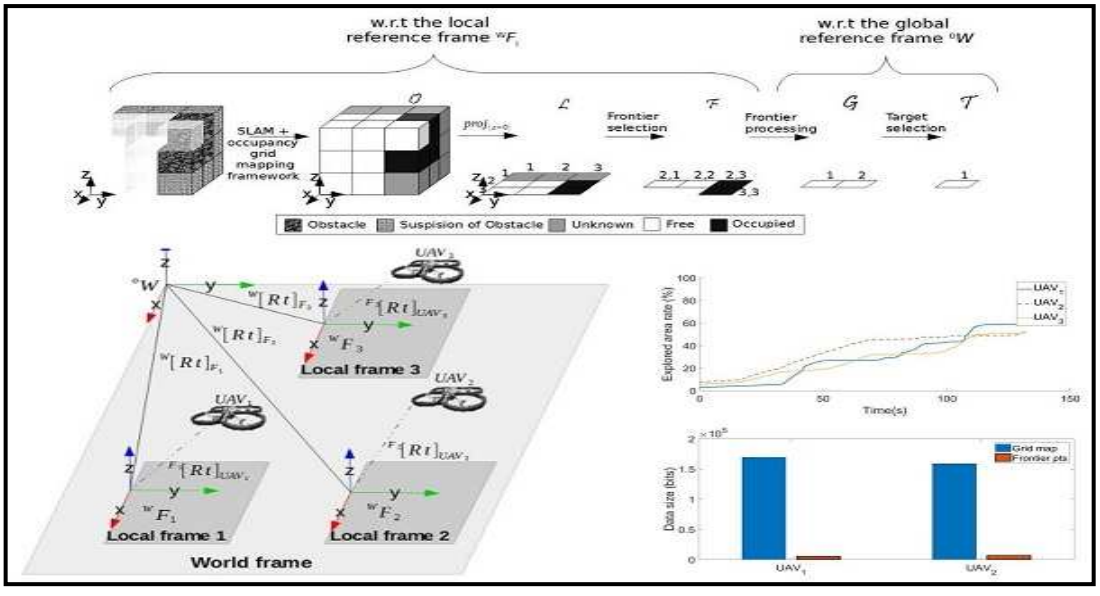
Communicating multi-UAV system for cooperative SLAM-based exploration

Nesrine Mahdoui Chedly

Par **Nesrine MAHDOUI CHEDLY**

Communicating multi-UAV system for cooperative SLAM-based exploration

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 7 décembre 2018
Spécialité : Technologies de l'Information et des Systèmes : Unité de recherche Heudyasic (UMR-7253)

D2447

Communicating Multi-UAV System for cooperative SLAM-based Exploration

Nesrine MAHDOUI CHEDLY

Spécialité : Technologies de l'Information et des Systèmes

Defended on December 7th, 2018. Thesis Committee:

Reviewers:

Olivier SIMONIN	*Ouiddad LABBANI-IGBIDA*
Professor	Professor
INSA Lyon	Univ de Limoges

Examiners:

Véronique CHERFAOUI	*Isabelle FANTONI*
Professor	Director CNRS-LS2N
Univ de Technologie de Compiègne	Centrale Nantes

Supervisors:

Vincent FRÉMONT	*Enrico NATALIZIO*
Professor	Professor
Centrale Nantes	Université de Lorraine

Université de Technologie de Compiègne

Heudiasyc Laboratory, UMR CNRS 7253

07 - 12 - 2018

Communicating Multi-UAV System for cooperative SLAM-based Exploration

Nesrine MAHDOUI CHEDLY

Spécialité : Technologies de l'Information et des Systèmes

Thèse soutenue le 07 Décembre 2018 devant le jury composé de :

Rapporteurs:

Olivier SIMONIN
Professeur des universités
INSA Lyon

Ouiddad LABBANI-IGBIDA
Professeur des universités
Univ de Limoges

Examinateurs:

Véronique CHERFAOUI
Professeur des universités
Univ de Technologie de Compiègne

Isabelle FANTONI
Directrice CNRS-LS2N
Centrale Nantes

Directeurs de Thèse:

Vincent FRÉMONT
Professeur des universités
Centrale Nantes

Enrico NATALIZIO
Professeur des universités
Université de Lorraine

Université de Technologie de Compiègne

Laboratoire Heudiasyc UMR CNRS 7253

07 - 12 - 2018

Contents

List of figures

List of tables

Acknowledgment

I'm so grateful to my supervisors Prof. Vincent Frémont and Prof. Enrico Natalizio for their patience, their continuous encouragement and their constructive advices. They gave me precious directives so that this work come into existence.

I am also grateful to Prof. Ouiddad LABBANI-IGBIDA and Prof. Olivier SIMONIN for accepting being the reviewers of this thesis. Also, I would like to thank Prof. Véronique CHERFAOUI and Prof. Isabelle FANTONI for agreeing to be part of my thesis committee.

I would like to thank all Heudiasyc members, and the SyRI team for their support and kindness. I'm also thankful to my colleagues especially Asma ben said and Arbia Sfar, that become true friends.

Moreover, I thank the Labex for their financial support and for providing the necessary for the smooth running of this thesis.

My gratitude goes to my family without whom, this thesis would not be achieved in good conditions. They provided me continuous support, motivations and positive vibes. My special and heartily thanks goes to my husband who encourages me to always go further.

This thesis is heartily dedicated to my father who took the lead to heaven before the completion of this work.

List of symbols

UAV_i	UAV of index i.
$\mathbf{p}_i$, v_i, ω_i	Pose, linear velocity, and angular velocity of UAV_i.
${}^0W$	Global reference frame.
${}^W F_i$	UAV_i's local reference frame w.r.t ${}^0W$.
${}^W\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}_{F_i}$	Transform of rotation $\mathbf{R}$ and translation $\mathbf{t}$ from reference frame ${}^0W$ to ${}^W F_i$.
$\mathcal{P}_S$, $\mathcal{O}$, $\mathcal{L}$, $\mathcal{F}$, $\mathcal{G}$, $\mathcal{T}$, $\mathcal{C}$	3D points cloud computed by SLAM, 3D voxels, 2D cells, frontier points, candidate frontier points (candidate targets), assigned targets and cluster sets, respectively.
$\mathbf{o}_u$, $\mathbf{o}_f$, $\mathbf{o}_o$	Unknown, free and occupied 3D voxels, respectively.
$\mathbf{l}_u$, $\mathbf{l}_f$, $\mathbf{l}_o$	Unknown, free and occupied 2D cells, respectively.
$\mathbf{f}_{i,j}$	Frontier point j of UAV_i
$\mathbf{t}_i$	Target point i.
$\boldsymbol{I}(\mathbf{t}_i)$	Information gain of $\mathbf{t}_i$.
$\boldsymbol{U}(UAV_i, \mathbf{t}_j)$	Utility of reaching target j by UAV_i.
$\theta(i, j)$	Assignment of UAV_i with target j.
id	Identification number of UAV.
r	Loop rate.
s	Sensor maximum range.
λ	Tuning parameter $\in [0, 1]$.
n, n_c	Number of UAVs in the fleet and in $\mathcal{C}$, respectively.
$n_t, n_g,$	Number of targets in $\mathcal{T}$ and in $\mathcal{G}$, respectively.
n_i	Number of frontier points of UAV_i.
$[r_{min}, r_{max}]$	Range to schedule information gain.
σ_x, σ_y	Parameter to spread the blob in x and y axis, respectively.
d_{tot}	Average of distances of other UAVs to the considered target.
$\mathcal{T}_{0:n}, \mathcal{C}_{0:n}, \mathcal{P}_{0:k}, \mathcal{U}_{0:k}$	Camera motions, camera poses, robot poses, and control vectors sets, respectively.
$\mathbf{T}_i, \mathbf{C}_i, \mathbf{p}_i, \mathbf{u}_i$	Camera motion, camera pose, robot pose, and control vector, respectively.

$\mathcal{M}_{0:k}, \mathcal{Z}_{0:k,i}, \mathcal{B}_{0:n-1}$	States of landmarks, observations of landmarks, and odometry measurements sets, respectively.
$\mathbf{m}_i, \mathbf{z}_{1,i}, \mathbf{b}_i$	State of landmark, observation of landmark, and odometry measurement, respectively.
$I_t, \mathbf{C}_t, \mathbf{D}_t$	image, colored image, and depth image, respectively.
$\mathbf{M}_1$	Occupancy grid map.
$\mathbf{T}_k$	Rigid body transformation.
$\boldsymbol{F}(\theta)$	Graph factorization function .

Acronyms

AP:	Access Point
AODV:	Ad-hoc On-demand Distance Vector
BBB:	Beagle Bone Black
BATMAN:	Better Approach to Mobile Adhoc Networking
BLE:	Broadcast of Local Eligibility
CFG:	Constrained Factor Graph
CSFM:	Collaborative Structure From Motion
DDF-SAM:	Decentralized Data Fusion-SAM
DVO:	Dense Visual Odometry
DEMO:	Depth Enhanced Monocular Odometry
EKF:	Extended Kalman Filter
FANET:	Flying Ad hoc NETworks
GPU:	Graphic Processing Unit
GPS:	Global Positioning System
HWMP:	Hybrid Wireless Mesh Protocol
iSAM2:	Incremental Smoothing and Mapping 2
IMU:	Inertial Measurement Unit
LSD-SLAM:	Large-Scale Direct monocular SLAM
LED:	based on Light Emitting Diodes
MANET:	Mobile Ah doc NETwork
MTSP:	Multiple Traveling Salesman Problem
MAV:	Micro Aerial Vehicle
MRS:	Multi-Robot System
NASA:	National Aeronautics and Space Administration
OSPF:	Open Shortest Path First
OGM:	OriGinator Messages
OLSR:	Optimized Link State Routing
PTAM:	Parallel Tracking and Mapping
RMSE:	Root Mean Square Error
ROVIO:	RObust Visual Inertial Odometry

ROVIOLI:	ROVIO with Localization Integration
RSS:	Received Signal Strength
RX:	Receiving mode
ROS:	Robot Operating System
SMR:	SystÃĺmes Multi-Robot
SoS:	System of Systems
SLAM:	Simultaneous Localization And Mapping
SVO:	Semi-Direct Monocular Visual Odometry
SFM:	Structure From Motion
SAM:	Smoothing And Mapping
TX:	Transmitting mode
UAV:	Unmanned Aerial Vehicle
VANET:	Vehicular Ah doc NETwork
VLC:	Visible Light Communication
VTAL:	Vertical Taking off And Landing
VI:	Visual Inertial
VO:	Visual Odometry
WDS:	Wireless Distribution System

Résumé

Dans la communauté robotique aérienne, un croissant intérêt pour les systèmes multi-robot (SMR) est apparu ces dernières années. Cela a été motivé par *i)* les progrès technologiques, tels que de meilleures capacités de traitement à bord des robots et des performances de communication plus élevées, et *ii)* les résultats prometteurs du déploiement de SMR tels que l'augmentation de la zone de couverture en un minimum de temps. Le développement d'une flotte de véhicules aériens sans pilote (UAV: Unmanned Aerial Vehicle) et de véhicules aériens de petite taille (MAV: Micro Aerial Vehicle) a ouvert la voie à de nouvelles applications à grande échelle nécessitant les caractéristiques de tel système de systèmes dans des domaines tels que la sécurité, la surveillance des catastrophes et des inondations, la recherche et le sauvetage, l'inspection des infrastructures, et ainsi de suite. De telles applications nécessitent que les robots identifient leur environnement et se localisent. Ces tâches fondamentales peuvent être assurées par la mission d'exploration. Dans ce contexte, cette thèse aborde l'exploration coopérative d'un environnement inconnu en utilisant une équipe de drones avec vision intégrée. Nous avons proposé un système multi-robot où le but est de choisir des régions spécifiques de l'environnement à explorer et à cartographier simultanément par chaque robot de manière optimisée, afin de réduire le temps d'exploration et, par conséquent, la consommation d'énergie. Chaque UAV est capable d'effectuer une localisation et une cartographie simultanées (SLAM: Simultaneous Localization And Mapping) à l'aide d'un capteur visuel comme principale modalité de perception. Pour explorer les régions inconnues, les cibles – choisies parmi les points frontière situés entre les zones libres et les zones inconnues – sont assignées aux robots en considérant un compromis entre l'exploration rapide et l'obtention d'une carte détaillée. À des fins de prise de décision, les UAVs échangent habituellement une copie de leur carte locale, mais la nouveauté dans ce travail est d'échanger les points frontière de cette carte, ce qui permet d'économiser la bande passante de communication. L'un des points les plus difficiles du SMR est la communication inter-robot. Nous étudions cette partie sous les aspects topologiques et typologiques. Nous proposons également des stratégies pour faire face à l'abandon ou à l'échec de la communication. Des validations basées sur des simulations étendues et des bancs d'essai sont présentées.

Mots Clés : *coordination de système multi-robot , exploration autonome , exploration basée sur les frontières , SLAM , communication inter-robot.*

Abstract

In the aerial robotic community, a growing interest for Multi-Robot Systems (MRS) appeared in the last years. This is thanks to *i)* the technological advances, such as better onboard processing capabilities and higher communication performances, and *ii)* the promising results of MRS deployment, such as increased area coverage in minimum time. The development of highly efficient and affordable fleet of Unmanned Aerial Vehicles (UAVs) and Micro Aerial Vehicles (MAVs) of small size has paved the way to new large-scale applications, that demand such System of Systems (SoS) features in areas like security, disaster surveillance, inundation monitoring, search and rescue, infrastructure inspection, and so on. Such applications require the robots to identify their environment and localize themselves. These fundamental tasks can be ensured by the exploration mission. In this context, this thesis addresses the cooperative exploration of an unknown environment sensed by a team of UAVs with embedded vision. We propose a multi-robot framework where the key problem is to cooperatively choose specific regions of the environment to be simultaneously explored and mapped by each robot in an optimized manner in order to reduce exploration time and, consequently, energy consumption. Each UAV is able to perform Simultaneous Localization And Mapping (SLAM) with a visual sensor as the main input sensor. To explore the unknown regions, the targets – selected from the computed frontier points lying between free and unknown areas – are assigned to robots by considering a trade-off between fast exploration and getting detailed grid maps. For the sake of decision making, UAVs usually exchange a copy of their local map; however, the novelty in this work is to exchange map frontier points instead, which allow to save communication bandwidth. One of the most challenging points in MRS is the inter-robot communication. We study this part in both topological and typological aspects. We also propose some strategies to cope with communication drop-out or failure. Validations based on extensive simulations and testbeds are presented.

Keywords: *coordinated multi-robot, autonomous exploration, frontier-based exploration, SLAM, inter-robot communication.*

Introduction

Contents

1 Background

Unmanned Aerial Vehicles (UAVs) are flying robots or aircraft without a pilot or passengers. They can be remotely controlled by human or fully automated (drones). The history of UAVs, more generally robots, is closely linked to the technological feat. Historically, robots' usage has been limited to factory, that is for industrial use. In the early 20th century, the notion of using robots for other purposes arise. Indeed, the military field starts showing an interest for robotics and more particularly, for pilot-less Aerial Vehicles. One of the first radio controlled model has been developed and tested successfully by the US army during the first World War (See figure 1). Development and testing of radio-controlled aircraft increase significantly since 1935, and the term of drone starts to be used. One among the first enduring drone prototype was developed in 1946. It successfully flew remotely from Hilo Naval Air Station in Hawaii to Muroc Army Air Field in California. Since that, drones have become popular and started being used in the civilian applications. Consequently, the drone's size has considerably reduced. In early 1990, miniature UAVs, also called Micro Aerial Vehicles (MAV), have become available. From their introduction, these small sized flying robots have experienced a great and an increasing interest until now. There are several types of drones with a different typology for each. The type is defined with the number of rotors. A drone with one rotor is called helicopter; whereas, multicopters are those that are equipped with one

Figure 1 – Drone prototype[10].

[10]Source: `https://www.iwm.org.uk/history/a-brief-history-of-drones`

or more rotors. Among them, let's cite the Bicopter with two rotors, the Tricopter with three rotors in "T" or "Y" shapes, the famous and mainly used Quadcopter composed of four rotors in "+" or "x" configurations, the Hexacopter with six rotors, the Octocopter with eight rotors, and so on. The drones are equipped with on-board sensors to sense their environment. These sensors can be exteroceptive such as camera and radar, or proprioceptive such as Global Positioning System (GPS) and Inertial Measurement Unit (IMU).

2 Motivation

Having made a great progress in finding solutions for basic problems related to single robot, the research community is now interested in studying coordinated multi-robot systems. Cooperative multiple UAVs can remarkably increase the fleet's performances compared to a single UAV. In addition, several large scale applications require such system features in areas like security, disaster surveillance, inundation monitoring, search and rescue, infrastructure inspection, and so on.

Nevertheless, although the important advances in the deployment of a team of UAVs, such operations still present some challenges. Indeed, in the past few years multi-robot exploration state of the art focused on motion planning and collision avoidance [Latombe, 1991, Fujimura and Singh, 1996, Bennewitz and Burgard, 2000]. More recently, however, the emphasis in multi-robot exploration has been on coordination, cooperation and inter-robot communication [Yan et al., 2013, Dai et al., 2018, Min et al., 2018].

The first challenge in an unknown environment is to map the surrounding areas and to know the localization of the robots. Hence, an embedded simultaneous localization and mapping (SLAM) algorithm where *i)* no global positioning system is used [Andre et al., 2014, Heng et al., 2015] and *ii)* no initial known pose assumption is done [Yan et al., 2014], has to be adopted. Also, adequate embedded sensors have to be chosen to get reliable results.

Furthermore, when it comes to multi-UAV exploration mission, robots have to avoid to sense already explored areas, or to visit the same area at the same time. Hence, an exploration strategy has to be set up to coordinate between fleet members in order to simultaneously explore different areas in an optimized manner. Exploration strategies use mostly frontier based approaches introduced in [Burgard et al., 2000]. They consist on assigning a target – selected from the computed frontier points lying between free and unknown areas – taking into account the utility of reaching it. The utility function is defined in different manner depending on the mission's purpose.

Moreover, the target assignment process may be performed on-board by each robot [Sheng et al., 2006, Yuan et al., 2010] or on-board by a server/robot with enhanced capabilities, which makes decisions for the other robots in the fleet [Burgard et al., 2000, Schmuck, 2017].

To be able to have a cooperative behavior, fleet members have to collect information about each others while maintaining reliable wireless communications [Rooker and Birk,

2007]. Compared to a group of robots that do not involve communication, inter-robot data exchange is beneficial even if it is limited. However, communication is often subject to drop-out or failure, so the exploration strategy has to overcome these limitations. Still, mostly, literature [Burgard et al., 2005, Schuster et al., 2015] deals with MRS deployment while assuming an ideal communication or aims at keeping team members within range of one another in order to focus their attention on higher level problems. But considering communication losses and/or limited bandwidth help to prevent from mission failure and ensure a more realistic scenario.

Briefly, the main motivation for this thesis is to answer this question: How to efficiently coordinate a multi-UAV system in order to explore a bounded unknown environment while taking into account communication constraints?

3 Problem statement/Approach

In this thesis, an algorithm for coordinating a UAV fleet is presented in order to efficiently explore an unknown environment using an unknown number of potentially heterogeneous robots. The purpose is to investigate the following research directions:

- Using a multi-robot system for an efficient exploration.
- Performing localization and mapping.
- Improving area coverage using specific exploration strategies.
- Coordinating the fleet by using inter-robot communication.
- Treating the considered system as a System of Systems (SoS) to handle fleet scalability.

Multi-robot systems have shown their effectiveness to cooperatively explore an unknown environment. Though, some challenges arise to enhance the system performances such as: *i)* using a local localization algorithm, *ii)* choosing the right information to exchange, *iii)* making the robot-to-target assignment *iv)* taking into account communication network limitations.

In the context of a multi-robot system, this work proposes a Multi-UAV framework for cooperative exploration of an unknown environment. Each UAV is equipped with an embedded RGB-D sensor to perform grid-based Simultaneous Localization And Mapping (SLAM). The exploration strategy, based on the group-*leader* decision making, uses a novel utility function that takes into account the distance of each robot in the group from the unexplored set of targets, and permits to simultaneously explore the environment and get a detailed grid map of specific areas in an optimized manner. Usually, in a cooperative map construction task, robots exchange the whole copy of their individual maps. Whereas, in this work, the proposed novelty is to exchange only the frontier points of the computed local map to reduce the shared data volume, and, consequently, memory

consumption. Still, communication limitations have to be taken into account to ensure the accomplishment of the mission. The proposed strategy is designed to cope with communication drop-outs and failures.

Definitions and Assumptions

- The robots, throughout the thesis, are defined as flying robots with an on-board computation capacity and a visual sensor. They are potentially heterogeneous.
- The number of robots used for evaluation is limited to three, however, the proposed system architecture is not constrained to a fixed number of robots.
- The robots could be ground-based or aerial. They evolve in a 3D environment while maintaining a fixed z altitude which leads to a 2D exploration and navigation problem.
- The proposed system architecture is distributed and embedded over all fleet members. All robots in the fleet have the same computational capabilities.
- The robots in the fleet are autonomous. Each one is able to perform Simultaneous Localization And Mapping (SLAM), to plan a path to a target and to attempt to reach it.
- A group-*leader* is responsible for assigning tasks to others. The selected *leader* is not previously predefined.
- For exploration, a bounded simulated environment with no prior knowledge is used.
- Each robot has its own local reference frame.
- A global reference frame is defined such that it coincides with the first group-*leader*'s local reference frame.
- The inter-robot communication, available only in a specific range, is not assumed to be perfect.

4 Contributions and publications

The main contributions of this work are the following:

- To introduce a fully distributed Multi-UAV system architecture that does not exploit any global information (neither map nor GPS).
- To address the simultaneous Localization and Mapping problem using a monocular and an RGB-D cameras.

- To propose a novel utility function that takes into account the distance of each robot in the group from the unexplored set of targets. This function also makes a trade-off between fast exploration and getting a detailed grid map.
- To detail a coordinated exploration strategy, based on the group-*leader* decision making, that minimizes both the global exploration time and the average traveled distance by each UAV.
- To analyze the topology and typology of inter-UAV communication to cope with network limitations and failures.
- To present a technological solution that uses only a limited information exchange among UAVs. To the best of our knowledge, we are the first to propose to exchange frontier points instead of a whole copy of the local map.

Parts of the content presented in these chapters are the subject of the following publications:

- N. Mahdoui, E. Natalizio, and V. Frémont, "Multi-uavs network communication study for distributed visual simultaneous localization and mapping," in 2016 International Conference on Computing, Networking and Communications (ICNC). Kauai, Hawaii, USA : IEEE, Feb. 15-18, 2016, pp. 1 - 5.
- N. Mahdoui, V. Frémont, and E. Natalizio, "Cooperative exploration strategy for micro-aerial vehicles fleet," in 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2017). Daegu, Korea: IEEE, Nov. 16-18, 2017, pp. 1 - 6.
- N. Mahdoui, V. Frémont, and E. Natalizio, "Cooperative frontier-based exploration strategy for multi-robot system," in 2018 IEEE 13th System of Systems Engineering Conference (SoSE 2018). Paris, France: IEEE, Jun. 19-22, 2018, pp. 1 - 8.

5 Thesis pipeline

The rest of this thesis is organized as follows:

- **Chapter 1:** This chapter introduces an overview of a fully distributed Muti-UAV system architecture that does not exploit any global information. This framework is mainly composed of the following parts: SLAM, cooperative exploration strategy, and inter-robot communication. A more detailed description of each topic is addressed throughout chapters of this work.
- **Chapter 2:** This chapter considers the SLAM problem. The discussion is focused on using the visual sensor as the main perception modality. For the 3D pose estimation, two approaches are proposed: A graph-based inertial SLAM that fuses a monocular camera with an IMU; and a feature-based RGB-D SLAM. For each approach, simulation results are presented.

- **Chapter 3:** This chapter details relevant aspects related to exploration strategy, utility function, and robot-to-target assignment. A cooperative multi-robot exploration strategy, based on the group-*leader* decision making, is proposed. This strategy uses a novel utility function that makes a trade-off between fast exploration and getting a detailed grid map, and also takes into account the distance of other robots to the set of unexplored targets. Results are presented based on simulations.

- **Chapter 4:** The last chapter analyses the inter-robot communication problem for different topologies. It also studies strategies to face some communication limitations and proposes a solution that uses only limited information exchange among UAVs. Results based on testbeds are presented.

Multi-robot System Model

Contents

1 Introduction

In the recent years, researchers have become interested in using coordinated Multi-Robot System (MRS). Taking advantage of the great progress done in finding solutions for basic problems related to single robot, nowadays, studies are oriented to MRSs. The MRS is capable of implementing new features and executing new tasks over large scale environments such as disaster surveillance, inundation monitoring, environment discovery, infrastructure inspection, etc. One of the challenging problem for MRSs is to design an appropriate architecture for an efficient coordination of the fleet.

In this chapter, we study the state of the art of MRSs, and propose and detail a distributed multi-robot framework for unknown an environment exploration.

2 Related works

2.1 System of Systems

The term System of Systems (SoS) has been used since 1950s. The SoS concept is defined as "the arrangement of theoretical systems and constructs in a hierarchy of

complexity" [Boulding, 1956]. The term SoS was used to describe a set of independent and organized elements that interact with each other. These components act jointly to achieve a common goal. To distinguish SoS from conventional systems, more precise definition using characteristics are proposed. In [Maier, 1998], the SoS is identified by their Operational Independence, Managerial Independence, Geographic Distribution, Evolutionary Development, and Emergent Behavior. In [Boardman and Sauser, 2006], five features define a SoS: Autonomy, Belonging, Connectivity, Diversity, and Emerging. Since there is no single definition for SoS in the literature, authors in [Nielsen et al., 2015] classify the state of the art characterization of SoS in eight dimensions including: Autonomy, Independence, Distribution, Evolution, Dynamic reconfiguration, Emergence of behavior, Interdependence, and Interoperability. The definition of each characteristic may differ from one another. Autonomy in [Boardman and Sauser, 2006] refers to the capacity of a constituent system to pursue a specific purpose. While, according to [Maier, 1998], autonomy entails that constituents perform their own functions in accordance with their own rules. These classifications show that some dimensions are more frequent to use than others such as Emergence Behavior and Evolution. Also, in reality, these criteria are not totally satisfied.

According to the mentioned literature featuring, the multi-robot system is a SoS characterized by its:

- Autonomy: Achieving a specific task.
- Independence: Acting while being detached from the rest of the SoS.
- Distribution: Physical separation and network distribution.
- Evolution: Adapting itself to the surrounding environment.
- Dynamic behavior: Dynamic reconfiguration (modification of architecture) to ensure resilience of a SoS to faults.
- Emergent behavior: Resulting behavior from the SoS collaboration.
- Interdependence: Mutual dependencies are needed to meet the requirements of SoS since it is a trade of between Independence and interdependence where agents and their relations are interdependent.
- Interoperability: Incorporating a range of heterogeneous constituent systems and protocols.

2.2 Multi-robot system classification

MRS are characterized by their mission type, objective, behavior, environment, and so on. Thus MRS can be classified in different manner with no arbitrary characteristic. They can also be classified according to their degree of cooperation, coordination, robot type, inter-robot communication, and system architecture [Iocchi et al., 2000, Farinelli et al., 2004, Yan et al., 2013]. This list of classification axis is not exhaustive.

2.2.1 Degrees of cooperation

A system is said cooperative if its members operate with each other in order to achieve a common task. A classification of MRS, based on cooperation, is proposed in [Farinelli et al., 2004]. Depending on the knowledge that a robot may have about its neighbors when operating with them, a cooperative MRS can be aware or not. Indeed, a robot can be cooperative when it sends information or follows simple tasks that were defined before the beginning of the mission, while being unaware of what is going on around it. But it can also be directly or indirectly aware by dynamically adapting its behavior depending on other robots information.

2.2.2 Degrees of coordination

At a coordination level point of view, authors in [Yan et al., 2013] classify MRS into static or deliberative, if instructions are defined before the mission begin; and dynamic or active, if instructions are given during the execution of the task.

Authors in [Farinelli et al., 2004] define a coordinated system as a system where each robot takes into account the actions executed by the other robots. They proposed to classify the system's coordination level to strongly coordinated, weakly coordinated, and not coordinated at all; depending on the existence of a coordination protocol. This protocol defines a set of rules to follow in order to perform robot-to-robot and robot-to-environment interaction.

2.2.3 Robot types

In [Iocchi et al., 2000, Yan et al., 2013] two types of MRS are distinguished: Those composed of heterogeneous robots, with different capabilities, and those composed of homogeneous robots, with identical capabilities.

2.2.4 Inter-robot communication

The communication between robots is one of the most important feature that characterizes a MRS. Inter-robot communication in [Iocchi et al., 2000, Yan et al., 2013] is classified into direct or explicit, if robot shares explicitly information; and indirect or implicit, if the information is indirectly shared by altering the sensor or the environment. Authors in [Dudek et al., 1996] classified the communication in different axis including range, topology, and bandwidth.

2.2.5 System architecture

Regarding the MRS architecture and decision making, authors in [Yan et al., 2013] classify system architecture into centralized, those with a central server; and decentralized, those

with no central agent, including distributed and hierarchical architecture. In [Farinelli et al., 2004], MRS are distributed if agents are completely autonomous in their decision making, and centralized if one robot is in charge of organizing the work. Centralized approaches are also divided into weakly centralized, if more than one agent is allowed to become a leader; and strongly centralized if only one leader is designed to be the same during the mission.

These architectures present several advantages and disadvantages that are summarized in Table 1.1. The centralized architectures consider one robot/central server [Rocha et al., 2005] to manage all the computations and tasks assignment.However, they are subject to stranded mission when they do not take into account communications or robot failures. The distributed approaches use fully autonomous robots [Yuan et al., 2010, Sheng et al., 2006]. These approaches require robots with increased resources to exchange and process an important amount of information in order to synchronize agents and achieve a cooperative mission. Authors in [Wu and Zhang, 2012] propose a hybrid approach that consists in switching from individual to cooperative exploration behavior when agents are not able to converge to a local minimum at a satisfying rate. Important computation requirements are used for this later approach.

Table 1.1 – Multi-robot system architecture comparison.

Approach	**Centralized**	**Decentralized (distributed or hierarchical)**
Advantages	- Optimal solution. - Simple and lightweight processing on-board robots.	- Robustness in dynamic environment. - Reliability in case of other robots failure. - Adaptability and flexibility. - Decision making autonomy.
Disadvantages	- Weakness in dynamic environment. - Important network requirement. - System vulnerability in front of central control agent. - Additional computational requirements. - Unsuitable for large scale systems.	- Suboptimal solution. - Complex on-board processing. - Important amount of information to exchange.

Authors in [Scherer et al., 2015] suggest a system architecture with the following modules :

- Coordination/Planning module: It exchanges information and provides high level coordination.
- Plan Execution module: It controls the behavior of other modules by sending control commands.

- UAV Control module: It receives and forwards control commands.
- WIFI Control module: It receives command to change the behavior of the underlying network module and delivers information about the current connection quality.
- Streaming Control module: It starts or stops the streaming and can autonomously change and adjust the quality of the streamed video.
- Image/Video Streaming module: It encodes or decodes the images and video.
- Image Data Analysis module: It analyses the images.
- Image/Video Acquisition module: It provides images and video to the other modules.

Authors in this work propose to use centralized as well as decentralized decision making.

Taking into account the advantage of solution optimality from the centralized approach, and the advantage of mission reliability and adaptability from the decentralized approach; a *leader*-based approach has been proposed in our work [Mahdoui et al., 2017]. The fleet is subdivided into subgroups – called clusters – such that one robot in each subgroup is responsible for target assignment.

3 Applications

In the recent years, drones have become more and more available for general public thanks to the huge amount of advantages that they introduce, such that a small size (to reach inaccessible places), an ability to a Vertical Taking off And Landing (VTAL), on-board sensors to sense the environment, a low cost, etc. Furthermore, several new large scale applications require such SoS features to face and accommodate to extreme situations caused by human or nature. Indeed, UAVs are exploited in the military missions as well as in the civilian missions to encompass research and innovation activities such as:

- Civil and commercial: Disaster surveillance, inundation monitoring, traffic monitoring, search and rescue, infrastructure inspection, data collection, video images sharing, packet delivery, security, and so on.
- Military: Reconnaissance, surveillance, night operations, damage assessment, disaster evaluation, border security operations, drones with attack capabilities, target simulations of enemy aircraft or missile, and so on.

3.1 Search and rescue applications

Search and rescue are among the most serious social applications for robotic community. It consists in helping humans in searching victims by deploying UAVs after a disaster occurs

[Erdelj et al., 2017b, Erdelj et al., 2017a] in urban or rural zones such as earthquake (See Figure 1.1), landslide, fire, and so on.

Figure 1.1 – Search and rescue[1].

In the early 1990s, researchers began to tackle search and rescue missions while deploying MRS [Jennings et al., 1997, Kitano et al., 1999]. Rescue robotics is now widespread and several platforms for those missions are developed for a faster deployment, a more efficient on-board sensors, a more important line of sight for communication [Waharte and Trigoni, 2010, Tomic et al., 2012, Scherer et al., 2015], and so on.

The rescue task is a very difficult mission since robots need to be sufficiently mobile and small to avoid obstacles and access complex environments.

3.2 Reconnaissance and surveillance

Reconnaissance and surveillance were mostly used in the military domain. But, recently, they are adapted especially for civilian applications (See Figure 1.2). Those applications consist on identifying and tracking a target during critical situations such as hostage taking, intruder detection, and so on.

Figure 1.2 – Reconnaissance and surveillance[2].

In [Hougen et al., 2000], the MRS, that is used for reconnaissance and surveillance, is divided into two groups of scout: Miniature robots with limited capability for autonomous actions, and rangers consisting on a larger vehicles controlled by humans. Authors in [Hegazy et al., 2005] propose to optimally locate robots, and identify and track potential

[1]Source: `http://www.onyxstar.net/search-and-rescue-by-drone/`
[2]Source: `http://www.directindustry.fr/prod/riegl-lms/product-15822-1937769.html`

ground targets using a particle filtering framework in an urban environment. In [Oh and Green, 2004], a robot prototype for reconnaissance, surveillance and target acquisition tasks is proposed to ensure small, safe, and slow flying.

Reconnaissance and surveillance are tasks that need precision. Consequently, they need features, for both visual reconnaissance (for the target), and localization and mapping (for the environment).

3.3 Infrastructure inspection

As far as inspection is concerned, it focuses on the identification and tracking of an environment instead of a human target as done by reconnaissance and surveillance [Erdelj et al., 2017c] (See Figure 1.3).

Figure 1.3 – Infrastructure inspection[3].

Early inspection applications began in 1990s in [Stone and Edmonds, 1992] in order to localize and characterize incidents including toxic ones. Researches on inspection increased [Kuo et al., 2014, Máthé and Buşoniu, 2015, Eudes et al., 2018] and frameworks for infrastructure inspection were proposed in indoor as well as outdoor environment such as warehouse, railway, power line inspections, etc. Firms are interested in the infrastructure inspection so projects are created such as AIRMES[4] in France that includes SNCF, EDF, Aerosurveillance, and Heudiasyc partners. It aims at deploying heterogeneous UAVs cooperating within a fleet for infrastructures surveillance.

3.4 Exploration

Exploration is one of the most popular application in the robotic community [Marie et al., 2014, Matignon and Simonin, 2018]. It consists in using robots in order to explore and map a hostile or difficult environment (See Figure 1.4). The challenge in this mission is to perform a fast and robust exploration while taking into account some delicate challenges

[3]Source: `https://uavamerica.com/infrastructure-inspection-a-customer-perspective/`

[4]AIRMES - Heterogeneous UAVs cooperating within a fleet, funded by FUI. Single Inter-ministerial Funding, s linked to the "Investments for the future" Program supported by Heudiasyc: Labex MS2T and equipment excellence "Equipex Robotex".

such as limited payload capacity and computational power. However the absence of reliable maps and the employment of low speed robots may hinder the process.

Figure 1.4 – Exploration[5].

Early exploration mission were conducted by robots of the National Aeronautics and Space Administration (NASA) – called Mars rovers – such as Curiosity, Spirit and Opportunity[6]. Since the first mission launched in 1997 using a small rover called Sojourner in the Pathfinder mission (See Figure 1.5), exploration performances are still always improved and several missions are planned [7].

Figure 1.5 – NASA's rover called Sojourner for Pathfinder mission[8].

In addition to space, exploration missions take place on earth such as underwater or on land areas. The challenge in underwater missions is to guarantee a good signal propagation for data transmission despite the water absorption [Cui et al., 2006]. Ongoing researches about autonomous underwater vehicles and deep oceans exploration [Whitcomb et al., 2000, Fairfield et al., 2007, Kunz et al., 2008] are used to monitor pollution in marine environments, to study marine life, and to create and maintain underwater projects. More popular, land exploration missions are conducted in indoor spaces as well as outdoor spaces. Exploration in sensible areas such as mines and caves is risky. Those underground missions [Siles and Walker, 2009, Sahl et al., 2010, Maity et al., 2013] are used to collect and discover biological diversity, to detect chemicals and gazes, and to map these hostile environments.

[5]Source: `http://www.geologyin.com/2014/08/drones-for-geology.html`
[6]Source: https://spaceplace.nasa.gov/space-robots/en/
[7]Source: `https://mars.jpl.nasa.gov/mer/mission/status.html`
[8]Source: NASA/JPL-Caltech

3.5 Discussions

These applications need some additional features in the fleet of robots to complete their mission including robustness and fault tolerance, localization and mapping, autonomous navigation, etc. When multiple robots are involved, some aspects become very important, even essential, such as autonomy, reliable communication, and coordination. Among these applications, the exploration and mapping of unknown environment is a challenging and active area of research. This very useful application can be used as a tool for other missions' success.

This thesis addresses the multi-robot exploration problem for a bounded unknown environment using limited communication capabilities and a visual sensor as the main perception modality. We also consider that neither global information nor map or GPS are available.

4 Advantages of multiple UAVs deployment

UAV and particularly MAV, with their small size and high mobility, are often deployed in sensitive missions. A notable progress have been done in basic problems related to single robot deployment. Hence, recently in the robotic community, the emphasis has been on MRS deployment. The use of a fleet of UAVs instead of a single robot has several advantages identified from the literature [Burgard et al., 2005, De Hoog et al., 2009, Yan et al., 2013] such as:

- Accomplishing better overall system performances.
- Covering more areas in less time.
- Implementing simpler on-board processing.
- Reducing energy consumption for each robot.
- Tolerating failure due to redundancy and robustness from data fusion.
- Sensing the environment from multiple point of view (better spacial distribution).
- Ensuring flexibility in complex missions.
- Collaborating to achieve the mission's purpose.

Following the growing interest for MRS, many challenges arise such as coordination especially when using limited sensor range, processing capabilities and energy. Hence, the aforementioned MRS deployment benefits are taken into account only if the coordination condition is verified. The coordination includes inter-robots communication, efficient exploration strategy, cooperative decision making, and so on. Previous multi-robot exploration researches focused on motion planning and collision avoidance [Latombe, 1991, Fujimura and Singh, 1996, Bennewitz and Burgard, 2000]. More recently, the

emphasis in multi-robot exploration has been on coordination and cooperation [Simonin et al., 2014, Benavides et al., 2016, Schmuck, 2017].

5 Multi-robot system overview

Using potentially heterogeneous UAVs, the main objective of cooperative exploration is to achieve a full coverage of an unknown environment in minimum time. Multi-robot systems are mainly composed of three complementary components – perception [Yang et al., 2017], planning and control [Beard and McLain, 2003], and communication [Min et al., 2018] – that interact together to get a consistent and robust system. One of the main challenges of the perception component is the SLAM where no GPS is used. For the planning and control component, cooperative exploration represents one of the main problems. Thus, in the literature, cooperative exploration strategies have been proposed. Usually, these strategies are based on a utility function to assign a robot to a target. The target assignment decision is done using specific information exchanged among robots. Therefore, communication is a fundamental component for the multi-robot system. Moreover, communication issues must be taken into account. In fact, MRS have to cope with communication failures in order to ensure the mission continuity.

Therefore, in this work, we address each of the three components' challenges towards the definition of a system, which is able to provide robots with precise localization, to improve robots area coverage and to coordinate the fleet.

5.1 Architecture block diagram

The proposed framework in Figure 1.6 is an overview of the software architecture used for Multi-UAV system. It presents the different modules and data flows among them. This block diagram is distributed and embedded over all fleet members. The fleet is composed n UAVs where each one is equipped with a visual sensor.

To maintain an accurate estimate of the UAV's pose in the environment, a simultaneous localization (block **1**) and mapping (block **3**) is performed. Block **3** in the mapping layer is responsible for constructing a detailed grid map of the explored regions and keeping track of them. In the data processing block (block **4**), some specific information are picked out and exchanged using the communication layer where the network (block **7**) is in charge of maintaining data flow among UAVs. The collected data are then locally processed in the same block **4** to get exploitable information for exploration. Thereafter, block **5** performs targets selection. Planning the path and reaching it are the roles of block **6**.

Block **2** is used to visually detect other UAVs in the environment, then estimate their relative transformation using visual fiducial markers or tags. These identifiers are mounted on-board UAVs (See Figure 1.7). Different types of tags exist such as WhyCon [Nitsche et al., 2015, Krajník et al., 2014], ARTags [Higashino et al., 2016], AprilTag [Olson, 2011] or WhyCode [Lightbody et al., 2017]. By visually detecting tags, the framework precisely

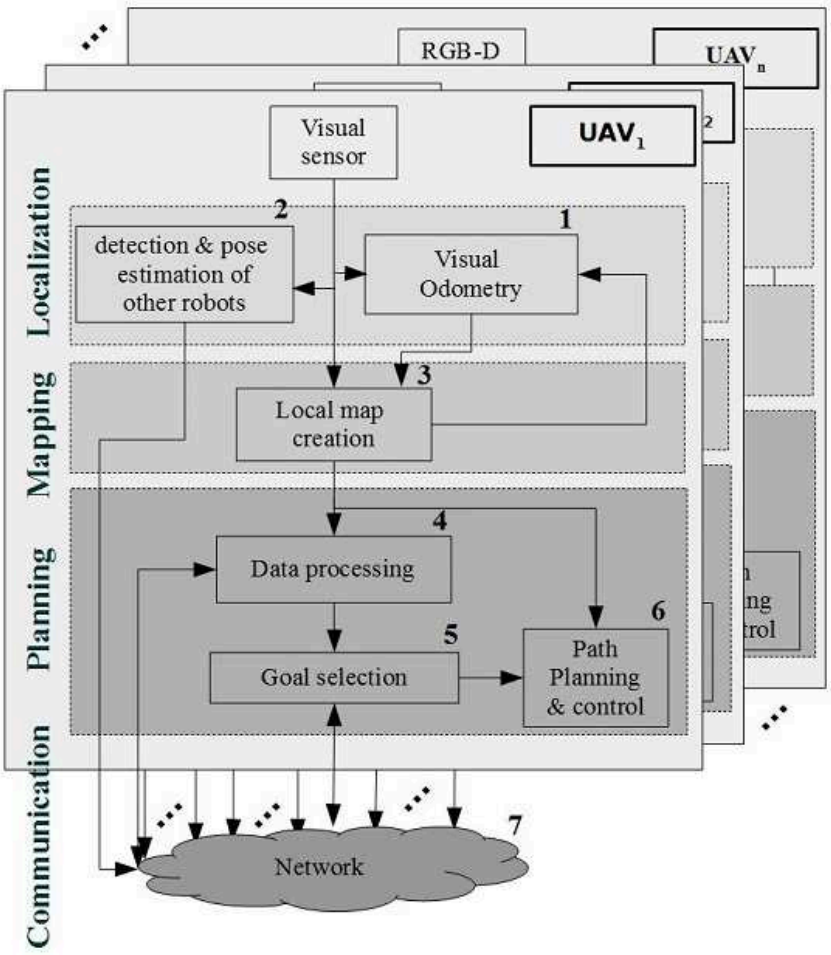
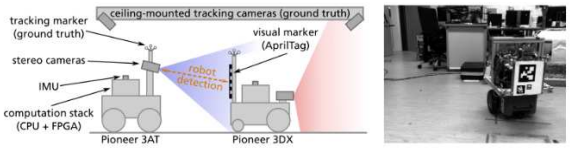


Figure 1.6 – Architecture block diagram.

computes the 3D position, the orientation, and the identity of the tags w.r.t. the camera frame of the robot.

(a) MRS used in [Olson, 2011].



(b) MRS used in [Schuster et al., 2015]

Figure 1.7 – Apriltags: Visual Fiducial System mounted on robots.

The software architecture, on-board each UAV, is composed of four layers that interact with each other:

- *Localization:* It estimates robot poses using measurements gathered from visual sensor (See Chapter 2 for more details). This layer contains two blocks: Visual

odometry (block **1**) which is used to estimate the UAV_i's own pose, and the detection and pose estimation of other robots block (block **2**) which is used to estimate the neighbors' relative transformation w.r.t. the considered UAV_i.

- *Mapping:* It creates, in different metric representation, a map of the environment using visual measurements and estimated robot transformation (See Chapter 2 for more details). This layer contains the local map creation process (block **3**) that, using measurement information along with robot estimated pose from block *1*, creates a sparse 3D point cloud representation and a 3D grid map of the environment.

- *Planning:* It collects information about other robots in the fleet and about the environment, selects a target, plans a path and attempts to reach it (See Chapter 3 for more details). This layer ensures the cooperation between robots. It contains the data processing block (block **4**) where specific information are picked out to be exchanged. These local and received information are fused and processed within this block. The goal selection process (block **5**) uses the processed information from block **4**, and the exchanged robot pose from block **7** to select – to itself or assign to others – a target to reach. Given a final and an initial pose, and also the mapped environment, the robot plans a path and attempts to reach the selected target using the path planning and control block (block **6**).

- *Communication:* It ensures a data exchange flow between robots in order to share data used for cooperation (See Chapter 4 for more details). The exchanged information contain relative transformation of detected robot w.r.t. the global reference frame from block *2*, specific information from block *4*, and assigned targets from block *5*. These information are exchanged at different steps of the exploration process.

5.2 System coordinate frames

In this thesis, we assume that the UAV fleet explores a 3D bounded unknown environment with a global reference frame ${}^0W$ (See Figure 1.8). Each robot (UAV_i, with $i \in [1..n], n \in \mathbb{N}^*$), maintains a relative motion matrix ${}^{F_i}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}_{UAV_i}$ w.r.t. its corresponding local reference frame ${}^W F_i$, and a global transformation ${}^W\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}_{UAV_i}$ w.r.t. the global reference frame ${}^0W$.

During the mission, the information computed within local frames ${}^W F_i$ of all the UAVs are processed in parallel. Before that, however, those information need to be converted in the global reference frame ${}^0W$ by knowing the UAV's local reference frame transformation w.r.t. ${}^0W$ (${}^W\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}_{F_i}$). Thereby, UAV's initial pose in ${}^0W$ needs to be known. To do that, the UAV, with the lowest *id* number in the fleet (UAV_1 in Figure 1.8), is considered as a marker. The global frame is defined such that it coincides with the marker's local frame such that Eq.1.1 is verified.

$$ {}^W\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}_{F_1} = \begin{bmatrix}\mathbb{I}_3 & 0_{3\times 1}\end{bmatrix}, \tag{1.1} $$

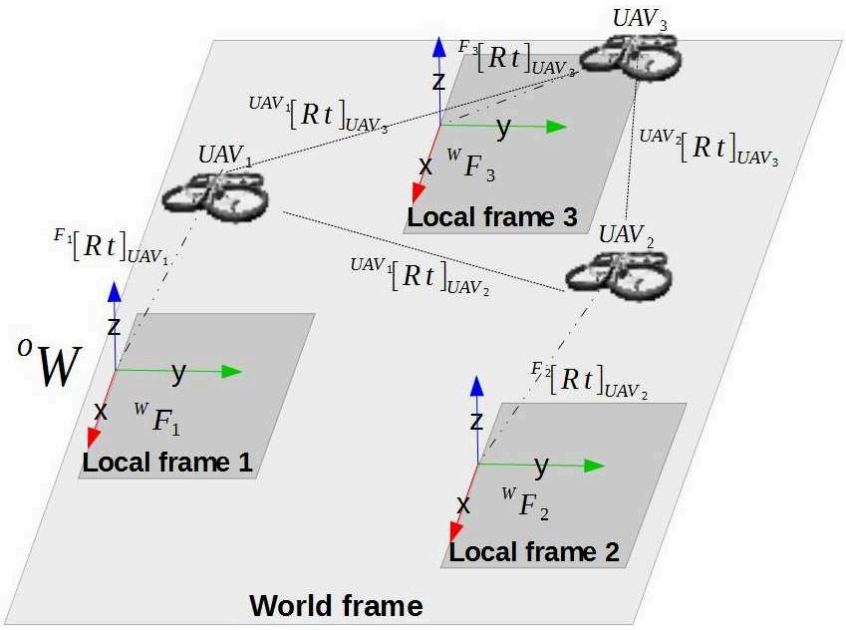


Figure 1.8 – Multi-robot coordinate system.

Using block **2** in Figure 1.6, the relative transformations ${}^{UAV_1}\left[\mathbf{R}\quad\mathbf{t}\right]_{UAV_i}$ with $i \in [1..n]$ are estimated. Knowing the latter information and assuming that the transformations ${}^{F_i}\left[\mathbf{R}\quad\mathbf{t}\right]_{UAV_i}$ are provided by the SLAM algorithm, the transformation ${}^{W}\left[\mathbf{R}\quad\mathbf{t}\right]_{UAV_i}$ is computed by applying Eq.1.2.

$$\begin{aligned} {}^{W}\left[\mathbf{R}\quad\mathbf{t}\right]_{UAV_i} &= {}^{W}\left[\mathbf{R}\quad\mathbf{t}\right]_{F_1} . {}^{F_1}\left[\mathbf{R}\quad\mathbf{t}\right]_{UAV_1} . {}^{UAV_1}\left[\mathbf{R}\quad\mathbf{t}\right]_{UAV_i}, \\ &= {}^{F_1}\left[\mathbf{R}\quad\mathbf{t}\right]_{UAV_1} . {}^{UAV_1}\left[\mathbf{R}\quad\mathbf{t}\right]_{UAV_i}, \end{aligned} \tag{1.2}$$

The transformation ${}^{W}\left[\mathbf{R}\quad\mathbf{t}\right]_{F_i}$ (equivalent to ${}^{F_1}\left[\mathbf{R}\quad\mathbf{t}\right]_{F_i}$) that describes UAV$_i$'s initial pose w.r.t. the global reference frame ${}^{0}W \equiv F_1$, is deduced.

5.3 Roles and team hierarchy

Cooperation in Multi-UAV systems often goes through the exchange of data [Yan et al., 2013]. In a limited communication ability, the data sharing link cannot always be correctly established due to the limited communication range, the data loss, the obstacles, the traffic congestion, and so on. In the proposed work, each group of robots which may communicate with one another, form a cluster $\mathcal{C}$. The fleet is composed of, at least, one cluster (if $n = n_c$). Figure 1.9 shows a fleet composed of $n = 8$ UAVs, and three clusters composed of $n_c = 1$, $n_c = 2$, and $n_c = 5$ UAVs, respectively.

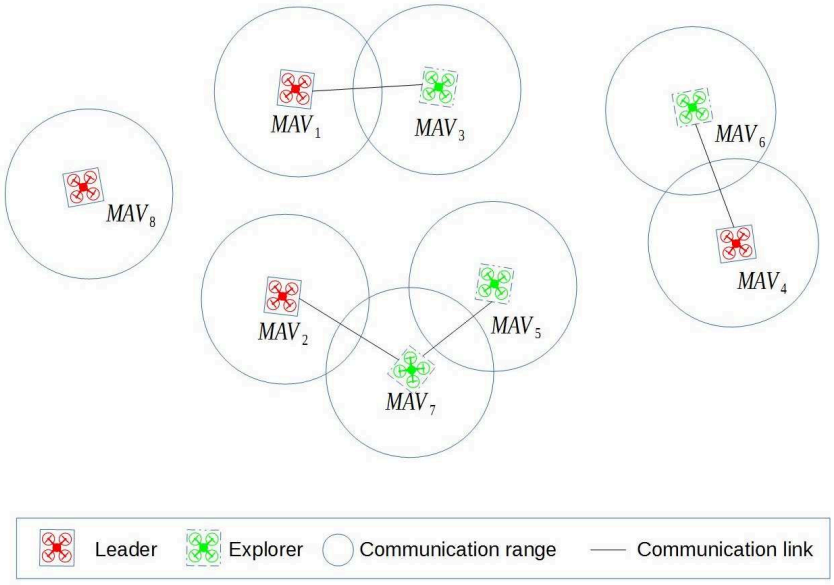


Figure 1.9 – Fleet of UAVs containing four clusters.

In each cluster $\mathcal{C}$, one robot takes the role of *leader*. It is in charge of making cooperative decision for all the other robots in $\mathcal{C}$ that have the role of *explorers*, based on some specific shared information. The decision making process relies completely on the *leader*, which can lead to mission interruption; especially when the *leader*-to-*explorer* communication link is lost, or the *leader* is out of order. To overcome these problems, the roles are constantly updated by running the role selection process (See Figure 1.10), in order to select a *leader* if the current one experienced any issue.

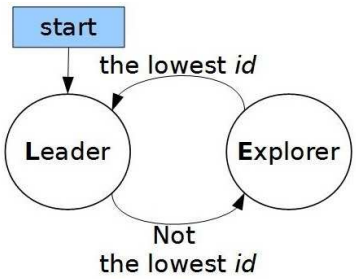


Figure 1.10 – Role selection process.

The roles are not previously defined but are adapted depending on the fleet topology changes. All UAVs' roles are initialized to *leader*. Then, as soon as UAVs start to exchange their identification number *id*, clusters are formed and then each UAV chooses its appropriate role. The *leader*'s role is taken by the UAV with the lowest *id* number in the cluster $\mathcal{C}$.

6 Conclusion

In this chapter, we presented a state of the art about the multi-robot SoS aspect. We also cited some SoS literature classifications. Furthermore, we mentioned a non exhaustive list of applications used for fleet of UAVs deployment. Still, the main challenge in MRS deployment is to make a cooperative framework. Thus, we detailed a distributed multi-robot architecture (shown in Figure 1.6) used for an unknown environment exploration.

Simultaneous Localization And Mapping

Contents

1 Introduction

This chapter concerns camera-based pose estimation study. This task is often required in several applications where, typically, the environment is unknown such as exploration, search and rescue, or surveillance. In these situations, the environment is mostly GPS-denied which inducts a challenging navigation for UAVs. Hence, we are interested in using a visual sensor as the main perception modality to map the surrounding environment and to perform localization within it.

In this chapter, a brief state of the art on pose estimation including Visual Odometry (VO) and Simultaneous Localization And Mapping (SLAM) systems is presented. We also introduce some commonly used metric map representation. After that, we propose and evaluate two approaches for SLAM systems, using both a monocular sensor and a RGB-D sensor.

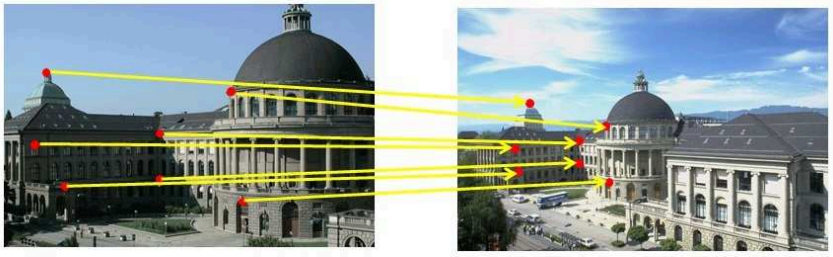
2 Pose estimation

In this chapter, we focus on the use of an embedded visual sensor for pose estimation. These sensors[1], mounted on-board moving robots, are used to gather information to map the environment and to estimate the robot's trajectory. To this end, VO and SLAM methods are dominant.

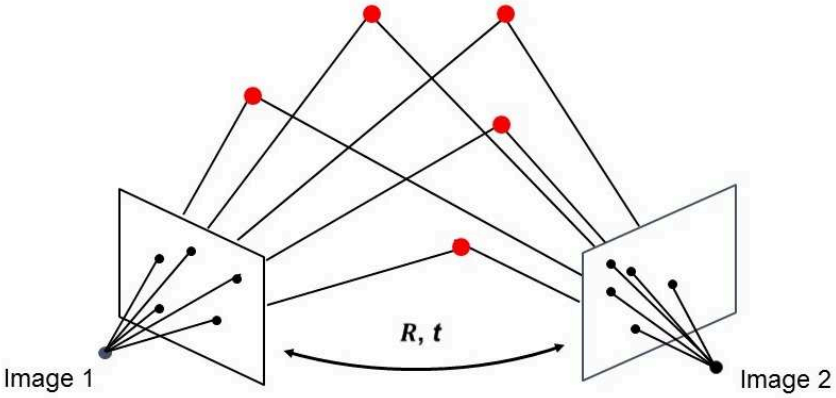
2.1 Visual Odometry

2.1.1 Overview

The visual odometry process consists in incrementally estimating the pose of a vehicle by examining the changes that motion induces on the images taken from its on-board rigidly attached camera [Scaramuzza and Fraundorfer, 2011]. VO is a 3D motion estimation (translation + rotation) computed from sequential optical sensors data such as images (See Figure 2.1).



(a) Consecutive images correspondences.



(b) Consecutive images motion estimation.

Figure 2.1 – The VO working principle. Images from [Schöps et al., 2014].

Assuming static scenes, two consecutive images at time k and $k-1$ are related by a rigid

[1]The used visual sensors are supposed to be calibrated.

body transformation $\mathbf{T}_k$ in Eq. 2.1.

$$\mathbf{T}_k = \begin{bmatrix} \mathbf{R}_{k,k-1} & \mathbf{t}_{k,k-1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{2.1}$$

Where $\mathbf{T}_k \in \mathbb{R}_{4\times 4}$ is the relative transformation, $\mathbf{R}_{k,k-1} \in \mathbb{R}_{3\times 3}$ is the rotation made on each axis from the previous pose to the next one, and $\mathbf{t}_{k,k-1} \in \mathbb{R}_{3\times 1}$ is the translation on the three axes. Figure 2.2 shows an illustration of VO problem formulation where the following sets can be introduced:

- $\mathcal{T}_{0:n} = \{\mathbf{T}_1, \mathbf{T}_1, .., \mathbf{T}_n\}$: The set of camera motions.
- $\mathcal{C}_{0:n} = \{\mathbf{C}_0, \mathbf{C}_1, .., \mathbf{C}_n\}$: The set of camera poses w.r.t. the initial coordinate frame.

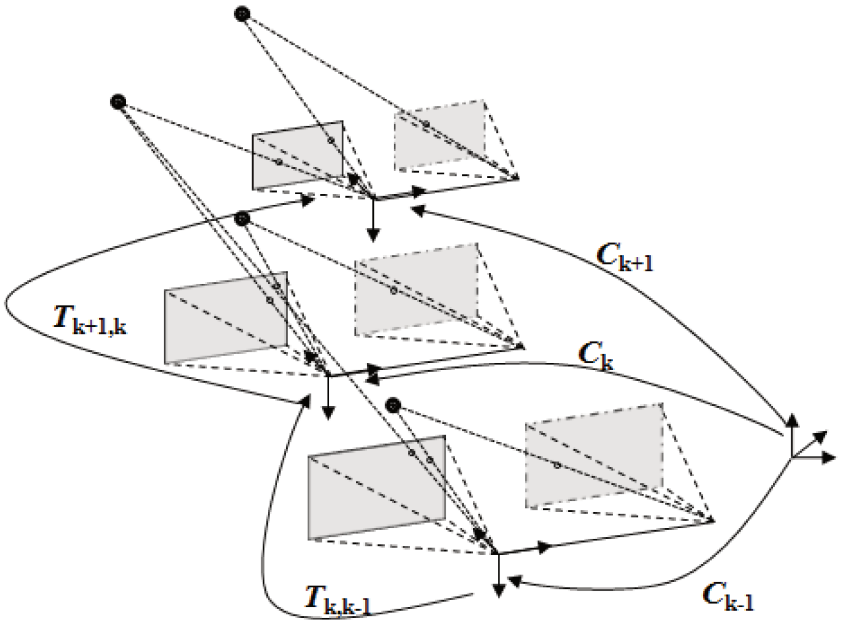


Figure 2.2 – Illustration of the visual odometry problem. Case of a stereo camera (At time k, two boxes representing two image frames are available). Image from [Scaramuzza and Fraundorfer, 2011].

The VO consists in estimating the transformations $\mathcal{T}_{0:n}$. Then, to recover the camera trajectory $\mathcal{C}_{0:n}$, the computed $\mathcal{T}_{0:n}$ is concatenated using Eq. 2.2 where, the initial camera pose $\mathbf{C}_0$ is arbitrary set.

$$\mathbf{C}_n = \mathbf{C}_{n-1}\mathbf{T}_n \tag{2.2}$$

VO estimation requires assumptions such as:

- The environment has to be sufficiently illuminated and structured.
- The scene has to be mostly static.
- The consecutive frames need sufficient overlap among them.

2.1.2 Related work

VO can be computed by two approaches classified into: Sparse visual odometry [Engel et al., 2016] (where only a part of the image data is used) and dense visual odometry [Kerl et al., 2013b] (where all the available image data are used). According to [Schöps et al., 2014], the existing VO methods (also SLAM methods) can be classified into Feature-based method and Direct method (See Figure 2.3).

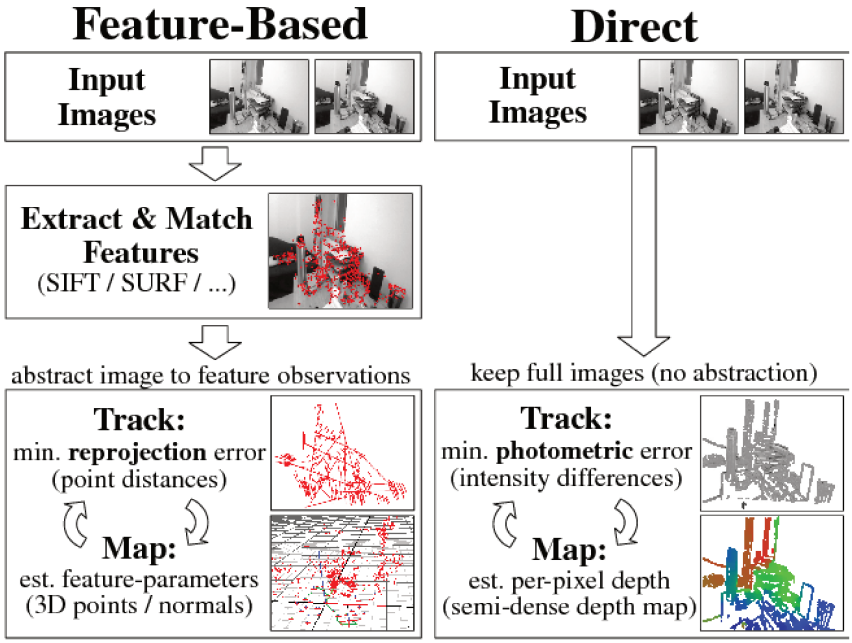


Figure 2.3 – Feature-based versus direct methods. Image from [Schöps et al., 2014].

The feature-based methods abstract images to features and discard all the other information. The main pipeline of these methods are:

- To acquire the image sequences.
- To detect/extract the features using approaches such as Scale-Invariant Feature Transform (SIFT) or Speed Up Robust Features (SURF).
- To match/track the features using techniques such as RANdom SAmple Consensus (RANSAC).
- To estimate the relative motion between the features (2D/2D, 3D/3D, 3D/2D).

- To optimize the estimated transformation.

These steps may slightly differ depending on the adopted method. For example, the bundle adjustment-based approach is a commonly used feature-based method where the main steps [Sibley et al., 2010, Shade and Newman, 2011] are:

- Image processing: To remove the distortion in lens and to filter the images to allow faster matching of features.
- Image alignment: To make an initial estimate of a 3D rotation using a gradient descent method-based on image intensity.
- Feature match in time: To project the 3D features into the left and the right images and match them using a sum-absolute-difference error metric.
- Initialize new features: To track about 100 to 150 features and to ensure a spatial distribution using a quad-tree.

Results, presented in [Shade and Newman, 2011], have shown that these estimations are robust even under difficult conditions. They are mostly adapted for large scale environments.

The direct methods [Schöps et al., 2014] perform tracking directly on the image intensity (instead of extracting and matching features). This allows to achieve a higher accuracy and robustness, especially in indoor environments where only few features are available. But, this method requires a powerful Graphic Processing Unit (GPU) to run in real time.

According to [Fang and Scherer, 2014, Fang and Zhang, 2015], the VO for RGB-D sensor can be classified within three categories:

- The image-based category [Huang et al., 2011, Kerl et al., 2013b, Endres et al., 2014, Li et al., 2015] that uses RGB-D and depth data. It is mostly adapted when there is a good gray image value or visual features.
- The depth-based category [Wang et al., 2017] that uses point cloud and is commonly used in featureless or dark environment.
- The hybrid category [Zhang et al., 2014] that uses point cloud and RGB-D.

The Fovis presented in [Huang et al., 2011], is a feature-based VO method that provides consistent motion estimation but needs to work at high frequencies for a correct estimation.

The Dense VO method (DVO), introduced in [Steinbrücker et al., 2011], estimates dense VO directly from the RGB-D frame by minimizing the difference between the previous image and the back-projected current RGB-D image. This approach is optimized in [Kerl et al., 2013b] by a probabilistic derivation and the possibility of prior integration of the motion and the sensor noise. It has been extended by adding weight to each pixel and

by incorporating a motion prior. Actually, this method is based on the photo-consistency assumption that assumes that if a point is observed by two cameras, it has the same brightness in both images.

Authors in [Zhang et al., 2014] propose the Depth Enhanced Monocular Odometry method (DEMO) to enhance VO from monocular images by the assistance of depth information even if it is sparsely or locally unavailable. According to [Fang and Scherer, 2014, Zhang et al., 2014, Fang and Zhang, 2015], DVO is adapted for environment with relatively dark illumination and DEMO [Zhang et al., 2014] for areas with no sufficient depth information.

A Fast Semi-Direct Monocular Visual Odometry called SVO is proposed in [Forster et al., 2014]. The algorithm operates directly on pixel intensities. The 3D points are estimated using probabilistic mapping method that allows to reduce the outliers (false matching points) and get more reliable points. Results show that the proposed method is robust, and faster than current state-of-the-art methods.

In some works, authors use Structure From Motion (SFM) term as a synonym of VO. Actually, VO is a particular case of SFM [Scaramuzza and Fraundorfer, 2011]. In fact, SFM is a more general process that treats both 3D problem of camera pose estimation and structure from images set that can even be unordered. They are generally refined with an off-line optimization known as bundle adjustment. The SFM's computation time grows when the image number grows too. Compared to the SFM, VO focuses on 3D sequential pose estimation in real time. In VO, the trajectory estimation optimization is optional.

2.2 Simultaneous Localization And Mapping (SLAM)

2.2.1 Overview

The SLAM problem is one of the most important topics in the robotic community. It consists in answering simultaneously two important questions: Where is the robot? And what does the world looks like? Let's consider a robot with a visual sensor mounted on it. It is moving in an a priori unknown environment and is collecting information about relative observations of landmarks. Figure 2.4 shows the evolution of the robot poses and landmarks during a short time of navigation [Durrant-Whyte and Bailey, 2006]. The following sets are then introduced:

- $\mathcal{P}_{0:k} = \{\mathbf{p}_0, \mathbf{p}_1, .., \mathbf{p}_k\}$ represents the set of poses of robot. Each pose includes the position and the orientation of the robot.
- $\mathcal{U}_{0:k} = \{\mathbf{u}_1, \mathbf{u}_2, .., \mathbf{u}_k\}$ represents the set of control vectors used to drive the robot from state $l-1$ to state l at time l with $l \in [1..k]$.
- $\mathcal{M}_{0:k} = \{\mathbf{m}_1, \mathbf{m}_2, .., \mathbf{m}_k\}$ represents the set of vectors that defines the states of landmarks. These locations are considered as time invariant.
- $\mathcal{Z}_{0:k,i}$=$\{\mathbf{z}_{1,i}, \mathbf{z}_{2,i}, .., \mathbf{z}_{k,i}\}$ represents the set of observations of the landmark i made at times $[0..k]$.

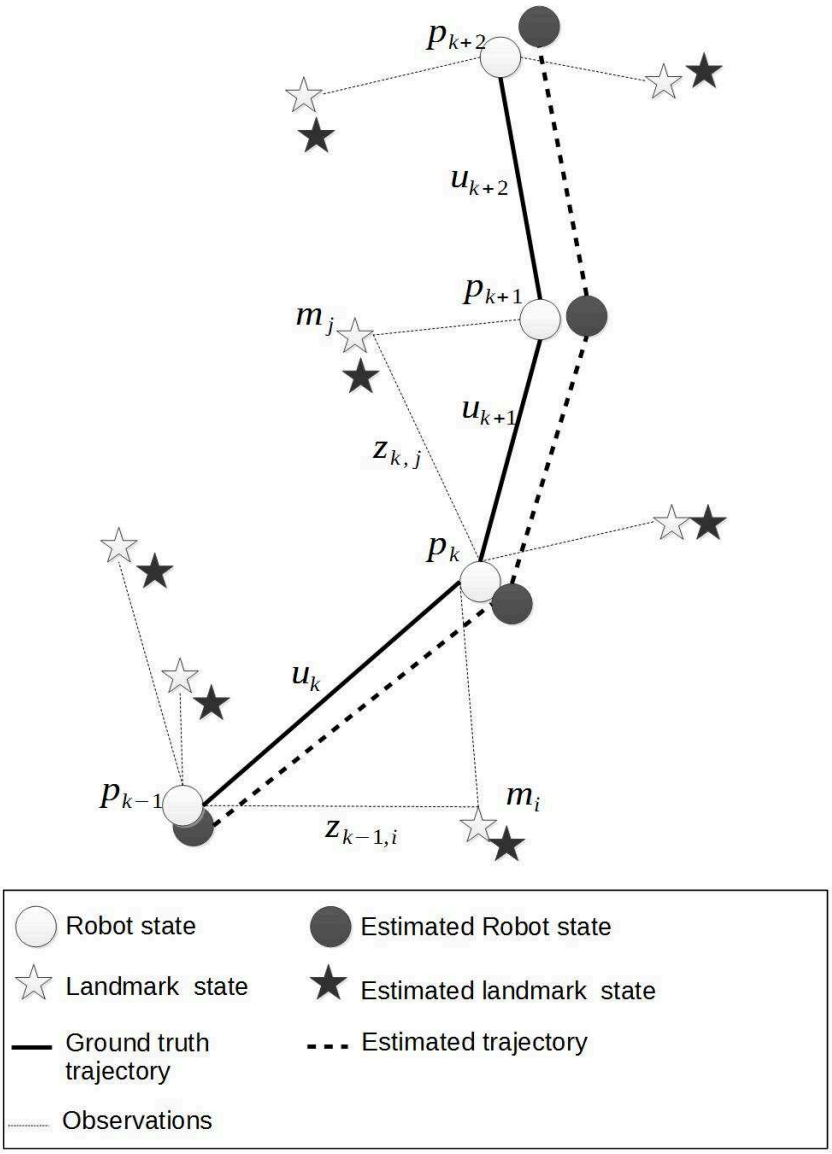


Figure 2.4 – SLAM problem formulation.

The formulation of a probabilistic SLAM can be written in a probabilistic form as expressed in Eq.2.3. This probability distribution has to be computed at each time k, knowing the observed landmarks, the control vectors, and the initial robot pose.

$$P(\mathbf{p}_k, \mathbf{m} \mid \mathcal{Z}_{0:k}, \mathcal{U}_{0:k}, \mathbf{p}_0) \tag{2.3}$$

Generally, the SLAM problem is resolved through a recursive solution that requires a motion model and an observation model. By far, the most commonly used approaches for these model representations are:

- The Extended Kalman Filter (EKF) for an EKF-SLAM solution.
- The Rao-Blackwellised particle filter for a Fast-SLAM solution.

Some other solutions, explained in the next section, are proposed to attempt to solve the SLAM problem [Cadena et al., 2016].

2.2.2 Related work

The VO's main objective is to ensure a local consistency while SLAM aims at a global consistency of the map [Renaudeau et al., 2018] and the trajectory [Mur-Artal and Tardós, 2017a]. Indeed, SLAM is used to obtain a global and consistent estimate of the robot path based on loop closure. It allows the algorithm to apply a global optimization to reduce drift on both the trajectory and the map. Whereas, VO aims at estimating the trajectory pose after pose, and applying optimization after a certain number of poses called windowed optimization. The choice between using VO and Visual SLAM is based on a trade-off between performance and consistency, and simplicity of implementation.

The Parallel Tracking and Mapping (PTAM) proposed in [Klein and Murray, 2007] is a monocular SLAM based on a parallel framework that includes a tracker and a mapper, in order to increase the responsiveness and robustness of the whole system. The tracker enables fast camera localization in real time; whereas keyframe based mapper builds the global map. Authors in [Ta et al., 2013] modified PTAM – originally designed for augmented reality – making it more suitable for robot navigation. Instead of using a motion model, odometry and visual measurements are fused into the framework to deal with the lack of visual features and the lack of motion in the environment. In addition, a loop closer mechanism is performed.

Authors in [Cunningham et al., 2010] use an extending Smoothing And Mapping (SAM) approach consisting on a graphical model approach that introduces the Constrained Factor Graph (CFG). A Decentralized Data Fusion-SAM (DDF-SAM), that satisfies the DDF requirements while taking into account the benefits of naive approach, is proposed. The framework is composed of three modules:

- The Local Optimizer Module performs the SLAM for one robot in its local environment and produces its local map and condensed local graph.
- The Communication Module shares the previous condensed local graph so that each robot maintains its local graph and a cache of neighboring robots' condensed graphs.
- The Neighborhood Optimizer Module merges the condensed graphs to obtain a neighborhood graph that can be used to build the map.

By applying loop closing process along with DVO method, authors in [Kerl et al., 2013a] propose a SLAM method that applies a global optimization to reduce drift on both the trajectory and the map. Yet, many implementation issues of this SLAM method rise due to versions incompatibility.

Authors in [Forster et al., 2013a] propose a distributed monocular SLAM for a multi-robot system. To determine each robot's individual motion, measurements from an on-board camera and Inertial Measurement Unit (IMU) are combined together. Specific data

such as image coordinates, descriptors as features of selected keyframes, and relative pose estimation are streamed to a ground station – called Collaborative Structure From Motion (CSFM) – where a map for each robot is created and merged if there is an overlap among them.

A software architecture is proposed in [Brand et al., 2014] to perform a distributed SLAM. The on-board stereo-vision based mapping system proves its effectiveness in indoor, unstructured outdoor as well as mixed environment.

A novel direct and feature-less Large-Scale Direct monocular SLAM (LSD-SLAM) method is proposed in [Engel et al., 2014]. It performs an accurate pose estimation using direct image alignment along with filtering-based estimation of semi-dense depth maps. A 3D reconstruction of the environment is represented as a pose graph where keyframes are vertices.

The OKVIS SLAM [Leutenegger et al., 2015] proposes a non-linear optimization approach that tightly fuses visual measurements along with readings from an IMU. This allows to significant advantages in quality of performance and computational complexity.

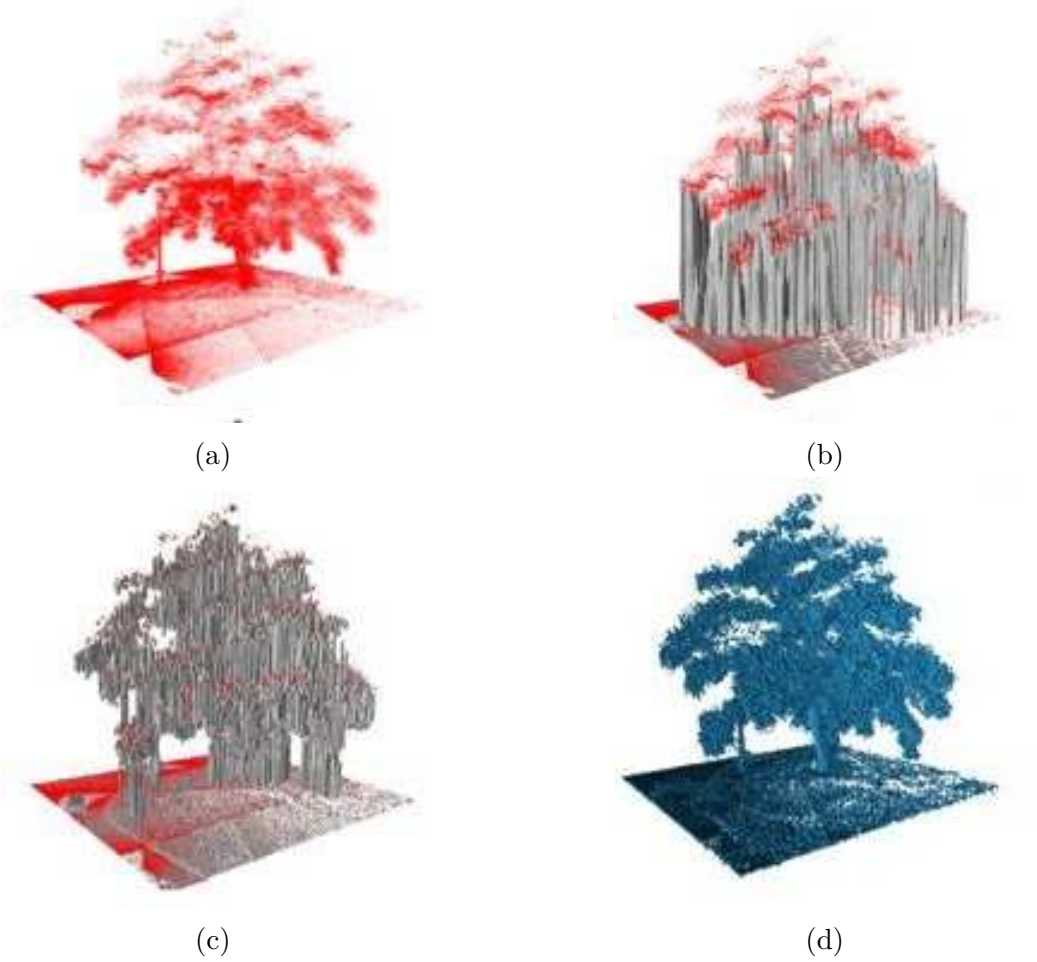
A novel tightly coupled visual-inertial SLAM system is proposed in [Mur-Artal and Tardós, 2017b]. This system is able to be applied to monocular as well as stereo and RGB-D sensors. It performs loop closing to attempt a zero-drift localization in already mapped areas.

Authors in [Mur-Artal and Tardós, 2017a] propose a lightweight RGB-D feature-based SLAM method called ORB-SLAM2. It is adapted for monocular (depth triangulated from different view), stereo and RGB-D sensors. Using the TUM RGB-D data-set [Sturm et al., 2012], in most cases, ORB-SLAM2 performs better than Elastic-Fusion [Whelan et al., 2016], kintinuous [Whelan et al., 2015], DVO SLAM [Kerl et al., 2013a] and RGB-D SLAM [Endres et al., 2014] in terms of Root Mean Square Error (RMSE) translation error.

A new open framework for research in Visual Inertial (VI) mapping and localization – called Maplab – is proposed in [Schneider et al., 2018]. It contains a RObust Visual Inertial Odometry (ROVIO) with Localization Integration (ROVIOLI) and an off-line Maplab-console. ROVIOLI, composed of an on-line Visual-Inertial Odometry (VIO) and a localization front-end [Bloesch et al., 2017], is used for pose estimation and visual-inertial map building. The Maplab-console is used to apply algorithms on map in an off-line batch fashion such as map alignment and merging, VI optimization, loop closure detection, etc. Using EuRoC data-sets for comparison, ROVIOLI outperforms ORB-SLAM2 which itself outperforms in its tern ROVIO in terms of position and orientation RMSE. Nonetheless, ROVIOLI requires a global shutter camera and an IMU to work. It also does not make any use of depth information which makes it not optimal when using a RGB-D camera.

3 Metric map representation

The map reflects the environment by representing its model using either topological or metric method. A topological map is a graph data structure composed of vertices that represent the locations in the map, and edges to show the connection/link between them. Whereas, a metric map is a geometric representation of the environment. Figure 2.5 shows an example of different map-structure representations.



(a) (b)

(c) (d)

Figure 2.5 – Examples of a map represented in different structures. (a) Point cloud map. (b) Elevation map. (c) Multi-level surface map. (d) Occupancy grid map based on an octree. Image from [Hornung et al., 2013].

3.1 Point cloud representation

The point cloud is one of the simplest metric map representation of the environment (see Figure 2.5a). The points gathered from a range sensor are transformed into a global coordinate frame. But, this representation is not adapted for dynamic environment and does not cope with sensor noise.

3.2 Occupancy grid representation

The occupancy grid map is a discretization of the environment in regularly sized 2D squares – called cells – or 3D cubic volumes – called voxels – (See Figure 2.5d). The occupancy grid map is based on a hierarchical data structure – called Octree – which represents a 3D space that is recursively subdivided until attending a minimum voxel size – called resolution –(See Figure 2.6). By increasing the resolution, the map becomes less coarser.

Figure 2.6 – Examples of an occupancy grid map with resolutions of $0.08m$, $0.64m$, and $1.28m$, respectively. Image from [Hornung et al., 2013].

The occupancy grid representation introduces several advantages [Hornung et al., 2013] such as:

- Arbitrary environment representation without prior assumptions.
- Fast data access.
- Flexibility in extending and combining different maps with different resolutions.
- Updatability in adding new informations or sensor readings.
- Compact memory storage.
- Obstacle distinction for safe robot navigation.

Using the sensor measurements, the cells of the 2D (or the voxels of the 3D) occupancy grid map are labeled as unknown, free or occupied as shown in Figure 2.7.

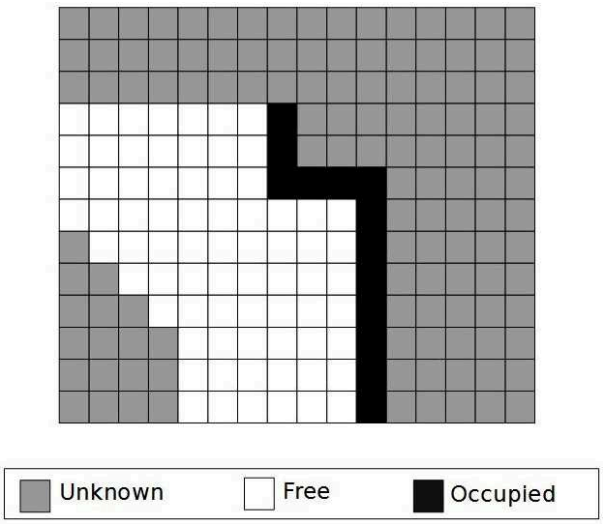


Figure 2.7 – 2D occupancy grid example.

Occupancy grid matching

We propose and implement a simple map matching process in order to evaluate and illustrate the global map during the mission. Suppose that we have two maps $\mathbf{M}_1$ and $\mathbf{M}_2$ (See Figure 2.8) with the following assumptions:

- Belonging to the same global frame.
- Having the same resolution.

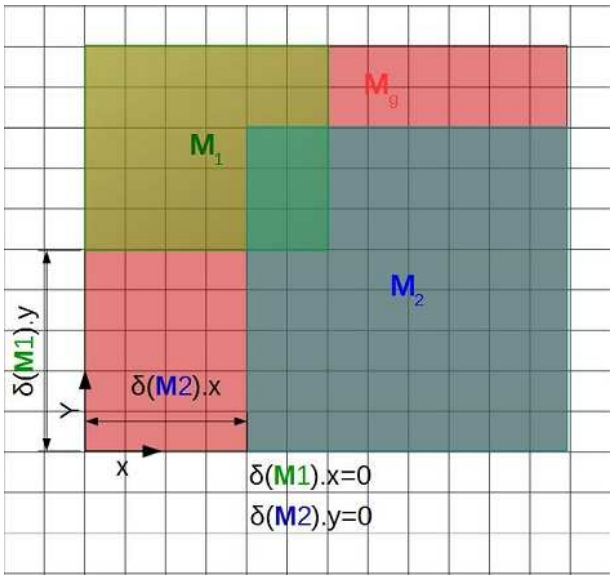


Figure 2.8 – Map matching of $\mathbf{M}_1$ and $\mathbf{M}_2$.

Each map is composed of:

- *Header*: It contains the sequence ID that is consecutively increasing, the stamp that defines the seconds and nanoseconds, and the frame this data is associated with.
- *Meta data*: It contains the time load map, which is the time at which the map hasd been loaded, the map resolution ($\mathbf{M}_i.resolution$) that defines the metric size of the cells, the map width ($\mathbf{M}_i.width$) and height ($\mathbf{M}_i.heigth$) in number of cells, and the origin of the map ($\mathbf{M}_i.origin$).
- *Data*: It contains the probability of occupancy of the cells in a row-major order.

Algorithm 2.1 describes the pipeline to match $\mathbf{M}_1$ and $\mathbf{M}_2$ where the steps can be summarized as follows:

- Steps 1 to 5: The meta data of the fused map are defined.
- Steps 6 to 7: Each cell in the fused map ($grid(\mathbf{M}_g)$) is initialized to $\mathbf{l}_u$ (cell labeled as $UNKNOWN$).
- Steps 8 to 17: Each unknown cell of the grid ($grid(\mathbf{M}_g) = \mathbf{l}_u$) is filled in using the value of either $grid(\mathbf{M}_1)$ or $grid(\mathbf{M}_2)$.
- Step 18: Return the fused map $\mathbf{M}_g$

Algorithm 2.1: Map matching.

Input: Maps $\mathbf{M}_i$ with $i \in n_c$.
Output: Fused map $\mathbf{M}_g$.

1: $\mathbf{M}_g.origin = argmin_{i \in n_c}(\mathbf{M}_i.origin)$;
2: $\delta(\mathbf{M}_i).x = \frac{(\mathbf{M}_g.origin.x - \mathbf{M}_i.origin.x)}{\mathbf{M}_i.resolution}$;
3: $\delta(\mathbf{M}_i).y = \frac{(\mathbf{M}_g.origin.y - \mathbf{M}_i.origin.y)}{\mathbf{M}_i.resolution}$;
4: $\mathbf{M}_g.width = argmax_{i \in n_c}(\mathbf{M}_i.width + \delta(\mathbf{M}_i).x)$;
5: $\mathbf{M}_g.height = argmax_{i \in n_c}(\mathbf{M}_i.height + \delta(\mathbf{M}_i).y)$;
6: **for** all grids in $\mathbf{M}_g$ **do**
7: $grid(\mathbf{M}_g) = \mathbf{l}_u$.
8: **for** $i = 0; i < n_c; i++$ **do**
9: **for** $j = 0; j < \mathbf{M}_i.width; j++$ **do**
10: **for** $k = 0; k < \mathbf{M}_i.height; k++$ **do**
11: **if** $grid(\mathbf{M}_i)[j + k * \mathbf{M}_i.width]! = \mathbf{l}_u$ **then**
12: $grid(\mathbf{M}_g)[(j - \delta(\mathbf{M}_i).x) + (k + \delta(\mathbf{M}_i).y) * \mathbf{M}_g.width] = grid[j + k * \mathbf{M}_i.width]$.
13: **end if**
14: **end for**
15: **end for**
16: **end for**
17: **end for**
18: **return** $\mathbf{M}_g$.

The proposed algorithm is a basic map matching approach used to merge two maps. It can be adapted to merge more than two maps at a time. This algorithm deals only with simple rectangular shape maps (This can be improved in our future work.).

4 Monocular SLAM

4.1 Monocular sensor

Monocular sensors are those where the only sensing device is a single camera. This sensor provides a set of images taken at discrete times $k = [0..n]$ (See Eq.2.4).

$$I_{0:n} = \{I_0, ..., I_n\} \tag{2.4}$$

Each image I_k is a set of pixels – also called color components – that are stored in a $h \times w \times 3$ matrix where the height h and width w of the image correspond to the matrix's number of rows and columns, respectively (See Figure 2.9). The element of (u, v) index in the matrix represents the pixels' RGB value.

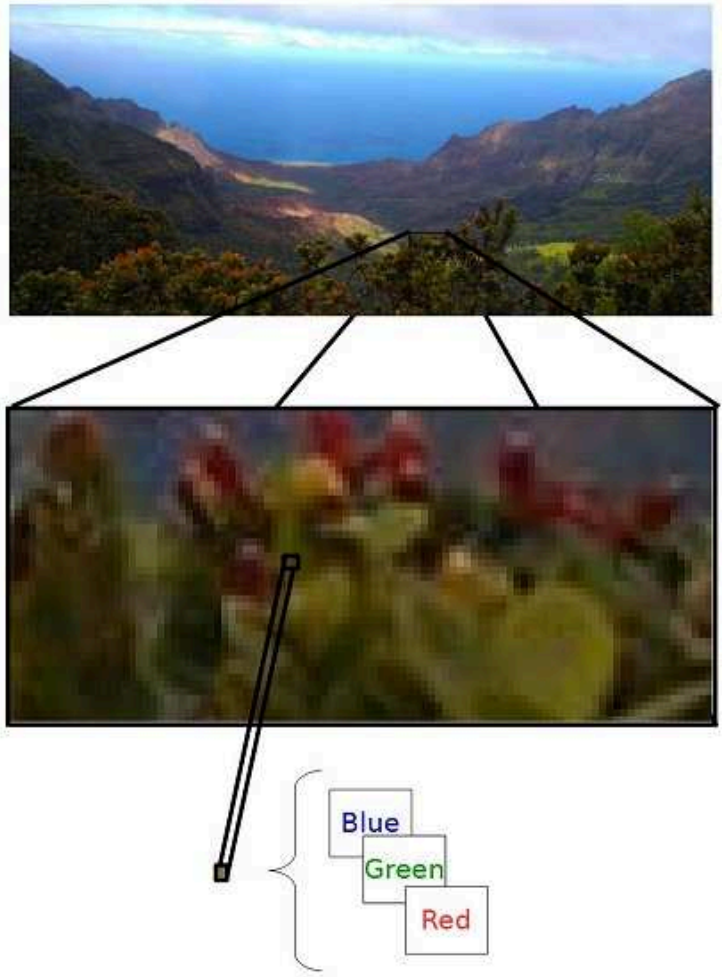


Figure 2.9 – Image[2]pixels representation.

[2]This image was taken in Kauai island, Hawaii.

4.2 Visual SLAM

Monocular sensors do not provide neither depth measurements (RGB-D camera example) nor two images of the same scene at the same time (stereo camera example) to compute these depth measurements. This leads to the inherent problem of scale ambiguity which consists in computing the scale factor. Despite the great progress in problems related to monocular SLAM, the main challenge is still the metric scale estimation. The scale[3] defines the relationship between sizes of the world and the created map. As a single camera cannot compute the scale factor, another sensor is added to be able to measure metrics from the environment. In the literature [Forster et al., 2015, Concha et al., 2016, Mur-Artal and Tardós, 2017b, Spaenlehauer et al., 2017], this additional sensor is mostly chosen to be an IMU due to its light weight, small size, and easiness to be mounted on an UAV.

An architecture for visual inertial SLAM system is proposed in Figure 2.10. It uses a monocular camera and an IMU as sensors. Similarly to the PTAM approach [Klein and Murray, 2007], two threads are used: A tracker for a fast response to changes in the environment and a mapper to build a high quality map of the environment. The only difference is that the proposed approach uses odometry measurements instead of a motion model.

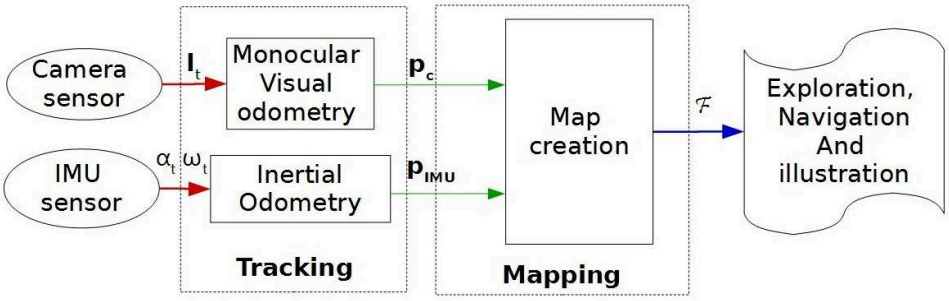


Figure 2.10 – Visual inertial SLAM outline. $\mathbf{I}_t$ represent the images input sets from the camera. α_t and ω_t are, respectively, the acceleration and angular measurements from the IMU sensor. $\mathbf{p}_c$ and $\mathbf{p}_{\text{IMU}}$ are the estimated states from the camera and the IMU sensors, respectively. $\mathcal{F}$ is the generated 2D factor graph.

The monocular visual odometry is computed by detecting and tracking features of images coming from an extrinsic camera sensor. Additional inertial odometry is computed using the acceleration (α_t) and the angular velocities (ω_t) measurements from an intrinsic IMU sensor. Both are then fused in the mapper thread where a factor graph is created. An optimized pose estimation is then generated from this graph. For the map creation, a sparse nonlinear incremental optimization approach – called iSAM2 – [Kaess et al., 2012] is used. It allows to provide updated pose estimations when new measurements are available.

The proposed architecture allows not only to overcome the scale ambiguity but also to reduce the accumulated drift from the estimated trajectory.

[3]Source: `https://www.kudan.eu/kudan-news/scale-simultaneous-localisation-mapping/`

4.2.1 Problem formulation

The proposed system is a graph-based SLAM problem that can be formulated in a factor graph. The graph is a factorization of a function $\boldsymbol{F}(\theta)$ described in Eq.2.5.

$$\boldsymbol{F}(\theta) = \prod_i \boldsymbol{F}(\theta_i), \tag{2.5}$$

with θ_i is a variable node i.

The purpose is to find the variable θ^* that maximizes the function $\boldsymbol{F}(\theta)$:

$$\theta^* = argmax_\theta \boldsymbol{F}(\theta), \tag{2.6}$$

The factor graph, presented in Figure 2.11, is composed of:

- $\mathcal{P}_{0:n}$=$\{\mathbf{p}_0, \mathbf{p}_1, .., \mathbf{p}_n\}$: The set of unknown poses.
- $\mathcal{M}_{0:m}$=$\{\mathbf{m}_0, \mathbf{m}_1, .., \mathbf{m}_m\}$: The set of landmarks.
- $\mathcal{Z}_{i=cste,j=0:m}$=$\{\mathbf{z}_{i0}, \mathbf{z}_{i1}, .., \mathbf{z}_{im}\}$: The set of visual measurements.
- $\mathcal{B}_{0:n-1}$=$\{\mathbf{b}_0, \mathbf{b}_1, .., \mathbf{b}_{n-1}\}$: The set of odometry measurements. $\mathbf{b}_i$ is the odometry between the pose $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$.

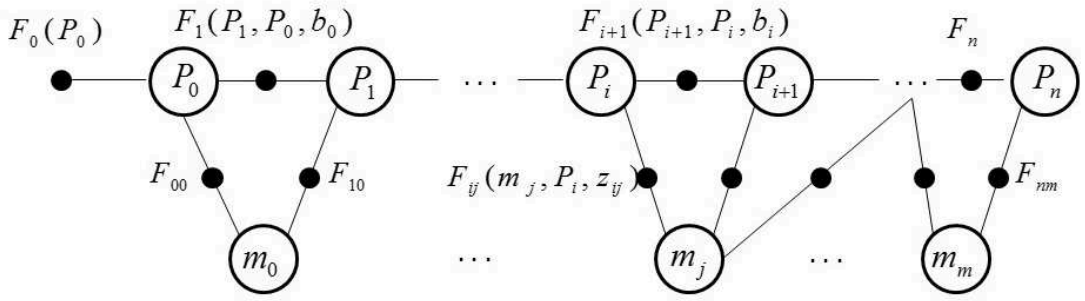


Figure 2.11 – Factor graph for monocular SLAM. $\boldsymbol{F}_0(\mathbf{p}_0)$ and $\mathbf{m}_i$ are the variable nodes. $\boldsymbol{F}(0)$ is the prior density factor. $\boldsymbol{F}_i(\mathbf{p}_{i+1}, \mathbf{p}_i, \mathbf{b}_i)$ is the odometry factor between the pose $\mathbf{p}_i$ and $\boldsymbol{F}_{ij}(\mathbf{m}_j, \mathbf{p}_i, \mathbf{z}_{ij})$ is the measurement likelihood model between the pose $\mathbf{p}_i$ and its landmark $\mathbf{m}_i$.

This bipartite factor graph is composed of:

- Variable nodes:
  - ◇ Camera pose $\mathbf{p}_i$.
  - ◇ Landmarks $\mathbf{m}_j$.
- Factor nodes:

◇ Prior densities on the variable nodes $\boldsymbol{F}_0(\mathbf{p}_0) = p(\mathbf{p}_0)$.

◇ The motion models between two camera poses $\boldsymbol{F}_i(\mathbf{p}_{i+1}, \mathbf{p}_i, \mathbf{b}_i) = p(\mathbf{p}_{i+1}|\mathbf{p}_i, \mathbf{b}_i)$ given the odometry measurement $\mathbf{b}_i$.

◇ The measurement likelihood models $\boldsymbol{F}_{ij}(\mathbf{m}_j, \mathbf{p}_i, \mathbf{z}_{ij}) = p(\mathbf{m}_j|\mathbf{p}_i, \mathbf{z}_{ij})$ between the pose $\mathbf{p}_i$ and the landmark $\mathbf{m}_j$ given the visual measurement $\mathbf{z}_{ij}$.

4.2.2 Incremental Smoothing and Mapping 2 (iSAM2)

To resolve the factor graph and estimate the poses, the iSAM2 approach [Kaess et al., 2012] is used. The estimation problem is based on three graphical models (See Figure 2.12):

- Explicit factor graph.
- Implicit chordal Bayes net.
- Bayes tree.

Therefore, the iSAM2 algorithm can incrementally obtain an estimate of unknown variables (such as UAV poses and landmarks) given a set of non linear factors (such as odometry) to finally construct the UAV's map.

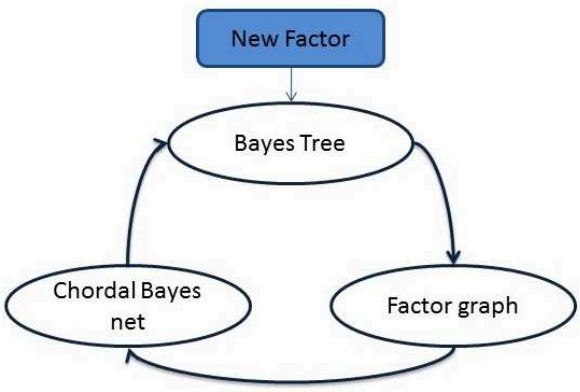


Figure 2.12 – iSAM2 graphical models.

When a new factor is taken into account, the affected part of the Bayes Tree is isolated. From it, data are formulated as a factor graph and an associated Jacobian matrix. Then, the factor graph is transformed into a chordal Bayes net and a square root information matrix using a specific variable order elimination. The last one eliminated is called the root. Finally, based on the clique structure in the chordal Bayes net, a Bayes tree is formed with the square root information matrix. Based on these models, the incremental resolution and update of the iSAM2 are resumed in five important steps presented in Figure 2.13.

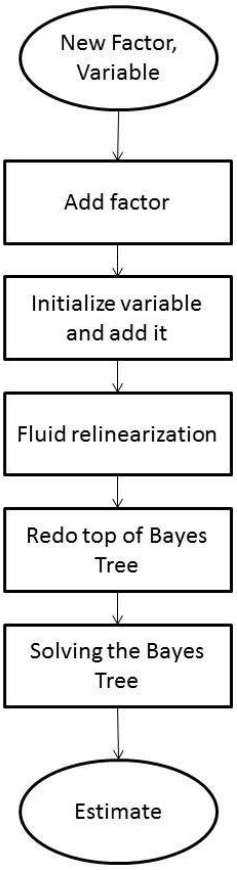


Figure 2.13 – The iSAM2 steps.

Taking into account the available new factors, variables are initialized and added to the variable nodes. Thereafter, the process performs a fluid relinearization for marked variables in order to track the validity of the linearization point for each variable when needed. These marked variables are chosen according to a threshold between its current estimate and the linearization point. Then, the Bayes tree is partially updated starting from the cliques involved by the marked variables and the variables affected by new factors, up to the top of the tree. Using the obtained new Bayes tree, an update of the current estimate is performed.

4.3 Results and discussions

The SLAM algorithm was implemented with the Georgia Tech Smoothing And Mapping toolbox[4] (GTSAM) using the factor graph implementation. The experimental code is written in MATLAB[5] and includes MEX functions[6] of the c++ library. Tests were performed on a 2.50GHz i5 Linux machine. To validate the SLAM algorithm, the measurements were simulated with raw data from the Kitti vision data-set [Geiger et al.,

[4]Source: `https://borg.cc.gatech.edu/index.html`

[5]Source: `https://fr.mathworks.com/`

[6]Source: `https://fr.mathworks.com/help/matlab/matlab_external/introducing-mex-files.html`

2013]. These data contain IMU measurements used for dead reckoning, and images used for the visual features. To compute the inter frame visual odometry, the libviso2 library[7] is used. Hence, given the set of non linear factors including IMU and visual odometry measurements, the iSAM2 estimates and optimizes poses and landmarks.

To see the improvement performed with the proposed approach, the estimated trajectory from iSAM2 is compared to those from IMU, libviso2 and GPS, in Figure 2.14.

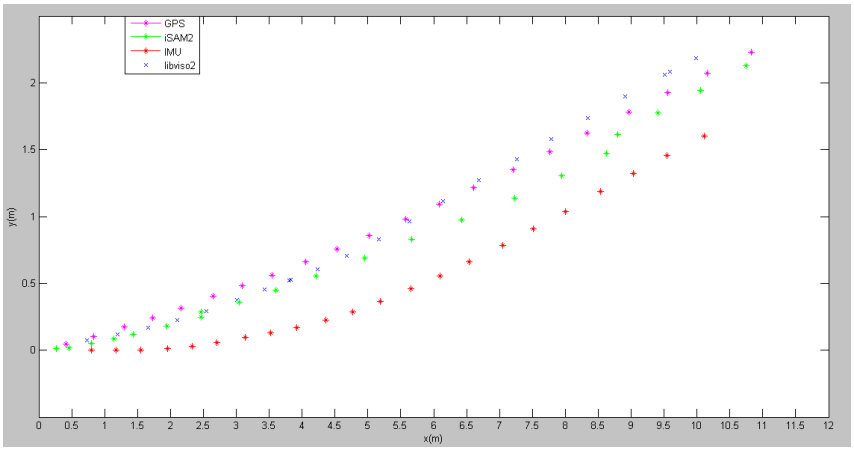


Figure 2.14 – Trajectory results (2D projection) using IMU, libviso2, iSAM2 and GPS.

The GPS trajectory is obtained using Eq. 2.7:

$$\begin{aligned} x &= R * cos(latitude) * cos(longitude), \\ y &= R * cos(latitude) * sin(longitude), \\ z &= R * sin(latitude), \end{aligned} \tag{2.7}$$

with R the radius of the earth.

Results show that combining the IMU and the libviso2 trajectory using the iSAM2 helps to improve the poses estimation, and to reduce the IMU drift compared to the ground truth. Consequently, by obtaining a graph to exchange data instead of images, the data size is considerably reduced. Indeed, theoretically, sending images require about 1.843 Mbps (assuming that an image is encoded in 24 bits); whereas sending poses and landmarks needs about 0.323 Mbps (assuming that an IEEE standard encodes reals under 32 bits). Hence, thanks to the graph-based SLAM, we manage to reduce data size by about 5 times.

5 RGB-D SLAM

The RGB-D SLAM uses an RGB-D camera as a visual sensor to sense the environment. This sensor avoids the scale estimation problem thanks to the depth information.

[7]Source: `http://www.cvlibs.net/software/libviso/`

5.1 RGB-D sensor

The RGB-D camera is composed of a digital camera for colored images $\mathbf{C}_t$, and another device for the depth image $\mathbf{D}_t$. The colored image $\mathbf{C}_t$ is a 3D matrix encoded in the RGB space where each color represents a dimension with a number of raws and columns corresponding to the image resolution (See section 4.1 on monocular sensor). The depth image $\mathbf{D}_t$ is a 2D matrix where $\mathbf{D}_t(u, v)$ is the depth of the pixel of column u and row v. The depth measurements can be computed by different technologies (e.g. camera, infrared) where the RGB-D sensor can be classified as follows:

- *Active:* It is composed of a camera and a projector. The projector emits known patterns that are compared with the camera to get the correspondences. The depth is computed from these correspondences and the known transformation between the camera and the projector.
- *Passive:* It represents stereo sensors composed of two RGB cameras. The depth is computed by knowing the transformation between them.

RGB-D sensor Comparison

Different types of RGB-D camera exist. One of the most famous is the kinect[8] camera (See Figure 2.15a). Other relatively new technologies are available such as DUO MLX[9] (See Figure 2.15b), ZED[10] stereo (See Figure 2.15c), and Intel RealSense ZR300[11] (See Figure 2.15d).

(a) Kinect camera. (b) DUO MLX camera. (c) ZED camera. (d) ZR300 camera.

Figure 2.15 – Examples of RGB-D sensors.

A brief comparison is summarized in Table 2.1. The ZED camera have the higher resolution and range compared to DUO and Kinect. However, this camera requires a powerful processor to work properly. DUO MLX is a lightweight camera with a good resolution, small dimensions but a small range and a relatively high price. The ZR300

[8]Source: `https://msdn.microsoft.com/en-us/library/hh438998.aspx`
[9]Source: `https://duo3d.com/product/duo-minilx-lv1`
[10]Source: `https://www.stereolabs.com/zed/specs/`
[11]Source: `https://software.intel.com/en-us/realsense/zr300`
[12]2018

Table 2.1 – RGB-D sensors comparison.

Characteristic	**kinect xbox 360**	**DUO MLX** (integrated IMU)	**ZED**	**ZR300**
Output Resolution	640 x 480 x 24 bpp 4:3 RGB @ 30fps 640 x 480 x 16 bpp 4:3 YUV @ 15fps	752 x 480 @ 56 fps	2x (1920 x 1080) @ 30fps	2 x VGA @ 60 fps
Depth Range (m)	0.8 to 4.0	0.23 to 2.5	1 to 15	0.4 to 2.8
Baseline (mm)	75	30	120	20
Dimensions (mm) (w×d×h)	279 × 50,8 × 25	52×24×13	175×30×33	9.5 × 101.56 × 3.8
Weight (grams)	750	12.5	159	-
System requirements	1.9 GHz CPU 4GB RAM	Modern Processor Intel i5/i7, AMD or ARM Minimum 4GB RAM	Dual-core 2,4GHz or faster processor Minimum 4GB RAM Nvidia GPU with compute capability > 3.0	Intel Joule compute module with Ubuntu 16.04
price[12]**($)**	33	595	449	129

has a small size and a good resolution but it is more expensive than the kinect. Despite its bigger dimension and weight, the kinect is the cheaper and the easiest to use. Taking into account the UAVs' mission requirements and purpose, we make the choice to use the kinect. Yet, the SLAM algorithm is not restricted to a defined type of a RGB-D sensor.

5.2 Proposed approach

The exploration task requires the UAV to implicitly maintain an accurate estimate of its pose in addition to a map of the observed environment. Figure 2.16 shows the outline of the localization and mapping processes. $\mathbf{C}_t$ and $\mathbf{D}_t$ represent, respectively, the colored and depth measurements gathered from the RGB-D sensor. $\mathbf{p}_i$ is the robot's estimated pose, $\mathcal{P}_S$ represents the 3D point cloud computed by the SLAM system, and $\mathcal{O}$ and $\mathcal{L}$ contain the 3D and 2D grid map, respectively.

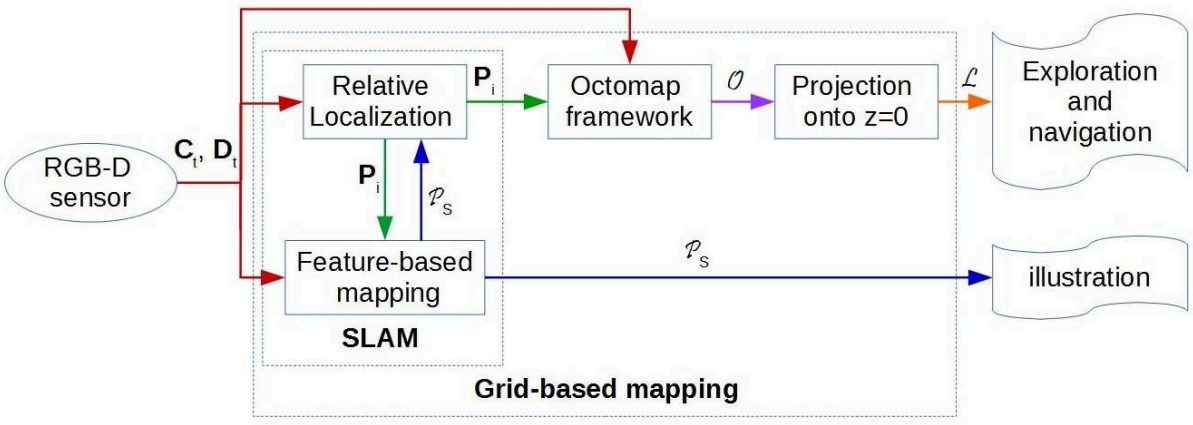


Figure 2.16 – Localization and Mapping layers outline.

5.2.1 SLAM module

The SLAM module in Figure 2.16 provides an estimate of the robot's 3D coordinates $\mathbf{p}_i$ w.r.t. the local reference frame ${}^{W}F_i$ (See Eq.2.8).

$$\mathbf{p}_i = [x, y, z, q_x, q_y, q_z, q_w]^\top \tag{2.8}$$

A sparse reconstruction of the environment is created using XYZ 3D point cloud. For pose estimation and map construction, among the existing RGB-D SLAM approaches, the ORB-SLAM2 vision-based framework [Mur-Artal and Tardós, 2017a] is used. This approach has shown promising results for pose estimation. The ORB-SLAM2 architecture is composed of threads and modules as represented in Figure 2.17. It mainly contains:

- Tracking thread for localization: To find and to match ORB feature, and to minimize the re-projection error.
- Local mapping thread for mapping: To optimize map using Bundle Adjustment.
- Loop closing thread in charge of detecting loops: To correct the accumulated drift using a pose graph optimization. This thread calls another thread in charge of performing full Bundle Adjustment (BA) to compute the optimal structure and the motion solution.

These three main threads work in parallel. A place recognition module is used for re-localization and loop detection.

5.2.2 Grid-based mapping module

Using the estimated motion and the point cloud from the RGB-D sensor, a 3D occupancy grid is built during the grid-based mapping process (See Figure 2.16). For that, the

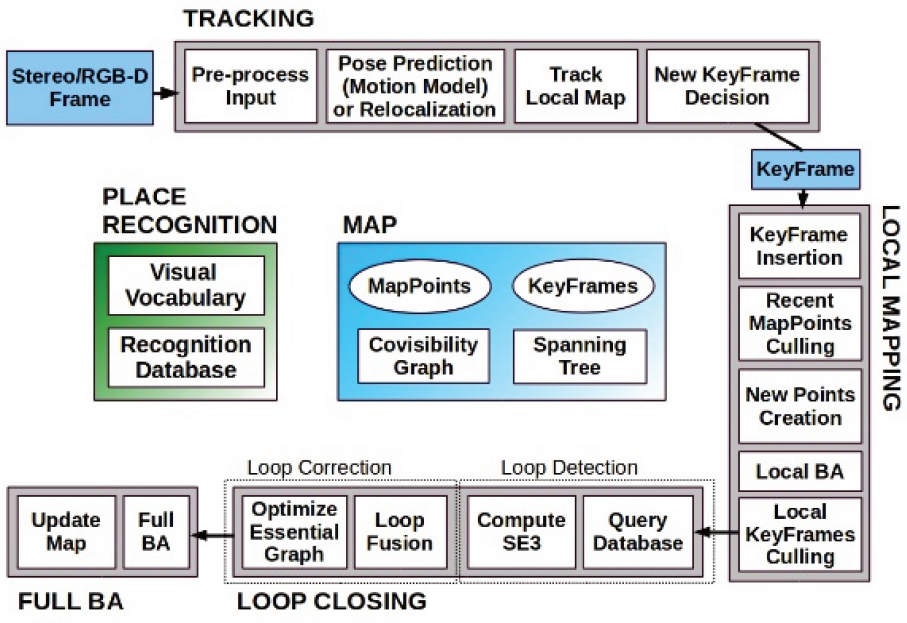
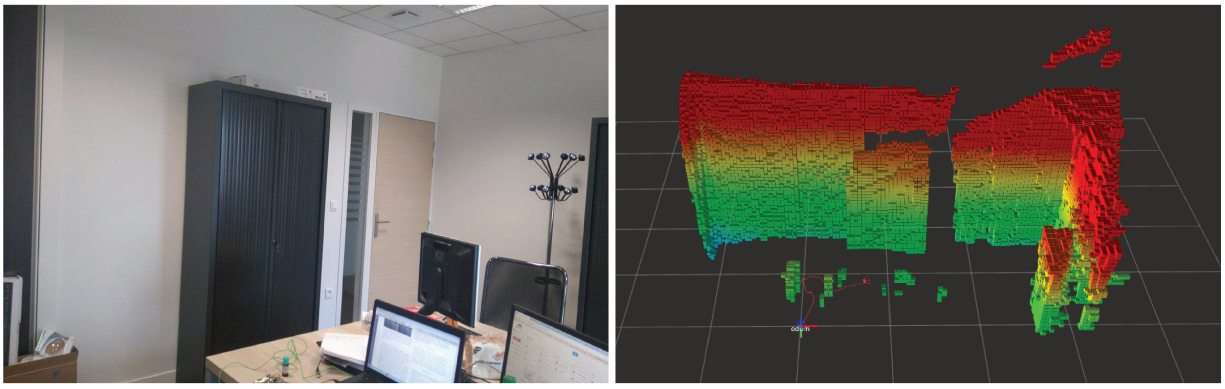


Figure 2.17 – ORB-SLAM2 threads and modules. Image from [Mur-Artal and Tardós, 2017a].

Octomap framework [Hornung et al., 2013] is used. It is a largely adopted 3D occupancy grid mapping framework. Furthermore, Octomap framework is able to perform grid mapping in challenging scenarios such as feature-less environments or fast camera motion [Endres et al., 2014]. Figure 2.18 shows an example of a desk environment representation using the Octomap framework.



(a) Real environment. (b) 3D occupancy grid environment.

Figure 2.18 – An example of environment representation using Octomap mapping framework (Resolution=0.05m).

The environment is approximated to a 3D voxels grid $\mathcal{O}$ where each voxel $\mathbf{o} \in \mathcal{O}$ is represented by its centroid (See Figure 2.19).

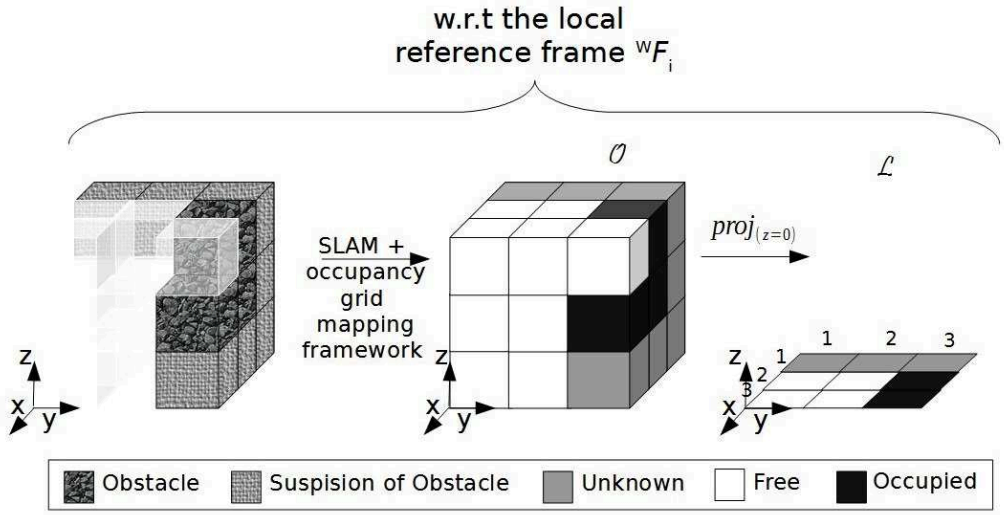


Figure 2.19 – Map structure evolution during the SLAM process: From point cloud in the environment to 3D voxels $\mathcal{O}$ to 2D cells $\mathcal{L}$.

The visual sensor provides 3D point cloud to represent the environment. A 3D occupancy grid is then constructed by applying the SLAM approach and the occupancy grid mapping framework to the point cloud. Using the sensor measurements, voxels of the 3D occupancy grid map are labeled to unknown $\mathbf{o}_u$, free $\mathbf{o}_f$ or occupied $\mathbf{o}_o$. This 3D occupancy grid $\mathcal{O}$ is down-projected onto the plane $z = 0$ of the local frame ${}^{W}F_i$ to get a 2D cell grid $\mathcal{L}$ (See Equation 2.9).

$$\mathcal{L} = proj_{(z=0)}(\mathcal{O}) \tag{2.9}$$

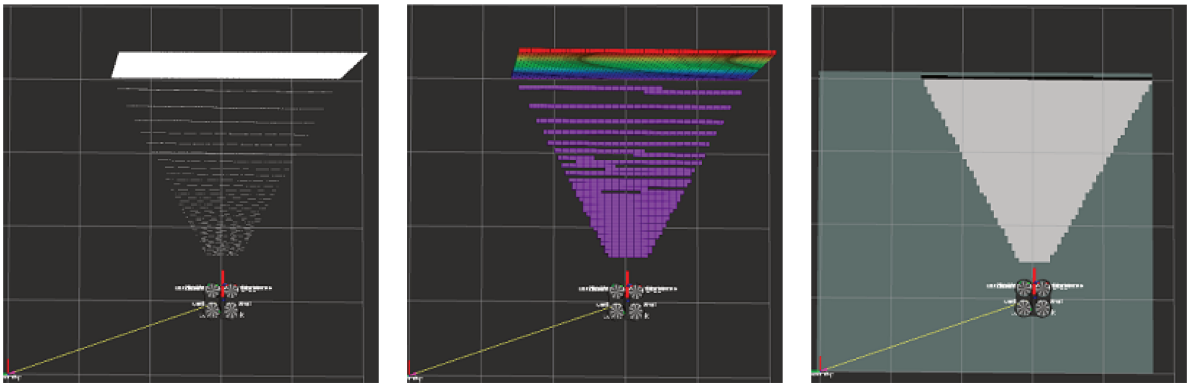
The cells are occupied as soon as there is an occupied voxel in the z cell range. And, they are free if all voxels in the z cell range are so. Indeed, in a z cell range, if there is occupied and unknown or free cells (See Equation 2.10), the 2D projected cell will be occupied; now if there is unknown and free cells (See Equation 2.11), the 2D projected cell will be unknown; and if there is only free cells (See Equation 2.12), the 2D projected cell will be free.

$$\mathbf{l}_o(x, y) = \mathbf{o}_o(x, y, z) \wedge (\mathbf{o}_u(x, y, z) \vee \mathbf{o}_f(x, y, z)), \tag{2.10}$$

$$\mathbf{l}_u(x, y) = \mathbf{o}_u(x, y, z) \wedge \mathbf{o}_f(x, y, z), \tag{2.11}$$

$$\mathbf{l}_f(x, y) = \mathbf{o}_f(x, y, z), \tag{2.12}$$

Where $\wedge$ and $\vee$ represent AND (conjunction) and OR (disjunction) boolean operations, respectively.
Figure 2.20 shows the evolution of the map structure of an UAV that is executing a mission within a simulated environment. A Kinect camera is mounted on-board the UAV. These maps represent the sensed environment in the view frustum of the sensor at $t = 0s$, that is, at the beginning of the mission.



(a) 3D point cloud. (b) 3D occupancy grid map. (c) 2D occupancy grid map.

Figure 2.20 – Example of one UAV's map structure evolution.

5.3 Results and discussions

5.3.1 Implementation details

The simulations are performed using Robot Operating System[13] (ROS) [Quigley et al., 2009] running on a 2.60GHz i7 Linux machine. For the quad-rotor simulation, the AR-Drone model[14] equipped with an RGB-D camera in a forward-looking configuration, is used. A bounded unknown environment is generated using the *Gazebo* simulator[15]. Figure 2.21 shows the implementation details related to the SLAM evaluation.

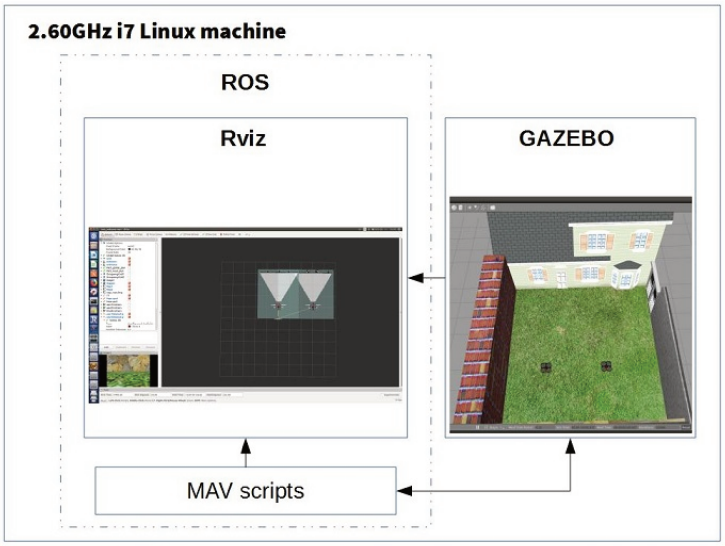


Figure 2.21 – Implementation details of multiple UAVs on a single Linux machine.

[13]Source: `http://www.ros.org/`
[14]Source: `http://wiki.ros.org/ardrone_autonomy`
[15]Source: `http://gazebosim.org/`

5.3.2 Simulation results

The ORB-SLAM2 approach [Mur-Artal and Tardós, 2017a] has been implemented and evaluated during an exploration mission[16]. Each robot runs its own ORB-SLAM2 within its local reference frame ${}^{W}F_i$. Using the *Gazebo* virtual environment and despite the need of structure and texture, the ORB-SLAM2 has been able to perform localization in a simulated area.

5.3.2.1 Parameters setting When using a relative localization, the motion is estimated by comparing the features in the current and previous frame. If the UAV's speed is too high, the tracking of these features is lost. Thus, some parameters related to the UAV's motion have been tuned. Table 2.2 resumes tests to set some of the required parameters (linear velocity v_i and angular velocity ω_i) to perform visual SLAM without tracking loss or – at least – a fast re-localization. For a tracking loss SLAM, the linear velocity v_i is set to $[0.1, 0.2]m.s^{-1}$ and the angular velocity ω_i to $0.1rad.s^{-1}$

Table 2.2 – SLAM behavior while modifying linear and angular velocity. T.L: Tracking Loss; RL: Re-Localization.

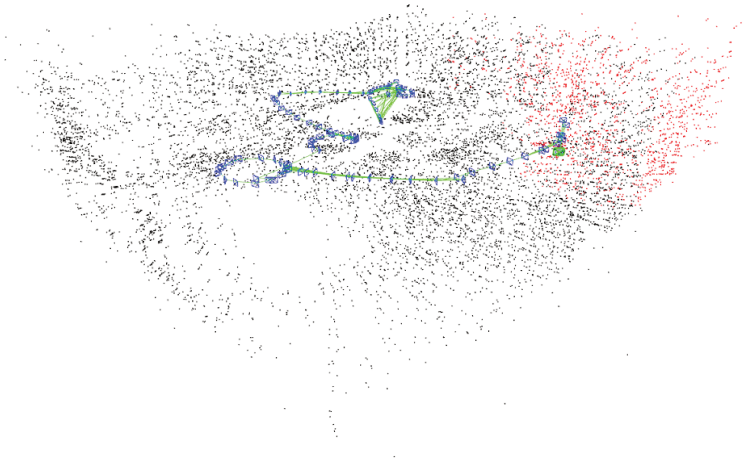
$v_i(m.s^{-1})$ \ $\omega_i(rad.s^{-1})$	0.1	0.2	0.3
0.1	No T.L.	No T.L.	T.L., R.L.
0.2	No T.L.	T.L., R.L.	T.L.
0.3	T.L., R.L.	T.L.	T.L.

5.3.2.2 SLAM performances using one UAV Figure 2.22 shows the SLAM system performances during one robot exploration. The drift as well as the trajectory errors are limited due to the loop-closure algorithm performed within ORB-SLAM2.

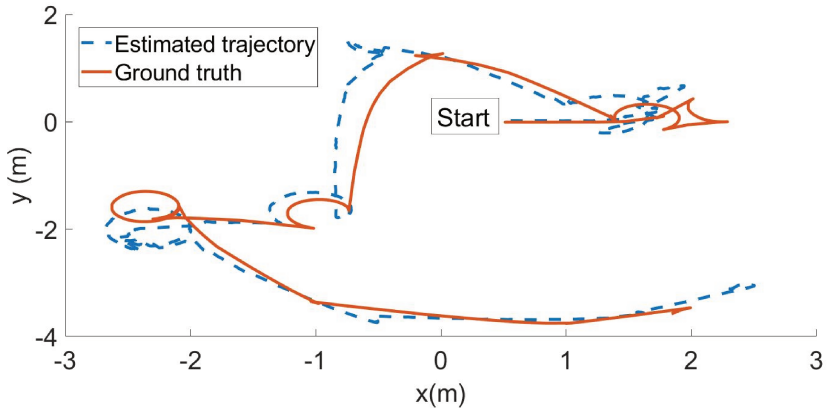
5.3.2.3 SLAM performances using two UAVs Results for exploration mission, using a fleet composed of two UAVs running each one the SLAM algorithm, are presented in Figure 2.23. As expected, the exploration time using one UAV is greater than using two UAVs. An important drift occurs at the end of the UAV_2's trajectory because it did not visit a known place and therefore it could not rectify its trajectory with a loop closure optimization.

Figure 2.22a and Figure 2.23a contain a 3D sparse point cloud along with robot trajectories computed by SLAM. The sparse representation of the environment is only used for illustration, not for exploration nor for navigation.

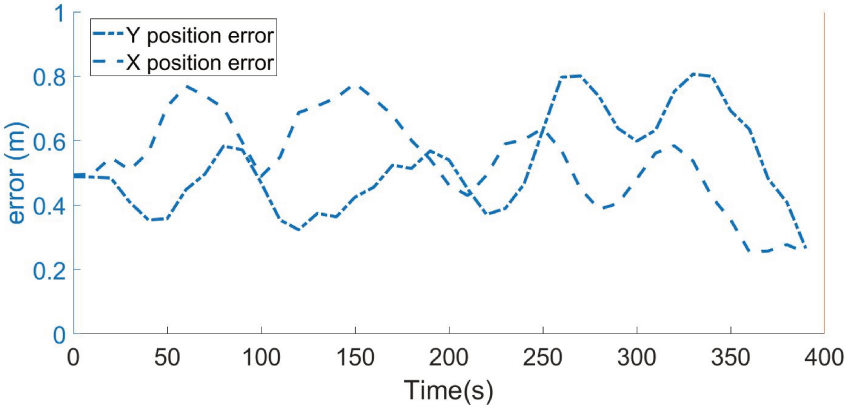
[16]Source: `https://github.com/raulmur/ORB_SLAM2`



(a) Sparce reconstruction of the environment. The blue markers represent the keyframes. The green lines represent the keyframe matching. The black and red points (present in the robot's current view frustum) represent the sparse reconstruction of the environment.

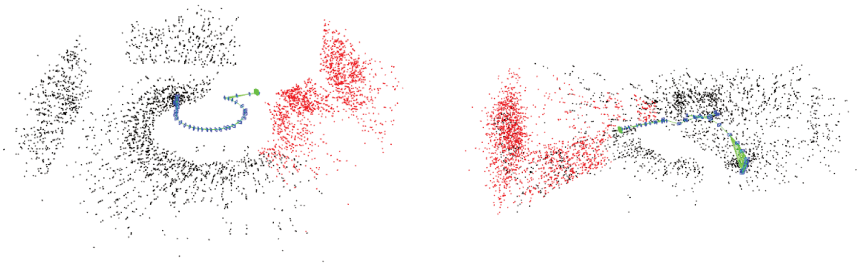


(b) The estimated trajectory versus the Ground truth.

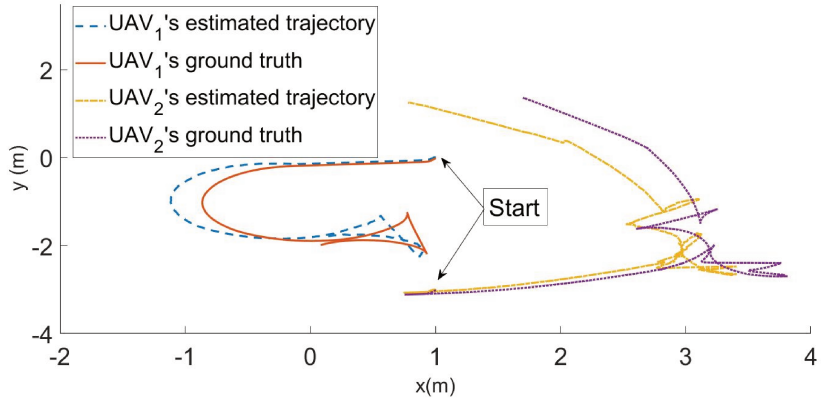


(c) The estimated position errors during the mission.

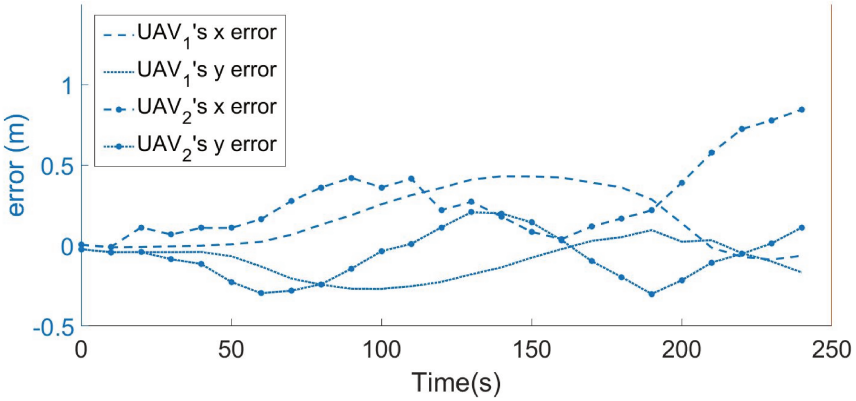
Figure 2.22 – One UAV exploration using SLAM algorithm. The kinect's maximum range is 4m, the UAV's linear velocity $v_i \in [0.1, 0.2]$ $m.s^{-1}$ and angular velocity $\omega_i = 0.1$ $rad.s^{-1}$. The RMSE(x)=0.2968 and RMSE(y)=0.2944.



(a) Sparce reconstruction of the environment of UAV_1 (left) and UAV_2 (right). The blue markers represent the keyframes. The green lines represent the keyframe matching. The black and red points (present in the robot's current view frustum) represent the sparse reconstruction of the environment.



(b) The estimated trajectory versus the Ground truth.



(c) The estimated position errors during the mission.

Figure 2.23 – Two UAVs cooperative exploration performing each one SLAM algorithm. The kinect's maximum range is 4m, the UAV's linear velocity $v_i \in [0.1, 0.2]$ $m.s^{-1}$ and angular velocity $\omega_i = 0.1$ $rad.s^{-1}$. For UAV_1, the RMSE(x)=0.2419 and RMSE(y)=0.1415 ;for UAV_2 the RMSE(x)=0.3649 and RMSE(y)=0.1614.

6 Conclusion

This chapter has addressed the problem of simultaneously determining the robot's location and mapping the environment, while the robot moves. For that, a brief state of the art on the existing approaches have been presented. Then, we detailed two approaches for the SLAM problem.

A first approach was an inertial SLAM where a monocular camera was fused with an IMU. Results have shown that combining these two sensors improves the pose estimation and reduces the drift.

The second approach consists in using an RGB-D camera as the main sensor for SLAM. For that, the ORB-SLAM2 approach was implemented and evaluated using one and two UAVs in an exploration mission. Results has shown that the adopted approach allows to minimize the drift and consequently reduce the pose estimation error.

The RGB-D SLAM will be used in the next chapter as the relative localization algorithm for each UAV during the cooperative exploration missions.

Coordinated exploration

Contents

1 Introduction

The exploration and mapping of large areas is an active field of research in aerial robotics. It consists in constructing a 3D model representation of the workspace as robots progress within it. Recently, the key subject in the exploration problem is the cooperative deployment of fleet of robots that promise enhanced performances compared to single robot exploration.

In this chapter, we address the problem of cooperative exploration strategies with no a-priori knowledge of the environment. We will try to answer the question: Where should each robot move next?

2 Related work

Recently, several works have proposed solutions for exploration using multi-robot teams to reduce the mission time and to increase the scalability [Bautin et al., 2012, Jensen and Gini, 2013, Yan et al., 2014]. Hence, the challenge is to have an efficient cooperation among the agents in the fleet while maintaining communication [Rooker and Birk, 2007]. For that, existing approaches can be centralized, that is, one robot in the fleet is responsible for assigning targets [Burgard et al., 2000]. In [Schmuck, 2017], a central server with increased computational resources is adopted to receive, treat, optimize and send back information

to other robots. Other works, such as [Yuan et al., 2010, Sheng et al., 2006], use distributed approaches where each robot chooses its own target. Another trend is to switch from individual to cooperative exploration behavior when robots are not able to converge to a local minimum at a satisfying rate [Wu and Zhang, 2012]. In [Konolige et al., 2003], authors propose to consider four possibilities taking into account robots' interactions in order to construct a distributed map. The situations involved are: no interaction (robots are not within each others' communication range), hypothesis generation (there is an interaction between the robots but they do not know their respective location), hypothesis verification (there is an interaction between the robots and they propose an hypothesis about their location), and coordinated exploration (there is an interaction between robots and they do share their maps).

2.1 Robot-to-target assignment

The mapping algorithm is performed while a robot attempts to reach a target. So, for an effective environment mapping, the target should be chosen carefully. There is a wide variety of goal assignment strategies to affect one robot to a target. The majority of them are centralized and use a cost function to compute the utility of reaching a target.

In the greedy assignment [Yamauchi, 1998], each robot chooses a target depending on its cost function without coordinating with other robots. Hence, one target can be visited by different robots. To solve this issue, already chosen targets can be discarded before being considered for assignment by others using Broadcast of Local Eligibility (BLE) [Werger and Matarić, 2000] also called Iterative Assignment. But, this method does not necessarily produce the optimal solution since it depends on the order of robots.

The K-means method [Solanas and Garcia, 2004] consists in dividing the environment to explore into regions of the same number of robots, then, assigning one robot to the closest region where it will choose a target from the frontiers depending on a cost function.

The Hungarian method [Pal et al., 2011] proposed by [Kuhn, 1955], solves the worker-task assignment written in the form of $n \times n$ matrix C where $c_{i,j}$ is the cost of the task j assigned to the worker i. The optimal assignment is found with time complexity $O(n^3)$. This algorithm requires that the number of workers is equal to the number of tasks which cannot be guaranteed. Else, imaginary robots or targets can be added to satisfy the assumption, and skipped later in the selection.

Authors in [Nanjanath and Gini, 2006] present an auction-based method to assign tasks to a group of robots. The distribution of tasks is accomplished by means of a first-price reverse auction which means that the auctioneer is the buyer. One task is auctioned one at a time by priority. Then, the auctioneer selects the best bid and assigns the task to the corresponding bidder. The algorithm is well adapted to dynamic environments, where unexpected obstacles might prevent a robot from reaching its target.

In [Zhao et al., 1996, Leigh et al., 2007], a genetic algorithm is used to optimally assign robot to tasks. It is an NP-hard problem since it is considered as a generalized two-dimensional multi-type bin packing problem.

Authors in [Faigl et al., 2012] present a new approach called Multiple Traveling Salesman Problem (MTSP) for robot-to-target assignment in multi-robot exploration problem. It consists in clustering an environment, determining the cost of TSP distance for each pair of robot cluster, assigning goals from non empty cluster, and finally fixing goals for empty clusters. This approach was compared with greedy, iterative and Hungarian method. The MTSP presents competitive results regarding the total computational requirements and the increasing number of robots.

In [Kulich et al., 2015], a comparison of some assignment strategy used for multi-robot system is done. The compared strategies include Hungarian method, Greedy, BLE method and K-means clustering. Results show that Hungarian method outperforms other approaches in majority of cases. Unlike the Hungarian method, the iterative assignment can be implemented in a distributed environment. Also the Hungarian methods are computationally heavy compared with the simple greedy algorithm which is preferred in applicable scenario.

2.2 Utility function

Most of the robot-to-target assignments are based on an utility function that defines the benefits that a robot have to reach this target, taking into account the mission's aim [Burgard et al., 2000]. The work proposed in [Benavides et al., 2016] presents a new utility function that takes into account the traveling cost to the target and the connectivity utility. This allows a trade off between minimizing the amount of exploration time and the connectivity. To speed up velocity, authors in [Cieslewski et al., 2017] propose a rapid frontier selection technique to select goals from the robot's field of view. This approach minimizes the overall mission time by minimizing the change in velocity of the robot. Nonetheless, this approach increases the total path length traveled. In [Heng et al., 2015], maximizing the reconstructed model is favored over the mission time. Furthermore, the proposed approach solves simultaneously exploration and coverage problems in order to maximize the completeness of the reconstructed model. Whereas in [Simmons et al., 2000], the aim is the maximization of the utility of targets that minimizes the potential for overlap in information gain amongst members of the fleet. The utility of reaching a target depends basically on the aim of the mission while taking into account some additional constraints such as time, completeness of the map, limited sensor and communication range, or number of robots.

3 Proposed exploration strategy

3.1 Overview

In a Multi-UAV system, the exploration strategy needs to be cooperative to maximize the efficiency. The main objective proposed here, is to cooperatively choose specific regions to be simultaneously explored using a frontier-based approach. Commonly, this is done

by selecting candidate targets and assigning them to each robot in an optimized manner. Figure 3.1 shows the proposed pipeline exploration process performed by each UAV.

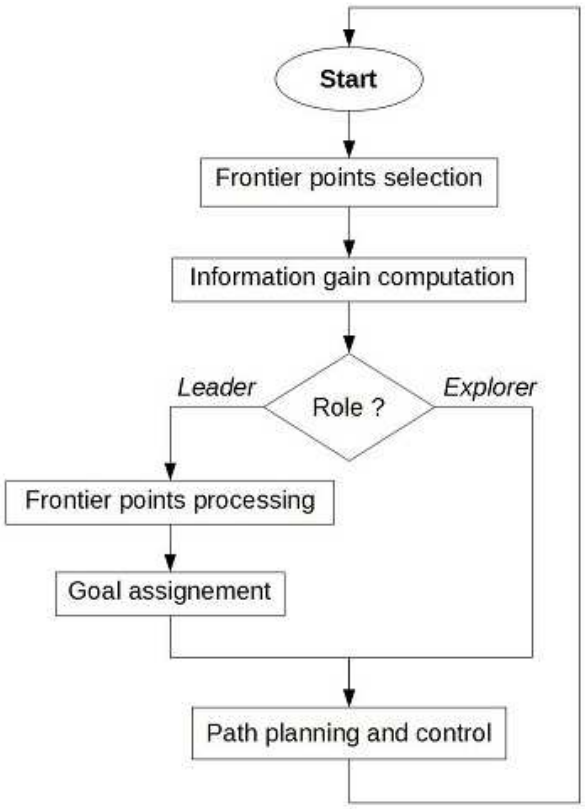


Figure 3.1 – Proposed exploration process pipeline.

Each UAV selects the frontier points of its constructed local map during the SLAM step. Then, it computes the corresponding information gain of these points. If the UAV's role is an *explorer*, it would passively wait until receiving instructions; else, if it acts as a *leader*, it would process the collected frontier points, and would assign a target to each robot in the group. Given a target, UAVs will plan a specific path to reach it.

During these steps, the map structure evolves from a projected 2D grid map obtained from the localization and mapping module, through frontier points during the frontier selection process, then candidate targets during the frontier processing step, to a final set of selected targets to reach. The evolution of the map structure is illustrated in Figure 3.2. Algorithm 3.1 describes the main steps performed during the exploration.

Algorithm 3.1: Exploration strategy for coordinated Multi-UAV.

1: From cells $\mathbf{l}_l \in \mathcal{L}$, select frontier points $\mathbf{f}_{i,j} \in \mathcal{F}$ and compute their respective information gain $\boldsymbol{I}(\mathbf{f}_{i,j})$.
2: Process frontier points $\mathbf{f}_{i,j}$ to get candidate goals $\mathbf{t}_k \in \mathcal{G}$ (See Algorithm 3.2).
3: Assign UAV$_i$ with target k (See Algorithm 3.5).
4: Send targets to the corresponding robots.

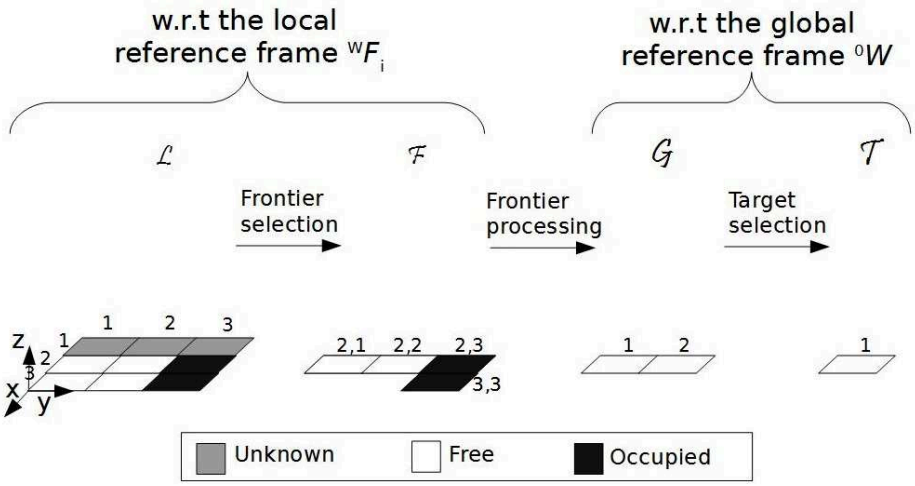


Figure 3.2 – Map structure evolution during the exploration process: From 2D cells $\mathcal{L}$ to 2D frontier cells $\mathcal{F}$ to candidate frontier cells/points $\mathcal{G}$ (candidate targets) to 2D target cells/points $\mathcal{T}$.

3.2 Frontier points selection

The frontier selection process is used to define the frontiers of regions bounded by obstacles or unknown spaces (See Figure 3.3).

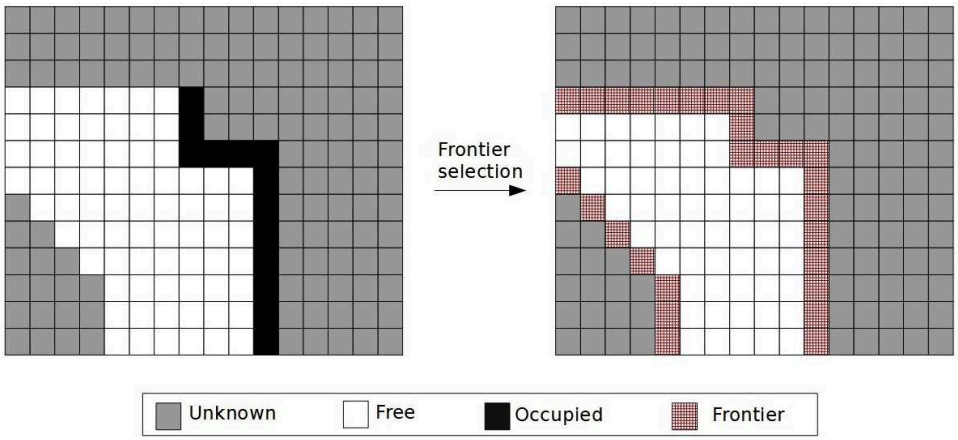


Figure 3.3 – Frontier cells/points selection of a 2D occupancy grid map.

The frontier cells $\mathbf{f}_{i,j} \in \mathcal{F}$ are selected from the set of cells $\mathcal{L}$ ($\mathcal{F} \subset \mathcal{L}$) such that they are either:

- Free $\mathbf{l}_f$ and adjacent to unknown.
- Labeled as occupied $\mathbf{l}_o$. The Occupied cells $\mathbf{l}_o$ are considered as frontier cells to be able to perform frontier processing in the next step. They could not be chosen as target and will be discarded later.

In Figure 3.2, the frontier cells are: $\mathbf{l}_f(2,1)$, $\mathbf{l}_f(2,2)$, $\mathbf{l}_o(2,3)$, and $\mathbf{l}_o(3,3)$. Thus, for a cluster $\mathcal{C}$ containing UAV_i, the frontiers are:

$$\mathcal{F} = \{\mathbf{f}_{i,1}(2,1), \mathbf{f}_{i,2}(2,2), \mathbf{f}_{i,3}(2,3), \mathbf{f}_{i,4}(3,3)\}, \tag{3.1}$$

3.3 Information gain computation

In frontier-based exploration approaches, only cells adjacent to unknown ones may be defined as candidate frontier points and are likely to be chosen as targets. Thereby, the information gain is associated to each of them in order to estimate the utility of reaching each frontier. This corresponding information gain can be defined in different manners depending on the mission purpose. Authors in [Burgard et al., 2005] propose to use a probability function to reduce an assigned constant value taking into account the relative distance to the UAV's pose. This strategy is general and does not take into account the updated explored cells. The approach proposed in [Heng et al., 2015] affects, to the information gain, the number of unknown and not occluded cells in the view frustum of the target. This method depends on the real estimate of information gained when visiting the considered pose. However, it requires more computation.

In the proposed strategy, the information gain is allocated so that it defines the amount of unknown cells surrounding the target (See Figure 3.4). It is a non-metric value that counts the number of cells labeled as unknown $\mathbf{l}_u$ from the 48 cells around the frontier point.

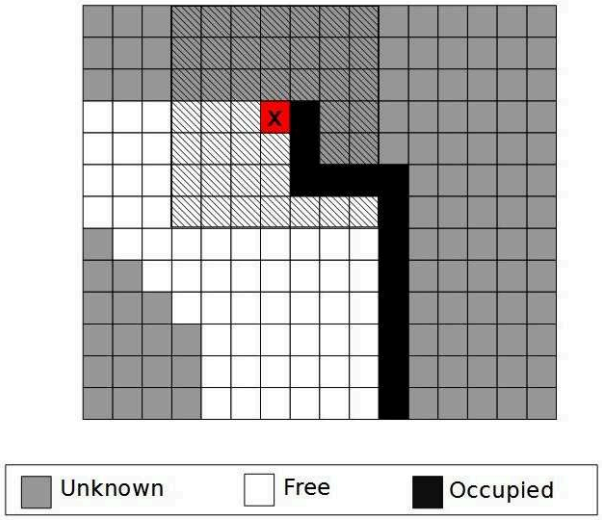


Figure 3.4 – The information gain computation of a frontier point $\mathbf{x}$ (in red). The hatched cells represent the 48 surrounding cells of the frontier point $\mathbf{x}$. The information gain of $\mathbf{x}$ is: $\boldsymbol{I}(\mathbf{x}) = 25$.

3.4 Frontier points processing

All frontier points $\mathbf{f}_{i,j} \in \mathcal{F}$ of UAV$_i$ in the cluster $\mathcal{C}$ with $i \in [1..n_c]$, are collected. Points in $\mathcal{F}$ are then processed using Algorithm 3.2 to get candidate frontier points considered as candidate targets $\mathbf{t}_k \in \mathcal{G}$ with $k \in [1..n_g]$ (See Figure 3.2)

Algorithm 3.2: Frontier processing algorithm.

Input: Frontier points $\mathbf{f}_{i,j} \in \mathcal{F}$ of UAV$_i$ with $i \in [1..n_c]$.

Output: Candidate targets $\mathcal{G}$.

1: $p_u = \bigcup_{i=1}^{n_c} \mathbf{f}_{i,j}$.
2: $p_i = \bigcap_{i=1}^{n_c} \mathbf{f}_{i,j}$.
3: $\mathcal{G} = p_u \setminus p_i$.
4: Delete the obstacle frontier points $\mathbf{f}_{i,j}(x, y) = \mathbf{l}_o(x, y)$ from $\mathcal{G}$.
5: **return** $\mathcal{G}$.

Figure 3.5 shows an example of frontier processing with two UAVs' map to get the candidate targets. The obstacle frontier points $\mathbf{f}_{i,j}(x, y) = \mathbf{l}_o(x, y)$ – labeled as occupied – are only kept to compute the intersection of frontier points. Only the free frontier cells $\mathbf{l}_f$ can be considered as candidate target.

When using local frontier points instead of local maps, the frontier process replaces the map matching process where the aim is to clear overlapping areas. Hence, in the frontier processing step, the frontier points that belong to overlapping areas are cleared. Therefore, using frontier points allows important memory saving. To compute the frontier points that belong to the overlapped areas (Steps 1, 2 and 3 in Algorithm 3.2), we propose two approaches with convex shapes and concave shapes assumptions.

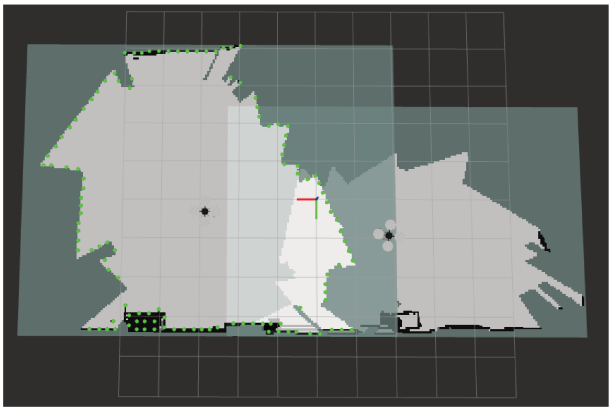
3.4.1 First approach: Convex shape map

In this approach, map shapes are considered convex, which simplifies the intersection computation in Algorithm 3.3. Figure 3.6 shows two examples of applying frontier processing algorithm while assuming convex shapes with two and three UAVs' map.

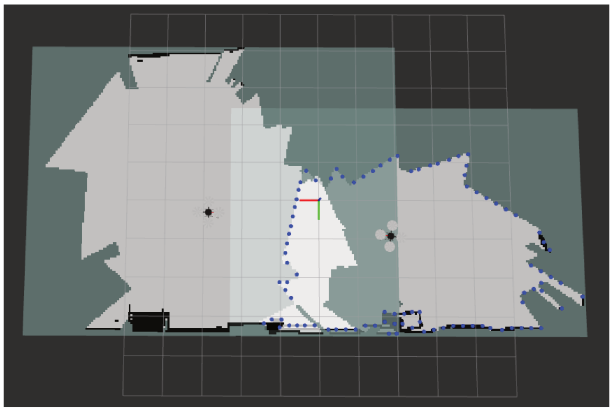
This algorithm is relatively easy to apply, but results show that it does not perform well. The convex assumption leads to some false overlapped frontier points. Consequently, some unknown areas will not be visited since their corresponding points are removed and, thus, will not be assigned as a target.

3.4.2 Second approach: Concave shape map

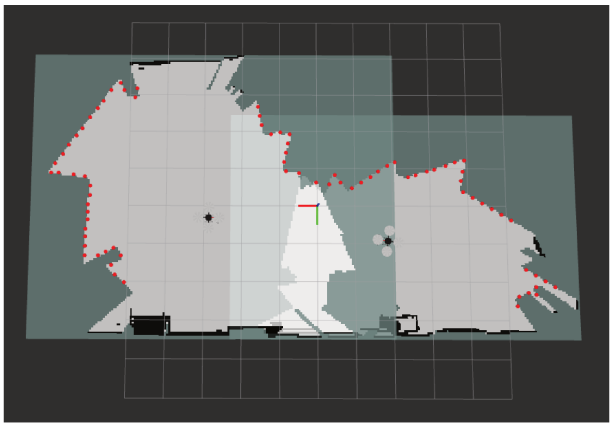
In the second approach, we make the assumption of concave shapes for the UAVs' map to compute their intersection in Algorithm 3.4. Figure 3.7 shows two examples of applying frontier processing algorithm while assuming convex shapes with two and three UAVs' map.



(a) Frontiers $\mathbf{f}_{1,j}$ of UAV_1.



(b) Frontiers $\mathbf{f}_{2,j}$ of UAV_2.



(c) Candidate targets $\mathcal{G}$.

Figure 3.5 – Frontier points processing example: (a) and (b) represent the frontier points of UAV_1 and UAV_2, respectively. (c) represent the candidate targets. The process consists in removing frontier points that belong to overlapped areas, and those that are adjacent to obstacles.

Algorithm 3.3: Intersection computation algorithm with convex shapes assumption.

Input: $Vect1$, $Vect2$
Output: $Vect\_Final$

for $i \in Vect1$ **do**
  $inf = 0$, $sup = 0$;
  for $j \in Vect2$ **do**
    if $Vect1(i).x < Vect2(i).x$ **then**
      $inf++$;
    else
      $sup++$;
    end if
  end for
  if $inf > 0$ and $sup > 0$ **then**
    $Vect\_inter.push\_back(Vect1(i))$
  end if
end for
for $i \in Vect\_inter$ **do**
  $inf = 0$, $sup = 0$;
  for $j \in Vect2$ **do**
    if $|Vect\_inter(i).x - Vect2(i).x| < 0.5$ **then**
      $inf++$;
    else
      $sup++$;
    end if
  end for
  if $inf > 0$ and $sup > 0$ **then**
    $Vect\_Final.push\_back(Vect\_inter(i))$
  end if
end for

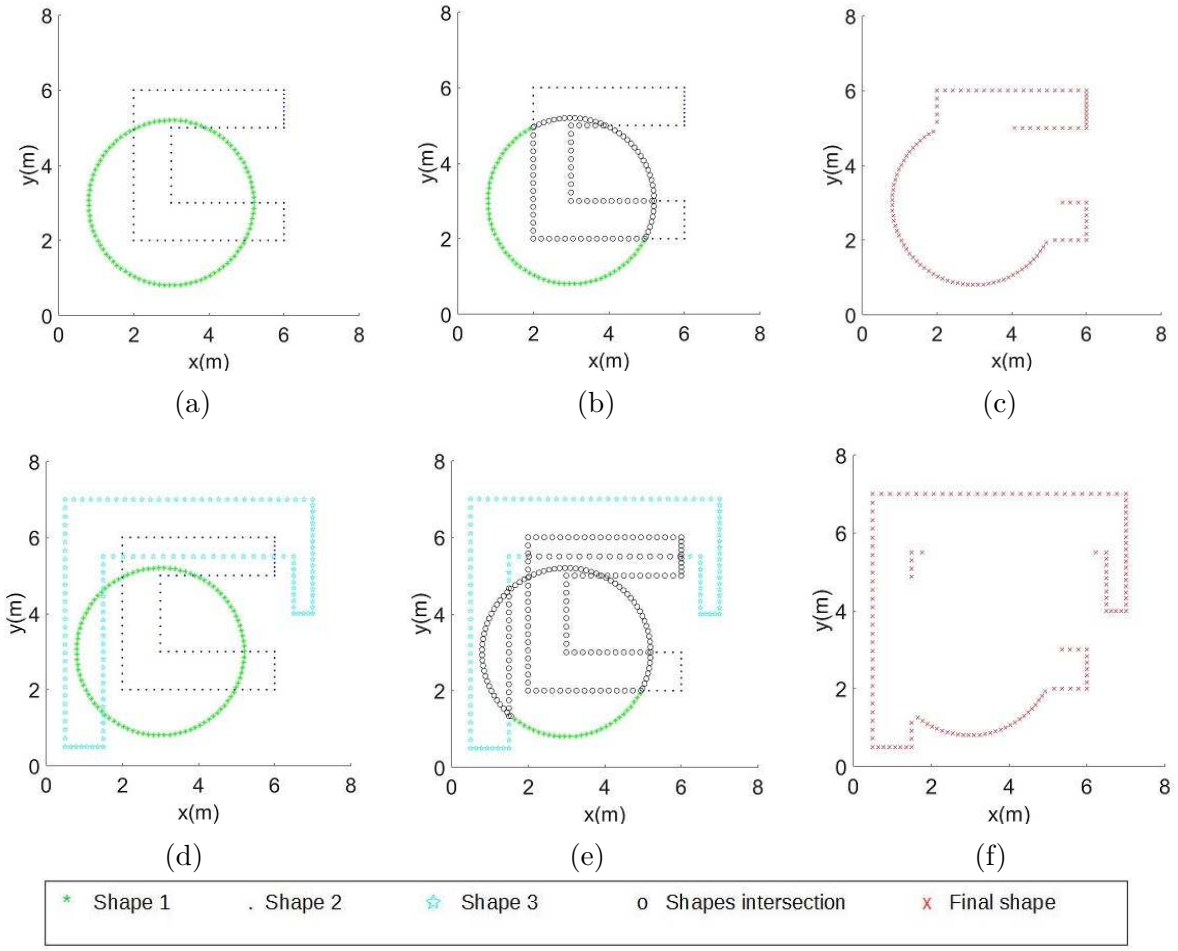


Figure 3.6 – Frontier points processing wile assuming two convex map shapes in the first line and three in the second line. (a) and (d) represent two and three convex map shapes, respectively; (b) and (e) represent the frontier points in overlap (intersection); and (c) and (f) represent the obtained frontier points after processing (final shape).

Algorithm 3.4: Intersection computation algorithm with concave shapes assumption.

Input: $Vect1$, $Vect2$
Output: $Vect\_Final$

for $i \in Vect1$ **do**
  $inf = 0$, $sup = 0$, $eq = 0$;
  for $j \in Vect2$ **do**
    if $Vect1(i).x < Vect2(i).x - 0.5$ **then**
      $inf++$;
    else if $Vect1(i).x > Vect2(i).x + 0.5$ **then**
      $sup++$;
    else
      $eq++$;
    end if
  end for
  if $(inf > 0$ and $sup > 0)$ or $eq > 0$ **then**
    $Vect\_inter.push\_back(Vect1(i))$
  end if
end for
for $i \in Vect\_inter$ **do**
  $inf = 0$, $sup = 0$, $eq = 0$;
  for $j \in Vect2$ **do**
    if $|Vect\_inter(i).y - Vect2(i).y| < 0.5$ **then**
      if $Vect\_inter(i).x < Vect2(i).x - 0.5$ **then**
        $inf++$;
      else if $Vect1(i).x > Vect2(i).x + 0.5$ **then**
        $sup++$;
      else
        $eq++$;
      end if
    end if
  end for
  if $(inf > 0$ and $sup > 0)$ or $eq > 0$ **then**
    $Vect\_Final.push\_back(Vect\_inter(i))$
  end if
end for

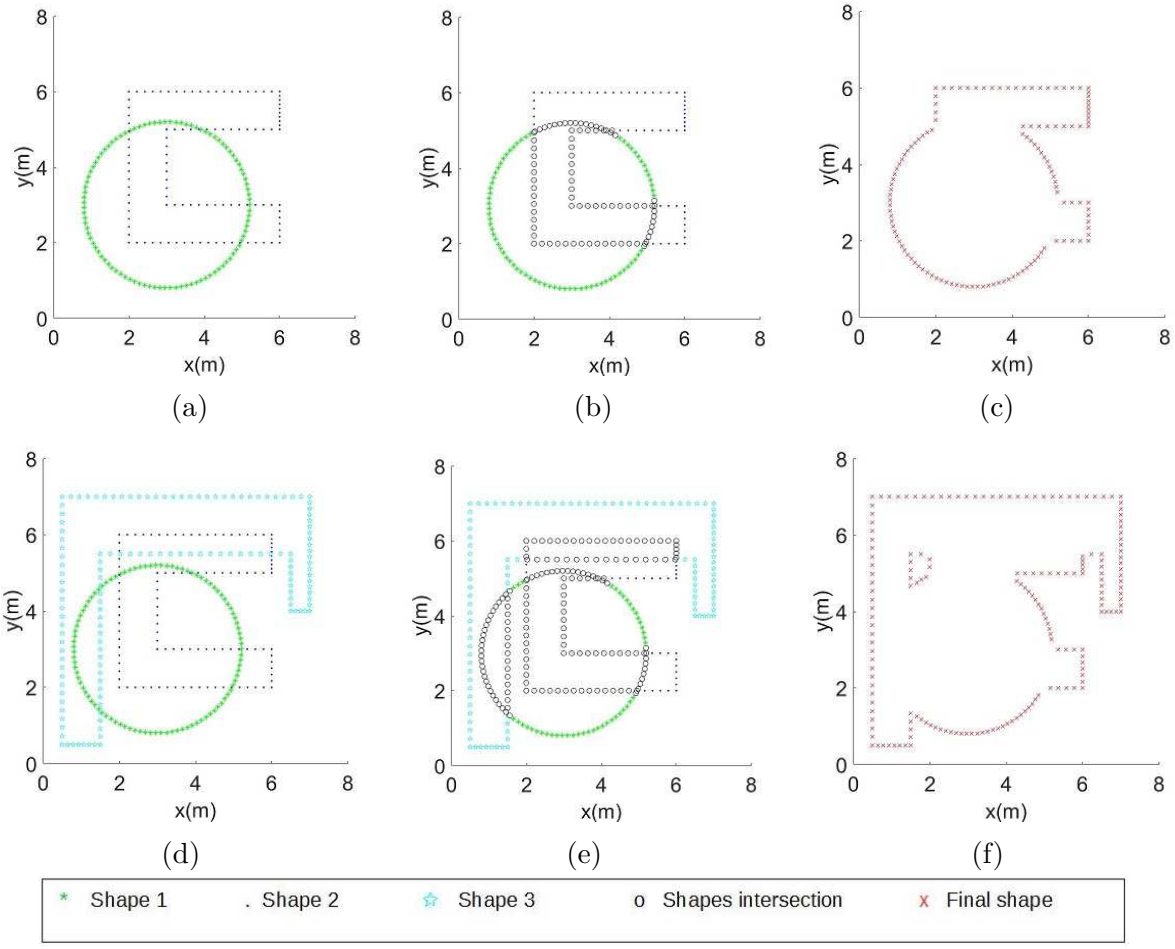


Figure 3.7 – Frontier points processing wile assuming two concave map shapes in the first line and three in the second line. (a) and (d) represent two and three convex map shapes, respectively; (b) and (e) represent the frontier points in overlap (intersection); and (c) and (f) represent the obtained frontier points after processing (final shape).

Results show that only the frontier points within overlapping areas are removed. Even if some ambiguous frontier points are located within/inside the global shape – which can be confusing –, they are not deleted. This algorithm, assuming concave shapes, performs better than the previous one that assumes convex shapes. Hence, for the frontier processing, the shape of local maps are assumed concave.

3.5 Utility function

The proposed utility function in Eq. 3.2 aims to simultaneously increase the explored area rate and to reduce the distance of each UAV to its corresponding target. The function also considers the average of distances between each robot in the group and this target in order to maximize distances among robots.

$$\boldsymbol{U}(UAV_i, \mathbf{t}_j) = \boldsymbol{I}(\mathbf{t}_j) \exp(-\lambda.(dmin(\mathbf{P}_i, \mathbf{t}_j) + \frac{n_c - 1}{d_{tot}})), \tag{3.2}$$

where UAV_i is the considered robot, $\mathbf{t}_j \in \mathcal{G}$ and $\boldsymbol{I}(\mathbf{t}_j)$ are respectively the candidate target and its corresponding information gain, $\lambda \in [0, 1]$ is a trade-off parameter, n_c is the number of UAVs in the cluster $\mathcal{C}$, and $d_{tot} = \sum_{k=1,k\neq i}^{n_c}(dmin(\mathbf{P}_k, \mathbf{t}_j))$ is the sum of the minimum distance from UAV_k's pose $\mathbf{P}_k$ to the candidate target j. The proposed utility function is inspired from [Heng et al., 2015] and it has been presented in our works [Mahdoui et al., 2017, Mahdoui et al., 2018]. This function performs a trade-off between rapid exploration and a precise filling the map using a tuning parameter λ. From Figure 3.8, we can notice that the larger λ, the less important the distance d_{tot}. Thus, a precise filling is favored over rapid exploration and vice versa.

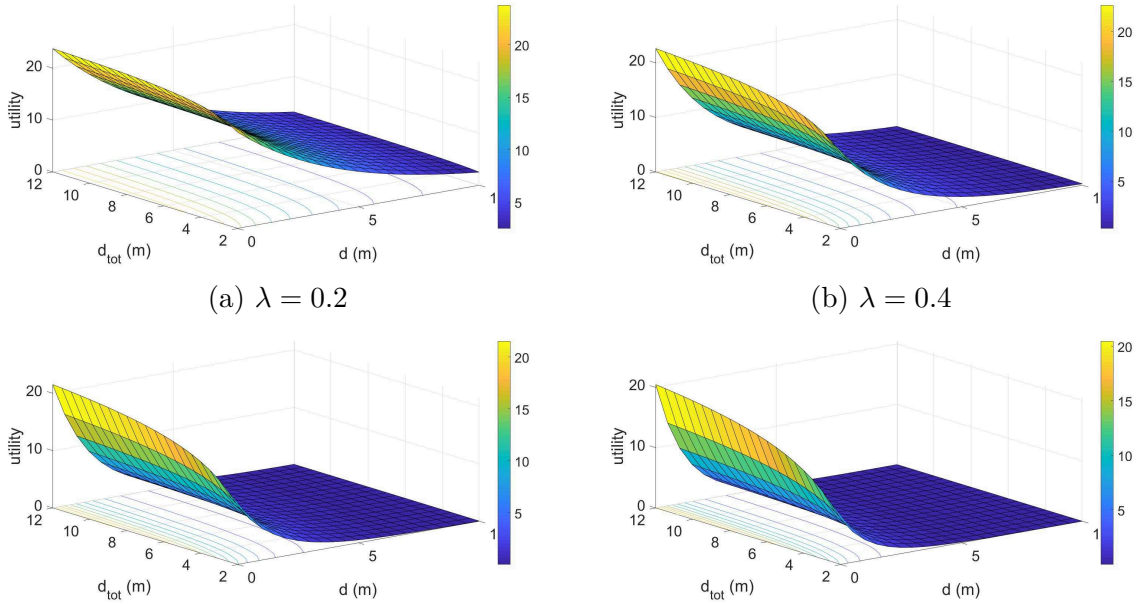
Regarding the Multi-UAV case, the utility function is based on the average of neighbors distances. As shown in Figure 3.8, with an information gain of $\boldsymbol{I}(\mathbf{t}_j) = 25$ and three UAVs in the cluster ($n_c = 3$); an increasing distance of UAV to the target will reduce the utility function. Whereas, the larger the average distance from other UAVs w.r.t. to the target, the more the utility. So the function tends to choose the closest target to the considered UAV; but at the same time, the farthest one from the others.

In the case of a single UAV, the utility function (See Eq. 3.3) tends to choose the closest target with the maximum of information gain:

$$\boldsymbol{U}(UAV_i, \mathbf{t}_j) = \boldsymbol{I}(\mathbf{t}_j) \exp(-\lambda.(dmin(\mathbf{P}_i, \mathbf{t}_j))), \tag{3.3}$$

The parameter $d_{tot} = \sum_{k=1,k\neq i}^{n_c}(dmin(\mathbf{P}_k, \mathbf{t}_j))$ represents the sum of the minimum distance between the target $\mathbf{t}_j$ and the neighbors' poses $\mathbf{P}_k$ with $k \in [1..n_c]\backslash i$. So, if $\mathbf{t}_j$ has neighboring UAVs that are too far, the utility function will increase, so $\mathbf{t}_j$ is more likely to be chosen.

The aim in the utility function is to maximize d_{tot}. We distinguish two cases where d_{tot} can be too close to zero or equal to it:



(a) $\lambda = 0.2$ (b) $\lambda = 0.4$

(c) $\lambda = 0.6$ (d) $\lambda = 0.8$

Figure 3.8 – Utility function behavior: $\boldsymbol{I}(t_j) = 25$, $n_c = 3$, $d_{tot} = \sum_{k=1,k\neq i}^{n_c}(dmin(\mathbf{P}_k, \mathbf{t}_j))$. The average distance of other UAVs d_{tot} has a minimum value different from zero since n_c is different from zero too.

- There are no neighbors. In this case, there is one UAV in the fleet: $n_C = 1$. The utility function in Eq. 3.3 is used.
- The candidate target $\mathbf{t}_j$ and the neighbors' poses $\mathbf{P}_k$ with $k \in [1..n_c]\backslash i$ are almost confused (too close to each others). In this case $\mathbf{t}_j$ is less likely to be chosen.

Yet, this parameter d_{tot} could have a high value when:

- There is one UAV too far from $\mathbf{t}_j$.
- There are several UAVs with relatively short distance from $\mathbf{t}_j$.

So, in order to avoid these ambiguous situations, the average of d_{tot} is considered by using $\frac{n_c}{d_{tot}}$.

3.6 Goal assignment process

In order to make appropriate UAV-to-target assignment, the utility of reaching each candidate frontier is considered. The goal assignment process is described in Algo. 3.5.

For each UAV_i, the utilities of reaching all the candidate targets are computed. The target $\mathbf{t}_g$ that maximizes the utility is computed and, the assignment $\theta(UAV_i, \mathbf{t}_g)$ is performed. Then, the remaining candidate targets $\mathcal{G} \setminus \mathcal{T}$ are scheduled in order to avoid to select a

Algorithm 3.5: Goal assignment algorithm.

Input: Candidate targets $\mathbf{t}_k \in \mathcal{G}$, $k \in [1..n_g]$ and their respective information gain $\boldsymbol{U}(\mathbf{t}_k)$, poses $\mathbf{P}_i$ of all robots in the considered cluster $\mathcal{C}$.
Output: $\theta(UAV_i, \mathbf{t}_g)$ assignment of UAV$_i$ with target g.

```
T = ∅.
while no goal for UAV_i do
    Compute its corresponding utility of reaching each remaining candidate goal
    U(UAV_i, t_k) with t_k ∈ G \ T.
    t_g = argmax_{t_k ∈ G\T} U(UAV_i, t_k).
    Schedule the information gain of the remaining candidates t_k ∈ G \ T.
    T = T ∪ t_g.
end while
return θ(UAV_i, t_g) assignment.
```

target too close to $\mathbf{t}_g$, in the next iteration. Also, $\mathbf{t}_g$ is removed from $\mathcal{G}$ to prevent from assigning the same target to different robots. This assignment process is performed for the available UAVs in $\mathcal{C}$ in a sequential manner until getting all assigned targets $\mathbf{t}_g \in \mathcal{T}$ with $g \in [1..n_t]$ (See Figure 3.2)

3.6.1 Stop condition

The goal selection process is realized by each cluster/group-*leader* (if $n > n_c$) or the Fleet-*leader* (if $n = n_c$). This assignment aims to, cooperatively, distribute the robots in the environment to explore simultaneously different unknown regions. As long as candidate frontier points are still available, the *leader* continues to assign targets to *explorers* and they attempt to reach their assigned goals. When the *leader* notices that no candidate targets are left, that means that all the environment has been explored successfully and the mission is accomplished. Thus, it has to send back to the *explorers* an acknowledgment to prevent them assuming a communication loss.

3.6.2 Loop rate

The target assignment process is performed at each loop. The frequency of assigning targets impacts the duration and the efficiency of the mission. In a distributed approach, as soon as the UAV reaches its current target, it selects a new one without consulting the others. In a centralized approach, the first UAV to reach its current target has to wait until the others reach their respective targets. This can be a problem as soon as one of them fails or leaves the mission. Another possibility is to begin to assign targets once one UAV reaches its target. But this may generate incomplete tasks. In the proposed strategy, the frequency of assignment or loop rate r is predefined depending on the average of time to reach a target (See Eq. 3.4).

$$r \in \Big[\frac{s}{v_{i,max}}, \frac{s}{v_{i,min}}\Big] \tag{3.4}$$

where s is the maximum sensor range and v_i is the UAV's velocity.

3.6.3 Scheduling the information gain

The information gain of each remaining candidate target $\boldsymbol{I}(\mathbf{t}_k)_{t-1}$ at time $t-1$ with $\mathbf{t}_k \in \mathcal{G} \setminus \{t_g\}$, that belongs to the threshold range $[r_{min}, r_{max}]$, is scheduled at time t depending on its distance w.r.t. the target t_g, using Eq. 3.5. Figure 3.9 represents an example of the function shape used to schedule the information gain. The example shows a Gaussian function with an amplitude that corresponds to the information gain maximum value; a center that corresponds to the target position $(\mathbf{t}_g(x), \mathbf{t}_g(y))$; and σ_x and σ_y that spread the blob in x and y axis, respectively.

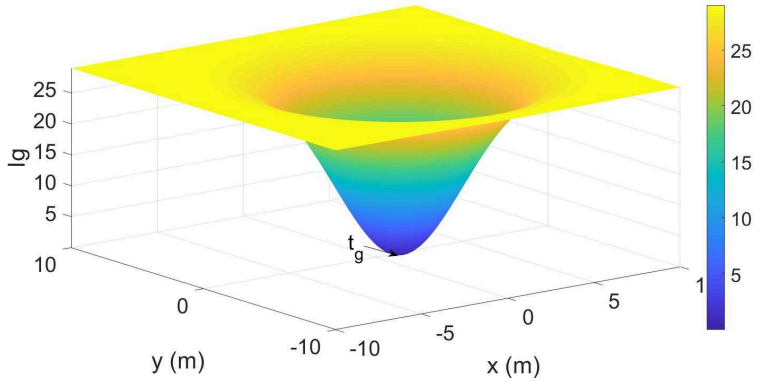


Figure 3.9 – Example of the shape function to schedule the information gain($\boldsymbol{I}(\mathbf{t}_k)_{t-1} = 29$, $(\mathbf{t}_g(x), \mathbf{t}_g(y)) = (1.125, -0.275)$, $\sigma_x = \sigma_y = 3$).

$$\boldsymbol{I}(\mathbf{t}_k)_t = \boldsymbol{I}(\mathbf{t}_k)_{t-1}(1 - \exp(-(\frac{(\mathbf{t}_k(x) - \mathbf{t}_g(x))^2}{2.\sigma_x^2} + \frac{(\mathbf{t}_k(y) - \mathbf{t}_g(y))^2}{2.\sigma_y^2}))), \tag{3.5}$$

where $\mathbf{t}_k(x)$ and $\mathbf{t}_k(y)$ are the remaining candidate target coordinates; $\mathbf{t}_g(x)$ and $\mathbf{t}_g(y)$ are the target coordinates; and σ_x and σ_y are the spreads of the blob. The smaller the distance of the frontier point $\mathbf{t}_k$ w.r.t. the target $\mathbf{t}_g$, the smaller the information gain. When reducing the information gain, the candidate targets are less likely to be chosen and thus, robots ensure a certain distance among their future targets.

3.7 Path planning and control

As explained in Section 5.2 of Chapter 1, UAVs are assumed to navigate in a simplified 2D environment with a fixed z value. Block **6** in Figure 1.6 is responsible for planning a path to the selected target and attempting to reach it. These tasks are ensured by the *move base*[1] package.

[1]Source: `http://wiki.ros.org/move_base`

For the navigation task, each UAV maintains a local and a global planner along with a local and a global costmap, respectively. The costmap is a 2D cell grid $\mathcal{L}$ with additional inflation that consists in propagating cost values out from occupied cells and decreasing them with distance. The global costmap has the size of the UAV's map whereas, the local costmap has a fixed size moving window. Given a starting point – the current pose – and an endpoint – the assigned target – in the global costmap, the global planner produces a plan using a navigation function computed with Dijkstra algorithm [Dijkstra, 1959]. It consists in following the adjacent free cells until reaching the goal. Then taking into account the local costmap, the local planner generates velocity commands for the UAV's mobile base. A recovery rotational behavior is also performed when needed in order to clear the robot's field of view.

The target assigned by the *leader* is ensured to belong to an unknown area using the exploration strategy. The trajectory planning process is performed locally on each robot. And since the UAVs do not exchange their local maps nor fuse them, they are likely to revisit already explored areas while following the planned path. To minimize these overlapped regions during navigation, a priority is given to frontier points $\mathbf{f}_{i,j}$ to be a target for UAV_i over UAV_k with $k \neq i$. This helps the UAV to maintain the same direction during exploration. The *move base* package is a 2D navigation stack. However, to avoid drifting on the z axis, a control command is added to keep a static z altitude (See Figure 3.10).

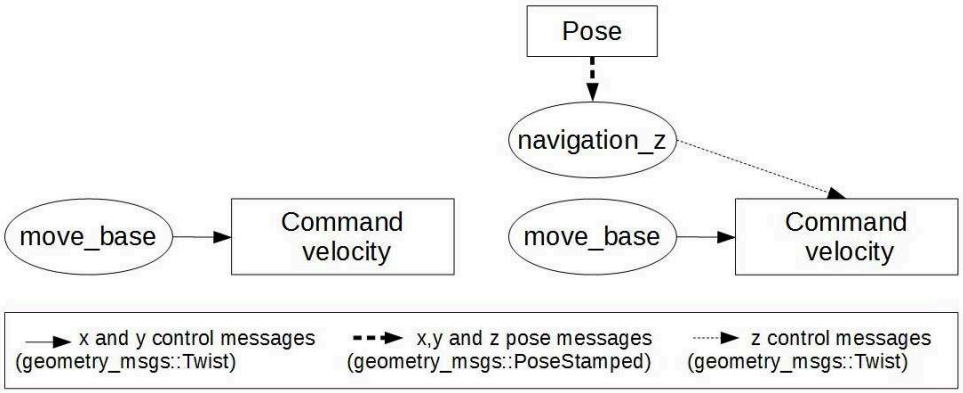


Figure 3.10 – From 2D (left) to 3D (right) navigation stack. The *navigation z* is a ROS process developed to provide a control command on the z axis.

4 Results and discussions

Simulations have been performed to evaluate the proposed exploration strategy. Additional tests while using relative localization have been done to measure the system performances. The simulations are performed using Robot Operating System (ROS) running on a 2.60GHz i7 Linux machine. For the quad-rotor simulation, the AR-drone model[2] equipped with an RGB-D camera in a forward-looking configuration, is used. A bounded unknown environment is generated using *Gazebo* simulator. The number of

[2]Source: `http://wiki.ros.org/ardrone_autonomy`

robots used for evaluation is limited to three, however, the proposed system architecture is not constrained to a fixed number of robots.

4.1 Parameters tuning

For an effective evaluation of the exploration strategy, we run some tests to set the most adequate parameters configuration.

4.1.1 Trade-off parameter λ

The utility function (See Eq. 3.2) used in the exploration strategy can be tuned, using a trade off parameter λ, between fast exploration and filling in details the map.

Figure 3.11 shows different runs while varying this parameter. By increasing λ, the information gained when reaching the goal is favored over the distance and thus, the cost to it, and vice versa. So, when λ is small, the traveled distance is small and so the exploration time. Though, some times during the mission, high values of λ are noticed to reach higher exploration rate than smaller ones.

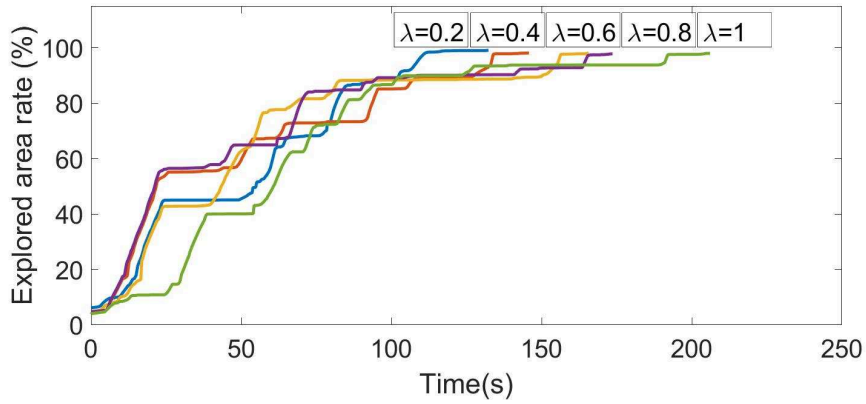


Figure 3.11 – The impact of varying the trade off parameter λ over exploration time.

4.1.2 Loop rate r

The frequency or loop rate r of target assignment may also affect exploration time performance. The values of r vary to take into account the robot velocity $\mathbf{v}_i$ and the sensor's maximum range s. The impact of varying the loop rate is evaluated in Figure 3.12.

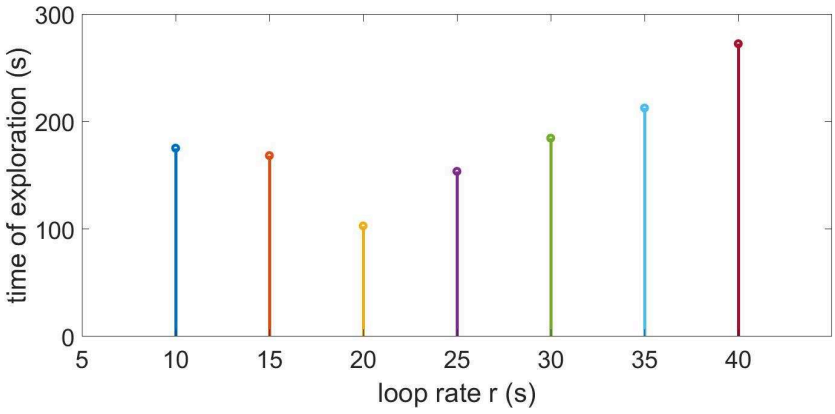


Figure 3.12 – Exploration time while varying the loop rate r.

Given a robot velocity $\mathbf{v}_i = \{0.1, 0.3\}\ m.s^{-1}$ and a maximum sensor range $s = 4\ m$, the loop rate variates in $r \in [10, 40]$. This parameter should not be too small to allow the robot to reach its target; nor too big to prevent long waiting times for the next goal assignment.

4.1.3 Common parameters

Depending on results in Figure 3.11 and Figure 3.12, respectively, λ is set to 0.2 and r to $20s$ in order to maximize the explored area rate while minimizing the mission time. The simulation parameters are summarized in Table 3.1.

Table 3.1 – Common parameters.

Parameter	**Value**
RGB-D horizental FoV	$\pi/3$
Trade-off parameter λ	0.2
RGB-D maximum range s (m)	4
Min distance among frontiers d (m)	0.3
Occupancy grid resolution (m)	0.05
Range to schedule the Ig $[\sigma_x, \sigma_y]$ (m)	$[3, 3]$
Loop rate l (s)	20
Environment dimension (m^2)	8×8
Linear velocity v_i $(m.s^{-1})$	$[0.1, 0.3]$
Angular velocity ω_i $(rad.s^{-1})$	$[0.1, 0.3]$

4.2 Exploration strategy performances

The proposed exploration strategy has been evaluated in terms of distribution of the robots in the environment, overlap rate, exploration time, and total traveled distance by each robot.

4.2.1 Maps evolution during the mission

While reaching their respective assigned goals, each robot is in charge of creating a detailed grid map of the visited area in order to get a global map of the environment. Figure 3.13 illustrates the growth of the reconstructed 3D occupancy grid map at different times during the mission. Note that to reach 99% of coverage, a lot of time is spent.

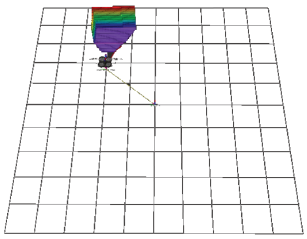
Figure 3.14 shows the evolution of the respective projected 2D local grid map of two robots during a cooperative exploration mission. The global projected 2D grid map is also created and represented for evaluation (the occupancy grid matching process introduced in Algorithm 2.1 in Chapter 2 is used). The robots' initial positions are (1,0,0) for UAV_1 and (1,-3,0) for UAV_2. Despite a relatively close initial position, the proposed strategy effectively spreads the robots so that UAV_1 is in charge of the left side of the environment and UAV_2 of the right one.

4.2.2 Frontier points evolution during the mission

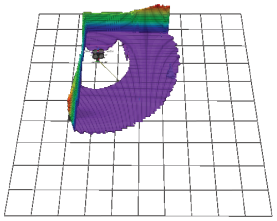
The target is chosen from the candidate frontier points that define the edges of an environment not previously explored. These candidates are selected from the final frontier points of each UAV in the fleet (See Figure 3.15). During the exploration mission, the local map size increases, which leads to an increasing number of local frontier points. At the beginning of the exploration, the number of candidate frontier points increases, but as soon as the exploration evolves in time, their number decreases. At the end of the mission, when all the environment is explored, no candidate frontier points should be left.

4.2.3 UAVs' trajectories during the mission

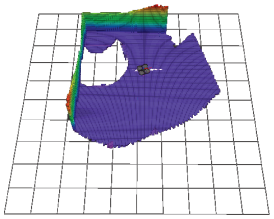
Figure 3.16 shows the explored map with the trajectories using one, two and three UAVs. The UAVs try to explore the full environment while avoiding already explored areas. In a cooperative way, each UAV is in charge of visiting an area by reaching a target that belongs to a non-explored environment. These targets are assigned by the *Leader* which, even if the initial poses of the UAVs are relatively close, effectively spreads them into unknown areas. The global map is composed of the superposition of all UAVs' local maps.



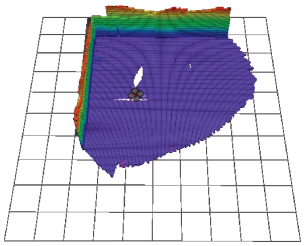
(a) 3% rate (5s).



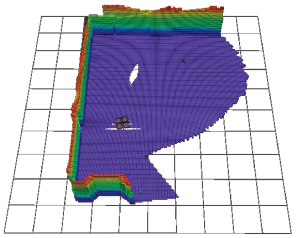
(b) 20% rate (28s).



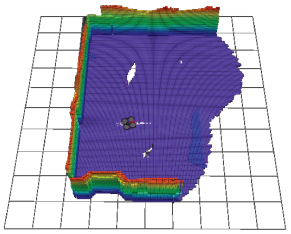
(c) 31% rate (50s).



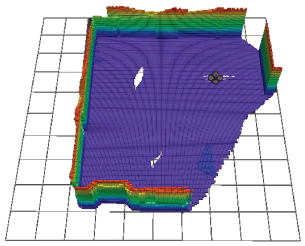
(d) 40% rate (69s).



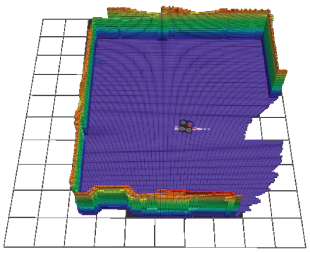
(e) 50% rate (75s).



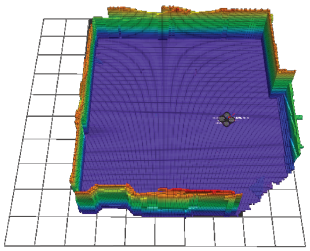
(f) 60% rate (83s).



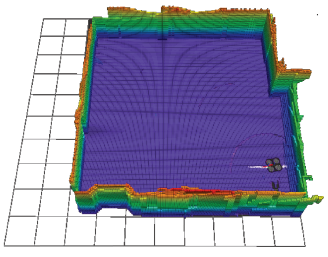
(g) 70% rate (98s).



(h) 81% rate (127s).



(i) 90% rate (133s).



(j) 99% rate (202s).

Figure 3.13 – Ratio of explored space during time for one UAV.

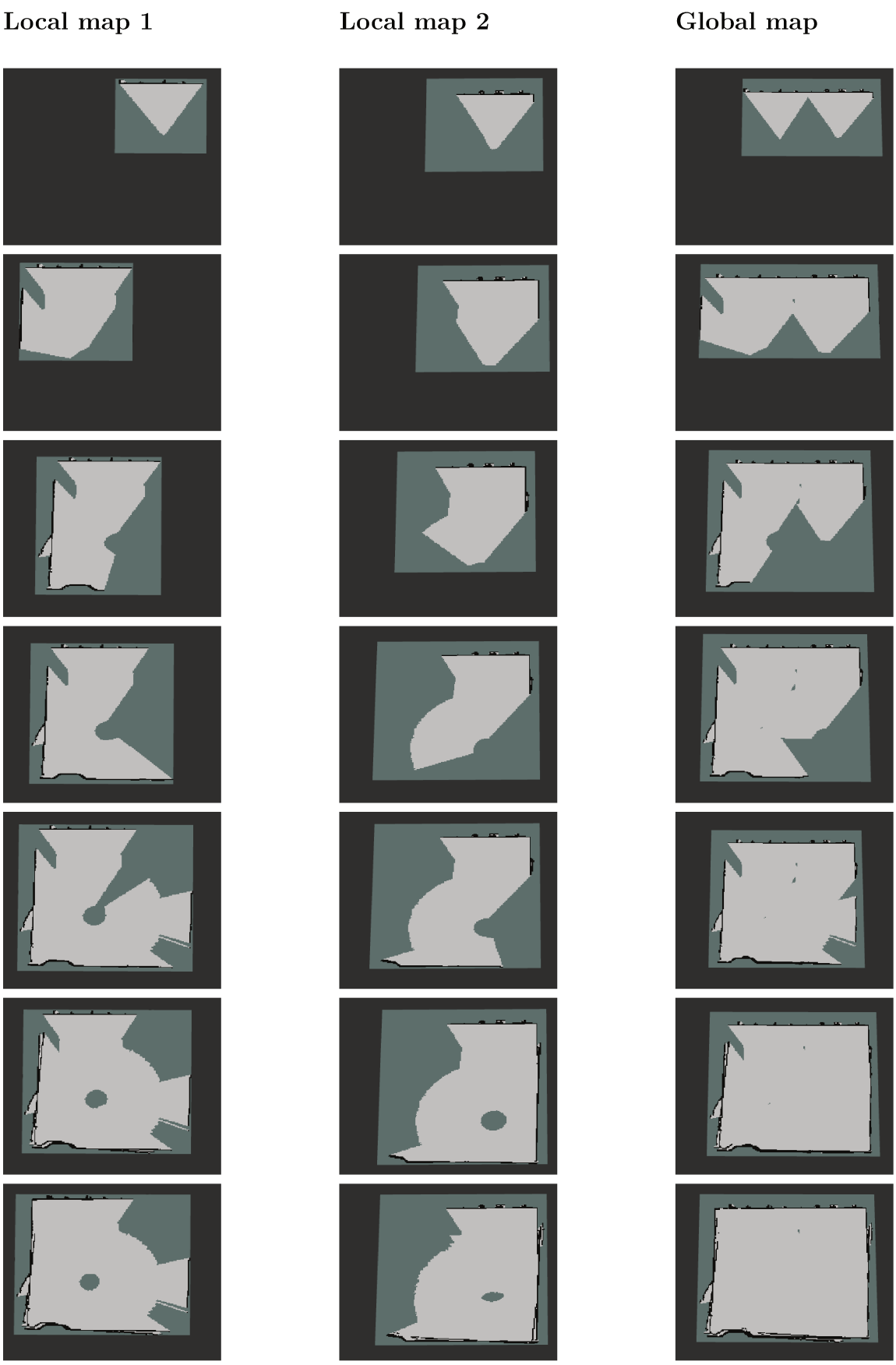
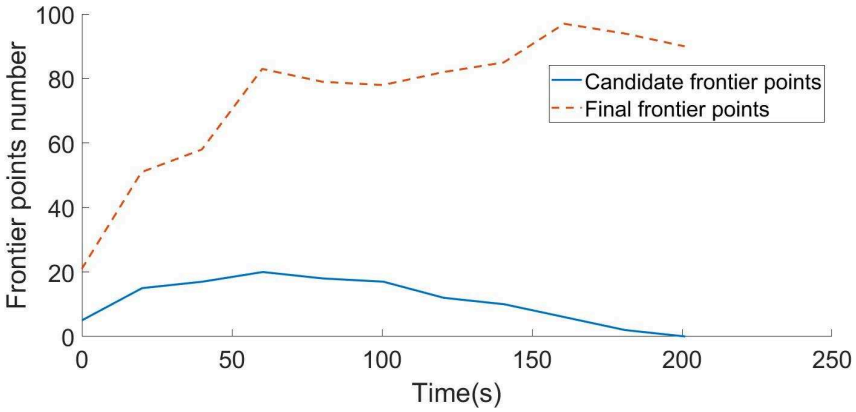
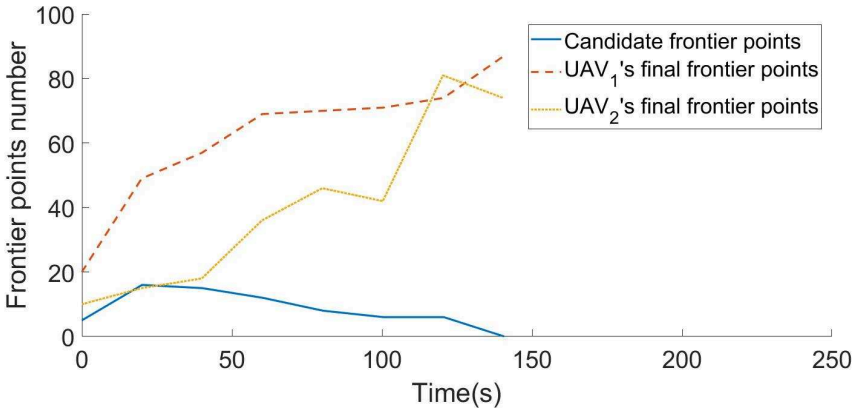


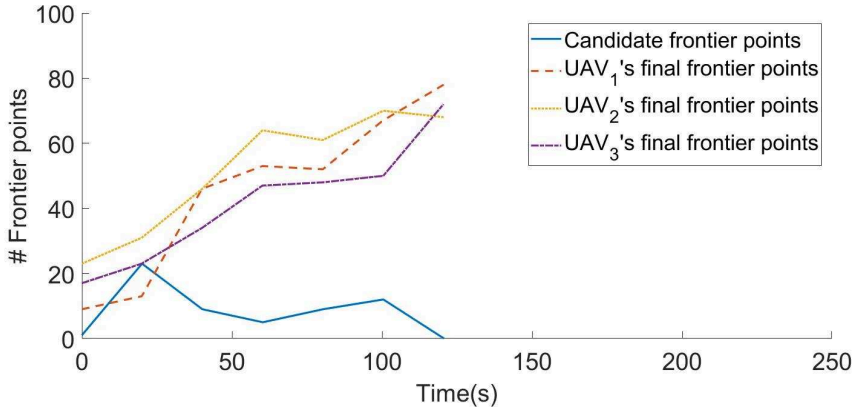
Figure 3.14 – Coordinated exploration using two robots. Columns 1, 2 and 3 show the evolution of the local map of the UAV_1, the UAV_2 and the global map over time, respectively.



(a) One UAV.



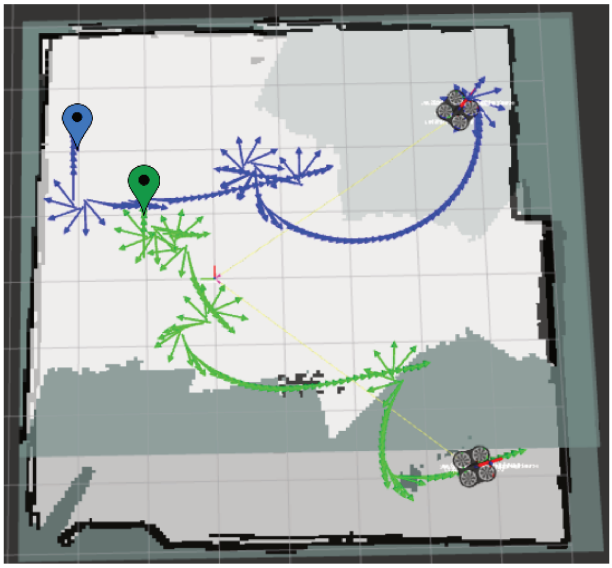
(b) Two cooperative UAVs.



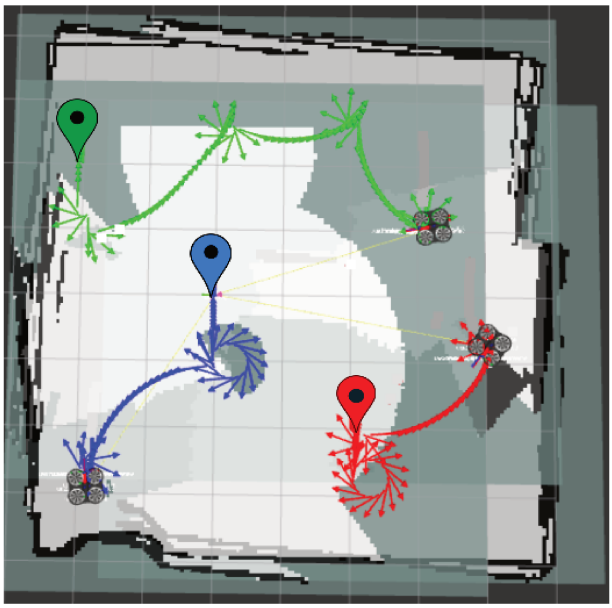
(c) Three cooperative UAVs.

Figure 3.15 – The evolution of candidate and final frontier points numbers during cooperative exploration.

(a) One UAV.



(b) Two cooperative UAVs.



(c) Three cooperative UAVs.

Figure 3.16 – Projected 2D map in a coordinated exploration with a team of one, two and three UAVs. Green, blue and red markers and arrows define, respectively, the initial position and the trajectory of UAV1, UAV2 and UAV3.

4.2.4 Goal assignment evaluation: Distribution of the robots in the environment

The goal assignment process is performed according to the algorithm described in Section 3.6. Nevertheless, after assigning a target to the first robot in the list, the same target or another one close to it may be assigned to the second robot in the list. To overcome these issues, the information gains of the remaining candidate targets are scheduled. This allows to discard an already assigned target and keep a certain distance between the new target and the previous one assigned.

Suppose that a target is assigned to the first robot in the cluster list. Figure 3.17 shows the goal selected for the second robot when a sequential assignment is performed:

- Without further frontier points processing (See Figure 3.17b). Consequently, the same target is assigned to two different robots.
- While removing the assigned target from the remaining candidate frontier points (See Figure 3.17c). Consequently, the second target is relatively close to the first one assigned.
- While scheduling the information gain after each target assignment (See Figure 3.17d). Consequently, the assigned targets are spaced out into the environment. The information gain is scheduled following Eq. 3.5. The information gain value increases with distance to the candidate target $\mathbf{t}_g$.

The goal assignment process may sometimes be not optimal since it depends on the robots' order in the list. For example, suppose that robots UAV_i and UAV_j have the same best target assignment $\mathbf{t}_k$ such that it offers the maximum utility over candidate frontier points where Eq. 3.6 and Eq. 3.7 are verified.

$$\mathbf{t}_k = argmax_{\mathbf{t}_m} \boldsymbol{U}(\text{UAV}_i, \mathbf{t}_m) \tag{3.6}$$

with $\mathbf{t}_m \in \mathcal{G}$

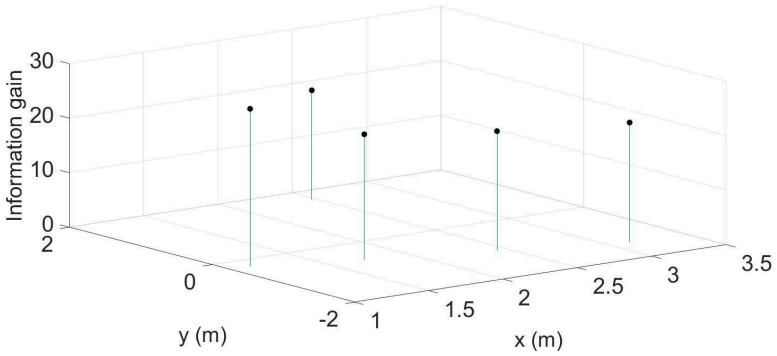
$$\mathbf{t}_k = argmax_{\mathbf{t}_n} \boldsymbol{U}(\text{UAV}_j, \mathbf{t}_n) \tag{3.7}$$

with $\mathbf{t}_n \in \mathcal{G}$.

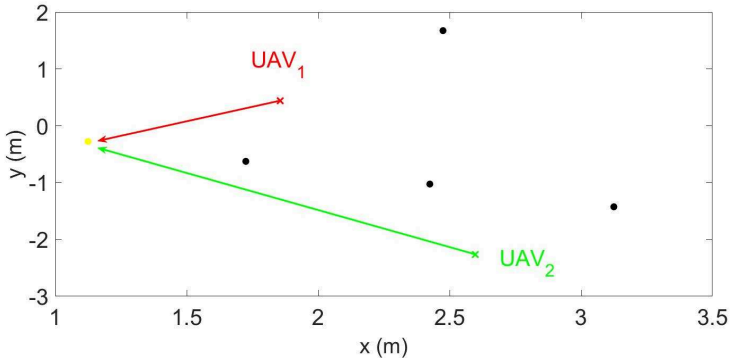
UAV_i have another candidate frontier point $\mathbf{t}_l$ with:

$$\boldsymbol{U}(\text{UAV}_i, \mathbf{t}_k) > \boldsymbol{U}(\text{UAV}_i, \mathbf{t}_l) > \boldsymbol{U}(\text{UAV}_j, \mathbf{t}_k) \tag{3.8}$$

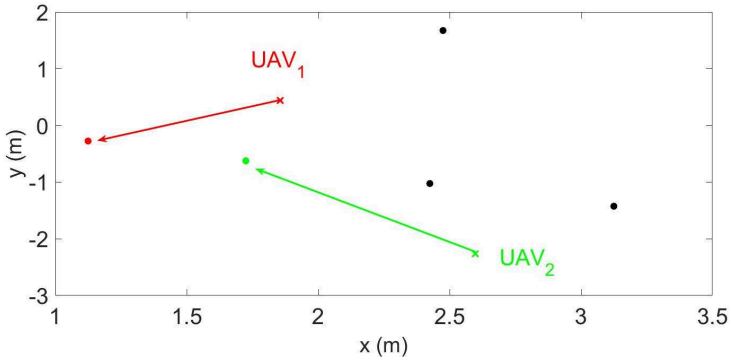
So the optimal solution would be to assign $\mathbf{t}_l$ to UAV_i and $\mathbf{t}_k$ to UAV_j. But, if UAV_i is the first in the list, $\mathbf{t}_k$ is assigned to it and another candidate frontier point with less utility than $\mathbf{t}_k$, is assigned to UAV_j. Thus, the solution with sequential goal assignment is not always optimal.



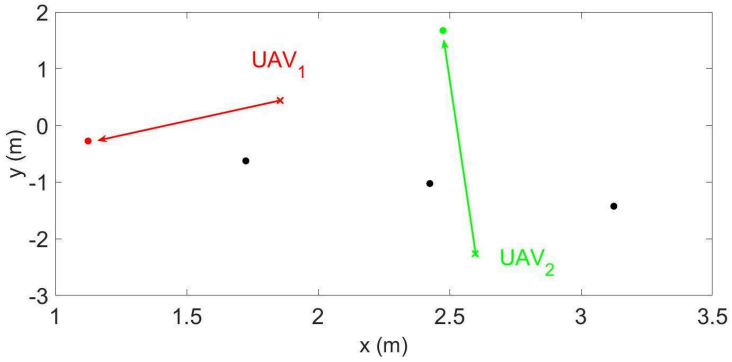
(a) Candidate targets.



(b) Case 1.



(c) Case 2.



(d) Case 3.

Figure 3.17 – Goal assignment: After assigning a target to UAV_1, a target is assigned in a sequential manner to UAV_2. (a) represents the candidate frontier points with their respective information gain. (b), (c), and (d) represent, respectively, the targets assignment when: No further process is performed for the remaining candidate targets; UAV_1's target is removed from the remaining candidate targets and; the information gain of the remained candidate targets are scheduled.

To overcome this problem, all the numbers of possible combination $\frac{n_g!}{n_g!\,(n_g-n_c)!}$ with n_g the number of candidate targets and n_c the number of robots, need to be considered. This increases considerably the computation time by increasing the number of robots. Therefore, in the proposed algorithm, sequential assignment is favored over computing all possible permutations.

4.2.5 Explored space rate evaluation

This evaluation aims at quantifying the amount of explored spaces during the mission. Figure 3.18 shows the explored space rate performed by each UAV in the fleet. The more UAVs in the environment, the less the exploration rate demanded by each one. A robot has no need to continue exploring an area if it has already been explored by another one. Thus, the mission time is considerably reduced.

4.2.6 Overlap rate evaluation

The use of an effective goal assignment process should limit the generated overlap. In Figure 3.19, the time evolution of overlap is evaluated using two cooperative robots. The overlap undergoes a significant increase at the end of the exploration to reach 33%. This is explained by the closeness of the local maps at the end of the mission to precisely fill the global grid map.

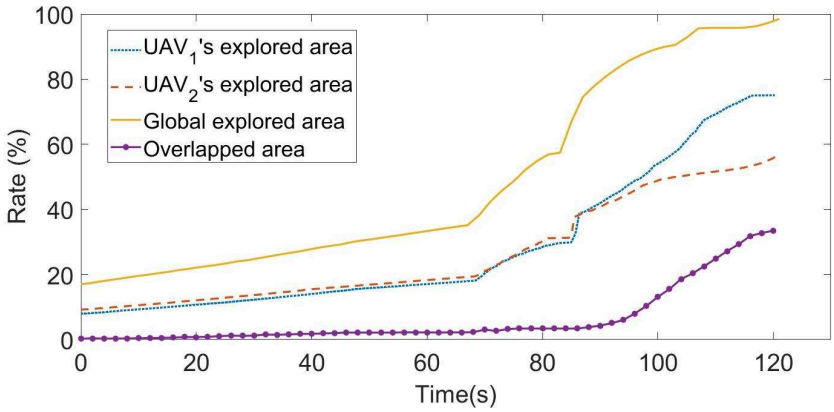
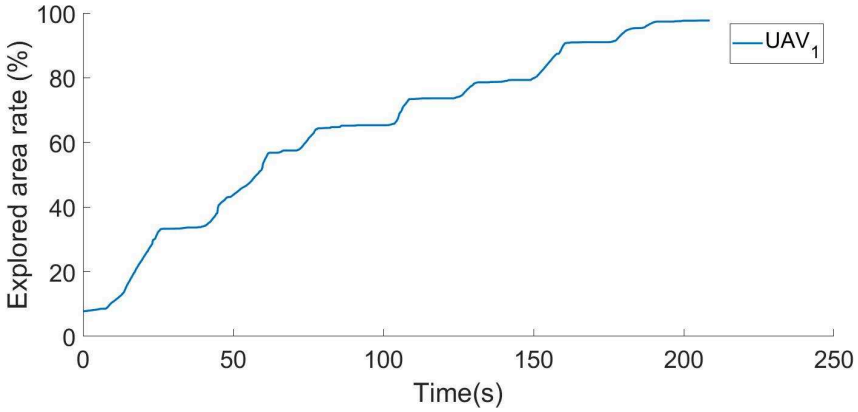


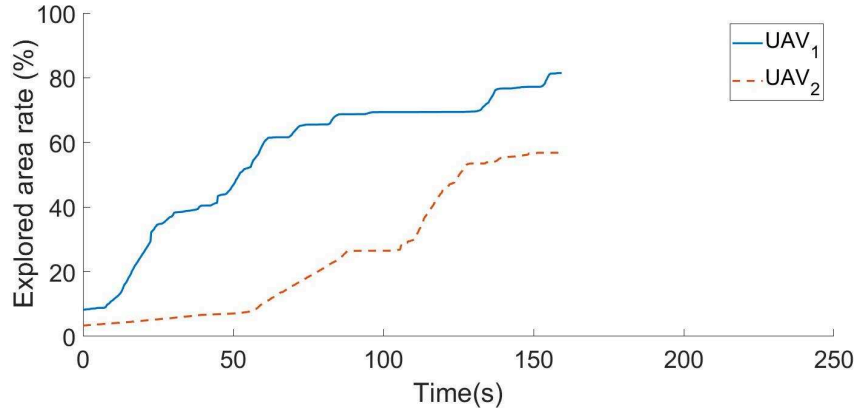
Figure 3.19 – Explored and overlapped area rate using two cooperative UAVs.

4.2.7 Traveled distance evaluation

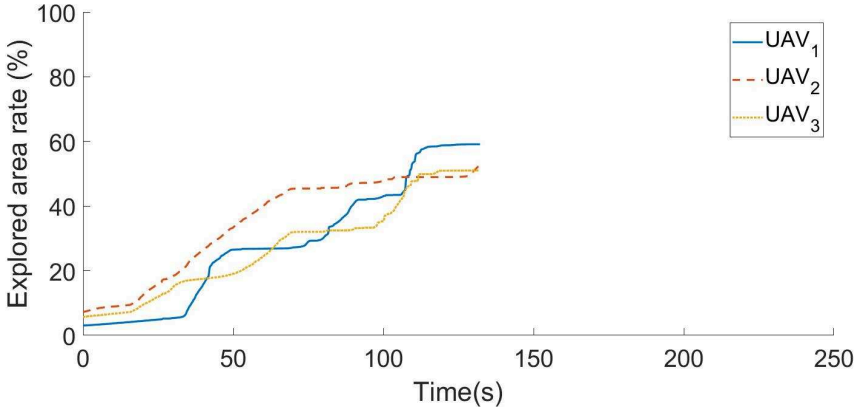
To effectively evaluate the exploration strategy performance in terms of distance traveled by each UAV, different runs with one, two and three UAVs have been conducted where explored area rate reaches almost 99%. Figure 3.20 shows the distance traveled by each UAV in the fleet during the mission.



(a) One UAV.

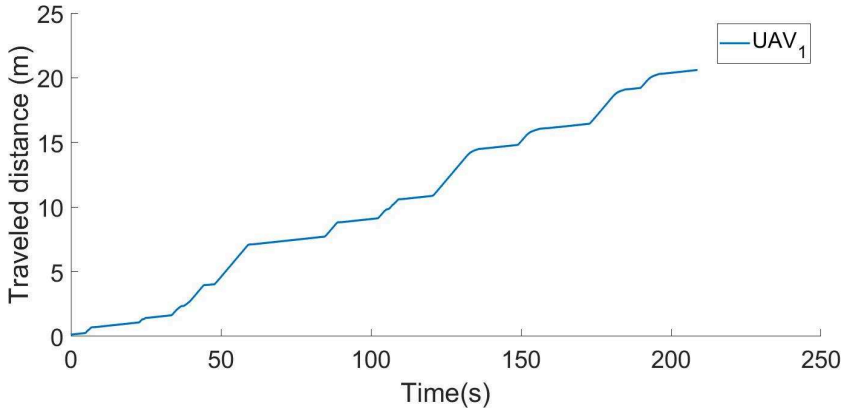


(b) Two cooperative UAVs.

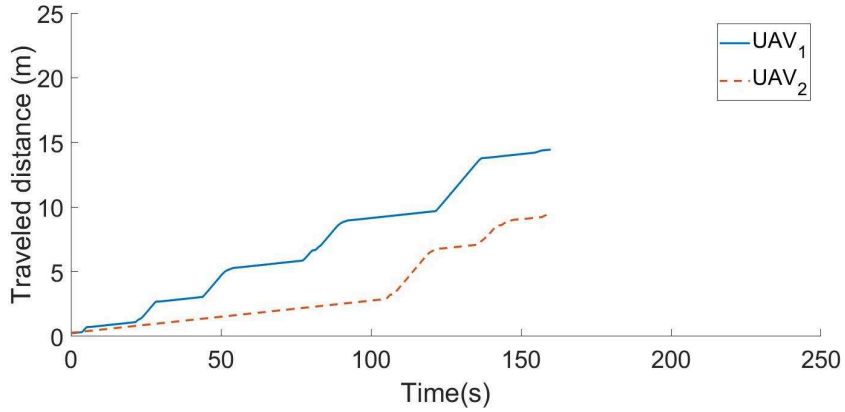


(c) Three cooperative UAVs.

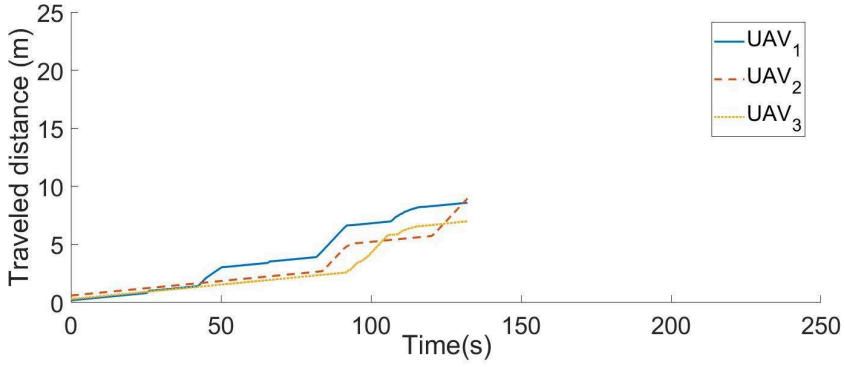
Figure 3.18 – Explored space rate with one, two and three UAVs.



(a) One UAV.



(b) Two cooperative UAVs.



(c) Three cooperative UAVs.

Figure 3.20 – Traveled distance by each UAV with one, two and three cooperative UAVs.

The distances traveled by each UAV using one, two and three UAVs in the fleet are compared in Figure 3.21. This distance decreases with the number of UAVs. The average distance traveled by each UAV is reduced by 55% for 2 UAVs and by 62% for 3 UAVs. The error of the traveled distance is slightly reduced from one to two and three UAVs.

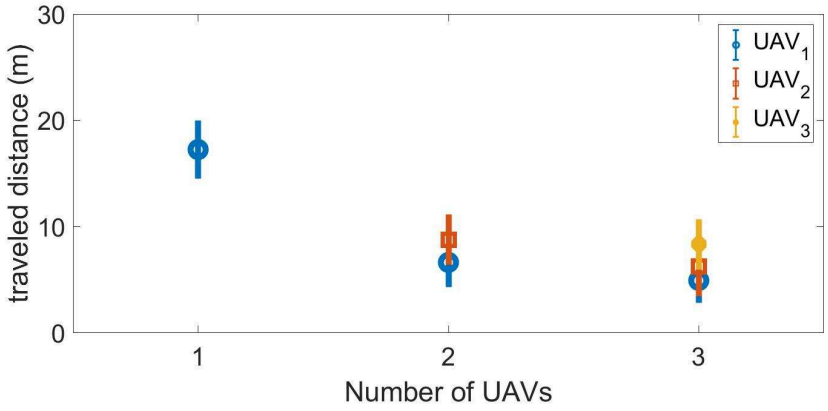


Figure 3.21 – Traveled distance evaluation.

4.2.8 Exploration time evaluation

For the exploration time evaluation, different runs have been performed using 1, 2 and 3 UAVs. Figure 3.22 shows that the average of exploration time decreases when the number of robots in the fleet increases. The computed error decreases as well. The time is reduced by 25% for 2 UAVs and by 30% for 3 UAVs. The exploration time and distance are not divided by 2 or 3 when multiplying by 2 or 3 the number of robots, respectively. During these simulations, the robots' initial positions are: (1,0,0) for one UAV; (1,0,0) and (1,-3,0) for two UAVs; and (1,1,0), (1,-1,0) and (1,-3,0) for three UAVs.

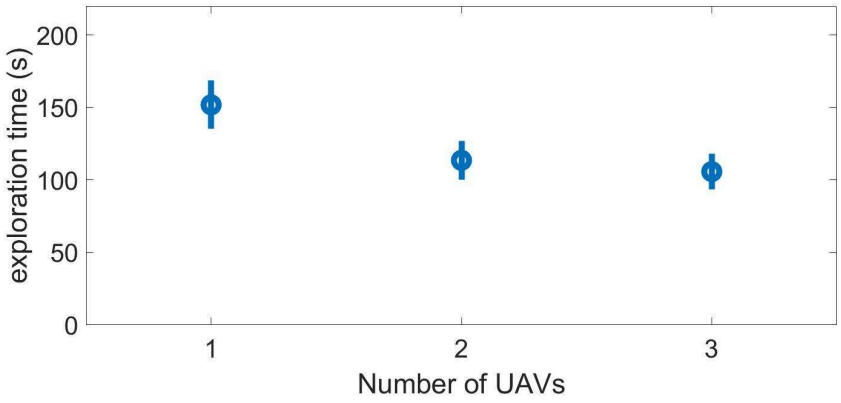


Figure 3.22 – Average exploration time.

The results presented in Section4.2 were evaluated without a relative localization. So, for a more challenging realistic scenario, runs with relative localization algorithm have been performed to evaluate system performances using SLAM.

4.3 Exploration mission using relative localization algorithm

Toward a more realistic scenario, the ORB-SLAM2 approach (introduced in Section 5 of Chapter 2) has been implemented to perform relative localization (See Figure 3.23).

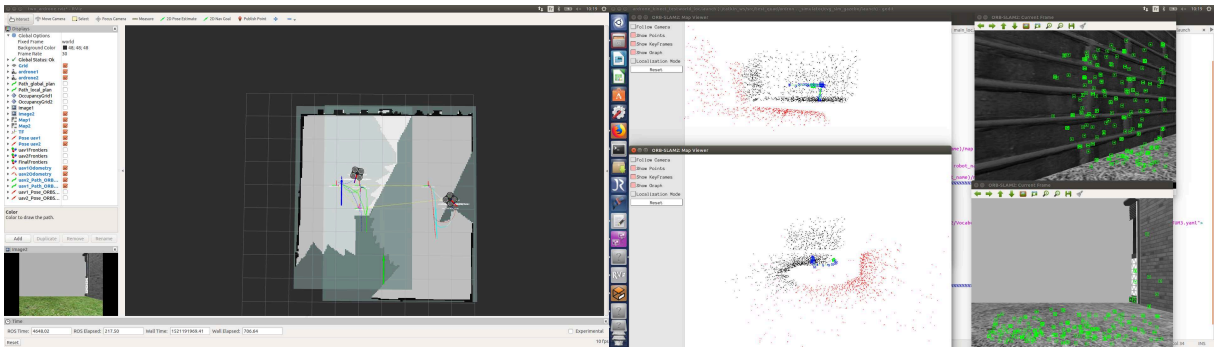


Figure 3.23 – Two UAVs navigation using ORB-SLAM2. Rviz image (left) shows, for each UAV, the constructed 2D occupancy grid map, its estimated trajectory and the corresponding ground truth. The point cloud (middle) represent the sparse reconstruction of the environment made by each UAV. And, the green markers (right) represent the features computed by each UAV to perform localization.

Each robot performs SLAM where it constructs its map in its local reference frame ${}^{W}F_i$, and estimates its relative pose $\mathbf{p}_i$ within it. Then, the fleet performs cooperative exploration using some specific information exchanged among UAVs. But, these information have to be in a common reference frame. Therefore, information such as the pose $\mathbf{p}_i$ and the frontier points $\mathbf{f}_{i,j}$ are necessarily transformed into the global reference frame ${}^{0}W$ before being exchanged. Hence, the *leader* makes all the needed computation and sends back to the *explorers* the targets in ${}^{0}W$. When a robot receives its assigned goal, it transforms it into ${}^{W}F_i$ to plan a path to it.

To perform a transformation from local reference ${}^{W}F_i$ to global one ${}^{0}W$, the robot has to know – at least – its initial pose w.r.t. ${}^{0}W$. As explained in Section 5.2 of Chapter 1, the global reference frame of the environment is initialized such that it coincides with the local reference frame of the first group-*leader* in the fleet which is UAV_1 in the considered example of Equation 3.9.

$$ {}^{0}W \equiv {}^{W}F_1, \tag{3.9} $$

Then, by detecting this robot using tags mounted on it, the other robots are able to estimate their respective transform to it ${}^{F_j}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}_{F_1}$, $j \in [2..n_c]$. For simulation evaluations, the information of transform – computed while detecting the tag – are assumed to be known. Figure 3.24 shows the exploration rate evolution during the exploration mission while using ORB-SLAM2 as the relative localization approach. The mission time using ORB-SLAM2 is reduced by 43% for 2 UAVs instead of 1 UAV.

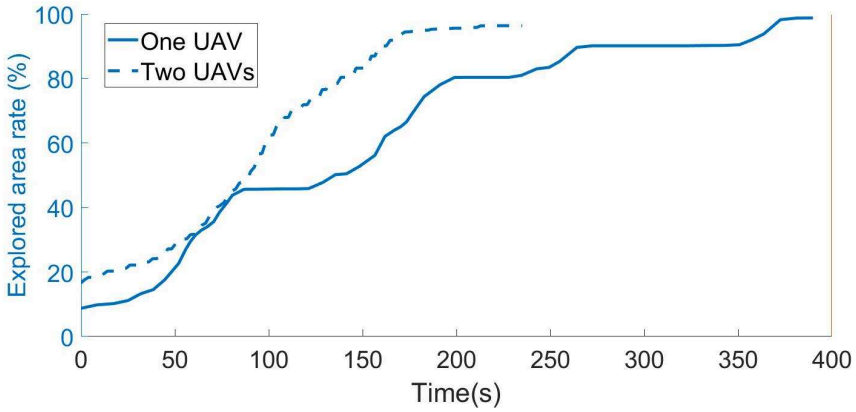


Figure 3.24 – Explored area rate evolution during exploration mission with one and two UAVs while performing ORB-SLAM2 by each UAV.

The exploration time when using a relative localization (See Figure 3.24) is relatively important compared to the exploration without SLAM (See Figure 3.18) since the velocity has been considerably reduced.

5 Conclusion

In this chapter, we presented a state of the art of multi-robot exploration mission including the strategy used to assign a robot to a target and an utility function adopted to estimate the interest of reaching it. Then, we introduced an exploration strategy based on the group-*leader* decision making. The robot-to-target assignment is performed using a novel utility function. This function makes a trade-off between fast exploration and getting a detailed grid map, and also takes into account the distance of each robot in the group from the unexplored set of targets. Also, we propose to schedule the information gain in order to efficiently spread the UAVs into the environment. Moreover, the strategy adopted exchanges the frontier points instead of a whole copy of the local map.

Results show that the proposed cooperative exploration strategy minimizes the global exploration time by 25% for 2 UAVs and by 30% for 3 UAVs, while minimizing the average traveled distance by each UAV by 55% for 2 UAVs and by 62% for 3 UAVs. Furthermore, the strategy was evaluated using a relative localization algorithm where the exploration time was reduced by 43% for 2 UAVs instead of 1 UAV.

Inter-robot communication

Contents

1 Introduction

A critical subject in multi-robot systems is the communication among robots. This feature is used for coordination and to share specific information. A fleet can be deployed in several missions within relatively difficult and hostile areas. But, these environments may have no pre-existing network infrastructure, in addition to non ideal communication links.

This chapter highlights one of the most challenging points in MRS, which is the inter-robot communication. This problem can be addressed from different perspectives; but, we have chosen to study two sub-problems, which are: network typology, and network topology and strategy for MRS robustness.

2 Network classification

The network is used to link between different entities and to establish possible communication between them. Several network types exist and can be classified in different ways.

2.1 Infrastructure versus infrastructureless mode

To manage network infrastructure, the existed modes can be classified into two major categories [Ärlker Bekmezci et al., 2013, Hayat et al., 2015]. The first kind is the

infrastructure mode, which can be called Access Point (AP) mode, and the second kind is the infrastructureless mode named Ad Hoc mode (See Figure 4.1). In the AP mode, the

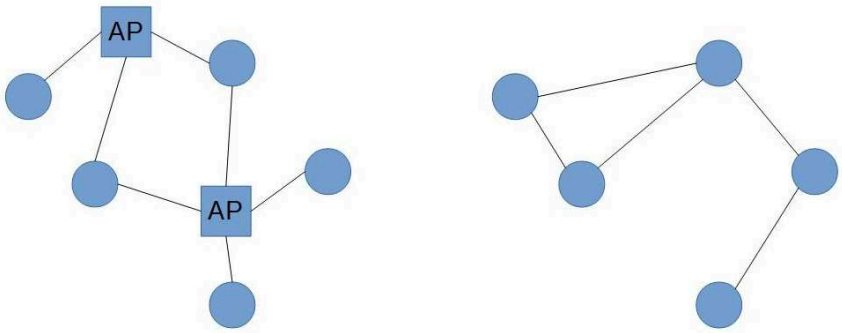


Figure 4.1 – Infrastructure (left) and infrastructureless (right) mode.

UAV-to-UAV communication is realized through the infrastructure (AP, routers, etc.), unlike the Ad Hoc mode in which nodes communicate directly with each other. The Ad Hoc mode is also called peer to peer mode. The Ad Hoc network can grow into another peer to peer mode called mesh network by enabling multi-hop capability. Indeed, the Ad-Hoc networks do not have any inherent capability for multi-hop. Nodes communicate with each other when they are within one another's communication range. Whereas, in mesh networks, nodes are able to communicate directly or through one or more intermediate nodes.

Table 4.1 – Infrastructure versus Infrastructureless mode.

	Infrastructure mode	**Infrastructureless mode**
Advantages	Reliability of communication.	Direct connection to each other, easy to set up, robust to node failure, allow expansion and modification in network topology.
Disadvantages	Expensive, complicated hardware, range restriction.	Redundancy in network connection, difficult maintenance.
Standard	802.11	802.11, 802.15, 802.16

Originally, robots have to perform their tasks in environments without network infrastructure. In fact, they exchange information when they act as routers and as APs. So, the MRS system can be seen as a mobile Ad Hoc network in a communication point of view.

2.2 Ad Hoc network classification

Ad Hoc Network is the commonly used network to manage multi-robot communication. With this network, each robot can move freely and forward packets to and from each other depending on the mode of distribution of data [Bouachir, 2014]. This network is characterized by sophisticated quality of service in which routing protocol determines the

optimal path of the information taking into account the frequent change of the topology. Added to that, it is cheaper to realize this kind of network instead of others (satellite, cellular, etc.).

The Ad Hoc network can be classified into three main categories which are Mobile Ah doc NETwork (MANET), Vehicular Ah doc NETwork (VANET), and Flying Ad hoc NETworks (FANET). For multi-UAV system, generally, the FANET model is adopted. Table 4.2 details each category's characteristics [Ärlker Bekmezci et al., 2013, Maistrenko et al., 2016] and also analyses our model's expectations.

Table 4.2 – Comparison between MANET, VANET, FANET and our model's network expectation.

Characteristics	**MANET**	**VANET**	**FANET**	**Our model's expectation**
Mobility	Men in certain terrain	Vehicle in highway	Plane in 3D plan	3D
Mobility degree	+	+	++	++
Mobility model	Random way points with random direction and speed	Highly predictable in roads	Not predetermined (random UAV movement model, pheromone based model)	Predictable
Topology changes	+	+	++	++
Distance between nodes	+	+	++	+
Node density	++	++	+	+
Radio propagation model	Rare presence of line of sight	Rare presence of line of sight	Frequent presence of line of sight	Always presence of line sight
Power consumption and life time computational power	+	+	++	++
Standard	IEEE 802.11, 802.15, 802.15.4, 802.16 and 802.20	IEEE 802.11p	IEEE 802.11	?

The comparison shows that FANET is the closest network to our model's expectations. Hence, among the available standards for this kind of network, IEEE 802.11b and IEEE 802.11g provide high data rates and are typically used for multi-robot systems.

2.3 Network topology classification

The network topology defines the starting points, the endpoints and the paths of the information in the network. Given target points, the topology defines all possible paths to reach them.

From a software point of view, the network can be mainly classified in three topology (See Figure 4.2[1]):

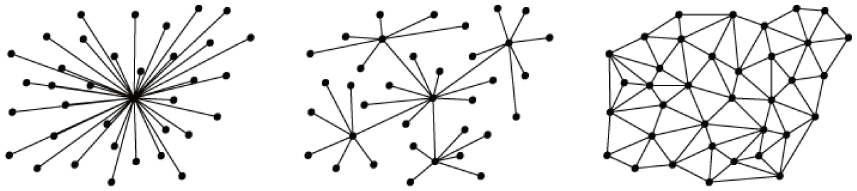


Figure 4.2 – Centralized (left), decentralized (middle) and distributed (right) networks

- Centralized network: It is the most used topology [Morgenthaler et al., 2012, Forster et al., 2013b]. All nodes – also called stations – are connected to each other via a centralized hub. That is when a node needs to send an information to anther one, the data necessarily pass through the central node to be routed to its destination. If the central node is out of service or unreachable, no data can be transmitted which makes this topology vulnerable to node failure.

- Decentralized network: This topology is considered as a set of centralized networks connected together [Fox et al., 2003, Brand et al., 2014]. One central hub is employed for every little group of nodes to send and receive the information. If a hub fails, nodes which are directly relied to it become inaccessible.

- Distributed network: This topology contains nodes that are connected to each others. Unlike the centralized topology, several paths are possible to reach one destination. Also, nodes are not affected by one node's failure. This allows to increase the survivability and to decrease the vulnerability of the network. The distributed topology is attractive to ensure reliable network [Cunningham et al., 2010]. Nevertheless, it is quite difficult to reach and to realize in reality, and the simulation are not fully distributed [Waharte et al., 2009, Cameron and Trigoni, 2009, Scherer et al., 2015]. This is because all processing and decision making have to be on-board the robot which requires an important memory and a heavy computation.

3 Network typology

The typology of the network represents its type and, accordingly, the characteristics introduced.

[1]Source: From user 1983 on Wikipedia. Licensed CC BY-SA

3.1 Related work

MRS communication is a critical problem to tackle in the robotics community. In the last decade, the great progress in wireless technologies and MRS has created a growing interest for inter-robot communication. Hence, different methods have been proposed.

To face disaster scenarios such as fire in an industrial warehouse, authors in [Witkowski et al., 2008] propose to use Wireless LAN, Bluetooth and ZigBee to form an Ad-Hoc network. Some robots of the fleet form the network infrastructure to support communication.

Authors in [Morgenthaler et al., 2012] use two network standards to build a multi-UAV system. A IEEE 802.11s wireless mesh network is build using UAVs carrying mesh nodes directly connected to the flight electronics. Each of these mesh nodes acts as an AP in order to form an IEEE 802.11g.

A methodology that profiles the wireless links for Linux-based networked aerial vehicles is presented in [Kuschnig et al., 2012]. The approach includes two options to collect link quality information and to monitor the 802.11 wireless interface.

An evaluation of the standard 802.11a used between an UAV and an AP is performed in [Kuschnig et al., 2012]. Results show that the Received Signal Strength (RSS) for the downlink decreases with the distance to the AP but remains at an acceptable level as well as the throughput.

Knowing that the visible light spectrum is much wider than the Radio Frequency spectrum, it seems interesting to uptake Visible Light Communication (VLC) based on Light Emitting Diodes (LEDs). IEEE has developed the 802.15.7 standard for short range communication using visible light. Authors in [Wang et al., 2014] introduce a bidirectional communication using Open VLC. This solution hardware needs a Beagle Bone Black (BBB) board and a font transceiver that employs a single LED to both transmit and receive. The solution is implemented on Linux driver that communicates directly with the LED front-end and the Linux networking stack. The main idea to reuse the same LED for both Transmitting (TX mode) and Receiving Light signal (RX mode) is that when a node is transmitting data, the other node can expect the LOW symbol of the bit 0 and makes use of this time to switch his mode from RX to TX and transmit data.

The Zigbee – based on IEEE 802.15.4 – is a protocol of high level which is particularly useful for short communication range and low consumption. Several UAVs' application use this network [Asadpour et al., 2014]. Communication can be made under three frequency bands depending on the chipset type: 868MHz for a bit rate of 20 kbps, 915MHz for 40 kbps and 2.4GHz for 250 kbps.

Authors in [Vidal et al., 2015] adopt different sized groups composed of tactical UAVs and UAVs moving in delimited geographical areas. Virtualization techniques are used to adapt the upgrade of any function, which allows the flexibility in heterogeneous services.

In a search and rescue mission, for example, authors in [Scherer et al., 2015] propose to do real-time streaming between UAVs over a large distance. The mission is divided into five

principle phases: The pre-planning to define the best paths in the search area allowing to reduce the time of the mission then searching by just following the predefined waypoints; the detection of the target and sending the new plans to the others, followed by the repositioning by setting a multi hop link to evaluate the situation by the base station and finally, the streaming of the video.

The multi-UAV, moving in a three dimensional space, need sophisticated transmission systems to ensure flexibility. In [Scherer et al., 2015], an omni-directional isotropic antenna is fixed on the base station and on all the deployed heterogeneous UAVs. For data exchange, the standard IEEE 802.11s mesh technology is used.

Authors in [Hayat et al., 2015] present a performances comparison between the standard IEEE 802.11n and the IEEE 802.11ac in a multi hop network. Both networks were experimented in indoor and outdoor environment, and in two mode, the access point mode (AP)/infrastructure mode and the mesh mode regarding throughput and fairness. In indoor experiments, for an infrastructure mode, the 802.11ac shows improved performances in both TCP and UDP throughput, and packet loss for UDP traffic. In outdoor experiments, for infrastructure mode, the throughput of 802.11n is three time higher than that of 802.11a however, the link quality drops more steeply. For mesh mode, 802.11n achieves higher throughput at close range but drops faster as soon as date rate gets higher and the range longer. The recorded throughput for mesh network is lower than infrastructure mode due to the longer inter-packet transmission times.

To face issues of maintenance connectivity, collision avoidance, robustness to failure and area coverage improvement, authors in [Ghedini et al., 2018] propose a novel model that provides more efficient network topologies.

In [Harms et al., 2018], a new communication layer is proposed to deal with networks that requires a high bandwidth. For that, some mechanism are used to buffer messages, to compress data or to react to unexpected situations.

Table 4.3 lists some standards used in multi-robot systems. For each standard, we detail the concerned layer in the OSI model, its characteristics, the resulting performances, and the hardware and software used for experiments.

Table 4.3 – Examples of some standards used in MRS.

Work	**Standard**	**Layer**	**Characteristics**	**Performances**	**Hardware**	**Software**
[Scherer et al., 2015]	IEEE 802.11s mesh technology	Network layer	Compatible with mesh mode and AP mode	Range: 100m, communication delay: 5ms, Throughput: 10-20Mbit/s	Heterogeneous UAVs + laptop BS + wifi module	Middleware Robot Operating System ROS
[Hayat et al., 2015]	IEEE 802.11n	PHY and MAC layer	High throughput in both mode (AP and mesh), acceptable degree of fairness	Outdoor + mesh mode + single hop; Range: 500m; TCP throughput: 35Mbit/s (50m); FB: 40Mhz	Two Pelican UAVs + laptop BS + Compex WLE300NX 802.11abgn mini PCIe modules	Ubuntu Linux Kernel 3.2. with ath9k driver
	IEEE 802.11ac	PHY and MAC layer	Do not support mesh mode	Outdoor + mesh mode + single hop; Range: 500m; TCP throughput: 10Mbit/s (50m); FB: 80Mhz	Two Pelican UAVs + laptop BS + Compex WLE900N5-18 802.11ac 5Ghz miniPCIe modules	Ubuntu Linux Kernel 3.2. with ath10k driver

[Kuschnig et al., 2012]	IEEE 802.11a	PHY and MAC layer	RSS and throughput decreases with the distance but still at an acceptable level	Outdoor over campus 150m*150m; Range: 10m; Throughput: 54Mbit/s (theo) and 27Mbit/s (prac); FB: 5Ghz	AP: Netgear WNDR3700 + Atheros AR9280-based wireless cards + UAV with Intel Atom Processor + SparkLAN WPEA-110N wireless card + antenna WIMO 18720.11 for UAV and AP	Linux-based OpenWRT Backfire 10.03.1-RC5
[Asadpour et al., 2014]	IEEE 802.11n multi hop	PHY and MAC layer	Long convergence time and high routing overhead (with BATMAN protocol for Network Layer)	In-flight experiment; Throughput for 200m :5.95Mbit/s; Convergence time (20-100m): 28s; Routing overhead (20-100m): 10msg/s	Two Arducopter + GB + WLAN IEEE 802.11n + XBEE-PRO	

[Morgenthaler et al., 2012]	IEEE 802.11s mesh nodes for UAV to UAV and IEEE 802.11g for UAV to AP	network layer	In single hop, flying UAVs reach higher throughput than UAVs in the ground. These results are performed with the location based position mode which are lower than those with the signal strength positioning mode. These performances are higher than in multi-hop.	Single hop (1UAV altitude 3-5m and AP-AP = 75m) in location positioning mode; TCP throughput =6.5Mbit/s + in signal strength positioning mode: TCP throughput =8.1Mbit/s	One UAVNet quadcopters Professional Mesh OM1P + two notebooks	Linux 2.6.37.6 Kernel generated by ADAM (embedded Linux distribution) + driver ath5k
[Muzaffar and Yanmaz, 2014]	IEEE 802.11ab	PHY and MAC layer	Throughput decreases with the increase of the number of nodes	Range: up to $1000m \times 1000m \times 50m$; Throughput: 54Mbit/s (prac); FB: 5Ghz	Simulation: UAVs + Groung Station	Omnet++

[Wang et al., 2014]	IEEE 802.15.7	PHY and MAC layer	Bidirectional transmission, matching filtering and timing error recovery can increase the communication range and stability	*One hop:* throughput: 1.6kb/s, packet loss ratio 5%; *Two hop:* throughput 0.65kb/s, packet loss ratio 15%	Embedded BBB board + LED front end	Linux Kernel 3.8.13

3.2 Network standards and protocols

Since the exchange of information is necessary for coordination, the network used for multi-UAV communication has to ensure that the information circulates properly throughout nodes. Consequently, we have to overcome some issues such as:

- Node failure: When the link is broken or defected, the network must find a path to reach the node.
- Topology changes: Nodes move in the environment and generate changes in the topology that the network have to deal with rapidly.
- Communication bandwidth: UAVs possess certain information to exchange with other and need some bandwidth that supports the data exchange.

Taking into account the above cited issues, and to ensure agood integration between the visual SLAM and the communication, the distributed and cooperative wireless mesh network seems to be the most adequate network typology. It allows to introduce the following advantages:

- Improving the network reliability because infrastructureless network offers several paths, so that information may reach the destination even if only one link is broken.
- Allowing self scalability of the network by a quick adaptation to topology changes.
- Enabling rapid deployment with lower cost back-haul.
- Providing easily coverage in areas which have difficult access.
- Saving battery life due to its lower power consumption.
- Enabling to face a growing number of UAVs in the fleet.

Among the existing mesh network standards, the 802.11s amendment, related to the MAC layer, is interesting for MRS applications. It is an Extended Service Set network that supports broadcast, multicast and unicast communication. It contains the Hybrid Wireless Mesh Protocol (HWMP) as the default routing protocol. The HWMP is a hybrid routing protocol inspired by the AODV (Ad-hoc On-demand Distance Vector: an on demand and reactive portion) and the tree based protocol (a proactive portion). Thereby, it contains the advantages of the reactive protocol since it prepares the routing table when nodes change their position, and thus provides the safer path. On the other hand, it contains the advantages of the proactive protocol since the routing table is ready which allow to save time when needed. In addition to the default routing protocol HWMP, the 802.11s mesh network supports other protocols like Optimized Link State Routing (OLSR), Better Approach to Mobile Ad hoc Networking (BATMAN), Wireless Distribution System (WDS), Open Shortest Path First (OSPF) and BABEL. According to [Wang et al., 2010], BATMAN proved better performances than HWMP and OLSR. Thus, experiments using HWMP and BATMAN have been performed to point out the saved data.

BATMAN protocol definition

The BATMAN[2] is a proactive routing protocol inspired by AODV and OLSR. It is supported by multi-hop Ad Hoc mesh networks. It uses different approaches to route selection node by periodically sending OriGinator Messages (OGM) to neighbors with node information to next hop and destination because the routing decisions are distributed on nodes. In this protocol, each node decides for the next hop and not for the whole route so nodes do not use or even know the topology of the network. In the case of detecting other nodes, BATMAN protocol finds the best route to them. It also keeps track of new nodes and informs its neighbors about their existence.

3.3 Results and discussion

For the evaluation of the networked system using BATMAN[3] protocol, we use heterogeneous nodes composed of three laptops: 2.40GHz dual core Linux machine, 2.27GHz i3 Linux machine, 2.50GHz i5 Linux machine and a Parrot AR-Drone 2.0 (See Figure 4.3).

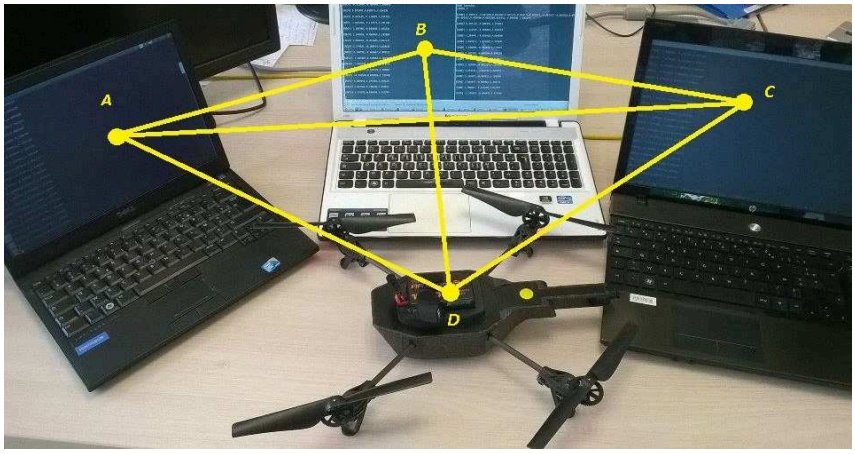


Figure 4.3 – Mesh network illustration between three laptops and one drone.

We simulate the broadcast of data between nodes in both Ad Hoc and mesh network with BATMAN protocol to underline the saved data. The drone was controlled from the laptop using a cross compilation. First, we perform an Ad Hoc network between endpoints and broadcast the data from node A to node B, C and D in the network. Then, we broadcast – in the same conditions – from node A to B, C and D with the BATMAN mesh network protocol. Results in Figure 4.4 show that the throughput achieved an average of 0.4 Mbits/s; whereas, the throughput evaluated in mesh network with BATMAN protocol, achieved an average of 0.65 Mbits/s. The BATMAN mesh network protocol improves by $1,5$ times the throughput of the network compared to a basic Ad Hoc network.

[2]Source: `https://www.open-mesh.org/projects/open-mesh/wiki`
[3]The BATMAN-adv version used for the testbed is available since 2013

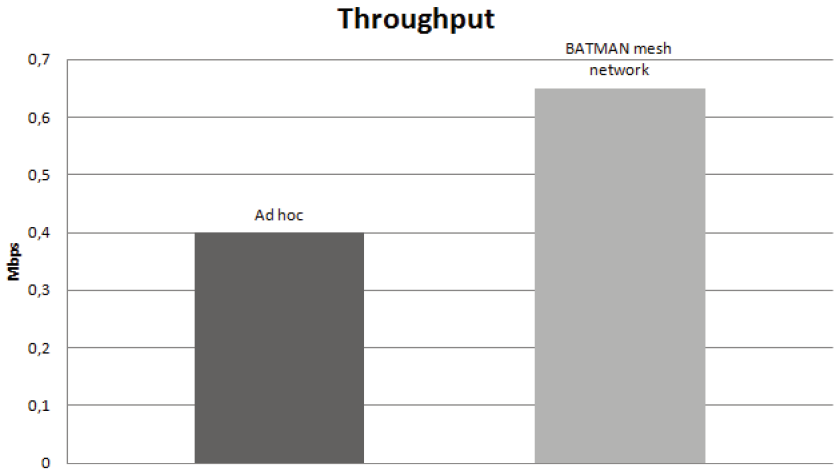


Figure 4.4 – Broadcast testbed throughput result in Ad Hoc and BATMAN mesh network protocol.

4 Network topology and strategy for MRS robustness

The topology of the network defines the geometric properties of nodes (robots in our case). The proposed strategy in this work represents the behavior adopted to face critical situations and to achieve a MRS robustness.

4.1 Related work

The challenge in MRS communication is to maintain reliable network during the mission in order to make the robots able to perform cooperative exploration [Rooker and Birk, 2007, Gupta et al., 2016]. The strategy used for exploration affects the exchange of data among robots including type, destination and frequency.

The information exchanged between a robot and a server may be key-frames and map points [Schmuck, 2017], or only features of selected key-frames and relative-pose estimates among robots and ground station [Forster et al., 2013b]. But mostly, robots exchange their local copies of the map and their poses [Fox et al., 2006, Bresson et al., 2015, Schuster et al., 2015].

The amount of data exchanged may rapidly increase in size, which can cause network congestion and data loss. In order to reduce the bandwidth requirements, authors in [Mohanarajah et al., 2015] propose to send only compressed key-frames and updated key-frame poses. Also, authors in [Cunningham et al., 2010] propose a Decentralized Data Fusion-Smoothing And Mapping (DDF-SAM) approach, where each robot propagates towards other robots, its condensed local graph in order to achieve scalability and robustness to node failure.

Most works deal with the problem of communication while assuming an ideal network or aim at keeping team members within range of one another in order to focus their attention on higher level problems [Schuster et al., 2015, Burgard et al., 2005].

Considering communication loss and/or limited bandwidth helps to prevent from mission failure and to ensure a more realistic scenario. Indeed, in real scenarios, many issues can arise such as having a distance among robots that exceeds the communication range, losing major information in a broken communication link, losing precious time in sending information due to limited bandwidth, etc. The exploration strategy has to take into account the mentioned issues to avoid mission failure in real world scenario. Some works began to tackle the exploration problem while considering communication limitations [Couceiro et al., 2014, Schmuck, 2017].

In [Dai et al., 2018], the aim is to sense a geometrically complex environment by assigning targets to robots when the spatial and temporal resolutions are satisfied. This approach uses a min-max energy path planning algorithm that obeys to a deadline time.

In this work, we make the choice to let UAVs exchange with each other only frontier points, robot poses, and assigned targets. This exchange happens at each iteration while considering UAVs' role, which are adapted according to the network topology. This adaptation allows also to cope with communication limitations.

4.2 Inter-robot communication approach

Interactions among members of the fleet are important especially in the exploration missions in order to prevent UAVs to explore the same regions, and to allow them to cooperatively discover the unknown areas more rapidly and in an optimized manner. However, inter-UAV communication is a challenging issue that requires to answer some practical questions: Which kind of data nodes must exchange? How often data should be shared? Should we consider a multi-hop data exchange? If so, how to identify the endpoints of the data exchange? How to cope with communication limitations? These questions are addressed in the following subsections.

4.2.1 Multi-UAV interaction and data exchange

In the proposed cooperative exploration strategy, local frontier points $\mathbf{f}_{i,j} \in \mathcal{F}$, current pose $\mathbf{p}_i$, and current target point $\mathbf{t}_m$ are exchanged instead of the whole copy of the local map. This is expected to produce a considerable reduction of exchanged data volume, and, consequently, memory consumption. The sequence diagram[4] in Figure 4.5 details (timing and information) the messages exchanged between two UAVs. UAV_i, with $i \in [1..n_c]$ and UAV_j with $j \in [2..n_c]$ $(i < j)$ are robots in the cluster $\mathcal{C}$. They forward their respective *id* number and current poses $\mathbf{p}_i$ and $\mathbf{p}_j$. Since $i < j$, the *explorer* UAV_j sends to the selected *leader* UAV_i its local frontier points $\mathbf{f}_{j,k}$ during the frontier processing (FP) step.

[4]This diagram uses Unified Modeling Language's sequence diagram notation.

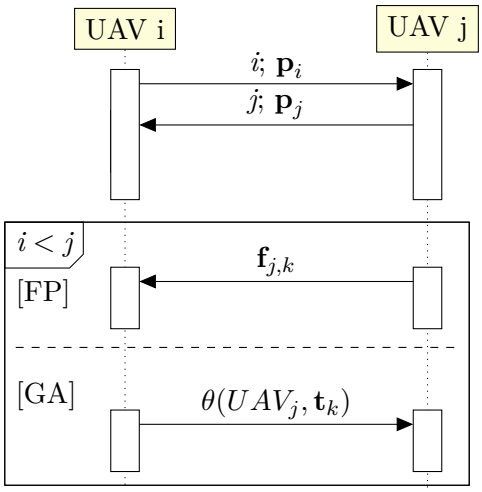


Figure 4.5 – Data flow between two robots. The FP and GA stand, respectively, for frontier processing and goal assignment.

Then, the *leader* performs the goal assignment process (GA) and sends back to UAV_j the selected target point $\mathbf{t}_m$. The cited example represents two UAVs in the cluster $\mathcal{C}$. In case of multiple UAVs in $\mathcal{C}$, the same sequences will be performed among one *leader* and multiple *explorers*.

4.2.2 Exploration strategy to face communication loss

In the proposed system, considering the communication limitations is important to ensure the mission continuity. In case of losing contact with the *leader* due to communication failure or UAV getting stuck, another *leader* is self-selected in the next iteration so that the mission can continue. In Figure 4.6, at $t = t_n$, the fleet is composed of one cluster where UAVs are able to communicate with each other. One *leader* handles the decisions for others. At $t = t_{n+1}$, the communication link fails between UAV_3 and UAV_4. The fleet is divided into two clusters with one *leader* each.

Particular case In case of losing contact with the *leader* and before another one is selected, *explorers* let a timer τ expires while waiting for target assignment. If no target is received, the *explorer* selects its own target according to local information.

Using this strategy, as long as – at least – one UAV exists in the fleet, the mission will continue until all the bounded environment is explored (no candidate frontier points are left).

4.2.3 Data exchange strategy discussion

In the proposed strategy, data flow exchange is repeated at each iteration while taking into account network topology changes to define clusters. The starting points and endpoints

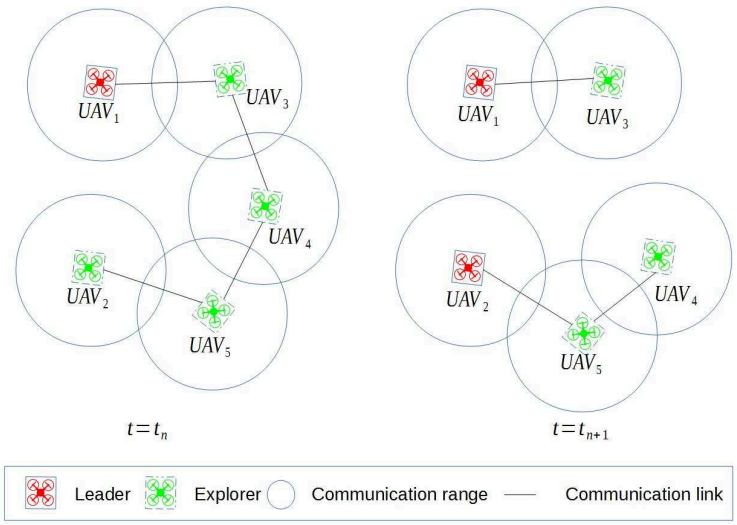


Figure 4.6 – Role evolution in limited communication range.

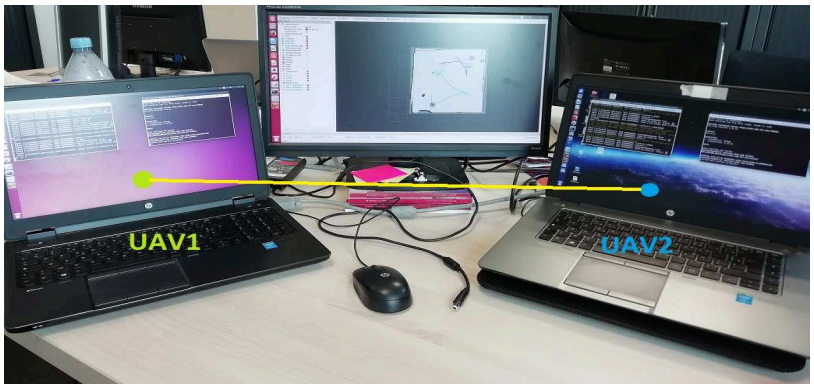
are defined according to these roles. The UAV's role also specifies the type of exchanged data. In addition to the exchanged current pose $\mathbf{p}_i$ and *id* number i, if the UAV$_i$ is an *explorer*, it would passively share information about itself and its surrounding environment with the *leader* (frontier points $\mathbf{f}_{i,j} \in \mathcal{F}$); else, its role would be to send targets to visit to the *explorers* (target points $\mathbf{t}_k \in \mathcal{G}$).
The proposed strategy ensures a mission continuity in case of communication loss. Nevertheless, the UAV may explore regions already explored by other nodes, since no local maps are exchanged nor fused to keep track of visited areas. Thus, in case of communication loss, the mission accomplishment is favored over consumption minimization of resources, such as time and battery.

4.3 Results and discussion

As UAVs are equipped with IEEE 802.11b,g wireless card, we set up an infrastructureless network within the set of robots to quantify the data exchange among members of the fleet, as well as, to determine the performance of the robot network. Runs with 2 and 3 UAVs were performed (See Figure 4.7). The network was composed of two 2.60GHz i7 Linux machines and a 2.50GHz i7 Linux machine. The number of robots used for evaluation is limited to three, however, the proposed system architecture is not constrained to a fixed number of robots.

4.3.1 Network setting: From one to multiple machines

When running multiple UAVs on a single machine (like in previous simulations in chapter 3), one ROS master is responsible of managing the intra and inter-processes communi-



(a) Two UAVs (Laptops)

(b) Three UAVs (Laptops).

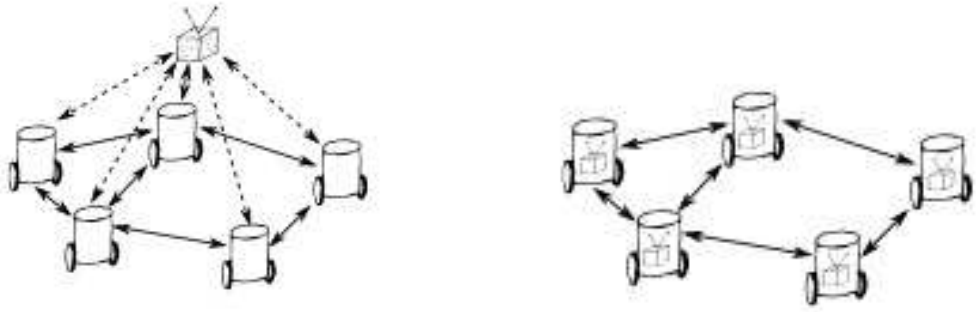
Figure 4.7 – Ad Hoc network illustration during exploration mission.

cation using publisher/subscriber. In case of multiple machines, two configurations are possible:

- One ROS core for multiple machines (See Figure 4.8a): Even with multi-UAV on different machines, one ROS master/core can be adopted by specifying the machine running the ROS core. In this case, the SoS will manage inter-processes communication the same way as if they are on the same machine. The exception is that a real world communication is used instead of a shared memory for a multi-UAV running on a single machine.
- Multiple ROS cores for multiple machines (See Figure 4.8b): when running different ROS cores on different machines, each UAV manages its own master. In this multi-cores system – also called multi-master system –, a synchronization among these masters needs to be done.

Nonetheless for both cases, the virtual environment to be explored in *Gazebo* has to be the same so that UAVs explore the same environment at the same time. Specifically, this means that the IP client of *Gazebo* has to match the IP server of the machine running the *Gazebo* world by setting $GAZEBO\_MASTER\_URI$.

When running a single ROS core for multiple UAVs, a reliable network is needed, else ROS processes would not work properly when the network connection is unstable. Also,



(a) One ROS core for multiple machines. (b) Multiple ROS cores for multiple machines.

Figure 4.8 – Two ROS configurations in multiple machines case. Image from [Andre et al., 2014].

running multi-master system is more realistic because in real world, each UAV has to be functional, independent and cooperative to achieve the mission objectives, which is, in this work, the exploration of an unknown environment. A multi-master system represents a distributed system configuration, which has different advantages such as scalability to fault tolerance.

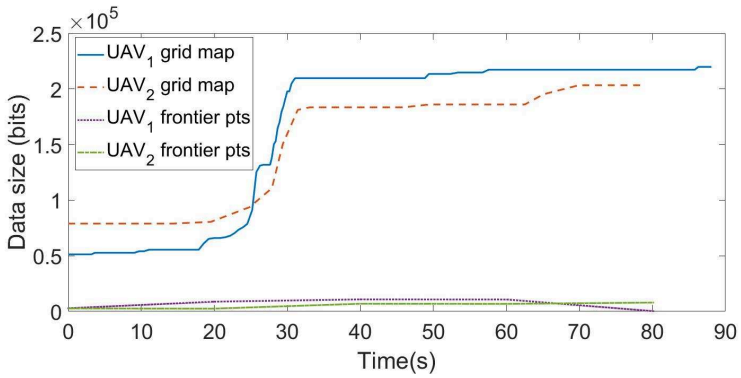
Multi-master systems require synchronization. For this, different packages exist such as the *multi-master fkie* package[5] which allows unicast as well as multicast transmissions using UDP protocol. The $wifi\_comm$ package[6] implements the Optimized Link State Routing (OLSR), but can be used with different routing algorithms. The *recon_multimaster* package[7] is a centralized multi-master system that implements building blocks around the ROS communication layer but do not implement communication itself. Authors in [Andre et al., 2014] propose a distributed approach where each robot runs a master managing its local communication using the *Adhoc_communication* package[8] where AODV protocol is implemented. Another multi-master package is the $Nimbro\_network$ package[9] which offers a robust transport of ROS topics and services over unreliable networks. The above cited multi-master packages are not an exhaustive list and other synchronization packages exist.

For simplicity and as a first approach, the *multi-master fkie* package[10] is used to run the adopted multi-core system. This package allows us to both use and synchronize multiple cores using the default protocol UDP. For ROS topics data exchange, TCP protocol is used. For an effective evaluation especially concerning the time, clock synchronization needs to be ensured. Network Time Protocol (NTP) is used to synchronize laptops within a few milliseconds of Coordinated Universal Time (UTC).

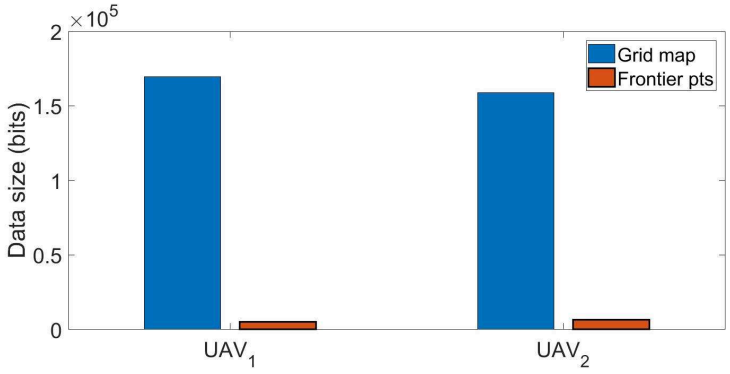
[5]Source: `http://wiki.ros.org/multimaster_fkie`
[6]Source: `http://wiki.ros.org/wifi_comm`
[7]Source: `http://wiki.ros.org/rocon`
[8]Source: `http://wiki.ros.org/adhoc_communication`
[9]Source: `https://github.com/AIS-Bonn/nimbro_network`
[10]Source: `http://wiki.ros.org/multimaster_fkie`

4.3.2 Exchanged data size evaluation

The first evaluation aims at pointing out the amount of exchanged data by sharing local frontier points instead of local grid maps (See Figure 4.9).



(a) Evolution of the amount of exchanged data during the mission.



(b) Average of the amount of exchanged data during the mission.

Figure 4.9 – Data size when UAVs exchange a whole copy of their local grid map versus frontier points of it.

Figure 4.9a shows that the size of grid maps increases consequently in time compared to the size of frontier points that is almost constant during the mission. According to results in Figure 4.9b, the size of data saved, when exchanging frontier points instead of grip maps, is almost divided by 10.

4.3.3 Exchanged data average time evaluation

Depending on the size and frequency of the exchanged data, the time allocated for communication may increase with the increasing number of robots. Thus, evaluations of time behavior and its potential impact on the exploration performances have been conducted. Figure 4.10 shows the network topology evolution during data exchange.

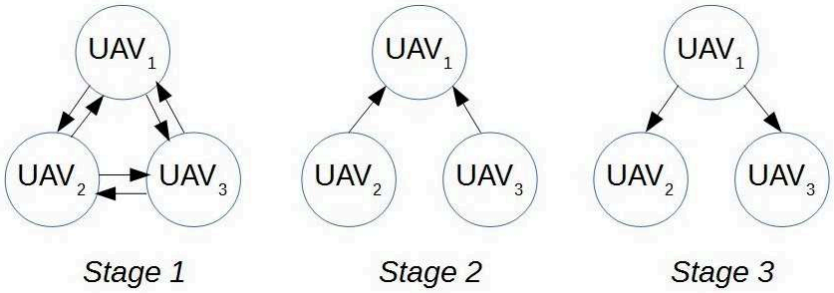


Figure 4.10 – Network topology evolution during a loop with three cooperative UAVs.

The information is exchanged in three stages and they are the following:

- Stage 1: The *id* number and current poses $\mathbf{p}_i$ with $i \in [1..n_c]$.
- Stage 2: The frontier points $\mathbf{f}_{i,j}$ with $i \in [1..n_c]$ and $j \in [1..n_i]$.
- Stage 3: The target points assignment $\theta(UAV_i, \mathbf{t}_k)$ with $i \in [1..n_c]$ and $k \in [1..n_c]$.

Table 4.4 shows the average time spent in data exchange during exploration. A slight increase in the computed average time occurs when increasing the number of robots. The time spent in communication is relatively negligible compared to the total time of exploration.

Table 4.4 – Communication module timings.

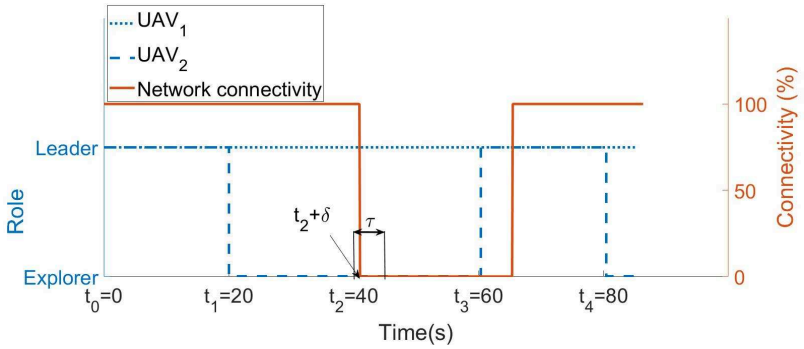
UAVs		**Time spent in *stage 1* (s)**			**Time spent in *stage 2* (s)**			**Time spent in *stage 3* (s)**			**Time for exploration (s)**
		UAV_1	UAV_2	UAV_3	UAV_1	UAV_2	UAV_3	UAV_1	UAV_2	UAV_3	
Two UAVs	UAV_1	∅	0.136± 0.139		∅	∅		∅	0.022± 0.017		120,1
	UAV_2	0.065± 0.068	∅		0.026± 0.008	∅		∅	∅		
Three UAVs	UAV_1	∅	0.056± 0.065	0.575± 0.769	∅	∅	∅	∅	0.335± 0.407	0.765± 0.678	86
	UAV_2	0.107± 0.111	∅	0.483± 0.678	0.185± 0.244	∅	∅	∅	∅	∅	
	UAV_3	0.267± 0.165	0.616± 0.549	∅	0.251± 0.109	∅	∅	∅	∅	∅	

4.3.4 Network interruption evaluation

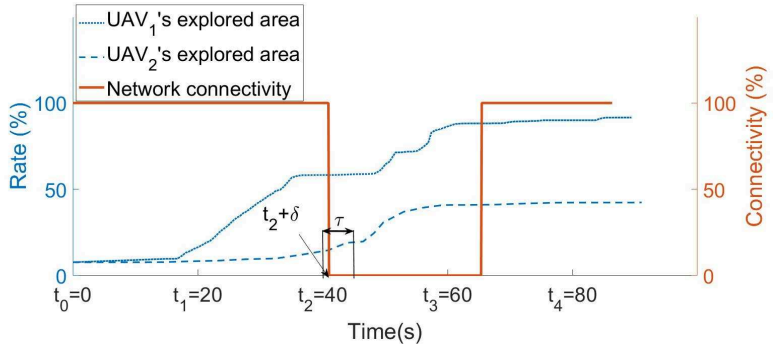
To evaluate the system behavior during communication failure, the network connectivity has been voluntarily interrupted during the exploration. Figure 4.11 shows the robot's role and the exploration rate performance when the network connectivity is interrupted and then recovered.

The system performs neighbor discovering, role selection and target assignment at each loop of $t_{i+1} = t_i + i.r$ with $t_0 = 0s$ and $r = 20s$. In Figure 4.11a, at $t = t_0$, both robots begin with a *leader* role. When discovering each other (at $t = t_1$), UAV_1 selects itself as *leader* and UAV_2 becomes *explorer*. Consequently, UAV_1 assigns a target to UAV_2. UAV_2 receives the target and attempts to reach it. At $t = t_2 + \delta$, the connectivity is voluntarily interrupted, just after the role selection but before the target information is assigned to UAV_2. After a time period τ, UAV_2 selects a target taking into account its own local data. In the next loop (at $t = t_3$), since the connectivity is still interrupted, UAV_2 finds no neighbors and selects itself as a *leader*. Both robots perform exploration independently, that is, without cooperation. Shortly after $t = t_3$, the connectivity is re-established. Thus, robots are able to cooperate again and UAV_2 takes over the role of *explorer*.

In Figure 4.11b, even after the network connectivity is interrupted, the exploration continues to be performed by both UAVs. When the connectivity is re-established, the *leader* collects the frontier points and performs frontier processing where it finds that no candidates targets are remaining, that is, all the environment is now explored. Therefore, the mission is accomplished.



(a) UAV's role along with network connectivity.



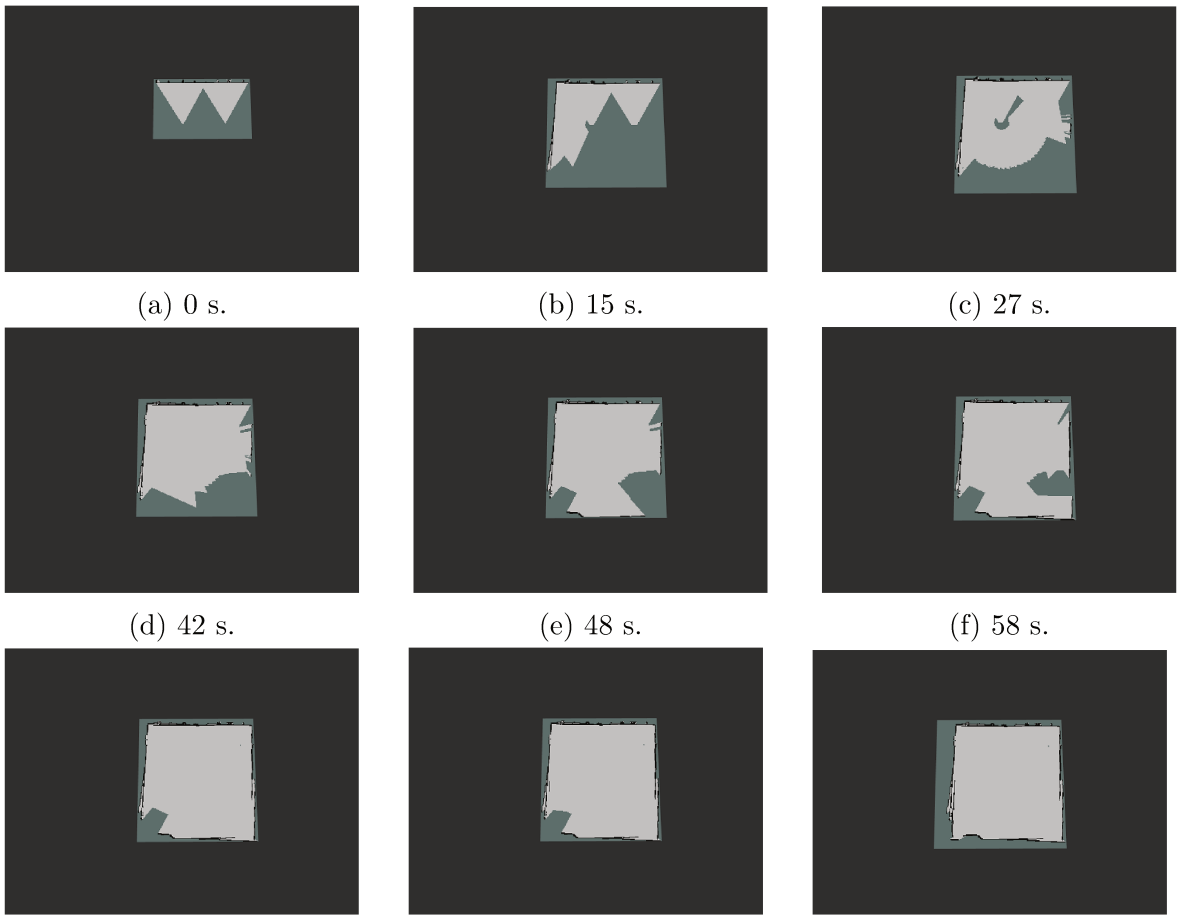
(b) Exploration rate along with network connectivity.

Figure 4.11 – Two cooperative robots exploration along with network connectivity.

4.3.5 Global map evaluation

The exploration mission aims at creating a global map of the environment, in an efficient manner. Hence, we want to evaluate the obtained global map during a cooperative mission running with infrastructureless network.

Figure 4.12 illustrates the growth of the reconstructed global 2D occupancy grid map during the exploration mission.



(a) 0 s. (b) 15 s. (c) 27 s.

(d) 42 s. (e) 48 s. (f) 58 s.

(g) 64 s. (h) 71 s. (i) 73 s.

Figure 4.12 – Global reconstructed 2D occupancy grid map during exploration mission with two cooperative UAVs.

5 Conclusion

In this chapter, we addressed two subproblems of the MRS network. The first one deals with the network typology that is the characteristics that should be ensured by the network; whereas, the second one points out to the geographic distribution of nodes in the network, called topology. We also propose strategies to face critical situations.

The typology of the proposed network is based on adopting a mesh network along with BATMAN protocol. Results of broadcasting data performed in testbed with heterogeneous nodes including three laptops and one AR-Drone show that BATMAN mesh network protocol improves the throughput by 1.5 times compared to basic Ad Hoc network.

In a limited communication range, the topology of the network varies during the mission. Consequently, the UAVs' behaviors are adapted by always updating their roles to *leader* or *explorer* during the mission. Even with an *explorer* role, the UAV is able to continue its mission if the network experiences some issues. Furthermore, we propose to exchange the frontier points of the local map instead of the whole copy of it, which allows to reduce the shared data volume, and consequently memory consumption. The results of the testbed performed with three UAVs, show that the proposed communication module is able to cope with network limitations. They also show that the proposed strategy uses 10 times less data than a strategy that makes the robots exchange the whole local map.

Conclusions and future work

Contents

1 Summary

In this thesis, we introduced a new distributed multi-UAV system framework for cooperative unknown environment exploration. Equipped with an embedded visual sensor, each UAV is able to localize itself, to discover its neighbors, and to create a 3D grid map of its environment. UAVs are also able to communicate and to exchange specific data for coordinated exploration purpose. Each of the main topics of SLAM, exploration strategy, and inter-robot communication was studied throughout this work.

We started this dissertation by introducing a software architecture for multi-UAV system (See Figure 1.6). This block diagram aims at managing the different modules and the data flows between them. We proposed to divide the fleet into clusters, where each cluster defines UAVs within each others' communication range. Each cluster selects the UAV with the lowest *id* number as the *leader* that is in charge of making decisions, based on some specific shared information, for all the other robots in the group that have the role of *explorers*. An overview on the UAVs' system coordinate frames was also presented.

In the second chapter, we studied two approaches for the SLAM problem. The first approach was using a monocular camera for an inertial graph-based SLAM system. Results show that the fusion of these sensors reduces the trajectory drift. The second approach used an RGB-D camera as input for a feature based SLAM approach. Simulation results show that the RGB-D SLAM is able to make an efficient pose estimate in difficult conditions such as simulated environment. Using these SLAM approaches, we consider that neither global information nor map or GPS are available. Also, no global frame is predefined. Only an arbitrary frame, that coincides with the *leader*'s reference frame, is set. However, the time of the mission increased since the speed of the robot was reduced to avoid the tracking loss.

Also, we proposed an exploration strategy to coordinate the UAVs in the fleet. This strategy is based on a novel utility function that takes into account the distance of each

UAV in the group from the unexplored set of targets, and makes a trade-off between fast exploration and getting a detailed grid map. Using the group-*leader* decision making, targets are assigned to UAVs in order to simultaneously explore different regions of the environment in an optimized manner. Results show that the strategy adopted minimizes the mission time by 25% for 2 UAVs and by 30% for 3 UAVs, while decreasing the average traveled distance by each UAV in the fleet by 55% for 2 UAVs and by 62% for 3 UAVs . In addition, by scheduling information, UAVs are efficiently spread out into the environment while avoiding to select the same target or another one close to it. Furthermore, the previously proposed RGB-D SLAM (in Chapter 2) was used as the relative localization system for each UAV in the fleet during a cooperative exploration mission.

Moreover, we studied and evaluated the inter-UAV communication from two aspects: the typology and the topology of the network. We first analyzed the network requirements for a multi-UAV system. Then, based on this study, we proposed to use the mesh network with the BATMAN protocol. Testbed results show that the throughput of the network is improved. For the second aspect, we addressed the strategies adopted to cope with communication limitations during an exploration mission. we also evaluated the amount of data exchanged during the mission. Testbed results show that by exchanging frontier points, local poses and assigned targets, the adopted strategy uses 10 times less data than a strategy that makes the robots exchange the whole local map. Furthermore, the group-*leader* decision making allows to take into account the communication drop-out or failure by adapting the UAV's role according to the network topology changes.

2 Future research

Several aspects of this project can be improved: First, how to deal with the map tracking when the communication is lost? Indeed, when the network is interrupted, the main goal of each robot is to finish the exploration mission regardless of the rest of the fleet because local maps are neither exchanged nor fused. Thus, with the proposed exploration strategy, we aim to investigate the possibility of keeping track of other UAVs' explored area using the frontier points. Each robot receives all other robots' local frontier points, stores them and process them to get the frontier points of the global map. For achieving this purpose, the frontier points are broadcasted to all the robots in the fleet.

Second, we plan to include more complex environments in our experiments with more obstacles (the simulated environment in this work is free from obstacles). This assumption is due to the adopted frontier points processing approach that needs maps that are bounded and without obstacles inside (other frontier points). Also, the utility function takes into account the distance of the robot from the target. This can cause false target choice if an obstacle exists between this target and the actual robot pose. Instead of the robot-to-target distance, we can consider the path or even the energy to spend to reach the future target.

Third, an important point is to improve the environment to a 3D one. In this case, the frontier points will be computed from the 3D occupancy grid map. We also need to consider a 3D path planning and control approach.

Fourth, another aspect is the large scale evaluation. The proposed system architecture is not limited to a fixed number of robots. We aim at increasing the size of the fleet for a quantitative performance evaluation. By increasing the number of UAVs, we aim to evaluate not only the exploration time curve evolution, but also the processing time performances.

Furthermore, we plan to implement the framework on a quad-rotor fleets. Hence, the UAV chosen for the experiments have to support the kinect's payload, and the developed ROS packages. This experiment will be conducted in an indoor environment with a global localization (OptiTrack: motion capture system) to evaluate the exploration performances. Then, the SLAM approach will be implemented to test the overall mission with a relative localization approach.

And last but not the least, we aim at mounting visual fiducial markers on each UAV in order to estimate the relative transformation among robots by detecting these tags. The markers have to be mounted all around the robot with different unique identifiers to recognize its orientation. At first, this experiment can be tested using static robots and then be used during the exploration mission.

This list is not exhaustive and continuation of this work may address many other aspects.

Development tools

Contents

1 Robot Operating System

1.1 Overview

The Robot Operating System (ROS) [Quigley et al., 2009] is an open source (BSD license) project, that accumulates a significant number of community developed drivers and packages. As its name suggests, ROS is not only a middleware, but it is a complete operating system that aids to the development of various robot software. It supplies low-level drivers, communication layer, and high-level recognition and planning applications. ROS contains a set of individual package executable entities written using mainly the following client libraries roscpp (C++ library) or rospy (Python library). ROS is continually updated and improved with different distributions. In our work, we adopt the ROS indigo distribution which is compatible with Ubuntu 14.04 that we use.

1.2 Framework

ROS[1] is a flexible framework composed of a group of processes that communicate with each other. For that, it offers a message passing interface that provides inter-process communication, commonly called middleware. This middleware is composed of:

[1]Source: `http://www.ros.org`

- Master: The master is the key component of the ROS middleware without which processes would not work properly. The ROS master allows the inter-processes communication in synchronous (node-to-node via services) and asynchronous (node-to-topic via messages) manner.

- Nodes: The Node is one of the most fundamental component of ROS. It is an individual processing module in charge of performing a set of tasks.

- Messages: The messages contain information of different types such as integer, boolean, string, and so on. There a variety of commonly used and predefined messages such as *Pose*[2] and *OccupancyGrid*[3] messages. A new message can also be defined by creating a ".msg" file.

  In order to exchange frontier points between UAVs (See Section 4 of Chapter 4), we created a message called *VectorFrontier* composed of a set of created messages *FrontierPoint*. Figure A.1 shows the *VectorFrontier* message definition.

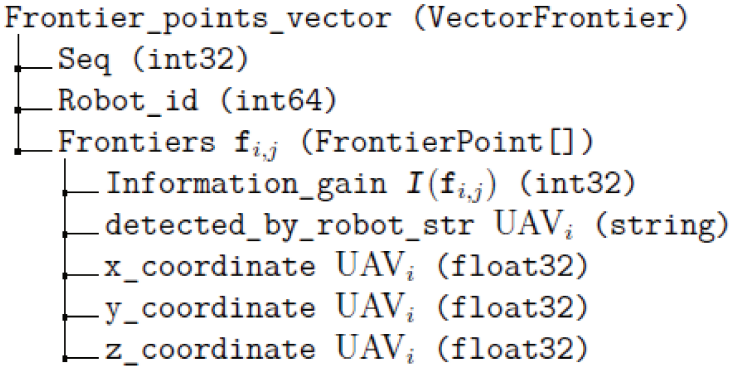


Figure A.1 – Frontier points vector message definition.

- Topics: The topics are used to share messages between nodes. A node subscribe to a topic if it needs to listen to some information that are encapsulated in messages. whereas, if the node needs to share some information, it publishes the messages to a topic. A node can subscribe/publish messages to several topics at a time, however, both node and topic have to have the same message type.

- Services: The services are used for synchronous communication between the nodes. A node, called client, requests for a service from a node, called server. Services are similar to messages but are composed of a request and a response components.

Figure A.2 illustrates the interaction between some components of the ROS middleware.

[2]Source: `http://docs.ros.org/melodic/api/geometry_msgs/html/msg/Pose.html`
[3]Source: `http://docs.ros.org/melodic/api/nav_msgs/html/msg/OccupancyGrid.html`

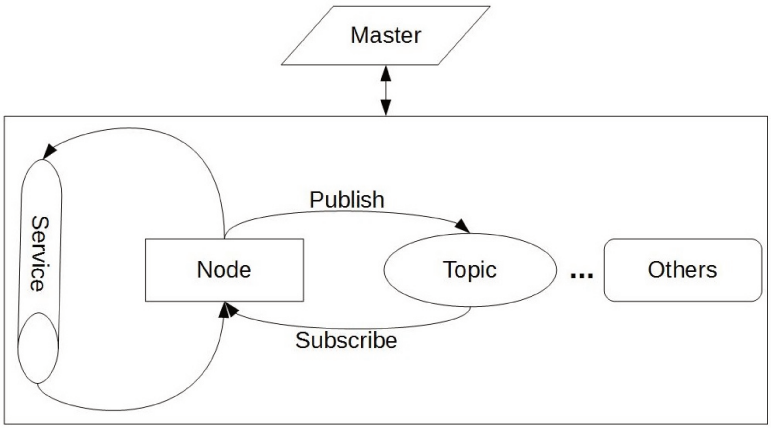


Figure A.2 – Illustration of ROS middleware components interaction.

1.3 ROS development tools

ROS provides several development tools to assist the creation of robotic applications. These are some of the available tools that we used to develop our multi-UAV exploration system:

1.3.1 Visualization tool: Rviz

Rviz[4] is a visualization tool used to display the information in a 3D world (See Figure A.3). For that, several built-in displays are available such as image, pose, axes, and so on. Displays can be added, removed and configured to fit the application.

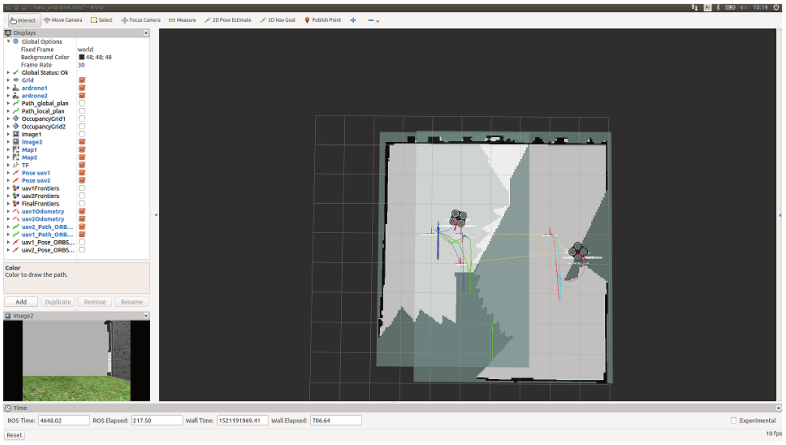


Figure A.3 – Visualization of two UAVs navigation using *Rviz*.

[4]Source: `http://wiki.ros.org/rviz`

1.3.2 Graph Qt-based framework: Rqt_graph

Rqt_graph[5] is a GUI plug-in to visualize the inter-process communication graph. It allows to keep track of the different interaction between ROS modules. Mainly, we use this tool to visualize the interaction between the nodes and topics. As an example, Figure A.4 shows the ROS topics involved during an exploration mission with two cooperative UAVs where each one performs ORB-SLAM2 (See Section 4.3 in Chapter 3).

[5]Source: `http://wiki.ros.org/rqt_graph`

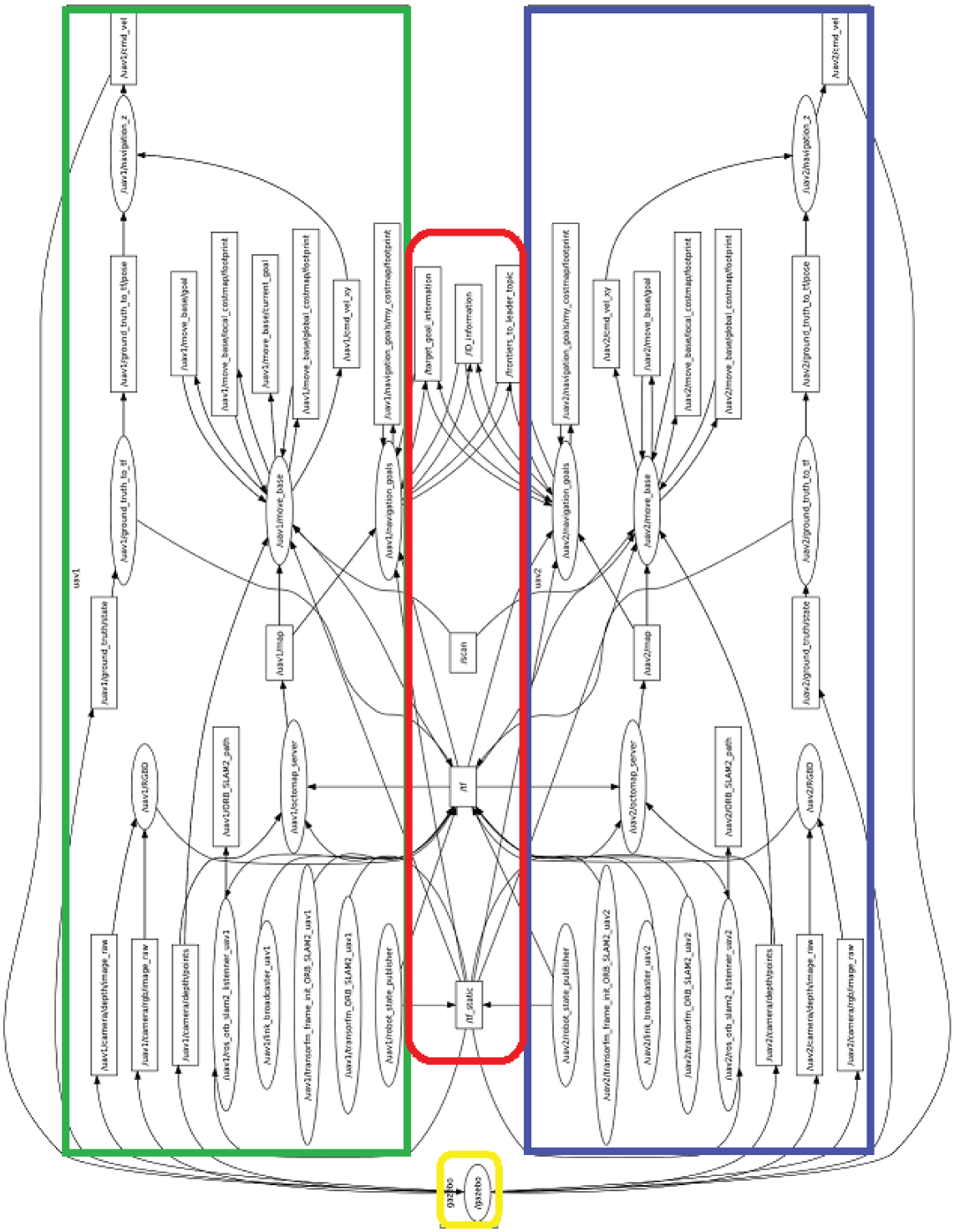


Figure A.4 – ROS graph diagram showing topics involved in two cooperative UAVs exploration mission using ORB-SLAM2 approach. Topics related to: UAV_1 are within the Green box (top), UAV_2 are within the Blue box (bottom), the exchanged data are within the Red box (middle right), and *Gazebo* environment are within the Yellow box (middle left).

1.3.3 Transform libraby: Tf

Tf[6] is an important package in ROS that is used to visualize the relationship between coordinate frames. This package has been very useful for us to see the tree structure of the frames. Figure A.5 shows the frames related to each UAV w.r.t. the world frame using ROS TF tool (See Section 5.3.2 in Chapter 2).

[6]Source: `http://wiki.ros.org/tf`

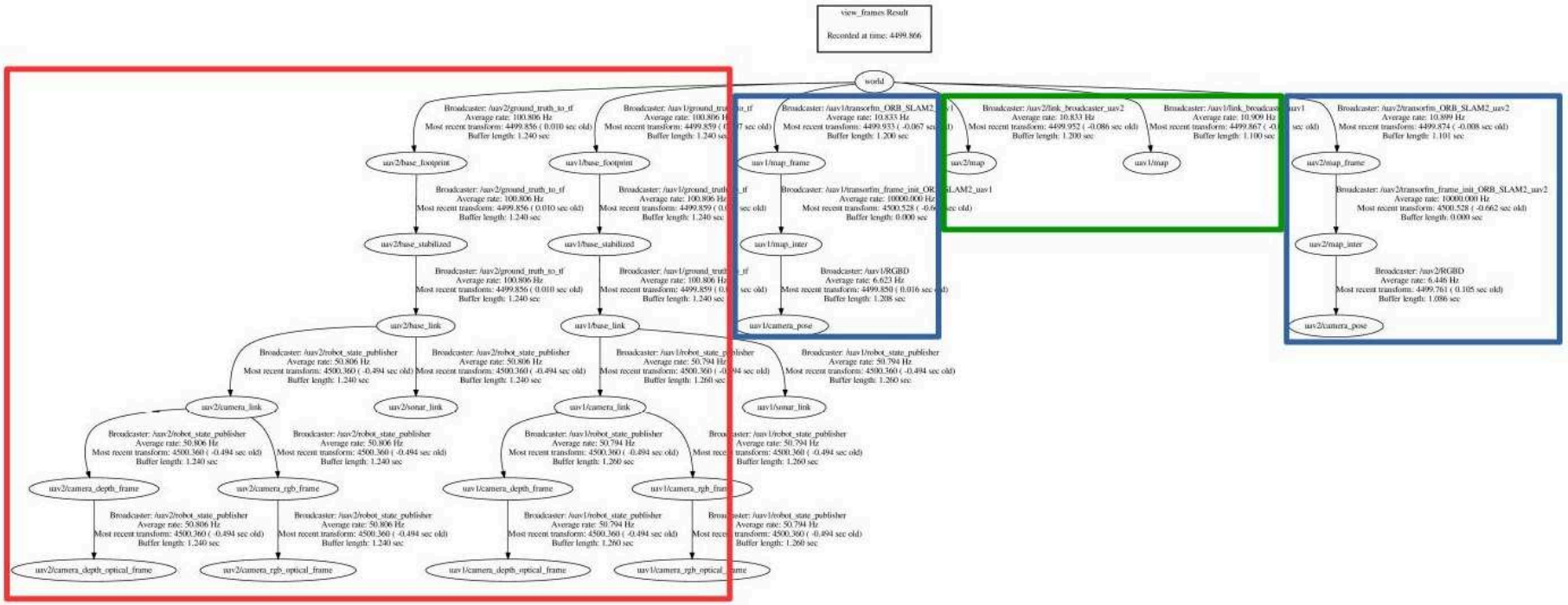


Figure A.5 – ROS TF tree showing the relative frames of two UAVs w.r.t. the world frame. The figure contains frames related to the global localization in the red box (left), frames related to the relative localization computed by ORB-SLAM2 in th two blue boxes (the second box starting from the left and the box on the right), and frames related to the local maps in the green box (the third box starting from the left).

1.3.4 Bag tools

Bag[7] files are used to store some information by subscribing to topics. The recorded information are played back using the *rosbag* tool in on-line or off-line manners. To visualize the content of a bag file, the *rqt bag* tool is used.

Retrieve information from bag files In this work, we use bag files to record results from cooperative UAVs exploration such as pose $\mathbf{p}_i$, frontier point $\mathbf{f}_{i,j}$, and 2D occupancy grid map $\mathcal{L}$. Then, to evaluate the proposed strategy's performances, we use Matlab[8]. We retrieve the information from the bag files using the *rosbag* function from the *Robotic System Toolbox*[9].

1.4 Important ROS packages

ROS supplies several open source packages to assist in the development of robotic applications. These packages are always updated with the latest releases. Each package commonly provides topics to subscribe to, topics where the information will be published, and some tuning parameters (e.g. frequency, time, etc.) to be able to adapt the package to the application. These popular packages are used for visualization, path planning, navigation, communication, and so on.

1.4.1 Move base package

The *move base* is a freely available node used for two dimensional navigation. Actually, the *move base* node is the major component of the navigation stack. It allows to plan a path to a target and to attempt to reach it. In addition to the assigned target, it takes into account inputs such as sensor measurements (e.g. point cloud and laser scans), odometry, 2D map, etc. Using these information, it provides a path to be followed by giving velocity commands to the base controller. The *move base* framework is described in Figure A.6.

The *move base* constructs a global and a local costmap with their corresponding global and local planner. The costmap is a representation of the environment with inflated obstacles. The global costmap represents a model of the whole environment whereas the local costmap represents a part of the global costmap that is moving with the robot. Based on these costmaps, the node computes the global and local plan, respectively. The global planner is based on the *Navfn*[11] planner which provides a fast interpolated navigation function – computed with Dijkstra's algorithm [Dijkstra, 1959] – that can be used to create plans. For the

[7]Source: `http://wiki.ros.org/Bags`
[8]Source: `https://fr.mathworks.com/products/matlab.html`
[9]Source: `https://fr.mathworks.com/products/robotics.html`
[10]Source: `http://wiki.ros.org/move_base`
[11]Source: `http://wiki.ros.org/navfn`

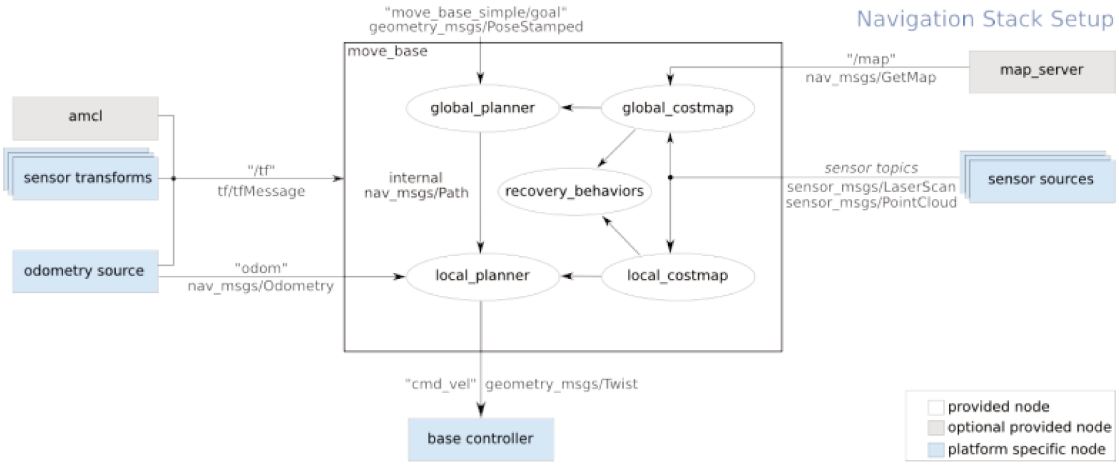


Figure A.6 – *move base* package framework[10].

local navigation on a plan, the *base local planner*[12] provides implementations of the Trajectory Rollout and Dynamic Window approaches. When the robot perceives itself as stuck, the *move base* node may optionally perform recovery behaviors to attempt to clear out space.

1.4.2 Fkie package

The *fkie* package is an open source ROS stack that deals with communication in systems with multiples ROS cores. It is composed of a set of nodes in order to establish and manage a multimaster network.

To synchronize ROS cores (See Figure A.7), the package make use of two main modules:

- Master discovery module: It is used to discover the existing masters in the local network by sending echo heartbeat messages to a defined unicast robot or multicast group. This module allows the connection with the ROS master (link 1). It also publish/listen to changes over the network (link 2). These changes are then published to the ROS topics (link 3).

- Master synchronizing module: It is used to synchronize the local master with the discovered ones in the network. This module is connected to the *master discovery module* (link 4). It asks for the actual ROS state and registers the remote topics/services on the local ROS master (link 5). One master synchronization module should be started for each ROS system for a complete synchronization.

[12]Source: `http://wiki.ros.org/base_local_planner`
[13]Source: `http://wiki.ros.org/multimaster_fkie`

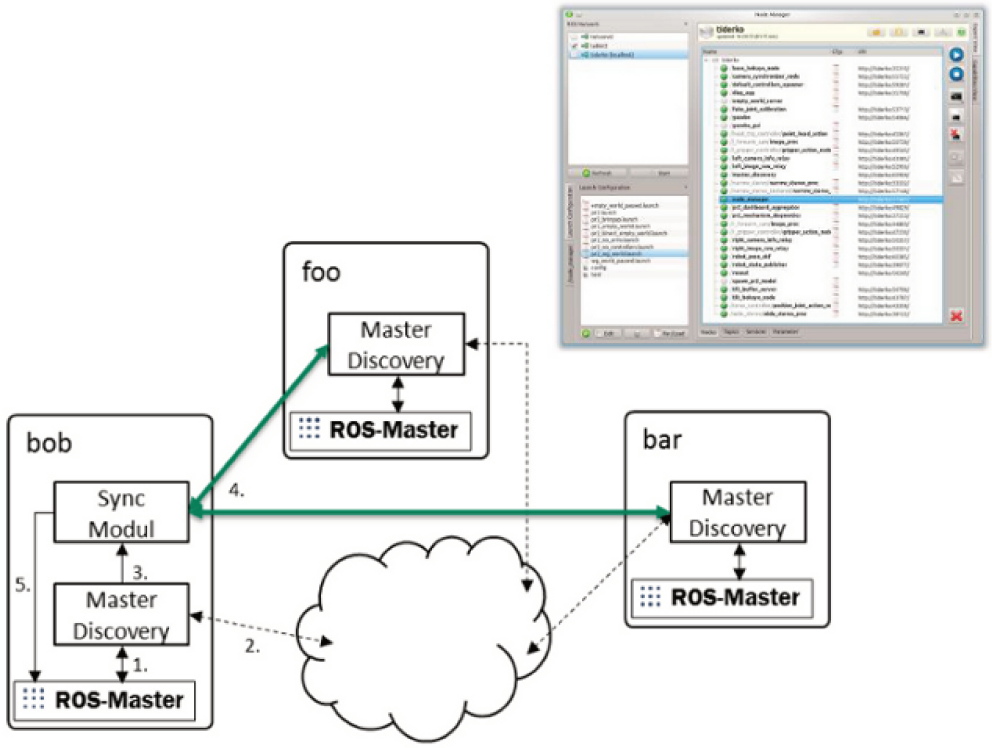


Figure A.7 – Multimaster *fkie* overview[13].

1.4.3 Ardrone autonomy package

For the UAV simulation, we used the Parrot AR-Drone 1.0 quadrocopter model from the *ardrone autonomy* package[14] (See Figure A.8). It was developed in the Autonomy Lab of Simon Fraser University by Mani Monajjemi and other contributors. This package provides the ROS driver with all the needed models (camera, kinect, Lidar, IMU, etc.) to simulate an AR-drone quadrocopter (This site[15] is available for more details on usage and implementation).

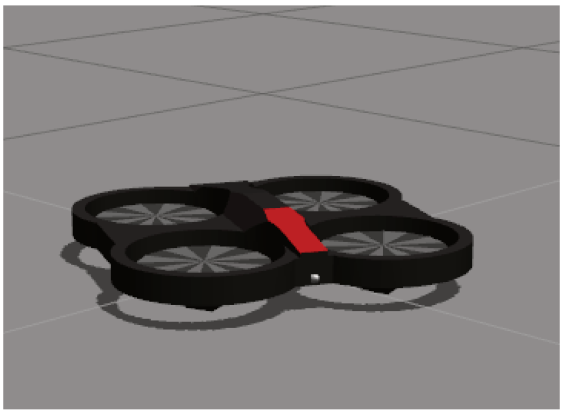


Figure A.8 – AR-Drone model.

[14]Source: `http://wiki.ros.org/ardrone_autonomy`
[15]Source: `http://ardrone-autonomy.readthedocs.io/en/latest/`

2 Gazebo simulator

Gazebo[16] is a 3D dynamic simulator able to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments. *Gazebo* simulator is based on Ubuntu and is compatible with ROS. Consequently to the ROS indigo distribution that we use, *Gazebo* 2.x version is chosen.

Gazebo simulates the environment using a set of models such as sensors, robots, objects, etc. *Gazebo* environment is a ".world" file containing some static and dynamic objects. We can use available worlds[17] such as the willow garage, or build our own world by inserting existing *Gazebo* models[18]. These models can be configured to reach the environment's expectations. Also, a new tool called *building editor*[19], that allows to design buildings for simulation, can be used. It is available on *Gazebo* 2 but, the most of its advanced features, such as color and texture, make appearance in *Gazebo* 5. Hence, we created an environment (See Figure A.9) using mesh models from 3D Warehouse[20]. These mesh models are availible in the Collada format ".dae" suitable for *Gazebo*.

Figure A.9 – *Gazebo* environment.

[16]Source: `http://gazebosim.org/`
[17]Source: `https://github.com/arpg/Gazebo/tree/master/worlds`
[18]Source: `https://bitbucket.org/osrf/gazebo_models`
[19]Source: `http://gazebosim.org/tutorials?tut=building_editor`
[20]Source: `https://3dwarehouse.sketchup.com`

Bibliography

[Andre et al., 2014] Andre, T., Neuhold, D., and Bettstetter, C. (2014). Coordinated multi-robot exploration: Out of the box packages for ros. In *Globecom Workshops (GC Wkshps), 2014*, pages 1457–1462. IEEE.

[Asadpour et al., 2014] Asadpour, M., Egli, S., Hummel, K. A., and Giustiniano, D. (2014). Routing in a fleet of micro aerial vehicles: First experimental insights. In *Proceedings of the Third ACM Workshop on Airborne Networks and Communications*, AIRBORNE '14, pages 9–10, New York, NY, USA. ACM.

[Bautin et al., 2012] Bautin, A., Simonin, O., and Charpillet, F. (2012). Stratégie d'exploration multirobot fondée sur les champs de potentiels artificiels. *Revue des Sciences et Technologies de l'Information-Série RIA: Revue d'Intelligence Artificielle*, 26(5):523–542.

[Beard and McLain, 2003] Beard, R. W. and McLain, T. W. (2003). Multiple uav cooperative search under collision avoidance and limited range communication constraints. In *42nd IEEE International Conference on Decision and Control*, volume 1, pages 25–30.

[Benavides et al., 2016] Benavides, F., Monzón, P., Chanel, C. P. C., and Grampín, E. (2016). Multi-robot cooperative systems for exploration: Advances in dealing with constrained communication environments. In *Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR), 2016 XIII Latin American*, pages 181–186. IEEE.

[Bennewitz and Burgard, 2000] Bennewitz, M. and Burgard, W. (2000). An experimental comparison of path planning techniques for teams of mobile robots. In *Autonome Mobile Systeme 2000*, pages 175–182. Springer.

[Bloesch et al., 2017] Bloesch, M., Burri, M., Omari, S., Hutter, M., and Siegwart, R. (2017). Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*, 36(10):1053–1072.

[Boardman and Sauser, 2006] Boardman, J. and Sauser, B. (2006). System of systems - the meaning of of. In *2006 IEEE/SMC International Conference on System of Systems Engineering*, pages 6 pp.–.

[Bouachir, 2014] Bouachir, O. (2014). *Design and implementation of communication architecture for civil mini-UAVs*. Theses, Université Toulouse 3 Paul Sabatier (UT3 Paul Sabatier).

[Boulding, 1956] Boulding, K. E. (1956). General systems theory - the skeleton of science. *Management Science*, 2(3):197–208.

[Brand et al., 2014] Brand, C., Schuster, M. J., Hirschmüller, H., and Suppa, M. (2014). Stereo-vision based obstacle mapping for indoor/outdoor slam. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1846–1853.

[Bresson et al., 2015] Bresson, G., Aufrère, R., and Chapuis, R. (2015). A general consistent decentralized simultaneous localization and mapping solution. *Robotics and Autonomous Systems*, 74:128 – 147.

[Burgard et al., 2000] Burgard, W., Moors, M., Fox, D., Simmons, R., and Thrun, S. (2000). Collaborative multi-robot exploration. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 476–481. IEEE.

[Burgard et al., 2005] Burgard, W., Moors, M., Stachniss, C., and Schneider, F. E. (2005). Coordinated multi-robot exploration. *IEEE Transactions on robotics*, 21(3):376–386.

[Cadena et al., 2016] Cadena, C., Carlone, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., and Leonard, J. J. (2016). Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332.

[Cameron and Trigoni, 2009] Cameron, S. and Trigoni, N. (2009). Collaborative sensing by unmanned aerial vehicles. In *3rd International Workshop on Agent Technology for Sensor Networks (ATSN 09)*.

[Cieslewski et al., 2017] Cieslewski, T., Kaufmann, E., and Scaramuzza, D. (2017). Rapid exploration with multi-rotors: A frontier selection method for high speed flight. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

[Concha et al., 2016] Concha, A., Loianno, G., Kumar, V., and Civera, J. (2016). Visual-inertial direct slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1331–1338.

[Couceiro et al., 2014] Couceiro, M. S., Figueiredo, C. M., Rocha, R. P., and Ferreira, N. M. (2014). Darwinian swarm exploration under communication constraints: Initial deployment and fault-tolerance assessment. *Robotics and Autonomous Systems*, 62(4):528 – 544.

[Cui et al., 2006] Cui, J.-H., Kong, J., Gerla, M., and Zhou, S. (2006). The challenges of building mobile underwater wireless networks for aquatic applications. *IEEE Network*, 20(3):12–18.

[Cunningham et al., 2010] Cunningham, A., Paluri, M., and Dellaert, F. (2010). Ddf-sam: Fully distributed slam using constrained factor graphs. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference*. IEEE.

[Dai et al., 2018] Dai, R., Fotedar, S., Radmanesh, M., and Kumar, M. (2018). Quality-aware uav coverage and path planning in geometrically complex environments. *Ad Hoc Networks*.

[De Hoog et al., 2009] De Hoog, J., Cameron, S., and Visser, A. (2009). Role-based autonomous multi-robot exploration. In *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD'09. Computation World:*, pages 482–487. IEEE.

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271.

[Dudek et al., 1996] Dudek, G., Jenkin, M. R. M., Milios, E., and Wilkes, D. (1996). A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397.

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110.

[Endres et al., 2014] Endres, F., Hess, J., Sturm, J., Cremers, D., and Burgard, W. (2014). 3-d mapping with an rgb-d camera. *Robotics, IEEE Transactions on*, 30(1):177–187.

[Engel et al., 2016] Engel, J., Koltun, V., and Cremers, D. (2016). Direct sparse odometry. *CoRR*, abs/1607.02565.

[Engel et al., 2014] Engel, J., Schöps, T., and Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. In *Computer Vision–ECCV 2014*, pages 834–849. Springer.

[Erdelj et al., 2017a] Erdelj, M., KrÃşl, M., and Natalizio, E. (2017a). Wireless sensor networks and multi-uav systems for natural disaster management. *Computer Networks*, 124:72 – 86.

[Erdelj et al., 2017b] Erdelj, M., Natalizio, E., Chowdhury, K. R., and Akyildiz, I. F. (2017b). Help from the sky: Leveraging uavs for disaster management. *IEEE Pervasive Computing*, 16:24–32.

[Erdelj et al., 2017c] Erdelj, M., Saif, O., Natalizio, E., and Fantoni, I. (2017c). Uavs that fly forever: Uninterrupted structural inspection through automatic uav replacement. *Ad Hoc Networks*.

[Eudes et al., 2018] Eudes, A., Marzat, J., Sanfourche, M., Moras, J., and Bertrand, S. (2018). Autonomous and safe inspection of an industrial warehouse by a multi-rotor mav. In *Field and Service Robotics*, pages 221–235. Springer.

[Faigl et al., 2012] Faigl, J., Kulich, M., and Přeučil, L. (2012). Goal assignment using distance cost in multi-robot exploration. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3741–3746. IEEE.

[Fairfield et al., 2007] Fairfield, N., Kantor, G., and Wettergreen, D. (2007). Real-time slam with octree evidence grids for exploration in underwater tunnels. *Journal of Field Robotics*, 24:03–21.

[Fang and Scherer, 2014] Fang, Z. and Scherer, S. (2014). Experimental study of odometry estimation methods using rgb-d cameras. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 680–687. IEEE.

[Fang and Zhang, 2015] Fang, Z. and Zhang, Y. (2015). Experimental evaluation of rgb-d visual odometry methods. *International Journal of Advanced Robotic Systems*, 12.

[Farinelli et al., 2004] Farinelli, A., Iocchi, L., and Nardi, D. (2004). Multi robot systems: A classification focused on coordination. *IEEE Transactions on System Man and Cybernetics, part B*, 34(5):2015–2028. New York, (USA).

[Forster et al., 2015] Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2015). Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. In *Robotics: Science and Systems*.

[Forster et al., 2013a] Forster, C., Lynen, S., Kneip, L., and Scaramuzza, D. (2013a). Collaborative monocular slam with multiple micro aerial vehicles. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3962–3970.

[Forster et al., 2013b] Forster, C., Lynen, S., Kneip, L., and Scaramuzza, D. (2013b). Collaborative monocular slam with multiple micro aerial vehicles. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference*. IEEE.

[Forster et al., 2014] 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.

[Fox et al., 2006] Fox, D., Ko, J., Konolige, K., Limketkai, B., Schulz, D., and Stewart, B. (2006). Distributed multirobot exploration and mapping. *Proceedings of the IEEE*, 94(7):1325–1339.

[Fox et al., 2003] Fox, K. K. D., Limketkai, B., Ko, J., and Stewart, B. (2003). Map merging for distributed robot navigation. In *Intelligent Robots and Systems*. IEEE.

[Fujimura and Singh, 1996] Fujimura, K. and Singh, K. (1996). Planning cooperative motion for distributed mobile agents. *Journal of Robotics and Mechatronics*, 8:75–80.

[Geiger et al., 2013] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*.

[Ghedini et al., 2018] Ghedini, C., Ribeiro, C. H., and Sabattini, L. (2018). Toward efficient adaptive ad-hoc multi-robot network topologies. *Ad Hoc Networks*, 74:57–70.

[Gupta et al., 2016] Gupta, L., Jain, R., and Vaszkun, G. (2016). Survey of important issues in uav communication networks. *IEEE Communications Surveys Tutorials*, 18(2):1123–1152.

[Harms et al., 2018] Harms, H., Schmiemann, J., Schattenberg, J., and Frerichs, L. (2018). Development of an adaptable communication layer with qos capabilities for a multi-robot system. In Ollero, A., Sanfeliu, A., Montano, L., Lau, N., and Cardeira, C., editors, *ROBOT 2017: Third Iberian Robotics Conference*, pages 782–793, Cham. Springer International Publishing.

[Hayat et al., 2015] Hayat, S., Yanmaz, E., and Bettstetter, C. (2015). Experimental analysis of multipoint-to-point uav communications with ieee 802.11n and 802.11ac. In *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1991–1996.

[Hegazy et al., 2005] Hegazy, T., Ludington, B., and Vachtsevanos, G. (2005). Reconnaissance and surveillance in urban terrain with unmanned aerial vehicles. *IFAC Proceedings Volumes*, 38(1):103–108. 16th IFAC World Congress.

[Heng et al., 2015] Heng, L., Gotovos, A., Krause, A., and Pollefeys, M. (2015). Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1071–1078. IEEE.

[Higashino et al., 2016] Higashino, S., Nishi, S., and Sakamoto, R. (2016). Arttag: Aesthetic fiducial markers based on circle pairs. In *ACM SIGGRAPH 2016 Posters*, SIGGRAPH '16, pages 38:1–38:2, New York, NY, USA. ACM.

[Hornung et al., 2013] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206.

[Hougen et al., 2000] Hougen, D. F., Benjaafar, S., Bonney, J. C., Budenske, J. R., Dvorak, M., Gini, M., French, H., Krantz, D. G., Li, P. Y., Malver, F., Nelson, B., Papanikolopoulos, N., Rybski, P. E., Stoeter, S. A., Voyles, R., and Yesin, K. B. (2000). A miniature robotic system for reconnaissance and surveillance. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings*, volume 1, pages 501–507 vol.1.

[Huang et al., 2011] Huang, A. S., Bachrach, A., Henry, P., Krainin, M., Fox, D., and Roy, N. (2011). Visual odometry and mapping for autonomous flight using an rgb-d camera. In *In Proc. of the Intl. Sym. of Robot. Research*.

[Iocchi et al., 2000] Iocchi, L., Nardi, D., and Salerno, M. (2000). Reactivity and deliberation: a survey on multi-robot systems. In *Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, pages 9–32. Springer.

[Jennings et al., 1997] Jennings, J. S., Whelan, G., and Evans, W. F. (1997). Cooperative search and rescue with a team of mobile robots. In *Advanced Robotics, 1997. ICAR '97. Proceedings., 8th International Conference on*, pages 193–200.

[Jensen and Gini, 2013] Jensen, E. A. and Gini, M. L. (2013). Rolling dispersion for robot teams. In *IJCAI*. Citeseer.

[Kaess et al., 2012] Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J. J., and Dellaert, F. (2012). isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235.

[Kerl et al., 2013a] 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 et al., 2013b] Kerl, C., Sturm, J., and Cremers, D. (2013b). Robust odometry estimation for rgb-d cameras. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 3748–3754. IEEE.

[Kitano et al., 1999] Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjou, A., and Shimada, S. (1999). Robocup rescue: search and rescue in large-scale disasters as a domain for autonomous agents research. In *Systems, Man, and Cybernetics, 1999. IEEE SMC '99 Conference Proceedings. 1999 IEEE International Conference on*, volume 6, pages 739–743 vol.6.

[Klein and Murray, 2007] 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.

[Konolige et al., 2003] Konolige, K., Fox, D., Limketkai, B., Ko, J., and Stewart, B. (2003). Map merging for distributed robot navigation. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 1, pages 212–217 vol.1.

[Krajník et al., 2014] Krajník, T., Nitsche, M., Faigl, J., Vaněk, P., Saska, M., Přeučil, L., Duckett, T., and Mejail, M. (2014). A practical multirobot localization system. *Journal of Intelligent & Robotic Systems*.

[Kuhn, 1955] Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 2(1-2):83–97.

[Kulich et al., 2015] Kulich, M., Juchelka, T., and Přeučil, L. (2015). Comparison of exploration strategies for multi-robot search. *Acta Polytechnica*, 55(3):162–168.

[Kunz et al., 2008] Kunz, C., Murphy, C., Camilli, R., Singh, H., Bailey, J., Eustice, R., Jakuba, M., i. Nakamura, K., Roman, C., Sato, T., Sohn, R. A., and Willis, C. (2008). Deep sea underwater robotic exploration in the ice-covered arctic ocean with auvs. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3654–3660.

[Kuo et al., 2014] Kuo, C.-H., Kanlanjan, S., Pagčs, L., Menzel, H., Power, S., Kuo, C.-M., Boller, C., and Grondel, S. (2014). Effects of enhanced image quality in infrastructure monitoring through micro aerial vehicle stabilization. In *EWSHM-7th European Workshop on Structural Health Monitoring*.

[Kuschnig et al., 2012] Kuschnig, R., Yanmaz, E., Kofler, I., Rinner, B., and Hellwagner, H. (2012). Profiling ieee 802.11 performance on linux-based networked aerial robots. In *Proceedings of the Austrian Robotics Workshop, Graz, Austria*, page pp. 8.

[Latombe, 1991] Latombe, J.-C. (1991). Robot motion planning. Technical report, Norwell, MA, USA.

[Leigh et al., 2007] Leigh, R., Louis, S. J., and Miles, C. (2007). Using a genetic algorithm to explore a*-like pathfinding algorithms. In *2007 IEEE Symposium on Computational Intelligence and Games*, pages 72–79.

[Leutenegger et al., 2015] 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.

[Li et al., 2015] Li, D., Li, Q., Tang, L., Yang, S., Cheng, N., and Song, J. (2015). Invariant observer-based state estimation for micro-aerial vehicles in gps-denied indoor environments using an rgb-d camera and mems inertial sensors. *Micromachines*, 6(4):487–522.

[Lightbody et al., 2017] Lightbody, P., Krajník, T., and Hanheide, M. (2017). A versatile high-performance visual fiducial marker detection system with scalable identity encoding. In *Proceedings of the Symposium on Applied Computing*, pages 276–282. ACM.

[Mahdoui et al., 2017] Mahdoui, N., Fremont, V., and Natalizio, E. (2017). Cooperative exploration strategy for micro-aerial vehicles fleet. In *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI 2017)*, pages 1–6, Daegu, Korea. IEEE.

[Mahdoui et al., 2018] Mahdoui, N., Fremont, V., and Natalizio, E. (2018). Cooperative frontier-based exploration strategy for multi-robot system. In *2018 IEEE - 13th System of Systems Engineering Conference (SoSE 2018)*, pages 1–8, Paris, France. IEEE.

[Maier, 1998] Maier, M. W. (1998). Architecting principles for systems-of-systems. *Systems Engineering*, 1(4):267–284.

[Maistrenko et al., 2016] Maistrenko, V. A., Alexey, L. V., and Danil, V. A. (2016). Experimental estimate of using the ant colony optimization algorithm to solve the routing problem in fanet. In *2016 International Siberian Conference on Control and Communications (SIBCON)*, pages 1–10.

[Maity et al., 2013] Maity, A., Majumder, S., and Ray, D. N. (2013). Amphibian subterranean robot for mine exploration. In *2013 International Conference on Robotics, Biomimetics, Intelligent Computational Systems*, pages 242–246.

[Marie et al., 2014] Marie, R., Labbani-Igbida, O., and Mouaddib, E. M. (2014). Exploration autonome et cartographie topologique en environnement inconnu référencées vision omnidirectionnelle. *Traitement du Signal*, 31(1):221–243.

[Máthé and Buşoniu, 2015] Máthé, K. and Buşoniu, L. (2015). Vision and control for uavs: A survey of general methods and of inexpensive platforms for infrastructure inspection. *Sensors*, 15(7):14887–14916.

[Matignon and Simonin, 2018] Matignon, L. and Simonin, O. (2018). Multi-robot simultaneous coverage and mapping of complex scene-comparison of different strategies. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 559–567. International Foundation for Autonomous Agents and Multiagent Systems.

[Min et al., 2018] Min, B.-C., Parasuraman, R., Lee, S., Jung, J.-W., and Matson, E. T. (2018). A directional antenna based leader-follower relay system for end-to-end robot communications. *Robotics and Autonomous Systems*, 101:57 – 73.

[Mohanarajah et al., 2015] Mohanarajah, G., Usenko, V., Singh, M., D'Andrea, R., and Waibel, M. (2015). Cloud-based collaborative 3d mapping in real-time with low-cost robots. *IEEE Transactions on Automation Science and Engineering*, 12(2):423–431.

[Morgenthaler et al., 2012] Morgenthaler, S., Braun, T., Zhao, Z., Staub, T., and Anwander, M. (2012). Uavnet: A mobile wireless mesh network using unmanned aerial vehicles. *Globecom Workshops (GC Wkshps),IEEE*.

[Mur-Artal and Tardós, 2017a] Mur-Artal, R. and Tardós, J. D. (2017a). ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262.

[Mur-Artal and Tardós, 2017b] Mur-Artal, R. and Tardós, J. D. (2017b). Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803.

[Muzaffar and Yanmaz, 2014] Muzaffar, R. and Yanmaz, E. (2014). Trajectory-aware ad hoc routing protocol for micro aerial vehicle networks. In *European Conference on Networks and Communications (EuCNC 2014)*.

[Nanjanath and Gini, 2006] Nanjanath, M. and Gini, M. (2006). Dynamic task allocation for robots via auctions. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2781–2786. IEEE.

[Nielsen et al., 2015] Nielsen, C. B., Larsen, P. G., Fitzgerald, J. S., Woodcock, J., and Peleska, J. (2015). Systems of systems engineering: Basic concepts, model-based techniques, and research directions. *ACM Comput. Surv.*, 48:18:1–18:41.

[Nitsche et al., 2015] Nitsche, M., Krajník, T., Čížek, P., Mejail, M., and Duckett, T. (2015). Whycon: An efficent, marker-based localization system. In *IROS Workshop on Open Source Aerial Robotics*.

[Oh and Green, 2004] Oh, P. Y. and Green, W. E. (2004). Cqar: Closed quarter aerial robot design for reconnaissance, surveillance and target acquisition tasks in urban areas. *International Journal of Computational Intelligence*, 1(4):353–360.

[Olson, 2011] Olson, E. (2011). AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE.

[Pal et al., 2011] Pal, A., Tiwari, R., and Shukla, A. (2011). Multi robot exploration using a modified a* algorithm. In *Asian Conference on Intelligent Information and Database Systems*, pages 506–516. Springer.

[Quigley et al., 2009] Quigley, M., Conley, K., Gerkey, B. P., 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*.

[Renaudeau et al., 2018] Renaudeau, B., Labbani-Igbida, O., and Mourioux, G. (2018). Hybrid map mosaicing: A novel approach for large area mapping. In *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots, SIMPAR 2018, Brisbane, Australia, May 16-19, 2018*, pages 23–28.

[Rocha et al., 2005] Rocha, R., Dias, J., and Carvalho, A. (2005). Cooperative multi-robot systems:: A study of vision-based 3-d mapping using information theory. *Robotics and Autonomous Systems*, 53(3):282 – 311.

[Rooker and Birk, 2007] Rooker, M. N. and Birk, A. (2007). Multi-robot exploration under the constraints of wireless networking. *Control Engineering Practice*, 15(4):435–445.

[Sahl et al., 2010] Sahl, J. W., Fairfield, N., Harris, J. K., Wettergreen, D., Stone, W. C., and Spear, J. R. (2010). Novel microbial diversity retrieved by autonomous robotic exploration of the world's deepest vertical phreatic sinkhole. *Astrobiology*, 10(2):201–213.

[Scaramuzza and Fraundorfer, 2011] Scaramuzza, D. and Fraundorfer, F. (2011). Visual odometry [tutorial]. *IEEE robotics & automation magazine*, 18(4):80–92.

[Scherer et al., 2015] Scherer, J., Yahyanejad, S., Hayat, S., Yanmaz, E., Andre, T., Khan, A., Vukadinovic, V., Bettstetter, C., Hellwagner, H., and Rinner, B. (2015). An autonomous multi-uav system for search and rescue. In *Proceedings of the First Workshop on Micro Aerial Vehicle Networks, Systems, and Applications for Civilian Use*, pages 33–38. ACM.

[Schmuck, 2017] Schmuck, P. (2017). Multi-uav collaborative monocular slam. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 3863–3870. IEEE.

[Schneider et al., 2018] Schneider, T., Dymczyk, M. T., Fehr, M., Egger, K., Lynen, S., Gilitschenski, I., and Siegwart, R. (2018). maplab: An open framework for research in visual-inertial mapping and localization. *IEEE Robotics and Automation Letters*.

[Schöps et al., 2014] Schöps, T., Engel, J., and Cremers, D. (2014). Semi-dense visual odometry for ar on a smartphone. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 145–150.

[Schuster et al., 2015] Schuster, M. J., Brand, C., HirschmÃijller, H., Suppa, M., and Beetz, M. (2015). Multi-robot 6d graph slam connecting decoupled local reference filters. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5093–5100.

[Shade and Newman, 2011] Shade, R. and Newman, P. (2011). Choosing where to go: Complete 3d exploration with stereo. In *2011 IEEE International Conference on Robotics and Automation*, pages 2806–2811.

[Sheng et al., 2006] Sheng, W., Yang, Q., Tan, J., and Xi, N. (2006). Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems*, 54(12):945–955.

[Sibley et al., 2010] Sibley, G., Mei, C., Reid, I., and Newman, P. (2010). Planes, trains and automobiles – autonomy for the modern robot. In *2010 IEEE International Conference on Robotics and Automation*, pages 285–292.

[Siles and Walker, 2009] Siles, I. and Walker, I. D. (2009). Design, construction, and testing of a new class of mobile robots for cave exploration. In *2009 IEEE International Conference on Mechatronics*, pages 1–6.

[Simmons et al., 2000] Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S., and Younes, H. (2000). Coordination for multi-robot exploration and mapping. In *AAAI/IAAI*, pages 852–858.

[Simonin et al., 2014] Simonin, O., Charpillet, F., and Thierry, E. (2014). Revisiting wavefront construction with collective agents: an approach to foraging. *Swarm Intelligence*, 8:113–138.

[Solanas and Garcia, 2004] Solanas, A. and Garcia, M. A. (2004). Coordinated multi-robot exploration through unsupervised clustering of unknown space. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 717–721. IEEE.

[Spaenlehauer et al., 2017] Spaenlehauer, A., Frémont, V., Sekercioglu, Y. A., and Fantoni, I. (2017). A loosely-coupled approach for metric scale estimation in monocular vision-inertial systems.

[Steinbrücker et al., 2011] Steinbrücker, F., Sturm, J., and Cremers, D. (2011). Real-time visual odometry from dense rgb-d images. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 719–722. IEEE.

[Stone and Edmonds, 1992] Stone, H. W. and Edmonds, G. (1992). Hazbot: a hazardous materials emergency response mobile robot. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 67–73.

[Sturm et al., 2012] Sturm, J., Engelhard, N., Endres, F., Burgard, W., and Cremers, D. (2012). A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*.

[Ta et al., 2013] Ta, D.-N., Ok, K., and Dellaert, F. (2013). Monocular parallel tracking and mapping with odometry fusion for mav navigation in feature-lacking environments.

[Tomic et al., 2012] Tomic, T., Schmid, K., Lutz, P., Domel, A., Kassecker, M., Mair, E., Grixa, I. L., Ruess, F., Suppa, M., and Burschka, D. (2012). Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue. *IEEE Robotics Automation Magazine*, 19(3):46–56.

[Vidal et al., 2015] Vidal, I., Valera, F., Diaz, M. A., Garcia, J., and Azcorra, A. (2015). A multi-service multi-mav communication framework for future secure societies. Technical report, University Carlos III of Madrid and IMDEA Networks Institute.

[Waharte and Trigoni, 2010] Waharte, S. and Trigoni, N. (2010). Supporting search and rescue operations with uavs. In *2010 International Conference on Emerging Security Technologies*, pages 142–147.

[Waharte et al., 2009] Waharte, S., Trigoni, N., and Julier, S. J. (2009). Coordinated search with a swarm of uavs. *Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2009. SECON Workshops '09. 6th Annual IEEE Communications Society Conference on*.

[Wang et al., 2010] Wang, J. C. P., Hagelstein, B., and Abolhasan, M. (2010). Experimental evaluation of ieee 802.11s path selection protocols in a mesh testbed. In *2010 4th International Conference on Signal Processing and Communication Systems*, pages 1–3.

[Wang et al., 2014] Wang, Q., Giustiniano, D., and Puccinelli, D. (2014). Openvlc: Software-defined visible light embedded networks. In *Proceedings of the 1st ACM MobiCom Workshop on Visible Light Communication Systems*, VLCS '14, pages 15–20, New York, NY, USA. ACM.

[Wang et al., 2017] Wang, X., Sekercioglu, Y. A., Drummond, T., Frémont, V., Natalizio, E., and Fantoni, I. (2017). Relative pose based redundancy removal: Collaborative RGB-D data transmission in mobile visual sensor networks. *CoRR*.

[Werger and Matarić, 2000] Werger, B. B. and Matarić, M. J. (2000). Broadcast of local eligibility for multi-target observation. In *Distributed autonomous robotic systems 4*, pages 347–356. Springer.

[Whelan et al., 2015] Whelan, T., Kaess, M., Johannsson, H., Fallon, M., Leonard, J. J., and McDonald, J. (2015). Real-time large-scale dense rgb-d slam with volumetric fusion. *The International Journal of Robotics Research*, 34(4-5):598–626.

[Whelan et al., 2016] Whelan, T., Salas-Moreno, R. F., Glocker, B., Davison, A. J., and Leutenegger, S. (2016). Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716.

[Whitcomb et al., 2000] Whitcomb, L., Yoerger, D. R., Singh, H., and Howland, J. (2000). Advances in underwater robot vehicles for deep ocean exploration: Navigation, control, and survey operations. In Hollerbach, J. M. and Koditschek, D. E., editors, *Robotics Research*, pages 439–448, London. Springer London.

[Witkowski et al., 2008] Witkowski, U., El Habbal, M. A. M., Herbrechtsmeier, S., Tanoto, A., Penders, J., Alboul, L., and Gazi, V. (2008). Ad-hoc Network Communication Infrastructure for Multi-robot Systems in Disaster Scenarios. In *Proceedings of IARP/EURON Workshop on Robotics for Risky Interventions and Environmental Surveillance (RISE 2008), Benicassim, Spain.*

[Wu and Zhang, 2012] Wu, W. and Zhang, F. (2012). Robust cooperative exploration with a switching strategy. *IEEE Transactions on Robotics*, 28(4):828–839.

[Yamauchi, 1998] Yamauchi, B. (1998). Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53. ACM.

[Yan et al., 2014] Yan, Z., Fabresse, L., Laval, J., and Bouraqadi, N. (2014). Team size optimization for multi-robot exploration. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 438–449. Springer.

[Yan et al., 2013] Yan, Z., Jouandeau, N., and Cherif, A. A. (2013). A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12):399.

[Yang et al., 2017] Yang, S., Scherer, S. A., Yi, X., and Zell, A. (2017). Multi-camera visual slam for autonomous navigation of micro aerial vehicles. *Robotics and Autonomous Systems*, 93:116–134.

[Yuan et al., 2010] Yuan, J., Huang, Y., Tao, T., and Sun, F. (2010). A cooperative approach for multi-robot area exploration. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1390–1395. IEEE.

[Zhang et al., 2014] Zhang, J., Kaess, M., and Singh, S. (2014). Real-time depth enhanced monocular odometry. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4973–4980. IEEE.

[Zhao et al., 1996] Zhao, L., Tsujimura, Y., and Gen, M. (1996). Genetic algorithm for robot selection and work station assignment problem. *Computers & Industrial Engineering*, 31(3-4):599–602.

[Ärlker Bekmezci et al., 2013] Ärlker Bekmezci, Sahingoz, O. K., and Åđamil Temel (2013). Flying ad-hoc networks (fanets): A survey. *Ad Hoc Networks*, 11(3):1254–1270.