and (2) Stereo VO (3) RGB-D VO.

- **Monocular VO:** Monocular VO approaches estimate the 6D pose of the robot using

a single camera. Nister et al., 2004 presents the first real-time feature based monocular VO algorithm using RANSAC for outlier removal and 5-point solver approach (see Nistér, 2004) to solve the motion hypothesis in the RANSAC. Corke et al., 2004 and Lhuillier, 2005 present monocular VO approach using omni-directional cameras. In the field of appearance based or hybrid approaches, Forster et al., 2014 presents an monocular approach called Semi-direct VO (SVO), which works at pixel intensity level, at high frame rates of 55 FPS. The authors test the algorithms on-board an aerial robot with an embedded computer. Pizzoli et al., 2014 presents a real-time probabilistic dense monocular VO for 6D pose estimation as well as dense 3D reconstruction of the environment. Engel et al., 2018 presents a direct and sparse monocular VO running real-time on a CPU.

- **Stereo VO:** Stereo VO uses a combination of two cameras fixed at a known distance called baseline in order to estimate the 6D pose of the camera. Stereo VO approaches have an advantage over the monocular VO approaches that, they are able to estimate the camera motion with true metric scale and are robust to high rotational motions. Matthies, 1989 presents one of the first work related to stereo VO using a binocular system by detecting and tracking corners in the two images. Howard, 2008 presents a real-time stereo VO approach, without making prior assumptions regarding the camera motion and using a separate dense stereo VO algorithm for generating the disparity map. Fu et al., 2015 presents a stereo VO approach for aerial robots, consisting of a low cost ARM based stereo pre-processing system, thereby saving the computational load on the on-board computer. Manthe et al., 2018 presents a stereo VO approach based on FAST keyfeatures and rotated BRIEF descriptors to obtain an initial pose estimate, which is refined by solving the optimization problem of minimizing reprojection error as well as the cost function based on the epipolar constraint.
- **RGB-D VO:** RGB-D odometry is performed using RGB-D cameras, which provide both color images as well as the depth images. The main difference between RGB-D and stereo approaches is that many of the RGB-D cameras can already provide the depth information, whereas in case of stereo cameras, the depth needs to be calculated using images from the two RGB cameras. Steinbrücker et al., 2011 presents a real-time VO odometry for RGB-D cameras using a proposed energy function to find the best rigid transformation between the subsequent RGB-D images. Kerl et al., 2013b presents a robust RGB-D odometry approach registering two consecutive RGB-D frames directly by minimizing the photometric error between them. Their algorithm is able to run real-time on robotic systems with limited computational capability. Whelan et al., 2013 provides a real-time and fast GPU based implementation of an RGB-D odometry presented in Steinbrücker et al., 2011. Huang et al., 2017 presents a feature based RGB-D VO approach mounted on-board an aerial robot, in order to estimate its 6D pose as well as create a dense map of the environment using the depth image.

2.1.2.2 Geometric Visual Odometry and Filtering

This subsection presents the VO techniques integrated with additional sensors such as an IMU, using filtering techniques such as Extended Kalman Filters (EKF), in order to add robustness to the VO estimations. Mourikis and Roumeliotis, 2007 presents a real-time EKF based visual inertial odometry algorithm for navigation. The authors propose a measurement model that is able to express the geometric constraints arising from the static features, thus not requiring to add the 3D positions of the feature point into the EKF state. This keeps the computational complexity of the algorithm linear in the number of features. Li and Mourikis, 2013 presents a high precision EKF based VIO framework, improving the observability properties of the linearized system model and performing an online calibration of the camera-IMU parameters.

Bloesch et al., 2015 presents a tightly coupled monocular Visual Inertial Odometry (VIO) approach using an EKF. The authors detect the direct pixel intensity features at multilevel image patches, which are then tracked using an EKF which also integrates the measurements from a calibrated IMU sensor in order to achieve robust and real-time performance. Karamat et al., 2018 presents an approach for visual/inertial navigation fusing stereo visual odometry with the IMU data along with the wheel odometry data using an EKF, in order to increase the robustness of the system in presence of insufficient features for the VO.

2.1.2.3 Geometric Visual odometry and Non-Linear Optimization

Research based on non-linear optimization techniques for VIO estimation has increased due to the fact that, non-linear based optimization methods can handle highly non-linear motion and observation models as compared to the EKF based methods, which propagate the covariance through the linearization of the non-linear model.

Leutenegger et al., 2015 presents a VIO based approach using non-linear optimization, the authors use the harris corner detector followed by BRISK descriptor. The reprojection errors of the computed features along with the IMU measurements are optimized in order to estimate a robust pose of the robot. Forster et al., 2017 presents a real-time visual inertial odometry framework using the incremental smoothing framework for optimization which optimizes the states over a given window of previous states, thus avoiding optimization over all the 3D points, further accelerating the computation. Qin et al., 2018 presents a monocular Visual Inertial Navigation System (VINS) system, a tightly coupled non-linear optimization system, with pre-integrated IMU measurements and image feature observations. Rosinol et al., 2019 presents a feature based VIO pipeline, tightly coupling into it 3D mesh constraints, using a non-linear optimization formulation, thus improving the localization accuracy of the VIO algorithm in environments with structural regularities.

2.1.2.4 Geometric and Semantic Visual Odometry

All the above mentioned VO and VIO techniques use low-level geometric features such as keypoints, planes or intensity changes in order to estimate the pose of the robot.

These methods can accumulate errors in their estimations in absence of characteristic features in the environment or in presence of repetitive patterns due to degradation of the feature matching process. Due to advances in the field of object based detectors, several precise and real-time object based detectors have been researched and developed. Thus the research community is also focusing on researches based on additional high-level visual features obtained from these object detectors called the semantic features, in order to add additional robustness to the VO and VIO systems.

Atanasov et al., 2014 presents a Monte-Carlo based semantic localization for robots, using a prior map of semantic landmarks annotated with labels consisting of doors and chairs. The authors represent the obtained semantic information in form of random finite sets, which allows to incorporate missed and false detections. The authors use likelihood of these random finite sets in order to achieve robust data association of the detected semantic objects with the semantic map of the landmarks. Ma et al., 2017 provides an efficient self localization system for autonomous cars, incorporating semantic cues such as sun direction, road types, road intersections, into a particle filter. The direction of the sun is modeled as a Gaussian distribution whereas, the road type are exploited by means of a discrete distribution. An et al., 2017 presents a visual odometry algorithm using semantic attributes of the pixels in an RGB image as factors for candidates selection for the VO algorithm. Gaweł et al., 2018 presents, *X-view*, a semantics based global localization algorithm. The approach presents graph extraction from semantic input data and graph matching with a global graph using graph based descriptors. Lianos et al., 2018 proposes a Visual Semantic Odometry (VSO) framework, to enable tracking of the points using semantic information. The approach utilizes a cost function minimizing the semantic reprojection errors using Expectation Maximization (EM) scheme. They demonstrate the inclusion of the semantic cost term in the standard VO approaches reduces the translation drift. Their presented framework is easily integrable in the standard state of the art VO and VIO approaches. Radwan et al., 2018 proposes VLocNet++ architecture, a multi task learning based approach to exploit the inter relationships between learning semantics and regressing to a global pose as well as estimating the relative poses. The authors present a novel fusion strategy in the 3D pose regression network, incorporating information from the previous timesteps to estimate motion specific information by fusing the semantic information extracted from the semantic segmentation network.

2.2 Visual SLAM

Simultaneous Localization and Mapping (SLAM) enables a robot to create a map of an unknown environment and simultaneously localize itself in it. As compared to visual odometry techniques described in Sect. 2.1.2, creation of the map of the environment enables the robot to correct its localization error whenever it revisits an explored area, called loop closure. Whereas in the absence of a map, odometry based techniques can quickly drift over time due to noise in the sensor measurements. The research community has seen a wide interest in SLAM based applications applied to robotics, hence several SLAM algorithms have been proposed, resulting a vast SLAM based literature. We divide this section into Subsect. 2.2.1 which provides a review of the SLAM techniques using

Figure 2.2: A Layout of the different Visual SLAM Techniques.

the extracted geometric features of the environment, such as points, lines or planes and direct methods, based on direct pixel intensity errors. Subsect. 2.2.2 gives a review of the recent SLAM techniques combining the geometric as well as semantic information from the environment. Fig. 2.2 presents the layout of visual SLAM techniques presented in this section.

2.2.1 Geometric SLAM

Geometric SLAM using feature points, lines or planes can be performed using standard filtering techniques such as EKF-SLAM or Rao-Blackwellised particle filters or using non-linear optimization techniques. Non-linear optimization SLAM techniques also known as Graph SLAM have gained attention during the recent years due to their superior performance over the filtering techniques, as they optimize over all the previous known robot poses, whereas the filtering techniques optimize over the single last pose of the robot. This subsection reviews the different visual geometric SLAM approaches namely; (1) Monocular (2) Stereo and (3) RGB-D SLAM using both filtering as well as optimization techniques.

- **Monocular SLAM:** Two of the most convincing early works presented in the community of Monocular SLAM using single handheld cameras consists of work from Davison, 2003, Davison et al., 2007 using EKF-SLAM and Eade and Drummond, 2006 using Rao-Blackwellised particle filters. Davison, 2003, Davison et al., 2007 framework utilizes a top down Bayesian approach for searching landmarks in image

regions constrained by estimate uncertainty, instead of the costly bottom up image processing and feature matching. The approach although is not scalable to larger environments. Whereas, Eade and Drummond, 2006 presents the monocular SLAM using particle filters with a top down search to allow for real-time performance, mapping large number of landmarks, scalable to large scale environments.

Klein and Murray, 2007 propose a novel feature based monocular approach called PTAM, where the authors separate the feature tracking and mapping task into two parallel threads. This results in running the computationally expensive non-linear optimization technique in the background, without effecting the real-time performance of the algorithm. Thereby the authors are able to create a detailed map of the environment without effecting the real-time performance. Engel et al., 2012 present monocular camera based autonomous aerial robot, using the PTAM algorithm. The authors propose a novel-closed form solution to estimate the absolute scale of the visual map using the on-board inertial and altitude measurements. Brockers et al., 2014 modified the PTAM algorithm, in order to run it on-board a mini-aerial robotic platform with an embedded computer at 30 FPS. As an extension to PTAM, Mur-Artal and Tardos, 2016 present a monocular implementation of the feature based SLAM based on non-linear optimization using ORB features, called the ORB-SLAM2 Mono. ORB-SLAM2 consists of a local bundle adjustment, keypoints based current robot pose estimation, vision based loop closure detector and robot pose graph optimization. As known so far ORB-SLAM2 framework is one of the most complete monocular SLAM framework.

Several proposed SLAM based methods rely on direct image intensity errors as apposed to the feature based approaches and are called direct approaches. Newcombe et al., 2011b proposes the Direct Tracking and Mapping (DTAM) approach which uses direct image pixel intensities as apposed to the feature based PTAM approach. In DTAM, the tracking of the robot pose is performed using the alignment of the entire image against a created dense model of the environment. This makes the algorithm computationally expensive, and needs to be executed on a GPU for real-time performance. Engel et al., 2014 proposes a direct (featureless) semi dense monocular SLAM approach called Large-Scale Direct SLAM (LSD-SLAM), running real-time and capable of creating large scale consistent map of the environment. As opposed to feature based approaches the LSD-SLAM tracks the robot pose using the photometric error of each keyframe while creating a per-pixel depth map of the environment. LSD-SLAM performs global non-linear optimization using a direct image alignment formulation of the keyframes, which allows to detect loop closures between the keyframes, correcting the scale drift and thus generating globally consistent map of the environment. Greene et al., 2016 proposes an extension to LSD-SLAM algorithm, using a dense approach on top of the LSD-SLAM, thus improving the density of the depth maps and the reconstruction accuracy.

- **Stereo SLAM:** The essential difference between a stereo and monocular SLAM system is that the stereo SLAM system uses a setup of two cameras, and hence is able to estimate the pose of the robot and the visual map with a true metric scale. Several

works have been presented in the field of stereo SLAM systems. Iocchi et al., 2001, Garcia and Solanas, 2004, Sim et al., 2005 present some of the earliest works based on stereo based SLAM. Sim et al., 2005 presents a vision based Rao-Blackwellised particle filter approach with stereo camera setup, where the 3D landmark estimates are obtained using stereo vision and motion estimates using the visual odometry.

Pire et al., 2015 present an extension to the monocular PTAM approach, called Stereo-PTAM adding the parallel nature of the PTAM architecture to a stereo camera setup. Bellavia et al., 2013 present a technique called Selective-SLAM, which combines a loop chain matching strategy with an effective frame selection strategy. The idea of this algorithm is that the errors in the pose estimation are due to the uncertainty of 3D points. One key feature of this approach is that it does not require specific loop closure or bundle adjustment, the approach detects Harris corners from the images and matches them using the Gradient Location and Orientation Histogram descriptor.

Engel et al., 2015 provide an extension to the previous mentioned monocular LSD-SLAM in order to incorporate information from a stereo pair. Mur-Artal and Tardos, 2016 present ORB-SLAM2 Stereo as an extension of the ORB-SLAM2 Mono, consisting of the same approach applied to a stereo camera setup. Gomez-Ojeda et al., 2019 proposes a stereo based SLAM system called PL-SLAM, using a combination of keypoints and lines, in order to achieve robust performance in wider variety of scenarios with lower textures. PL-SLAM uses points and lines at all instances of the SLAM process, i.e motion estimation, keyframe selection and bundle adjustment. The loop closure is performed using the robust visual descriptors extracted using both points and lines.

- **RGB-D SLAM:** In recent years in the robotics community, several SLAM techniques using RGB-D cameras have been proposed due to emergence of newly developed low cost and lightweight RGB-D sensors. Newcombe et al., 2011a is one of the earliest techniques using RGB-D cameras, which fuses all the depth information into volumetric dense models, in order to track the camera pose using Iterative Closest Point (ICP) algorithm. This technique is unable to operate in large scale environment and cannot perform loop closing, which is extended by Whelan et al., 2012 implementing these capabilities. Sturm et al., 2012 provides a benchmark dataset for evaluation of RGB-D SLAM approaches and Endres et al., 2012 provides an evaluation of an RGB-D based algorithm on the provided dataset. The SIFT, SURF or ORB features are tracked in the RGB images with the true metric 3D transformation being computed using the tracked features in the depth information and is optimized using pose graph optimization technique. Kerl et al., 2013a presents a dense visual SLAM for RGB-D cameras, which minimizes photometric and the depth error over all pixel values. As opposed to feature based errors, using the photometric error, the algorithm is able to better exploit the available information for higher accuracy.

Whelan et al., 2015 presents ElasticFusion technique, estimating in real-time the pose of the camera along with creating dense globally consistent surfel based maps

of indoor room like environments. The authors argue, that using small local model to model loop closures along with global loop closures, the method can achieve globally consistent maps of the environment without requiring graph based optimization or other post processing. The approach presented by Mur-Artal and Tardos, 2016, along with ORB-SLAM2 mono and stereo setups also works with RGB-D cameras. The approach remains similar to its stereo SLAM approach except the pre-processing step which uses the depth image to synthesize a stereo coordinate for each extracted feature from the RGB image, making the framework unaware of the input being stereo or RGB-D.

2.2.2 Geometric and Semantic SLAM

An important part of a SLAM algorithm is correct data association which is required for accurate local and global loop closures. The previously presented existing low-level geometric SLAM approaches can present data association errors in cluttered or repetitive scenarios. This geometric information can be fused with high-level semantic information from the environment, improving the data association accuracy and adding robustness to the SLAM systems. Hence recently the research community has witnessed a rise in SLAM based approaches involving addition of semantic information to the SLAM framework.

Salas-Moreno et al., 2013 presents a real-time semantic SLAM approach called *SLAM++* based on an RGB-D camera. This approach tracks the current 3D pose of the camera using the ICP algorithm adding it to a pose graph. It then integrates into the pose graph, the relative 3D poses estimated from the successfully detected semantic objects, previously stored in a database, in order to optimize the estimated poses of the camera and the objects. Murali et al., 2017 presents an approach of integrating semantic information into a visual SLAM system. The gated factor graph framework, uses the semantic information for deciding the inliers/outliers of the system in order to achieve robust performance in presence of dynamic obstacles. A pre-trained deep learning based object detector is used to extract the semantic information of the objects. Sünderhauf et al., 2016 proposes an integrated approach for building point cloud maps including semantic labels using previously unknown instances of object classes. Parkhiya et al., 2018 presents a monocular semantic SLAM approach, where they use a deep network to learn 2D characteristic features from a category specific object like chair and match it with a 3D CAD model to estimate the relative 3D pose of the semantic object. These semantic objects are added as landmarks along with the VO estimated pose of the robot into a graph optimization framework to obtain a true metric pose of the robot. Grinvald et al., 2019 presents a semantic mapping framework, requiring a pose acquired from a geometric VIO sensor. This method utilizes geometric planar segmentation of a point cloud data, which is refined using the semantic detections, as well as uses the semantic information during the data association step. McCormac et al., 2018 presents an object-level SLAM system using RGB-D cameras called *Fusion++*, segmenting Truncated Signed Distance Function (TSDF) representations of the objects using the Mask-RCNN object detector. The objects are used for tracking, re-localization and loop closure, and their extracted pose is optimized over a pose graph. Bowman et al., 2017 provides an extension to their work presented in Atanasov et al.,

2014, presenting a semantic SLAM approach, using semantic objects such as chairs and doors. The authors decompose the joint metric semantic SLAM problem into subcategories namely (1) Continuous optimization of pose and (2) Discrete optimization of the semantic data association and semantic labels. The framework tightly couples the inertial, geometric and semantic information. Atanasov et al., 2018 provides an extension to the framework presented in Bowman et al., 2017, extracting descriptive semantic features using a convolutional neural network from semantic objects like cars, in order to tightly couple them with the geometric and inertial information.

2.3 Autonomous Flights

As seen from Sect. 2.1 and Sect. 2.2 advances in real-time perception and localization algorithms have made the recent robotic systems advanced and intelligent for solving challenging and difficult real world applications. This has increased the demand of the autonomous aerial robots carrying several on-board sensors to efficiently solve real world applications. Performing autonomous flights using aerial robots requires a proper integration of all such individually developed algorithms and a rigorous testing in order to validate their proper functionality in the required environmental conditions. The following Subsect. 2.3.1 gives a brief review of the wide range of industrial applications solved by autonomous aerial robots and Subsect. 2.3.2 provides the review of the challenging problems solved by autonomous aerial robots proposed by several international robotic challenges.

2.3.1 Industrial Applications

Aerial robots are increasingly being used in industrial applications such as visual inspections of confined, dangerous and elevated spaces, which generally requires use of cranes and scaffoldings involving constant human interaction. Autonomous aerial robots are the preferred choices in such scenarios as they can reach the required altitudes for visual inspections without any supported structures and minimum human intervention.

Nikolic et al., 2013 presents an autonomous aerial robotic system capable of inspecting a confined thermal power plant. The system consists of a small and light weight aerial robotic system equipped with a pair of stereo cameras along with a inertial measurement unit and on-board illumination source, capable of performing stereo visual inertial odometry, which serves an input to the Modern Predictive Control (MPC) for reliable control of the aerial robot. The authors provide preliminary results only inspecting a part of power plant upto a certain approximate altitude of 15-17 m not inspecting the important parts of the power plant such as the burners located at altitudes of 40-60 m. Tripicchio et al., 2018 presents a semi-autonomous approach using an aerial robot for inspection of a thermal power plant using laser range finder. The system is teleoperated by a user, who sends high-level position commands to the aerial robot, while the 2D pose of the aerial robot is estimated using 2D SLAM based on a particle filter technique and the altitude is estimated using the upward and downward facing sonar altimeters.

Nieuwenhuisen et al., 2017b, Quenzel et al., 2019 present an autonomous aerial robotic system for visual inspections of chimneys. The system consists of a lightweight aerial robotic platform, with an on-board 3D laser sensor for obstacle avoidance, high resolution stereo cameras for performing visual odometry as well as visual inspections of the structure, along with bright LEDs for on-board illumination. Their system is able to inspect an entire industrial chimney and perform a surface reconstruction in order to detect the defects on the surfaces from the reconstructed surfaces.

Bonnin-Pascual et al., 2012, Bonnin-Pascual et al., 2015 present an approach using an intelligent aerial robotic platform, for inspecting marine vessels. Bonnin-Pascual et al., 2012 consists of a fully autonomous system using laser based SLAM for localization and navigation using waypoints commands to the position control loop. Bonnin-Pascual et al., 2015 consists of a semi-autonomous system with user introduced in the position control loop in order to indicate the required displacements commands to the intelligent aerial robotic system capable of performing obstacle avoidance, while the user can focus on the visual inspection. Their system is able to detect the corruptions and cracks on the vessel surfaces.

Beul et al., 2018 presents an autonomous warehouse inventory solution using an aerial robot with an on-board 3D laser sensor and two high resolution cameras. The aerial robot is equipped with an RFID reader which detects and makes an inventory of the stock as it navigates through the shelves, avoiding obstacles and following the trajectory generated by a trajectory planner which receives the mission points from a mission planner.

ozaslan et al., 2016, ozaslan et al., 2017 present an approach for autonomous inspections of cylindrical penstocks using aerial robots. The authors estimate the pose and velocity of the aerial robot using a fusion of on-board sensors like the IMU, 3D laser sensor and four RGB cameras as well as use on-board LEDs for illumination inside the dark penstocks. Their approach which is combination of control, state estimation and environment mapping is validated through several experiments at a real penstock. Papachristos et al., 2019 performs an autonomous navigation and mapping of an underground metal mine using an aerial robot. The system fuses a combination of thermal and RGB cameras along with a lidar and IMU sensor to estimate the state of the aerial robot in dark and featureless environments. A path planning system is used for collision avoidance along with an MPC for robust trajectory tracking of the aerial robot.

2.3.2 International Competitions

Several challenging international competitions are being organized every year for autonomous aerial robots, where these robots have to solve real life problems arising in the every day life. Some of the commonly known competitions related to aerial robots include:

- *International Micro-Aerial Vehicles Competition*¹: The IMAV which was inaugurated in the year 2007, is organized every year in different countries around the world, focused on challenges for autonomous aerial robots in indoors as well as outdoor scenarios.

¹<http://www.imavs.org/>

- *Mohamed Bin Zayed International Robotics Challenge*²: The MBZIRC competition is one of the worlds largest competition for autonomous aerial robots organized every 2 years in Abu Dhabi. It has several challenges for aerial robots in outdoor scenarios. Universities as well as companies from all around the world participate in this competition which has a total prize amount of 3 million dollars.
- *IROS Drone Racing*³: The IROS drone racing is organized every year at the International Conference on Intelligent Robots and Systems (IROS), with a goal to reward the fastest autonomous aerial robot capable of crossing the entire racing circuit.
- *AI Robotic Racing*⁴: The AIRR is a new competition focused on solving the drone racing challenge using autonomous aerial robots powered by artificial intelligence. The autonomous aerial robot which wins finally competes against the human world champion in drone racing.

Several research works presented during the international competitions solving complex challenges are mentioned below:

Pestana et al., 2014, Sanchez-Lopez et al., 2014 present an autonomous framework for swarm of aerial robots in order to solve the mission of the IMAV 2013 competition. Some of the important tasks of the mission consisted of the aerial robots to pass through a window, fly through the obstacles in the obstacle zones, recognize a target and perform precision landing. Each aerial robot consisted of a Parrot A.R drone 2.0, performing visual localization, integrating measurements from Visual ArUco markers, IMU and optical flow data using an EKF-based localization and mapping algorithm. The Aruco markers are used to detect and avoid the obstacles in real-time. This solution was awarded the first prize in the indoor competition.

Sanchez-Lopez et al., 2015 presents an autonomous aerial robot for the International Aerial Robotics Competition (IARC), consisting of a system of five RGB based cameras, an optical flow sensor and a ultrasonic altimeter. The mission consisted of navigation of an aerial robot in GPS/laser denier environment, performing robot-robot interaction, avoiding obstacles in order to solve a ground robot herding problem. The authors propose an autonomous aerial robotic solution using a vision based localization strategy using grid keylines/points, autonomous control algorithm and a dynamic mission planner.

Beul et al., 2019, Beul et al., 2017, Nieuwenhuisen et al., 2017a demonstrate their autonomous aerial robotic system for solving the Mohamed Bin Zayed International Robotics (MBZIRC) 2017 competition. The authors present the research performed and results obtained in the two challenges. Challenge one requires an aerial robot to perceive a moving target and perform fast and precise control to land on it, whereas challenge three consists of a treasure hunt, requiring a coordination between the aerial and ground robots. The authors present a detailed explanation of the algorithms used for perception, localization, planning and control of the aerial robot/s, in order to achieve first position in the grand

²<https://www.mbzirc.com/>

³<https://www.iros2019.org/call-for-competitions>

⁴<https://thedroneracingleague.com/airr/>

challenge of the MBZIRC 2017 competition. Baca et al., 2019 presents their autonomous hexarotor based aerial robotic platform for landing on the moving target moving at 15 km/h in the challenge one of the MBZIRC 2017 competition. This paper presents a novel control strategy using MPC which generates a reference trajectory for the aerial robot, which is tracked by a non-linear feedback control, allowing to minimize the position errors of the aerial robot with respect to the moving target. This approach won the first price for the challenge one.

Kaufmann et al., 2018 presents a fast and agile autonomous aerial robot for enabling high speed flights in dynamic environments. The approach won first place the IROS 2018 drone racing challenge. It consists of a Constitutional Neural Network (CNN) detecting and predicting the poses of the gates along with its uncertainty using an RGB camera. These predictions are incorporated into an EKF in order to filter any spurious predictions. Using the EKF pose estimates of the gates, the authors use MPC, rapidly passing through the gates and navigating through the entire track.

Part II

POSE ESTIMATION

Chapter 3

Altitude Estimation using Conventional Sensors

3.1 Introduction

In autonomous indoor missions, the aerial robots are required to have capabilities which allow them to navigate in unknown and unstructured environments in presence of several ground obstacles, perceiving the environment and interacting with its diverse components. To successfully execute such kind of missions, aerial robots are required to maintain the commanded flight altitude while performing these navigation, perception and interaction tasks. In such scenarios, a fast and robust estimation of the aerial robot flight altitude becomes a critical necessity for such autonomous missions.

It can be assumed that the vertical and horizontal motion of aerial robots can be decoupled, hence several research works have focused on fusing the horizontal localization component (example, 2D SLAM) with a vertical localization component. To maintain a constant desired flight altitude even when flying over non-flat surfaces (obstacles undetected by the horizontal localization component), it is essential to accurately perceive the non-flat ground surface in order to estimate the true flight altitude of the aerial robot.

Different kinds of techniques have been introduced in the literature for altitude estimation of aerial robots in GPS-denied indoor environments using several kinds of conventional sensors such as: laser altimeters, ultrasonic altimeters, barometers and accelerometers (IMU). Laser altimeters are very accurate but, in cluttered environment with various ground obstacles, they measure the altitude only with respect to the ground obstacles. The ultrasonic altimeters are similar to the laser altimeters, generally have shorter maximum range as compared to laser altimeters and measure the altitude with respect to the ground obstacles. Hence when directly using the measurements from the laser or the ultrasonic altimeters in a cluttered environment consisting of ground obstacles, it is not possible to accurately measure the flight altitude of the aerial robot.

Barometers measure the altitude from the pressure changes in the environment, but they are very noisy and biased when used indoors and on-board the aerial robots, due to

the ceiling, ground, and wall effects. Accelerometers can be used to provide an estimate of altitude, double integrating the vertical accelerations, but can accumulate huge drift and bias in its measurements due to the vibrations during the flight.

In order to overcome the drawbacks of each sensor, being able to robustly estimate the flight altitude of aerial robots in indoor environments in presence non-flat ground surfaces, this chapter presents a modular and open-source multi-sensor fusion state estimator, capable of fusing measurements from any number of different sensors such as IMUs, barometers and distance sensors. Additionally, the proposed state estimator is capable of estimating the elevation of the surface below the aerial robot even in presence of complex and unstructured ground obstacles, allowing a steady flight navigation over such non-flat ground surfaces.

Figure 3.1: The aerial platform used validating the proposed state estimator describing its frame of references.

3.2 Proposed Multi-Sensor Fusion State Estimator

In this work, we present a modular state estimator based on an extended Kalman filter (EKF). Our proposed state estimator, represented in Fig. 3.2, combines measurements from several sensors, including IMU, barometer and distance sensor, and estimates the robot states $\hat{\mathbf{x}}_R$ and the ground object state $\hat{\mathbf{x}}_G$ (elevation of the ground surface). The proposed state estimator has the following main features:

- Modularity on the number of sensors used, what allows a usage of a variable number and kind of sensors for the state estimation.
- Auto-calibration of sensors, allowing to estimate the internal state of some sensors (e.g. biases on the measurement), providing a more robust and accurate state estimation.

Figure 3.2: The inputs and outputs of the proposed state estimator.

- Synchronous prediction of the estimated state: Whenever an estimation of the state is required, the prediction stage is executed.
- Asynchronous update of the estimated state based on the available measurements: Whenever a measurement arrives to the state estimator, both prediction and update stage are executed consecutively, allowing the accurate usage of sensors with different measurement rate.

As stated before, the proposed state estimator is modular, being its available components, deeply described in Sect. 3.3, the following: robot (R), ground object (G), IMU sensor ($I\star$), barometer sensor ($B\star$) and distance sensor ($D\star$).

In the following sections, the combination between the different components is described in detail. Sect. 3.2.1, explains the state and process model composition taking into account the designed modularity; and Sect. 3.2.3, indicates how the available measurements are gathered, forming the measurement model. Additionally, Sect. 3.2.2 and Sect. 3.2.4 present the widely known extended Kalman filter equations used for the prediction stage and update stage respectively. For the purpose of this chapter, $\sin(x)$ and $\cos(x)$ are abbreviated as $s(x)$ and $c(x)$ respectively.

3.2.1 State and process model

The state vector, $\mathbf{x}$, is formed by the state vector of all the components, similar to the process model noise $\mathbf{n}$, being:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_R \\ \mathbf{x}_G \\ \mathbf{x}_{I\star} \\ \mathbf{x}_{B\star} \\ \mathbf{x}_{D\star} \end{bmatrix} \quad \mathbf{n} = \begin{bmatrix} \mathbf{n}_R \\ \mathbf{n}_G \\ \mathbf{n}_{I\star} \\ \mathbf{n}_{B\star} \\ \mathbf{n}_{D\star} \end{bmatrix}$$

where the process model noise has an associated covariance matrix $\mathbf{T}$.

In the same way, the discrete time process model formed by the individual discrete time process models of all the components, being:

$$\mathbf{x}(k) = \mathbf{f}(\mathbf{x}(k-1), \Delta t, \mathbf{n})$$

where Δt is the change in time between k and $k + 1$.

The Jacobian matrices of the discrete time process model are given by:

$$\mathbf{F}_x = \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \quad \mathbf{F}_n = \frac{\partial \mathbf{f}}{\partial \mathbf{n}}$$

3.2.2 Prediction Stage

During the prediction stage, the state $\mathbf{x}$ is estimated in the current time instant k , taking into account its previous estimate in $k - 1$, and based on the process model, following the EKF equations:

$$\hat{\mathbf{x}}(k|k-1) = \mathbf{f}(\hat{\mathbf{x}}(k-1|k-1), \Delta t) \quad (3.1)$$

$$\mathbf{P}(k|k-1) = \mathbf{F}_x \cdot \mathbf{P}(k-1|k-1) \cdot \mathbf{F}_x^T + \mathbf{F}_n \cdot \mathbf{T} \cdot \mathbf{F}_n^T \quad (3.2)$$

$\mathbf{P}$ being the covariance matrix of the estimated state $\hat{\mathbf{x}}$.

3.2.3 Measurements and measurement model

The available measurements $\mathbf{z} = \{\mathbf{z}_{I\star}, \mathbf{z}_{B\star}, \mathbf{z}_{D\star}\}$ at a particular instant time k , determine the measurement model, that changes dynamically, being:

$$\mathbf{z}(k) = \mathbf{h}(\mathbf{x}(k), \boldsymbol{\epsilon})$$

where $\boldsymbol{\epsilon}$ is the measurement model noise, with an associated covariance matrix $\mathbf{R}$.

The Jacobian matrices of the measurement model are given by:

$$\mathbf{H}_x = \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \quad \mathbf{H}_\epsilon = \frac{\partial \mathbf{h}}{\partial \boldsymbol{\epsilon}}$$

3.2.4 Update Stage

During the update stage, the state is modified taking into account its prediction, the measurement model and the available measurements, following the EKF equations.

First of all, the measurement residual $\mathbf{v}$ is computed as:

$$\mathbf{v} = \tilde{\mathbf{z}}(k) - \mathbf{h}(\hat{\mathbf{x}}(k|k-1)) \quad (3.3)$$

with an associated covariance matrix:

$$\mathbf{S} = \mathbf{H}_x \cdot \mathbf{P}(k|k-1) \cdot \mathbf{H}_x^T + \mathbf{H}_\epsilon \cdot \mathbf{R} \cdot \mathbf{H}_\epsilon^T \quad (3.4)$$

The near optimal kalman gain $\mathbf{K}$ is then computed as

$$\mathbf{K} = \mathbf{P}(k|k-1) \cdot \mathbf{H}_x^T \cdot \mathbf{S}^{-1} \quad (3.5)$$

And the estimated state is updated as:

$$\mathbf{x}(k|k) = \mathbf{x}(k|k-1) + \mathbf{K} \cdot \mathbf{v} \quad (3.6)$$

$$\mathbf{P}(k|k) = (\mathbf{I}_n - \mathbf{K} \cdot \mathbf{H}_x) \cdot \mathbf{P}(k|k-1) \quad (3.7)$$

n being the dimension of the state $\mathbf{x}$, and $\mathbf{I}_n$ the identity matrix of dimension $n \times n$.

3.3 Components of the Proposed State Estimator

This section describes the components of the proposed state estimator. All the sensors are assumed to be located in the center of the robot, with their reference frame being aligned with the robot reference frame as shown in Fig. 3.1.

3.3.1 Aerial Robot

The aerial robot state $\mathbf{x}_R$ includes:

- Vertical coordinate of the robot reference frame in world coordinates, $t_{z_R}^W$.
- Vertical coordinate of the velocity of the robot with respect to world, in world coordinates, $v_{z_R|W}^W$.
- Vertical coordinate of the acceleration of the robot with respect to world, in world coordinates, $a_{z_R|W}^W$.
- Attitude of the robot reference frame in world coordinates, $\boldsymbol{\theta}_R^W = [\theta_{x_R}^W, \theta_{y_R}^W, \theta_{z_R}^W]^T$, as Euler angles.
- Angular velocity $\boldsymbol{\omega}_{R|W}^R = [\omega_{x_R|W}^R, \omega_{y_R|W}^R, \omega_{z_R|W}^R]^T$, of the robot with respect to world in robot coordinates, .

The aerial robot is assumed to change its vertical acceleration and its angular velocity slowly, permitting the usage of a constant vertical acceleration and constant angular velocity as process model. Thus this model is only valid when assuming the aerial robot is a multirotor. Other important consequence of this slow motion, is that their pitch and roll angles, can be approximated as zero when required: $\theta_{x_R}^W \approx 0$ and $\theta_{y_R}^W \approx 0$.

According to the kinematics of a moving body, relationship between Euler rates and angular velocities can be given as:

$$\begin{bmatrix} \omega_{x_R|W}^R \\ \omega_{y_R|W}^R \\ \omega_{z_R|W}^R \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s(\theta_{y_R}^W) \\ 0 & c(\theta_{x_R}^W) & s(\theta_{x_R}^W) \cdot c(\theta_{y_R}^W) \\ 0 & -s(\theta_{x_R}^W) & c(\theta_{x_R}^W) \cdot s(\theta_{y_R}^W) \end{bmatrix} \begin{bmatrix} \dot{\theta}_{x_R}^W \\ \dot{\theta}_{y_R}^W \\ \dot{\theta}_{z_R}^W \end{bmatrix} \quad (3.8)$$

As the $\theta_{x_R}^W \approx 0$ and $\theta_{y_R}^W \approx 0$ relationship between the Euler angles and angular velocities can be reduced as:

$$\omega_{x_R|W}^R \approx \dot{\theta}_{x_R}^W \quad (3.9)$$

$$\omega_{y_R|W}^R \approx \dot{\theta}_{y_R}^W \quad (3.10)$$

$$\omega_{z_R|W}^R \approx \dot{\theta}_{z_R}^W \quad (3.11)$$

Therefore, the discrete time process model of the robot, $\mathbf{f}_R$, is simplified to:

$$t_{zR}^W(k) = t_{zR}^W(k-1) + v_{zR|W}^W(k-1) \cdot \Delta t \quad (3.12)$$

$$+ \frac{\Delta t^2}{2} \cdot a_{zR|W}^W(k-1) \quad (3.13)$$

$$v_{zR|W}^W(k) = v_{zR|W}^W(k-1) + a_{zR|W}^W(k-1) \cdot \Delta t \quad (3.14)$$

$$a_{zR|W}^W(k) = a_{zR|W}^W(k-1) + n_{a_z} \quad (3.15)$$

$$\boldsymbol{\theta}_R^W(k) \approx \boldsymbol{\theta}_R^W(k-1) + \boldsymbol{\omega}_{R|W}^R(k-1) \cdot \Delta t \quad (3.16)$$

$$\boldsymbol{\omega}_{R|W}^R(k) = \boldsymbol{\omega}_{R|W}^R(k-1) + \mathbf{n}_\omega \quad (3.17)$$

where the noise of the process model of the robot, $\mathbf{n}_R$, with an associate covariance matrix $\mathbf{T}_R$, is given as:

$$\mathbf{n}_R = \begin{bmatrix} n_{a_z} \\ \mathbf{n}_\omega \end{bmatrix} \quad (3.18)$$

3.3.2 Ground object

The state of the ground object $\mathbf{x}_G$ includes:

- Altitude of the ground object in world coordinates, t_{zG}^W .

The world reference frame is assumed to be in the lower location of the environment in the z -axis coordinate, that means, that the altitude of the ground object in world coordinates is always positive.

Being, the discrete time process model of the ground object, $\mathbf{f}_G$, given by:

$$t_{zG}^W(k) = \begin{cases} t_{zG}^W(k-1) + n_{zG}, & \text{if } t_{zG}^W(k-1) > 0 \\ 0 + n_{zG}, & \text{if } t_{zG}^W(k-1) \leq 0 \end{cases} \quad (3.19)$$

where the noise of the process model of the ground object, $\mathbf{n}_G$, with an associate covariance matrix $\mathbf{T}_G$, is given as:

$$\mathbf{n}_G = [n_{zG}] \quad (3.20)$$

3.3.3 IMU sensor: accelerometer, gyro and magnetometer

The IMU sensor considered in our state estimator is an intelligent component formed by an accelerometer, a gyro and a magnetometer, with an internal state estimator that does not provide the raw measurements, but a filtered estimation of them. This consideration is not too strong as every autopilot hardware used for multirotor platforms incorporates this kind of IMU sensor.

The acceleration measurement is assumed to have a bias modeled as a random process, as it randomly changes its value with the time.

The state of the IMU sensor, $\mathbf{x}_{I*}$, includes:

- Bias on the vertical measurement of the acceleration, $b_{az\star}$.

Being the discrete time process model of the IMU sensor, $\mathbf{f}_{I\star}$, given by:

$$b_{az\star}(k) = b_{az\star}(k-1) + n_{b_{az\star}} \quad (3.21)$$

where the noise of the process model of the IMU sensor, $\mathbf{n}_{I\star}$, with an associate covariance matrix $\mathbf{T}_{I\star}$, is given as:

$$\mathbf{n}_{I\star} = [n_{b_{az\star}}] \quad (3.22)$$

The IMU sensor provides the following measurements, $\mathbf{z}_{I\star}$:

- Acceleration measured, $\mathbf{z}_a = [z_{ax}, z_{ay}, z_{az}]^T$.
- Attitude measured, $\mathbf{z}_\theta = [z_{\theta x}, z_{\theta y}, z_{\theta z}]^T$.
- Angular velocity measured, $\mathbf{z}_\omega = [z_{\omega x}, z_{\omega y}, z_{\omega z}]^T$.

Nevertheless, only the following measurements are taken into account: z_{az} , $z_{\theta x}$, $z_{\theta y}$, $z_{\omega x}$, and $z_{\omega y}$.

Being its measurement model, $\mathbf{h}_{I\star}$, given by:

- Acceleration measured:

$$\mathbf{z}_a = \mathbf{a}_{R|W}^R - \mathbf{g}^R + \mathbf{b}_a + \boldsymbol{\epsilon}_a \quad (3.23)$$

$$= (\mathbf{R}_R^W)^T \cdot (\mathbf{a}_{R|W}^W - \mathbf{g}^W) + \mathbf{b}_a + \boldsymbol{\epsilon}_a \quad (3.24)$$

where $\mathbf{R}_R^W$ is the 3D rotation matrix derived from the Euler angles.

Since we consider $\theta_{xR}^W \approx 0$ and $\theta_{yR}^W \approx 0$ the rotation matrix can be approximated as:

$$\mathbf{R}_R^W \approx \begin{bmatrix} c(\theta_{zR}^W) & -s(\theta_{zR}^W) & 0 \\ s(\theta_{zR}^W) & c(\theta_{zR}^W) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.25)$$

As only the vertical component is taken into account:

$$z_{az} \approx a_{zR|W}^W - g_z^W + b_{az\star} + \epsilon_{az} \quad (3.26)$$

- Attitude measured:

$$\mathbf{z}_\theta = \boldsymbol{\theta}_R^W + \boldsymbol{\epsilon}_\theta \quad (3.27)$$

And only two components are taken into account as:

$$z_{\theta x} = \theta_{xR}^W + \epsilon_{\theta x} \quad (3.28)$$

$$z_{\theta y} = \theta_{yR}^W + \epsilon_{\theta y} \quad (3.29)$$

- Angular velocity measured:

$$\mathbf{z}_\omega = \boldsymbol{\omega}_{R|W}^R + \boldsymbol{\epsilon}_\omega \quad (3.30)$$

And only two components are taken into account as:

$$z_{\omega x} = \omega_{xR|W}^R + \epsilon_{\omega x} \quad (3.31)$$

$$z_{\omega y} = \omega_{yR|W}^R + \epsilon_{\omega y} \quad (3.32)$$

where the noise of the measurement model of the IMU sensor $\boldsymbol{\epsilon}_{I\star}$, with an associate covariance matrix $\mathbf{R}_{I\star}$, is given as:

$$\boldsymbol{\epsilon}_{I\star} = \begin{bmatrix} \epsilon_{az} \\ \epsilon_{\theta x} \\ \epsilon_{\theta y} \\ \epsilon_{\omega x} \\ \epsilon_{\omega y} \end{bmatrix} \quad (3.33)$$

3.3.4 Barometer sensor

The Barometer sensor considered in our state estimator is an intelligent component that provides direct measurements of the flight altitude instead of raw pressure measurements thanks to an internal state estimator. As with the IMU sensor, this consideration is not too strong since every autopilot hardware used for multirotor platforms incorporates this kind of sensor.

Similar to the acceleration measurement, the barometer measurement is assumed to have a bias modeled as a random process, as it randomly changes its value with the time.

The state of the Barometer sensor, $\mathbf{x}_{B\star}$, includes:

- Bias on the vertical measurement of the barometer, $b_{bz\star}$.

Being the discrete time process model of the barometer sensor, $\mathbf{f}_{B\star}$, given by:

$$b_{bz\star}(k) = b_{bz\star}(k-1) + n_{b_{bz\star}} \quad (3.34)$$

where the noise of the process model of the barometer sensor, $\mathbf{n}_{B\star}$, with an associate covariance matrix $\mathbf{T}_{B\star}$, is given as:

$$\mathbf{n}_{B\star} = [n_{b_{bz\star}}] \quad (3.35)$$

The barometer sensor provides the following measurements, $\mathbf{z}_{B\star}$:

- Vertical coordinate in world reference frame, z_{bz}

Being its measurement model, $\mathbf{h}_{B\star}$, given by:

$$z_{bz} = t_{zR}^W + b_{bz\star} + \epsilon_{bz} \quad (3.36)$$

where the noise of the measurement model of the barometer sensor $\boldsymbol{\epsilon}_{B\star}$, with an associate covariance matrix $\mathbf{R}_{B\star}$, is given as:

$$\boldsymbol{\epsilon}_{B\star} = [\epsilon_{bz}] \quad (3.37)$$

3.3.5 Distance sensor

The distance sensor does not incorporate any state, and therefore, it does not add any process model.

The distance sensor provides the following measurement, $z_{D\star}$:

- Distance measured to the ground object, z_d .

Being its measurement model, $h_{D\star}$, given by:

$$z_d = \frac{t_{z_R}^W - t_{z_G}^W}{\cos(\theta_{x_R}^W) \cdot \cos(\theta_{y_R}^W)} + \epsilon_d \quad (3.38)$$

where the noise of the measurement model of the distance sensor $\epsilon_{D\star}$, with an associate covariance matrix $R_{D\star}$, is given as:

$$\epsilon_{D\star} = [\epsilon_d] \quad (3.39)$$

3.4 Experiments and Results

In this section, the experiments that have been conducted in order to validate the proposed approach are presented. The aim of the validation is twofold. First, to demonstrate the capabilities of the designed state estimator in presence of several obstacles located on the ground and second, to evaluate the behavior of the proposed system in presence of unexpected sensor measurements. We refer the reader to a video¹ demonstrating the capabilities of the proposed state estimator. This section is divided into two subsections, in Sect. 3.4.1 the entire experimental setup including the sensors and computational hardware as well as the software used is explained. In Sect. 3.4.2 the results obtained in real flights are shown and analyzed.

3.4.1 Experimental Setup

3.4.1.1 System Setup

Fig 3.1 represents the custom aerial platform used for validating the proposed state estimator. The autopilot utilized in the real flight experiments is Pixhawk. Three different onboard sensors are used in the presented state estimator: a 1D laser altimeter Lightware SF/10A with a minimum and maximum range of 0 and 25 m respectively, a resolution of 1 cm and an accuracy of 0.05 m in the absence of sensor noise; a 9-axis IMU integrated within the Pixhawk autopilot; and a barometer also integrated with the Pixhawk autopilot with an altitude resolution of 0.1 m.

The entire computation is performed on-board an Intel-NUC 6i5SYK computer. All the above mentioned sensors communicate with the on-board computer through USB communication. The software modules are developed in C++, using C++ 11 standards.

¹<https://vimeo.com/205774733>

Regarding the software system, Aerostack software framework was utilized. A full description of Aerostack architecture and its components is out of scope of this chapter, and can be reviewed in Sanchez-Lopez et al., 2016 and Chapter. 7. The complete description of the components used for the autonomous flights can be found in Sampedro et al., 2017 and Chapter. 8.

3.4.1.2 Real Flights configuration

In order to validate the approach, several fully autonomous real flights are performed in different challenging indoor scenarios.

- *Navigation in unknown and unstructured indoor environment:* The scenario, with a size of $5\text{ m} \times 9\text{ m}$, consists of several obstacles with height smaller than the flight altitude of the aerial robot in order to be undetected by the 2D SLAM. The obstacles consist of several tables with a uniform surface. Several chairs are placed with random discontinuity in their surface increasing the noise in the measurements provided by the 1D laser altimeter.
- *Take-off and land from a platform:* The platform, located in an indoor scenario, with a size of $5\text{ m} \times 9\text{ m}$, is a square sized platform with an area of 140 cm^2 and height of 47 cm . In this experiment the taking-off and landing is performed over the platform.

3.4.2 Results

In this section, all the results obtained during the execution of the real flight experiments explained in Sect. 3.4.1.2 are described in detail. In both the experiments the commanded flight altitude to the aerial robot is 1.4 m .

3.4.2.1 Navigation in unknown and unstructured indoor environment:

Fig. 3.3 represents the results of the complete real flight performed during this experiment. The experiment is divided into three main phases: (1) Take-off (2) Navigation (3) Land.

- *Take-off and land phase:* As can be observed in Fig. 3.3 between time instances $t = 0\text{ s}$ to $t = 10\text{ s}$ the aerial robot performs the take-off task from a flat ground surface, whereas from instances $t = 210\text{ s}$ to $t = 220\text{ s}$ the landing task is executed.
- *Navigation phase:* After the take-off phase, the navigation phase is performed over several complex shaped ground obstacles such as chairs, tables, boxes, etc. The time instance between $t = 20\text{ s}$ to $t = 200\text{ s}$ (see Fig. 3.3) presents the navigation phase.

It can be seen that during the entire experiment the aerial robot traverses over different ground obstacles, with its flight altitude being accurately estimated by our proposed state estimator. The results of this experiment are discussed in Section. 3.5

Figure 3.3: Results obtained while performing take-off, navigation and land tasks during the execution of a real flight.

3.4.2.2 Take-off from the platform

In this experiment, an important variation in terms of the configuration of the proposed scenario is presented. In this case, the aerial robot is required to take-off and land from an elevated surface (platform) adding additional noise in the sensor measurements. The experiment can be divided as:

- *Take-off from the platform:* As shown in Fig. 3.5 during time instances $t = 0$ s and $t = 20$ s the aerial robot which is landed on the platform, performs the take-off task. The state estimator is initialized with both, the initial flight altitude and the ground obstacle height equal to the height of the platform (0.47 m).
- *Navigation Phase:* Once the aerial robot performs the take-off through the platform it performs the navigation phase over several other similar ground obstacles such as chairs, tables, boxes, etc. The time instance between $t = 20$ s to $t = 110$ s in Fig. 3.5 presents the navigation phase.
- *Land on the platform:* Between time instances $t = 110$ s and $t = 120$ s the aerial robot performs the task of landing on the platform (see Fig. 3.5). The landing task consists of two stages the (1) Approach stage where the aerial robot starts descending on the platform and (2) The fast landing stage (a free fall of the aerial robot on the platform), is triggered if the current flight altitude is less than the threshold altitude to land.

Figure 3.4: Navigation phase of the same flight represented in Fig. 3.3, when the aerial robot is flying over several obstacles.

Even with the added complexity during the take-off and landing tasks our proposed flight altitude estimator accurately estimates the flight altitude of the aerial robot. The results of this experiment are discussed in Sect. 3.5

3.5 Discussions

In this section the results obtained during the performed experiments are discussed in detail.

3.5.0.1 Navigation in unknown and unstructured indoor environment

- *Take-off and Land Phase:* It can be noticed that, the state estimator is able to accurately estimate the flight altitude without any delay during these phases (which is essential for fast take-off and landing maneuvers) when compared to the altitude provided by the 1D laser altimeter. As seen in Fig. 3.3, the barometer provides biased flight altitude measurements, with its measurement far from the true flight altitude of the aerial robot. Due to its auto-calibration feature and integration of the laser altimeter measurements along with the IMU measurements, the state estimator accurately estimates the barometer bias in order to estimate the accurate flight altitude of the aerial robot.
- *Navigation Phase:* As highlighted in Fig. 3.4, the navigation is performed over a

Figure 3.5: Results obtained during the execution of a complete flight, where take-off and land tasks are performed over an elevated ground surface (platform).

wide variety of obstacles in terms of size, height, 3D shape and surface. Despite the heterogeneity of the configuration, the state estimator is capable of providing a robust flight altitude estimate. Even in presence of obstacles composed of several uneven surfaces like chairs where the 1D laser altimeter measurements have higher noise, (see time instances between $t = 130$ s to $t = 140$ s and time instances $t = 150$ to $t = 160$ s in Fig. 3.4) the state estimator provides an accurate estimate of the flight altitude.

Through this experiment, we demonstrate that despite of presence of noise in all the individual sensor measurements at several time instances, the state estimator is able to estimate the accurate flight altitude in order to maintain the commanded flight altitude.

3.5.0.2 Take-off from the platform

- *Take-off from the platform:* Although the measurements provided by the 1D laser altimeter are with respect to the platform when the aerial robot is on top of the platform (see take-off phase in Fig. 3.5), the state estimator is able to deal with the initial height offset introduced by the platform, providing an accurate flight altitude estimate on top of the platform.
- *Navigation Phase:* When comparing the altitude estimated by the state estimator with the 1D laser altimeter altitude, it can be observed in Fig. 3.5 that, after time

instance $t = 40$ s as the aerial robot exits the platform the state estimator maintains the accurate estimate of the flight altitude even as the laser altimeter measurements are deflected from the platform to the flat ground surface, measuring with respect to the ground instead of the platform. After the aerial robot exits the platform, during the entire navigation phase it traverses over several ground obstacles and the flight altitude estimator maintains the accurate estimate of the true flight altitude of the aerial robot.

- *Land on the platform:* It can be highlighted through Fig. 3.5 that, during the approach stage the state estimator estimates accurately the height of the platform but due to high velocity and acceleration constraints during the fast landing stage, the estimated flight altitude drops to zero instead of the height of the platform. This error has no effect in any of the functionalities of the system, as landing being the last task to be executed during a mission.

Through this experiment we demonstrate the good and acceptable performance of the state estimator to estimate the flight altitude when taking-off and landing from elevated ground surfaces. We compare the estimated flight altitude with the altitude measured by the 1D laser altimeter over flat ground surfaces to achieve an average error less than 10 cm.

Additionally, our proposed state estimator was successfully used in IMAV 2016 indoors competition for solving the problem of vertical localization of the aerial robot as well as several industrial inspection scenarios due to the presence of uneven ground surfaces (see Chapter. 8).

3.6 Conclusions

In this chapter, we present a robust multi-sensor fusion state estimator for accurately estimating the flight altitude of a multirotor aerial robot as well as the elevation of the floor below it in highly unstructured indoors environments. Our proposed EKF-based state estimator has the following features: (1) Modularity in the number of sensors used, in order to fuse the information provided by variable number and type of sensors, including IMU sensors, barometer sensors and distance sensors, thus improving robustness; (2) Auto-calibration of sensors, which improves the accuracy of the state estimation; (3) Synchronous prediction of the estimated state whenever it is needed by the controller; and (4) Asynchronous update of the estimated state whenever a measurement is received.

Several real indoor flight experiments have been conducted in order to validate the proposed approach, with a minimal sensor setup of an IMU, a barometer and a 1D laser altimeter. We achieve an average error less than 10 cm when comparing the estimated altitude by the state estimator with the altitude measured by the 1D laser altimeter over flat ground surfaces. Finally, we release this algorithm as an open-source software² within Aerostack framework³, allowing the scientific community to use it for their experiments.

²https://bitbucket.org/zorana/altitude_filtering

³www.aerostack.org

In Chapter. 5 we focus to improve the altitude estimation for the aerial robots using downward facing light weight RGB-D sensor. The RGB-D sensor provides a richer information of the environment in the form of point cloud as compared to a 1D laser altimeter which provides information only about a single point. We focus on using plane segmentation method, for computing the planes from the point cloud data and mapping them in order to calculate the relative change in altitude with respect to the computed planes as well as estimating the true flight altitude of the aerial robot. The presented algorithm in this chapter can also be referred in Bavle et al., 2017.

Chapter 4

Pose Estimation using Semantic Stereo Visual Odometry

4.1 Introduction

Localization techniques using 2D or 3D laser sensors are more robust as compared to vision based localization techniques, as the laser sensors are more precise and robust against changing indoor environmental conditions such as illumination etc. However, 2D or 3D laser sensors are much heavier in weight as compared to cameras and require a larger sized aerial robotic platform, which is a clear disadvantage for narrow and cluttered indoor environments. Most of the VO/VIO localization techniques using lightweight RGB or RGB-D sensors, depend on low-level characteristic features from the environment such as points or lines. These techniques suffer from inherent limitations regarding view point dependency, adequate lighting conditions and repetitive patterns, deteriorating the feature matching step thus effecting the overall accuracy of the system.

Advances in faster and robust object detection and classification techniques have given rise to several semantic based localization techniques as mentioned in Chapter. 2. Low-level features along with high-level semantic features can be incorporated into such VO/VIO based localization systems, adding robustness during the feature matching step, overcoming to a great extent the view point dependency limitations. Although higher level semantic information ensures robustness in terms of data associations, extracting the relative position of the semantic objects for accurate localization can be a challenging task with noisy sensors, and especially in unstructured indoor scenarios where the 3D structure of even a same class of object may vary significantly.

Taking advantage of the lightweight property of RGB and RGB-D sensors and improving the localization by means of semantic objects, in this chapter we present a particle filter based localization algorithm for accurately estimating the pose of mini-aerial robotic platforms in unstructured and cluttered indoor environments. The prediction stage of the filter consists of our stereo VO algorithm which is accurate during short time intervals but suffers an accumulation of error in the long term. This accumulated error is corrected

Figure 4.1: The mini-aerial robotic platform used to validate the proposed approach, including its relevant frame of references.

in the update stage, in two phases using: (1) The Inertial Measurement Unit (IMU) data correcting the orientation obtained from the predicted pose. (2) The relative 3D position computed from the horizontal planar surface, associated with the corresponding semantic landmark in the map, updating the 3D predicted position of the aerial robot.

4.2 Stereo Visual Odometry

Visual odometry can be performed either by a monocular or a stereo setup, although in monocular cases it is difficult to estimate the true scale of the estimated trajectory and its performance degrades in presence of pure rotations. Stereo VO approaches overcome these problems, providing more reliable results due to the two cameras fixed at a known baseline. Hence the prediction stage of the particle filter consists of our previously developed stereo VO approach Manthe et al., 2018. Its main feature is the maximization of information gain by utilizing key points with and without depth information.

The algorithm extracts FAST key points from the stereo image pairs and filters them by means of bucketing (Zhang et al., 1995). To obtain key point correspondences, a key point matching between the key points of the left and the right camera image is applied, while the key points of the left image are tracked with the KLT-Tracker in order to obtain correspondences over time. Based on the matched key points the algorithm triangulates 3D points and deduces 3D to 2D correspondences. The motion estimation is divided into an initialization and a refinement stage:

- For motion initialization the algorithm extracts a relative pose from the essential matrix of two temporally consecutive images from the left stereo camera. However

direct computation of the correct scale of the translation is not possible. Hence it is estimated from 3D points with correct scale triangulated from key point matches and corresponding 3D points triangulated with the wrong scaled translation vector of the relative pose.

- The motion refinement is done with a bundle adjustment approach. In many cases it is not possible to match key points between the left and the right stereo image while they get tracked correctly in the left stereo image. Therefore a large number of key point correspondences without 3D information exist, which still contain information about motion. These can be exploited with the epipolar constraint, by computing the distance between an epipolar line and a corresponding key point. In addition for 3D points which were tracked correctly in the left image, the reprojection error is computed. Both errors get small if the estimated pose is correct. Therefore a sum of squared errors is minimized by means of the Levenberg-Marquardt algorithm.

4.3 Semantic Detection and Segmentation

4.3.1 Semantic Detection

For real-time detection of the semantic data we use the You Only Look Once (YOLO) object detector and classifier (Redmon and Farhadi, 2016), pre-trained on the VOC dataset (Everingham et al., 2015) providing in real-time a bounding box of the detected object along with its class and its probability. In our case, we configure the object detector to detect and classify only the required semantic data types. The semantic detection is performed on a single RGB camera. The detected bounding box is transformed into its corresponding depth image in order to extract the 3D point cloud data inside the bounding box.

4.3.2 Semantic Segmentation

The computed bounding box of the detected object can vary in size due to view-point changes, so the corresponding 3D point cloud data can include 3D points not relevant to the corresponding semantic data, e.g. floor points in the detections shown in Fig. 4.1. Hence, taking a median of all the 3D measurements of the points inside the bounding box cannot provide an accurate relative 3D position of the semantic object. In order to minimize the errors in the extracted relative position, we propose to segment the horizontal plane from the detected semantic data. We divide this technique into two parts:

Normal Based Clustering: In order to cluster the 3D planar surfaces we compute the point normals from the segmented 3D point cloud data. We use the integral normal estimation technique presented in Holzer et al., 2012. Using the orientation of the computed normals, we can cluster all the 3D planes present in the detected semantic data

based on their orientation. We use the k -means clustering algorithm which provides the normal centroids of all the available planar surfaces.

Horizontal Plane Segmentation: In accordance to the defined world reference frame W as shown in Fig. 4.1, the normal orientation of horizontal planes is always parallel to the z -axis. Using this rule, we segment the normal centroid of all the horizontal planes obtained previously, along with their corresponding 3D points. These segmented 3D points are used for computing the vertical height of the horizontal planes. We apply a second k -means clustering in order to cluster all the height centroids. As in our case, we use a horizontal plane just above the ground plane, its corresponding height centroid can be easily segmented along with its corresponding 3D points. A third k -means clustering is used for finding the closest centroid of the obtained 3D points.

This method ensures that the relative 3D position of the detected semantic data is extracted only when its horizontal plane is present, discarding all the unnecessary 3D points within the detected bounding box. Thus providing an accurate semantic cue for the update stage of the particle filter.

4.4 Particle Filter based Localization

In order to accurately fuse the stereo VO data presented in Sect. 4.2, the extracted relative 3D position of the semantic data from Sect. 4.3 as well as the inertial data, we use the non-parametric implementation of Bayes filter, known as the particle filter. We choose a particle filter over a standard Kalman filter implementation in order to incorporate different distributions from the detected semantic data in the future. We also select this approach because of its global localization capabilities and its capability of better handling non-linearities. The filter consists of M particles. Each state vector $\mathbf{x}^{[m]}$, where m is from 1 to M , models a state vector hypothesis comprising of $[x \ y \ z \ \theta \ \phi \ \psi]^T$, where x , y and z are the positions of the aerial robot in the x , y and z -axis respectively, in the world reference frame W, and θ , ϕ and ψ are the respective pitch, roll and yaw represented by Euler angles with respect to frame W. We assume that the initial state of the aerial robot is well known and thus the initial state vector of all the particles has the same value.

The particle filter also contains a map of the semantic landmarks of the environment in the form of $\mathbf{L} = \mathbf{L}_1, \dots, \mathbf{L}_n$. $\mathbf{L}_i = (\mathbf{l}_i^z, l_i^c)$, where $\mathbf{l}_i^z$ is the 3D position vector of the i -th landmark in the world frame of reference and l_i^c is the class of the corresponding semantic landmark. The filter presents the following stages:

4.4.1 Prediction

The prediction stage uses the 3D pose measurements of the aerial robot obtained from the stereo VO algorithm in the frame W. The state of each particle is propagated from the previous measurements $t - 1$ following a normal distribution as:

$$\mathbf{x}_t^{[m]} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{[m]}, \mathbf{u}_{s_t}); \quad (4.1)$$

where $\mathbf{x}_{t-1}^{[m]}$ is the pose of the m -th particle at time instant $t - 1$. $\mathbf{u}_s$ is the pose increment obtained from the difference between the stereo VO measurements between time instances $t - 1$ and t given as:

$$\mathbf{u}_{st} = \ominus \mathbf{x}_{st-1} \oplus \mathbf{x}_{st} \quad (4.2)$$

where $\mathbf{x}_{st-1}$ and $\mathbf{x}_{st}$ are the VO measurements received at time $t - 1$ and t respectively. The prediction stage of the filter runs whenever a new VO measurement is received and is based on the composition of each particle's estimate with the obtained increment given by Eq. 4.2.

4.4.2 Update

The update stage of the filter consists of three parts:

4.4.2.1 Update using the IMU measurements

The roll, pitch and yaw angles of the state predicted by the VO can accumulate error, especially in the absence of characteristic features in the environment. In order to overcome this error, we update the roll, pitch and yaw angles using the angles provided by the IMU. We calculate the importance factor for all the particles as follows:

$$w^{[m]} = \mu \exp(-(\phi_{diff} + \theta_{diff} + \psi_{diff})) \quad (4.3)$$

Where,

$$\begin{aligned} \phi_{diff} &= \frac{(\phi_m - \mu_\phi)^2}{2 \cdot \sigma_\phi^2} \\ \theta_{diff} &= \frac{(\theta_m - \mu_\theta)^2}{2 \cdot \sigma_\theta^2} \\ \psi_{diff} &= \frac{(\psi_m - \mu_\psi)^2}{2 \cdot \sigma_\psi^2} \end{aligned} \quad (4.4)$$

ϕ_m , θ_m and ψ_m are the predicted roll, pitch and yaw angles of each m -th particle respectively. μ_ϕ , μ_θ , μ_ψ are the roll, pitch and yaw angles measured by the IMU. σ_ϕ , σ_θ and σ_ψ being their respective standard deviations and μ a constant normalization factor based on their expected noise standard deviations. In the resampling stage, the roll, pitch and yaw angles of each particle are resampled according to this distribution of the calculated importance weights. This stage is executed only when the corresponding IMU data is available.

4.4.2.2 Update using the semantic data

The semantic data is received in the form of $\mathbf{S}_i = (\mathbf{s}_{r_i}^z, s_i^c, s_i^n, s_i^p)$, where $\mathbf{s}_{r_i}^z$ is the relative position of the i -th semantic object extracted from Sect. 4.3, s_i^c being its class type, s_i^n being the number of segmented 3D points and s_i^p being the detection probability. For an accurate data association, we divide the problem into two stages:

- First, for the given semantic data it is checked whether its class label s_i^c equals to any of the class labels of the mapped landmarks l^c and whether its probability s_i^p is higher than a certain threshold. It is further checked whether the number of obtained 3D points s_i^n for the semantic data is greater than a certain threshold.
- Second, if the detected semantic data satisfies the first stage, the relative 3D measurement vector $\mathbf{s}_{r_i}^z$ composed of x_{r_i} , y_{r_i} and z_{r_i} , is converted to the world frame of reference (see Fig. 4.1) as follows:

$$\begin{aligned} x_{w_i} &= x_a \oplus x_{r_i} \\ y_{w_i} &= y_a \oplus y_{r_i} \\ z_{w_i} &= z_a \oplus z_{r_i}. \end{aligned} \quad (4.5)$$

Where x_a , y_a and z_a are the average of the position measurements obtained from the prediction stage. In order to associate the semantic data with a mapped element, we compute the average difference between the semantic measurements in the world frame obtained from Eq. 4.5 with the already mapped measurements of the landmarks containing the same class type. If the difference is less than a given threshold, we select the minimum average difference in order to associate the current semantic data with the mapped data. This simplified data association strategy works well in our approach, since the semantic objects are quite far away from each other and the consecutive estimated aerial robot estimates do not have large uncertainties.

Once the detected semantic data is associated with its corresponding mapped landmark k , the importance factor of all the particles is calculated as:

$$w^{[m]} = \frac{\exp(-(x_{diff} + y_{diff} + z_{diff}))}{\sqrt{2 \cdot \pi \cdot \sigma^2}} \quad (4.6)$$

where,

$$\begin{aligned} x_{diff} &= \frac{(x_{w_i}^m - x_{w_k}^{map})^2}{2 \cdot \sigma^2} \\ y_{diff} &= \frac{(y_{w_i}^m - y_{w_k}^{map})^2}{2 \cdot \sigma^2} \\ z_{diff} &= \frac{(z_{w_i}^m - z_{w_k}^{map})^2}{2 \cdot \sigma^2} \end{aligned} \quad (4.7)$$

where $x_{w_i}^m$, $y_{w_i}^m$ and $z_{w_i}^m$ are the positions of the semantic data in the world frame calculated for each predicted m -th particle positions x^m , y^m and z^m obtained as:

$$\begin{aligned} x_{w_i}^m &= x^m \oplus x_{r_i} \\ y_{w_i}^m &= y^m \oplus y_{r_i} \\ z_{w_i}^m &= z^m \oplus z_{r_i} \end{aligned} \quad (4.8)$$

and $x_{w_k}^{map}$, $y_{w_k}^{map}$ and $z_{w_k}^{map}$ are the positions of the corresponding mapped semantic landmark k in the world frame. σ , is the standard deviation of the expected noise in the detected semantic data distance. Using the given distribution of the calculated importance weights, the positions of all the particles are resampled, converging to particle positions with higher importance factors. This stage is executed only when the semantic segmented data is available.

4.4.2.3 Incorporation of new map elements

New map elements are added in a Maximum Likelihood fashion. If the measurement of the semantic data belonging to a class type does not associate with the current mapped landmarks, due to the average difference being higher than the threshold, the corresponding semantic data is incorporated as a new map element along with its class type. The position of the new map element j is computed as:

$$\begin{aligned}x_{w_j}^{map} &= x_a \oplus x_{r_i} \\y_{w_j}^{map} &= y_a \oplus y_{r_i} \\z_{w_j}^{map} &= z_a \oplus z_{r_i}.\end{aligned}\tag{4.9}$$

4.5 Experiments and Results

4.5.1 System Setup

For a demonstration of the real flight experiments, the reader is advised to refer the following video¹. Fig. 4.1 shows the mini-aerial robot platform named Parrot Bebop-2, used for validating the performance of our proposed approach. We use the IMU on-board the Bebop-2 for the roll, pitch and yaw measurements. Additionally, we use two on-board sensors namely the Parrot-SLAM dunk and the Intel-RealSense R200 depth camera. The Parrot-SLAM dunk consists of a low performance ARM-architecture on-board computer along with two fisheye lens cameras. The RGB images from the two cameras, each with a resolution of 640×480 pixels, are utilized for the stereo VO algorithm (Sect. 4.2). The individual images are acquired at an approximate frequency of 15 Hz. Due to the computational limitations of the Parrot-SLAM dunk in computing a decent quality depth image without delay, we use a comparatively accurate Intel-RealSense connected to the Parrot-SLAM dunk computer, for performing the semantic detection and segmentation explained in Sect. 4.3. The data from the RealSense is received at 3 Hz. All the images used for computation are compressed and sent along with other required sensor data through WiFi connection between the Bebop-2 to an Intel Core i7-8700K on-ground computer. Our algorithm runs at a real-time frequency of 10 Hz. For performing autonomous flights, we embed it into our Aerostack software framework (refer Sanchez-Lopez et al., 2017a and Chapter. 7), which provides the position estimated by our algorithm to the position controller.

4.5.2 Results

In order to validate our approach, we perform several autonomous real flight experiments in cluttered and unstructured indoor environments. In this chapter, we only use a single type of semantic data i.e chairs, although this approach can be easily extended to another type of semantic data containing planar surfaces as elaborated in Chapter. 6. We evaluate our proposed algorithm in two different manners:

¹<https://vimeo.com/259349563>

(a)

(b)

Figure 4.2: Results obtained from autonomous real flight experiments of the aerial robot in a cluttered indoor environment. The numbered black boxes represent the order of the commanded 3D trajectory points. (a) presents the 3D plot with the detections from the yolo detector and (b) presents 2D plots for the ground truth evaluation experiment

(a)

(b)

Figure 4.3: Results obtained from autonomous real flight experiments of the aerial robot in a cluttered indoor environment. The numbered black boxes represent the order of the commanded 3D trajectory points. (a) shows the 3D plot with detections from the yolo detector and (b) represents 2D plot for the long trajectory evaluation experiment. The columns are obstacles for navigation, not included in the localization map.

Table 4.1: Absolute Trajectory Error for the compared algorithms.

Error	Our approach	Stereo VO	Stereo ORB-SLAM2
ATE [m]	0.18	0.85	0.39

4.5.2.1 Ground Truth Evaluation

Firstly, we evaluate the performance of our algorithm with an Optitrack ground truth system, in a $6\text{ m} \times 4\text{ m}$ indoor environment containing several columns, which are solely used for the purpose of adding complexity to the environment in the form of obstacles and are not used in any way by our algorithm. For these experiments we use 4 different types of chairs as semantic landmarks. Fig. 4.2 depicts the results obtained for the estimation of the trajectory followed by the aerial robot, compared with the ground truth trajectory. The aerial robot follows predefined 3D trajectories in a selected area at an average velocity of 0.2 m/s . In order to demonstrate the robustness of the estimated pose, the trajectory loop is performed twice with a total length of 27.32 m . The sequence of commanded trajectory points can be appreciated in Figures 4.2a and 4.2b. We compare our algorithm, with the state of the art ORB-SLAM2 stereo visual SLAM algorithm (Mur-Artal and Tardos, 2016). Although ORB-SLAM2 uses only a stereo image pair provided by the Parrot-SLAM dunk as compared to our method which additionally uses an Intel-RealSense, the aim of this comparison is to present degradation in the accuracy of methods depending on low-level characteristic features of the environment, as apposed to our method utilizing the combination of low-level and high-level semantic cues.

4.5.2.2 Long Trajectories Evaluation

The purpose of this experiment is to validate the performance of our algorithm when the aerial robot has to traverse long trajectories in larger spaces. We perform the experiment in a larger cluttered indoor environment of $8\text{ m} \times 7\text{ m}$, also containing similar obstacles as in the previous experiment and 4 chairs as semantic landmarks. The aerial robot performs the 3D trajectory loop twice, spanning an approximate trajectory length of 57.8 m at an average velocity of 0.3 m/s . Fig. 4.3a and Fig. 4.3b present the 3D and the 2D plots respectively of the executed long trajectory experiment. In this experiment, due to the larger coverage of the area the ground truth data is unavailable and the both the estimated trajectories by the VO as well as our approach are compared using the reference mission points commanded to the aerial robot with the same initial take-off and landing point.

4.6 Discussions

4.6.0.1 Ground Truth Evaluation

As shown in Fig. 4.2b, the stereo VO algorithm during the first trajectory loop has little error compared to the ground truth trajectory between points 1 and 3. Between points 3 and 4, as the aerial robot moves closer to a white wall representing a repetitive

pattern, the estimated pose begins to accumulate errors due to presence of less number feature points and wrong feature point matching. On the other hand, in our approach this accumulated error is corrected whenever a detection of a chair along with the 3D position of the segmented horizontal plane is received. As it can be observed in Fig. 4.2a, even though the environment contains several obstacles in the form of columns, the accurate position estimated by the algorithm enables the aerial robot to maintain the desired trajectory during the entire experiment.

ORB-SLAM2 maintains an accurate trajectory along the x and y -axis between trajectory points 1 and 4, whereas between points 3 and 4, consisting a repetitive pattern, the feature matching and tracking step using low-level characteristic features degrades in performance and the errors in the estimated positions increases (see Fig. 4.2a and Fig. 4.2b). Table. 4.1 compares the Absolute Trajectory Errors (ATE) (Sturm et al., 2012) obtained during the execution of this experiment, where our proposed approach obtains better results as compared to both the visual localization techniques depending only on the low-level characteristic features.

4.6.0.2 Long Trajectories Evaluation

It can be seen from Fig. 4.3a and Fig. 4.3b, that the aerial robot accurately reaches the commanded trajectory points and returns back close to the initial take-off point 1, whereas the pose estimated by the VO accumulates drift in its estimated pose. The VO pose estimates start diverging from the true pose estimates at the commanded point 3, while the aerial robot performs the first trajectory loop. This error is due to the presence of a featureless wall present at 8 m from the initial take-off point. This error accumulation in the VO pose can also be observed from Fig. 4.3b, while performing the second trajectory loop; according to the pose estimated by the VO algorithm, the aerial robot only reaches 4 meters, whereas our approach is able to correct this error, enabling the aerial robot to accurately reach the commanded point. Using only the VO pose estimates the aerial robot could not have been able perform the autonomous mission accurately due the it error accumulation. Through this experiment we can prove that our approach can enable an aerial robot with the capability of traversing larger distances without large accumulation of errors using only vision based sensors and few additional semantic objects, in challenging indoor environments.

4.7 Conclusions

In this chapter, we present a particle filter based localization approach for mini-aerial robots, using stereo VO and semantic information for cluttered and unstructured indoor environments. We present a novel technique to accurately estimate the relative position of the detected semantic objects in presence of noisy depth measurements, by segmenting the dominant horizontal planar surfaces, generalizing for several types of objects irrespective of their structure. We test our algorithm in such indoor scenarios using a common class of semantic data i.e chairs, of different shapes and sizes, providing the real-time pose information of the aerial robot at 10 Hz. We validate our approach by comparing the esti-

mated pose with the ground truth data as well as comparing its performance with the VO pose estimated and state of the art visual SLAM algorithm in challenging conditions. The code² provides the implementation of our presented approach. The presented algorithm in this chapter can be found in Bavle et al., 2018.

Chapter. 6, presents an improvement in the relative position estimation of the segmented semantic data by obtaining the convex hull of the surfaces and extracting all the planar surfaces available from a semantic object. Furthermore, we implement a full graph based semantic SLAM approach for aerial robots without using prior mapped landmarks, accurately estimating and optimizing the pose of the aerial robot and the semantic landmarks.

²https://bitbucket.org/hridaybavle/semantic_slam_workspace_old

Part III

**GEOMETRY AND OBJECT
BASED SLAM**

Chapter 5

Altitude Estimation and Planar Mapping using Point Cloud Sensors

5.1 Introduction

Chapter. 3 presents a flight altitude state estimator for aerial robots in indoor environments fusing sensors such as laser altimeters, barometers and accelerometers. Although fusing information from different sensors provides an accurate estimate of the flight altitude of the aerial robot as shown in the results of Chapter. 3, this approach can face limitations in presence of highly unstructured and complex shaped ground obstacles, which are not accurately perceived by a 1D laser or sonar altimeter.

Computer vision systems based on RGB and RGB-D cameras, on the other hand provide richer perception of the environment at a given time interval, in terms of an image as compared to a 1D laser altimeter perceiving only a single point below it. Chapter. 2 presents several vision based techniques used for estimating the 3D pose of the aerial robot including its altitude. However, computer vision based approaches face the following limitations; the monocular systems cannot recover the scale from the environment and their performance is effected in presence of high speed aerial robotic flights with high angular motions. Stereo and RGB-D based systems are computationally expensive due to the simultaneous processing larger size RGB and depth images and not all algorithms can guarantee high frequency estimation of the robot state. The estimations from the vision based systems are also degraded in presence of dynamic obstacles in the environment.

In order to better perceive and localize in the indoor environment using the computer vision systems, keeping into account their limitations, this chapter, presents an open source algorithm for rapid and robust estimation of the flight altitude of the aerial robot in presence of unstructured and dynamic ground obstacles, simultaneously mapping the horizontal planar surfaces, using a 3D point cloud sensor and an Inertial Measurement Unit (IMU). The proposed algorithm is capable of estimating the flight altitude and mapping the horizontal planes at a high frequency of upto 54 hz. To summarize, the following main contributions are proposed:

Figure 5.1: The custom aerial robotic framework used in the real experiments, depicting all the frame of references as well as the normal orientations of the horizontal planes in the world frame of reference.

- Fast 3D planar clustering and horizontal plane segmentation from 3D point cloud data.
- Kalman filter-based mapping of the segmented horizontal planes for accurate and high speed flight altitude estimation of aerial robots.
- Experimental validation of the proposed algorithm on-board the aerial robot, in the presence of several static and dynamic obstacles of different shapes and size.

5.2 3D Planar Clustering and Horizontal Plane Segmentation

In an indoor flight, normally, the flight altitude of a aerial robot is required to be computed with respect to the ground plane, where the world frame is located on the ground at the take-off point. The obtained 3D point cloud information from a 3D sensor, can be used to extract the 3D surface information of the objects present in the environment. Hence, a 3D point cloud sensor partially inclined downwards, can detect the ground surface as the most prominent 3D planar horizontal surface. Clustering the 3D points relevant to the ground surface and segmenting the 3D planar information can provide us with an estimate of the flight altitude of the aerial robot with respect to the ground surface.

In presence of ground obstacles, the prominence of the ground surface decreases due to occlusions. In cases of the ground surface being completely occluded, the estimated flight altitude will be inaccurate, as it will be relative the ground obstacle, occluding the ground surface. In order to overcome this problem, we propose to first cluster and segment all

Figure 5.2: Chair as a ground obstacle used in the real flight experiments with all the extracted normals. The normals within the red colored area represent are the horizontal plane clusters and the ones within the green-colored area represent the vertical plane clusters.

the horizontal planes present at a given time instant. Then, we continuously map the segmented planes in order to accurately estimate the flight altitude of the aerial robot.

5.2.1 Normal Based Clustering

In order to cluster 3D planar surfaces, first we compute the point normals at each 3D point. We use the integral image normal estimation technique as explained in Holzer et al., 2012, for fast and real-time normal estimation.

An integral image $\mathbf{I}_i$ of an image $\mathbf{I}_m$ is the sum of all values of $\mathbf{I}_m$ within a particular region $r(m, n)$. In brief, this technique first converts the z -component of the point cloud into an integral image, thus requiring an organized 3D point cloud. This reduces the computation cost and makes it independent of the size of the smoothing windows used to reduce the noise in the 3D point cloud data. A smoothing is then performed choosing appropriate smoothing areas calculated using depth change maps. The normal $\mathbf{n}_p$ at a point $\mathbf{p}$ is then calculated as the cross product of two vectors: $\mathbf{n}_p = \mathbf{p}_h \times \mathbf{p}_v$, where $\mathbf{p}_h$ is a vector between 3D right and left neighbors of the point $\mathbf{p}$ and $\mathbf{p}_v$ is a vector between 3D upper and lower neighbors of point $\mathbf{p}$, in the organized point cloud.

The orientation of these computed normals is used for the clustering all the 3D planes present in the environment. We cluster all the computed normals using a K -means clustering algorithm, which provides the normal centroids of all the present 3D planes including all the horizontal planes. These clustered centroids of the normals constitute the matrix $\mathbf{N}_p^c$. Fig. 5.2 shows an example containing a ground obstacle as seen from the sensor on-board the aerial robot, with all the computed normals with their orientations and their

respective clusters.

5.2.2 Horizontal Plane Segmentation

As shown in Fig. 5.1, the normal centroids computed from the above mentioned Sect. 5.2.1 are with respect to the sensor frame of reference F_c . In accordance to the definition of the world reference system, the normals of the horizontal planes are always parallel with respect to the z -axis of the world reference frame, which is used for segmenting the horizontal planes. Converting all the normal centroids from sensor frame to world frame becomes a computationally expensive and time-consuming process. In order to speed up the computation, instead of converting all the computed normal centroids to the world frame, we convert the orientation of a single horizontal ground plane normal in the world frame, which is always known and fixed, into the sensor frame as:

$$\mathbf{n}_p^c = (\mathbf{T}_c^w)^T \cdot \mathbf{n}_p^w \quad (5.1)$$

$\mathbf{T}_c^w$ being the transformation of the sensor frame F_c to the world frame F_w . $\mathbf{T}_c^w$ is a composition of the transformation the of sensor frame F_c to robot frame F_r given by $\mathbf{T}_c^r$ and transformation of the robot frame F_r to world frame F_w given by $\mathbf{T}_r^w$:

$$\mathbf{T}_c^w = \mathbf{T}_c^r \oplus \mathbf{T}_r^w \quad (5.2)$$

The computed normal centroids of the 3D planes, comprising the matrix $\mathbf{N}_p^c$ from Sect. 5.2.1, are compared with computed horizontal ground plane normal in the sensor frame, $\mathbf{n}_p^c$ from Equation (5.1). Using a given threshold, the normal centroid of the horizontal planes present in the normals matrix $\mathbf{N}_p^c$ is segmented. From this segmented normals, all their corresponding 3D points of all the horizontal planes are segmented. The segmented 3D points along with ground plane normal in the sensor frame $\mathbf{n}_p^c$, are used for computing the vertical height of the segmented horizontal planes with respect to the sensor as follows:

$$h = \mathbf{p} \cdot \mathbf{n}_p^c \quad (5.3)$$

where $\mathbf{p}$ is a 3D point composed of p_x, p_y, p_z in Cartesian coordinates of x, y , and z axis respectively and $\mathbf{n}_p^c$ composed of the three components $n_{p_x}^c, n_{p_y}^c, n_{p_z}^c$.

A second K -means algorithm is used for clustering the height vector $\mathbf{h}$, which is a vector of all the computed heights h from Equation (5.3). The height centroids obtained from the second K -means clustering are used by the localization and mapping stage as explained in the following Sect. 5.3, in order to accurately estimate the flight altitude of the aerial robot. Algorithm 1 explains briefly the implementation of this section.

Algorithm 1: 3D Planar Clustering and Horizontal Plane Segmentation

Input: Point cloud of length l ($m1 \times m2$)**Output:** Height centroids $\mathbf{h}_{centroids}$ from the segmented horizontal planes

```

1 for  $i = 1$  to  $l$  do
2   |  $\mathbf{N} \leftarrow$  compute normal  $\mathbf{n}_i$  at point  $\mathbf{p}_i$ ;
3 end
4  $\mathbf{N}_{centroids} \leftarrow$   $K$ -means clustering of  $\mathbf{N}$ ;
5  $\mathbf{n}_{centroid} \leftarrow \mathbf{N}_{centroids}$  closest to the orientation of the ground plane normal  $\mathbf{n}_p^c$ ;
6 for  $i = 1$  to  $l$  do
7   | if  $(\mathbf{n}_i - \mathbf{n}_{centroid}) \leq thres$  then
8     |   |  $\mathbf{h} \leftarrow \mathbf{p}_i \cdot \mathbf{n}_p^c$ ;
9     | else
10    |   | continue;
11    | end
12 end
13  $\mathbf{h}_{centroids} \leftarrow$   $K$ -means clustering of  $\mathbf{h}$ 

```

5.3 Horizontal Plane-Based Localization and Mapping

In order to deal with the problem of multiple horizontal planes, we formulate a Kalman filter based localization method to estimate the flight altitude of the aerial robot based on mapping all the segmented horizontal planes for generating a complete map.

We consider a Kalman filter with the state $x \in \mathbb{R}$, comprising of aerial robot state component x_r . It consists of a noise $n \in \mathbb{R}$ comprising of the robot's process model noise n_r .

- $t_{z_r}^w$, the estimated height of the aerial robot in the world reference frame.

5.3.1 Process Model

The discrete time process model of the robot's estimated height can be given as:

$$t_{z_r}^w(k) = t_{z_r}^w(k-1) + n_z \quad (5.4)$$

where n_z is the Gaussian white noise in the vertical component process model and is the only component of the robot's process model noise n_r .

5.3.2 Prediction Stage

The prediction stage follows the standard Kalman filter equations, where the state x is estimated in the current time instant k , using the previous estimate at time $k-1$.

$$\hat{x}(k|k-1) = \hat{x}(k-1|k-1) \quad (5.5)$$

$$\tilde{\sigma}_x^2(k|k-1) = \hat{\sigma}_x^2(k-1|k-1) + \hat{\sigma}_q^2 \quad (5.6)$$

where σ_x^2 and σ_q^2 are the process and noise variance.

5.3.3 Measurement Model

The height centroids obtained from the horizontal plane segmentation explained in Sect. 5.2, form the vector of measurements $\mathbf{z}_p$ as:

- Distance measured to the ground object z_p .

And its measurement model $\mathbf{h}_p$ being:

$$\hat{z}_p = t_{z_r}^w - z_p \quad (5.7)$$

5.3.4 Data Association and Update Stage

The estimated horizontal plane centroids from Equation (5.7), are compared with the ones already in the map vector $\mathbf{z}_{map}$ by means of the Mahalanobis distance. It is assumed that the ground plane is always present at the lowest location of the map, in the world reference frame. Hence the map vector $\mathbf{z}_{map}$, initially contains at least the ground plane as a horizontal plane with height zero. For a given measurement $\hat{z}_{pn}$, the measurement innovation is computed as:

$$v_i = z_{map_i} - \hat{z}_{pn} \quad (5.8)$$

where i ranges from 1 to m , and m being the number of map elements in $\mathbf{z}_{map}$. The associated innovation variance can be given as:

$$\sigma_s^2 = \tilde{\sigma}_x^2(k|k-1) + \tilde{\sigma}_r^2 \quad (5.9)$$

where $\tilde{\sigma}_r^2$ is the associated variance of the current plane measurement $\hat{z}_{pn}$.

The Mahalanobis distances, between the map elements and current measurement are computed as:

$$d_i = \sqrt{v_i^T \cdot \sigma_s^2 \cdot v_i} \quad (5.10)$$

All the distances within the given chi-square threshold are sorted in order to obtain the minimum value, and thus the corresponding map element. This selected map element is then associated with the measurement $\hat{z}_{pn}$.

If all the computed Mahalanobis distances are above the given threshold, the plane element $\hat{z}_{pn}$ is added as a new map element.

All the n matched plane elements at a given time instance, constitute the innovation vector $\mathbf{v}_n \in \mathbb{R}^n$ as well as the noise covariance matrix $\mathbf{R}_n \in \mathbb{R}^{n \times n}$. Thus, the new innovation covariance matrix given as:

$$\mathbf{S}_n = \mathbf{H}_{x_n} \cdot \tilde{\sigma}_x^2(k|k-1) \cdot \mathbf{H}_{x_n}^T + \mathbf{R}_n \quad (5.11)$$

where $\mathbf{H}_{x_n} \in \mathbb{R}^n$ is the Jacobian of the measurement model of the extracted planes with respect to the state x . $\mathbf{H}_{x_n}$ in our case is a column vector, with all elements equal to 1.

After which, the near optimal Kalman gain $\mathbf{K} \in \mathbb{R}^{1 \times n}$ is calculated as:

$$\mathbf{K} = \tilde{\sigma}_x^2(k|k-1) \cdot \mathbf{H}_{x_n}^T \cdot \mathbf{S}_n^{-1} \quad (5.12)$$

Finally, the corrected values of the state and its covariance are obtained by:

$$\hat{x}(k|k) = \tilde{x}(k|k-1) + \mathbf{K} \cdot \mathbf{v}_n \quad (5.13)$$

$$\hat{\sigma}_x^2(k|k) = (1 - \mathbf{K} \cdot \mathbf{H}_{x_n}) \cdot \tilde{\sigma}_x^2(k|k-1) \quad (5.14)$$

Algorithm 2 provides a summary of the above explained process.

5.4 Experiments and Results

5.4.1 Standard Datasets

In order to evaluate and validate our approach, we first test it with a standard dataset provided by the research community, estimating the camera altitude using the provided point cloud data.

5.4.1.1 IROS 2011 Kinect Dataset

This dataset ¹ consists of point cloud data obtained from a Kinect RGB-D sensor and ground truth data from a Motion Capture (MoCap) system in indoor environment. The data is captured in three distinct indoor environments with increasing complexity of ground obstacles (see Fig. 5.3) and with three different types of motions performed in slow, medium and fast speeds.

We provide the results for the translation motion of the camera, applied at all three mentioned speeds in low, medium and high complexity ground obstacles. The data consists of 3D point cloud of resolution 160×120 at 30 Hz and provides a transformation between the camera frame and the world frame, with our algorithm running at the same frequency of the point cloud data acquisition (30 Hz). For these experiments, we choose initial clusters in both the K -means algorithms mentioned in Sect. 5.2 based on experimental trials and assign it a numerical value of 3.

Fig. 5.4 and Fig. 5.5 elaborate the results of our algorithm in terms of graphs comparing it with the ground truth altitude information obtained from the MoCap system. Fig. 5.4a–c describe the plots obtained during a slow motion of the camera in low, medium and high complexity environments respectively. The algorithm performs robustly in slow motion even in the presence of high level of complexity of cluttered environment achieving an average RMSE of 0.0533 m. Increasing the speed of the translation motion of the camera decreases the accuracy of the algorithm, with the medium speed motions (see Fig. 5.4d–f) achieving an average RMSE of 0.0629 m. Whereas the high-speed motion of the camera results in an average RMSE of 0.1272 m as seen in Fig. 5.5a–c. Table. 5.1 represents the individual RMS errors obtained during the execution of each experiment.

¹<http://projects.asl.ethz.ch/datasets/doku.php?id=kinect:iros2011kinect>

Algorithm 2: Horizontal Plane-based Localization and Mapping**Input:** Height centroids $\mathbf{h}_{centroids}$ and current map vector of the horizontal planes $\mathbf{z}_{map}$ **Output:** Estimated flight altitude $\hat{x}$

```

1
2 Prediction stage:
3
4 predicted flight altitude  $\tilde{x}(k|k-1) \leftarrow \hat{x}(k-1|k-1)$ ;
5 predicted process covariance  $\tilde{\sigma}_x^2(k|k-1) \leftarrow \hat{\sigma}_x^2(k-1|k-1) + \hat{\sigma}_q^2$ ;
6
7 Data association and update stage:
8
9 for  $i = 1$  to  $length(\mathbf{h}_{centroids})$  do
10
11   Data association stage:
12
13   for  $j = 1$  to  $length(\mathbf{z}_{map})$  do
14     mahalanobis distance  $\mathbf{d} \leftarrow d_j = \sqrt{v_j^T \cdot \sigma_s^2 \cdot v_j}$ ;
15   end
16    $d_k = \min(\mathbf{d})$ 
17   if  $d_k \leq threshold$  then
18     associate current horizontal plane  $\mathbf{h}_{centroids_i}$  with the corresponding mapped
19     element of  $\mathbf{z}_{map}$ ;
20     data association successful  $\leftarrow$  true;
21   else
22     data association successful  $\leftarrow$  false;
23     continue;
24   end
25
26   Update stage:
27
28   if data association successful then
29     innovation covariance  $\mathbf{S}_n \leftarrow \mathbf{H}_{x_n} \cdot \tilde{\sigma}_x^2(k|k-1) \cdot \mathbf{H}_{x_n}^T + \mathbf{R}_n$ ;
30     Kalman gain  $\mathbf{K} \leftarrow \tilde{\sigma}_x^2(k|k-1) \cdot \mathbf{H}_{x_n}^T \cdot \mathbf{S}_n^{-1}$ ;
31     updated flight altitude  $\hat{x}(k|k) \leftarrow \tilde{x}(k|k-1) + \mathbf{K} \cdot \mathbf{v}_n$ ;
32     updated process covariance  $\hat{\sigma}_x^2(k|k) \leftarrow (1 - \mathbf{K} \cdot \mathbf{H}_{x_n}) \cdot \tilde{\sigma}_x^2(k|k-1)$ ;
33   else
34     add the current horizontal plane  $\mathbf{h}_{centroids_i}$  as a new map element to  $\mathbf{z}_{map}$ ;
35   end
36 end

```

Figure 5.3: Indoors Environment used in IROS 2011 dataset with increasing complexity of ground obstacles from left to right. (a) Low complexity environment; (b) Medium complexity environment; (c) High complexity environment.

Complexity \ Speed	Slow	Medium	Fast
	Low	Medium	High
Low	0.046	0.07	0.14
Medium	0.0424	0.0517	0.1218
High	0.0714	0.067	0.1197

Table 5.1: Individual RMS errors obtained during the execution of IROS2011 kintect dataset experiments.

5.4.2 Real Flights

Video² presents the visual demonstration of the real flight experiments. The aim of the real flight experiments, on one hand is to evaluate the capabilities of our proposed approach in presence of unknown, cluttered and dynamic ground obstacles, and on the other hand, to evaluate the robustness of the estimation with fast and rapid movements of the aerial robot. Sect. 5.4.2.1 explains briefly the setup of our aerial robotic system as well as the configuration of the indoors real flight scenario. After which, Sect. 5.4.2.2 presents the results obtained during the performed autonomous indoor missions.

5.4.2.1 Experimental Setup

System Setup: Fig. 5.1 shows the custom aerial robotic platform, used for validating the proposed algorithm. The autopilot used for low-time control (attitude control) is Pixhawk³. The Pixhawk consists of an integrated IMU sensor, providing the angular velocities and linear accelerations in all 3-axis. We use on board a Intel RealSense Z200 RGB-D⁴ camera for real-time computation of the 3D point cloud. The RGB-D camera is placed at a fixed angle of -50° with respect to the y -axis of the robot frame (see Fig. 5.1), not pointing perpendicular to the ground surface in order that the camera can be used

²<https://vimeo.com/235911971>

³<https://pixhawk.org/>

⁴<https://software.intel.com/en-us/articles/realsense-r200-camera>

Figure 5.4: Comparison of our algorithm with the MoCap ground truth data from the IROS 2011 dataset for slow and medium camera speed scenario. (a) Low complexity and slow speed scenario; (b) Medium complexity and slow speed scenario; (c) High complexity slow speed scenario; (d) Low complexity medium speed scenario; (e) Medium complexity medium speed scenario; (f) High complexity medium speed scenario.

Figure 5.5: Comparison of our algorithm with the MoCap ground truth data from the IROS 2011 dataset for high camera speed scenario. (a) Low complexity fast speed scenario; (b) Medium complexity fast speed scenario; (c) High complexity fast speed scenario.

Figure 5.6: Real flights performed in an indoor environment with different configuration of the ground obstacles. (a) Static ground obstacles environment with each number representing a ground obstacle; (b) Dynamic ground obstacles environment with each number representing a ground obstacle. Numbers 2, 3, 4 and 5 represent dynamic obstacles.

for other future autonomous mission tasks such as object detection, avoiding the need for a dedicated sensor for altitude estimation task. All the computation is carried on board the aerial robot on an Intel-NUC 6i5SYK mini-computer.

We use an Optitrack MoCap system⁵, providing the ground truth flight altitude of the aerial robot, in order to compare and evaluate our estimated flight altitude. An on-board Lightware SF/10 laser altimeter is also used, only to demonstrate its inaccuracies in presence of ground obstacles.

The software modules are developed in C++ and Robot Operation System (ROS). For a complete autonomous flight the Aerostack architecture (refer Sanchez-Lopez et al., 2017a and Chapter. 7) is used, integrating our newly developed software modules of the proposed altitude estimation approach.

Real Flight Configuration: Several autonomous flights are performed in different challenging scenarios in order to validate our approach as shown in Fig. 5.6.

- *Navigation in unknown and unstructured indoor environment:* (see Fig. 5.6a) This configuration consists of an indoor area 4 m × 4 m in dimensions. The environment includes several static ground obstacles, with their height lower than the flight altitude of the aerial robot, placed in an unknown and random manner. The ground obstacles include several chairs with random discontinuity in their surface as well as tables and other objects with uniform surfaces. It also consists of a static iRobot⁶ with a box on top as a ground obstacle.
- *Take-off and land on a platform and navigation in dynamic, unknown and unstructured indoor environments:* (see Fig. 5.6b) Similar to the previous evaluation sce-

⁵<http://optitrack.com/>

⁶<http://www.irobot.com/About-iRobot/STEM/Create-2.aspx>

nario, this configuration consists of an indoor area $4\text{ m} \times 4\text{ m}$ in dimensions, consisting of several static as well as dynamic ground obstacles. The static ground obstacles are the same as mentioned in the previous configuration. In order to increase the complexity, we add several dynamic and random ground obstacles, placing different boxes of varying shapes and height on top of iRobots. The iRobots move in a complete random and unknown manner, thus providing random dynamic obstacles. The aerial robot takes-off and lands on a platform of height 0.47 m and area 1.4 m^2 , further increasing the complexity of the mission.

5.4.2.2 Results

In this section we present the results obtained from performing autonomous flights in the previously mentioned scenarios. The navigation controller uses as feedback the flight altitude estimated by the proposed algorithm.

The aerial robot is always commanded to maintain a constant flight altitude of 1.5 m . The flight altitude estimation is performed at an average real-time frequency of 54 Hz , with the point cloud being received at 60 Hz . The point cloud resolution is 160×120 . In all the experiments, the aerial robot performs an autonomous but pre-defined squared trajectory over the mentioned area. The initial clusters in both the K -means algorithms mentioned in Sect. 5.2 are chosen through experimental evaluations and assigned a numerical value of 3. The squared trajectory is performed twice, in order to validate the performance of the mapping stage of our proposed algorithm as explained in Sect. 5.3.

- *Navigation in unknown and unstructured indoor environment:* In this experiment, the aerial robot performs the autonomous squared trajectory with two different velocities, one at lower velocity of 0.3 m/s and the other at a higher velocity of 1.0 m/s in the static ground obstacles environment.
 - *Low Velocity Flight:* Fig. 5.7a represents the results of the low velocity (0.3 m/s) autonomous flight of the aerial robot. It can be seen from Fig. 5.7a, when comparing the estimated altitude with the ground truth altitude we get an RMS error of 0.079 m .
 - *High Velocity Flight:* Fig. 5.7b represents the results obtained from the high velocity autonomous flight of the aerial robot. The RMS error achieved is 0.11 m , when compared to the ground truth altitude.
- *Take-off and land on a platform and navigation in dynamic, unknown and unstructured indoor environments:* In these experiments, we perform autonomous low (0.3 m/s) as well as high (1 m/s) velocity trajectories in dynamic and unstructured environment. The aerial robot takes-off and lands on a platform and as the height of a platform is well known, we predefine the initial height of the aerial robot as the height of the platform. The initial map of the horizontal height centroids contains the height of the platform plane along with the ground plane.

- *Low velocity flight:* Fig. 5.8a represents the results obtained during the low velocity flight of 0.3 m/s. As it can be seen from the figure, during the take-off phase ($t = 0$ s to $t = 22$ s) the altitude measured by the laser altimeter is inaccurate as it is with respect to the platform. Whereas the altitude estimated by our algorithm is accurate even though the ground plane is not detected most of the time during this phase. After the take-off phase the aerial robot navigates through the dynamic environment (see time $t = 22$ to $t = 85$ in Fig. 5.8a), consisting of several moving ground obstacles as well as static ground obstacles. We achieve an RMS error of 0.037 during this proposed experiment.
- *High velocity flight:* Similar to the previously mentioned low velocity experiment, we perform the autonomous mission at a higher velocity of 1 m/s. We achieve an RMS error of 0.0424 m when comparing the estimated altitude with the ground truth altitude.

5.5 Discussion

5.5.1 Standard Datasets

As presented in Sect. 5.4.1 our algorithm accurately estimates the correct camera altitude with very low RMS error, as it correctly segments and maps the horizontal planes, in presence of several ground obstacles irrespective of the complexity of their shape and size (see Fig. 5.4a–f and Table 5.1) at slow as well as medium speed camera translation motions. During high speed camera translation motions, at several time instances, there is absence of horizontal planes in the point cloud data adding errors in the estimation (for example in Fig. 5.5c between time instances $t = 0$ to $t = 10$). Nevertheless, as the algorithm maps all the horizontal planes (Sect. 5.3), it recovers the correct altitude estimation when the horizontal planes appear again in the point cloud data, matching them appropriately with the already mapped planes.

In the experiments, the camera motion is performed in a completely random manner and the new ground obstacles added for complexity are non-planar, smaller in size just occluding to some amount the present horizontal planes, thus they neither increase or decrease the accuracy of the algorithm. Because of the above reasons for a given speed of the camera the RMS errors do not follow a pattern (see Table. 5.1), and are based on the random translation motions performed. But it can be clearly seen from Table. 5.1 that, as the speed of these translation motions increases, due to the fact that at several time instances the horizontal planes are absent in the point cloud data, the RMS errors increases.

5.5.2 Real Flights

As it can be seen from the Sect. 5.4.2.2, our algorithm performs quite robustly on-board the aerial robot flights, in presence of high complexity shaped dynamic and static ground obstacles.

(a)

(b)

Figure 5.7: Real flight performed in static environment at different velocities of the aerial robot. (a) Low velocity flight in static environment where each number represents the laser measurement errors for each obstacle; (b) High velocity flight in static environment with each number represents the laser measurement errors for each obstacle.

(a)

(b)

Figure 5.8: Real flight performed in dynamic environment at different velocities of the aerial robot. (a) Low velocity flight in dynamic environment where each number represents the laser measurement errors for each obstacle; (b) High velocity flight in dynamic environment where each number represents the laser measurement errors for each obstacle.

It can be appreciated from Fig. 5.7 **a-b** in presence of static ground obstacles, with low as well as high velocities of 0.3 m/s and 1 m/s, our proposed algorithm efficiently estimates the flight altitude when comparing it with the ground truth altitude. It can be seen from Fig. 5.7 **a** between time instances $t = 20$ s and $t = 30$ s and time instances $t = 30$ s and $t = 40$ s, that the laser altimeter measurements have several peaks which are caused due to obstacles numbered 4 and 7. Obstacle number 4 is a chair with an asymmetric shape which causes the laser measurements to get deflected causing such peaks. Whereas in case of obstacle number 7 due to the speed of the aerial robot, laser gets deflected at the edge of the obstacle causing such large peak. Due to these errors in the measurements, laser altimeter measurements cannot be used directly for closing the altitude control loop of the aerial robot. A heuristic approach for calculating the flight altitude of the aerial robots, based on filtering the laser measurements can accumulate large errors and hence would not work accurately in presence of such obstacles either. Whereas in our proposed approach, as we segment and map the 3D horizontal planes, the complexity of the shape of the ground obstacles does not affect the accuracy of the flight altitude estimate of the aerial robot as seen from Fig. 5.7 **a-b**. These laser measurement errors can also be seen in Fig. 5.7 **b** between time instance $t = 30$ s and $t = 40$ s as the laser measurements are deflected by the obstacle number 4, and estimation from our proposed approach remains unaffected.

The flight altitude estimation in presence of dynamic obstacles can be appreciated from Fig. 5.8 **a-b**. During the take-off phase in presence of obstacle number 1, the laser measurements are with respect to the obstacle 1 whereas our algorithm having the input height of the obstacle 1, estimates the correct flight altitude of the aerial robot during the entire take-off phase. During the entire navigation stage due to the presence of several random moving dynamic as well as static obstacles, several laser measurements get referred to the obstacles instead of the ground, making it impossible to use such measurements for estimating the flight altitude of the aerial robot. As our approach rapidly maps all the present horizontal planes, it can accurately estimate the flight altitude of the aerial robot in such environment at both slow as well as high velocities of 0.3 m/s and 1 m/s respectively. It can also be appreciated from Fig. 5.8 **a-b** that as our algorithm maps and localizes all the present horizontal planes, random and dynamic nature of the ground obstacles has no effect on the performance of our algorithm.

It can be seen from Sect. 5.4.2.2 that the average RMS errors in presence of dynamic obstacles is less as compared to static obstacles, this is because in the dynamic scenario the number of ground obstacles with horizontal planes is greater as compared to static scenario. Higher number of ground obstacles thus provide higher number of horizontal planes for the localization and mapping stage (Sect. 5.3), improving the accuracy of the algorithm. Even though the obstacles have dynamic random motion, our algorithm maps them based on their horizontal plane height and matches them accurately when they are re-encountered, making it robust against dynamic motion of the ground obstacles, and thus improving the accuracy of the estimated flight altitude of the aerial robot.

It should be noted that even though in all the performed experiments the commanded flight altitude is kept constant to 1.5 m in order to compare the algorithm with the ground truth data and the laser measurements, our algorithm can perform equally at

different commanded flight altitudes during the execution of the mission, provided that the commanded altitudes are within the minimum and maximum ranges of the point cloud sensor.

5.5.3 Limitations

Our proposed approach currently faces a limitation that, during the absence of any horizontal plane, the estimated flight altitude can have inaccuracies. These errors can be resolved fusing more sensors such as accelerometers, barometers etc., as presented in Chapter 3. As this research is focused on presenting the flight altitude estimation results of an aerial robot using point cloud sensors, we currently do not fuse other sensor measurements.

This limitation can be clearly pointed out in the Sect. 5.4.1.1, while performing very fast camera motions, the computed point cloud lacks information of any horizontal plane, during several time instances. This can be observed in Fig. 5.5c, between time instance $t = 0$ s and $t = 10$ s. During such time instances, the Kalman filter in our proposed approach, propagates the state using the constant velocity noise model (see Sect. 5.3). This causes the increase in the RMS error with increasing motion of the camera, whereas the complexity of ground obstacles in the environment does not affect the accuracy our proposed approach. This limitation can also be pointed out in real flight experiments from Fig. 5.7a-b, as the estimated altitude during the landing phase is not accurately estimated when compared with the ground truth altitude. This is because during the landing phase the Intel RealSense sensor is very close to ground surface, unable to provide any reliable point cloud information and hence no horizontal plane is extracted.

Even though suffering from this above mentioned limitation, in a real flight with RGB-D camera on board the aerial robot, inclined downwards and with fast motions up to 1 m/s, the computed point cloud always contains at least one horizontal plane. In addition, as we map all the horizontal planes, the performance of our proposed approach is not affected when testing with real and fast flights except during the landing phase, which usually does not affect performance the autonomous mission, as it being the last and concluding phase of the mission (see Sect. 5.4.2.2).

Our proposed approach can map the ground obstacles only below the flying altitude of the aerial robot. With the current sensor setup of our aerial robot used in the experiments, if there is a ground obstacle taller than the flight altitude, it will be discarded by our algorithm as no horizontal plane would be detected. Whereas, during an autonomous mission such ground obstacles would be detected and avoided using the 2D laser sensor on board the aerial robot. Obstacle detection and avoidance using the 2D laser can be found in our work in Chapter. 8 and Sampedro et al., 2018.

5.6 Conclusions

In this chapter, we present a fast and robust point cloud-based altimeter, accurately estimating the flight altitude of aerial robots, in presence of several static as well as dynamic ground obstacles, in indoor environments.

Our proposed approach is based on fast segmentation of 3D planar surfaces from a 3D point cloud data computed from any point cloud sensor. We cluster the obtained 3D planar surfaces in order to segment the 3D horizontal planar surfaces. We map the vertical distances of the 3D horizontal planar surfaces, in order to provide a robust estimation of the flight altitude of the aerial robot. Our algorithm works at a fast real-time frequency of 54 Hz, with the point cloud acquisition frequency of 60 Hz.

Several experiments have been performed in order to validate our approach. We first compare our results with IROS 2011 standard Kinect dataset obtaining very low RMS errors when compared to the ground truth data. We also perform several real indoors flights in presence of several static as well as dynamic obstacles, achieving an RMS error as low as 0.037 m, when comparing it with the ground truth data. We also release our work as an open-source software⁷ within our Aerostack framework⁸, for the research community to benefit for their experiments. The presented algorithm in this chapter can also be referred in Bavle et al., 2018

In Chapter. 6 we extend our approach to robustly estimate the x -axis and y -axis position of a aerial robot along with the robust flight altitude estimation using the point cloud sensors, in dynamic and unstructured indoors environment. Since obtained the point cloud, is generally very noisy in the x -axis and y -axis direction, segmenting 3D planar surfaces in these directions and achieving robust localization becomes a challenging task, hence we use a fusion of VO/VIO approaches with semantic based planar extraction and mapping, to simultaneously estimate 3D pose of the aerial robot and create a map of the planar semantic landmarks.

⁷https://bitbucket.org/hridaybavle/multiplane_localization

⁸www.aerostack.org

Chapter 6

Pose and Map Estimation using Visual Semantic SLAM

6.1 Introduction

Many indoor autonomous missions using aerial robots, limit the usage of smaller sized aerial robots, capable of navigating narrow constrained spaces. Smaller sized aerial robots require light weight sensors such as RGB or RGB-D cameras. Many of the semi-dense SLAM techniques like Mur-Artal and Tardos, 2016, Engel et al., 2014, Trevor et al., 2012, rely on low-level characteristic features of the environment such as points, lines and planes, which deteriorate in performance in presence of illumination changes and repetitive patterns, whereas some other state of the art dense SLAM based techniques like Newcombe et al., 2011a, Whelan et al., 2015, Keller et al., 2013 perform dense 3D mapping of the environment, requiring high end CPU and GPU in order to achieve real-time performance, which makes them difficult to run on-board an aerial robot with limited computational resources.

As presented in Chapter. 4, recent improvements in computer vision algorithms have made it possible to achieve object based detectors running real-time on the lower end CPUs or GPUs. Combining such detectors with Visual Odometry (VO)/ Visual Inertial Odometry (VIO) systems depending on low-level features can provide more accurate data associations and more robust loop closures without high computational requirements as shown in Atanasov et al., 2014, Bowman et al., 2017. Although adding semantic information to SLAM systems, provides additional knowledge to achieve more robust data association and loop closures, extracting accurate 3D position of the semantic objects is a challenging problem, and errors in the position estimation can induce errors in the data associations and mapping of such semantic objects. The inaccuracies in estimating the 3D positions of the semantic objects are mainly due to two factors; (1) Uneven and complex 3D structure of the different semantic objects. (2) Errors in the detections of the semantic objects i.e bounding boxes provided by the object detectors do not fit accurately around the detected objects. However, semantic objects are abundantly present

Figure 6.1: The aerial robotic platform used for validating the presented visual semantic slam approach with the frame of references.

with several vertical and horizontal planar surfaces, which can be extracted to improve the relative position estimation of these objects. Hence, in order to overcome the above mentioned limitations and to achieve a robust and lightweight SLAM algorithm, we propose a semantic SLAM approach, using planar surfaces extracted from semantic objects capable of running on-board an aerial robotic platform.

The proposed algorithm can be divided into two parts; in the first part, the robot state is propagated using a VO/VIO estimate, the VO/VIO uses low-level features from the environment for the propagation of the robot state. Due to the inaccuracies in low-level feature detection and matching part as well as errors and biases in the IMU measurements (for VIO systems), the VO/VIO estimations of the robot state accumulate errors over time, which is corrected associating the high-level planar information of the detected semantic objects with the mapped semantic landmarks. Hence, the second part of the algorithm consists of a sparse semantic map of the planar surfaces extracted from the semantic detections. The semantic detections provided by state of the art object detectors, are combined with a planar extraction technique to extract the planar surfaces within the detections. The created semantic map thus consists of planar surfaces with their centroids and normal orientations along with their class labels as well as planar surface type (horizontal or vertical), which are augmented with new detections of the semantic objects. The main contributions of the presented work are:

- Robust and lightweight semantic SLAM algorithm capable of running on-board an aerial robot.
- Incorporation of fast planar extractions inside the semantic detections, for accurate

high-level data association and mapping of the semantic landmarks.

6.2 Semantics based Plane Extraction

In order to extract high-level planar information of the semantic objects, we divide the process in two steps:

6.2.1 Semantic Object Detection

The semantic object detection can be performed using any state of the art object based detectors. We select the You Only Look Once (YOLOv2) (Redmon and Farhadi, 2016) object detector, in order to satisfy the on-board computational limitations of the aerial robots. We select the lightweight Tiny-YOLOv2 model trained on the COCO dataset (Lin et al., 2014), providing a real-time performance with an average GPU consumption of 300 mb. The object detector is modified in order to detect the provide detections only of the relevant objects, as well as provide the bounding boxes for the objects only above a particular probability threshold. The object detection is performed on the images received from the RGB camera.

Although Tiny-YOLO does not require high computation, it requires an on-board computer with a GPU support. Hence, in order to also test our approach on-board aerial robots without any GPU support, we utilize a CPU based implementation of a shape and color based object detector, which is capable of detecting in real-time blue as well as red colored cube shaped objects. The detector first processes the color based information in HSV colored space in order to filter out objects based on color. The filtered image is then processed using a shape image processor which takes into account the approximate shape of the corresponding object in order to detect cube shaped objects of the corresponding color. More details regarding the detector can be found in Sampedro et al., 2017. The detected bounding boxes are then provided to the object segmentation (Sect. 6.2.2) which segments the planar information from the 3D point clouds generated from the depth image registered with the RGB image.

6.2.2 Semantic Object Segmentation

The received bounding boxes of the semantic object from Sect. 6.2.1, which are extracted from the RGB images, are used to extract the relevant 3D point cloud data of the semantic object. This 3D point cloud is generated from the corresponding depth image registered with the RGB image. As shown in Fig. 6.7c the received bounding boxes from the detector can have errors, as the bounding boxes do not fit the objects perfectly. Taking a mean of characteristic 3D points or all 3D points within the detected bounding boxes as presented in Bowman et al., 2017 can cause errors in the calculation of the relative 3D position of the semantic object, which would induce errors in the mapping of the semantic data. In order to minimize the errors in the mapping of the semantic objects, inspired from our previous approaches of planar clustering and segmentation as shown in Chapter. 5 and Chapter. 4, we segment all the horizontal and vertical planar surfaces

present within the detected bounding boxes with their centroids as well as their normal orientations in the following manner:

6.2.2.1 Normal Extraction

In order to perform fast and real-time extraction of normals at each 3D point, we use the integral normal estimation technique (Holzer et al., 2012), also presented in Chapter. 5 and Chapter. 4.

Briefly, this technique first converts the z component of the input 3D point cloud $\mathbf{I}_i$ to an integral image $\mathbf{I}_m$. Since in an integral image, the sum of all values within a particular region is calculated, it makes the computation very fast and efficient. After performing a smoothing, choosing the appropriate smoothing areas using depth change maps, a normal $\mathbf{n}_p$ at point $\mathbf{p}$ is calculated as: $\mathbf{n}_p = \mathbf{p}_h \times \mathbf{p}_v$. Where $\mathbf{p}_h$ is the vector of 3D right and left neighbors of point $\mathbf{p}$ and $\mathbf{p}_v$ is the vector of the 3D top and bottom neighbors of point $\mathbf{p}$. Fig. 6.1 shows a representation of normals extracted from an object. The matrix $\mathbf{N}_p^c$ contains all the normal orientations of each 3D point in the point cloud, and is passed to the centroid extraction step explained below.

6.2.2.2 Centroid Extraction

For fast and robust planar surface and centroid extraction, we use the method proposed in Trevor et al., 2013. This approach takes as input the $\mathbf{N}_p^c$ computed from the previous step as well as all the corresponding 3D points vector. All planes are represented by the planar equation $ax + by + cz + d = 0$ and each 3D point is thus represented as a vector consisting of the euclidean point, its normal and n_d :

$$p = \{x, y, z, n_x, n_y, n_z, n_d\} \quad (6.1)$$

where $n_d = \{x, y, z\} \cdot \{n_x, n_y, n_z\}$. Euclidean distances between the normal directions and the distances n_d are computed for the neighboring points in order to find all the connected components. The computed connected components are then checked for their curvatures in order to filter non-planar components. After the extraction of all the planar surfaces with their centroids and normals, we first check whether the planar surfaces are horizontal planes as follows:

$$d_{hor} = \|\mathbf{n}_p - \mathbf{n}_g\| \quad (6.2)$$

where $\mathbf{n}_p$ is the normal to the extracted planar surface and $\mathbf{n}_g$ is the normal of the ground planar surfaces which is known. If the d_{hor} is less than t_{hor} , then the planar surface $\mathbf{n}_p$ is labeled as a horizontal plane. If the d_{hor} is greater than the t_{hor} , the planar surface is checked for vertical threshold as:

$$d_{vert} = \mathbf{n}_p \cdot \mathbf{n}_g \quad (6.3)$$

If d_{vert} is less than the t_{vert} the planar segment is $\mathbf{n}_p$ is labeled as a vertical plane. The extracted planar surfaces thus contain the following information:

- Centroids $\mathbf{s}_p = \{p_x, p_y, p_z\}$

- Normals $\mathbf{s}_n = \{n_x, n_y, n_z, n_d\}$
- Planar type label $s_o = \begin{cases} 1, & \text{if horizontal} \\ 0, & \text{if vertical} \end{cases}$
- Class type label $s_c =$ corresponding detected class type.

6.3 Graph SLAM

As mentioned above, the VO/VIO estimates can accumulate errors in absence of characteristic features in the environment and as seen from 6.2.2 semantic detections can have errors due to occlusions, insufficient lighting, resulting in uncertainties in the 3D position estimates as well. Due to the presence these non-linear uncertainties the use of filtering techniques like the Extended Kalman Filter (EKF) SLAM can cause divergence in the estimate of the robots as well as the landmarks poses. As opposed to filtering techniques which consider only the most recent previous state of the robot, graph slam based techniques provide the advantage of considering all the previous robots states, as well as account for higher non-linearities. Hence, in order to robustly fuse the measurements from the VO/VIO and the semantic detections, we use graph slam based optimization. The algorithm can be divided into three main stages:

6.3.1 VO/VIO Odometry

The advantage of using loosely coupled approach for fusing VO/VIO estimations with semantic data, allows for integrating several state of the art VO/VIO systems, into the framework, using the best approach for a particular environment. In this work we integrate into our framework three different visual odometry algorithms namely : (1) ROVIO (Bloesch et al., 2015) (2) Snap VIO ¹ (3) RTAB-map odometry (Labbe and Michaud, 2019).

ROVIO is a monocular visual inertial odometry algorithm based on an Extended Kalman Filter (EKF). ROVIO uses direct image intensity errors of image patches in order to achieve robust tracking. These image intensity errors are used as innovation terms during the update stage of the EKF. These image intensity measurements are tightly coupled with an IMU to accurately estimate the pose of the robot with true metric scale. But due to the noise of image intensity calculations as well as the noise in the IMU measurements, pose estimates provided by ROVIO can accumulate huge drift over time and in many scenarios the pose estimate can diverge without recovery.

During very high angular motions of the aerial robots and less characteristic features in the environment, the pose estimated by ROVIO can tend to diverge completely from the true robot pose. Due to this limitation, for high speed flights on-board the aerial robots, we use the snap VIO algorithm. The snap VIO algorithm is also a monocular approach which is able to estimate the pose of the robot, where the robot pose estimates drift with time but the algorithm does not tend to diverge completely as in case of ROVIO.

¹<https://github.com/ATLFlight/ros-examples>

Figure 6.2: The structure of the graph created, using the VIO and the semantic landmarks. (a) VIO feature extraction and pose estimation. (b) The semantic detections in the RGB image with the planar surfaces in green extracted from the registered and segmented point cloud data. (c) The constructed graph, where K_i is the keyframe created for an VIO pose connected with the relative poses between them.

In order to compare our proposed approach with a standard RGB-D dataset (Sturm et al., 2012) which does not provide IMU data time synchronized with the images, we use the RTAB-map RGB-D based visual odometry module. The odometry algorithm uses Feature to map (F2M) approach which registers a new keyframe with local map of features created from the previous keyframe. It uses the Good Features to Track (GFTT) (Jianbo Shi and Tomasi, 1994) for extracting the keypoint features and uses the BRIEF descriptors of the extracted features to match against those of the local map created. The motion estimation is performed using Perspective-n-Point algorithm (PnP) as presented in the OpenCV library (Bradski, 2008). A local bundle adjustment is performed in order to refine the obtained odometry estimate.

6.3.2 Graph Construction

The robot state vector $\mathbf{x} = [\mathbf{x}_r, \mathbf{R}_r]$ is propagated over keyframes k . Where $\mathbf{x}_r = [x, y, z]$, x, y and z are the robots position estimates along the x, y and z axis respectively with respect to the world frame of reference W (Fig. 6.1) and $\mathbf{R}_r$ is the rotation matrix of the robot with respect to the world frame W . We assume that the initial state of the robot is known. Each landmark consists of it state and covariance as well as the labels for the planar surface type as well as the class type represented as $\mathbf{L} = \mathbf{L}_1, \dots, \mathbf{L}_n$. Where $\mathbf{L}_i = (l_{z_i}, l_{\sigma_i}, l_{o_i}, l_{c_i})$, l_{z_i} being the 3D position of the i -th landmark, l_{σ_i} is the covariance

of the i -th landmark and l_{o_i} and l_{c_i} being the planar as well as the class type of the i -th landmark.

The front end of the algorithm comprising of VO/VIO provides the 3D pose estimates in the world frame of reference W . The estimate of the robot state $\mathbf{x}_r$ at time t is added to factor graph as a keyframe node K_t . The constraint between adjacent keyframes K_{t-1} and K_t is added in the form of an edge using the pose increment between them $\mathbf{u}_r(k)$. The pose increment obtained from the the VO/VIO poses at time $t - 1$ and time t can be derived as:

$$\mathbf{u}_{rt} = \ominus \mathbf{x}_r(k - 1) \oplus \mathbf{x}_r(k) \quad (6.4)$$

$\mathbf{x}_r(k - 1)$ and $\mathbf{x}_r(k)$ are the pose measurements received at time $t - 1$ and t respectively. Each keyframe K_i is added to factor graph depending on time as well as motion constraints of the robot. Each detected semantic object $\mathbf{S}_i$ after undergoing the process of data association (Sect. 6.3.3), is either added to the factor graph as a landmark node augmenting the map of semantic landmarks $\mathbf{L}$ or associated with the currently mapped semantic landmarks. The relative pose of landmark observed from the keyframe K_i is added as the constraint between the landmark and the respective keyframe. Fig. 6.2 shows the graph constructed using n keyframes and the segmented semantic data from the detections, with their extracted planar surfaces in green.

6.3.3 Data Association

The semantic planar surfaces extracted from Sect. 6.2.2 are received first by the data association stage in the following manner: $\mathbf{S}_i = \{\mathbf{s}_{z_i}, \mathbf{s}_{n_i}, s_{o_i}, s_{c_i}\}$. $\mathbf{S}_i$ is the first detected and extracted semantic planar surface i , containing $\mathbf{s}_{z_i}$ and $\mathbf{s}_{n_i}$ as the centroids of the detected planar surface and its corresponding normal orientations in the camera frame C (see Fig. 6.1). s_{o_i} and s_{c_i} the planar surface type and the class labels respectively. The first received semantic object does not undergo the data association and is directly mapped as the first semantic landmark which can be represented as:

$$\mathbf{L}_i = \{\mathbf{l}_{z_i}, \mathbf{l}_{n_i}, \mathbf{l}_{\sigma_i}, l_{o_i}, l_{c_i}\} \quad (6.5)$$

Where the $\mathbf{l}_{z_i}$ and $\mathbf{l}_{n_i}$ are the semantic landmark centroids and normals in the world frame of reference obtained as:

$$\mathbf{l}_{z_i} = \mathbf{x}_r \oplus {}^w\mathbf{R}_c \cdot \mathbf{s}_{z_i} \quad (6.6)$$

$$\mathbf{l}_{n_i} = {}^w\mathbf{R}_c \cdot \mathbf{s}_{n_i} \quad (6.7)$$

Where ${}^w\mathbf{R}_c$ is the rotation matrix from the camera frame to the world frame. $\mathbf{l}_{\sigma_i}$ is the uncertainty of the estimated position of the semantic landmark. The initial value of $\mathbf{l}_{\sigma_i}$ is decided based on the number of 3D points contained in the planar surface. Semantic object with lower number of 3D points corresponds to high certainty of error in it estimated centroids and normal orientations and will have a higher $\mathbf{l}_{\sigma}$. After the mapping of the first semantic landmark, each detected semantic planar surface $\mathbf{S}_k$ undergoes the data association process in the following three steps:

- First, the received semantic planar surface is checked for whether its class label matches the class label of the semantic landmarks, and whether its planar type is equal to the planar type of the semantic landmark. It further undergoes a check for whether the number of 3D points representing the planar surface are greater than a certain threshold t_p and if the area of the semantic planar surface is greater than a threshold t_a . This step ensures exclusion of erroneous centroids of the planar surfaces extracted due to the detected bounding boxes fitting incompletely over a semantic object.
- In the second step the normal orientations of the planar surface in the camera frame are converted to the world frame using equation. 6.7 as:

$$\mathbf{l}_{n_k} = [{}^w\mathbf{R}_c^T \cdot (\mathbf{s}_{n_k})] \quad (6.8)$$

The difference $\mathbf{l}_{n_k}$ between $\mathbf{l}_{n_i}$ has to be lower than a predefined threshold t_n , for the planar surface to pass to the next step.

- If the semantic planar surface passes the first step and the second step, the relative 3D measurements of the centroids are converted from the camera frame to the world frame using the equation. 6.6 as:

$$\mathbf{l}_{z_k} = [{}^w\mathbf{R}_c^T \cdot (\mathbf{s}_{z_k})] \quad (6.9)$$

We can then compute the Mahalanobis distance for the detected semantic object using the error obtained by subtracting $\mathbf{l}_{z_k}$ and $\mathbf{l}_{z_i}$. If computed the Mahalanobis distance is greater than a given threshold, the detected semantic object is mapped as a new landmark $\mathbf{L}_j$, else the semantic object $\mathbf{S}_k$ is matched with the current semantic landmark $\mathbf{L}_i$.

During the graph optimization step (Sect. 6.3.4), both the positions $\mathbf{l}_z$ as well as the covariance $\mathbf{l}_\sigma$ of all the mapped semantic landmarks are optimized.

6.3.4 Graph Optimization

After the construction of the graph (Sect. 6.3.2), the graph optimization step consists of finding a configuration of the nodes that best fits the given VO/VIO measurements and the semantic landmarks. $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_m)^T$ is the vector of state of the robot, where $\mathbf{x}_i$ and $\mathbf{x}_j$ are the poses of node i and j connected using the edge $\hat{\mathbf{z}}_{ij}$, which is the relative pose between them obtained from the VO/VIO estimates. $\mathbf{\Omega}_{ij}$ is the information matrix between nodes i and j . $\mathbf{z}_{ij}$ is the semantic landmark measurement observed by the nodes i and j . The log likelihood of the measurement therefore can be given as:

$$l_{ij} = [\mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}(\mathbf{x}_i, \mathbf{x}_j)]^T \cdot \mathbf{\Omega}_{ij} \cdot [\mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}(\mathbf{x}_i, \mathbf{x}_j)] \quad (6.10)$$

Where,

$$\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}(\mathbf{x}_i, \mathbf{x}_j) \quad (6.11)$$

is the difference between the expected observations from the VO/VIO odometry and the received measurements from the semantic landmarks. For a pair of observations C , the least square estimation problem thus seeks to find $\mathbf{x}^*$ that best fits all the previous observations, given as:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{i,j \in C} \mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{e}_{ij} \quad (6.12)$$

In order to increase the robustness to possible outliers present in the semantic detections, Psuedo-Huber cost function is added to all the semantic measurement constraints. The solution to eq. 6.12 can be found by linearizing around the initial guess $\tilde{\mathbf{x}}$, which leads to iteratively solving for the a linear system with matrix $\mathbf{H}$ and right hand vector $\mathbf{b}$ as:

$$\mathbf{H} = \sum_{i,j \in C} \mathbf{J}_{ij}(\tilde{\mathbf{x}})^T \boldsymbol{\Omega}_{ij} \mathbf{J}_{ij}(\tilde{\mathbf{x}}) \quad (6.13)$$

$$\mathbf{b}^T = \sum_{i,j \in C} \mathbf{e}_{ij}^T \boldsymbol{\Omega}_{ij} \mathbf{J}_{ij}(\tilde{\mathbf{x}}) \quad (6.14)$$

Where, $\mathbf{J}_{ij}$ is the Jacobian of the error function computed in $\tilde{\mathbf{x}}$.

In this optimization process, the robot pose as well the semantic landmarks positions are optimized. After the factor graph optimization we can also recover the updated covariances of the mapped landmarks, in order to compute the Mahalonobis distance required for the data association step (Sect. 6.3.3). Re-observing the semantic object and correctly associating it the the mapped landmark results in the loop closure after the optimization step. For fast and efficient computation of the non-linear optimization problem, G2O (Kümmerle et al., 2011) framework is utilized with Lavenberg-Marquardt solver. Algorithm. 3 explains the complete working the proposed algorithm.

6.4 Experiments and Results

6.4.1 Standard Dataset

To validate our approach, we test on standard dataset and compare it with several state of the art SLAM approaches.

6.4.1.1 RGB-D SLAM TUM Dataset

This dataset ² (Sturm et al., 2012) consists of point cloud data provided from a kinect sensor and Motion Capture ground truth trajectory of the camera. The odometry data for the dataset is obtained using the RTAB map RGB-D visual odometry algorithm explained

²<https://vision.in.tum.de/data/datasets/rgbd-dataset/download>

Algorithm 3: Visual Semantic SLAM

Input: VO/VIO estimated output of the robots pose $\mathbf{x}$, along with the semantic detected and segmented data $\mathbf{S}_j$

Output: Corrected pose of the robot $\mathbf{x}_r$ and the semantic landmark map $\mathbf{L}$

```

1 VO/VIO
2 if  $time > timethres$  &  $dist > distthres$  then
3   Add estimated pose of the robot from the VO/VIO as a keyframe node  $K_i$ 
4   Add an edge between the  $K_{i-1}$  and  $K_i$  using the relative pose
5 Data Association
6 if  $first\_semantic\_data$  then
7    $\mathbf{l}_i = \{\mathbf{l}_{z_i}, \mathbf{l}_{\sigma_i}, \mathbf{l}_{o_i}, \mathbf{l}_{c_i}\}$ 
8   Add a new landmark node with its relative pose
9   Add an edge between the landmark node and current keyframe  $K_i$ 
10 else
11   for  $i=1$  to  $num_{detections}$  do
12     for  $j=1$  to  $num_{landmarks}$  do
13       if  $s_{o_i} = l_{o_k}$  and  $s_{c_i} = l_{c_k}$  then
14         Get the optimized landmark covariances  $\mathbf{l}_{\sigma_i}$ 
15          $\mathbf{l}_{\sigma_j} \leftarrow$  based on the number of 3D points in the semantic planar surface
16         Calculate the min-Mahalonobis distance
17          $\mathbf{v}_i = \mathbf{l}_{z_i} - \mathbf{l}_{z_j}$ 
18          $\mathbf{Q} = \mathbf{H} \cdot \mathbf{l}_{\sigma_i} \cdot (\mathbf{H})^T + \mathbf{l}_{\sigma_j}$ 
19          $d_i = \mathbf{v}_i^T \cdot \mathbf{Q} \cdot \mathbf{v}_i$ 
20       else
21         Map a new semantic landmark
22          $\mathbf{l}_{\sigma_k} = \mathbf{l}_{\sigma_j}$  &  $\mathbf{L}_k = [\mathbf{l}_{z_k}, \mathbf{l}_{n_i}, \mathbf{l}_{\sigma_k}, \mathbf{l}_{o_k}, \mathbf{l}_{c_k}]$ 
23         Add a new landmark node with its relative pose
24         Add an edge between the landmark node and current keyframe  $K_i$ 
25       end
26     end
27     if  $min\_Mahalonobis\ distance \leq thres_{dist}$  then
28       Add an edge between the mapped landmark with the current keyframe  $K_i$ 
29     else
30       Map a new semantic landmark
31        $\mathbf{l}_{\sigma_k} = \mathbf{l}_{\sigma_j}$  &  $\mathbf{L}_k = [\mathbf{l}_{z_k}, \mathbf{l}_{n_i}, \mathbf{l}_{\sigma_k}, \mathbf{l}_{o_k}, \mathbf{l}_{c_k}]$ 
32       Add a new landmark node with its relative pose
33       Add an edge between the landmark node and current keyframe  $K_i$ 
34     end
35   end
36 end
37 Back-End
38 Sparse map of the semantic landmarks
39 Optimize the robot pose  $\mathbf{x}_r$ 
40 Optimize the landmark poses and covariances  $\mathbf{l}_{z_i}$  and  $\mathbf{l}_{\sigma_i}$ 

```

in Sect. 6.3.1 and several semantic objects such as chairs, tv-monitors, books and keyboard are detected and mapped as semantic landmarks and used for loop closure detection. We describe in detail the experiments performed on several sequences of the dataset:

Freiburg3 Long Office Household (fr3/office): In this sequence the kinect RGB-D camera is moved along an environment consisting of several office materials like chairs, tables, books, tv-monitors etc. The camera is moved in different translational motions as well as rotational motions in order to validate the robustness of the proposed approaches. Fig. 6.4 shows the 3D point cloud map generated using the pose estimated by the semantic SLAM algorithm. Figures 6.3a and 6.3b present the trajectory of the camera estimated by our proposed algorithm and the ground truth trajectory. During the execution of this sequence, the average trajectory error (ATE) (Sturm et al., 2012) estimated by our proposed algorithm is 0.033029 m, whereas the ATE of the VIO algorithm is 0.042817 m.

Freiburg2 XYZ (fr2/xyz): In this sequence the kinect camera is moved in translational motion along with x, y and z axis without any rotational motion. Only two semantic objects, a TV-monitor and a keyboard are used as semantic landmarks. Fig. 6.3c and Fig. 6.3d present the estimated trajectory by our proposed algorithm when comparing it to the ground truth trajectory. The ATE of our proposed algorithm for this sequence is 0.018077 m and that of the VIO is 0.018152 m.

Freiburg2 RPY (fr2/rpy): This sequence is similar to the Freiburg2 XYZ experiment, with additional rotational motion of the camera along with with the translational motion. Only two semantic objects, a TV-monitor and a keyboard are used as semantic landmarks. Fig. 6.3e and Fig. 6.3f present the estimated trajectory by our proposed algorithm when comparing it to the ground truth trajectory. The ATE of our proposed algorithm for this sequence is 0.018825 m and that of the VIO is 0.020231 m.

Freiburg2 desk (fr2/desk): In this sequence the kinect camera is moved in translational as well as rotational motion along a desk containing semantic objects such as tv-monitors, chairs, keyboard and books. Fig. 6.3g and Fig. 6.3h compare the trajectory estimated by our proposed algorithm with the ground truth trajectory. In this sequence as can be seen from the figures, during several time instances there is absence of ground truth trajectory estimation. The ATE for the robot pose and the VIO pose are 0.075782 m and 0.102724 m respectively.

6.4.2 Field Experiments

We evaluate our algorithm on several experiments using different system setups and validate it in different indoor scenarios. All the field experiments are compared to the state of the art ORB-SLAM2 (Mur-Artal and Tardos, 2016) as its one of the widely used open-source SLAM framework available. Although our approach uses VIO, which incorporates additional IMU information, the aim of this comparison is to evaluate the

Figure 6.3: The trajectory of the camera estimated by the proposed algorithm when comparing it to the ground truth estimated trajectory for the RGB-D TUM dataset. (a) and (b) present the 3D and 2D trajectories for Freiburg3 Long Office Household sequence. (c) and (d) present the 3D and 2D trajectories for Freiburg2 XYZ, (e) and (f) show the 3D and 2D trajectories for sequence Freiburg2 RPY and (g) and (h) show the 3D and the 2D trajectories for Freiburg2 desk.

Table 6.1: Absolute Trajectory Error (ATE) m of the compared algorithms for the TUM dataset.

	fr2/xyz	fr2/rpy	fr2/desk	fr3/office
<i>RTAB-VO</i>	0.0182	0.020	0.102	0.043
<i>Our approach</i>	0.0181	0.019	0.076	0.033
ORB-SLAM2	0.004	–	0.009	0.010
Elastic Fusion	0.011	0.015	0.071	0.017
Kintinuous	0.029	0.018	0.034	0.030
DVO SLAM	0.018	0.018	0.017	0.035
RGB-D SLAM	0.08	–	0.057	0.032
MaskFusion	0.041	0.076	0.108	0.102
Fusion++	0.020	–	0.114	0.108

performance of our proposed framework which uses higher-level semantic information, against a SLAM system using only lower-level features.

6.4.2.1 System Setup

For the experiments two different system setups are tested as described below:

Hand-held Setup: In this experimental setup, we use a single Intel RealSense D435i camera³. This version of the RealSense consists of an RGB camera, two infrared cameras providing depth information and an IMU sensor. Visual Inertial Odometry is performed using the RGB camera and the IMU. The RGB camera is also used for semantic detections. The depth information is used for extracting the 3D point cloud data required for extracting the planar surfaces from the detected semantic objects.

Aerial Robotic Setup: In this experimental setup, we use the same Intel RealSense camera except that, we use the odometry obtained from Snapdragon VIO⁴ sensor setup. We use the Snapdragon VIO as it is optimized for working on-board the aerial robots, hence can achieve high speed flights without the complete divergence of the VIO algorithm.

6.4.2.2 Results

In this section, we present the results obtained using the different system setups explained in the previous section, in several challenging indoor scenarios in order to validate the accuracy of our proposed approach.

Long Hall Experiment: We perform several experiments in an indoor environment consisting of a long passageway of approximately 22 m in length and 6 m in width. The experiments are performed using the on-board aerial robotic setup explained in Sect. 6.4.2.1.

³<https://www.intelrealsense.com/depth-camera-d435i/>

⁴<https://github.com/ATLFlight/ros-examples>

Figure 6.4: The freiburg3 long office household sequence presenting the generated 3D point cloud map of the environment with the estimated trajectory of the camera shown in red, along with detections received from the yolo detector.

Due to the restrictions of flying in the area, the aerial robotic setup is moved in a hand-held fashion. During this experiment, a total of 5 rounds are performed, covering an approximate total trajectory length of 250 m. The idea of this experiment is to test the robustness of the algorithm in presence of longer trajectories and compare the results with VIO estimates of the robot. Since no ground truth measurements are present during this experiment, the 5 rounds are performed in a repetitive pattern, in order to demonstrate the drift accumulated by the VIO after each round and the accurate drift free pose estimate provided by our approach. We present the accuracy of the algorithm through the 3D map of the environment constructed using the estimated pose estimated by our algorithm.

Planar surfaces extracted from the recycle bins, which were commonly present in the environment, are used as semantic objects. A total of only 15 of these randomly placed semantic objects are used for mapping and improving the drift of the VIO. Fig. 6.5 shows the 3D and the 2D plots obtained when performing the experiments and Fig. 6.5c shows the 3D point cloud map generated during this experiment along with the detections of the recycle bins.

Long Corridor Experiment: The aim of this experiment is to validate the robustness of our algorithm in presence of huge estimation errors of the VIO/VO algorithms as well as in presence of large clutter of semantic objects. The experiment is performed in a long corridor of length 14 m and width 20 m, with challenging illumination conditions adversely

Figure 6.5: Long Hall experiment where (a) represents the 3D plot and (b) represents the 2D plot. (c) represents the 3D map generated during this experiment with the detections of the recycle bins, with each detection number representing the it location on the 3D map.

Figure 6.6: Long Corridor experiment where (a) represents the 3D plot and (b) represents the 2D plot. (c) represents the 3D map generated during this experiment with the detections from the yolo detector, with each detection number representing the it location on the 3D map and letter S on the map shows the start and end point.

Figure 6.7: Repetitive trajectory with several semantic objects experiment. (a) and (b) representing the 3D and the 2D plots respectively. (c) shows the detections of the semantic objects using the yolo detector.

(a)

(b)

(c)

Figure 6.8: Random trajectory with several semantic objects experiment. (a) and (b) representing the 3D and the 2D plots respectively. (c) shows the detections of the semantic objects using the yolo detector.

(a)

(b)

(c)

Figure 6.9: Experiment performed on-board the aerial robot. (a) and (b) are the 3D and 2D plots respectively. (c) shows the detections of the recycle bins.

effecting the performance of the VIO algorithm. This experiment is performed using the hand-held camera setup as explained in Sect. 6.4.2.1. The environment consists of a clutter of several semantic objects in the form of chairs and tables. Only the planar surfaces of chairs are used for the semantic mapping, as the horizontal planes of the tables do not fit entirely into the bounding boxes, which cause errors in its relative position estimation. A total of three repetitive trajectory loops are performed around the corridor, covering an approximate total trajectory length of 204 m. Fig. 6.6a and Fig. 6.6b present the 3D and the 2D plots obtained during this experiment, comparing the pose estimated using our approach and the VIO pose estimate. In order to demonstrate the accuracy of our algorithm in absence of ground truth data, we present the 3D map of the corridor (Fig. 6.6c), generated using the 3D pose estimated by our approach, also showing the trajectories estimated by our approach and the VIO, along with the detections at distinct time instances.

Repetitive Trajectory with Several Semantic Objects: In order to validate the approach in presence of several semantic objects of different shapes and size, we perform an experiment with the hand-held camera setup explained in Sect. 6.4.2.1. Several semantic objects such as chairs, tv-monitors, keyboards and books are used, placing them in random positions. A total of 4 rounds are performed in a repetitive manner during this experiment with a total approximate trajectory length of 69 m. Fig. 6.7 shows the 2D and the 3D plots obtained in this experiment, comparing the results of output of the algorithm with the ROVIO, ORB-SLAM2 and ground truth data. Our proposed approach has an ATE of 0.225635 m, whereas ATE of the VIO algorithm used is 0.312486 m. The ORB-SLAM2 algorithm has an ATE of 0.267485 m.

Random Trajectory with Several Semantic Objects: The aim of this experiment is to test the robustness of the proposed approach in presence of random translational as well as rotational motions of the camera with a random clutter of several semantic objects. The experiment is performed using the hand-held camera setup explained in Sect. 6.4.2.1. The semantic objects present are chairs, tv-monitors, books and keyboards. The clutter of the semantic objects induces errors in the detections of the yolo detector as seen in Fig. 6.8c. Fig. 6.8a and Fig. 6.8b show the 3D and the 2D plots obtained during the execution of this experiment. The results obtained from our approach are compared with the ROVIO, and the ORB-SLAM2. Our approach has an ATE of 0.393335 m, whereas the ATE's of ROVIO and ORB-SLAM2 are 0.550146 m and 0.316575 m respectively.

On-board Aerial Robot: In order to test the robustness of the algorithm in presence of high speed motions of an aerial robot, we run it on-board an aerial robot flying at maximum speeds of 2 m/s. The on-board aerial robotic setup explained in Sect. 6.4.2.1 is used with the aerial robot flown manually, performing a squared trajectory with several loops. The semantic objects used in this scenario are the blue colored recycle bins. A total of only 6 recycle bins are used, which proved sufficient along with the VIO data in order to correctly estimate the trajectory of the aerial robot. During the take-off phase of the aerial robot, due to its sudden motion, it can induce huge drift in the z -axis estimate of

the VIO pose which cannot be corrected as no semantic landmark is mapped during this initial time instant. In order to compensate for this drift, we introduce the approximate location of the first landmark, which is optimized during the optimization process along with the other semantic landmarks and the robots poses. Fig. 6.9 represents the 3D and 2D plots obtained during this experiment. During this experiment, our proposed approach has a ATE of 0.280243 m and the VIO algorithm used has an ATE of 0.651487 m. ORB-SLAM2 losses feature tracking during this experiment due to high speed motions as well as insufficient features.

6.5 Discussions

6.5.1 Standard Datasets

Sect. 6.4.1.1 presents the evaluations for the RGBD TUM dataset and Table. 6.1 compares the results obtained using our proposed algorithm with the VO algorithm as well as other state of the art techniques. It can be observed from Freiburg3 Long Office Household (Fig. 6.3a and Fig. 6.3b) and Freiburg2 desk (Fig. 6.3g and Fig. 6.3h) experiments that, the front end VO algorithm using the low-level features, accumulates errors as the camera navigates around the environment and our method combining the VO and a sparse map of the semantic landmarks is able to correct the error accumulated (see Table. 6.1). In the Freiburg2 XYZ (Fig. 6.3c and Fig. 6.3d) and Freiburg2 RPY (Fig. 6.3e and Fig. 6.3f) experiments, even though our method provides better results than the VO algorithm, the pose estimate is improved only by a short margin. This is due to the fact that, during these experiments the camera performs very short trajectories hence the VO has less drift.

As it can be seen from Fig. 6.4, even with errors in the predicted bounding boxes around the detected semantic objects, our algorithm is able to estimate the camera pose (see Fig. 6.3), correctly extracting, mapping and associating the planar surfaces of the detected objects. The effect of the noise of the detections is minimized in the data associations, due to the fact that the extracted planar surfaces are only considered valid if their area is beyond a certain threshold, which in this case is empirically set to 15 cm². Hence the extracted planar surfaces due to the incomplete bounding boxes around the semantic objects, or due to the incomplete semantic object present in the image during certain time instances, are rejected as their computed area is smaller than the threshold.

It can be observed from the Table. 6.1, that in these sequences ORB-SLAM2 outperforms in most of them, this is partially due to the fact that ORB-SLAM2 compensates the scale bias in the depth maps for some of the sequences. Our algorithm although in the presence of errors in the VO estimates and the detections, achieves results comparable to the state of the art techniques and better results as compared to the object based SLAM techniques i.e *MaskFusion* and *Fusion++*.

6.5.2 Field Experiments

Sect. 6.4.2 presents the system setup and the results obtained from the three experiments performed using our own field tests. We discuss the obtained results during all

these performed experiments below:

Long Hall Experiment: Fig. 6.5 compares the results of our algorithm obtained during this experiment with the VIO algorithm. During this experiment the aerial robotic setup repeated the same 5 rounds around the long hallway. As seen from the 3D plot Fig. 6.5a and the 2D plot Fig. 6.5b the VIO estimates accumulate an error of over 3 m in the z -axis and x -axis along with errors in the orientation, whereas our proposed which uses these noisy estimates from the VIO, along with the planar surfaces of the recycle bins is able to correctly estimate the pose of the robot without any drift. It can also be appreciated from this experiment, even with presence of huge errors in the pose estimates of the VIO, our algorithm requires very few semantic features to accurately estimate the pose of the robot. In order to check the accuracy of the pose estimate of the robot, Fig. 6.5c presents the 3D map of the environment constructed using the pose estimated from our algorithm. Visually, it can be observed that using the pose estimated by our approach, the long hall way is accurately constructed. ORB-SLAM2 is also tested on this experiment, but since ORB-SLAM2 uses the front RealSense camera, it loses tracking when moving close to the walls, as the walls are featureless, whereas since our method depends on the odometry information generated from the bottom camera it is not limited to cases when passing close to the walls. The advantage of loosely coupled VIO integration, allows us to run our algorithm using VIO algorithms which do not lose tracking in such scenarios.

Long Corridor Experiment: Fig. 6.6 presents the execution of the proposed long corridor experiment. This proposed experiment is performed in very challenging indoor scenario, which consists changing illumination conditions effecting to a great extent the pose estimate of the VIO algorithm. This changing illumination can be observed from Fig. 6.6c, where the detection images numbered 3 and 4, present very low illumination as compared to the detection images numbered 1 and 2. This low illumination conditions, effect the accuracy of the pose estimated by the VIO algorithm, accumulating large errors in its x and y positions and a huge error of around 6 m in its z direction (see VIO pose estimate in Fig. 6.6).

Due to the cluster of semantic objects (chairs), the yolo object detector, also estimates several inaccurate bounding boxes around the detected objects (see Fig. 6.6c). In our approach this noise in the detections, does not adversely effect the relative pose estimation of the semantic objects, due to several safety checks performed before adding it as a semantic landmark, including the planar surface threshold check, which in this experiment is empirically set to 10 cm^2 . Using these highly noisy VIO pose estimates and the semantic detections in the corridor, as seen from Fig. 6.6 our approach clearly is able to correct the drift present in the VIO estimations, performing accurate loop closures, even in such challenging indoor environment. We also evaluate the performance of ORB-SLAM2 during this experiment and as seen from the detection images (Fig. 6.6c), the environment consists of several featureless white colored walls, because of which the ORB-SLAM2 which only uses low-level feature information loses its tracking when passing close to the surfaces. Whereas, since our proposed framework can easily integrate both VO or VIO algorithms, we choose the estimates from the ROVIO VIO algorithm, which accumulates large drift

but does not lose feature tracking due to the additional inertial information.

Repetitive Trajectory with Several Semantic Objects: Fig. 6.7 presents the experiment performed using several semantic objects with different shapes and sizes such as chairs, tv-monitors, keyboards and books. Even with semantic objects like chairs, which have different complex 3D structure, the algorithm is able to accurately estimate the 3D position of such semantic objects and map them, hence is able to estimate a drift free trajectory of the robot, when comparing it with the VIO pose estimates and the ground truth pose. As seen in Fig. 6.7c the yolo detector estimates the bounding boxes around the object with errors. Many times the bounding boxes do not completely fit the object or overfit the object. Due to these detection errors, the estimated 3D pose of the semantic object using a median of the 3D point inside the bounding boxes can deviate from the true position. Since we extract the planar surface information from these semantic object and allow only for planar surfaces above a certain area which in this experiment is set to 15 cm^2 , we can accurately map and localize using the semantic landmarks, irrespective of the errors in the detections.

We also compare the results obtained from our experiments with the with ORB-SLAM2 based on only geometric features. Since in this experiment, the camera is tilted towards the ground around 25° and since the floor contains several repetitive patterns, the ORB-SLAM2 degrades in tracking performance having a higher ATE of 0.267485 m as compared to our approach which has an ATE of 0.225635 m using the ROVIO with an ATE of 0.312486.

Random Trajectory with Several Semantic Objects: Fig. 6.8 demonstrates the experiment performed in a random trajectory fashion with a clutter of semantic objects. Due to insufficient lighting conditions and several rotational as well as translation motions of the camera, during this experiment the trajectory estimated by the ROVIO algorithm degrades in its performance and accumulates huge drifts in its estimations (see Fig 6.8a and Fig. 6.8b). It can also be observed from Fig. 6.8c during this experiment the detections received from the yolo detector have significant errors, for example due to the clutter of the semantic objects, several erroneous bounding boxes are estimated around a chair. Planar extraction is not effected to a great extent even with detection errors as the planar surfaces are extracted only if they satisfy the 3D points threshold and the planar area threshold which in this case is 0.15 cm^2 . Thus, even with the huge drift present in the VIO estimations as well as errors in the detections, our approach is able to correct the estimated trajectory of the camera to the ground truth trajectory, with the ATE estimated by our algorithm being 0.393335 m, whereas as the ATE of ROVIO, accumulating large drift in its estimates, being 0.550146 m.

The ATE of the trajectory estimated by the ORB-SLAM2 during this experiment is 0.316575 m. The ORB-SLAM2 also degrades in its performance during this experiment, but does not diverge as much as the ROVIO. During this experiment the odometry estimated by the ROVIO diverges as much as 1.5 m from the ground truth trajectory, which is corrected to a great extent by our algorithm using the semantic landmarks (see Fig. 6.8b). As the front end of our algorithm does depend on the estimated VIO trajectory, these

huge errors in the VIO estimates, results in the performance of our algorithm being a bit inferior when comparing with the ORB-SLAM2.

On-board Aerial Robot: Fig. 6.9 are the plots for the experiments performed on-board the aerial robot. The robot performs several loops flying at a maximum speed of 2m/s. Due to high motions of the aerial robot, the VIO accumulates large drift in its position as well as orientation. Whereas our approach, as can be seen from 3D and 2D plots (Fig. 6.9a and Fig. 6.9b respectively), even with such high speeds of the aerial robot our algorithm does not diverge from the ground truth pose of the robot only using a total of only 6 recycle bins. This experiment proves that our proposed approach is able to correct the position as well as orientation errors present in the VIO algorithm using few semantic features and at high speed motions of the aerial robot.

The indoor environment during this experiment consists of a monochromatic white colored surface with few geometric features. Robot also presents high speed angular and translational motions. Due to these reasons, the ORB-SLAM2 loses the feature tracking and is unable to estimate the pose of the robot. Since our framework loosely couples the VIO estimates, it can integrate several VIO algorithms, using the VIO algorithm best suited for the particular application. The Snap VIO being optimized to run on-board an aerial robot is hence used which also degrades in its performance along with time, but does not lose featuring tracking, always providing an estimate of the robot pose, which is corrected using the semantic landmarks.

6.6 Conclusions

In this chapter we present a fast, robust and lightweight visual semantic slam algorithm capable of running on-board the aerial robots, in order to estimate a drift free trajectory of the robot. We test the algorithm with a standard dataset available in the literature, comparing its accuracy in the form of Average Trajectory Error (ATE) with several state of the art algorithms, showing that the algorithm is able to provide results comparable to the state of the art techniques. We also perform several experiments using our own datasets made in several challenging indoor scenarios, with hand-held camera setup as well as on-board aerial robotic setup, demonstrating the capability of the algorithm to work in these challenging indoor environments as well as on-board aerial robots flying at high velocity. Our algorithm is able estimate low drift estimations of the aerial robots even at such high speed flights of 2 m/s and using few semantic objects. Due to the loosely coupled VIO and the semantic mapping approach we are able to integrate and test our algorithm with several state of the art VIO or VO approaches, taking advantage of each during different scenarios. Video⁵ demonstrates the working of the algorithm in the proposed field tests. Finally, we make the algorithm⁶ publicly available in order for the scientific community to take advantage of the presented work as well improve it for

⁵<https://www.youtube.com/watch?v=1cFWTG-T17U>

⁶https://bitbucket.org/hridaybavle/semantic_slam

adding further enhancements. The presented algorithm along with the results obtained in this chapter is submitted and under review in a well known scientific journal.

Part IV

FIELD TESTS AND VALIDATION

7

Chapter

Validation Framework

7.1 Introduction

To achieve fully autonomous missions using the aerial robots, it is necessary to integrate the localization and mapping approaches (e.g the ones presented in the previous chapters) with several other approaches like perception approaches, control-based approaches, automated planning, coordination approaches etc. The combination of such approaches in an operational aerial robotic system is a technical challenge that requires efficient architectural solutions. In context of industrial engineering solutions, where productivity is important, it is also important to minimize the efforts for the development of software modules and enable their ease of integration.

However, the development of a completely integrated solution for fully autonomous mission, requires a combination of highly interrelated and specialized software modules inside a single architecture. This integration creates new challenges (e.g., efficient multi-tasking execution, adaptability to be used in different problems, scalability, etc.). The research community has presented several open-source software frameworks such as:

- *PX4 Autopilot*: PX4 autopilot¹ is an open-source flight control software framework for autonomous aerial robots. The PX4 framework is based on the open-source hardware Pixhawk. PX4 can integrate several on-board sensors as well as their measurements in order to allow developers to perform autonomous flights tailored to their applications.
- *Generalized Autonomous Aviation System*: GAAS² is an open-source software platform for autonomous aerial robots. It provides developers with a vision based software development platform for performing autonomous aerial robotic flights along with features like collision avoidance, path planning etc. The GAAS software plat-

¹<https://px4.io/>

²<https://www.gaas.dev/>

form is built on top of the PX4 software framework and supports only those aerial robotic platforms with the Pixhawk autopilot.

- *Asctec_mav_framework*: *Asctec_mav_framework*³ developed by ASL-ETHZ, has a special focus on autonomous navigation of aerial robots from Ascending Technologies⁴ and it is not compatible with any other aerial robotic platforms.
- *Paparazzi*: Paparazzi (Brisset et al., 2006) project is developed and used by ENAC and MAVLAB-TU Delft. This project includes not only the software framework but also the hardware autopilot and sensors, but it is not compatible with any other commercial hardware.
- *Twirre*: Twirre (Van de Loosdrecht et al., 2014) architecture, developed by NHL Computer Vision, proposes a hardware and software design. It is focused mainly on hardware and it does not report a high level of autonomy.

Even though these research works have produced important advances, some important challenges remain unaddressed mainly related to: (1) Level of autonomy, i.e. more complex hybrid architectures able to provide more degree of autonomy and (2) Versatility, i.e. more versatile integrated solutions able to be used for different applications and physical aerial platforms.

As a response to these needs, the Aerostack⁵, an open-source software framework was developed. This framework facilitates the creation of autonomous aerial robots, by providing a set of reusable components specialized in functional tasks of aerial robotics (mission planning, self localization, control methods etc.) together with an integration method in a popular three layered architecture namely: Interface, Functional and Executive layer. The multi-layered architecture includes both low-level layers for reactive behavior and high-level layers for intelligent behaviors. At the low-level, Aerostack provides a number of specialized reusable components for visual perception, motion controllers, etc. At the high-level, Aerostack includes a number of components to provide a high degree of autonomy and self-adaptation in complex and dynamic environments.

Aerostack provides versatility in terms of two main features. (1) It is flexible for a wide range of applications from teleoperated flights of single aerial robots to highly autonomous missions of multi-aerial robotic platforms. (2) It is hardware independent, the developed software modules can be reusable for different types of aerial robotic platforms. Aerostack can thus provide higher degrees of autonomy and is more versatile, which can be applied to different types of aerial robotic system used in different types of missions such as industrial inspections, swarm of aerial robots etc.

In this thesis, all the work performed in positioning and mapping algorithms mentioned in the previous chapters, along with other state of the art localization algorithms have been integrated and tested with the Aerostack framework to enable autonomous flights. Additional work has also been performed integrating new aerial robotic systems as well as

³http://wiki.ros.org/asctec_mav_framework

⁴<http://www.asctec.de/>

⁵www.aerostack.org

Figure 7.1: The multi-layered aerostack architecture with all its layers and their corresponding components which can be used for autonomous flights of a single aerial robotic agents or n heterogeneous agents.

new on-board sensors into the Aerostack architecture, making the Aerostack architecture compatible with variety of aerial robotic platforms with different on-board sensor configurations. This chapter thus provides a brief explanation of the current architecture of the Aerostack software framework for aerial robotics, along with the contributions made in integrating and testing several new hardware and software components. The introduction provided in this chapter can help the reader understand the following Chapter. 8, explaining the successful accomplishment of autonomous industrial inspections and international competitions using the Aerostack. For an in-depth understanding of the Aerostack framework reader can refer to Sanchez-Lopez et al., 2017a and Molina et al., 2019.

7.2 Aerostack System Architecture

This section explains in brief the Aerostack architecture in two steps; first, explaining the specifications of its software components and second, the organization of the software components its multi-layered model.

7.2.1 Aerostack Components Specifications

Aerostack provides a library of reusable software components for aerial robotics. The name of process to call each elemental component of the library is used. The notion of process is appropriate since it helps to divide the whole problem of automated support

for aerial robots into partial functional roles. Each process has a purpose or practical use for which the process is designed, and it is named as an agent according to its main function, for example: mission planner process (main function: planning a mission) or motion controller process (main function: trajectory control).

The computational support of a process is designed as an atomic executable unit (a data processor) that receives input data and, as a result of a certain information processing, generates output data. The library of processes is implemented using ROS (Robot Operating System) (Quigley et al., 2009) and each process in Aerostack is implemented as a ROS node. Aerostack runs concurrently processing in a multitasking operating system using the inter-process communication methods provided by ROS: (1) a publish-subscribe mechanism using messages and topics, and (2) a request-reply scheme (services). This multitasking support is important, for example, to execute independent processes at lower level layer (functional layer) and higher level layer (executive layer).

The library of components provides modularity and it is open, so new processes can be included easily, without changing the core of the system and the rest of the processes. To create a new process it is necessary to program the corresponding algorithms and data structures in a class (e.g., in C++ or python language) and apply certain Aerostack conventions to be part of the library. For example, it must be created as a ROS node and it must be subclass of a specific class (called `RobotProcess`) that provides a common functionality (supervision, execution control and standard error messages)

Processes can also be used by developers outside a multi-layered architecture to reuse certain algorithms (e.g., computer vision algorithms) for a new aerial robotic platform. In this case, they can be used as single classes without the dependence on the multi-layered architecture.

7.2.2 Multi layered Model Organization

The Aerostack framework has been organized in a multi-layered fashion for efficient distribution of all the software modules as well as ease of integration of new developed components. The architecture consists of three main layers as seen in Fig. 7.1 given as: (1) *Interface* (2) *Functional* and (3) *Executive*. In Fig. 7.1 ellipses represent the data processing unit or processes which are implemented as a ROS nodes.

Interface layer includes processes that interact with the environment and the aerial robot. It consists of processes which receive on-board sensor data from the aerial robot as well as receive robot actuator commands. It also includes the Graphical User Interfaces (GUI) acting as a communication process with the human supervisor along with the communication processes to communicate with other aerial robotic agents.

The other two layers (functional and executive) are common components of the multi-layered architectures (Bonasso et al., 1996, Alami et al., 1998, Gat, 1998, Kortenkamp and Simmons, 2008) for the autonomous systems. The Aerostack uses standard communication channel, implemented using standard ROS message types common in aerial robotics. This channel facilitates interoperability and enables reusability of the components of the functional and executive layer across different types of aerial robotic platforms.

The functional layer of the Aerostack architecture includes processes for providing for

mechanical abilities to the aerial robots. Aerostack provides a software library to implement these processes and the developer can use and combine them to build a particular robotic system architecture. Examples of these components are:

- *Feature Extraction System*: System that consists of processes that simply read a sensor state or implement complex perception algorithms.
- *Localization and Mapping System*: System with processes that implement localization and mapping for the aerial robot.
- *Planning System*: System with processes responsible for planning the high-level obstacle free trajectories for the aerial robot.

The third layer of the Aerostack architecture is the executive layer and includes the processes that make up the executive system. The main function of these processes is to execute the mission plan by activating and monitoring the execution of the mechanical abilities of the functional layer.

The executive layer in Aerostack creates a logical interface that helps specify mission plans. This layer relieves the human operator from having to specify unnecessary execution details that are common for similar tasks and provides user error protection by verifying the correct syntax and semantics of written commands. The executive layer also verifies that a mission is executed as expected. For example, the executive system checks in advance that each task to be executed is consistent with the situation of the environment, preventing the robot from performing wrong or dangerous behaviors. The executive system also detects the presence of unexpected events and coordinate multiple decisions about what the robot must perform (e.g., deliberative or reactive decisions).

7.3 Aerostack Subsystem Architecture

This section describes in detail the main subsystems within the multi-layered model of the Aerostack framework.

7.3.1 Interface Layer

The Interface layer consists of four main processes as described below:

- *User Interfaces*: The processes of the user interface are responsible for interacting with the human supervisor during the autonomous mission. Using these processes the human supervisor can check the initial state of the robot before starting the autonomous mission as well as additionally monitor the state of the aerial robot during the flight.
- *Robot Interfaces*: These processes are responsible for interacting with the robot autopilot systems. These processes, process the commands from the motor systems (Subsect. 7.3.2.3), in order to convert them to the suitable values required by the autopilots of the aerial robotic systems. They also receive and publish the measurements from the default sensors present on the autopilots.

- *Sensor Interfaces*: These processes are responsible for reading the raw sensor data from the additional on-board sensors of the aerial robot and publishing them through the communication channel to the other layers.

During the development of this thesis, several robotic interfaces were created for integrating different aerial robotic autopilots and platforms. The integrated platforms include: (1) Pixhawk⁶ (2) Parrot Bebop⁷ (3) DJI Matrice 100⁸ and (4) DJI Matrice 210⁹. Sensor interfaces were also created for additional on-board sensors such as 1D, 2D and 3D laser sensors along with RGB-D cameras.

7.3.2 Functional Layer

7.3.2.1 Feature Extraction System

The goal of the feature extraction system is to transform the raw measurements provided by the sensors into a simpler and more usable information (see Fig. 7.2). These extracted features simplify the raw measurements in a way that the components that receive them are able to use them more efficiently.

Figure 7.2: General description of the Feature Extraction System, and its relationship with the rest of the components of Aerostack.

The inputs of this system are the raw measurements given by the sensors; and the perception mode, given by the executive layer (Sect. 7.3.3), that encodes the configuration

⁶<http://pixhawk.org/>

⁷<https://www.parrot.com/us/drones/parrot-bebop-2>

⁸<https://www.dji.com/es/matrice100>

⁹<https://www.dji.com/es/matrice-200-series>

of the complete perception system (including the sensors, the feature extraction system, and the localization and mapping system). Its only output are the extracted features.

Similarly to the other systems, the specification of the feature extraction system imposes no prior restriction on the type or quantity of its inputs and outputs and the restrictions will only come with its particular implementation (and its related systems). As a result of this open definition, any kind of feature extractor, using any kind of sensor information might be used in Aerostack architecture. Aerostack counts on with a large number of components included in the Feature Extraction System, ranging from:

- Signal filters that extract a particular frequency component of a measurement given by a sensor, for example low-pass, high-pass, band-pass, etc.
- Computer vision based detectors and trackers, for example a FAST keypoints extractor, or ArUco visual marker detector (Garrido-Jurado et al., 2014).
- Shape and color based object detectors as well as deep learning based lightweight object detectors explained in Chapter. 4 and Chapter. 6.
- Point cloud based planar segmentation and clustering algorithms. Explained in Chapter. 5

7.3.2.2 Localization and Mapping System

The localization and mapping system, represented in Fig. 7.3, has the goal to interpret the information provided by the sensors and the feature extraction system to create a useful understanding of the current situation to be used in the decision making and control processes. Along with the motor system (Subsect. 7.3.2.3), it is a critical system of the autonomous flights, as failure of any of these components can result in catastrophic failures of the autonomous mission. The localization and mapping system is highly dependent on the mission, the environment, and the sensors, architecture and robot setup.

The inputs of this system can be either one or all of the below; the raw measurements given by the sensors, the processed information given by the feature extraction system, and the motion references and actuator commands from the motor system. It is worth to note that this specification of the localization and mapping system impose no prior restriction on the number, type, or nature of the used sensors, or the exploited feature extractors. The sensors can as well be located on-board the robotic agent or on ground. The restrictions will come only with the particular implementation of the localization and mapping system.

In general and for completion, as the other subsystems, the localization and mapping system could also include as inputs, some additional information given by other agents of the multi-robot system (if any), like their estimated state, their estimated state of the map, or other information given by their sensors or feature extraction systems. Including this information provided by other agents of the system allows the creation of a multi-robot localization and mapping system.

The outputs of the localization and mapping system, that define its functionality, are the following:

Figure 7.3: General description of the localization and mapping system, and its relationship with the rest of the components of Aerostack.

- *Self-localization*: estimation of the full situation state of the robot, as for example its pose and velocity.
- *Environment Understanding*: estimate of the map of the environment in a format that is usable for the rest of the components of Aerostack.

It is important to note that there is no restriction about how the localization and mapping system represents internally the map or environment. This internal model needs to be converted to the appropriate representation, to make it usable for the rest of the components of Aerostack, conferring more flexibility internally but ensuring its performance within the rest of the components.

During this thesis work, an open-source, robot state estimation framework (Moore and Stouch, 2014) has been integrated into the Aerostack architecture, which is able to fuse any number measurements from of the on-board sensors. This development has resulted in addition of several additional on-board sensor measurements, easily integrable into state estimator in a plug and play manner, increasing the rapidness of sensor integration and data processing in the Aerostack framework.

All the presented localization and mapping modules in the above Chapters. 3, 4, 5 and 6 have also been integrated as components of the localization and mapping system of the Aerostack, in order to achieve robust autonomous flights in challenging indoor environments. The estimations from these algorithms can be directly provided as an input to the motor system or they can be further filtered using the integrated state estimator. This depends on the requirements of the autonomous mission being performed. Additional

integrated localization and mapping components are described in the following Chapter. 8 and also in Pestana et al., 2014b, Pestana et al., 2015, Sanchez-Lopez et al., 2014, Sanchez-Lopez et al., 2015

7.3.2.3 Motor System

The motor system, represented in Fig. 7.4, has the responsibility to generate the actuator commands for the hardware elements of the robot, ensuring that the commanded motion references are followed, knowing the motion feedback. Along with the localization and mapping system, it forms a part of the critical system for the autonomous flight, which cannot permit any kind failures.

Figure 7.4: General description of the Motor System, and its relationship with the rest of the components of Aerostack.

The inputs of the motor system are grouped in:

- *Motion References*, that are motion values to be considered as goals, and are given by the planning system.
- *Motion Feedback*, including the raw measurements given by the sensors; the estimated state of the robot and the environment given by the localization and mapping system; and the extracted features given by the feature extraction system.
- *Motion Mode*, that encodes the configuration of the motor system and modifies its behavior, and that is given by the executive layer.

The only output of the motor system are the actuator commands passed to the robot interfaces of the Interface layer (Sect. 7.3.1), which are eventually sent to the aerial robot.

Similarly to the other systems, the specification of the motor system imposes no prior restriction on the type or quantity of its inputs and outputs and the restrictions will only come with the particular implementation of the motor system (and its related systems). Thanks to this on purpose open definition, any kind of controller, using any kind of motion feedback might be used in Aerostack architecture.

The motor system comprises, therefore, a set of controllers that can be used independently or in cascade to allow the robot to follow a given motion reference. To ensure the correct operation of the motor system, the executive layer (described in Sect. 7.3.3), enables, disables and connects, by means of the motion mode input, the available controllers in a way that they will never cause a fault on the system. It is therefore a requirement that the controllers included in the motor system count with a proper start up and a shutdown routine. The different controllers of the current version of Aerostack are the following:

- Low-level embedded controllers (provided by the hardware autopilot of the aerial platform): motor speed controller, angular velocity control, horizontal attitude control.
- Navigation controllers (described in Pestana et al., 2013a, Pestana et al., 2014a): linear velocity controllers, position controllers, heading controller, point to look control component, path following component.
- Visual servoing controller (described in Pestana et al., 2013b, Pestana et al., 2014c and Chapter. 8).

During this thesis research, the navigation controllers of the Aerostack framework, were tested and tuned for the newly added aerial robotic platforms, in order to achieve their accurate autonomous navigation. It is worth to highlight that the definition of the motor system of Aerostack does not enforce to use Aerostack controllers, being possible to use the embedded controllers provided in some cases by the autopilot hardware of the aerial platform.

7.3.2.4 Planning System

The planning system consists of an agent level mission planning module along with a path planning module. The main objective of the agent level mission planner is to sequentially execute and monitor the commanded mission points by the mission planning system at each agent level (Sect. 7.3.3). In the recent version of the Aersotack, this capability of the agent level mission planning is incorporated into the executive layer, and hence the planning system only consists of the motion planning component. The new planning system as represented in Fig. 7.5, generates the motion references for the motor system to achieve a desired goal. The main objective of the planning system is to generate obstacle free trajectories for the aerial robots. The planning system receives input from the executive layer providing it with the suitable mission point to be reached. The planning

Figure 7.5: General description of the motion planning system, and its relationship with the rest of the components of Aerostack.

system knowing the state of the robot and the environment using the input from the localization and mapping system, can generate trajectory references to the motion control system, in order to perform the shortest obstacle free trajectory.

The planning system has to be able to generate goals fast enough to make possible an efficient reaction to changes in the mission, in the environment, or in the state of the robot. This is specially critical in aerial robots, since their unstable nature disqualifies them to passively wait a slow response of the planning system.

7.3.3 Executive Layer

The Aerostack executive layer as shown in Fig. 7.6 is based on the concept of behaviors, which define behaviors that carry out particular actions of a robot. For example a behavior of *take-off* activates all the necessary processes of the subsystem in order for the aerial robot to take-off and similarly for other kinds of behaviours such as *move_to_point*, *land* etc. The executive layer in the Aerostack framework consists of three main subsystems namely: (1) Mission Control System (2) Execution Control System (3) Belief Management System.

7.3.3.1 Mission Control System

The mission control system receives as an input a mission specified using a certain formal language. Earlier versions of Aerostack included XML based mission specification language, which was complemented using more task based mission specification language

Figure 7.6: General description of the Executive layer, and its relationship with the rest of the components of Aerostack.

TML (Molina et al., 2017). Missions can also be specified by the user using standard Python based language as well as a user friendly graphical editor with behavior trees. The mission control system verifies the correctness of the provided mission and interprets it into sequentially generated execution requests to the executive control system, in order to execute the mission tasks. The mission control system also receives the current state of the environment as well as the aerial robot from the belief management system, after the current task execution, in order to decide the subsequent tasks to be performed.

7.3.3.2 Execution Control System

The execution control system receives sequential task execution requests from the mission control system, in order to execute the required tasks. Each task is performed by the execution control system in the form of behaviors. The execution control system uses the communication channel (see Fig. 7.1) in order to verify if the given task is consistent with the environment situation, thus preventing the aerial robot from executing any dangerous behaviors. Furthermore, the execution control system communicates the result of the given task in terms of success or failure to the mission control system, in order it for the mission control system to make the next decision.

7.3.3.3 Belief Management System

After the execution of each behavior by the execution control system, the environment around the aerial robot can be altered. For example, if the aerial robot performs a behavior

of interaction with an object like picking the object, the belief management system takes into account the altered state of the environment, in order to inform it to the mission control system. The mission control system based on the input decides the subsequent tasks to be performed related to the mission plan.

7.4 Conclusions

This chapter provides a brief introduction to the Aerostack software framework along with its multi-layered component organization and presents the work done in this thesis for extending the capabilities of the Aerostack software framework. The versatile and modular nature of the Aerostack framework has allowed for fast integration and testing of new aerial robotic platforms and sensors as well as newly developed software components with its existing software components, achieving autonomous aerial robotic missions. The Aerostack has demonstrated successful results in several industrial inspections as well as challenging international competitions. Chapter. 8 presents research performed in this thesis using the Aerostack framework in two real-world industrial inspections and an international competition, along with the obtained results and discussions. The Aerostack architecture can be found in several of the following publications; Sanchez-Lopez et al., 2016, Sanchez-Lopez et al., 2017b, Molina et al., 2016, Molina et al., 2019.

Chapter 8

Field Tests and Applications

The aim of this chapter is to demonstrate the research performed and the results obtained, integrating into the Aerostack framework, (Chapter. 7) localization and mapping modules along with the additional modules of perception, control, planning etc with the existing Aerostack modules, in order to perform autonomous flights for industrial inspections and solving challenging international indoor competitions.

Industrial inspections using autonomous aerial robots has gained a lot of attention in the past few years due to their faster mission accomplishment and their ability to reach higher altitudes in dangerous zones. It thus minimizes to a great extent the human intervention, minimizing the danger to human life in hazardous inspection scenarios inside confined spaces. Sect. 8.1 and Sect. 8.2 present work towards this direction, with Sect. 8.1 describing the thermal power plant inspection performed using an autonomous aerial robot and Sect. 8.2 describing the research and development of an autonomous aerial robotic platform for airplane inspections.

As presented in Chapter. 2, in recent years there has also been a surge in the international aerial robotic competitions held world wide due to the increase in the demand for aerial robots, and many research works have been proposed solving these challenging competitions. Sect. 8.3 presents research and development of an autonomous aerial robot to solve the challenging tasks of an international competition as described below.

8.1 Autonomous Thermal Power Plant Inspection

8.1.1 Introduction

Industrial inspection such as thermal power plant inspection is critical for the proper working of the plant and is required to be performed at regular time intervals. During such inspections, the power plant is required to be shutdown completely and the human workers enter the power plant using scaffolding's or similar structure for visual inspections. This exposes workers to hazardous confined spaces requiring the use of careful safety measures for each worker inside. The workers not only have to work in hazardous conditions, but this process has to be performed carefully and slowly, which increases the downtime of the power plant. On the other hand, autonomous aerial robots with high resolution cameras, can enter such confined spaces without the use of additional structures, minimizing the risk to the human workers and decreasing the downtime of the power plant.

Thermal power plants pose several challenges to the autonomous working of these aerial robots. There is absence of GPS signal for localization. The surface of the power plant is made of ferrous material, which causes complete failure of the orientation sensors such as magnetometers. There is almost low to zero illumination inside the power plant, which results in the failure of several RGB or RGB-D based systems for localization of the aerial robots. The above mentioned limitations, result in a very challenging problem for the autonomous aerial robots.

In order to perform autonomous inspection of a thermal power plant overcoming the above mentioned difficulties, an autonomous aerial robotic platform is developed capable of entering such confined spaces with its own illumination in order to navigate to the desired locations inside the power plant to gather visual information of the required structures. Fig. 8.1 shows the exterior as well as the interior of the thermal power plant along with the aerial robotic platform used. The following sections describe in detail the system architecture as well as the results obtained performing the autonomous flights inside a thermal power plant.

Figure 8.1: (a) The exterior view of the inspected thermal power plant (b) Interior of the thermal power plant, with the aerial robot performing the autonomous inspection mission.

Figure 8.2: The DJI Matrice 100 aerial robot used for the thermal power plant inspection with all its components.

8.1.2 Hardware Architecture

In order to successfully perform the inspection mission, the DJI matrice 100 ¹ is mounted with an on-board computer and several on-board sensors for accurate localization and navigation. The total take-off weight of this platform is 3.2 kg and the maximum flight time is around 12 minutes. The on-board autopilot consists of an IMU sensor measuring the angular velocities as well as the linear accelerations, it also consists of a barometer sensor capable of providing the flight altitude of the aerial robot.

Additionally, for accurate localization of the robot in its horizontal axis, we use the Hokuyo UTM-30LX² 2D laser scanner. The 2D laser scanner is also used for active perception and avoidance of the obstacles which might be present inside the power plant. In addition to the barometer, we use the SF30/C³ 1D laser sensor for the measurement of the altitude of the robot, with a maximum altitude measurement range of 100 m. The aerial robot is also equipped with an on-board zenmuse X3 camera⁴, capable of recording images or videos in 4K resolution. We also add a 10000 lumens LED illumination source on the robot for proper visibility of the power plant structure in the captured images and videos. The DJI manifold⁵ on-board computer for real-time processing of the all the algorithms on-board the aerial robot is used. Fig. 8.2 shows the DJI matrice 100 platform used with all its mentioned components.

¹<https://www.dji.com/bg/matrice100>

²<https://www.robotshop.com/en/hokuyo-utm-03lx-laser-scanning-rangefinder.html>

³<https://lightware.co.za/products/sf30-c-100-m>

⁴<https://www.dji.com/bg/zenmuse-x3>

⁵<https://www.dji.com/bg/manifold>

Figure 8.3: The software architecture of the components used during the inspection mission. With the circular boxes representing the main components integrated.

8.1.3 Software Architecture

The Aerostack software framework described in Chapter. 7, is utilized for achieving real-time autonomous flights inside the thermal power plant. Integration of existing with the newly added software components is performed to execute the autonomous mission. Fig. 8.3 shows the architecture of the components used during the execution of this inspection mission and they are described below:

8.1.3.1 Execution System

The execution system consists of a mission planner with the following properties:

Global Mission Planner: The Global Mission Planner (GMP) is a centralized coordinator of the entire mission and was developed in order to coordinate a single aerial robot or a swarm of aerial robots as presented in Sampredo et al., 2016. For the project of power plant inspections its following capabilities are used:

- *Mission Generation:* The GMP has the capability of autonomous mission generation based on the given area to be explored or manual addition of the mission points. In this case, since the area of the power plant to be explored was known before hand, mission points are manually inserted to the GMP in the form of XML file format.

Once the GMP receives the desired mission points from the user, it concatenates all of them with other preliminary tasks such as *take-off*, *hovering* and *landing*. A

Figure 8.4: The aerial robot inside the thermal power plant, with its frame of reference (R) and the world frame of reference (W).

complete mission generated from the GMP can be: $\{take-off, hover, missionpoint1, missionpoint2, land\}$. For this project since the take-off and landing maneuver are performed manually, the GMP mission does not consist of these two stages.

The GMP also has additional capabilities of event handling such as stop the current exploration mission and *go_for_the_target_object*. These capabilities are not used during this inspection mission and are explained further in detail in Sect. 8.3.

8.1.3.2 Planning System

The planning system consists of an agent level mission planner along with a motion planner as given below:

Agent Mission Planner: The main task of the agent mission planner is scheduling each task of the mission received from the GMP and its execution, acting as an interface between the high-level mission received and the actions needed for the accomplishment of the same. The functionalities of AMP used for this project are described below:

- *Dynamic Re-planning:* This capability of the AMP allows it to replan the received mission point from the GMP, if the given mission point lies on an obstacle. The information regarding presence of obstacle on the mission point is provided by the motion planner (Sect. 8.1.3.2). In this case the AMP replans a new mission point

in a predefined radius of the current mission point until an obstacle free point is found.

- *Task Monitoring:* Once the mission execution is started, AMP communicates with the motion planner (Sect. 8.1.3.2) sending it the mission points in a sequential manner. The motion planner generates an obstacle free trajectory to reach the commanded mission point. Once the robot reaches the mission point the AMP receives a *goal reached* from the motion planner, which triggers the scheduling of the next point of the mission.

Motion Planner: The motion planner component is based on the standard Robot Operating System (ROS) implementation of the algorithm proposed in Marder-Eppstein et al., 2010, for differential drive and holonomic robots. This aforementioned motion planner is adapted for aerial robots, due to its robustness against obstacle avoidance and ease of integration.

The motion planning module receives input the 2D occupancy map generated by the 2D laser perception module described in the following Sect. 8.1.3.3 as well as a mission point from the AMP (Sect. 8.1.3.2), additionally it has the parameter of the inflation radius of the robot, which can be set based on the size of the aerial robot. Using this information the motion planning algorithm generates a 2D cost map of the environment, in which the cost values are propagated out of the occupied cells. The types of cost in the cost map can be *lethal*, *inscribed*, *possibly circumscribed*, *freespace* or *unknown space*. For additional information regarding the cost map the reader can refer to Marder-Eppstein et al., 2010. The optimal free motion for the aerial robot is computed based on this cost map. Example of the 2D cost map can be seen in Fig. 8.5. Since the proposed motion planner is limited to 2D, additional capability is added in order to provide 3D trajectories. In this case the third parameter, which is the altitude of the aerial robot can be sent directly using the AMP module.

The motion planner module has an additional capability of defining the *yaw* of the aerial robot. This orientation data is received from the AMP module.

8.1.3.3 Localization and Mapping System

The localization and mapping system used in this inspection mission is composed of the following components:

2D Laser Perception: We integrate the state of the art 2D SLAM module (Kohlbrecher et al., 2011) into our autonomous system. We choose this specific module due to its robust scan matching even with angular motions of the 2D laser sensor, which are common in an aerial robot, and its low computational cost compared to other state of the art 2D SLAM approaches.

As opposed to other 2D grid based SLAM approaches, this approach assumes that the presented system can be usable on platforms that exhibit full 6DoF instead of 3DoF. Due to this assumption that the 2D laser sensor can exhibit 6DoF motion, laser scans are transformed into a local stabilized frame of reference, using the estimated attitude of the

laser from an on-board IMU sensor. These transformed laser scans can be converted to point clouds on scan endpoints, which are preprocessed based on the z distance from the laser.

Using the 2D laser sensor data, the algorithm creates an occupancy grid map with each map coordinate consisting of an occupancy value as well as its gradient. It then performs optimization of the alignment of the beam endpoint with the created map so far, using a Gaussian-Newton based approach in computer vision (Lucas and Kanade, 1981). As the scans get aligned with the existing map, matching is indirectly performed with all the previous scans and hence it does not require an explicit data association. The main objective of the algorithm is estimate the rigid transformation $\epsilon = (p_x, p_y, \phi)^T$ that minimizes

$$\epsilon^* = \operatorname{argmin} \sum_{i=1}^n [1 - M(\mathbf{S}(\epsilon))] \quad (8.1)$$

Where, $\mathbf{S}(\epsilon)$ are the world coordinates of the scan endpoints and the function $M(\mathbf{S}(\epsilon))$ returns the map value of the coordinate at $\mathbf{S}(\epsilon)$.

The robust 2D pose p_x, p_y and ψ of the aerial robot obtained from this module is integrated with other on-board sensors from the aerial robot using the multi-sensor fusion state estimator (Sect. 8.1.3.3) in order to obtain a 6DoF pose of the robot.

Flight Altitude State Estimator: As the thermal power plant is a conic closed surface, using direct measurement from a 1D laser altimeter for the estimation of the altitude can cause errors, as it measures the altitude with respect to the surface below it. Since the power plant is also a confined surface, estimating altitude from barometers can introduce errors in the estimation. Error in the estimation of the altitude during such inspection mission can be critical, as the surfaces to be inspected are located at known altitudes and an error in the estimation can result in required surfaces inspected incompletely. In order to overcome these errors, we improve and use the approach presented in Chapter. 3.

The state vector consists of $\mathbf{x} \in \mathbb{R}^{10}$

$$\mathbf{x} = \{\mathbf{x}_r, x_g, x_i, x_b\}$$

Where $\mathbf{x}_r$ consists of the state of the aerial robot given as:

$$\mathbf{x}_r = \{t_z, v_z, a_z, \boldsymbol{\omega}, \boldsymbol{\gamma}\}$$

t_z, v_z and a_z being the vertical position, velocity and acceleration about the z axis in the world frame (see Fig. 8.4).

$$\boldsymbol{\omega} = \{\omega_x, \omega_y\}$$

$$\boldsymbol{\gamma} = \{\phi, \theta\}$$

ω_x and ω_y being the angular velocities of the robot about the x and y axis respectively. And ϕ and θ being the Euler roll and pitch angles of the aerial robot.

$x_g = \{t_{gz}\}$ is the state of the ground obstacles with their vertical height. x_i , x_b being the bias of the accelerometer and the barometer respectively.

The state estimator fuses the measurement from the sensors using the following measurement models:

- *Distance sensor*: Measurement model consist of measurements of the laser altimeter with respect to the ground obstacle.
- *Barometer*: The measurement model consists of the altitude from the barometer with respect to the world frame (Fig. 8.4) and the barometer bias x_b
- *IMU*: Accelerations of about z axis and the bias x_i . It also consists of the measured angular velocities in x and y axis with respect to the robot frame.

Multi-Sensor Fusion State Estimator: This module consists of an EKF-state estimator responsible for integrating all the sensor data, in order to estimate a complete 6DoF pose of the aerial robot.

The state estimator is based on the implementation presented in Moore and Stouch, 2014. The state estimator consists of a non-linear kinematic process model, able to fuse imu, pose as well as velocity measurements from an unlimited number of sensors. The state vector consists of $\mathbf{x} \in \mathbb{R}^{12}$, consisting of $(\mathbf{p}^T, \mathbf{v}^T, \boldsymbol{\omega}^T)$, where $\mathbf{p}$ and $\mathbf{v}$ represent the positions and velocities of the aerial robot in x , y and z axis. $\boldsymbol{\omega}$ is a vector consisting of the angles, angular velocities and linear accelerations in x , y and z axis respectively.

For this project, the state estimator integrates the following sensor data:

- *2D Laser Perception*: Positions p_x and p_y and velocities v_x and v_y about x and y axis respectively, provided by the 2D laser perception algorithm mentioned in Sect. 8.1.3.3.
- *Altitude*: The altitude estimation p_z provided by the flight altitude state estimator (Sect. 8.1.3.3) explained previously.
- *IMU data*: The imu data fused from the autopilot of the aerial robot consists of roll, pitch and yaw angle of the aerial robot in quaternions and angular velocities ω_x , ω_y and ω_z along the x , y and z axis respectively.

8.1.3.4 Motor System

The motor system for this autonomous mission consists of only the motion controllers as described below:

Motion Controller: The motion controllers used in this approach from the Aerostack framework are divided into two parts: high-level control and low-level control. In order to achieve a smooth trajectory execution between the commanded mission points, the implementation of motion controller described in Pestana et al., 2014a for the high-level control is used. The high controller consists of three control modes: position, speed and

Figure 8.5: The 2D map of the thermal power plant created during the real flight experiment with the frames representing the aerial robot (R) and the world fixed frame (W).

trajectory and as feedback it uses the state estimation from the multi-sensor fusion state estimator mentioned above in Sect. 8.1.3.3.

During this autonomous inspection mission, the aerial robot is controlled in the trajectory control mode. The trajectory control mode consists of a finite state machine (FSM), which consists of two main control strategies: (1) Follow a given straight trajectory segment and (2) Follow a given heading for that trajectory segment. The execution of the FSM to navigate to a desired mission point can be summarized as:

- Turning in yaw axis (if commanded) along with controlling the speed direction in order to align with the commanded yaw of the mission point.
- Follow a straight trajectory line segment, accelerating first and slowing down as the desired mission point is reached.

The FSM first receives the desired obstacle free trajectory from the path planning module (Sect. 8.1.3.2). It then commands to the low-level controller roll, pitch and yaw speed reference commands along with altitude speed references.

The low-level controller consists of a high speed attitude controller provided by the autopilot system of DJI matrice 100, controlling the motor speeds according to the high-level control commands. It also consists of an altitude controller, receiving the altitude speed reference commands from the high-level controller.

8.1.4 Experiments and Results

Several experiments were first performed in the simulated and real laboratory environments to test the integration and working of all the developed components. The final

Figure 8.6: The aerial robot entering inside the thermal power plant through the entrance of width 1.13 m.

experiments are performed inside the thermal power plant in Meirama, Galicia, Spain. Video⁶ shows the execution of the presented autonomous inspection mission.

8.1.4.1 Experimental Setup

Simulation Tests: The aim of the simulated experiments is to perform initial tests of the entire proposed system inside a structure resembling the thermal power plant, in order to evaluate its limitations and source of errors. The simulated power plant has the similar dimensions of the actual power plant measuring 70 m $\times$ 18 m in height and width respectively, along with the cone shaped surface until the initial height of 12 m. We use the Gazebo simulation environment (Koenig and Howard, 2004) in combination with RotorS (Furrer et al., 2016) Software-In-The-Loop (SITL) allowing for realistic emulation of the required sensors and environmental conditions.

The simulation experiments performed proved that the cone shaped surface of the thermal power plant, due to change in its width with increase in the altitude, induces estimation errors in the 2D laser perception module (Sect. 8.1.3.3), as the 2D laser sensor is unable to perceive the entire cone shaped structure. The aerial robot is thus flown autonomously only after the end of the cone shaped surface. Fig. 8.7 presents the experiment performed simulating the inspection of the burners on the 4 corners of the power plant. Table. 8.1 demonstrates the commanded mission points to the aerial robot during this experiment.

⁶<https://vimeo.com/269340493>

Figure 8.7: The trajectory of the aerial robot inside the thermal power plant during the simulation experiment.

Table 8.1: The commanded mission points in m and deg to the aerial robot during the simulated autonomous mission.

x	y	z	yaw
-1.0	0.0	15.6	-120.0
-1.0	0.0	50.0	-120.0
11.0	12.0	50.0	45.0
11.0	12.0	15.6	45.0
-1.0	12.0	15.6	120.0
-1.0	12.0	50.0	120.0
11.0	0.0	50.0	-45.0
11.0	0.0	15.6	-45.0
0.0	0.0	15.6	0.0

Table 8.2: The commanded mission points in m and deg to the aerial robot during the autonomous mission.

x	y	z	yaw
0.0	0.0	33.0	0.0
0.0	6.5	33.0	45.0
0.0	6.5	50.0	45.0
0.0	6.5	33.0	45.0
0.0	0.0	33.0	0.0
0.0	0.0	15.0	0.0

Field Tests: After the validation of the system using simulated experiments as well real flight experiments in indoor laboratory environment, the final field tests are performed inside the actual thermal power plant. The thermal power plant to be inspected has dimensions of 93.7 m $\times$ 18.6 m height and width respectively. The power plant has several circular shaped entry points each with diameter of 0.5 m, as well as one entrance at the bottom with dimension of 1 m $\times$ 1 m. We choose the entrance at the bottom due to it larger size and as it also has a take-off and landing surface for the aerial robot.

The surface above the take-off area has an opening of 1.13 m wide and 1.6 m tall, followed by cone shaped structure extending until 15 m from the take-off surface. As we validate through simulation experiments that the 2D laser perception module cannot function accurately in presence of the conic surface, the aerial robot is taken-off and flown manually approximately to an altitude of around 15 m, where all the processes related to the autonomous flight are launched, in order to start the autonomous mission. It should be noted that since the flight altitude estimator (Sect. 8.1.3.3) estimates the flight altitude with respect to the world frame (W), it is not required to start the autonomous mission at an accurate altitude after the cone shaped structure, an approximate altitude reference is sufficient. Fig. 8.6 shows entrance of the aerial robot inside the power plant and Fig. 8.8 shows the entrance to the thermal power plant, with Table. 8.2 presenting the commanded mission points during this experiment.

Figure 8.8: The real trajectory of the aerial robot inside the thermal power plant, with a true scaled image of the plant in meters, presenting its entrance and the burners. The red colored trajectory is the trajectory followed by the aerial robot, and the white trajectory represents the trajectory the aerial robot would have followed if using the laser altimeter measurements. The white arrows and white crosses are trajectory directions and commanded mission points respectively.

Figure 8.9: Images of a section of the burner taken during the execution of the autonomous inspection mission.

8.1.4.2 Results and Discussions

Simulation Tests: Fig. 8.7 demonstrates the complete execution of the simulation experiment, with the aerial robot commanded the mission points in order to inspect the four corners of the power plant. Since the aerial robot is taken-off from an initial altitude of 12 m the flight altitude estimator (Sect. 8.1.3.3) is initialized at that altitude, hence as can be noted from Fig. 8.7 the laser measurements start from an initial altitude of 0 m, whereas the flight altitude estimator fusing the laser sensor, barometer and accelerometer is able to estimate the true altitude of the aerial robot. During the execution of the mission, the aerial robot traverses over the conic surface which induces errors in the measurements of the laser sensor (Fig. 8.7), which are also accurately filtered using the flight altitude estimator. The 2D laser perception module works accurately with increase in the altitude of the aerial robot, as the surface of the power plant is uniform after the conic surface. The simulation experiment also proves useful in testing the working of the high level modules such as the GMP, AMP and the path planner and validating the communication between all the software modules. As the size of the power plant is known, mission points can be pre-planned and as can be observed from Table. 8.7, for inspecting the entire thermal power plant only few mission points are required to be planned, making the power plant inspection execution process easy and fast from the point of view of the plant maintenance engineers.

Field Tests: The goal of the real inspection mission is to inspect the burners of the thermal power plant located at one of the lateral sides, ranging from altitudes of 33 m to 48 m. The aerial robot is taken-off manually from the world frame of reference (W) as shown in Fig. 8.8. As after 15 m in z -axis the width of the thermal power plant is constant, the autonomous flight of the aerial robot is performed from the center of the power plant at an approximate altitude of 15 m from the take-off point and all the mission points are commanded with respect to the take-off point. The aerial robot is commanded mission points along the x , y , z -axis as well as the yaw , in order for it to successfully perform the inspection mission. Table. 8.2 shows the list of the commanded mission points for the presented inspection mission.

Fig. 8.8 presents the scaled version of the thermal power plant, showing the burners to be inspected along with the trajectory of the aerial robot performed during the inspection mission in red. Since the take-off and landing procedures are performed manually, they are not included as a part of the autonomous mission. During the mission execution, the mission points commanded by the GMP, are sequentially executed and monitored by the AMP. The trajectory planner plans and commands the obstacle free trajectories to the high level trajectory controller. In the entire mission, the aerial robot captures the visual information of the power plant in the form of a 4k resolution video with an operator controlling the focus and the zoom of the camera from a safer distance. Fig. 8.9 shows the images of a section of the burner taken during the execution of the autonomous mission. As can be seen from the images, the aerial robot perfectly reaches the desired mission points capturing minute details of the burners with the help of the on-board camera and the illumination. As can be seen from Fig. 8.8, only one lateral side with the burners was inspected, out of the four sides. The other sides were not inspected due to the maintenance work carried out simultaneously on the other sides limiting the access to one lateral side of the thermal power plant. However, the presented work proves the autonomous visual inspection by an aerial robot inside a challenging environment of a thermal power plant and inspecting other lateral surfaces of the power plant is similar and would only require few more additional commanded mission points as seen in the simulation tests. It can be seen from Table. 8.2 that the autonomous mission does not require as input several mission points, only a few mission points at specific inspection areas are required. This way knowing the map of the thermal power plant and axis conventions, the user can easily and rapidly create inspection missions with minimum points, in order to inspect other required parts of the power plant.

It is also interesting to note from the red colored trajectory of the aerial robot in Fig. 8.8 that, during the entire execution of the autonomous mission, the flight altitude estimator (Sect. 8.1.3.3) accurately estimates the flight altitude of the aerial robot with respect to the take-off point, even as it flies over the conic surface, which induces errors in the laser altimeter measurements. This measurement errors can be observed in the white colored trajectory in Fig. 8.8 which presents the hypothetical trajectory performed by the aerial robot if using only the laser altimeter measurements. As can be seen from the figure, since the laser altimeter measurements are always with respect to the surface, the increase in the altitude of the conic surfaces results in decrease in the altitude measured by the laser altimeter. Thus using only the laser altitude measurements, the aerial robot

would not be able to accurately perform the autonomous inspection mission.

During this inspection mission, the captured video as well as the extracted images are analyzed by the inspection officers and have proved sufficient for the primary visual inspections of the structures of the power plant.

8.1.5 Conclusions

In this section, an autonomous aerial robotic platform capable of performing visual inspection inside a confined and hazardous thermal power plant is presented. The system consists of a DJI matrice 100 platform mounted with an additional 2D laser sensor for horizontal perception and 1D laser sensor for vertical perception. Using the combination of the 2D and 1D laser sensors, the aerial robot is able to accurately perceive the 3D environment of the power plant in order to perform the autonomous inspection mission. The presented approach has been tested and validated inside a thermal power plant with the aerial robot flown at altitudes of 50 m in order to inspect the burners of the power plant. The proposed visual inspection system can perform inspections rapidly without requiring any additional structures, reducing the downtime of the thermal power plant and minimizing the risk to human life. The presented autonomous system with the obtained results is submitted and under review in a well known scientific journal.

8.2 Autonomous Airplane Inspection

8.2.1 Introduction

Commercial airplane inspection is a critical task from the point of view of safety of all the passengers. During an airplane inspection all the parts of the aircraft need to be inspected and revised with careful detail. Currently this process involves several maintenance engineers using cranes or similar structures to visually inspect each part of the airplane and locate for possible defects on the aircraft. If any defects are found in any part of the structure, the aircraft undergoes repairs and further safety check carried out by the engineers. This first step of manual visual inspection of the airplane, can be a tedious job and requires a lot of manual labor consuming several human resources as well as time. This results in the airplane to be grounded for inspections for more time and can incur huge losses to the airplane service provider.

Autonomous aerial robots can play a vital role during this first visual inspection phase, due to their capability to reach higher altitudes, making it possible to inspect almost the entire surface of the airplane without the use of time consuming machinery like cranes. These aerial robots can follow the user commanded mission points around the airplane structures in order to collect high resolution images of these structures from several angles, which can then be analyzed by the maintenance engineers, in order to locate any kinds of defects.

However, such aerial robots can pose several challenges when flying inside an airplane hanger. The first limitation is the absence of GPS signals required for localization of the aerial robot. Second, the inspection environment inside the hanger poses several other challenges such as moving obstacles like cranes and other structures, which requires the aerial robots to have a complete perception of its environment as well as the intelligence for the avoidance maneuvers. Third, the aircraft hangars are huge in dimensions requiring larger range laser sensors as well as the hangars can have monochromatic surfaces causing errors in the vision based localization techniques.

Keeping these requirements in mind, we propose a completely autonomous aerial robotic solution, using on-board cameras, a 3D laser sensor and an on-board computer, to autonomously inspect an aircraft located inside an airplane hangar, following the commanded mission points by the user.

8.2.2 Hardware Architecture

Autonomous inspection of airplanes is a critical task as the aerial robot is required to inspect the airplane with high precision and safety without inflicting any damages to the vital and expensive aircraft components. These safety and precision requirements enforce the use of high precision sensors as well as an industrially qualified aerial robot. Thus for this project, we use the industrially qualified DJI Matrice 210⁷ mounted with several custom on-board sensors as shown in Fig. 8.10 with a total take-off weight of 4.8 kg and a maximum flight time capacity of 17 minutes. The DJI Matrice 210 autopilot

⁷<https://www.dji.com/bg/matrice-200-series>

Figure 8.10: The Matrice-210 aerial robotic platform used during the autonomous airplane inspection mission with its on-board sensors.

comes embedded with an Inertial Measurement Unit (IMU) and barometer, providing information regarding its orientation, angular velocities, linear accelerations and vertical height. For accurate and high speed measurement of the velocities of the aerial robot, the DJI Matrice 210 is equipped with two front cameras as well as two downward facing cameras. This system performs visual odometry using each sensor and is able to calculate the velocity of the aerial robot fusing velocities from the sensor in both directions.

Perception and localization using only cameras can degrade in environments of the hangar without proper illumination and monochromatic surfaces. In order to increase the robustness in the localization of the aerial robot as well as its perception of the indoor environment, we use a 3D point cloud sensor called the ouster OS-1⁸ consisting of 64 channels. The OS-1 sensor has a range of 100 m and provides a precision in the range of ± 1.5 -10 cm. For capturing high resolution images of airplane defects, we use the DJI Zenmuse X7 camera with a maximum resolution of 6016×4008 .

The real-time on-board processing of the sensor data as well as real-time on-board execution of the autonomous algorithms is performed using the DJI Manifold on-board computer. The DJI manifold consists of an ARM-architecture quad-core processor, along with NVIDIA keplar based GeForce graphics processor.

⁸<https://www.ouster.io/product-os1>

Figure 8.11: The Aerostack architecture with the main software components used in order to perform the airplane inspection mission.

8.2.3 Software Architecture

We use our Aerostack software framework as elaborated in Chapter. 7 and Sanchez-Lopez et al., 2017a, for performing complete autonomous flights using the previous mentioned hardware architecture. Figure. 8.11 represents the Aerostack architecture used in the proposed approach, several new components along with the combination of the exiting components have been used in order to achieve fully autonomous airplane inspections. The utilized software components are described below:

8.2.3.1 Executive System

The main component of the executive system utilized during the inspection mission is described below:

Mission Planner: The primary task of the mission planner (MP) is to receive the user commanded mission points and distribute them based on the number of robotic agents present. Currently, since only a single robotic agent is used, the MP distributes all the commanded mission points to that single agent. The input to the MP consists of the desired mission points commanded by the user written in the form of a text file. These mission points are commanded taking into account the 3D model of the airplane, an initial take-off point with respect to the 3D model and the required regions of the aircraft to be inspected. The commanded mission points are in the format of $r_x, r_y, r_z, c_{yaw}, c_{pitch}$.

Figure 8.12: The graphical user interface used for generating the user commanded mission points for the aerial robot with respect to an initial point in front the airplane.

Where r_x , r_y and r_z are the desired positions of the aerial robot with respect to the initial predefined take-off point. c_{yaw} being the desired yaw angle and c_{pitch} being desired pitch angle of the on-board camera, in order to capture the images at desired camera angles for each commanded mission point. The MP converts all the commanded mission points from user commanded frame of reference to the frame of reference required by the robotic agent. After this step the MP executes the mission as follows:

- *Alignment of the mission points:* The MP first aligns all the commanded mission points in order to arrange them one after the other in an orderly manner.
- *Execution and monitoring of each aligned mission points:* The MP then performs the execution of the aligned mission points one at a time and monitor their execution before sending the subsequent mission point.

The MP has the default tasks of take-off, hover and land with each mission. The MP performs the mission execution in the following manner:

- *Take-off:* Send the take-off command and monitor for its completion.
- *Hover:* Hover for desired seconds in order for the aerial robot to stabilize.
- *Mission Point 1:* Command the first mission point and monitor its execution.
- *Hover at Mission Point 1:* After the first mission point is reached, hover in order to stabilize the aerial robot and record an image with the desired camera angle.
- $\vdots$
- *Mission Point n:* Command the last nth mission point and monitor its execution.

- *Hover at Mission Point n* : After the last mission point is reached, hover in order to stabilize the aerial robot and take an image with the desired camera angle.
- *Land*: After the completion of all the commanded mission point, perform the task of landing.

At each mission point the MP records an image of the airplane along with the current position of the aerial robot and its camera angle. The MP at the end of each mission generates a text file for the user containing the image names with the respective poses of the aerial robot along with its camera angles with respect to the take-off point. The captured images can then be superimposed correctly on the 3D model of the airplane in order to identify the location of each captured image on the airplane. Figure. 8.12 represents an example of the graphical interface for commanding the mission points to the aerial robot.

The MP also has an additional functionality to pause the mission at the current position of the aerial robot if it receives a flag from the obstacle sensing system (Sect. 8.2.3.2) regarding presence of a moving obstacle in its vicinity. The MP resumes the mission from the current point whenever it receives a flag from the obstacle sensing system regarding the obstacle being no longer in its vicinity.

8.2.3.2 Localization and Mapping System

The localization and mapping system is responsible of perception and awareness of the environment during the inspection mission. It consists of the following components:

Obstacle Sensing System: The obstacle sensing system is mainly designed for the efficient execution of the mission in presence of several dynamic obstacles in the inspection scenario such as humans, or structures like cranes etc. The input to the obstacle avoidance system is the 3D point cloud from the 3D laser sensor described in the hardware architecture (Sect. 8.2.2). Each 3D point is referenced with respect to the 3D laser, and in order to calculate the relative distance we calculate the L^2 norm at each point. If the average of a given number of relative distances is less than a given threshold, the obstacle sensing system sends a flag to the mission planner in order to stop the trajectory at the current point. As soon as the obstacle is outside the minimum relative distance threshold the obstacle sensing system sends a continue trajectory flag to the MP, in order to continue the mission.

3D Laser Odometry: To accurately estimate the pose of the aerial robot in indoor environments, we use the 6D pose of the aerial robot derived from the 3D laser odometry algorithm.

In order to perform fast and real-time processing of the entire 3D point cloud data on-board the aerial robot, we use the Three Dimensional Normal Distribution Transform (3D-NDT) based scan matching algorithm. 3D point cloud data does not explicitly provide information regarding the characteristics of a surface such as its orientation, smoothness, etc, and large number of points requires large amount of memory for processing.

Normal distribution transforms on the other hand represent the 3D surfaces much more compactly. The normal distribution algorithm subdivides the 3D point cloud into 3D subspaces (cubes) providing the piecewise smooth representation of the 3D point cloud. It then computes the Probability Distribution Function (PDF) at each subspace, which can be seen as an approximation of a local surface, describing its surface, orientation and smoothness. Further details of the algorithm can be found in Merten, 2008.

The 3D-NDT scan registration algorithm calculates the pose (translation and rotation) which maximizes the likelihood of the current points of the scan lying on the reference scan surface. The 3D point cloud data is pre-filtered before being provided as an input to the 3D-NDT scan matcher. In the pre-filtering stage, all 3D point cloud data within a given distance threshold is selected. The data is then downsampled using voxelgrid filter and further processed for outlier removal using statistical outlier removal method. This pre-filtered point cloud is used by the 3D-NDT scan matcher in order to perform registration. The registration of the current point cloud data is performed matching it with a given previous keyframe thus obtaining the translation and rotation with respect to that keyframe. The keyframes are selected after each predefined translation, rotation or time steps. The odometry is calculated composing the current pose with the previous obtained poses. For filtering inaccuracies in the calculated current pose, the translations and rotations beyond a given threshold are discarded.

Multi-Sensor Fusion State Estimator: This module consists of an EKF state estimator responsible for integrating all required sensor data, in order to estimate a complete 6D pose of the aerial robot. The state estimator model is the same as described in Sect. 8.1.3.3. For this project, the state estimator integrates the following sensor data:

- *3D Laser Odometry:* Positions p_x and p_y about x and y axis respectively and roll, pitch and yaw angle of the aerial robot with respect to the world frame in quaternions, provided by the 3D-NDT scan matcher algorithm mentioned in Sect. 8.2.3.2.
- *DJI Visual Positioning System:* Velocities v_x and v_y of the aerial robot in the robot frame of reference, provided by the DJI sensor described in Sect. 8.2.2 along with the flight altitude of the aerial robot p_z .
- *IMU data:* The IMU data fused from the autopilot of the aerial robot consists of roll, pitch and yaw angle of the aerial robot with respect to the world frame in quaternions along with angular velocities ω_x , ω_y and ω_z along the x , y and z axis respectively.

8.2.3.3 Motor System

The motor system consists of the motion controller described below:

Motion Controller: The same motion controller described in Sect. 8.1.3.4, is used during the execution of the autonomous airplane inspection mission. The high-level motion controller receives commands directly from the mission planner explained in Sect. 8.2.3.1.

The motion controller then commands to the low-level controller of the DJI autopilot system the desired roll, pitch, yaw speed reference and altitude speed reference commands, depending on the commanded mission point.

The low-level controller consists of a high speed attitude controller provided by the autopilot system of DJI matrice 210. It also consists of an altitude controller, receiving the altitude speed reference commands from the high-level controller. The low-level altitude controller is also responsible for the take-off and landing maneuvers of the aerial robot.

8.2.4 Experiments and Results

Since this is an on-going project, our approach is being validated using several real flight experiments in an airplane hanger inspecting different exterior surfaces of the airplane. We present two real flight experiments performed for the inspection mission explaining the entire experimental setup and the results obtained.

8.2.4.1 Experimental Setup

The experiments are performed in an airplane hanger of dimensions of length $\times$ width $\times$ height equal to 100m $\times$ 100m $\times$ 30m respectively. The airplane to be inspected is in the center of the hanger. Several moving structures such as cranes as well as humans working in the workplace simultaneously have to be taken into account during the autonomous mission. The world frame of reference (W) for the aerial robot is fixed on the ground surface at a distance of 6.38m from the nose of the aircraft and the robot frame (R) is fixed on the aerial robot (see Fig. 8.13 and Fig. 8.14).

Validation Experiment: The first experiment consisted of a validation of the entire autonomous system, inspecting the fuselage of the airplane in a horizontal manner. The commanded mission points to the aerial robot are in the world frame of reference (W) with the aerial robot taking-off from an initial point with a transformation with respect to world frame of -20.7 m in x axis and -10.17 m in y axis as seen in Fig. 8.13 and 90 deg in the yaw axis. The aerial robot is commanded mission points with increasing in x axis direction, at constant altitude and with suitable changes in the yaw angle of the camera for keeping the airplane in field of view. The aerial robot maintains approximately a distance of 5 m from the aircraft fuselage. Table. 8.3 shows all the commanded mission points for this inspection mission.

Aircraft Front Inspection Experiment: After the validation of the first experiment, the second experiment consists of inspecting the front of the aircraft in detail. During the entire experiment, the distance of the aerial robot to the aircraft is maintained between 2.0 m to 2.5 m to even capture miniature sized details of the aircrafts like nuts etc. The mission points are selected in order to provide an overlap between the captured images, keeping in mind the field of view of the camera. The commanded mission points consisted of a ladder like pattern performed by the aerial robot as seen in Fig. 8.14, inspecting the aircraft in the same horizontal plane at different altitudes. The on-board camera is

Table 8.3: The commanded mission points in m and deg to the aerial robot during the autonomous mission.

x	y	z	cam_yaw	cam_pitch
-20.7	-10.17	5.0	0.0	0.0
-18.7	-8.17	5.0	0.0	0.0
-16.7	-8.17	5.0	0.0	0.0
-14.7	-8.17	5.0	0.0	0.0
-12.7	-8.17	5.0	0.0	0.0
-10.7	-8.17	5.0	-45.0	0.0
-10.7	0.0	5.0	-45.0	0.0
-10.7	-8.17	5.0	-45.0	0.0
-20.7	-8.17	5.0	0.0	0.0
-20.7	-10.17	5.0	0.0	0.0

commanded angular positions in the yaw direction keeping the aircraft always in its field of view. The aerial robot takes-off with an initial transformation of -23.03 m in x axis and 4.9 m in y axis with respect to the world frame (W) and yaw angle of 0 deg (see Fig. 8.14). A total of 85 mission points are commanded to the aerial robot during this mission. For brevity, Table. 8.4 provides only a list of first and the final commanded mission points to the aerial robot, in order for the reader to understand the pattern of the commanded mission points.

In addition to the commanded mission points, each inspection mission of the aerial robot consists of autonomous take-off and landing maneuver at the beginning and at the end of the mission respectively. During each mission, the images of the airplane are captured at each mission point, along with the meta-data consisting of the current location of the aerial robot.

8.2.4.2 Results and Discussions

Validation Experiment: During the validation mission, the aerial robot is commanded a simple mission, with it tracing a shorter trajectory with a greater distance from the aircraft with a total flight time of 7.2 min. Fig. 8.13 shows the 3D point cloud map of the airplane as well as the hanger obtained during the performance of this experiment, with the yellow line representing the trajectory traced by the aerial robot during the mission execution and the white arrows represent the direction of the trajectory traced. The point cloud map is generated by back projecting the point cloud data using the pose of the aerial robot. It can be appreciated from the accuracy of generated 3D map (Fig. 8.13) that, during this experiment, the estimated pose of the aerial robot does not accumulate huge drift, and the aerial robot is able to correctly follow the commanded mission points presented in Table. 8.3. As the aerial robot is commanded the same take-off and landing point, the accuracy of the pose estimate can also be observed while it takes-off and lands back at the same point.

(a)

(b)

Figure 8.13: The map of the airplane created during the execution of the first inspection mission where the yellow line represents the trajectory traced by the aerial robot, the white arrows represent the direction of the trajectory and the red arrows represent the camera yaw angle.

(a)

(b)

Figure 8.14: The map of the airplane created during the execution of the second inspection mission where the yellow line represents the trajectory traced by the aerial robot and the white arrows represent the direction of the trajectory.

Table 8.4: The commanded mission points in m and deg to the aerial robot during the autonomous mission.

x	y	z	cam_yaw	cam_pitch
-23.7	-4.9	2.80	-90.0	0.0
-23.7	-4.97	4.05	-90.0	0.0
-23.7	-4.97	5.30	-90.0	0.0
-23.7	-4.97	6.55	-90.0	0.0
-22.4	-4.97	6.55	-90.0	0.0
-22.4	-4.97	5.30	-90.0	0.0
-22.4	-4.97	4.05	-90.0	0.0
-22.4	-4.97	2.80	-90.0	0.0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
-15.9	4.83	6.55	90.0	0.0
-15.9	4.83	5.30	90.0	0.0
-15.9	4.83	4.05	90.0	0.0
-15.9	4.83	2.80	90.0	0.0

Aircraft Front Inspection Experiment: Fig. 8.14 presents the 3D point cloud map obtained during the execution of the aircraft front inspection experiment, with the yellow line representing the trajectory traced by the aerial robot during the mission execution and the white arrows represent the direction of the trajectory traced. After several validation flights, the entire aircraft front inspection experiment is executed, flying the aerial robot at a closer distance of 2-2.5 m with respect to the aircraft, capturing minute detail of the aircraft. This mission lasts for a total time of 13.2 minutes, inspecting almost the entire front of the aircraft. From the accuracy of the generated 3D point cloud data it can be observed that, the pose estimate does accumulate some drift over time, but it is sufficiently small enough allowing the aerial robot to carry out the entire commanded mission. It should be noted that, the final part of the second experiment is only performed until a certain distance in x axis with respect to the world frame (W) (see Fig. 8.14) as a ladder like structure obstructed the accomplishment of the rest of the final points. This is due to the fact that currently, the inspection missions are performed simultaneously in presence of other maintenance work carried out on the aircraft. Nevertheless, through these experiments, we prove that our proposed system is able to navigate the aerial robot in the large sized indoor environments of an airplane hangar, correctly tracing the commanded mission points to capture minute details of the airplane structure in the form of images. The captured images are saved with the corresponding coordinates of the aerial robot at that position in the world frame of reference (W). Knowing the coordinates of each of the captured images, these images can be superimposed on the corresponding 3D model of the airplane, in order for the maintenance engineers to analyze the airplane for possible defects knowing their locations on the airplane.

8.2.5 Conclusions

In this section we present a solution for autonomous visual inspection of airplanes in indoor environments integrating several on-board visual sensors and a 3D laser sensor. The proposed system is able to run entirely on-board the aerial robot and can inspect the required airplane surfaces in large sized indoor environments of the hangar. The real flight experiments performed in an airplane hangar validate the working of the proposed system capturing high resolution images of the aircraft at several mission points required by the user. Even though this is an on-going project, the current solution has proved useful in terms of minimizing the time and cost of airplane inspections as several of the preliminary manual visual inspection tasks can be carried out quickly by the autonomous aerial robot.

8.3 International Micro Aerial Vehicles Competition (IMAV) 2016

8.3.1 Introduction

The International Micro Aerial Vehicles (IMAV) competition in 2016 was organized in Beijing, China. It posed a very difficult indoor challenge for the autonomous aerial robots to perform high-level missions such as exploration, manipulations of targets etc. The mission comprised of an area of $13\text{ m} \times 13\text{ m}$ (mission zone) with an unknown layout and unknown locations of the objects to be manipulated. The take-off and landing of the aerial robot was supposed to be performed in the operating zone 10 m away from the mission zone. A summary of the indoor mission (see Fig. 8.15) to be performed can be given as:

- Taking-off, either from the floor or from a moving platform located in the operating zone.
- Entering the mission zone via three possible entries: doorway ($1.2\text{ m} \times 2.0\text{ m}$), window ($1.0\text{ m} \times 1.0\text{ m}$) or chimney ($1.0\text{ m} \times 1.0\text{ m}$) (see figures 8.15 and 8.16).
- Picking and releasing cylindrical objects, which are classified into two categories: items ($\varnothing 0.1\text{ m}$ and 0.1 m height) and buckets ($\varnothing 0.25\text{ m}$ and 0.3 m height), which were painted in red (bucketA, itemA) and blue (bucketB, itemB) color. The items were supposed to be picked and released in the same colored buckets.
- Exiting the mission zone via the door or window.
- Landing inside the operating zone, either on the ground or the moving platform, which followed a periodic trajectory.
- Generating an indoor map of the mission zone.

A detailed description of the above mentioned tasks can be found on the official IMAV 2016 website⁹. Taking into account all the previously mentioned tasks, a custom made autonomous aerial robot was developed in order to perform high-level missions in a consecutive manner able to dynamically adapt to changes in the mission and to solve these challenging tasks.

8.3.2 Hardware Configuration

The IMAV 2016 missions required miniaturization and long endurance capabilities from an aerial robot. Furthermore, several sensors and high-end computing capabilities needed to be integrated on-board the aerial robot in order to perform the missions autonomously. Additionally, actuators had to be integrated for interaction with objects. All these factors made the platform design a very complex challenge.

⁹<http://www.imavs.org/2016/>

Figure 8.15: The indoor mission area of the IMAV 2016 competition, illustrating the dimensions of the operating zone and the mission zone.

Figure 8.16: The entrance to the IMAV 2016 house using the door and the window.

Figure 8.17: The aerial robot used in the IMAV 2016 competition performing the release of the red item into the red bucket.

Figure 8.18: Quadrotor aerial robot designed for IMAV 2016, showing 1) Hokuyo laser range finder UTM-30 LX. 2) Intel-NUC 6i5SYK computer. 3) Pixhawk autopilot. 4) Lightware SF10/A altimeter. 5) Fisheye bottom camera. 7) Intel realsense RGB-D camera.

To overcome the aforementioned challenges, a custom aerial robot is built with a total take-off weight of 3.2 kg, a maximum payload capacity of 1 kg and a maximum flight time of 12 mins. An Intel NUC 6i5SYK is the on-board computer utilized. The autopilot used on-board is Pixhawk (Meier et al., 2011) with its low-level controllers. Proprioceptive sensors used include an Inertial Measurement Unit (IMU) and a magnetometer, both integrated in the pixhawk autopilot. A hokuyo laser range finder UTM-30 LX, an intel realsense (R200) RGB-D camera, a standard RGB 180 degree fisheye lens camera (bottom camera) and a lightware altimeter SF10/A are the exteroceptive sensors used, modularly integrated on the aerial robotic platform (see Fig. 8.18). The communication between the autopilot, proprioceptive, exteroceptive sensors and the on-board computer is performed over USB.

For executing the release tasks of IMAV 2016, two analog servos are included on the aerial robotic platform and are controlled using PWM signals from an on-board Arduino pro-mini board. The communication between the Arduino and the on-board computer is performed over USB.

8.3.3 Software Architecture

The Aerostack software framework has been utilized in order to solve the challenges of the IMAV competition. The feature of re-usability of the Aerostack framework allows for several of the software components in IMAV 2016, to be reused in the industrial inspection (Sect. 8.1), even though the hardware architectures are different. The organization of the components and their capabilities are described below:

8.3.3.1 Executive System

The main component of the high-level executive system adapted for the autonomous missions of the IMAV 2016 is described below:

Global Mission Planner: The Global Mission Planner (GMP) as described in Sect. 8.1.3 acts as a central coordinator of all the tasks during IMAV 2016. In addition to the capability of mission generation as described in Sect. 8.1.3, during this competition the GMP has additional capabilities as described below:

- *Dynamic Mission Adaptability:* A new functionality for dynamic mission adaptability is implemented in the GMP. In this way, the GMP has the ability of dynamically reacting to some events that can occur during the execution of the current mission (e.g. the target object has been found in the exploring area). Once an event has been detected, the GMP generates a new mission based on the detected event, saving the state of the previous mission that was being performed and commanding the new generated mission to the Agent Mission Planner (AMP) (Sect. 8.3.3.2). Once the new generated mission has been accomplished by the aerial robot, the GMP recovers the previous mission in its corresponding state. This functionality, with their respective state, provides the system a high-level reactive behavior which allows the aerial robot to perform complicated high-level missions with a high level

Figure 8.19: The Aerostack framework with its software components used during the execution of the IMAV 2016 mission.

of flexibility and autonomy. In order to perform the dynamic mission adaptability the GMP communicates with the perception manager (see Sect. 8.3.3.3), which communicates the occurrence of mission adaptation events, like *go for picking*, *go for releasing*. Based on this mission adaptation event the GMP generates the new dynamic mission, pausing the current one.

8.3.3.2 Planning System

The planning system used in the IMAV 2016 has the following components explained below:

Agent Mission Planner: Agent Mission Planner (AMP) as described in Sect. 8.1.3 is the agent level planner, responsible for scheduling the different tasks that compose the mission sent by the GMP, acting as an interface between a high-level mission received by the GMP and the execution of the actions needed for the accomplishment of such mission. Apart from the capabilities of dynamic mission re-planning and task monitoring explained in Sect. 8.1.3 the AMP has been added with additional capabilities as explained below:

- *Interaction with the Motor System:* This communication has been implemented for object interaction purposes, with the following objectives; First, interacting with the visual servoing (Sect. 8.3.3.5) component for achieving high accuracy in the

object interaction tasks. For this purpose, the AMP receives a *target locked* message from the visual servoing component (Sect. 8.3.3.5) when the object is at the desired position in the image. Second, sending the appropriate command to the corresponding servo motor (actuator) for releasing the carried object. Whenever the visual servoing centers the bucket for releasing the item is at the desired distance, a *target locked* message is sent to the AMP and the AMP responds to it actuating to the appropriate servo motor for release.

Path Planner: The path planner component used in IMAV 2016 is the same as explained in Sect. 8.1.3.2, planning an obstacle free trajectory, using the 2D map of the environment received from the 2D laser perception component (Sect. 8.3.3.3) as well as the mission points received from the AMP.

8.3.3.3 Localization and Mapping System

Responsible for the perception and awareness of the environment, the localization and mapping system used in the IMAV 2016 has the following components:

2D laser perception: The 2D laser perception module is used for the IMAV 2016 is the same as described in Sect. 8.1.3.3, which uses the 2D laser data in order to create a 2D map of the indoor environment, which is updated simultaneously in order to localize the aerial robot in its horizontal axis.

Flight Altitude State Estimator: The IMAV 2016 competition consisted of taking-off from an elevated moving platform and also several ground obstacles in the form of tables, chairs etc. Thus direct measurements from the laser altimeters could not be used. Hence the EKF based flight altitude state estimator as described in Chapter. 3 is used. The process model of the EKF is the same as described in Sect. 8.1.3, with the EKF fusing the measurements from the following sensors:

- *Distance Sensor:* Measurement model consists of the measurements of the SF10/A laser altimeter with respect to the ground.
- *Barometer:* The measurements of the altitude from the barometer sensor provided by the pixhawk autopilot with a bias x_b
- *IMU:* The accelerations in z direction of the aerial robot provided by the IMU of the pixhawk autopilot along with bias x_a . And the angular velocities in x and y axis.

Multi-Sensor Fusion State Estimator: EKF based robot state estimator estimates the 6D pose of the aerial robot as described in Sect. 8.1.3 and Sect. 8.2.3. For the IMAV 2016 competition the following sensor measurements were fused:

- *2D Laser Perception:* The positions p_x and p_y and velocities v_x and v_y provided by the 2D laser perception module (Sect. 8.3.3.3)

Figure 8.20: The aerial robotic platform with the frame of references.

- *Altitude Filtering*: The estimated flight altitude of the aerial robot provided by flight altitude estimator described in Sect. 8.3.3.3.
- *IMU*: The IMU data fused from the pixhawk autopilot, consisting of orientation measurements, angular velocity measurements and linear accelerations along the x , y and z -axis respectively.

Perception Manager: The perception manager manages all the visual recognition events that can occur during the execution of the mission. It includes the following functionalities given below:

- *Object State Management*: One of the main functionalities of the perception manager is the management of the states of the objects which the aerial robot can interact with. The number and characteristics of these objects (i.e. size and color), as well as their initial state are provided to the perception manager in a configuration file, which is loaded at the beginning of the mission.

During the execution of the current mission some perception events can occur (e.g. *object recognized*, *picked* and *released*). When a perception event occurs, the perception manager updates the object states and its own internal state. The internal states of the perception manager considered during the IMAV 2016 competition are: $\{Exploring, Going\ For\ Releasing\ Item\}$. Since the picking is not performed during the IMAV 2016 the perception manager is initialized with a state of the items already picked.

- *Object Mapping*: The perception manager is also responsible for mapping in world frame of reference the position of the detected object received from the visual recognizer (see Fig. 8.20). The extracted position of the bucket is refined by the perception

manager using two filtering stages (1) The perception manager filters out the wrong detections received by the visual recognizer if their altitude is greater than 0.3 m in the z -axis. (2) Instead of using a single estimation the perception manager uses a median of n accumulated estimations in order to accurately estimate the position of the bucket.

- *Mission Adaptation*: Based on the current state of the objects if an event is triggered, the perception manager communicates with the GMP in order to generate a mission adaptation event, for example *go for release* if a bucket is detected.

Landing on a Moving Platform: Landing on the moving platform was the ultimate task in IMAV 2016 where the aerial robot has very little battery in order to perform the landing maneuver. Hence it required a simple, fast and efficient strategy required in order to rapidly land on the moving platform. The moving platform performs a periodic trajectory and its size is previously known. The laser altimeter presented in Sect. 8.3.2 is used for the calculating the speed of the moving platform as it passes below the aerial robot. The time of fall of the aerial robot is computed empirically using experimental validations. Using the speed of the moving platform along with the time of fall of the aerial robot, the exact moment for the aerial robot to land is calculated for it to autonomously land on a moving platform. A detailed explanation of the approach can be found in Rodriguez-Ramos et al., 2017

8.3.3.4 Feature Extraction System

Visual Object Recognizer and Classifier: The visual object recognizer is responsible for detection and classification of the items and buckets, using the on-board RGB camera. It is also responsible for calculating the 3D position of the bucket in the camera frame of reference (see Fig. 8.20). The visual recognizer has the following components:

- *Color and Shape based Filter*: This component first detects and extracts the blue and red colored buckets in HSV space. It then applies a filter based on the 3D geometry of the buckets in order to filter out wrong color based detections. The 3D shape based filter also helps to distinguish between items as well as the buckets.
- *Pose Estimator*: The pose estimator component computes the 3D position of the bucket and the item in the camera frame of reference using the Perspective-n-Point (PnP) algorithm as the sizes of the items and buckets are known.
- *Visual Classifier*: The visual image classifier uses the Region of Interest (ROI) ratio with a decision criteria, in order to classify the 2 items as well as the two buckets. In case of the items the ROI decision criteria is set to 1.0 and the ROI decision criteria for the buckets is 0.833.

8.3.3.5 Motor System

The motor system used in the IMAV 2016 has the following components:

Motion Controller: The motion controller used in IMAV 2016 is responsible for the high-level position control of the aerial robot, as described in Sect. 8.1.3.4. The high-level motion controller receives commands from the path planner module described in Sect. 8.3.3.2. As compared to the previous explained industrial inspection approaches where the aerial robot is controlled in a trajectory mode of the motion controller (following smooth trajectories), in IMAV 2016 the aerial robot is controlled in position mode, partly the reason for the aerial robot to navigate at a slower velocity.

The low-level controller consists of a high speed attitude controller provided by the pixhawk autopilot, converting the attitude reference commands to appropriate motor speeds. The thrust commands generated by the high-level altitude controller are directly send to the pixhawk autopilot.

Visual Position Controller: In order to release the items precisely in the buckets, the aerial robot needs to be centered on top of the bucket. In order to achieve this precision the bottom camera described in Sect. 8.3.2 is used to provide a visual feedback in the form detections. The detections are received from the visual object recognizer (Sect. 8.3.3.4) in the form of ROIs. The received ROI is used to calculate the errors in the image coordinates of the center of the ROI with respect to the center of the bucket. Using these errors, with the focal length of the camera, along with the parameters d_{ref} , O_a and A_m , the position errors, ΔX , ΔY and ΔZ are calculated as:

$$\Delta X = \frac{(x_{ref} - x) d_{ref}}{f_x} \quad (8.2)$$

$$\Delta Y = \frac{(y_{ref} - y) d_{ref}}{f_y} \quad (8.3)$$

$$\Delta Z = d_{ref} - \sqrt{\frac{f_x f_y O_a}{A_{im}}} \quad (8.4)$$

Where, d_{ref} is the desired altitude of the aerial robot over the bucket, O_a is the bucket area in m^2 and A_{im} is the area of the received ROI. Knowing the pose of the aerial robot in the world frame of reference, as well as the rigid transformation of the bottom camera with respect to the center of aerial robot, the calculated position errors can be converted to world frame of reference. These converted position errors are used as the position reference to the motion controller described in Sect. 8.3.3.5, in order to center the aerial robot over the bucket. Whenever the error obtained is lower than the threshold, the visual position controller communicates the AMP (Sect. 8.3.3.2) *target locked* for performing the release maneuver.

8.3.4 Experiments and Results

8.3.4.1 Experimental Setup

Field Tests: During the field tests, several experiments are performed in a similar replica of the environment described in the IMAV 2016 rules (Sect. 8.3.1) in order to test and validate the integration of all the proposed components on-board the aerial robot.

(a)

(b)

Figure 8.21: The performed IMAV 2016 experiment with the orange arrows showing the movement of the aerial robot and the triangles representing the commanded mission points. (a) 3D plot of the trajectory of the aerial robot (b) 2D plot of the trajectory of the aerial robot.

Figure 8.22: The 3D map created during the exploration mission in the IMAV 2016 competition, with the estimated trajectory of the aerial robot in red and white arrows demonstrating the movement of the aerial robot.

The experimental setup consists of an indoor environment of total dimensions $10\text{ m} \times 10\text{ m}$. The operating zone comprises of dimensions of $2\text{ m} \times 9\text{ m}$ and the mission zone with dimensions $6\text{ m} \times 9\text{ m}$, consisting of a facade composed of the windows and doors with the same dimensions as in the IMAV 2016, as well as the same separation between them. Since the mission zone consisted of an unknown layout, several configurations of the mission zones are experimented and tested, each time modifying the dimensions of each room along with random locations of ground objects like chairs and tables. Since no picking task is performed, the aerial robot takes-off with the items pre-loaded. The buckets are also placed at random locations during each experiment.

IMAV 2016: The experimental setup during the IMAV 2016 competition can be referred from Sect. 8.3.1.

8.3.4.2 Results and Discussions

Field Tests: Fig. 8.21 presents the results obtained during the autonomous mission of the aerial robot exploring the unknown indoor environment created for the field tests, as explained in Sect. 8.3.4.1. This experiment validates the autonomous completion of most of the tasks of the IMAV 2016 competition. During this experiment the aerial robot performs the following task in an orderly fashion: (1) Take-off from an initial point. (2) Entering the house through the door. (3) Exploring the house in order to find the buckets

(3) Release of the items in the corresponding buckets. (4) Exit the house through the window (5) Landing on the moving platform.

The GMP is pre-programmed with a given set a mission points in order to explore and find the buckets. These predefined mission points can be seen as the black triangles in Fig. 8.21a and Fig. 8.21b. This complete mission is then executed by the AMP which sends and monitors the completion of each mission point. The points are received by the path planner which plans an obstacle free trajectory passing accurately through restricted spaces such as the door as well as the window, along with avoiding other obstacles in its trajectory. Whenever a bucket is detected by the visual object recognizer (Fig. 8.21) with its relative pose, the perception manager is alerted in order for it to check its current status and compute the pose of the bucket in the world frame of reference. The perception manager sends this pose to the GMP in order for it to pause the current exploration mission and command the aerial robot to a point close to the detected bucket. The visual position controller is then activated receiving the detections of the bucket from the bottom camera (Fig. 8.21). The visual position controller alerts the AMP for performing the release task if the error in the image is below a certain threshold. The GMP continues with the mission once the release task is performed. After both the release tasks are performed the aerial robot exits through the window as it being the shortest path to the moving platform.

The aerial robot executes its final task of landing once it is on top of the moving platform and as soon as it knows the velocity of the moving platform, it performs the landing maneuver on the moving platform. As can be observed during the experiment, the aerial robot navigates on top of moving platform undetected by the 2D laser perception algorithm, but is accurately perceived by the flight altitude estimator as it estimates the true commanded flight altitude of the aerial robot during the landing mission as well during the entire mission. The flight altitude estimator is also able to correctly estimate the altitude of the aerial robot when it is on top of the buckets (their edges which effect the laser altitude measurements), allowing the aerial robot accurately perform the release maneuver. Video¹⁰ presents the results of the above explained experiment along with other performed experiments during the field tests.

IMAV 2016: During the IMAV 2016 competition, two autonomous missions were performed. The first autonomous mission consisted of exploring the house in order to locate the buckets and also create the 3D map of the environment. Fig. 8.22 shows the 3D point cloud map generated using the estimated trajectory of the aerial robot, during the first autonomous exploration mission performed during the IMAV competition. Mostly two rooms were explored as the both buckets where found in the room numbered 2. The second autonomous mission after the exploration mission, was performed with the pre-loaded items performing the following tasks in a row manner (1) Take-off from a moving platform (2) Entering the house through the door (3) Release of the red and the blue items in the corresponding buckets. (4) Exit the house through the door. (5) Landing on the moving platform. During the execution of the mission, the aerial robot only fails during the release of the blue item, but executes the other mission tasks successfully and

¹⁰<https://vimeo.com/205775713>

in a completely autonomous manner. We refer the reader to video¹¹ for the performance of the aerial robot during the second autonomous mission.

8.3.5 Conclusions

In this section we present a complete autonomous solution of the aerial robot, which is able to successfully perform tasks such as exploration and manipulation of the objects present in the environment. The presented solution consists of a custom aerial robotic platform mounted with several hardware components and modular software components. The software components consists of several high-level intelligent modules such as GMP and AMP able to dynamically plan the commanded mission. The system also consists of 2D laser based perception capable of detecting and avoiding obstacles along with localizing the aerial robot in the created 2D map. The system presents a flight altitude estimator is able to accurately estimate the flight altitude of the aerial robot as well as the ground obstacles undetected by the 2D laser sensor along with a multi sensor fusion state estimator able to estimate the complete state of the aerial robot. The system has the ability to detect and identify the objects and release the given items within the objects. During the IMAV 2016 competition the aerial robotic system was the unique system to perform the tasks of the mission in a completely autonomous and in a in-a-row manner. However due to size penalization during the IMAV 2016, the proposed solution is awarded the 4th position in the competition. The results of the competition can be found in the following link¹². The presented approaches in this section can be found in Sampedro et al., 2017, Bavle et al., 2017, Rodriguez-Ramos et al., 2017.

¹¹https://youtu.be/ws799gob_a4

¹²<http://www.imavs.org/2016/documents/Result.pdf>

Chapter 9

Conclusions and Future Work

This chapter provides an overview of the goals achieved using the presented research during the development of this thesis and also outlines the future directions for the evolution of the proposed research.

9.1 Conclusions

The market demand for autonomous aerial robots has grown exponential in the past few years. But autonomous navigation of aerial robots in unknown and unstructured indoor environments still remains an open problems. Perception, positioning and mapping of the environment are the core software modules in order to enable autonomous aerial robots. Presented state of the art (Chapter. 2) demonstrates numerous research advancements made in positioning and mapping of robots in specific environments, achieving successful results assuming specific imposed environmental conditions. Nevertheless, conditions such as sensor inaccuracies, changing environmental factors, etc, can effect the efficiency of the algorithms, resulting in partial or inaccurate mission completion. Research still needs to be carried out to obtain fully intelligent aerial robotic systems capable of accurately perceiving all kinds of indoor environments and localizing themselves in it as well as their cost efficient integration with all the software modules along with their rigorous testing, to solve real world applications.

Firstly, the contributions of this thesis are towards the achievement of intelligent, robust, low computational cost and open-source algorithms for accurate localization of aerial robots in indoor environments using on-board sensors for perception. Secondly, this thesis contributes towards extension of the Aerostack framework for autonomous aerial robots, integrating all the previously mentioned localization modules as well as additional software components and rigorously testing of all the software along with the hardware components accomplishing several heterogeneous autonomous aerial robotic systems, capable of solving different industrial applications and international competitions. The main contributions of this thesis can be listed below:

- **Flight Altitude Estimation using Conventional Sensors**

An open-source, fast and robust EKF based flight altitude estimator for estimating the flight altitude of the aerial robot, along with the estimation of the height of the ground obstacles. The proposed state estimator is tested in presence of several complex shaped 3D ground obstacles combining the information from IMU, barometer and laser altimeter sensors. The state estimator has the following properties; (1) Modularity in terms of number of sensors used. (2) Auto-calibration of sensors, for estimating the sensor bias. (3) Synchronous prediction stage to provide input to the altitude controller whenever required. (4) Asynchronous update stage, update the estimator states whenever a new measurement is received.

- **Pose Estimation using Semantic Stereo Visual Odometry**

Particle filter based open-source, fast and robust stereo visual odometry and semantics based localization framework for estimating the 6D pose of the aerial robot, in cluttered and unstructured indoor environment. The framework is validated using real flights of aerial robots in indoor environments, comparing it with the ground truth data and a state of the art geometric SLAM algorithm. The framework has the following features: (1) Incorporating high-level semantic information into the VO algorithm in order to provide a drift free pose estimate of the aerial robot. (2) Modularity in terms of the integration of VO based algorithms, framework is easily integrable with different VO or VIO based algorithms. (3) Novel approach for estimating the relative position of the semantic objects, extracting the horizontal planar surfaces, hence being generalization for several objects with horizontal planar surfaces.

- **Altitude Estimation and Planar Mapping using Point Cloud Sensors**

An open-source, fast, robust and lightweight EKF based SLAM for accurately estimating the flight altitude of the aerial robots, mapping the planar horizontal surfaces. The approach is tested on-board an aerial robot in presence of static as well as dynamic ground obstacles. The algorithm has the following features: (1) RGB-D perception for altitude estimation, perceiving a richer information of the environment and the ground obstacles. (2) Fast planar segmentation, clustering and mapping using low resolution RGB-D images, in order to estimate the flight altitude of the aerial robot at frequencies of 50-54 hz. (3) Accurate flight altitude estimation and planar mapping even in presence of dynamic ground obstacles, where many standard RGB-D based SLAM algorithms fail.

- **Pose and Map Estimation using Visual Semantic SLAM**

An open-source, fast, robust and lightweight visual semantic SLAM framework integrating geometric as well as semantic information in a loosely coupled manner, in order to estimate the 6D pose of the aerial robot. This method is tested on a standard RGB-D dataset along with several other field experiments including real flight experiments using an aerial robot flying a high speed of 2 m/s. The framework has the following features: (1) Incorporating high-level semantic information along with the low-level geometric feature based estimation, for improving the data

association errors in presence of low feature environment or repetitive patterns. (2) Modularity in the integration of the VO or VIO algorithms, this method has been tested using three different VIO/VO algorithms and can be easily integrated for additional VO/VIO algorithms if required. (3) Fast and efficient planar surface extraction from the detected objects, for accurate position estimation and mapping of the planar surfaces of the detected objects, generalizing to many indoor objects with planar surfaces.

- **Autonomous Missions for Industrial Inspections and International Competitions**

The last part of this thesis presents the work performed in extension of the proposed Aerostack framework, integrating several new heterogeneous aerial robotic framework along with the integration of previously developed positioning and mapping modules, with additional components enabling their autonomous flights in unknown and unstructured indoor environment. One of the important functionality incorporated into the Aerostack was to perform localization and mapping without the aid of external ArUco based visual markers, enabling autonomous flights in challenging unknown indoor environments. A new robot state estimation framework was also added into the Aerostack, enabling quick integration of newer sensor information in a plug and play fashion. After the testing of the newly integrated aerial robotic platforms with their respective sensor setup, the extended Aerostack framework was used for performing the following real industrial inspections: (1) Autonomous visual inspection of a thermal power plant. (2) Autonomous visual inspection of an airplane inside an indoor airplane hangar. This extended Aerostack framework was also used for solving a challenging tasks of the indoor competition called the IMAV 2016.

9.2 Future Work

The proposed work in this thesis along with the strong experimental evaluations point towards the following futures lines of research:

- In the field of positioning and mapping, incorporating high-level semantic information has proved to add robustness to the localization and the SLAM systems in indoor environments. In this thesis, the two approaches using high-level semantic information have utilized object based detection algorithms providing bounding boxes around the detected objects. Although extracting planar information from such bounding boxes has proved to provide accurate relative position estimation results, integrating newly developed object based detectors such as He et al., 2017 which provide pixel-wise segmentation of detected objects can add further robustness to the relative position estimation of the detected objects. New developments in deep learning based object detectors such as Xiang et al., 2017 and Zhou et al., 2019 which estimates the relative pose of the semantic object along with the object detection can also be incorporated in the localization and mapping framework

in order to test its robustness against the proposed classical planar segmentation, clustering and mapping technique.

- In case of autonomous missions for industrial inspections, even though the thermal power plant inspection using the aerial robot provided successful results, as future line of research work visual inspection can be performed using on-board 3D laser sensor instead of 2D laser sensor, providing a better perception of the thermal power plant, in order to perform the take-off and landing phase autonomously as well, favoring the inspection without the need of a human supervisor entering inside the hazardous conditions of the power plant unless required. In case of the visual airplane inspection, an immediate line of future work is to research on the relative localization of the aerial robot with respect to the airplane. This can enable the aerial robot to correct the error in its pose, when commanded specific waypoints with respect to the airplane and ensure that the images of the airplane structures are always captured from required position and angles.

In autonomous missions using aerial robots for international competitions, several challenges have been accomplished in indoor scenarios. Although outdoor scenarios have the presence of GPS signals, aerial robots have to face other perception based challenges like robust obstacle avoidance such as trees, buildings, etc, as well as have robust control based algorithms integrating factors such as wind gusts. Hence another line of future work which is already being executed, is extending the Aerostack framework in order to incorporate measurements from GPS sensors as well as perception algorithms for obstacle avoidance in outdoor scenarios. This future work is supposed to be used in the challenge one of the MBZIRC 2019 competition, where the aerial robot has to detect and chase a target aerial robot in outdoors, in order to capture a spherical ball attached to it. This challenge also requires a development of a state estimator for the target aerial robot, in order to predict its trajectory for interception and capture of the spherical object.

Part V

Appendices

Appendix

Scientific Dissemination

This appendix presents a list of related scientific contributions provided during the development of this thesis.

A.1 Publications

Several scientific contributions have been provided through publications in well known scientific journals and conferences as given below:

A.1.1 Journals

- Jose Luis Sanchez-Lopez, Martin Molina, **Hriday Bavle**, Carlos Sampedro, Ramon A. Suarez Fernandez, Pascual Campoy. “*A Multi-Layered Component-Based Approach for the Development of Aerial Robotic Systems: The Aerostack Framework*”. Journal of Intelligent & Robotic Systems.
- **Hriday Bavle**, Jose Luis Sanchez-Lopez, Paloma De la Puente, Alejandro Rodriguez-Ramos, Carlos Sampedro and Pascual Campoy. “*Fast and Robust Flight Altitude Estimation of Multirotor UAVs in Dynamic Unstructured Environments Using 3D Point Cloud Sensors*”. Journal of Aerospace, MDPI.
- Carlos Sampedro, Alejandro Rodriguez-Ramos, **Hriday Bavle**, Adrian Carrio, Paloma de la Puente, Pascual Campoy. “*A Fully-Autonomous Aerial Robot for Search and Rescue Applications in Indoor Environments using Learning-Based Techniques*”. Journal of Intelligent & Robotic Systems.
- Alejandro Rodriguez-Ramos, Carlos Sampedro, **Hriday Bavle** Paloma de la Puente, Pascual Campoy. “*A Deep Reinforcement Learning Strategy for UAV Autonomous Landing on a Moving Platform*”. Journal of Intelligent & Robotic Systems.

- Adrian Carrio, **Hriday Bavle**, Pascual Campoy. “*Attitude estimation using horizon detection in thermal images*”. International Journal of Micro Air Vehicles.
- Martin Molina, Alberto Camporredondo, **Hriday Bavle**, Alejandro Rodriguez-Ramos, Pascual Campoy. “*An execution control method for the Aerostack aerial robotics framework*”. Frontiers of Information Technology & Electronic Engineering.

A.1.2 Conferences

- Jose Luis Sanchez-Lopez, Ramon A Suarez Fernandez, **Hriday Bavle**, Carlos Sampedro, Martin Molina, Jesus Pestana, Pascual Campoy. “*Aerostack: An architecture and open-source software framework for aerial robotics*”. International Conference on Unmanned Aircraft Systems.
- Ramon A Suarez Fernandez, Jose Luis Sanchez-Lopez, Carlos Sampedro, **Hriday Bavle**, Martin Molina, Pascual Campoy. “*Natural user interfaces for human-drone multi-modal interaction*”. International Conference on Unmanned Aircraft Systems.
- Carlos Sampedro, **Hriday Bavle**, Jose Luis Sanchez-Lopez, Ramon A Suarez Fernandez, Alejandro Rodriguez-Ramos, Martin Molina, Pascual Campoy. “*A flexible and dynamic mission planning architecture for uav swarm coordination*”. International Conference on Unmanned Aircraft Systems.
- A Rodriguez-Ramos, Carlos Sampedro, Adrian Carrio, **Hriday Bavle**, RA Fernandez, Z Milosevic, Pascual Campoy. “*A monocular pose estimation strategy for uav autonomous navigation in gnss-denied environments*”. Proceedings of the International Micro Air Vehicle Conference and Flight Competition, Beijing, China.
- **Hriday Bavle**, Jose Luis Sanchez-Lopez, Alejandro Rodriguez-Ramos, Carlos Sampedro, Pascual Campoy. “*A flight altitude estimator for multirotor UAVs in dynamic and unstructured indoor environments*”. International Conference on Unmanned Aircraft Systems.
- Carlos Sampedro, **Hriday Bavle**, Alejandro Rodriguez-Ramos, Adrian Carrio, Ramon A Suarez Fernandez, Jose Luis Sanchez-Lopez, Pascual Campoy. “*A fully-autonomous aerial robotic solution for the 2016 international micro air vehicle competition*”. International conference on unmanned aircraft systems.
- Alejandro Rodriguez-Ramos, Carlos Sampedro, **Hriday Bavle**, Zorana Milosevic, Alejandro Garcia-Vaquero, Pascual Campoy. “*Towards fully autonomous landing on moving platforms for rotary unmanned aerial vehicles*”. International conference on unmanned aircraft systems.
- Martin Molina, Pedro Frau, Dario Maraval, Jose Luis Sanchez Lopez, **Hriday Bavle**, Pascual Campoy. “*Human-Robot Cooperation in Surface Inspection Aerial Missions*”. International Micro Air Vehicle Conference and Flight Competition (IMAV), Toulouse, France.

- **Hriday Bavle**, Stephan Manthe, Paloma de la Puente, Alejandro Rodriguez-Ramos, Carlos Sampedro, Pascual Campoy. “*Stereo Visual Odometry and Semantics based Localization of Aerial Robots in Indoor Environments*”. IEEE/RSJ International Conference on Intelligent Robots and Systems.
- Carlos Sampedro, **Hriday Bavle**, Alejandro Rodriguez-Ramos, Paloma de la Puente, Pascual Campoy. “*Laser-Based Reactive Navigation for Multirotor Aerial Robots using Deep Reinforcement Learning*”. IEEE/RSJ International Conference on Intelligent Robots and Systems.
- Alejandro Rodriguez-Ramos, Carlos Sampedro, **Hriday Bavle**, Ignacio Gil Moreno, Pascual Campoy. “*A Deep Reinforcement Learning Technique for Vision-Based Autonomous Multirotor Landing on a Moving Platform*”. IEEE/RSJ International Conference on Intelligent Robots and Systems.

A.2 Industrial Projects and International Competitions

The section presents the numerous projects and international competitions executed using the proposed approaches presented in this thesis.

A.2.1 Industrial Projects

- *Vision based Autonomous Landing on a Net*: Participated as a team member in the research of vision based techniques for enabling autonomous landing of fixed wing aerial robots on a net on-board a ship.
- *Inspection of a Thermal Power Plant*: Participated as one of the principal team members in the project of thermal power plant inspection using an autonomous aerial robot for the Spanish company, Naturgy, in Galicia, Spain.
- *Inspection of Airplanes*: Active member of the on-going project of airplane inspections using autonomous aerial robots, for Airbus, Spain.

A.2.2 International Competitions and Public Demonstrations

- *European Night Researchers 2015*: Participated as researcher and main team member in autonomous flight demonstrations using swarm of aerial robots in presence of almost 400 spectators held in Madrid, Spain.
- *Civil Drone 2015*: Participated as a team member in an exhibition of swarm of autonomous aerial robots, at the civil drone conference¹ held in Madrid, Spain.

¹<https://www.youtube.com/watch?v=vsMBIQ3gloQ>

- *International Micro-Aerial Vehicle Competition (IMAV) 2016*: Participated as a researcher and principal team member in IMAV 2016 competition held in Beijing, China. Complete autonomous accomplishment of most of the mission elements using the developed aerial robot was performed. The team achieved 4th position during this competition.
- *International Micro-Aerial Vehicle Competition (IMAV) 2017*: Participated as a researcher and lead team member in IMAV 2017 held in Toulouse, France. The team achieved 2nd position during this competition.
- *International Conference on Intelligent Robots and Systems (IROS) Drone Racing*: Participated as a researcher and lead team member in the IROS 2018 drone racing, held in Madrid, Spain. Research was done for developing an autonomous aerial robot for detecting and traversing frames. Unfortunately, the team did not receive any award during this competition.

A.3 Video Demonstrations

This section provides the link to the video demonstrations of all the algorithms and approaches presented in this thesis.

Chapter. 3: Altitude Estimation using Conventional Sensors

<https://vimeo.com/205774733>

Chapter. 4: Pose Estimation using Semantic Stereo Visual Odometry

<https://vimeo.com/259349563>

Chapter. 5: Altitude Estimation and Planar Mapping using Point Cloud Sensors

<https://vimeo.com/235911971>

Chapter. 6: Pose and Map Estimation using Visual Semantic SLAM

<https://www.youtube.com/watch?v=lcFWTG-T17U>

Chapter. 8: Field Tests and Applications

<https://vimeo.com/269340493>

<https://vimeo.com/205775713>

Bibliography

- Alami, R., Chatila, R., Fleury, S., Ghallab, M., and Ingrand, F. (1998). An architecture for autonomy. *The International Journal of Robotics Research*, 17(4):315–337.
- An, L., Zhang, X., Gao, H., and Liu, Y. (2017). Semantic segmentation-aided visual odometry for urban autonomous driving. *International Journal of Advanced Robotic Systems*, 14(5):1729881417735667.
- Atanasov, N., Bowman, S. L., Daniilidis, K., and Pappas, G. J. (2018). A unifying view of geometry, semantics, and data association in slam. In *IJCAI*, pages 5204–5208.
- Atanasov, N., Zhu, M., Daniilidis, K., and Pappas, G. J. (2014). Semantic localization via the matrix permanent. In *Robotics: Science and Systems*, volume 2.
- Baca, T., Stepan, P., Spurny, V., Hert, D., Penicka, R., Saska, M., Thomas, J., Loianno, G., and Kumar, V. (2019). Autonomous landing on a moving vehicle with an unmanned aerial vehicle. *Journal of Field Robotics*, 36(5):874–891.
- Bavle, H., Manthe, S., de la Puente, P., Rodriguez-Ramos, A., Sampedro, C., and Campoy, P. (2018). Stereo visual odometry and semantics based localization of aerial robots in indoor environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1018–1023.
- Bavle, H., Sanchez-Lopez, J. L., De la Puente, P., Rodriguez-Ramos, A., Sampedro, C., and Campoy, P. (2018). Fast and robust flight altitude estimation of multirotor uavs in dynamic unstructured environments using 3d point cloud sensors. *Aerospace*, 5(3).
- Bavle, H., Sanchez-Lopez, J. L., Rodriguez-Ramos, A., Sampedro, C., and Campoy, P. (2017). A flight altitude estimator for multirotor uavs in dynamic and unstructured indoor environments. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1044–1051.
- Bellavia, F., Fanfani, M., Pazzaglia, F., and Colombo, C. (2013). Robust selective stereo slam without loop closure and bundle adjustment. In Petrosino, A., editor, *Image Analysis and Processing – ICIAP 2013*, pages 462–471, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Beul, M., Droschel, D., Nieuwenhuisen, M., Quenzel, J., Houben, S., and Behnke, S. (2018). Fast autonomous flight in warehouses for inventory applications. *IEEE Robotics and Automation Letters*, 3(4):3121–3128.

- Beul, M., Houben, S., Nieuwenhuisen, M., and Behnke, S. (2017). Fast autonomous landing on a moving target at mbzirc. In *2017 European Conference on Mobile Robots (ECMR)*, pages 1–6.
- Beul, M., Nieuwenhuisen, M., Quenzel, J., Rosu, R. A., Horn, J., Pavlichenko, D., Houben, S., and Behnke, S. (2019). Team nimbro at mbzirc 2017: Fast landing on a moving target and treasure hunting with a team of micro aerial vehicles. *Journal of Field Robotics*, 36(1):204–229.
- Bloesch, M., Omari, S., Hutter, M., and Siegwart, R. (2015). Robust visual inertial odometry using a direct ekf-based approach. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 298–304. IEEE.
- Bonasso, R. P., Kortenkamp, D., Miller, D. P., and Slack, M. (1996). Experiences with an architecture for intelligent, reactive agents. In Wooldridge, M., Müller, J. P., and Tambe, M., editors, *Intelligent Agents II Agent Theories, Architectures, and Languages*, pages 187–202, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Bonnin-Pascual, F., Garcia-Fidalgo, E., and Ortiz, A. (2012). Semi-autonomous visual inspection of vessels assisted by an unmanned micro aerial vehicle. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3955–3961.
- Bonnin-Pascual, F., Ortiz, A., Garcia-Fidalgo, E., and Company, J. P. (2015). A micro-aerial platform for vessel visual inspection based on supervised autonomy. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 46–52.
- Boutteau, R., Sturm, P., Vasseur, P., and Demonceaux, C. (2018). Circular laser/camera-based attitude and altitude estimation: Minimal and robust solutions. *Journal of Mathematical Imaging and Vision*, 60(3):382–400.
- Bowman, S. L., Atanasov, N., Daniilidis, K., and Pappas, G. J. (2017). Probabilistic data association for semantic slam. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1722–1729.
- Bradski, A. (2008). *Learning OpenCV, [Computer Vision with OpenCV Library ; software that sees]*. O’Reilly Media, 1. ed. edition. Gary Bradski and Adrian Kaehler.
- Brisset, P., Drouin, A., Gorraz, M., Huard, P.-S., and Tyler, J. (2006). The paparazzi solution. In *MAV 2006, 2nd US-European Competition and Workshop on Micro Air Vehicles*.
- Brockers, R., Hummenberger, M., Weiss, S., and Matthies, L. (2014). Towards autonomous navigation of miniature uav. In *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 645–651.

- Cherian, A., Andersh, J., Morellas, V., Papanikolopoulos, N., and Mettler, B. (2009). Autonomous altitude estimation of a uav using a single onboard camera. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3900–3905.
- Chirarattananon, P. (2017). A direct optic flow-based strategy for inverse flight altitude estimation with monocular vision and imu measurements. *Bioinspiration and biomimetics*, 13 3:036004.
- Corke, P., Strelow, D., and Singh, S. (2004). Omnidirectional visual odometry for a planetary rover. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 4, pages 4007–4012 vol.4.
- Davison, A. J. (2003). Real-time simultaneous localisation and mapping with a single camera. In *Iccv*, volume 3, pages 1403–1410.
- Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1052–1067.
- Dryanovski, I., Valenti, R. G., and Xiao, J. (2013). An open-source navigation system for micro aerial vehicles. *Autonomous Robots*, 34:177–188.
- Eade, E. and Drummond, T. (2006). Scalable monocular slam. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1*, CVPR ’06, pages 469–476, Washington, DC, USA. IEEE Computer Society.
- Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., and Burgard, W. (2012). An evaluation of the rgb-d slam system. In *Icra*, volume 3, pages 1691–1696.
- Engel, J., Koltun, V., and Cremers, D. (2018). Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625.
- Engel, J., Schöps, T., and Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. In Fleet, D., Pajdla, T., Schiele, B., and Tuytelaars, T., editors, *Computer Vision – ECCV 2014*, pages 834–849, Cham. Springer International Publishing.
- Engel, J., Sturm, J., and Cremers, D. (2012). Camera-based navigation of a low-cost quadcopter. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2815–2821. IEEE.
- Engel, J., StÅžckler, J., and Cremers, D. (2015). Large-scale direct slam with stereo cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942.
- Everingham, M., Eslami, S. M. A., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136.

- Eynard, D., Vasseur, P., Demonceaux, C., and Fršmont, V. (2010). Uav altitude estimation by mixed stereoscopic vision. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 646–651.
- Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2017). On-manifold preintegration for real-time visual-inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21.
- Forster, C., Pizzoli, M., and Scaramuzza, D. (2014). Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22.
- Fu, C., Carrio, A., and Campoy, P. (2015). Efficient visual odometry and mapping for unmanned aerial vehicle using arm-based stereo vision pre-processing system. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 957–962.
- Furrer, F., Burri, M., Achtelik, M., and Siegwart, R. (2016). *Robot Operating System (ROS): The Complete Reference (Volume 1)*, chapter RotorS—A Modular Gazebo MAV Simulator Framework, pages 595–625. Springer International Publishing, Cham.
- Garcia, M. A. and Solanas, A. (2004). 3d simultaneous localization and modeling from stereo vision. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 1, pages 847–853 Vol.1.
- Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., and Marín-Jiménez, M. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280 – 2292.
- Gat, E. (1998). On three-layer architectures. In Kortenkamp, D., Bonasso, R. P., and Murphy, R., editors, *Artificial Intelligence and Mobile Robots*. AAAI Press.
- Gawel, A., Del Don, C., Siegwart, R., Nieto, J., and Cadena, C. (2018). X-view: Graph-based semantic multi-view localization. *IEEE Robotics and Automation Letters*, 3(3):1687–1694.
- Gomez-Ojeda, R., Moreno, F.-A., Zuñiga-Noël, D., Scaramuzza, D., and Gonzalez-Jimenez, J. (2019). Pl-slam: a stereo slam system through the combination of points and line segments. *IEEE Transactions on Robotics*.
- Greene, W. N., Ok, K., Lommel, P., and Roy, N. (2016). Multi-level mapping: Real-time dense monocular slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 833–840.
- Grinvald, M., Furrer, F., Novkovic, T., Chung, J. J., Cadena, C., Siegwart, R., and Nieto, J. (2019). Volumetric Instance-Aware Semantic Mapping and 3D Object Discovery. *IEEE Robotics and Automation Letters*, 4(3):3037–3044.

- Grzonka, S., Grisetti, G., and Burgard, W. A Fully Autonomous Indoor Quadrotor.
- He, K., Gkioxari, G., Dollár, P., and Girshick, R. B. (2017). Mask R-CNN. *CoRR*, abs/1703.06870.
- Holzer, S., Rusu, R. B., Dixon, M., Gedikli, S., and Navab, N. (2012). Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2684–2689. IEEE.
- Howard, A. (2008). Real-time stereo visual odometry for autonomous ground vehicles. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3946–3952.
- Huang, A. S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., and Roy, N. (2017). *Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera*, pages 235–252. Springer International Publishing, Cham.
- Iocchi, L., Konolige, K., and Bajracharya, M. (2001). Visually realistic mapping of a planar environment with stereo. In *Experimental Robotics VII*, pages 521–532. Springer.
- Jianbo Shi and Tomasi (1994). Good features to track. In *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600.
- Karamat, T. B., Lins, R. G., Givigi, S. N., and Noureldin, A. (2018). Novel ekf-based vision/inertial system integration for improved navigation. *IEEE Transactions on Instrumentation and Measurement*, 67(1):116–125.
- Kaufmann, E., Gehrig, M., Foehn, P., Ranftl, R., Dosovitskiy, A., Koltun, V., and Scaramuzza, D. (2018). Beauty and the beast: Optimal methods meet learning for drone racing. *arXiv preprint arXiv:1810.06224*.
- Keller, M., Lefloch, D., Lambers, M., Izadi, S., Weyrich, T., and Kolb, A. (2013). Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision - 3DV 2013*, pages 1–8.
- Kerl, C., Sturm, J., and Cremers, D. (2013a). Dense visual slam for rgb-d cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2100–2106.
- Kerl, C., Sturm, J., and Cremers, D. (2013b). Robust odometry estimation for rgb-d cameras. In *2013 IEEE International Conference on Robotics and Automation*, pages 3748–3754.
- Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234.

- Koenig, N. and Howard, A. (2004). Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3.
- Kohlbrecher, S., Meyer, J., von Stryk, O., and Klingauf, U. (2011). A flexible and scalable slam system with full 3d motion estimation. In *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE.
- Kortenkamp, D. and Simmons, R. (2008). *Robotic Systems Architectures and Programming*, pages 187–206. Springer Berlin Heidelberg, Berlin, Heidelberg.
- Kumar, G. A., Patil, A. K., Patil, R., Park, S. S., and Chai, Y. H. (2017). A lidar and imu integrated indoor navigation system for uavs and its application in real-time pipeline classification. In *Sensors*.
- Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., and Burgard, W. (2011). g2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613. IEEE.
- Labbe, M. and Michaud, F. (2019). Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446.
- Lange, S., Sünderhauf, N., Neubert, P., Drews, S., and Protzel, P. (2012). Autonomous corridor flight of a uav using a low-cost and light-weight rgb-d camera. In Rückert, U., Joaquin, S., and Felix, W., editors, *Advances in Autonomous Mini Robots*, pages 183–192, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., and Furgale, P. (2015). Keyframe-based visual-inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334.
- Lhuillier, M. (2005). Automatic structure and motion using a catadioptric camera.
- Li, M. and Mourikis, A. I. (2013). High-precision, consistent ekf-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711.
- Lianos, K.-N., Schonberger, J. L., Pollefeys, M., and Sattler, T. (2018). Vso: Visual semantic odometry. In *The European Conference on Computer Vision (ECCV)*.
- Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.
- Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision (darpa).

- Ma, W., Wang, S., Brubaker, M. A., Fidler, S., and Urtasun, R. (2017). Find your way by observing the sun and other semantic cues. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299.
- Manthe, S., Carrio, A., Neuhaus, F., Campoy, P., and Paulus, D. (2018). Combining 2d to 2d and 3d to 2d point correspondences for stereo visual odometry. In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP*, pages 455–463.
- Marder-Eppstein, E., Berger, E., Foote, T., Gerkey, B., and Konolige, K. (2010). The office marathon: Robust navigation in an indoor office environment. In *2010 IEEE International Conference on Robotics and Automation*, pages 300–307.
- Matthies, L. H. (1989). *Dynamic Stereo Vision*. PhD thesis, Pittsburgh, PA, USA. AAI9023429.
- McCormac, J., Clark, R., Bloesch, M., Davison, A. J., and Leutenegger, S. (2018). Fusion++: Volumetric object-level SLAM. *CoRR*, abs/1808.08378.
- Meier, L., Tanskanen, P., Fraundorfer, F., and Pollefeys, M. (2011). Pixhawk: A system for autonomous flight using onboard computer vision. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 2992–2997. IEEE.
- Merten, H. (2008). The three-dimensional normal-distributions transform. *threshold*, 10:3.
- Molina, M., A. Suarez-Fernandez, R., Sampedro Perez, C., Sanchez-Lopez, J., and Campoy, P. (2017). Tml: a language to specify aerial robotic missions for the framework aerostack. *International Journal of Intelligent Computing and Cybernetics*, 10:00–00.
- Molina, M., Camporredondo, A., Bavle, H., Rodriguez-Ramos, A., and Campoy, P. (2019). An execution control method for the aerostack aerial robotics framework. *Frontiers of Information Technology & Electronic Engineering*, 20(1):60–75.
- Molina, M., Diaz-Moreno, A., Palacios, D., Suarez-Fernandez, R. A., Sanchez-Lopez, J. L., Sampedro, C., Bavle, H., and Campoy, P. (2016). Specifying complex missions for aerial robotics in dynamic environments. In *International Micro Air Vehicle Conference and Competition, IMAV 2016*, Beijing, China.
- Mondragon, I. F., Campoy, P., Martinez, C., and Olivares-Mendez, M. A. (2010). 3d pose estimation based on planar object tracking for uavs control. In *2010 IEEE International Conference on Robotics and Automation*, pages 35–41.
- Moore, T. and Stouch, D. (2014). A generalized extended kalman filter implementation for the robot operating system. In *IAS*.
- Moravec, H. (1980). Obstacle avoidance and navigation in the real world by a seeing robot rover. Technical Report CMU-RI-TR-80-03, Carnegie Mellon University, Pittsburgh, PA.

- Mourikis, A. I. and Roumeliotis, S. I. (2007). A multi-state constraint kalman filter for vision-aided inertial navigation. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3565–3572.
- Mur-Artal, R. and Tardos, J. D. (2016). Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras. *arXiv preprint arXiv:1610.06475*.
- Murali, V., Chiu, H., Samarasekera, S., and Kumar, R. T. (2017). Utilizing semantic visual landmarks for precise vehicle navigation. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8.
- Natraj, A., Demonceaux, C., Vasseur, P., and Sturm, P. (2011). Vision based attitude and altitude estimation for uavs in dark environments. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4006–4011.
- Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohi, P., Shotton, J., Hodges, S., and Fitzgibbon, A. (2011a). Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136.
- Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011b). Dtam: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*, pages 2320–2327.
- Nieuwenhuisen, M., Beul, M., Rosu, R. A., Quenzel, J., Pavlichenko, D., Houben, S., and Behnke, S. (2017a). Collaborative object picking and delivery with a team of micro aerial vehicles at mbzirc. In *2017 European Conference on Mobile Robots (ECMR)*, pages 1–6.
- Nieuwenhuisen, M., Quenzel, J., Beul, M., Droschel, D., Houben, S., and Behnke, S. (2017b). Chimneyspector: Autonomous mav-based indoor chimney inspection employing 3d laser localization and textured surface reconstruction. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 278–285.
- Nikolic, J., Burri, M., Rehder, J., Leutenegger, S., Huerzeler, C., and Siegwart, R. (2013). A uav system for inspection of industrial facilities. In *2013 IEEE Aerospace Conference*, pages 1–8.
- Nistér, D. (2004). An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6):756–777.
- Nister, D., Naroditsky, O., and Bergen, J. (2004). Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I.
- ozaslan, T., Loianno, G., Keller, J., Taylor, C. J., Kumar, V., Wozencraft, J. M., and Hood, T. (2017). Autonomous navigation and mapping for inspection of penstocks and tunnels with mavs. *IEEE Robotics and Automation Letters*, 2(3):1740–1747.

- ozaslan, T., Mohta, K., Keller, J., Mulgaonkar, Y., Taylor, C. J., Kumar, V., Wozencraft, J. M., and Hood, T. (2016). Towards fully autonomous visual inspection of dark featureless dam penstocks using mavs. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4998–5005.
- Papachristos, C., Khattak, S., Mascarich, F., and Alexis, K. (2019). Autonomous navigation and mapping in underground mines using aerial robots. In *2019 IEEE Aerospace Conference*, pages 1–8.
- Parkhiya, P., Khawad, R., Murthy, J. K., Bhowmick, B., and Krishna, K. M. (2018). Constructing category-specific models for monocular object-slam. *CoRR*, abs/1802.09292.
- Pestana, J., Mellado-Bataller, I., Fu, C., Sanchez-Lopez, J. L., Mondragon, I. F., and Campoy, P. (2013a). A general purpose configurable navigation controller for micro aerial multirotor vehicles. In *ICUAS 2013*.
- Pestana, J., Mellado-Bataller, I., Sanchez-Lopez, J. L., Fu, C., Mondragón, I. F., and Campoy, P. (2014a). A general purpose configurable controller for indoors and outdoors gps-denied navigation for multirotor unmanned aerial vehicles. *Journal of Intelligent & Robotic Systems*, 73(1-4):387–400.
- Pestana, J., Sanchez-Lopez, J., Campoy, P., and Saripalli, S. (2013b). Vision based gps-denied object tracking and following for unmanned aerial vehicles. In *Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on*, pages 1–6.
- Pestana, J., Sanchez-Lopez, J., de la Puente, P., Carrio, A., and Campoy, P. (2014b). A vision-based quadrotor swarm for the participation in the 2013 international micro air vehicle competition. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 617–622.
- Pestana, J., Sanchez-Lopez, J., Saripalli, S., and Campoy, P. (2014c). Computer vision based general object following for gps-denied multirotor unmanned vehicles. In *American Control Conference (ACC), 2014*, pages 1886–1891.
- Pestana, J., Sanchez-Lopez, J. L., de la Puente, P., Carrio, A., and Campoy, P. (2014). A vision-based quadrotor swarm for the participation in the 2013 international micro air vehicle competition. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 617–622.
- Pestana, J., Sanchez-Lopez, J. L., de la Puente, P., Carrio, A., and Campoy, P. (2015). A vision-based quadrotor multi-robot solution for the indoor autonomy challenge of the 2013 international micro air vehicle competition. *Journal of Intelligent & Robotic Systems*, pages 1–20.
- Pire, T., Fischer, T., Civera, J., De Cristoforis, P., and Berlles, J. J. (2015). Stereo parallel tracking and mapping for robot localization. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1373–1378.

- Pizzoli, M., Forster, C., and Scaramuzza, D. (2014). Remode: Probabilistic, monocular dense reconstruction in real time. *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2609–2616.
- Qin, T., Li, P., and Shen, S. (2018). Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020.
- Quenzel, J., Nieuwenhuisen, M., Droschel, D., Beul, M., Houben, S., and Behnke, S. (2019). Autonomous mav-based indoor chimney inspection with 3d laser localization and textured surface reconstruction. *Journal of Intelligent & Robotic Systems*, 93(1):317–335.
- Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., and Ng, A. Y. (2009). Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5.
- Radwan, N., Valada, A., and Burgard, W. (2018). Vlocnet++: Deep multitask learning for semantic visual localization and odometry. *IEEE Robotics and Automation Letters*, 3(4):4407–4414.
- Redmon, J. and Farhadi, A. (2016). YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242.
- Rodriguez-Ramos, A., Sampedro, C., Bavle, H., Milosevic, Z., Garcia-Vaquero, A., and Campoy, P. (2017). Towards fully autonomous landing on moving platforms for rotary unmanned aerial vehicles. In *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 170–178.
- Rosinol, A., Sattler, T., Pollefeys, M., and Carlone, L. (2019). Incremental visual-inertial 3d mesh generation with structural regularities. *CoRR*, abs/1903.01067.
- Salas-Moreno, R. F., Newcombe, R. A., Strasdat, H., Kelly, P. H. J., and Davison, A. J. (2013). Slam++: Simultaneous localisation and mapping at the level of objects. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359.
- Sampedro, C., Bavle, H., Rodriguez-Ramos, A., Carrio, A., Suárez Fernández, R. A., Sanchez-Lopez, J. L., and Campoy, P. (2017). A fully-autonomous aerial robotic solution for the 2016 international micro air vehicle competition. In *Unmanned Aircraft Systems (ICUAS), 2017 International Conference on*. IEEE.
- Sampedro, C., Bavle, H., Sanchez-Lopez, J. L., Fernandez, R. A. S., Rodriguez-Ramos, A., Molina, M., and Campoy, P. (2016). A flexible and dynamic mission planning architecture for uav swarm coordination. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 355–363.

- Sampedro, C., Rodriguez-Ramos, A., Bavle, H., Carrio, A., de la Puente, P., and Campoy, P. (2018). A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques. *Journal of Intelligent & Robotic Systems*.
- Sanchez-Lopez, J., Pestana, J., de la Puente, P., Suarez-Fernandez, R., and Campoy, P. (2014). A system for the design and development of vision-based multi-robot quadrotor swarms. In *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pages 640–648.
- Sanchez-Lopez, J. L., Molina, M., Bavle, H., Sampedro, C., Suárez Fernández, R. A., and Campoy, P. (2017a). A multi-layered component-based approach for the development of aerial robotic systems: The aerostack framework. *Journal of Intelligent & Robotic Systems*, 88(2):683–709.
- Sanchez-Lopez, J. L., Molina, M., Bavle, H., Sampedro, C., Suárez Fernández, R. A., and Campoy, P. (2017b). A multi-layered component-based approach for the development of aerial robotic systems: The aerostack framework. *Journal of Intelligent & Robotic Systems*, 88(2):683–709.
- Sanchez-Lopez, J. L., Pestana, J., Collumeau, J., Suarez-Fernandez, R., Campoy, P., and Molina, M. (2015). A vision based aerial robot solution for the mission 7 of the international aerial robotics competition. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1391–1400.
- Sanchez-Lopez, J. L., Pestana, J., de la Puente, P., Suarez-Fernandez, R., and Campoy, P. (2014). A system for the design and development of vision-based multi-robot quadrotor swarms. In *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 640–648.
- Sanchez-Lopez, J. L., Pestana, J., Puente, P., and Campoy, P. (2015). A reliable open-source system architecture for the fast designing and prototyping of autonomous multi-uav systems: Simulation and experimentation. *Journal of Intelligent & Robotic Systems*, pages 1–19.
- Sanchez-Lopez, J. L., Suarez-Fernandez, R., Bavle, H., Sampedro, C., Molina, M., and Campoy, P. (2016). Aerostack: An architecture and open-source software framework for aerial robotics. In *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, page 0.
- Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry [tutorial]. *IEEE Robotics Automation Magazine*, 18(4):80–92.
- Sim, R., Elinas, P., Griffin, M., Little, J. J., et al. (2005). Vision-based slam using the rao-blackwellised particle filter. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, volume 14, pages 9–16.

- Steinbrücker, F., Sturm, J., and Cremers, D. (2011). Real-time visual odometry from dense rgb-d images. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 719–722. IEEE.
- Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE.
- Sünderhauf, N., Pham, T., Latif, Y., Milford, M., and Reid, I. D. (2016). Meaningful maps - object-oriented semantic mapping. *CoRR*, abs/1609.07849.
- Szafranski, G., Czyba, R., Janusz, W., and Blotnicki, W. (2013). Altitude estimation for the uav’s applications based on sensors fusion algorithm. *2013 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 508–515.
- Trevor, A. J., Gedikli, S., Rusu, R. B., and Christensen, H. I. (2013). Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration (SPME)*.
- Trevor, A. J. B., Rogers, J. G., and Christensen, H. I. (2012). Planar surface slam with 3d and 2d sensors. In *2012 IEEE International Conference on Robotics and Automation*, pages 3041–3048.
- Tropicchio, P., Satler, M., Unetti, M., and Avizzano, C. A. (2018). Confined spaces industrial inspection with micro aerial vehicles and laser range finder localization. *International Journal of Micro Air Vehicles*, 10(2):207–224.
- Van de Loosdrecht, J., Dijkstra, K., Postma, J., Keuning, W., and Bruin, D. (2014). Twirre: Architecture for autonomous mini-uavs using interchangeable commodity components. In *IMAV 2014: International Micro Air Vehicle Conference and Competition 2014, Delft, The Netherlands, August 12-15, 2014*. Delft University of Technology.
- Wang, F., Wang, K., Lai, S., Phang, S. K., Chen, B. M., and Lee, T. H. (2014). An efficient uav navigation solution for confined but partially known indoor environments. In *11th IEEE International Conference on Control Automation (ICCA)*, pages 1351–1356.
- Whelan, T., Johannsson, H., Kaess, M., Leonard, J. J., and McDonald, J. (2013). Robust real-time visual odometry for dense rgb-d mapping.
- Whelan, T., Kaess, M., Fallon, M., Johannsson, H., Leonard, J. J., and McDonald, J. (2012). Kintinuous: Spatially extended kinectfusion.
- Whelan, T., Leutenegger, S., Salas-Moreno, R., Glocker, B., and Davison, A. (2015). Elasticfusion: Dense slam without a pose graph. *Robotics: Science and Systems*.
- Xiang, Y., Schmidt, T., Narayanan, V., and Fox, D. (2017). Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *CoRR*, abs/1711.00199.

BIBLIOGRAPHY

- Zhang, Z., Deriche, R., Faugeras, O., and Luong, Q.-T. (1995). A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial intelligence*, 78(1-2):87–119.
- Zhou, X., Wang, D., and Krähenbühl, P. (2019). Objects as points. *CoRR*, abs/1904.07850.