UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MAIK BASSO

A COOPERATIVE NAVIGATION SYSTEM WITH DISTRIBUTED ARCHITECTURE FOR MULTIPLE UNMANNED AERIAL VEHICLES

Porto Alegre
2022

MAIK BASSO

A COOPERATIVE NAVIGATION SYSTEM WITH DISTRIBUTED ARCHITECTURE FOR MULTIPLE UNMANNED AERIAL VEHICLES

Thesis presented to Programa de Pós-Graduação em Engenharia Elétrica of Universidade Federal do Rio Grande do Sul in partial fulfillment of the requirements for the degree of Doctor in Electrical Engineering.
Area: Control and Automation

ADVISOR: Prof. Dr. Edison Pignaton de Freitas

Porto Alegre
2022

MAIK BASSO

A COOPERATIVE NAVIGATION SYSTEM WITH DISTRIBUTED ARCHITECTURE FOR MULTIPLE UNMANNED AERIAL VEHICLES

This thesis was considered adequate for the awarding of the degree of Doctor in Electrical Engineering and approved in its final form by the Advisor and the Examination Committee.

Advisor: _____
Prof. Dr. Edison Pignaton de Freitas, UFRGS
PhD from Halmstad University, Sweden and Federal University of Rio Grande do Sul, Brazil

Examination Committee:

Prof. Dr. Paulo Fernando Ferreira Rosa, IME
PhD from Niigata University, NIIDAI, Japan

Prof. Dr. Edson Prestes e Silva Junior, UFRGS
PhD from Federal University of Rio Grande do Sul, Brazil

Prof. Dr. Carlos Eduardo Pereira, UFRGS
PhD from Technische Universitat Stuttgart, Germany

Coordinator of PPGEE: _____
Prof. Dr. Sérgio Luís Haffner

Porto Alegre, March 2022.

DEDICATÓRIA

Dedico esta tese em memória ao meu querido avô Adelino José Basso, cuja com o qual eu tive a oportunidade de conviver, aprender e me inspirar. Vô Adelino, você sempre estará vivo em minhas memórias!

Dedico esta tese em memória ao meu querido avô Alceu Cadoná, que apesar de ter convivido pouco tempo comigo, sempre me alegrava e me divertia. Vô Alceu, tenho certeza que deve estar muito feliz com todas as nossas conquistas.

AGRADECIMENTOS

Aos meus pais Devanir Giovani Basso e Eliane Marisa Cadoná Basso, que apesar de todas as dificuldades estiveram sempre ao meu lado em todas as minhas escolhas, me orientando e me motivando sempre com muito amor, carinho e compreensão.

Ao meu irmão Gabriel Cadoná Basso, ao qual consegue mesmo em tempos de dificuldade ter sempre um sorriso enorme no rosto, me motivando e me alegrando.

As minhas avós Ondina Piaia Basso e Delmiria Vanin Cadoná que sempre me deram o apoio necessário para que eu conseguisse enfrentar os momentos difíceis.

À minha namorada Jéssica Andressa Kloster, por estar sempre ao meu lado, me apoiando e acima de tudo compartilhando das angústias e alegrias que esta jornada proporcionou nas nossas vidas.

Aos amigos e colegas, Carlos Felipe Emydgio de Melo, Túlio Dapper e Silva, Marcos Rodrigues Vizzotto, Mateus Schein Cavalheiro Corrêa, David Rutherford Armstrong, Lucas Bortolanza Grazziotim, Tiago Giacomelli Alves, Diego Alvim Stocchero, Pedro Schnarndorf e Luíza Caetano Garaffa que participaram de partes desta tese, seja com ideias ou me ajudando e dando apoio para que eu conseguisse superar todos os desafios.

Ao Programa de Pós-Graduação em Engenharia Elétrica (PPGEE), por me conceder a oportunidade de realização desta pesquisa, e em especial à Miriam Rosek, pela disponibilidade em me atender e me ajudar com a parte burocrática sempre que necessário.

Ao meu professor e orientador Edison Pignaton de Freitas, pelo tempo que dedicaste a me orientar nesta tese, por me entusiasmar e não me deixar desistir dos objetivos mesmo nos momentos mais complicados. Saiba professor que serei sempre grato pelas oportunidades que colocastes na minha vida, por nossa amizade. Muito obrigado.

Ao professor e amigo Renato Ventura Bayan Henriques por possibilitar a realização dos estágios docência na graduação.

À CAPES pela concessão de bolsa para a minha manutenção durante o período de dedicação exclusiva a pesquisa contida nesta tese.

A todas as pessoas que se fizeram presentes nessa etapa da minha vida me apoiando para que a conclusão deste curso fosse possível.

Por fim, agradeço à Deus, por estar sempre ao meu lado me guiando e me protegendo em todos os momentos.

''Eu quero, eu sei, eu posso e eu consigo!''

Eliane Marisa Cadoná Basso

ABSTRACT

Unmanned aerial vehicles (UAVs) have been widely used in many applications due to, among other features, their versatility, reduced operating cost, and small size. These applications increasingly demand that features related to autonomous navigation be employed, such as mapping. However, the reduced capacity of resources such as, for example, battery and hardware (memory and processing units) can hinder the development of these applications in UAVs. Thus, the collaborative use of multiple UAVs for mapping can be used as an alternative to solve this problem, with a cooperative navigation system. This system requires that individual local maps be transmitted and merged into a global map in a distributed manner. In this scenario, there are two main problems to be addressed: the transmission of maps among the UAVs and the merging of the local maps in each UAV. In this context, this work describes the design, development, and evaluation of a cooperative navigation system with distributed architecture to be used by multiple UAVs. This system uses proposed structures to store the 3D occupancy grid maps. Furthermore, maps are compressed and transmitted between UAVs using algorithms specially proposed for these purposes. Then the local 3D maps are merged in each UAV. In this map merging system, maps are processed before and merged in pairs using suitable algorithms to make them compatible with the 3D occupancy grid map data. In addition, keypoints orientation properties are obtained from potential field gradients. Some proposed filters are used to improve the parameters of the transformations among maps. To validate the proposed solution, simulations were performed in six different environments, outdoors and indoors, and with different layout characteristics. The obtained results demonstrate the effectiveness of the system in the construction, sharing, and merging of maps. Still, from the obtained results, the extreme complexity of map merging systems is highlighted.

Keywords: Multi-UAVs Applications, Cooperative Navigation Systems, 3D Occupancy Grid Maps, 3D Mapping, Map Sharing, 3D Map Merging.

RESUMO

Os veículos aéreos não tripulados (VANTs) têm sido amplamente utilizados em muitas aplicações devido, entre outros recursos, à sua versatilidade, custo de operação e tamanho reduzidos. Essas aplicações exigem cada vez mais que recursos relacionados à navegação autônoma sejam empregados, como o mapeamento. No entanto, a capacidade reduzida de recursos como, por exemplo, bateria e hardware (memória e capacidade de processamento) podem atrapalhar o desenvolvimento dessas aplicações em VANTs. Assim, o uso colaborativo de múltiplos VANTs para mapeamento pode ser utilizado como uma alternativa para resolver este problema, criando um sistema de navegação cooperativo. Este sistema requer que mapas locais individuais sejam transmitidos e fundidos em um mapa global de forma distribuída. Nesse cenário, há dois problemas principais a serem abordados: a transmissão dos mapas entre os VANTs e a fusão dos mapas locais em cada VANT. Neste contexto, esta tese apresenta o projeto, desenvolvimento e avaliação de um sistema de navegação cooperativo com arquitetura distribuída para ser utilizado por múltiplos VANTs. Este sistema usa estruturas propostas para armazenar os mapas de grade de ocupação 3D. Além disso, os mapas são compactados e transmitidos entre os VANTs usando os algoritmos propostos. Em seguida, os mapas 3D locais são fundidos em cada VANT. Neste sistema de fusão de mapas, os mapas são processados antes e juntados em pares usando alguns algoritmos adequados para torná-los compatíveis com os dados dos mapas da grade de ocupação 3D. Além disso, as propriedades de orientação dos pontos-chave são obtidas a partir de gradientes de campos potenciais. Alguns filtros propostos são utilizados para melhorar as indicações dos parâmetros das transformações entre mapas. Para validar a aplicação proposta, foram realizadas simulações em seis ambientes distintos, externos e internos, e com características construtivas distintas. Os resultados apresentados demonstram a efetividade do sistema na construção, compartilhamento e fusão dos mapas. Ainda, a partir dos resultados obtidos, destaca-se a extrema complexidade dos sistemas de fusão de mapas.

Palavras-chave: Aplicações com Múltiplos VANTs, Sistemas de Navegação Cooperativos, Mapas de Grade de Ocupação 3D, Mapeamento 3D, Compartilhamento de Mapas, Fusão de Mapas 3D.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1D	One-dimensional
2D	Two-dimensional
3D	Three-dimensional
6D	Six-dimensional
AUV	Autonomous Underwater Vehicle
BRIEF	Binary Robust Independent Elementary Features
CG	Center of Gravity
CSV	Comma-Separated Values
DoF	Degrees-of-Freedom
DSM	Data Sharing Module
EKF	Extended Kalman Filter
ESC	Electronic Speed Controller
FAST	Features from accelerated segment test
FCU	Flight Control Unit
GB	Gigabyte
GHz	Gigahertz
GNSS	Global Navigation Satellite System
GPS	Global Positioning Systems
HIMM	Histogramic In Motion Mapping
ID	Identifier
IMU	Inertial Measurement Unit
IP	Internet Protocol
KF	Kalman Filter
LIDAR	Light Detection and Ranging
LTS	Long-Term Support
MAS	Multi-Agent System

MB	Megabyte
MM	Mapping Module
MMM	Map Merging Module
MTM	Map Transformation Matrix
MTP	Map Transmission Protocol
ORB	Oriented FAST and Rotated BRIEF
PSO	Particle Swarm Optimization
RAM	Random Access Memory
RANSAC	Random Sample Consensus
rFAST	Rotated features from accelerated segment test
RGB	Red, Green, and Blue
RGB-D	Red, Green, Blue, and Depth
RL	Reinforcement Learning
ROS	Robot Operating System
RPLIDAR	RoboPeak Light Detection and Ranging
SIFT	Scale Invariant Feature Transform
SITL	Software In The Loop
SLAM	Simultaneous Localization and Mapping
SSD	Solid-State Drive
SURF	Speeded Up Robust Features
SVD	Singular Value Decomposition
TCP	Transmission Control Protocol
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle

CONTENTS

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are mobile robots that can be remotely guided or can operate with varying degrees of autonomy. These vehicles can be disposable, when they are designed to perform a non-return mission, such as battle vehicles, for example, or reusable, such as those used in monitoring operations (NONAMI *et al.*, 2013).

Economically, UAVs represent a great opportunity for the market of equipment manufacturers, investors, and service providers, among other businesses. According to a report of a study of commercial applications of UAV technology made available by PWC (2016), the emerging global market for services and businesses using UAVs is valued at over $ 127 billion, being distributed across industries from agriculture to the film industries.

Thus, the use of UAVs is increasing due to the different desirable characteristics of these platforms, such as versatility to be used in different applications, reduced cost and embedded intelligence (AL-KAFF *et al.*, 2018). They are used in different applications in both the civilian (BASSO; DE FREITAS, 2020) and the military domain (ORFANUS; DE FREITAS; ELIASSEN, 2016). In particular, the use of multiple UAV systems is gaining increasing attention due to their ability to scale in challenging scenarios, such as law enforcement support in urban areas (DE MORAES; DE FREITAS, 2020).

These applications increasingly demand that intelligent features related to navigation autonomy be employed to facilitate mission control and even support autonomous navigation of these vehicles (BASSO; DE FREITAS, 2020; VISION IN INDOOR AND OUTDOOR DRONES, 2020). One of the most used practices to aid autonomous navigation is the construction of virtual representations in the environment, known in mobile robotics as occupancy grid maps (Birk; Carpin, 2006).

However, the reduced capacity of features such as battery lifetime and hardware (memory and processing capacity) can be an obstacle for those applications that involve UAVs in complex environments of considerable size. Thus, the collaborative use of multiple UAVs can be an alternative to face the challenge imposed by these conditions (ZHAO; LI; ZHANG, 2017; HAN *et al.*, 2016), creating a cooperative navigation system. In these systems, the mapping task is spread across multiple UAVs, making the application more robust and reducing the time needed to build a map compared to single UAV systems.

In this way, each UAV component of the cooperative navigation system builds its virtual representations in three dimensions and shares them with the UAVs within its communication range. Then, each UAV that receives the environment virtual representation of other UAVs performs the map merge in its embedded system.

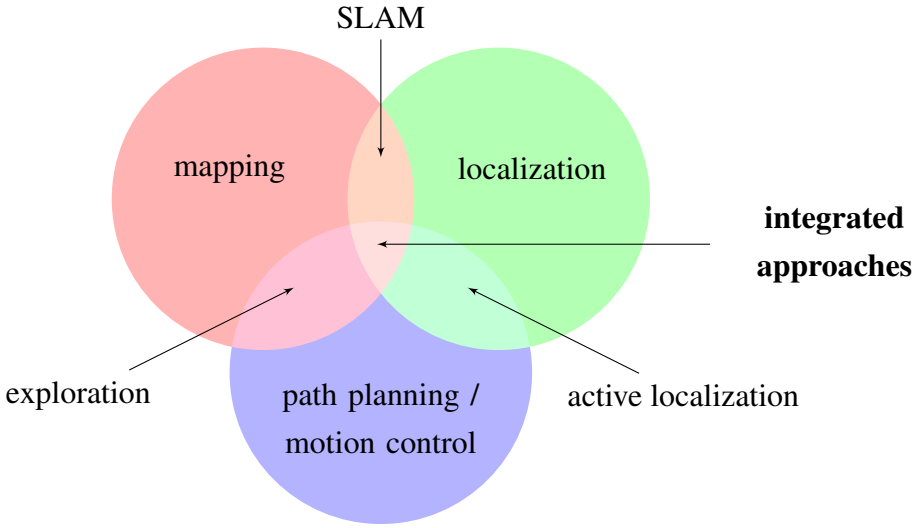
In this sense, it is observed that the map merging is carried out in a distributed way because, in addition to sharing the local maps, these vehicles share the parameters of the previous merged maps already computed, avoiding that the complete merging process has to be performed by the others UAVs of the network, distributing required battery consumption and processing load.

In mobile robotics, the activity of building maps is covered by the study area known as mapping. As can be seen in Figure 1. Furthermore, the mobile robots carrying out the construction of maps need to self-localize and plan the future trajectory. These other activities encompass the two other areas of study known as localization and path planning or motion control.

In the scenario presented above, there are two problems which are: the transmission of information between the UAVs and the merging of local maps in each vehicle. In mobile robotics, these problems are popularly known as integrated approaches (Makarenko *et al.*, 2002), as demonstrated in Figure 1. These problems are so-called because they integrate different research areas within mobile robotics.

The information transmission problem can be addressed by a communication problem. Here, there is a large set of information to be completely transmitted in a short period. The map merging problem, on the other hand, is addressed as a computer vision problem. Here, the visual information must be used to fuse the information in the same data set. This thesis addresses these two problems by proposing adequate solutions for both.

Figure 1 – The different fields of study in mobile robotics (localization, mapping and motion control) and the problems originated by their overlapping areas.



Source: Adapted from Makarenko *et al.* (2002).

The first step in solving these problems is to select the type of structure suitable for storing the information gathered from the environment. The 3D occupancy grid maps represent the environment by storing the probability of occupancy of each portion of the real environment space captured by the sensors. Each of these parts of the environment is called a cell, and the size of each cell is determined by the resolution of the map.

The use of 3D occupancy grid maps allows adequate mobility of UAVs in three dimensions and better operation of exploration systems, in addition to a more faithful representation of the environment and better correction of estimates from the mapping systems (SCHMUCK; SCHERER; ZELL, 2016).

The map transmission problem is addressed as a communication problem that deals with data compression, formatting, and correct distribution using and high-level protocol.

However, the problem of merging 3D occupancy grid maps becomes more complex due to the type of data, volume of information, and noise that can be accidentally generated from sensor data or by mapping algorithms. Also, conventional occupancy grid map merge techniques cannot be applied as these algorithms are generally designed to process images or data in two dimensions.

Generally, the map merging problem is associated with a highly complex system with high computational cost, and great research efforts are carried out to make these systems operate in real-time (Velásquez Hernández; Prieto Ortiz, 2020). This problem can be divided into two basic steps: map matching and map merging (Yue *et al.*, 2018). The map matching step is responsible for identifying similar regions contained in local maps, and obtaining matching points. The map merging step is responsible for using these combinations to build transformations between the maps used to perform the merge. Also at this stage, the information from the cells between the local maps is merged taking into account predetermined rules for each of the states assumed in the occupation scale.

In this context, this thesis proposes a cooperative navigation system with distributed architecture to be used by multiple UAVs. This system solution proposes structures to store the 3D maps. Furthermore, maps are compressed and transmitted between UAVs using algorithms designed for this purpose. Then local 3D maps are merged into each UAV. In this map merging system, maps are firstly processed, then merged in pairs using suitable algorithms to make them compatible with the data of the 3D occupancy grid maps. In addition, keypoint orientation properties are obtained from potential field gradients. Even so, some proposed filters are used to improve the indications of the parameters used in the transformations between maps. To validate the proposed solution, simulations were carried out in six distinct environments, outdoors and indoors, and with different layout characteristics.

In the proposed application and state-of-the-art map merging applications, it is necessary a minimum overlap region among the maps before the merge process. In this thesis, the minimum overlap region is defined as a minimum set of five real matches identified

between the maps. No matter the size of the area in which these correspondences are found on the maps.

Another important point about the solution presented in this thesis is the proposed architecture considers that the processing load is allocated in a distributed way on the embedded hardware of the UAVs. Also, these UAVs can be heterogeneous, having different physical capabilities. Still, this thesis does not take into account the utilization of a ground station to extend the processing capabilities. This definition aims to make the system independent of modifications in the environment, in which UAVs can be completely spread to unknown and unexplored locations. Another justification is that these sites may be inaccessible by ground.

The energy consumption of the proposed application will not be evaluated in this thesis because it is dependent on the software, the hardware, the structure of the vehicles, the environment, and the climatic conditions of the ecosystem in which the application will be running. The UAV navigation speed is another point that can influence the updating and construction of occupancy grid maps. The higher speeds may require faster processing, but similarly to energy consumption, the evaluation depends on many factors specific to each application's scenario. The evaluation of battery consumption and UAV speed are considered outside of the thesis scope.

Section 1.1 describes in a simplified way the hypotheses underlying the research around this thesis. Section 1.2 shows the goals for this research. The most significant contributions of this thesis are presented in Section 1.3. Section 1.4 then describes the organization of this document.

1.1 Hypotheses

This thesis is based on three basic research hypotheses, which are:

a) It is possible to design a cooperative navigation system for multiple UAVs based on the construction of 3D occupancy grid maps;

b) It is possible to share these maps with UAVs fast enough to support a cooperative work among UAVs and without consuming too many resources;

c) It is possible to merge the 3D occupancy grid maps in each UAV.

1.2 Goals

The main goal of this thesis is to complete the design, implementation, and validation of a cooperative navigation system for multiple UAVs with a distributed architecture. Besides the main goal, it is possible to state the following specific goals:

- Develop the construction of 3D local occupancy grid maps;

- Develop the local maps sharing;
- Develo the merging process of the local maps obtained by each UAV in runtime;

1.3 Contributions

The most significant contributions of this thesis are:

- The definition of dynamic 3D occupancy grid map structures;
- The map compression and decompression method;
- The map serialization protocol;
- The map transmission protocol (MTP);
- The extension of the legacy 2D FAST algorithm (ROSTEN; DRUMMOND, 2006), for detecting keypoints in 3D occupancy grid maps. This thesis proposes the 3D version of the FAST algorithm;
- The proposal to use potential field gradients to obtain the orientation of keypoints;
- Adaptation of the BRIEF algorithm (CALONDER *et al.*, 2010), to describe keypoints taking into account the characteristics of 3D occupancy grid maps;
- The modeling and proposal of a 3D correspondence filter, in 3D occupancy grid maps, based on RANSAC and Homography concepts;
- Calculation of map transformation parameters taking into account the scale used in the downsample process of the map preprocessing stage;
- Experiments based on simulations performed with multiple UAVs operating in a distributed and cooperative scenario;

1.4 Thesis Organization

This thesis is organized as follows. First, Chapter 2 presents all the theoretical foundations involving navigation systems for mobile robots, computer vision techniques used, concepts of cooperative systems for multiple UAVs, and tools used for implementations proposed in this thesis. The application scenario as well as the problems and difficulties identified were detailed and explained in Chapter 3. Then, Chapter 4 presents a study on related works, as well as a discussion and comparison with the present thesis. Chapter 5 presents the architecture of the navigation system proposed in this thesis and a brief description of the functioning of each of the proposed software modules. Chapters 6 to 8 describe each of the previously presented modules in detail. The implementation details and description of the experiments are described in Chapter 9. Chapter 10 presents the results of the proposed experiments, as well as an analysis and discussion of them. Finally, the conclusions and indication of possible future works are presented in Chapter 11.

2 BACKGROUND CONCEPTS REVIEW

This chapter detail the concepts and techniques used throughout this thesis, providing theoretical background. First, in Section 2.1, the main concepts around the navigation systems are presented. Section 2.2 describes the main computer vision algorithms theory used to develop this thesis. Section 2.3 discuss the main characteristics and problems around the multiple UAVs cooperative systems. Finally, Section 2.4 explain the implementation tools used to develop this thesis.

2.1 Navigation Systems

Taking into account the original meaning of the word, the term navigation is applied to the activity related to driving a ship to its destination. This process consists of three steps performed sequentially: (a) determine the position of the ship on a chart as accurately as possible; (b) relate your position to the destination, landmarks, and possible hazards; (c) define the new course of the ship based on this information (FRANZ; MALLOT, 2000).

Levitt and Lawton (1990) define the navigation as a process that answers the following three questions: (a) "Where am I?"; (b) "Where are other places with me?"; (c) "How do I get to other places here?". This context relates to the ability of the navigator to self-locate, know, interact and move through the environment in which it is inserted. This environment is, in many cases, unknown to the robot.

In mobile robotics, navigation can be described as the process of determining a suitable and safe path between a starting point and a goal for a robot traveling among them (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011; GALLISTEL, 1990). Navigation is a basic skill for autonomous robots (KRUSE *et al.*, 2013).

Systems that use computer vision for navigation can be divided into those that need prior knowledge (map, obstacles positions, free space) of the entire environment to which they navigate and those that perceive the environment as they navigate through it (BONIN-FONT; ORTIZ; OLIVER, 2008).

A navigation system for autonomous robots is composed of a vision system, which from the sensors' inputs of the robot is constructed a virtual representation of the environ-

ment. A system capable of generating the trajectory based on this virtual representation was created.

Also, navigation systems must-have modules that avoid collision with obstacles, using robust navigation strategies such as the minimum distance required to avoid obstacles or stopping and waiting for behavior in conflict situations to ensure safety. The performance of a good navigation algorithm is deeply associated with the precise estimated location of the robot in the environment.

The type of vehicle that uses the navigation system can also affect performance, either by limiting the hardware it can carry or by directly affecting its processing capacity, such as a UAV navigation system. The UAVs can move in 3D space, they do not have the limitations of ground robots, which often cannot overcome rocks, climb stairs, or gain access to high places. However, navigation systems designed especially for UAVs have their complexity increased by the addition of the third axis.

This section describes the main components and techniques related to and applied to mobile robot navigation.

2.1.1 State Representation

To perform missions in real or simulated environments, autonomous UAVs collect data from multidimensional sensors strategically positioned around their chassis. From these data, these systems must be able to construct a virtual representation of the environment in which it is inserted. From this representation should it have the ability to navigate autonomously, locating, and avoiding possible obstacles.

The construction of these virtual representations is based on the accumulation of all possible states of the vehicle along its trajectory.

Table 1 presents the variables used in the state representation commonly used in mobile robotics and which will be used in this thesis to introduce the concepts around navigation systems (STACHNISS; LEONARD; THRUN, 2016; THRUN *et al.*, 2005).

2.1.2 Mapping

One of the most common types of navigation systems used in mobile robots is those that do not require prior knowledge of the entire environment to navigate. One of the strengths of these systems is that the environment does not need to be changed, meaning that cameras and other marks in the environment are not required. This type of system is specially designed to operate in regions unknown to the robot.

Autonomous mobile robots with systems of this category, through the acquisition and memorization of the experiences obtained along their path, need to maintain a model of the environment in which they are inserted. This virtual environment model is known as a map.

A map should contain representative information of the entire environment already

Table 1 – List of symbols used for state representation.

Symbol	Meaning
$\boldsymbol{x}_t$	Robot pose at the instant t. The pose vector is represented by the three dimensional position and the three dimensional orientation. The robot pose is given by $\boldsymbol{x}_t = \{x, y, z, \alpha, \beta, \theta\}$.
$\boldsymbol{x}_{0:t}$	Represents the complete trajectory of the robot, and is given by $\boldsymbol{x}_{0:t} = \{\boldsymbol{x}_0, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_t\}$.
$\boldsymbol{u}_t$	The command vector applied to move the robot pose $\boldsymbol{x}_{t-1}$ to $\boldsymbol{x}_t$.
$\boldsymbol{u}_{0:t}$	Represents the history of all command vectors and is given by $\boldsymbol{u}_{0:t} = \{\boldsymbol{u}_0, \boldsymbol{u}_1, \cdots, \boldsymbol{u}_t\}$.
$\boldsymbol{z}_t$	Vector of measurements acquired by the robot at instant t and is given by $\boldsymbol{z}_t = \{\boldsymbol{z}_t^1, \boldsymbol{z}_t^2, \cdots, \boldsymbol{z}_t^i\}$, where $\boldsymbol{z}_t^i$ is the i-th observation at instant t.
$\boldsymbol{z}_{0:t}$	The set of measurements acquired during the trajectory is given by $\boldsymbol{z}_t = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_t\}$.
$\boldsymbol{m}_t^i$	The local map of robot i at the instant t.
$\boldsymbol{W}_t$	The global map at the instant t.

navigated, such as free space available for navigation, obstacles to be avoided, and also borders with regions that have not yet been explored. Also, some map representations may contain information about points or objects of interest to the system.

The mapping process consists of the construction of the virtual representation of the environment, that is, of the map $\boldsymbol{m}$, based on the acquisition of the correct estimation of the robot's $\boldsymbol{x}_t$ pose by the odometer system and the readings $\boldsymbol{z}_t$ of all sensors captured at the same time t. Thus, a map $\boldsymbol{m}$ can be described basically as the correct representation of all measurements $\boldsymbol{z}_{0:t}$ of the sensors of the robot during its trajectory $\boldsymbol{x}_{0:t}$.

One of the most common problems that can affect proper map construction is due to noise captured by sensors. Noise can insert obstacles, close narrow passages and completely deform the map obtained. Thus, the mapping technique adopted must be adequate to the environmental requirements and the needs and limitations that the robot has. The mapping process is a primary activity for these navigation systems, if it is performed incorrectly, the entire system becomes flawed.

Map representations can be made in two-dimensional (2D) space, suitable for ground robots' map representation, and also in three-dimensional (3D) representations, suitable for aquatic and aerial robots, such as UAVs, which do not have the same movement restrictions as terrestrial robots.

Map representations can be divided into two different proposals, either metric or topological (THRUN *et al.*, 2002). Metric approaches seek to capture the geometric properties of the environment for map construction. A classic example of this type of approach is

the occupancy grid mapping technique (Elfes; Matthies, 1987). An example of this type of map can be observed in Figure 2. In this type of map, each cell represents the state of a small part of the environment, whose size depends on the scale being computed. The possible states for each cell are unexplored, free space, and occupied space. This type of representation can consume memory in proportion to the size of the environment represented. On the other hand, the maps can be very detailed allowing the robot to navigate as much free space as possible.

Figure 2 – Occupancy grid map example. White cells represent all free space. The black cells represent the obstacles. Finally, the gray cells represent the unknown regions. All sets of gray cells completely surrounded by obstacles are treated as non-accessible regions.

Source: HOWARD; PARKER; SUKHATME (2006).

On the other hand, topological maps describe the environment through a list of keypoints (places) connected one to the other through arcs. They consist of maps with graph representations. Keypoints describe specific features of a particular place, these characteristics are specificities that can differentiate one place from another. The arcs determine how free space connects between these places, thus allowing the robot to navigate from one environment to another. Figure 3 shows an example of topological maps.

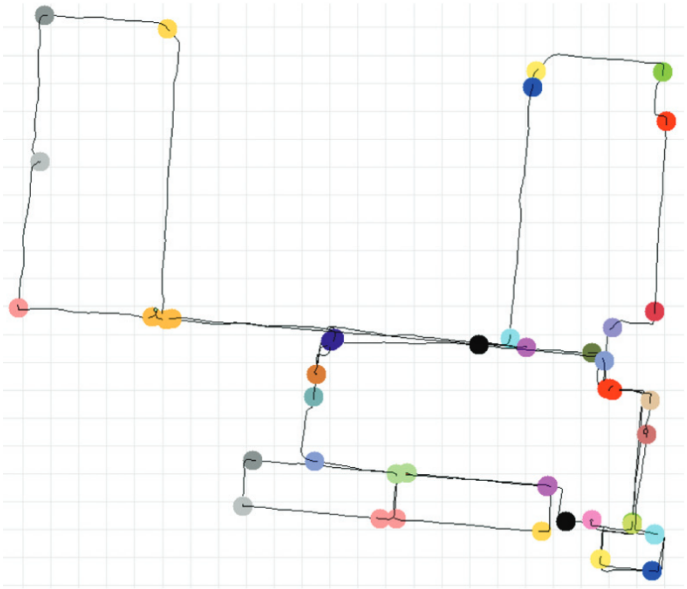
Graph maps take up less memory and may facilitate later path planning, while only a path between two points is known to limit the robot's navigation through this free space.

2.1.3 Localization

For an autonomous robot to be able to move around the environment, it must know its pose. Knowing its location in the environment in which it is inserted, the robot can move around avoiding obstacles, and interacting with the environment while performing the task for which it was designed.

According to Drumheller (1987), localization can be treated as a process directly related to measuring the robot's position relative to the environment. For Stachniss (2006)

Figure 3 – Topological map example.



Source: RANGANATHAN; DELLAERT (2011).

localization is the problem of estimating the robot's pose $\boldsymbol{x}_t$ at the time t relative to a map $\boldsymbol{m}$. To perform this estimation process, the robot's sensor reading history $\boldsymbol{z}_{0:t}$ is used as well as its pose $\boldsymbol{x}_t$ provided by an odometer when available. Algorithms aimed for estimating robot localization can also use as inputs the history of control commands $\boldsymbol{u}_{0:t}$ that were used for moving the robot (LEONARD; DURRANT-WHYTE, 1991). All this data is integrated into the function of time to estimate the correct robot pose.

According to Thrun et al. (2005), the localization problem can be simplified to a transformation problem among coordinate systems. In this problem, the map can be described as a global coordinate system independent of the robot coordinate system. Thus, localization is the transformation determination process that matches the map coordinate system with the robot coordinate system. The term localization may assume variations linked to the specifics of the application to which this system is being applied.

The process of localization of mobile robots themselves can be classified into two categories, local localization, and global localization (THRUN *et al.*, 2005). In the global localization problem, the robot pose is unknown. In this process, it is needed to calculate the position of the robot to the map representation or the real world, also known as the absolute pose. The global localization process is considered a problem with higher difficulty since the robot can be anywhere in the space of the environment. Moreover, in this case, it is not possible to assume the problem as an uncertain local positioning.

The second approach is local localization, where the robot pose is known as a priori. In this type of approach, the goal is to correctly estimate the robot pose in the robot coordinate space, also known as the relative pose. However, this approach is more simple than global localization because it assumes that the uncertainty about the robot pose is

characterized by small errors around the true local pose.

Thus, as in the case of mapping, sensor noise can strongly affect localization systems, which in turn are dependent on the conditions imposed by the environment in which the robot is inserted. In this context, environments are usually classified into two distinct classes, namely static and dynamic environments. In a virtual representation of static environments objects always remain in the same position, decreasing noise and extracting information tends to be less uncertain.

Dynamic environments, in turn, have moving objects and structures, such as doors, tables, and chairs. Depending on the time the robot comes to visit these structures, it may close paths or open previously non-existent passages. This process can difficult the operation of localization estimation systems.

2.1.4 Exploration

Every autonomous mobile robot must have the ability to make a decision and also undertake some planning steps as to which path to follow. This planning step is often strongly associated with the activity that the robot is being designed to perform. For example, mobile robots intended to execute the exploration activity must have the ability to choose the best path to achieve their goal. Also, in conjunction with this activity, it should perform as much area coverage as possible to minimize the unexplored area. Also, they must ensure that saving scarce resources, such as battery or fuel charge, are avoided, and avoid collision with obstacles.

In summary, exploration can be defined as moving through an unknown environment while building a map that can be used for subsequent navigation (YAMAUCHI, 1997). This activity is performed without any prior map information. Thus, mobile robots need to be equipped with some sensors to detect environmental characteristics. Based on the sensor data, the exploration strategy then defines the best path to be followed by the robot.

There are different strategies to execute exploration activities. Generally, the different exploration strategies can be summarized as an activity in which from the sensor readings $\boldsymbol{z}_t$ obtained at the instant of time t and the information previously stored in the local map $\boldsymbol{m}$, if available, makes it possible to generate a control command $\boldsymbol{u}_{t+1}$ responsible for moving the robot pose $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t+1}$, causing the robot to always move to a region of interest. This process is executed recursively generating the trajectory $\boldsymbol{x}_{0:t}$, which can be considered the best path to be taken by the robot at that period.

One of the most reputable exploration strategies is frontier-based exploration (YAMAUCHI, 1997). This technique is proposed for exploring unknown environments. In its logic, the mobile robot scan the environment using its sensors, such as a sonar, a laser scanner, or a camera. From these readings, the robot moves to the frontier, which is the boundary contained between an explored area and an unexplored area. From this, it performs this task repeatedly, moving iteratively toward the frontiers. This way the robot can

explore an entire region while building a map.

Another highly-regarded exploration strategy is potential field-based exploration (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011; KOREN; BORENSTEIN, 1991). In this strategy, the sensor data is first used to increment the current local map. Iteratively, the algorithm copies the instance of this local map and based on the position of the obstacles calculates a gradient of the same size as the current map. The robot accesses the gradient cells relative to their current position on the map and then moves in the downward gradient direction, ie to the lowest potential field cell value, visiting all unexplored and accessible regions and avoiding all obstacles. This technique can be improved to soften the movement of the robot and maintain a reasonable distance from obstacles (BORENSTEIN; KOREN *et al.*, 1991a).

Finally, the Reinforcement Learning (RL) based exploration strategies (LEERINK; SCHULTZ; JABRI, 1995; HESTER; LOPES; STONE, 2013). These strategies are often bioinspired and seek to solve exploration problems based on dynamic environments. As ants example, if their path from the nest to their food is not the best, they are wasting some time and energy on exploring the environment. These techniques always try to maximize the rewards involved in the execution of the activities. In this type of RL application, there is a tradeoff between exploration (where the first task is to find the objective and then optimize the path to it) and exploitation (use the best path and avoid obstacles).

2.1.5 Map Merging

In robust navigation systems, applied to autonomous mobile robots, it is common a process of building several virtual maps to represent the same environment or to represent different parts of this same environment. These maps can be created at different time intervals, by the same sensor set, or even created at the same time by different sensor sets arranged by the robot body, or by two different robots. These different virtual representations of environments are extremely important to the navigation system, as each of these representations describes one of the robot's views of the environment in which it is located.

Merging multiple maps into a single global map seeks to benefit navigation systems, such as reducing the amount of information to be transmitted and stored, improving the quality of information, and enabling better localization of robots. This activity of merging different maps is known in mobile robotics as map merging (Birk; Carpin, 2006; JIANG *et al.*, 2019).

Birk and Carpin (2006) present in their work a formal definition of map merging that can be applied in case maps are formatted by two-dimensional (2D) structures. From this definition it is possible to extend it, in a general way, to exemplify the case where maps are composed of three-dimensional (3D) structures.

Then, a map can be defined as a function represented by (1), where w, d, and h are

positive integers and represent the size of the three dimensions of the map, being width, depth and height respectively. Each map cell can be accessed by $\boldsymbol{m}(x, y, z)$ and represents the state corresponding to a region or a specific feature of the environment.

$$\boldsymbol{m} : [0, w] \times [0, d] \times [0, h] \to \mathbb{R} \tag{1}$$

For purposes of illustration, it assumes that there are two distinct maps $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ with some regions in common.

Being (2), (3) and (4) matrices responsible for rotating around the x, y, and z axes , it is possible to define the function represented by (5) responsible by rotating a map cell $\boldsymbol{m}(x, y, z)$ around the three axes simultaneously.

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2}$$

$$R_y = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

$$R_z = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{4}$$

$$R_{xyz}(x, y, z) = R_x \cdot R_y \cdot R_z \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{5}$$

Now, it is possible define a function represented by (6) which is a transformation composed of translation and rotation operations.

$$T_{t_x,t_y,t_z,\alpha,\beta,\theta}(x, y, z) : \mathbb{R}^3 \to \mathbb{R}^3, \tag{6}$$

and can be defined by:

$$T_{t_x,t_y,t_z,\alpha,\beta,\theta}(x, y, z) = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot R_{xyz}(x, y, z). \tag{7}$$

From the function represented in (6), it is possible to apply a transformation in $\boldsymbol{m}_2$, recursively, until the perfect match between similar regions contained in the $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ maps. This transformation matrix can also be called by Map Transformation Matrix (MTM).

In summary, the map merging activity is intended to find the rotation and translation and, in some cases scale, parameters responsible for matching the maps $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ to merge them into a global map $\boldsymbol{W}$. For cases where the number of maps is greater than or equal to three, this activity should be performed sequentially to determine the transformation parameters for all maps. This technique is popularly known in the literature as pair-wise map merging.

Several methods can be used as a metric to measure how good the map merging is. This merging quality metric is often used as a stopping condition for the map merging algorithms. One of the most common methods is based on image similarity (Birk; Carpin, 2006). This technique is often used in combination with a heuristic to identify the alignment of overlapping regions among maps. Another approach found in the literature is based on the idea of retrieving all global motions to merge maps from a set of relative motions of different robots (JIANG *et al.*, 2019). Also, maps can be merged using computer vision techniques (Velásquez Hernández; Prieto Ortiz, 2020; FERRãO; VINHAL; DA CRUZ, 2017).

2.1.6 Simultaneous Localization and Mapping (SLAM)

Simultaneous localization and mapping (SLAM) can be considered one of the most challenging and important problems nowadays related to the construction of autonomous mobile robots (Cadena *et al.*, 2016; STACHNISS; LEONARD; THRUN, 2016). While performing the activity of navigating the environment, the robot seeks to build a map and at the same time needs to locate itself using this incomplete and often inaccurate map. The problems addressed by SLAM can be motivated by two different fronts: the first seeks to generate detailed environment models, and the second, seeks to maintain an accurate sense of localization of a mobile robot.

A formalization for the SLAM problem is defined as follows: A mobile robot is designed to navigate in an unknown environment, starting at a location denoted as the robot's initial pose $\boldsymbol{x}_0$. Its movement is produced by the $\boldsymbol{u}_{0:t}$ control commands. This movement is considered uncertain. At the same time, a sensing process is applied to the environment by taking the readings $\boldsymbol{z}_{0:t}$ used directly to construct the local map $\boldsymbol{m}$. From this data, it becomes possible to estimate the robot's pose on the map. However, the uncertainty of your current pose $\boldsymbol{x}_t$ increases over time, making it gradually more difficult to determine your current pose in the global coordinates with a certain level of accuracy.

The major difficulty surrounding the SLAM problem is related to the tight coupling or simultaneity of the tasks employed. Thus, the problem can be described as a problem

similar to the approach to the chicken-egg problem (ENDRES *et al.*, 2012). Looking at the problem from this point of view, it is possible to perceive that a correct knowledge of the robot pose is required to accurately construct a map, and on the other hand, a precise map is also required to locate and correctly estimate the robot pose.

Even in robust navigation systems, after a certain amount of time performing an exploration activity, or even by the topology presented by an environment of appropriately small proportions, it is likely that the robot will revisit a known region. Often, because of the error in the robot pose estimation process, these revisited locations appear in other locations of the map previously generated. This type of problem can distort the map, and in the worst case, can overwrite some parts of it, producing an unusable map.

Loop closure detection is one of the key tools used by SLAM, helping the robot to identify these intersections among map regions, and making corrections in the map based on new intersections discovered by the robot over time (Cadena *et al.*, 2016). Through loop closures, the robot can interpret the actual topology of the environment and even make it possible to find paths among previously visited locations.

A problem associated with possible corrections detected by loop closure algorithms is known as a data association problem. In this problem, an algorithm is employed to merge new data into the current map representation. SLAM algorithms require a reliable method to estimate correspondences between your sensor measurements and observed reference points. Incorrect associations between measurements and existing landmarks on the map can often not be reviewed, resulting in incorrect estimates of map regions that may be unrecoverable (BAILEY; DURRANT-WHYTE, 2006).

2.2 Computer Vision Techniques

This section describes the main computer vision algorithms theory used for the development of this thesis.

2.2.1 Erosion and Dilation Morphological Operations

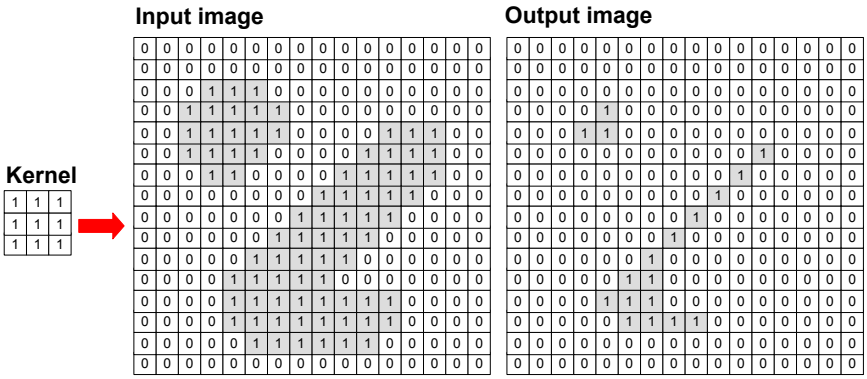
Erosion and dilation are two basic operators in the field of mathematical morphology (GONZALEZ, 2008; JONKER, 2000).

These operators are applied to modify the structure of the images from the pixels in the foreground, that is, from the edges evident in the image, generating the effect of erosion and dilation. Both operators are applied to images through a process of convolution of a kernel over an image.

An example of the erosion filter can be seen in Figure 4. As a basic effect of this filter, the areas near the edges of the image decrease in size, and the holes within those areas become larger.

Figure 5 demonstrates an example of the dilate filter. As a basic effect of this filter,

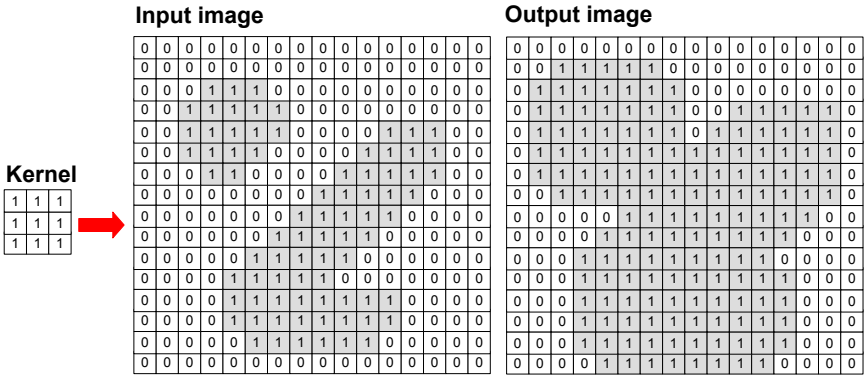
Figure 4 – Erosion filter example.



Source: author.

the areas near the edges of the image increase in size, and the holes inside those areas become smaller or even disappear.

Figure 5 – Dilation filter example.



Source: author.

These filters when combined are excellent tools for removing noise in images. The examples shown in Figures 4 and 5 can be reproduced through a basic implementation in the repository[1] git available.

2.2.2 Keypoint Detectors

Keypoints are regions of relevance that can be used to prevent computer vision algorithms from having to perform processing in all regions of the image, significantly speeding up processing. In this section, the main keypoint detectors are presented.

[1] https://github.com/maikbasso/mathematical_morphology_erode_dilate.git

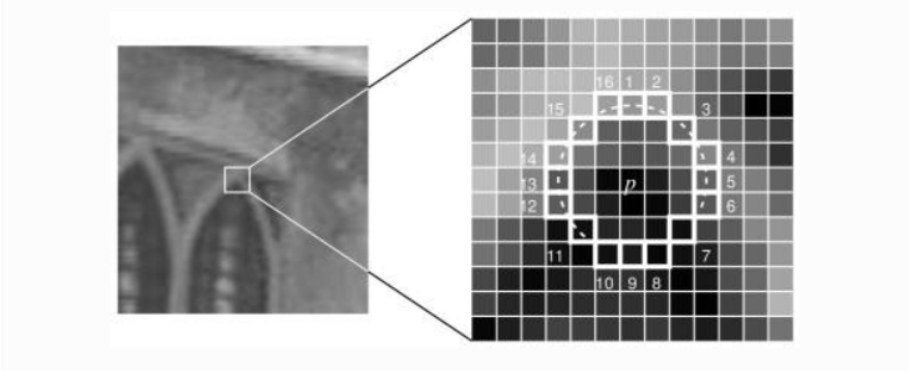
2.2.2.1 Features From Accelerated Segment Test (FAST)

Features from accelerated segment test (FAST) (ROSTEN; DRUMMOND, 2006) is the best performing corner detector among the best known. Originally the algorithm was proposed for images in two dimensions.

The main idea of the algorithm is the application of rapid tests based on the difference in intensities between the analyzed point and its neighbors. From the results of these tests, identify the corners.

The keypoint selection process performed by the proposed algorithm can be seen in Figure 6.

Figure 6 – FAST circle around p point.



Source: ROSTEN; DRUMMOND (2006).

First, a point p is selected to check whether it is of interest or not, its intensity being defined by I_p. Afterward, an appropriately selected threshold t. Then a 16-point circle around p is established.

From these selected points, a high-speed test is performed to exclude points that do not correspond to corners. For this, the four points with index 1, 9, 5, and 13 (points directly above, below, to the right, and the left respectively in Figure 6) are analyzed.

For p to be a corner, then at least three of these points must have an intensity greater than $I_p + t$ or less than $I_p - t$. Otherwise, the point p cannot be a corner.

If the quick test returns a positive value, it is checked if there is a sequence of at least n continuous points where the points are more intense than $I_p + t$ or less intense than $I_p - t$, where n is a previously defined continuity parameter.

FAST also includes a filter to remove adjacent keypoints. Through the use of the Non-Maximum Suppression filter, adjacent keypoints are then removed. In this filter, every keypoint is assigned a score. After among a set of adjacent points, those with the lowest score are removed. Finally, the algorithm returns a list with the most relevant keypoints.

2.2.2.2 *Harris*

Harris detector (HARRIS; STEPHENS, 1988) is a well-known keypoint detector algorithm, one of the best known and used in computer vision applications.

The purpose of the detector is to use a combined edge and corner detection solution because corners are regions in the image with a considerably high-intensity variation.

The algorithm consists of the task of finding the difference in intensity for a given point (u, v) in all directions within a window of a predetermined size. This process is performed by using (8) which corresponds to the local autocorrelation function of the signal.

$$E_{x,y} = \sum_{u,v} w_{u,v}[I_{x+u,y+v} - I_{u,v}]^2 \tag{8}$$

where $w_{u,v}$ is the window function that weights the points within it, usually defined as a Gaussian.

The next step of the algorithm consists of applying a Taylor Expansion on (9) to maximize the detection of corners.

$$E(u, v) = \begin{bmatrix} u & v \end{bmatrix} M \begin{bmatrix} u \\ v \end{bmatrix} \tag{9}$$

where M is defined by (10).

$$M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \tag{10}$$

where I_x and I_y are derived from the image in the directions x and y respectively. Finally, calculate the Harris response R, defined by (11).

$$R = det(M) - k(Tr(M))^2 = \lambda_1 \lambda_2 - k(\lambda_1 + \lambda_2)^2 \tag{11}$$

By defining some limits to R, it becomes possible to determine whether or not there is a corner in the analyzed region.

The original algorithm was proposed for two dimensions, in 2011 the algorithm received an extension to work in three dimensions (SIPIRAN; BUSTOS, 2011).

2.2.2.3 *Shi-Tomasi*

Shi-Tomasi detector (SHI; TOMASI, 1994) was developed based on the algorithm proposed by Harris. The main modification proposed was in the calculation of Harris' response. In the new proposed version, R is obtained by (12).

$$R = min(\lambda_1, \lambda_2) \tag{12}$$

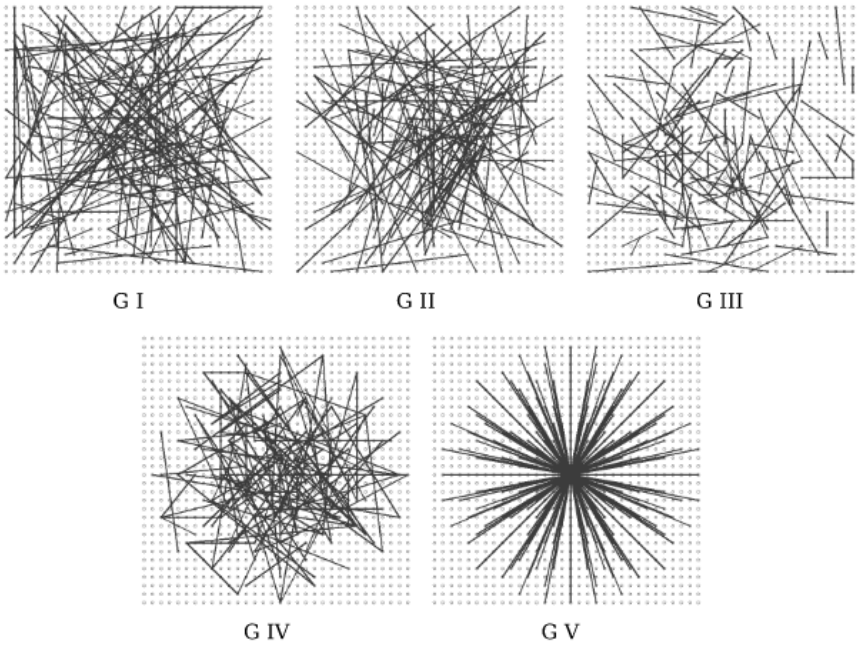
The results obtained by the proposed modification demonstrate a great increase in the performance of the algorithm to the original. The algorithm was also proposed for two dimensions.

2.2.3 Binary Robust Independent Elementary Features (BRIEF)

Binary Robust Independent Elementary Features (BRIEF) (CALONDER *et al.*, 2010) is a powerful keypoint descriptor that was developed to be fast and considerably reduce memory consumption. The proposed descriptor structure uses binary strings to describe the keypoints instead of floating-point vectors.

The algorithm operation flow starts with the definition of a window of size $S \times S$. Within this window, $n_d(x, y)$ pairs of randomly distributed points are created inside this window. This set of points is defined as a sample. Figure 7 demonstrates some examples of sample distribution for sample composition proposed by the author.

Figure 7 – BRIEF samples example.



Source: CALONDER *et al.* (2010).

The defined sample is then positioned over the keypoint. Then, for each points pair that makes up the sample, a binary test τ defined in (13) is defined.

$$\tau(p; x, y) := \begin{cases} 1 & \text{if } p(x) < p(y) \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

where $p(x)$ and $p(y)$ are the pixel intensity of the corresponding point pair in the image smoothed by a Gaussian filter.

BRIEF is then defined by a n_d-dimensional binary test string (bitstring) as demonstrated in (14).

$$f_{n_d}(p) := \sum_{1 \leq i \leq n_d} 2^{i-1} \tau(p; x, y) \tag{14}$$

In the article, the author suggests using n_d with values of 128, 256 or 512.

Finally, the corresponding bitstrings can be combined using Hamming Distance (HAMMING, 1950) to find possible matches between the analyzed keypoints.

2.2.4 Random Sample Consensus (RANSAC)

Random Sample Consensus (RANSAC) is a robust estimator of mathematical model parameters (FISCHLER; BOLLES, 1981). It is considered a non-deterministic iterative method, based on the principle of generating and verifying hypotheses.

The algorithm works from an observed data set that contains outliers. The main idea of the algorithm is to retrieve the best parameters through the performed iterations, and then return the best set corresponding to the data inliers.

The probability of success of the algorithm increases consecutively with the number of iterations performed. The RANSAC algorithm can operate with a percentage of outliers that can be greater than 50 % of the entire available dataset. This threshold is considered as the limit for the proper functioning of the proposal (HUBER; M., 2009).

2.3 Multiple UAV Cooperative Systems

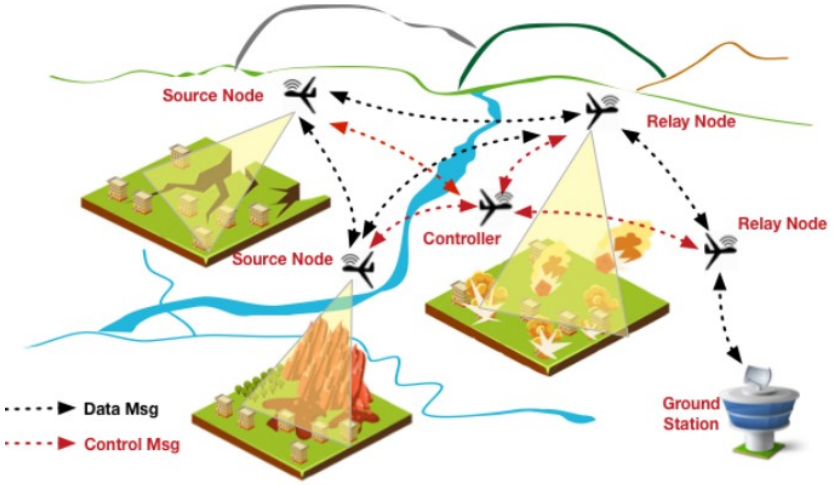
The increasing use of UAVs to perform previously unattainable tasks, such as deliveries, search and rescue, is related to the number of proposed applications that use sensor data and other multidimensional data integrated into the UAV control channel to enhance their skills of navigation and orientation, trying to provide autonomy to these almost independent systems. This makes it possible to develop autonomous (or semi-autonomous) air systems capable of completing missions independently of human interaction (or with very little human intervention) (AL-KAFF *et al.*, 2018; KANELLAKIS; NIKOLAKOPOULOS, 2017).

Missions consist of a set of tasks to be completed to fulfill one or more common objectives. Each task requires some specific resources to accomplish it. These resources can be provided by one or more systems that make up one or more UAVs. As a result, the latest studies are starting to address systems with multiple UAVs and heterogeneous resources (TRUJILLO *et al.*, 2017).

Multiple UAV systems can perform missions in unstructured environments, such as disaster scenarios (Figure 8), and achieve mission objectives with a reduced computational and temporal cost.

Compared to a single UAV system, multiple UAVs systems can benefit from greater accuracy and efficiency due to their resource heterogeneity, as well as better accessibility and robustness in the mission (ZHAO; LI; ZHANG, 2017). Besides, a multiple UAVs

Figure 8 – Video surveillance in a disaster recovery scenario.



Source: ZHAO *et al.* (2019).

system can be considered as a multi-agent system (MAS) in which an efficient consensus scheme can be implemented to accommodate any number of agents for different scenario configurations and applications (HAN *et al.*, 2016).

However, in a mission with UAVs, the mission's input parameters may be uncertain because these devices operate in a dynamic and uncertain environment. Effective mission control and guidance systems must be able to cope with environmental changes such as wind, which have a major impact on UAV battery consumption, which can significantly alter mission performance and progress (EVERS *et al.*, 2014). These problems are even more aggravating when it comes to systems with multiple UAVs.

The mission planning process for a team of UAVs involves the generation of tactical objectives, commanding structure, coordination, and schedule (RAMIREZ-ATENCIA *et al.*, 2017). In these cooperative systems, autonomous task allocation and planning are carried out for heterogeneous UAV networks (COOPERATIVE MISSION PLANNING FOR MULTI-UAV TEAMS, 2015). The proper functioning of these systems requires stable communication, and time synchronization between tasks and this is not always easy to achieve.

In the middle of these points, it is evident in multiple UAVs applications that vehicle navigation is considered a prime and essential point for the operation of these applications. There is a need for a robust navigation system capable of combining various UAV experiences to build a cooperative and robust system.

2.4 Implementation Tools

This section describes, in general, the main tools and frameworks that will be used for the development of this thesis.

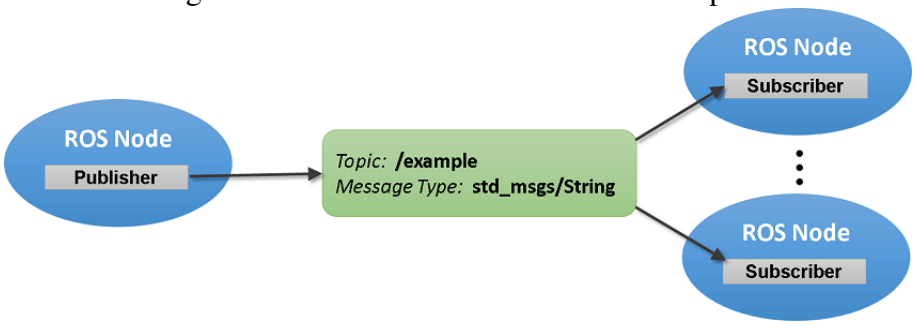
2.4.1 Robot Operating System (ROS)

Robot Operating System (ROS) is a highly regarded framework that provides support for developing robot applications (QUIGLEY; GERKEY; SMART, 2015). ROS is a pseudo operating system because it needs a host system, such as Linux, to be executed. ROS has package management tools, simulators, and hardware abstractions that can improve development. Also, applications developed using this framework are easily extensible and can be used on low-cost embedded hardware.

Applications developed in ROS are distributed in the form of *packages*. Each *package* consists of sets of programs and scripts used for execution, compilation, and simulation. These programs are called *nodes*.

The communication infrastructure between *nodes* is provided by a server named *roscore*, and enables the development of distributed applications. The simplest type of communication between *nodes* occurs through a scheme known as *publisher-subscriber*, as can be seen in Figure 9.

Figure 9 – ROS Publisher/Subscriber example.



Source: MATHWORKS (2019).

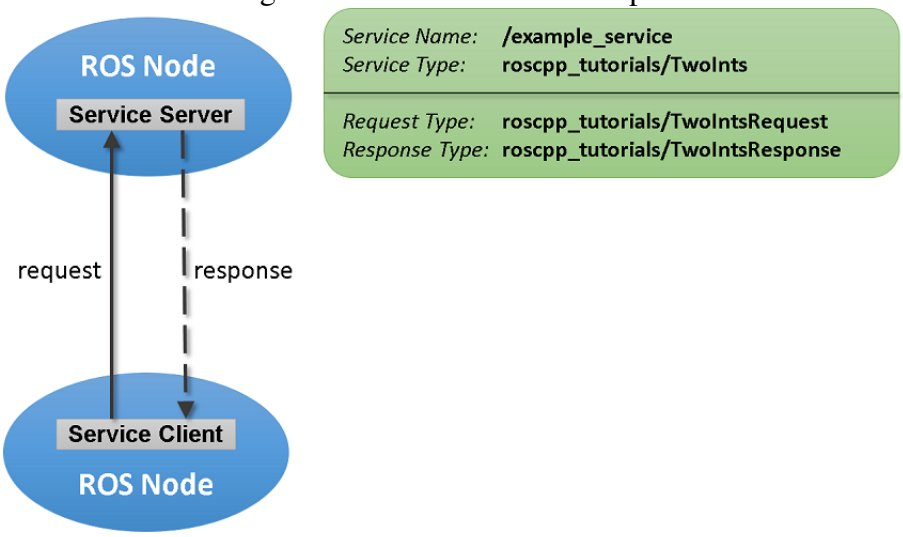
In this type of interlace of communication, the *nodes* executing the *publisher* function, publish data in *topics* on the server *roscore* either sporadically or continuously, without worrying about how it will be consumed. The *topics* are reserved addresses for publishing data on the server. Data is published in the form of structures known as *messages*. Each *message* has a structure type that can either be standard of the framework or simply designed for use in a specific *package*. Each *topic* is set to accept a specific type of message.

The *nodes* executing the *subscriber* function, receive the published data through callbacks triggered by the publish event. Ideally, in this structure there is only one *node* executing the *publisher* function while one or more nodes perform the *subscriber* func-

tion consuming the data for different purposes. Using more than one publisher *node* node over the same *topic* is allowed, but not recommended.

Another type of interlace of communication provided by ROS is *client-server* and can be seen in Figure 10.

Figure 10 – ROS Service example.



Source: MATHWORKS (2019).

In this communication model, the client *node* sends a request to a *topic* where this service is available and then waits for a response. The server *node* receives the request and performs the necessary processing. It then returns the response to the client *node*. This communication model can be used in sporadic situations that do not require constant data dissemination.

Through the presented resources, ROS was adopted as a base framework for the development of all the applications proposed in this thesis.

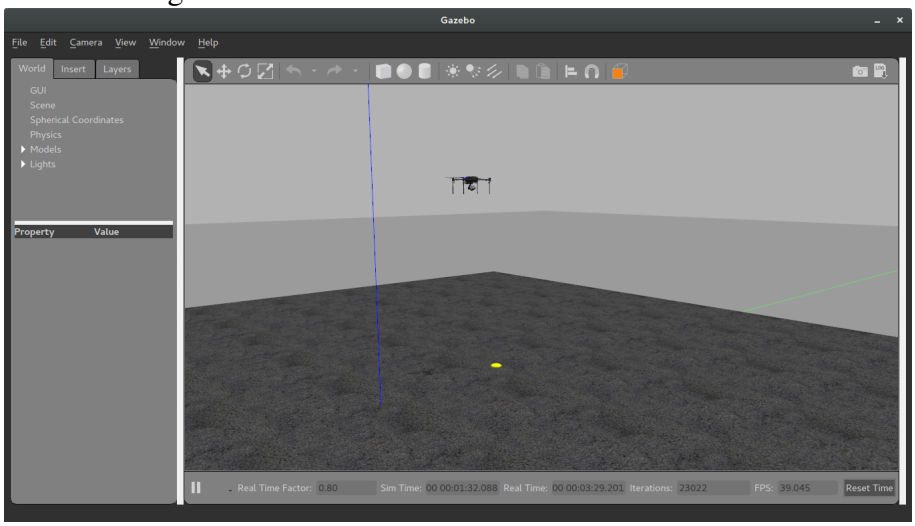
2.4.2 Gazebo Simulator

The Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate robots in complex environments (GAZEBO TUTORIALS, 2014). This simulator has a similar structure to the mechanisms used in games, but Gazebo offers high fidelity in the physical simulation, a set of sensors and interfaces for users and programs.

The Gazebo is designed to accurately reproduce the dynamic environments where a robot might encounter. All simulated objects have mass, speed, friction, and various other attributes that allow them to behave more realistically when pushed pulled, knocked over, or transported. Robots are simulated as dynamic structures composed of rigid bodies connected via joints. Forces, angular and linear, can be applied to surfaces and joints to generate locomotion and interaction with the environment (Koenig; Howard, 2004).

Figure 11 demonstrates the interface presented by Gazebo during a UAV takeoff simulation.

Figure 11 – UAV simulation on Gazebo interface.



Source: author.

2.4.3 PX4 Firmware

PX4 is an open-source flight control hardware and software project for drones and other unmanned vehicles (PX4 OPEN SOURCE AUTOPILOT, 2018). This project provides a huge set of tools and technologies to create custom solutions for these vehicles.

One of the most important tools in this package is known as software in the loop (SITL), which is a set of software, including the firmware, that allows the simulation of a flight controller unit (FCU) to be performed directly on the computer, without any special hardware. In addition to this tool, the package has three-dimensional models highly regarded and properly prepared for the integration and development of applications for UAVs based on a package implemented in the ROS-Gazebo ecosystem.

In this thesis, a fork of a specific version of the PX4 software package[2] is used as the basis of the integration of all proposed software.

[2]https://github.com/maikbasso/Firmware

3 PROBLEM STATEMENT

The objective of this chapter is to explore, in a simplified way, the possible problems surrounding the cooperative navigation systems, destined for multiple UAVs, based on the construction, sharing, and merging of maps produced in a distributed way. The central idea is to highlight the identified problems, analyze them, and propose alternative solutions for each problem under concern, whenever possible.

Section 3.1 demonstrates the application scenario, as well as problems related to the environment in which the system is inserted. Section 3.2 demonstrates the problems related to applying mapping in these environments. Section 3.3 highlights problems with communication systems between UAVs. Section 3.4 addresses the specific problems of applying map merging. Hardware limitation related issues are covered in Section 3.5. Section 3.6 highlights some issues related to privacy that are hardly addressed but are present in these systems.

3.1 Application Scenario

Figure 12 illustrates an example of a scenario for a navigation system composed of multiple UAVs operating cooperatively.

Figure 12 – Example of a cooperative navigation system application scenario.

Source: author.

In this scenario, several UAVs are randomly placed in the environment. Each UAV knows its local position, but it does not have an estimate of its global position nor of the positions of its neighbors.

The UAVs move through the environment in a completely independent way through an unknown area to perform the full recognition of the region and build a virtual representation, which can be used by other applications or even for autonomously vehicle navigation through the environment.

Whenever two or more UAVs are within communication range, they can share the acquired information in the form of maps. These local maps received by each UAV, go through a merging process in a fully distributed way, where each UAV tries to create and share its global representation of the environment in which it is inserted.

In this context, there are several problems with this type of application. One of the most obvious sets of problems is related to the environment. Generally, the environment is unstructured, which means that it is unlikely to be virtually represented by known basic geometric shapes.

The environments are also completely unknown and present numerous obstacles and restrictions to mobility and accessibility. They can also be static or dynamic, that is, present moving objects, such as doors, windows, people, or animals.

Additionally, these scenarios can be represented as indoor, outdoor, or both. Outdoor environments are generally unstructured and are even susceptible to climatic variations.

Indoor environments, in turn, may have artificial lighting and narrow passages. In these environments, it is also common for global positioning systems (GPS) to be partially available or completely unavailable. These environmental variations and diversities can generate great noise in the UAV sensors, making their operation difficult.

In this thesis, the navigation system is designed not to be concerned with the specific characteristics of each set of environments, that is, it can operate both indoors and outdoors.

3.2 Mapping Problems

In this section, problems related to the mapping system are presented. The mapping system is responsible for collecting the information coming from the sensors and building the local virtual representations with a certain level of quality.

The first identified problem refers to the mapping technique, which in many cases is used to generate representations of the environment in a two-dimensional format, ignoring the possibility of mobility at different altitudes, whether in the air or hilly regions.

This is an important limitation, as can be seen in the examples of Figure 13. In these examples, the UAV explores two environments where its possible trajectories require changing altitude to avoid obstacles and access regions in different positions.

Two-dimensional virtual representations cannot provide enough information for the vehicle to navigate these gaps of free space. This type of approach can also prevent navigation in steep areas in multi-floor environments.

Figure 13 – Two examples of exploration scenarios that require flight at different altitudes (flight levels).

a) Small house example. b) Bookstore example.

Source: author.

The second issue is related to map quality. The accumulated error with pose and sensor readings can somehow cause distortions in very large rectilinear regions, closing passages and taking the UAV to possible collisions if the map is used as input to generate its trajectory.

In this thesis, the maps are built in three dimensions to solve the first problem related to the mobility model of UAVs. The map quality problem is partially solved in this thesis by estimating the correct values of the inputs in two stages, in the first stage, the vehicle pose is estimated by an Extended Kalman Filter (EKF) inside the vehicle. Then the sensor data is refined using a simple Kalman Filter (KF) applied to each of the sensor data inputs.

3.3 Communication Problems

This section describes the problems directly related to communication among members of the cooperative network composed of multiple UAVs. Despite presenting advantages for the application as a whole, as already mentioned, the multiplicity of UAVs adds some problems and difficulties to the navigation systems. Cooperative systems, for the most part, require constant communication and synchronization between the tasks they perform. In addition, they are more prone to collisions, as, in addition to being concerned with environmental obstacles, they need to know their positions to avoid collisions with each other.

Thus, constant communication is a key factor for the proper functioning of these systems. In this thesis, UAVs do not require constant communication and all processes occur in a fully distributed way. Maps are shared whenever two or more vehicles approach within an acceptable range of communication.

A frequent problem of distributed systems is the overload in the communication channel due to the high amount of transmitted data. In this thesis, the maps go through a compression process to reduce the size and quantity of data to be shared. Also, maps are only shared if they have some kind of updated information to be shared.

Route management and the use of a protocol based on multi-hop could contribute even more to the good functioning of cooperative navigation systems, but in this thesis, these points were not addressed.

3.4 Map Merging Problems

In addition to moving around and mapping their environment, the various UAVs need to know how to gather information to create global representations of maps captured individually by each vehicle. In this context, map merging techniques are employed.

This activity has some clear issues to address. The first refers to regions of minimal overlap among two local maps. That is, although UAVs should avoid crossing the same route too many times, they should create maps with overlapping regions of a minimum acceptable size so that merge techniques can identify similarities and then generate acceptable parameters to create the virtual global representation.

The second problem concerns large maps or maps with many similar regions. They can often have similarities across multiple locations, making it difficult for merging techniques to estimate which is the best option to make the proper match.

These issues can be easily resolved using GPS data to calculate relative poses between vehicles, but as shown, this feature is often unavailable (or partially available) depending on the environment.

In this thesis, several filters are proposed and integrated into the merge system to deal with problems with similar regions. The problem with the minimum overlap region is directed to the mobility model adopted for vehicles, as this thesis does not deal with a trajectory control system, if the map merging parameters are not found, the local maps are simply not merged.

3.5 Hardware Limitations

Generally, UAVs have their hardware composed of a Flight Control Unity (FCU), which has a processor connected to several sensors such as gyroscopes, accelerometers, and barometers integrated to perform position and mission control. The hardware architecture also has Electronic Speed Controllers (ESCs) designed to receive input from the FCU and correctly command each of the UAV motors. This set of hardware combined with a frame, usually composed of plastic and carbon fiber, and a battery correspond to the basic set of hardware for building a UAV.

This architecture can be enhanced with GPS, radio systems, to enable remote control, and other sensors such as lasers, sonars, and cameras to aid navigation or simply collect data from the mission environment. Some actuators like liquid sprays and manipulators can also be used.

To execute more complex software applications, such as processing sensor data, controlling actuators, and executing a complex navigation system, usually, a computer with greater computational capacity is embedded and integrated into this entire system.

One of the most relevant points in this entire architecture is that usually more than 70 % of the weight of UAVs is made up of the battery that guarantees limited autonomy to these systems. There is then a tradeoff between weight and autonomy. So the use of hardware with high computational capacity is not always adequate, this type of hardware usually has a significant weight that can affect the vehicle's autonomy.

Finally, the biggest problems identified in the hardware architecture of these systems are related to limitations involving autonomy and computational capacity. These limitations can be overcome with a good balance of systems. Thus, the software of navigation systems needs to be well designed, have its architecture optimized, using the hardware resources as efficiently as possible.

To alleviate these problems and limitations, this thesis adopts a distributed architecture composed of a cooperative navigation system composed of multiple UAVs. In this way, the processing aimed at merging the maps is distributed among the vehicles.

3.6 Privacy Issues

This section addresses privacy issues related to navigation systems. These problems are associated with the use of visual sensors that can almost faithfully store and restore views of the environment in which the navigation system is inserted. In this context, some problems regarding privacy can be identified.

Exposing data such as documents and visual information about people, such as their faces, specific features, and even intimate parts, can be serious privacy issues. Through this information, people can be tracked with a certain level of precision and in a way, as an example, contribute to the construction of autonomous weapons.

Soon, these questions should be included in the most recent research, contributing to the construction of ethical principles in mobile robotics. In this thesis, the information obtained from the environment is used to compose occupancy grid maps with adequate cell resolution, later these raw data are discarded, which somehow partially solves the problems mentioned above.

Privacy issues will not necessarily have solutions proposed in this thesis, but they are highlighted here to encourage the scientific community to address these problems in their future research.

4 RELATED WORKS

In this chapter, related works are presented. First, Section 4.1 presents generally related works dealing with applications related to navigation systems. Then, in Section 4.2, the related works to map merging applications are presented. Finally, Section 4.3 presents a comparison between the related works presented, including a brief discussion about the main problems, approaches, and possible solutions relating these works presented to the solutions present in this thesis.

4.1 General Related Works

KOCH *et al.* (2016) describe in your work, a 2D Simultaneous Localization, and Mapping approach applicable to multiple mobile robots. The strategy uses data from 2D LIDAR sensors to build the representations based on Signed Distance Functions. The approach uses a joint map built-in parallel instead of occasional local map merging and the limited drift localization which requires no loop closure detection. The software architecture is multi-threaded and performs registration and data integration in parallel allowing for drift-reduced pose estimation of multiple robots. The experiments were performed on an Intel Core i7 quad-core with Ubuntu 14.04 LTS operating system and ROS Indigo. Several sets of experiments were performed involving single and multiple robots, where it was found that the application performs well on large maps. However, as the ground truth experiments showed, the registration module has weaknesses, causing in some cases problems such as erased walls.

A monocular-based cooperative SLAM approach for multiple UAV systems that can operate in environments without GPS signal is presented by TRUJILLO *et al.* (2018). According to the author, the main contribution of the work is to demonstrate that the observability properties of the whole system are improved through the use of visual information obtained from monocular cameras installed in air vehicles flying in formation. To improve observability properties, this approach proposes that some relative distance measurements, obtained from visual information between UAVs, be included in the system. The approach has been validated in two ways through the implementation of the MAT-

LAB software. The first was theoretically utilizing a nonlinear observability analysis. In the second stage, a set of computer simulations were performed. The author also points out that from the results obtained the proposed system can provide a good estimate of the position and orientation of air vehicles flying in formation.

Schmuck; Chli (2017) propose a centralized architecture for collaborative monocular SLAM that can be used on various UAVs, making them act as agents. According to the author, in this proposed architecture, each agent can explore the environment independently by executing SLAM algorithms, simultaneously sending all collected information to a central server, which is a ground station with enhanced computational resources. The central server, in turn, is used for managing the maps of all agents, triggering loop closure, map merging, optimization, and information distribution to agents. This approach allows agents to incorporate observations from other agents into their runtime SLAM algorithm estimates. Experiments were performed with up to four UAVs, and a Thinkpad T460s notebook with a Core i7-6600U @ 2.60GHz quad-core and 20 GB RAM is used as the server. Through the obtained results, it is possible to realize that agents can act cooperatively, and the trajectories of a single agent can be improved using a collaborative system. However, according to the author, the size of accumulated server experiences is likely to become the bottleneck, preventing effective collaboration between agents, especially when the number of agents or regions explored grows.

Qin *et al.* (2019) present in their work a new autonomous exploration, mapping, and navigation system using unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs) collaboratively. The system is proposed to work in unknown environments but makes use of GPS systems for localization. According to the author, the proposed system implements an exploration strategy based on two main layers. The first layer uses a UGV to perform rapid autonomous exploration and simultaneous location and mapping to generate a rough environment model, which serves as a navigation reference for subsequent complementary 3D thin mapping performed by a UAV, which in turn makes up the second system layer. The two layers share a trajectory planning framework, which, according to the author, provides optimal exploration paths and integrates collaborative exploration and mapping efforts through a volumetric motion planning interface. The results were obtained in two stages, first simulations were performed and later experiments were performed with real vehicles (a robot and a drone). The results show the ability of the proposed system to perform collaborative (UAV and UGV) and heterogeneous exploration as well as the structural reconstruction of the environments by SLAM algorithms.

A new vision-aided inertial navigation system approach is introduced by CHOWDHARY *et al.* (2013). According to the author, the proposed architecture efficiently combines visual information from a monocular camera with inertial sensor measurements. Where the inertial measurements are used to predict frame-to-frame the transition of selected feature locations, and the difference between predicted and observed feature loca-

tions is used to bind the inertial measurement unit drift, and account for initial misalignment errors. These estimates are used as a basis for managing the feature library and may add or remove according to the confidence level obtained. The feature point database can be considered as the map obtained by the mission. The algorithms were validated by autonomous flight tests on extended closed-loop operation indoors and outdoor using UAVs. According to the author, the results obtained are efficient and reliable to allow real-time deployment in resource-constrained unmanned air vehicles.

A three-dimensional collaborative mapping system for use in an earthquake-damaged building based on aerial and ground robots is presented by MICHAEL *et al.* (2012). The main purpose of the proposed application, according to the author, was to enable the generation of three-dimensional maps that capture the layout of a multi-floor environment. The experiments reported in the work took place on the first three floors of a structurally compromised building at Tohoku University in Sendai, Japan, which was damaged during the Tohoku earthquake in 2011. According to the author, one of the biggest problems faced in the experiments was the communication bandwidth required to transfer data between the aerial and the ground vehicles and the base station, which performs the map processing. The author complements the results discussion and firm that for operations involving search and rescue in unknown regions and rugged territories the maps must be more detailed to enable the navigation of vehicles and also possible detections related directly to the application.

Extensive research involving the problems related to autonomous navigation of a UAV in unknown, unstructured GPS-denied environments is presented by BACHRACH *et al.* (2011). According to the author, one of the main challenges is that the system must be able to estimate its position and speed by detecting an unknown environmental structure with sufficient accuracy and low latency to control the vehicle stably. Among the proposals presented in the paper are a data fusion filter, simultaneous high-level localization and mapping, and a goal-oriented exploration module. The results obtained by the study highlight the vehicle's ability to navigate in various unknown large-scale environments, indoors or outdoors in urban scenarios. Among the successful tasks performed by the proposed system is the ability to autonomously enter a dangerous unknown environment through a window, explore the internal structure without GPS and search for a visual target.

A new method of simultaneous 6 degree-of-freedom (DoF) localization and visual mapping based on the structural regularity of man-made environments is presented by Zhou *et al.* (2015). The main idea presented by the work is the detection and later use of the structure lines of buildings as features used for localization and mapping. According to the author, the lines of the building's structure encode global orientation information that restricts the camera's course over time, eliminating accumulated orientation errors and reducing position deviation. The results were obtained with the use of a handheld

camera and show that the proposal presented improvements related to the reduction of position and orientation errors that may harm the map formation.

Pi *et al.* (2014) present in their work a sparse visual SLAM system that focuses on the use of stereo cameras to estimate the movement of autonomous underwater vehicles (AUVs) and construct the feature map of the environment. The algorithm is based on feature detection and matching using the Speeded Up Robust Features (SURF) algorithm. After this process, the 3D coordinates of the identified features are calculated using the stereo vision system's disparity parameters. Finally, the map is generated by merging feature coordinates and the AUV pose with the Extended Kalman Filter (EKF). According to the author, the tests were performed based on raw data collected by the coupled system in a submarine and the results show a good performance in system efficiency. However, the author points out that sparse map representations are often insufficient for common tasks such as path planning, and collision avoidance and that a dense map representation is required.

A landmark-based heterogeneous visual navigation approach for a monocular mobile robot is presented by Lu; Song (2015). In this approach, heterogeneous visual features such as points, line segments, planes, and vanishing points are used for map construction, where internal geometric constraints are managed in a manner not supervised by a multilayer feature graph. According to the results presented by the author, the proposed method decreases the translational error in urban sequences where rectilinear structures dominate the scene.

A new approach to improving the performance of a UAV navigation system in challenging GPS environments is presented by Vetrella; Fasano; Accardo (2016) and by AUTONOMOUS FLIGHT IN GPS-CHALLENGING ENVIRONMENTS EXPLOITING MULTI-UAV COOPERATION AND VISION-AIDED NAVIGATION (2017). These papers explore relief measures for one or more cooperative UAVs flying under full GPS coverage. For this, the work employs sensor fusion based on an EKF. In this process, internal inertial sensor and magnetometer measurements, available GPS pseudo-intervals, position information from cooperative UAVs, and line of sight estimated by vision-based tracking are used as inputs. One of the applications of the proposed techniques aims to provide a pseudo GPS signal for vehicles that are in regions without a signal, for this, using the assistance of cooperative UAV systems. The system performance evaluation was performed using covariance propagation techniques. According to the authors, the results show that cooperative navigation has significant potential for safe flight in challenging GPS environments, also in the absence of range measurements between vehicles.

A study on the implementation of Visual SLAM techniques for UAV images in partially structured environments is presented by ARTIEDA *et al.* (2009). The proposed approach is concerned with optimizing the system by solving the problems related to the visual characteristics of objects in the scene, their distance to the UAV, and the im-

age acquisition system. The experiments were performed using a mini UAV in partially structured environments. According to the authors, the results obtained for location are compared based on flight GPS information and demonstrate that Visual SLAM provides a reliable location and mapping for use in these systems.

A new hierarchical SLAM approach for use in a UAV using the output of an inertial measurement unit (IMU) and bearing-only observations of an onboard monocular camera is presented by WANG *et al.* (2013). A homography technique is used to calculate vehicle motion by matching features in the image. The calculated motion per image, in turn, is merged with inertial outputs using an EKF for attitude and velocity estimation. The proposed approach also uses EKF to estimate vehicle position and feature location on the map. According to the authors, the results obtained through simulations and experiments demonstrate that the proposed system compared to GPS measurements can provide reliable state estimation in GPS denied environments.

A monocular visual SLAM system with application in UAVs is proposed by URZUA; MUNGUíA; GRAU (2017). The proposed method is based on sensory inputs that are obtained from a downward-facing monocular camera, an ultrasonic range finder, and a barometer. The proposed method is based on an observability analysis. According to the authors, the use of monocular vision presents some technical difficulties, such as the impossibility of directly recovering the metric scale of the world. Among the findings highlighted in the paper, it was found that the metric scale can become observable by including altitude measurements in the system and measuring the depth of a single reference point can improve system observability. According to the authors, experimental results with real data show that the proposed method can provide good results by recovering the vehicle's flight path and generating a map of the environment with low-cost hardware.

PEREZ-GRAU *et al.* (2018) present in their work a software architecture for autonomous navigation of aerial robots in denied-GPS areas. This paper presents a localization approach based on visual odometry and Monte Carlo location, and a variant of the Lazy Theta algorithm (NASH; KOENIG; TOVEY, 2010) for motion planning. According to the authors, the results of the extensive tests performed show that the proposed approach guarantees localization and state estimation without any external positioning system, autonomous navigation, collision avoidance with local obstacles, and local trajectory planning.

An active SLAM solution with an active loop closure component, exploration, high precision robot pose estimation, and complete environment mapping is presented by LENAC *et al.* (2016). This proposal uses as an input to the SLAM algorithm, RGB-D images, and odometry estimates obtained from the inertial measurement unit and wheel encoders. The SLAM algorithm used in the proposal is based on the exactly sparse delayed state filter for real-time estimation of the robot's trajectory, vision-based pose registration, and loop closure. The authors propose an active SLAM integration with the 2D

laser scanning algorithm that, according to the authors, ensures complete coverage of a polygonal environment and detailed mapping. The results of the work demonstrate that an active SLAM can maintain the location and accuracy of the map. One of the problems identified is why an active SLAM is cut from the path planning process.

The work from Meng *et al.* (2017) presents an autonomous takeoff, target search, task assignment, and tracking system using multiple fixed-wing UAVs in urban environments. According to the authors, the problem addressed in their research is how to design the flight autonomy built into each UAV to enable autonomous flight coordination and distributed tasks. The proposal focuses on control logic design based on a finite state automaton model, integrating four modes of operation, take-off mode, area-to-area operation mode, search mode, and tracking mode. The experiments were performed in simulation, and from the results, according to the authors, the proposed solution is efficient and can provide autonomy for fixed-wing UAVs.

An indirect cooperative relative localization method for estimating the position of UAVs relative to their neighbors based solely on distance and displacement measurements in GPS denied environments is presented by GUO *et al.* (2017). The proposed method consists of two steps. First, each UAV solves an active 2D relative location problem to obtain an estimate of its initial position relative to a hovering static quadcopter, which is further refined by an EKF to account for noise in distance and displacement measurements. In the second step, a strategy using EKF is employed for the case where all UAVs move simultaneously, thereby enhancing the cooperative location. The proposal was valid through simulations and experiments and, according to the author, obtained positive results in the estimation of relative location between vehicles.

An approach to mapping and exploring multiple distributed robots is presented by Fox *et al.* (2006). This system approach allows teams of robots to explore environments from different unknown locations. Robots use the estimation of their relative locations using a personalized particle filter to ensure consistency by combining their data into shared maps used to maximize exploration efficiency. The estimation of relative positions is integrated into a theoretical multi-robot coordination decision strategy. Mapping and map merging use a SLAM technique that models uncertainty by local probability constraints between laser scanning locations. Shared maps are used to coordinate robots and estimate the location of other robots. The proposed system was evaluated by four experiments where the robots successfully explored the environment. According to the authors, all maps generated during these experiments were virtually identical, indicating the high accuracy and robustness of our system.

CHOUDHARY *et al.* (2017) propose, in their work, a multi-robot SLAM approach that uses 3D objects (including planes and other geometric shapes) as landmarks for location and mapping. According to the authors, the proposed technique is fully distributed because there is no centralized server processing the information. Pose estimation, map-

ping, and other activities are processed locally by embedded computers, and information exchange among robots occurs whenever they are over each other's communication range. Among the results obtained in the realization of the work, the authors highlight that representations using landmarks reduce the memory consumption and information exchange requirements among robots. Several experiments have been designed to ensure that the system becomes robust and noise resistant when compared to centralized approaches.

A SLAM approach to solving the communication and computing problems that affect multi-robot systems is presented by Lázaro *et al.* (2013). The proposed method uses condensed data to exchange map information between the various robots. These compressed data are measurements that can represent, with a small data set, relevant parts of the local maps of each robot. According to the authors, the proposed approach results show a decrease in the data to be transmitted and processed by the robots, increasing the system efficiency. According to the experiments performed, the proposed approach improves performance but presents a little disadvantage in the accuracy of the proposed methods for map generation.

A new navigation system to be used on multiple UAVs that perform missions moving together but without forming is presented by Tang *et al.* (2019). In the proposed approach, vehicles perform search and exploration missions and use two types of vision sensors, day and thermal cameras, to measure relative motion between UAVs in different lighting conditions without the need to use wireless communication. Vehicle grouping during flight is performed by vision algorithms using an integrated tracking-learning-detection framework based on the coded correlation filter for vehicle characteristics. According to the authors, a flocking strategy was developed to deal with the cooperative flight of multiple autonomous UAVs. The proposed navigation system also has a laser sensor used for localization, mapping, and obstacle detection activities. According to the authors, experiments were performed in internal and external scenarios, and these, in turn, prove the effectiveness of the proposed visual algorithm for practical applications of multiple UAV autonomous flights.

4.2 Map Merging Specific Related Works

Birk; Carpin (2006) present in their work a formal definition of the concept of merging occupancy grid maps. They propose the use of a particular similarity metric and a stochastic search algorithm to find maximum overlap regions among the 2D occupancy grid maps, merging them. The problem with this approach is its computational cost, in addition, the metric used to compute the similarity between the maps can make the proposed algorithm fall into local minimums and the fusion never happens, being ideal only for maps with large regions of overlay.

The work presented by Velásquez Hernández; Prieto Ortiz (2020) demonstrates a new

technique for the robust and real-time merging of 2D occupancy grid maps. The proposed technique consists of a corner detector, and a cylindrical descriptor, a matching technique with filtering using the RANSAC algorithm. The author also mentions that the inclusion of 3D information can improve the accuracy of the merging algorithm.

A qualitative and quantitative study with several algorithms for merging 2D (occupancy grid) and 3D (point cloud) maps is presented by BOKOVOY; MURAVIEV; YAKOVLEV (2020). The work uses simulation data to compare different methods. The obtained results demonstrate that the application of map merging does not have a general solution, that is, numerous instances of problems remain unsolved by the latest generation of merging algorithms. The author further concludes that this situation provides avenues for future research, especially for merging 3D maps.

The work presented by JIAN *et al.* (2017) demonstrates an improvement of the ORB algorithm for merging 2D occupancy grid maps. In this approach, map merging is integrated as an image registration problem. The results demonstrate the good functioning of this system.

A new technique for merging 2D maps is presented by PARK *et al.* (2016). The idea of the proposed technique is to estimate the best-shared areas through rectangular features. The information of dimensions and connections of maximum empty rectangles allows the proposed algorithms to combine orientations and scales, allowing to find overlapping points. One of the problems with this technique is that it does not take into account the smallest details displayed on the maps, thus it requires that large overlapping free regions exist for the merge to be possible.

The fusion of 2D maps based on a multi-hypothesis map merging algorithm with sinogram-based particle swarm optimization (PSO) is proposed by LEE; ROH; LEE (2016). According to the authors, the proposed algorithm presented a greater precision in the fusion of maps than the existing techniques that use a single hypothesis.

A technique for merging 3D pose-graphs of point clouds is demonstrated in BONANNI; DELLA CORTE; GRISETTI (2017). The central idea behind the proposed solution is to locate the robot on a reference map using data from another map as observations. The solution is also able to eliminate inconsistencies resulting from distortions that affect the inputs of mapping systems. The solution has been validated against a publicly available dataset and the results demonstrate the complexity of the map merging task.

The work proposed by FERRãO; VINHAL; DA CRUZ (2017) presents a technique for merging 2D occupancy grid maps that use Scale Invariant Feature Transform (SIFT) (LOWE, 1999) to detect keypoints while calculating the transformations (rotation, translation, and scale) to merge the maps. The authors mention that although the solution produces successfully merged maps, a fine-tuning process is needed in the obtained transformations and also improvements in the selection of pivot keypoints.

A hierarchical probabilistic solution for merging 3D occupancy grid maps is presented

in YUE *et al.* (2018); Yue *et al.* (2018). In this solution, the maps are stored in octomaps (HORNUNG *et al.*, 2013). The solution consists of uncertainty modeling, map matching, transformation evaluation, and map merging. According to the author, the algorithm suffers from local minimum problems. Another problem is that many occupancy grid maps are generated by algorithms that do not conserve probability values or rapidly converge their cell values to their maximums and minimums, invalidating the application of the proposed methods or reducing the effectiveness of the solution.

A new map merging algorithm for 2D occupancy grid maps generated by various robots in structured environments is presented in SAEEDI *et al.* (2014). Here, map merging is performed by transforming individual maps into Hough space, thus creating an abstract representation of the maps. The process of defining transformation parameters between previously unknown maps is performed using the properties of the Hough transform. According to the authors, the experimental results obtained in simulation and the field with real robots demonstrate that the proposed approach is fast and accurate compared to other approaches.

A 2D map merging approach using Reinforcement Learning is presented in Dinnissen; Givigi; Schwartz (2012). According to the authors, in the proposed approach, a mobile robot should decide how best to merge its maps with another robot in an encounter, avoiding doing so immediately. For this, the approach uses a decision scheme based on the current status of the mapping particle filters and the current status of the environment in which the robots are inserted. From this, a model capable of defining the best time to merge the maps is defined. The proposal was validated using data obtained through simulations with multiple robots.

An approach to simultaneous merging of multiple grid maps by robust motion averaging is presented in JIANG *et al.* (2019). The main idea of the paper, according to the author of this approach, is to retrieve all global motions to merge maps from a set of relative motions. To make it possible, this approach adopts its first step, the pair-wise map merging method to estimate relative motions for grid maps. A graph-based sampling scheme is utilized to remove unreliable relative motions obtained from the pair-wise map merging. Finally, the accurate global motions can be recovered from the set of reliable relative motions by the motion averaging. The proposed experiments were implemented in MATLAB on a 3.6 GHz computer and four cores with 8 GB of memory. The local grid maps to perform the tests were obtained by applying SLAM algorithms executed by multiple robots. Experimental results demonstrate that the proposed approach can simultaneously merge multiple grid maps with good performances. One of the problems with the approach, according to the author, is that if the map has low overlap percentages with other maps, there is no way to integrate them. Also, according to the author, most approaches proposed so far share this limitation.

4.3 Summary And Discussions

This thesis presents a project for the development and evaluation of a cooperative navigation system for multiple UAVs. The project is based on an integrated approach, the proposed system has a distributed architecture, where all members of the cooperative system run the same software. The basis of the proposed system is divided into three main activities, which are the construction of local three-dimensional occupancy grid maps, the sharing of these maps among members of the cooperative system, and, later, the fusion of local maps in each of the UAVs.

To compare the system presented in this thesis with the related works described above, eight defined criteria were used, which are: vision sensor, GPS, map type, map sharing, map merging, architecture, environment, and vehicle type. Table 2 presents in detail the comparison of this proposal with related works. In this comparative table, the related works are presented sorted by publication date, in ascending order, where the most recent works are at the end of the table.

The first criterion refers to the type of vision sensor used. Choosing this criterion implies defining the level of precision, detail, and cost of the application. Generally, related applications that use laser sensors have greater accuracy compared to those using RGB-D cameras and even higher when compared to those using RGB cameras. Laser sensors are generally more expensive and cannot be carried by small UAVs. Furthermore, they have moving parts in their structure, which can contribute to noise in the estimation of the UAVs' pose.

Some RGB-D cameras, on the other hand, are reduced in size, with no moving parts, and can provide a depth map that can be easily converted to a point cloud, combined with a color map that makes up an image. As RGB cameras do not have this capability, data processing is required before building maps. For these reasons in this thesis, the sensor defined for use was RGB-D cameras. The only problem with this type of sensor is having the ability to capture information at a reduced angle of the environment, requiring a greater movement by the UAVs for the construction of local maps. This limitation is currently being overcome by the inclusion of more than one sensor of this type in the same drone, by some manufacturers.

The next criterion used for comparison is the use of global positioning systems (GPS) or Global Navigation Satellite System (GNSS). Although its use is beneficial for multiple UAV systems, the choice of this criterion is highly linked to the signal conditions required for its proper operation. As with most approaches, except for Vetrella; Fasano; Accardo (2016), in this thesis, the idea is that the map merging happens without any previous information about the environment or global location, so this sensor is not used. This choice allows for the application of greater flexibility, allowing the system to be used in various environments where GPS functionality is partially or unavailable.

Map type is the next criterion to be compared. This criterion defines how information from the virtual representation of the environment will be created, stored, distributed, or shared with various independent systems. The most used representations in the analyzed works are representations in the form of graphs (topological representations). This type of representation can consume less memory and lower processing load compared to occupancy grid representations.

On the other hand, occupancy grid maps represent the environment in more detail and allow mobile robots to move beyond navigation along edges or simple lines. UAV navigation systems need to be adequate so as not to affect their ability to move, that is, representations must be made in three-dimensional (3D) space. Unlike most related works, this thesis uses a virtual representation structure, for maps, the three-dimensional occupancy grids. This approach ensures a detailed representation of the environment compared to topological mapping. In this thesis, an approach is also employed to reduce the amount of data to be processed and transmitted when compared to the other approaches described.

The next criterion to be analyzed is how the works propose communication and data exchange between multiple robots. Among the works analyzed, only two are concerned with this criterion (Lázaro *et al.*, 2013; Schmuck; Chli, 2017). The work Schmuck; Chli (2017) presents the use of the ROS infrastructure to exchange information between robots. This practice may not be the most suitable for this type of application as it can overload the communication channel with an enormous amount of information that is often just discarded. On the other hand, the work Lázaro *et al.* (2013) proposes a kind of compression for exchanging pose graph information between robots. This approach is more suitable and ensures efficiency in this type of application. In this thesis, a map compression technique combined with a map transmission protocol is used so that the communication between the robots takes place as efficiently as possible. In this approach, maps are shared between two or more UAVs whenever they are within a communication range.

The map merging technique is the next criterion to be analyzed. Based on the works discussed above, it is possible to verify that the map merging is an open problem, with multiple possible solutions, but with issues not yet addressed. Some authors mention that the inclusion of three-dimensional information can make the application more robust (Velásquez Hernández; Prieto Ortiz, 2020). Still, the fusion algorithms based on image processing techniques generally present a better performance when compared to the other applications observed. Based on the study of the related literature, this thesis explores in depth the problem of merging 3D occupancy maps, proposing solutions and adjustments in state-of-the-art algorithms to achieve the objectives of the proposal.

Another criterion used in comparison with the state of the art is the type of architecture. This criterion implies how the system is implemented. Centralized systems concentrate the entire processing load on a single vehicle or even a server. This type of architecture generally has a high computational capacity but requires an always-on com-

munication system. Another type of architecture is decentralized, which aims to divide the processing load between one or more mobile robots connected in the same network. In this thesis, the architecture used is distributed, where all UAVs will run the same software, and data will be shared when one or more UAVs are in a communication range. This approach reduces the required communication rate and distributes the processing load among the UAVs.

The type of environment in which the system will be designed to function is also a determining factor. The external environments, in theory, have more noise and are disorganized and unstructured. Indoor environments can have controlled brightness, good navigability, and structures with known geometric characteristics. The main idea of this thesis is to produce a navigation system invariant to the type of environment.

As a final comparison criterion, the type and multiplicity of vehicles the system is designed to work with. Single robot systems are less prone to communication problems, generally having the most current version of the virtual representation in memory. Systems of multiple robots can, in turn, distribute the processing load, perform fractional tasks, split them up, and execute them faster. On the other hand, they are more prone to data synchronization and updating issues. The type of robot is a factor that directly affects navigation applications. Unmanned ground vehicles (UGVs) can charge more robust sensors, maintain their pose without wasting energy, and charge a large number of batteries or fuels.

On the other hand, aerial vehicles can move in any direction, overcome obstacles such as stairs, and access different floors and structures with ease of effort. However, they are subject to controlled requirements, limited battery, and less accurate sensors. Furthermore, these systems have six degrees of freedom, to have errors, just taking into account their mobility model. Multiple robot systems are still subject to more frequent collisions with each other or with the environment in which they operate. The proposal presented in this article is based on the development of a cooperative navigation system for multiple UAVs, directly benefiting from the mobility characteristics of this type of robot, but having to deal with the disadvantages presented with caution.

Table 2 – Comparison with related works.

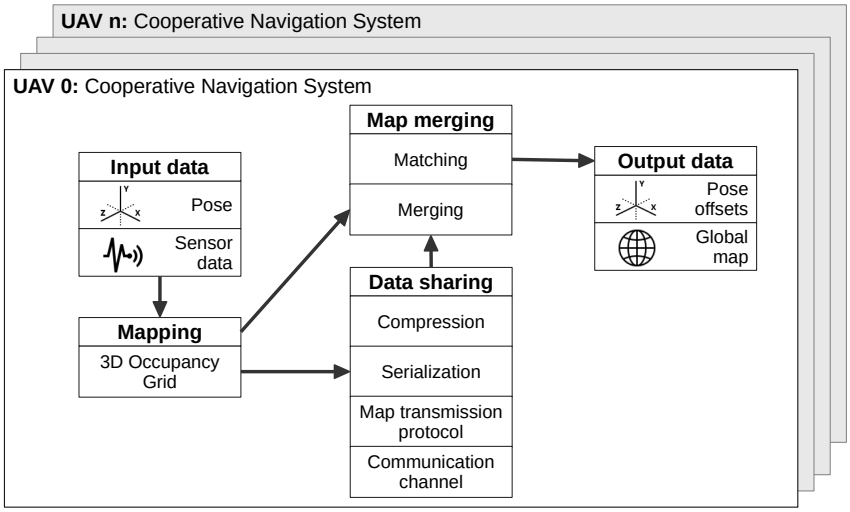
Reference	Vision Sensor	GPS	Map Type	Map Sharing	Map Merging	Architecture	Environment	Vehicle
Birk; Carpin (2006)	-	No	2D Occupancy Grid	-	Similarity metric and a stochastic search algorithm	Centralized	Indoor	Multiple UGVs
Fox *et al.* (2006)	Laser	No	2D Occupancy Grid	-	Based on general constraint graphs	Distributed	Indoor	Multiple UGVs
ARTIEDA *et al.* (2009)	RGB Camera	No	3D Graph	-	-	-	Outdoor	UAV
BACHRACH *et al.* (2011)	Laser	No	3D Graph	-	-	Centralized	Indoor, Outdoor	UAV
Dinnissen; Givigi; Schwartz (2012)	-	No	-	-	Reinforcement Learning	Centralized	Indoor	Multiple UGVs
MICHAEL *et al.* (2012)	Laser, RGB-D Camera	No	3D Occupancy Grid	-	Based on an initialization point	Centralized	Indoor	UAV, UGV
CHOWDHARY *et al.* (2013)	RGB Camera	No	Feature Point Database	-	-	Centralized	Indoor, Outdoor	UAV
WANG *et al.* (2013)	RGB Camera	No	3D Graph	-	-	Centralized	Outdoor	UAV
Lázaro *et al.* (2013)	Laser	No	3D Graph	Wireless ad-hoc network	Condensed measurements	Distributed	Indoor	Multiple UGVs
SAEEDI *et al.* (2014)	Laser	No	2D Occupancy grid	-	Hough peak matching	Centralized	Indoor	Multiple UGVs
Pi *et al.* (2014)	Stereo RGB Camera	No	3D Graph	-	-	-	Underwater	AUV
Zhou *et al.* (2015)	RGB Camera	No	3D Graph	-	-	-	Indoor	-
Lu; Song (2015)	RGB Camera	No	3D Graph	-	-	-	Outdoor	Multiple UGVs
LENAC *et al.* (2016)	Laser, RGB-D Camera	No	2D Graph, 2D Occupancy Grid	-	-	Centralized	Indoor	UGV
KOCH *et al.* (2016)	2D Laser	No	2D Occupancy Grid	-	Signed distance functions	Centralized	Indoor	UGV
Vetrella; Fasano; Accardo (2016)	RGB Camera	Yes	-	-	-	Centralized	Outdoor	Multiple UAVs
PARK *et al.* (2016)	Laser	No	2D Occupancy Grid	-	Connections of maximum empty rectangles	Centralized	Indoor	UGV
LEE; ROH; LEE (2016)	Laser	No	2D Occupancy Grid	-	Multi-hypothesis, synogram-based, PSO	Centralized	Indoor	Multiple UGVs
CHOUDHARY *et al.* (2017)	RGB-D Camera	No	3D Object Graph	-	-	Distributed	Indoor	Multiple UGVs
JIAN *et al.* (2017)	RPLIDAR	No	2D Occupancy Grid	-	Improvement of the ORB algorithm	Centralized	Indoor	UGV
Meng *et al.* (2017)	RGB Camera	No	-	-	-	Decentralized	Outdoor	Multiple UAVs
GUO *et al.* (2017)	-	No	-	-	-	Decentralized	Outdoor	Multiple UAVs
URZUA; MUNGUíA; GRAU (2017)	RGB Camera	No	3D Graph	-	-	Centralized	Outdoor	UAV
BONANNI; DELLA CORTE; GRISETTI (2017)	2D laser, depth-camera	No	3D Graph	-	Using a reference map	Centralized	Indoor, Outdoor	Multiple UGVs
FERRãO; VINHAL; DA CRUZ (2017)	-	No	2D Occupancy Grid	-	SIFT	Centralized	Indoor	-
Schmuck; Chli (2017)	RGB Camera	No	3D Graph	ROS infrastructure	Computing transform among two key frames	Centralized	Indoor, Outdoor	Multiple UAVs
YUE *et al.* (2018); Yue *et al.* (2018)	Laser	No	3D Occupancy Grid	-	Hierarchical probabilistic solution	Centralized	Indoor, Outdoor	Multiple UGVs
TRUJILLO *et al.* (2018)	RGB Camera	No	3D Graph	-	-	Centralized	-	Multiple UAVs
PEREZ-GRAU *et al.* (2018)	RGB-D Camera	No	3D Occupancy Grid	-	-	Centralized	Indoor	UAV
JIANG *et al.* (2019)	-	No	2D Occupancy Grid	-	Based on global and relative motions	Centralized	-	UGV
Qin *et al.* (2019)	Laser, Stereo Camera	No	2.5D, 3D Occupancy Grid	-	-	Centralized	Indoor	UAV, UGV
Tang *et al.* (2019)	Laser, RGB Camera, Thermal Camera	No	2D Occupancy Grid	-	-	Decentralized	Indoor, Outdoor	Multiple UAVs
Velásquez Hernández; Prieto Ortiz (2020)	Laser	No	2D Occupancy Grid	-	Corner detector, cylindrical descriptor, RANSAC	Centralized	Indoor	Multiple UGVs
BOKOVOY; MURAVIEV; YAKOVLEV (2020)	Monocular, Stereo and RGB-D cameras	No	2D Occupancy Grid, 3D Point Clouds	-	Comparison of some algorithms	Centralized	Indoor	UGV, UAV
This thesis	**RGB-D Camera**	**No**	**3D Dynamic Occupancy Grid**	**MTP protocol over TCP sockets**	**FAST3D, BRIEF3D, RANSAC and Homography**	**Distributed**	**Indoor, Outdoor**	**Multiple UAVs**

(-) Not applicable or unavailable information.

5 COOPERATIVE NAVIGATION SYSTEM ARCHITECTURE OVERVIEW

This thesis demonstrates a complete approach from the development to the evaluation of a cooperative navigation system suitable for use by multiple UAVs. The system has a distributed architecture, where all UAV components of the cooperative navigation system run the same software. Figure 14 demonstrates the proposed navigation system architecture.

Figure 14 – Architecture of proposed cooperative navigation system for multiple UAVs.



Source: author.

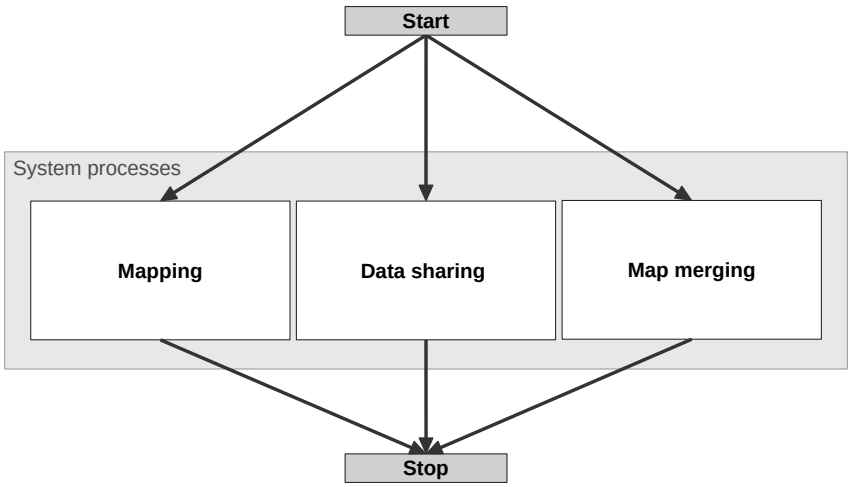
In this proposed architecture, all UAVs have additional hardware, an embedded computer, an RGB-D camera, and a communication channel used for data synchronization between vehicles. The embedded computer is responsible for running all the software modules of the proposed system. In addition, a connection is made between the proposed software and the flight controller to acquire UAV positioning estimates (odometry) and send commands to mission control. The RGB-D camera is responsible for acquiring the depth point clouds that will be used as input for the navigation algorithms.

In total, each UAV is equipped with three distinct proposed software modules. The first is the mapping module described in Section 5.1, responsible for building the maps

based on the readings obtained from the UAV sensors. The second module is the data sharing module, described in Section 5.2, which aims to efficiently distribute local maps between vehicles within the communication range. The third and last module is the map merging module, presented in Section 5.3, which is responsible for merging local maps to generate maps with global representation within each UAV.

The execution of the proposed system in each of the UAVs generates three distinct processes, as shown in Figure 15. Each of these processes corresponds to a system module. These processes work independently and simultaneously to ensure the complete functioning of the proposed system. In addition, there is no type of centralizing element and all proposed applications can run online during the flight on each vehicle's embedded hardware.

Figure 15 – System processes.



Source: author.

The proposed architecture is designed to work with an increasing number of UAVs. Furthermore, it is fully distributed, which means that each UAV by itself operates independently, but if there are other UAVs within the range of the communication channel, they tend to share their information and merge their experiences (maps). This approach prevents UAVs from repeatedly traveling in the same regions of the environment.

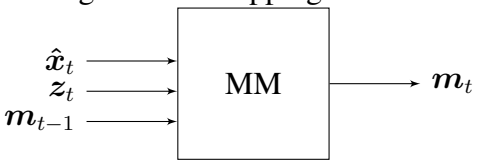
The purpose of using multiple UAVs in this architecture is to maximize area coverage and reduce the time needed to perform tasks such as exploring large areas. One of the difficulties of this type of application is the lack of signal availability of global localization systems (GPS) within certain environments. Under these conditions, estimating the parameters for merging local shared maps at runtime becomes a complex task.

The final result of using the proposed system is global map instances that allow UAVs and other mobile robots to access the regions contained in the map to perform some necessary activities related to the mission objective they are intended to accomplish.

5.1 Mapping Module

The mapping module (MM) is responsible for concentrating all the experiences and knowledge acquired by the mobile robot along its trajectory. Maps must have a certain consistency and fidelity to enable mobile robots to navigate in free spaces. Figure 16 provides a simplified diagram of the mapping module.

Figure 16 – Mapping module.



Source: author.

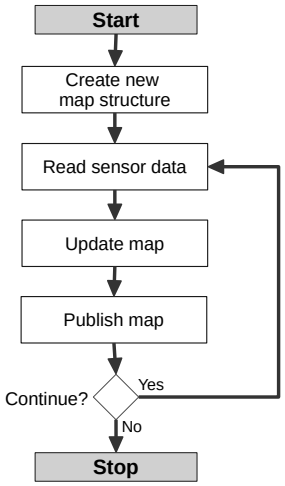
As input, the mapping module receives the best pose estimated of the robot $\hat{\boldsymbol{x}}_t$ at instant t, the readings $\boldsymbol{z}_t$ of the RGB-D sensor at time t and the last map representation $\boldsymbol{m}_{t-1}$ generated at the previous time $t-1$, if available.

From these input data, it becomes possible to construct the virtual representation of the 3D occupancy grid map $\boldsymbol{m}_t$ at instant t. The virtual representation $\boldsymbol{m}_t$, in turn, is composed of all the representations obtained previously $\boldsymbol{m}_{0:t}$ enhanced by the pose estimate $\hat{\boldsymbol{x}}_t$ of the robot.

The choice of 3D occupancy grid structures to store maps is justified by the fact that, in addition to storing environmental obstacles, it stores precious information about free areas and possible paths for navigation. The choice for the use of three-dimensional structures depends on the mobility model of these vehicles and the possibility of better describing the environments.

The process execution flow generated by the mapping module can be seen in Figure 17.

Figure 17 – Mapping module execution flow.



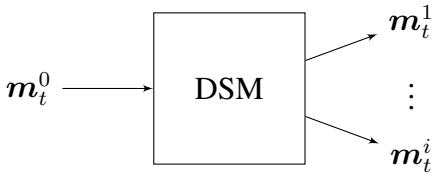
Source: author.

Upon module startup, it creates a new 3D occupancy grid map structure. After it loops throughout its runtime. In this cycle of iterations, the system obtains the readings from the sensors, updates the map, and publishes a new version of it at each new iteration. Map structures and their construction process are fully detailed in Chapter 6.

5.2 Data Sharing Module

The proposed data sharing module (DSM) aims to carry out all actions involving communication and data sharing between the UAVs of the team that make up the cooperative system. This module is of great importance for the cooperation of the proposed system, as it is through it that UAVs share their experiences (maps) and can then locate themselves concerning their neighbors. This module is responsible for providing the distributed architecture of the system, allowing the iteration between the UAVs and making them independent, that is, if an individual system fails, the set will possibly not be affected. Figure 18 shows the data sharing module diagram.

Figure 18 – Data sharing module.

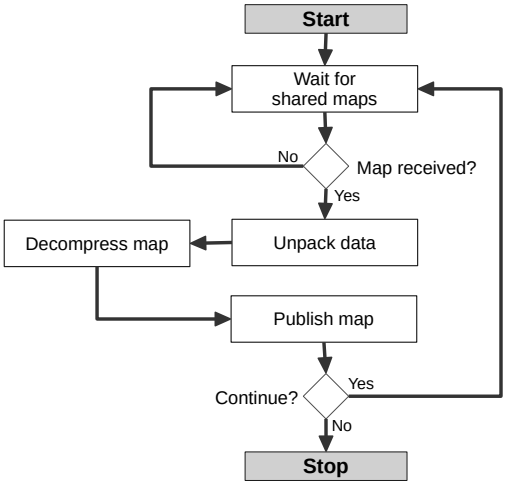


Source: author.

This module collects from the system the latest version of the map $\boldsymbol{m}_t^0$, at the instant of time t, and prepares it for sharing. When any UAV component of the cooperative system approaches within the communication range, this module detects and makes the connection with the neighbor's data sharing module. The current map is then shared. The local maps of the other vehicles $\boldsymbol{m}_t^{1:i}$ are received.

The execution process of the data sharing module then generates two new independent processes. The first process has a server responsible for receiving maps. The server execution flow is shown in Figure 19.

Figure 19 – Data sharing server execution flow.

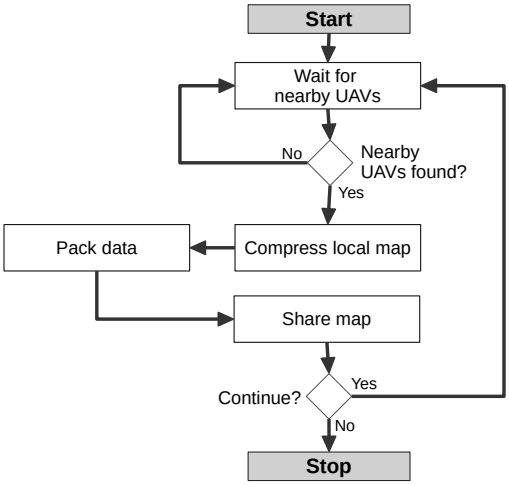


Source: author.

In this process, the system waits for shared maps, when a new map is received, the data is unpacked and the map goes through a decompression process. Afterward, the map is then published locally so that the other components of the system can access it.

The second process created is called the client process and is responsible for sending the local map to other vehicles. The execution flow of the client process is shown in Figure 20.

Figure 20 – Data sharing client execution flow.



Source: author.

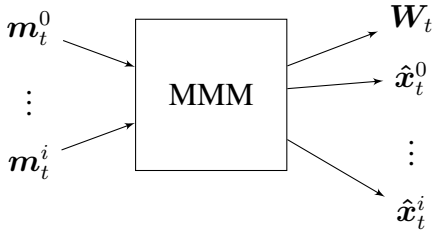
The client process waits until any connection is made with neighboring UAV servers. When the connection occurs, the local map is compressed and the data is packed. The map is then sent to this UAV. If any sharing with this vehicle has already been done, the local map is only shared if it receives any update to the previously shared map.

Map compression and serialization methods and also the transmission protocol used are fully detailed in Chapter 7.

5.3 Map Merging Module

The map merging module (MMM) is responsible for merging all local maps shared among the UAVs within the communication range. This module is present in all UAVs and the global map instance obtained as a result of this module refers to all experiences obtained by the mobile robot and the robots encountered during its journey. This way, at the end of the mission, there may be more than one global map representing different regions. The Figure 21 demonstrates the map merging module.

Figure 21 – Map merging module.

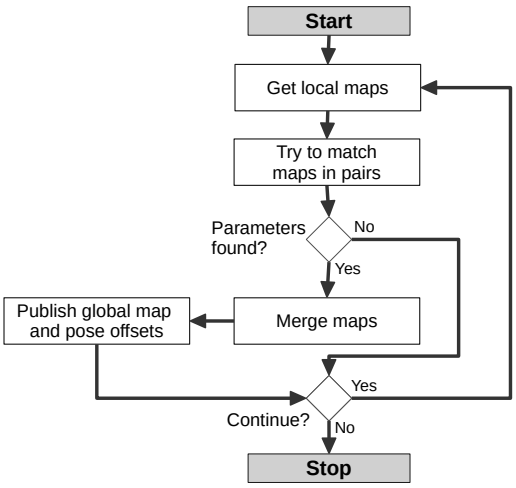


Source: author.

This module receives as input several local maps $\boldsymbol{m}_t^i$ obtained at the instant of time t by two or more UAVs identified with index i. It is assumed for this module that the map represented by the index $i = 0$ is the local map of the UAV that runs the module, thus, $\boldsymbol{m}_t = \boldsymbol{m}_t^0$. If there are no shared maps, the representation of the local map is the same as the global map, that is, $\boldsymbol{m}_t^0 = \boldsymbol{W}_t$. If there are shared maps, the module result is corresponding to the merge of all local maps whenever possible, so $\boldsymbol{m}_t^{0:i} = \boldsymbol{W}_t$. In addition to the global map, as a result, this module publishes the pose estimate $\hat{\boldsymbol{x}}_t^i$ of each of the UAVs to the global map. This result allows the other components of the system to identify the position of the other UAVs to their position.

The process execution flow generated by the map merging module can be seen in Figure 22.

Figure 22 – Map merging module execution flow.



Source: author.

After starting the map sharing module, all available local maps are collected. Then the algorithm tries to find the merge parameters through matching map pairs. If the respective parameters are found, the local maps are merged and the global map and pose estimates called offsets to the global map for each of the vehicles are published. The module continues its execution until the end of the mission.

The full description of the map merging module, including match and merge algorithms and the entire map merging system structure is available in Chapter 8.

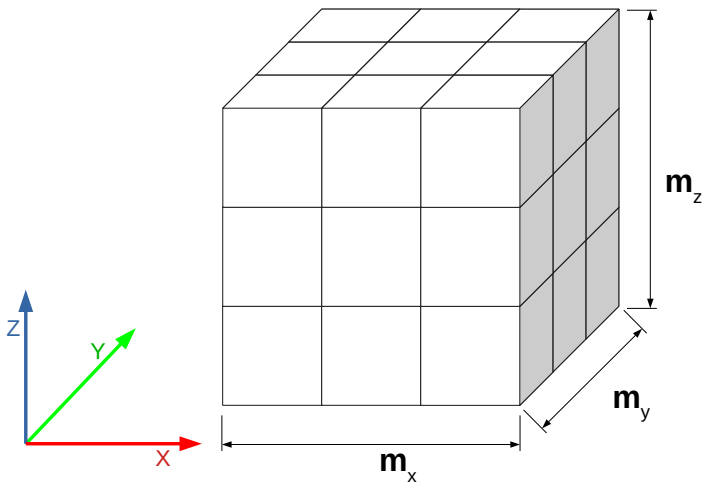
6 3D OCCUPANCY GRID MAPPING

In this thesis, 3D occupancy grid maps are generated from point clouds obtained from RGB-D cameras coupled in front of each of the UAVs. Section 6.1 introduces the concept of dynamic maps and demonstrates in detail the structures that make up the 3D dynamic occupancy grid maps. Section 6.2 demonstrates how 3D dynamic occupancy grid maps are allocated in memory. Finally, Section 6.3 explains in detail how the construction of maps takes place and presents the algorithms used.

6.1 3D Dynamic Occupancy Grid Maps

Occupancy grid maps are structures used to store the experiences collected from the environment by mobile robots during their exploratory navigation process through an unknown environment. In this thesis, these maps are produced in three dimensions to adapt to the mobility model provided by the UAVs, enabling navigation both horizontally and vertically. The basic structure of this type of map can be seen in Figure 23.

Figure 23 – The structure of 3D dynamic occupancy grid maps.

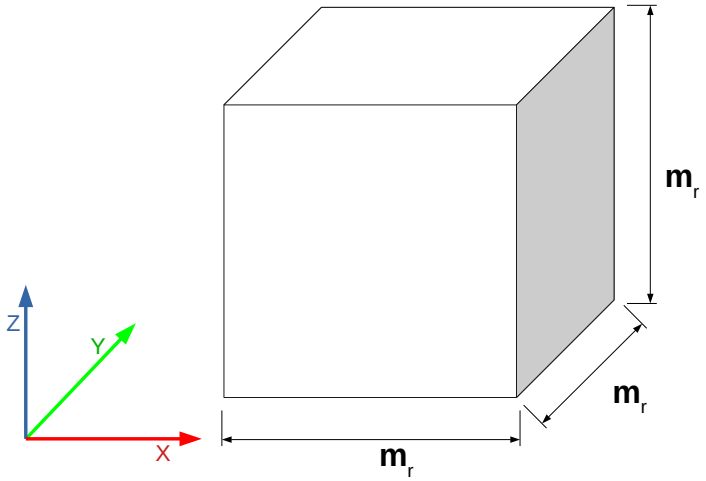


Source: author.

The cubic structure of this map $\boldsymbol{m}$ is conditioned by the variables $\boldsymbol{m}_x$, $\boldsymbol{m}_y$ and $\boldsymbol{m}_z$

representing the size of the map in the dimensions X, Y and Z, respectively. Each portion of this virtual representation of the environment is called a cell and its representation can be seen in Figure 24.

Figure 24 – The 3D dynamic occupancy grid map cell representation.



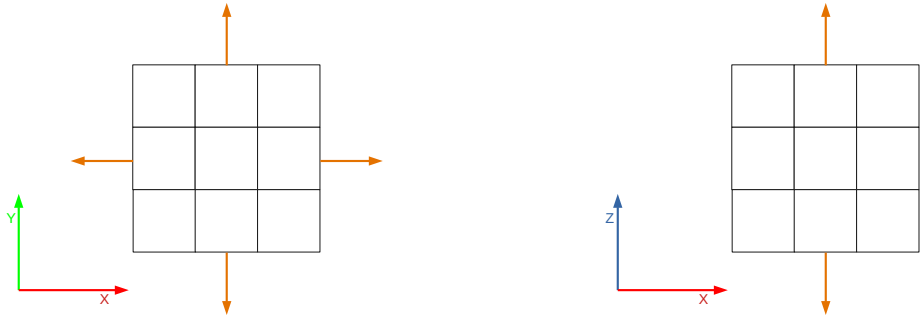
Source: author.

The cells that make up the map have a size defined by the resolution of the map, expressed by the parameter $\boldsymbol{m}_r$. Each $\boldsymbol{c}_{xyz}$ cell represents the state of the environment at the location x, y, and z in the map coordinate space. Possible values assumed by $\boldsymbol{c}_{xyz}$ can be: occupancy probability p, which can range from 0 (free space) to 100 (occupied space); -1 to represent an unknown space; -2 representing an inaccessible space.

To fix coordinate system references, in this thesis, the concept of robot pose as a world center is adopted, which places the robot in the map world coordinate system center during the initialization of the map construction activity.

This thesis uses the concept of dynamic structures of 3D occupancy grid maps. Here the "dynamic" concept refers to the structures that store the maps and not necessarily to dynamic mapping. These maps can dynamically change their size as shown in Figure 25.

Figure 25 – The possible map size increments.



a) Horizontal size increments. b) Vertical size increments.

Source: author.

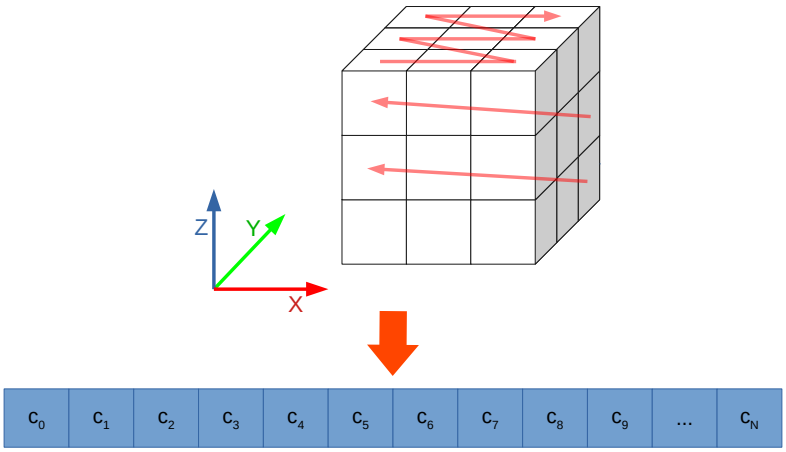
The size control of dynamic maps is done in two ways, horizontally and vertically, always respecting a safety margin, in the number of cells, defined by the parameter $\boldsymbol{m}_{sm}$. The size change occurs whenever any of the other parts of the proposed software try to access or enter information outside the regions supported by the map. The horizontal size control has a fixed increment for the dimensions X and Y to ensure that the robot's starting point is always the center of the map.

Also, maps with dynamic structures can transform (rotate and translate), merge with other maps (merge), quickly copy their content, and shrink their dimensions after a transformation process is applied and many free regions remain on the map. These maps even can compress and generate representations with relatively small sizes. These features will be explored with a greater level of detail in the course of the text of this thesis.

6.2 Map Storage

In this thesis, the 3D dynamic occupancy grid maps are stored in memory as one-dimensional (1D) structures through the use of an indexing method known as Row-major order (LEE *et al.*, 2005). A graphical representation of this structure conversion can be seen in Figure 26.

Figure 26 – The representation of map storage structures.



Source: author.

In this 1D structure, the index i corresponding to each cell $\boldsymbol{c}_i$ is obtained by (15).

$$i = z + \boldsymbol{m}_z * (y + \boldsymbol{m}_y * x) \tag{15}$$

where x, y and z represent the coordinates of the cell in three-dimensional space and $\boldsymbol{m}_y$ and $\boldsymbol{m}_z$ represent the size of the dimensions Y and Z of the three-dimensional map respectively.

The maximum size of the 1D structure is defined by N and obtained using (16).

$$N = \boldsymbol{m}_x * \boldsymbol{m}_y * \boldsymbol{m}_z \tag{16}$$

6.3 Map Construction

The experiences collected by UAVs during their trajectory are stored in the form of three-dimensional maps built using the Algorithm 1.

Algorithm 1 3D Mapping Algorithm

procedure $Mapping3D(\boldsymbol{m}_{t-1}, \boldsymbol{m}_r, \boldsymbol{m}_d, \hat{\boldsymbol{x}}_t, \boldsymbol{z}_t)$
  $\boldsymbol{m}_t \leftarrow \boldsymbol{m}_{t-1}$
  Rescale $\hat{\boldsymbol{x}}_t$ and $\boldsymbol{z}_t^i$ according to $\boldsymbol{m}_r$
  for $\boldsymbol{z}_t^i$ in $\boldsymbol{z}_t$ **do**
    Create a line $l = (\hat{\boldsymbol{x}}_t, \boldsymbol{z}_t^i)$
    Translate l to map center
    Rotate l to robot orientation $\hat{\boldsymbol{x}}_t$
    Compute all cells in l, $\boldsymbol{c} = BresenhamLine(l)$
    for $\boldsymbol{c}_j$ in $\boldsymbol{c}$ **do**
      $\boldsymbol{m}_{t\boldsymbol{c}_j} = HIMM(\boldsymbol{m}_{t\boldsymbol{c}_j}, \boldsymbol{m}_d)$
    end for
  end for
  return $\boldsymbol{m}_t$
end procedure

As input, this algorithm receives the latest available map version $\boldsymbol{m}_{t-1}$, the map resolution $\boldsymbol{m}_r$ in meters, the region distance parameter $\boldsymbol{m}_d$, the best estimate of the robot pose $\hat{\boldsymbol{x}}_t$, in six dimensions (6D), and a point cloud $\boldsymbol{z}_t$ obtained through an RGB-D camera.

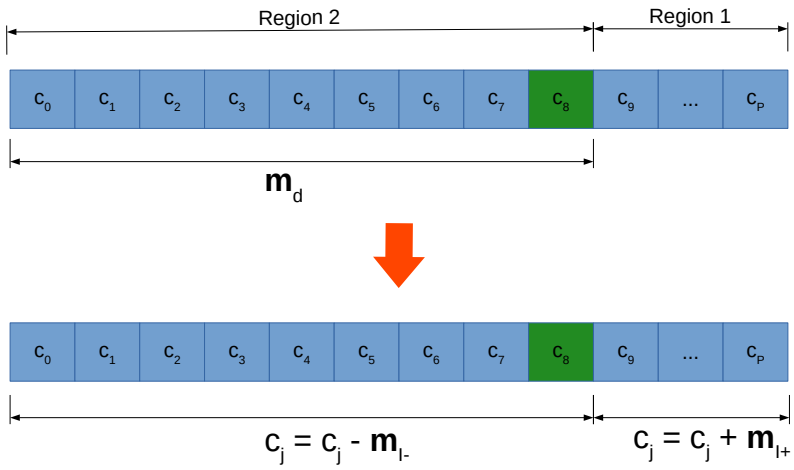
First the algorithm initializes the current version of the map $\boldsymbol{m}_t$ with the previous map $\boldsymbol{m}_{t-1}$. Then it applies a scale conversion to the robot pose $\hat{\boldsymbol{x}}_t$ and the readings $\boldsymbol{z}_t$ obtained using the resolution of the map $\boldsymbol{m}_r$.

Afterward, for each point of the obtained point cloud, a straight line l is constructed, where the origin point is the pose of the robot $\hat{\boldsymbol{x}}_t$ and, the destination point is the point of the point cloud $\boldsymbol{z}_t^i$. A transformation consisting of rotation and translation is applied to each of the lines, transforming them to the coordinate space of the map.

Then, Bresenham's line algorithm (BRESENHAM, 1965) is used to calculate the set of map cells $\boldsymbol{c}$ corresponding to the line l. For each one of the cells of the set $\boldsymbol{c}$, the Histogramic In Motion Mapping (HIMM) algorithm (BORENSTEIN; KOREN *et al.*, 1991b) is adapted and used to update the value corresponding to each cell. As input, this algorithm receives the cell of the line to have its value modified $\boldsymbol{m}_{t\boldsymbol{c}_j}$ and the parameter

for defining the distance between regions m_d. The operation of the HIMM algorithm can be seen in Figure 27.

Figure 27 – The HIMM algorithm operation.



Source: author.

If the distance from the analyzed cell c_j to the robot pose c_0 is greater than the parameter defined by m_d, this region is defined as region 1 otherwise it is defined as region 2. All cells classified as region 1 receive an increment as defined in (17) and cells defined as region 2 receive a decrement as defined in (18).

$$c_j = c_j + m_{I+} \tag{17}$$

$$c_j = c_j - m_{I-} \tag{18}$$

where m_{I+} is the increment parameter and takes the value 10, and m_{I-} is the decrement parameter and receives the value 30.

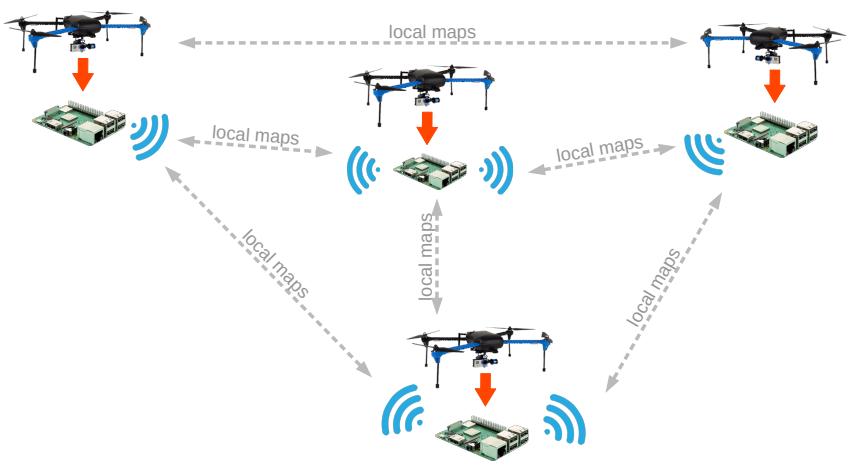
At the end of the operation, the mapping algorithm returns the current map m_t so that it can be used by other components of the proposed system. In addition to the mapping activity, a simple gap-filling algorithm is performed to fill isolated cells that can have their values defined based on their neighbors, to guarantee the integrity and minimum quality of the generated maps.

The mapping process is performed iteratively during the entire trajectory traveled by the UAVs and gives rise to local occupancy grid maps in three dimensions.

7 SHARING 3D OCCUPANCY GRID MAPS

In this thesis, there is a need to share the 3D local occupancy grid maps produced by each UAV component of the cooperative navigation system. Figure 28 demonstrates the scenario in which four vehicles exchange their local maps to improve their navigation experiences.

Figure 28 – Data sharing scenario.



Source: author.

In this scenario, maps are shared among vehicles whenever they receive new map updates and two or more UAVs are in the communication range. In this way, the maps can be merged in a distributed way and each of the UAVs in the network will have, at the end of their mission, a version of the global map composed of their experiences in combination with the experiences of the vehicles they encountered during their trajectory.

Data transmission and network management between mobile nodes have been frequently explored in recent research (Dapper e Silva *et al.*, 2019). Even so, it was not yet possible to share maps efficiently and with a reduced computational cost between vehicles. For this reason, this thesis proposes a set of algorithms that, when used together, enable the sharing of these maps in the cooperative UAV network.

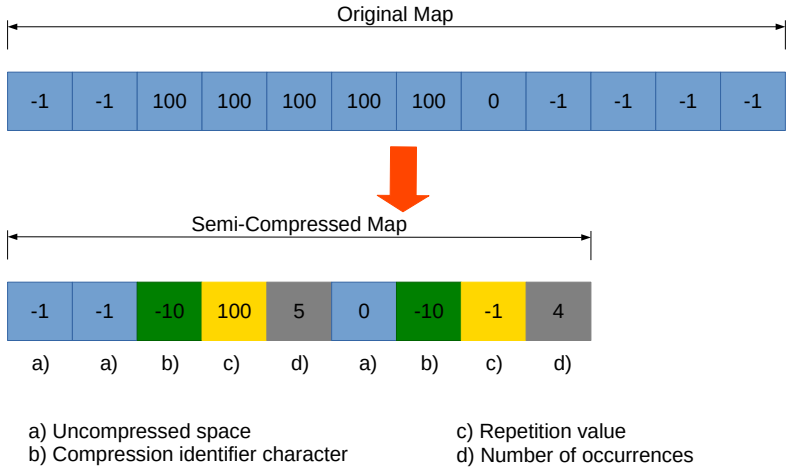
In this system, maps undergo a two-level compression process presented in Section

7.1. They are then serialized using a proposed protocol presented in Section 7.2. Then a map transmission protocol (Section 7.3.) is used to share the maps in a fully distributed way among the UAVs.

7.1 Map Compression

To significantly reduce the size of the maps and to speed up the sharing process, the maps are subjected to a compression process. Different from conventional compression methods, the proposed algorithm was specially designed for the compression of 3D occupancy grid maps. The proposed compression process is divided into two stages. The first step is occurrences compression and is shown in Figure 29.

Figure 29 – First step: occurrences compression.



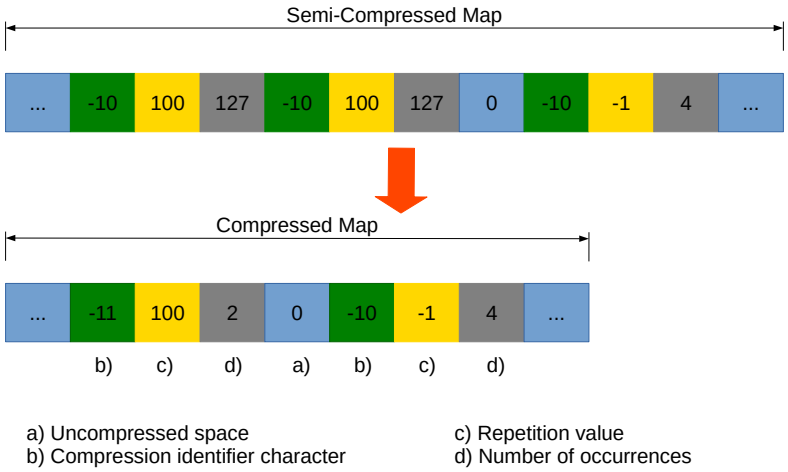
Source: author.

In this step, the one-dimensional vector of map data goes through the semi-compression process. In this process, all occurrences of elements are identified and stabilized. All elements that have three or more occurrences are replaced by a set of three distinct elements that represent the occurrences.

The first element of the set, represented in Figure 29 b, is the occurrence identifier and has the initial value -10. The second element of the set, represented in Figure 29 c, is the equivalent of the repeated element's value. The third element of the set, represented in Figure 29 d, is equivalent to the number of repetitions of the element of the occurrence in question. The maximum number is limited by the type of data used. In this case, the maximum number of repetitions is 127 because the data type used corresponds to an eight-bit signed integer. If the number of occurrences is greater than the maximum number allowed, two or more sets of occurrence representations are used for the representation.

The second step is sequence compression and is represented in Figure 30.

Figure 30 – Second step: sequences compression.



Source: author.

In this step, the semi-compressed map is subjected to compression again. First, all sequences of sets of occurrences are identified. Then the accounting of all sets in the sequence is performed. Then all sequences of identified sets of occurrences, which have two or more repetitions, are replaced by a new set of three representative elements.

The first element of this set, represented in Figure 30 b, is the sequence identifier, which is decremented to the value -11. The second element of the set, represented in Figure 30 c, is the equivalent of the repeated element's value. The third element of the set, represented in Figure 30 d, is equivalent to the number of element repetitions of the sequence in question. As in the previous step, the maximum number of repetitions is limited by the type of data used. If the number of sequence repetitions is greater than the maximum number allowed, two or more sets of sequence representations are used for representation.

Then new iterations are performed to compress sequences. At each new iteration, when substitutions occur, the sequence identifier is decremented until reaching the minimum value. This process is then repeated until replacements are no longer possible.

After these two steps, the content of the maps is compressed and ready to be shared. The contents of the compressed map cells cannot be accessed directly. For this, after sharing, the maps are decompressed with an algorithm that reconstructs the map content by replacing all the repetition sets with values corresponding to the original map. The operation of the decompression algorithm is equivalent to the inverse process of the compression.

The original map and the uncompressed map are identical, both in size and content. Therefore, the proposed compression processes do not affect the quality of the maps in any way. In addition, these algorithms significantly reduce the size of maps and help speed up the information sharing process.

7.2 Map Serialization

Before transmitting maps, compressed map data and other related information need to be organized in a way that allows for sharing. For this, a serialization protocol of map information is used. The proposed serialization protocol can be seen in Figure 31.

Figure 31 – Map serialization protocol.

Serialized map

Bytes	0	1-4	5	6-9	10-13	14-17	18-21	22	23-26	27-N
Type	int8_t	int	int8_t	int	int	int	float	int8_t	int	int8_t
Description	message Type	vehicle ID	map type	x size	y size	z size	map resolution	is compressed	data size	data
Available options	0 - map		0 - 2d 1 - 3d					0 - no 1 - yes		

Source: author.

This protocol is composed of 27 bytes of information plus $N - 27$ bytes of map data. The byte order 0 refers to the message type. In this case, the value 0 is adopted for the map type. New types can be added, but in this thesis, only maps are shared.

The bytes 1 to 4 represent a 32-bit integer equivalent to the identifier of the vehicle sharing the map. The 5 byte represents the map type, which can be 0 for 2D maps and 1 for 3D maps. The 6 to 17 bytes represent three 32-bit integers that represent the size of the three dimensions of the map to be transmitted. The map resolution is represented by the 18 to 21 bytes equivalent to the float type. The byte 22 indicates whether the map is compressed or not, with the option 0 for uncompressed and 1 for compressed.

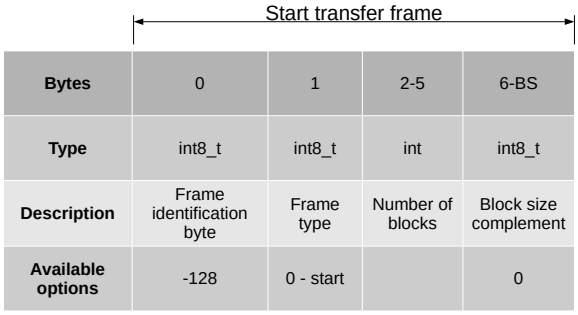
Finally, the bytes 23 to 26 represent a 32-bit integer equivalent to the size in bytes of the compressed map data array. The bytes 27 to N represent the map data to be shared, where $N - 27$ defines the size of the map data.

After serialization, the data byte array is already formatted to be shared. The information contained in the constructed message allows the reconstruction of data and facilitates the sharing of this information.

7.3 Map Transmission Protocol

With the data to be transmitted properly formatted, these maps are then shared using the proposed Map Transmission protocol (MTP). In this protocol, the data to be transmitted is allocated into three distinct message types and then sent in blocks. The first message defined in the protocol is the data transfer start frame and is represented by Figure 32.

Figure 32 – Start transfer frame.



Bytes	0	1	2-5	6-BS
Type	int8_t	int8_t	int	int8_t
Description	Frame identification byte	Frame type	Number of blocks	Block size complement
Available options	-128	0 - start		0

BS – Message block size

Source: author.

The purpose of this message is to signal the initiation of a map transfer. The message is made up of BS bytes, where BS is the size of the message block predefined as a parameter. The order byte 0 represents the identification character of a frame and has a value equal to 128. This value is not used by the compression process nor for the serialization process, being exclusive for the identification of a data transfer frame. The 1 order byte identifies the frame type, which in this case receives the value 0, signaling a start frame. The bytes 2 to 5 represent a 32-bit integer equivalent to the number of blocks to be transferred. The 6 to BS bytes are padded with zeros to build a message with defined block size.

The second message is a data frame and is represented by Figure 33.

Figure 33 – Data frame.

Bytes	0 - BS
Type	int8_t
Description	Data
Available options	-127 to 127

BS – Message block size

Source: author.

In this message, bytes 0 to BS are filled with the data formatted in the map serialization process. This message is replicated until all data is covered. If the last frame does not have the correct size BS, the remaining bytes are padded with zeros and ignored by

the receiver.

The last message signals the end of map transfer and is represented by Figure 34.

Figure 34 – Stop transfer frame.

Stop transfer frame

Bytes	0	1	2-5	6-BS
Type	int8_t	int8_t	int	int8_t
Description	Frame identification byte	Frame type	Number of transferred blocks	Block size complement
Available options	-128	1 - stop		0

BS – Message block size

Source: author.

This message has a structure similar to the initialization message, but the byte order 1 that signals the frame type receives the value 1, indicating the end of a transfer. The bytes 2 to 5 represent a 32-bit integer equivalent to the number of blocks transferred.

Messages formatted according to the proposed protocol are then forwarded over a TCP/IP connection via Sockets. It is worth noting that as this proposed protocol is of high level, message delivery is guaranteed by the TCP/IP protocol, with no need to include headers, for example, in data frames.

Each of the UAVs has a socket server responsible for receiving maps from its neighbors. Furthermore, the defined range of IP addresses is scanned respecting a pre-defined period to discover available servers. When a neighboring socket server is discovered, the map sharing process is started in a new thread.

When the stop message is received, the size of blocks contained in the start message and the stop message is compared, if these sizes are equal, the protocol can reconstruct the vector of serialized data.

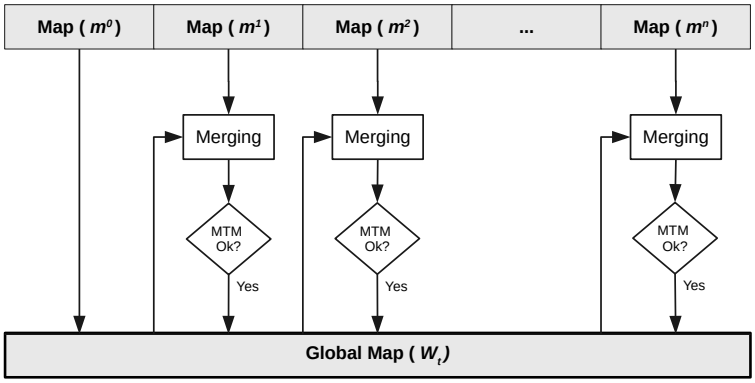
Finally, the messages are unpacked and the map in question is uncompressed, if necessary. After that, the received local maps are available for the local merge process to be applied.

8 3D OCCUPANCY GRID MAP MERGING

In this thesis, the local 3D occupancy grid maps collected from other vehicles that compose the cooperative navigation system are merged in each embedded system of each vehicle. For that, this system uses a technique of merging maps in pairs (Birk; Carpin, 2006). The purpose of this map merging system is to incorporate the collective navigation experiences with the local experiences of each vehicle, making the mapping process faster and more robust as it is performed in a distributed manner among several vehicles.

Figure 35 demonstrates the execution flow of merging several 3D occupancy grid maps using the pairwise merging technique in each of the vehicles that compose the cooperative navigation system.

Figure 35 – Pairwise map merging flow.



Source: author.

First, a list of local maps $\boldsymbol{m}_t$ is obtained at instant t of the cooperative navigation system. Each map in this list is represented by $\boldsymbol{m}_t^i$ indicating that it was collected from vehicle i. This way, $\boldsymbol{m}_t^0$ represents the map of the vehicle that is performing the merging process and is defined as a reference to the center of coordinates, by directly incorporating it into the global map $\boldsymbol{W}_t$ defined at instant t.

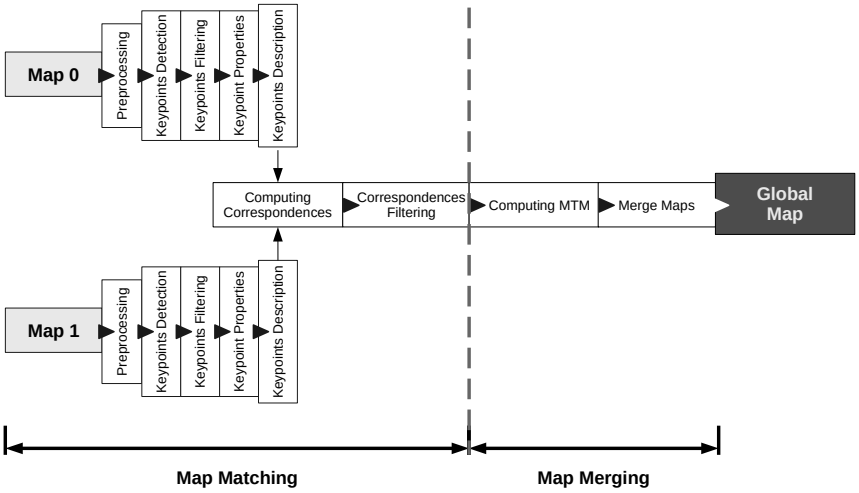
After this process, the current global map $\boldsymbol{W}_t$ is merged with the local map $\boldsymbol{m}_t^1$ which

is subsequent in the local maps list. If the merge process finds a valid Map Transformation Matrix (MTM), $\boldsymbol{m}_t^1$ is merged into $\boldsymbol{W}_t$ and the process repeats for all local maps collected.

The merging process occurs iteratively, and can merge maps by a sort of region inheritance. Taking for example a scenario with 3 UAVs, where it can have common areas between the maps $\boldsymbol{m}_t^0$ and $\boldsymbol{m}_t^1$, and between $\boldsymbol{m}_t^1$ and $\boldsymbol{m}_t^2$, but not between $\boldsymbol{m}_t^0$ and the $\boldsymbol{m}_t^2$. However, when merging $\boldsymbol{m}_t^0$ with $\boldsymbol{m}_t^1$, and $\boldsymbol{m}_t^1$ with $\boldsymbol{m}_t^2$, after the first merge, $\boldsymbol{m}_t^0$ and $\boldsymbol{m}_t^2$ will have common areas "inherited" from the first merge and from that point, they will all have common areas. In this sense, there is no need to have common areas on all maps.

Figure 36 demonstrates in a simplified way the proposed map merging system applied for each pair of 3D occupancy grid maps.

Figure 36 – Proposed map merging system.



Source: author.

The pairwise map merging is divided into two steps. The first step is Map Matching. At this stage, the pair of local maps undergo a preprocessing process for noise removal and size reduction to accelerate and turn the system more robust. The map preprocessing process is detailed in Section 8.1. Afterward, a process to detect the keypoints is applied (Section 8.2). The keypoints are then filtered (Section 8.3) and their fundamental properties are obtained (Section 8.4). The keypoints are then described (Section 8.5). The correspondences between the keypoints of the two maps are computed (Section 8.6) and then pass through a filter to remove false positives (Section 8.7), completing the first step of the merge.

After, the second step called Map Merging is started. In this step, the MTM is obtained from the set of best correspondences obtained at the end of the Map Matching step (Section 8.8). Finally, the local maps are combined following the rules described in Section 8.9.

8.1 Map Preprocessing

As in video and image processing, the preprocessing process aims to reduce noise, improving the performance and reliability of the results of the subsequent steps of the proposed algorithm. In this thesis, two steps are part of the preprocessing, filtering, and size reduction.

The first step consists of filtering the map to prevent noise or any other unqualified information from compromising the results of the proposed algorithm. In this step, the maps are submitted to two filters based on dilate and erode morphological operations successively applied to each map (GONZALEZ, 2008; JONKER, 2000). The kernel of these two filters is a 3D cube with radius D_r and E_r, for dilate and erode filters, respectively, to suit the data type of the proposed maps. The radius parameters and the iteration numbers D_i and E_i of each of these filters can be configured before starting the algorithm.

After applying these filters, much of the noise information is eliminated. After that, to equalize the information contained in the cells of the maps, a Gaussian filter with a radius equal to 3 is applied to the map (GONZALEZ, 2008). This filter guarantees smooth state transitions among map cells and improves the result of later steps of the algorithm. The number of iterations G_i of the Gaussian filter can also be configured.

The second stage of the preprocessing module consists in reducing the size of the maps, to significantly reduce the computational cost of the application. The size reduction is done based on a downsampling grid filter (OPPENHEIM; SCHAFER; BUCK, 1999), which based on a radius DS_r, travels the entire length of the map averaging the cells around the current cell, generating a new representation of the map with a reduced number of cells. The radius DS_r can be configured before starting the software, and default value is $DS_r = 1$.

8.2 Keypoints Detection

To detect keypoints in 3D occupancy grid maps, this thesis proposes some adaptations to the Features from Accelerated Segment Test (FAST) algorithm (ROSTEN; DRUMMOND, 2006). The proposed modifications are presented in the Algorithm 2, and is called *FAST 3D*. The contribution is the creation of the 3D version of the algorithm by executing the quick tests proposed in the original algorithm in 2D views of three planes in the coordinate space of the maps.

The proposed algorithm receives as input parameter a 3D occupancy grid map $\boldsymbol{m}$, the maximum number of continuous circular tests n, the mask radius r, the threshold th, and the maximum radius of the neighborhood used to average the cells at a given space nr.

The functioning of the algorithm starts with the creation of the mask based on the radius r (Algorithm 2, line 2). This mask is an array that contains the coordinates of the cells that make up the circle of radius r around a cell to be analyzed. Then the step of

Algorithm 2 FAST 3D algorithm

procedure $FAST3D(\boldsymbol{m}, n, r, th, nr)$
  $mk \leftarrow createMask(r)$
  $s \leftarrow r + 1$
  $kp \leftarrow []$
  for x from 0 to $size(\boldsymbol{m}_x)$ **do**
    for y from 0 to $size(\boldsymbol{m}_y)$ **do**
      for z from 0 to $size(\boldsymbol{m}_z)$ **do**
        if $\boldsymbol{m}_{xyz} \neq$ Unknown space **then**
          $Ip \leftarrow getCellNhMean(\boldsymbol{m}, x, y, z, nr)$
          $Ixy \leftarrow []$
          $Izx \leftarrow []$
          $Iyz \leftarrow []$
          for i from 0 to $size(mk)$ **do**
            $Ixy \leftarrow getCellNhMean(\boldsymbol{m}, x{+}mk_{i0}, y{+}mk_{i1}, z{+}mk_{i2}, nr)$
            $Izx \leftarrow getCellNhMean(\boldsymbol{m}, x{+}mk_{i2}, y{+}mk_{i0}, z{+}mk_{i1}, nr)$
            $Iyz \leftarrow getCellNhMean(\boldsymbol{m}, x{+}mk_{i1}, y{+}mk_{i2}, z{+}mk_{i0}, nr)$
          end for
          if $rFAST(Ip, Ixy, s, th, n)$ **then** $kp \leftarrow \boldsymbol{m}_{xyz}$ **end if**
          else if $rFAST(Ip, Izx, s, th, n)$ **then** $kp \leftarrow \boldsymbol{m}_{xyz}$ **end if**
          else if $rFAST(Ip, Iyz, s, th, n)$ **then** $kp \leftarrow \boldsymbol{m}_{xyz}$ **end if**
        end if
      end for
    end for
  end for
  return kp
end procedure

the mask s is calculated and an array of keypoints kp is defined. Then the algorithm goes through all the cells in the map checking for cells that are not part of the unknown space in the map (Algorithm 2, line 8).

For all cells of known space, the intensity of the neighborhood of radius nh of the cell to be analyzed Ip is captured, and the intensity of the neighborhood of all cells in the map that make up the circle contained in the mask mk for the plans XY, ZX and YZ represented by I_{xy}, I_{zx} and I_{yz}, respectively.

The next step is to apply the FAST algorithm to the three planes captured in the previous step and return the list with the identified keypoints kp. With this objective, the present thesis uses an adaptation of FAST called *rFAST*, presented in the Algorithm 3. This version of the algorithm takes into account the rotation around the cell to be analyzed (cross-check test rotated).

Algorithm 3 rFAST algorithm

```
1:  procedure rFAST(Ip, I, s, th, n)
2:      for i from 0 to s do
3:          c ← 0
4:          if (I_i − Ip) > th then c ← c + 1 end if
5:          if (I_{i+s} − Ip) > th then c ← c + 1 end if
6:          if (I_{i+(s·2)} − Ip) > th then c ← c + 1 end if
7:          if (I_{i+(s·3)} − Ip) > th then c ← c + 1 end if
8:          if c ≥ 3 then
9:              c ← 0
10:             II ← concatenate(I, I)
11:             for j from 0 to size(II) do
12:                 if (II_j − Ip) > th then
13:                     c ← c + 1
14:                 else
15:                     c ← 0
16:                 end if
17:                 if c ≥ n then return true end if
18:             end for
19:             break
20:         end if
21:     end for
22:     return false
23: end procedure
```

This proposed algorithm receives as a parameter the intensity of the region to be analyzed Ip, the vector of intensities of the neighborhood of the cells in the circle I, the mask

step s, the threshold th and the minimum number of continuous tests n for identification of a keypoint.

The functioning of the proposed algorithm aims to apply the test of the FAST algorithm by rotating the cross-check to extend the check to all possible directions (Algorithm 3, lines 4 to 7). As with FAST, if at least three of the tests are true, the continuous test check applies (3, lines 8 to 20). In this step, the list I is then duplicated generating the circular list II. A count is applied to check the maximum number of tests continuity, if this count is greater than or equal to the n parameter, then the algorithm will return true for a possible keypoint, otherwise, it will return false.

After the complete execution of the proposed algorithms, a list with all keypoints kp corresponding to the map $\boldsymbol{m}$ is returned and can be used in the subsequent steps of the map merging algorithm. Another important point to be highlighted is that thanks to the separability of the proposed algorithm's operations, it can be accelerated through parallelism of processes to obtain better performance.

8.3 Keypoints Filtering

Depending on the size of the map or even the level of detail contained in this virtual representation, it is common for a high number of keypoints to be detected, which can significantly reduce the final performance of the algorithm. The solution to this problem is to apply a filter capable of selecting the most significant keypoints around each detected point of interest.

An important point to notice is that a considerable number of keypoints in the detected set kp can be considered adjacent to the same point of interest. To solve this problem and select only the keypoints that best describe each point of interest, this thesis uses the Non-maximal Suppression filter (ROSTEN; DRUMMOND, 2006), adapting it to the case of 3D occupancy grid maps.

First, the score V is defined in (19) and calculated for all keypoints from the previously detected set kp. The V score can be defined as the sum of the absolute differences among the keypoints around a point of interest.

$$V_{xyz} = \sum_{i=x-KF_r}^{x+KF_r} \sum_{j=y-KF_r}^{y+KF_r} \sum_{k=z-KF_r}^{z+KF_r} |\boldsymbol{m}_{xyz} - \boldsymbol{m}_{ijk}| \tag{19}$$

where x, y and z represent the coordinates of the current keypoint and V_{xyz} the final score value for this keypoint. The maximum radius of the cube for calculating the score around each keypoint is represented by the parameter KF_r. And, $\boldsymbol{m}_{xyz}$ represents the probability value contained in the map cell referring to the keypoint to be analyzed, and $\boldsymbol{m}_{ijk}$ the probability of the cells around it.

Then, based on the Euclidean distance between the keypoints using the maximum

radius for adjacent keypoints KF_{ar} defined as a parameter, the set of filtered keypoints fk with the highest score V within the radius KF_{ar} is obtained. KF_r and KF_{ar} are selected empirically.

The set of filtered keypoints fk has a size relatively smaller than the set kp without necessarily being less representative for the processes of the map matching step. This makes using fk in later steps to improve system performance considerably.

8.4 Keypoints Properties

For the next steps of the map matching process, the two sets of filtered keypoints obtained from the two maps in the pairwise map merging process need to be combined in some way. This process takes into account some information previously obtained from each keypoint. Considering that the position information and the occupancy probability value are not enough to carry out this combinatorial process, this thesis adds the orientation information of the point of interest as one of the properties of each keypoint.

Commonly in image processing, the *intensity centroid* method based on *image moments* is used to obtain the orientation of keypoints (RUBLEE *et al.*, 2011; ROSIN, 1999). This method is suitable for obtaining orientations in the 2D image plane, but in the case of map merging it becomes an extra processing step. Rosin in (ROSIN, 1999) also presents the method of *gradient centroid*, where it is possible to obtain the orientation based on gradients. These methods have the advantage that, unlike the intensity centroid method, no special care needs to be taken concerning light and dark points. In addition to presenting results similar to the intensity centroid method, as in mobile robotics gradients are computed for the construction of potential fields used in autonomous exploration systems, the use of these same gradients to obtain orientation becomes an ideal strategy to avoid extra computational costs.

This thesis uses two approaches to obtain the orientations that can be configured at the beginning of the software initialization. The first approach is called *global*. In this approach, the potential field of maps is iterated over the entire range of maps and can be shared with an exploration system. The second approach is called *local*. In this approach, the radius KP_r is used as a parameter for the construction of the potential field around each keypoint. In both approaches, from the local maps $\boldsymbol{m}_t^i(x, y, z)$ the gradient maps $\boldsymbol{G}_t^i(x, y, z)$ are complete or partial obtained using the Gauss-Seidel method (SASAKI, 1998).

In the next step, the gradients $\boldsymbol{G}_t^i(x, y, z)$ are used to obtain the orientations of the filtered keypoint sets fk according to (20).

$$\begin{aligned}\alpha &= atan2(\boldsymbol{G}_t^i(x,y,z-1) - \boldsymbol{G}_t^i(x,y,z+1), \boldsymbol{G}_t^i(x,y-1,z) - \boldsymbol{G}_t^i(x,y+1,z)) \\ \beta &= atan2(\boldsymbol{G}_t^i(x-1,y,z) - \boldsymbol{G}_t^i(x+1,y,z), \boldsymbol{G}_t^i(x,y,z-1) - \boldsymbol{G}_t^i(x,y,z+1)) \\ \gamma &= atan2(\boldsymbol{G}_t^i(x,y-1,z) - \boldsymbol{G}_t^i(x,y+1,z), \boldsymbol{G}_t^i(x-1,y,z) - \boldsymbol{G}_t^i(x+1,y,z))\end{aligned} \quad (20)$$

where x, y and z are the coordinates of the current keypoint and α, β and γ are the Euler angles (roll, pitch and yaw, respectively) obtained as a property of the current keypoint $O_t^i(x,y,z) = \{\alpha, \beta, \gamma\}$. Finally, the list of orientations O_t^i can be used to better describe the set of filtered keypoints fk and enable the next phases of the map matching step.

The selection of local or global approaches to use should depend on the application for which the system will be employed. If there is no intention to use the potential field for exploration, then there is no need to use the global method, avoiding having to iterate the potential field over the entire map and reducing the computational cost of the application by iterating the potential field only around the keypoints region.

8.5 Keypoints Description

The main objective of map matching is to identify the regions that have combinations in their characteristics and available information. These combinations are called keypoint correspondences. To make these combinations possible, the information obtained from the keypoints needs to be properly formatted and condensed. In this sense, a descriptor algorithm needs to be applied.

For that, this thesis proposes some adaptations to the Binary robust independent elementary features (Brief) algorithm (CALONDER *et al.*, 2010), making it compatible for use in 3D occupancy grid maps. Also, the rotation component employed in (RUBLEE *et al.*, 2011) is adopted. The adapted version proposed in this thesis is called *Steered BRIEF 3D* and is presented in the Algorithm 4. In the version proposed in this thesis, the keypoints are 3D, the average of a cubic region around the keypoint is adopted for the binary test and, the mask that applies the orientation parameters to the keypoints is also 3D.

The proposed algorithm receives by parameter the map $\boldsymbol{m}_t^i$ obtained from the vehicle i at the instant of time t, the set of filtered keypoints fk_t^i corresponding to the map i, the list of orientations O_t^i, the size of the descriptor ds in the number of binary tests, the size of the window ws used to build the sample S, the seed ss of the random point generation algorithm and the maximum radius of the neighborhood nr used to obtain the value of each test point.

First the algorithm builds the 3D random sample based on the ss seed (Algorithm 4, line 2). The sample is constructed around the origin $(0,0,0)$ in a cube of radius $ws/2$ and is composed of $ds * 2$ 3D points. Then the vector of descriptors is created d_t^i. For each keypoint k in the filtered keypoint vector fk_t^i, the rotation matrix is created based

Algorithm 4 Steered BRIEF 3D algorithm

procedure $SteeredBRIEF3D(\boldsymbol{m}_t^i, fk_t^i, O_t^i, ds, ws, ss, nr)$
  $S \leftarrow makeSample3D(ds, ws, ss)$
  $d_t^i \leftarrow []$
  for each k in fk_t^i **do**
    $R_\theta \leftarrow createRotationMatrix(O_t^i(x, y, z))$
    $s = S \cdot R_\theta$
    $kpd \leftarrow []$
    $j \leftarrow 0$
    while $j < size(s)$ **do**
      $c_1 \leftarrow getCellNhMean(\boldsymbol{m}_t^i, k_x + s_{x,j}, k_y + s_{y,j}, k_z + s_{z,j}, nr)$
      $c_2 \leftarrow getCellNhMean(\boldsymbol{m}_t^i, k_x + s_{x,j+1}, k_y + s_{y,j+1}, k_z + s_{z,j+1}, nr)$
      if $c_1 \leq c_2$ **then**
        Add 1 to kpd
      else
        Add 0 to kpd
      end if
      $j \leftarrow j + 2$
    end while
    Add kpd to d_t^i
  end for
  return d_t^i
end procedure

on the rotation parameters of the respective keypoint $O_t^i(x, y, z)$ and applies to sample S generating the transformed sample s (Algorithm 4, line 5 and 6).

Then for each pair of points in the transformed sample, obtain the result of the binary test that compares the value of the average set of cells c_1 and c_2 within a radius nr. The set of binary tests was obtained to compose the descriptor kpd representing the current keypoint and added to the list of descriptors d_t^i. Finally, the list of descriptors d_t^i is returned and contains the set of binary tests that represent the set of filtered keypoints fk_t^i. The list of descriptors d_t^i is then used to calculate the correspondences in Section 8.6.

8.6 Computing Correspondences

With d_t^0 and d_t^1 being two lists of binary descriptors obtained from the maps $\boldsymbol{m}_t^0$ and $\boldsymbol{m}_t^1$, respectively. The list of correspondences C is obtained from the comparison of the respective descriptors.

In this thesis, binary descriptors are combined using the brute force comparison method (SINGLA; GARG, 2012). The algorithm receives by parameter the limit number of the acceptable distance CC_d to validate a combination and the maximum number of comparisons per thread CC_n to which the combination process will be divided.

First the comparison loops are allocated in the $t_n = (size(d_t^0) \cdot size(d_t^1))/CC_n$ threads. The parallelization of compare loops ensures reduced RAM consumption due to memory frees at the end of each reduced scope.

Then the Hamming distance (HAMMING, 1950) is computed for each pair of descriptors. Then add to the correspondence vector C all pairs of points where the computed distance is less than or equal to the parameter CC_d. The list of correspondences C represents the possible match points between the two maps and will go through a filtering process presented in Section 8.7 to remove false-positive correspondences.

8.7 Correspondences Filtering

To build a robust keypoint correspondences filter suitable for 3D spaces, RANSAC (FISCHLER; BOLLES, 1981) and Homography (CHUM; PAJDLA; STURM, 2005) are applied together.

Homography is the relation among two views of the same point or set of points in the space. A point in 3D space can be defined as a vector with three coordinates and a scale parameter. Consider a point $x = (x, y, z, 1)$ in a view and $x' = (x', y', z', 1)$ in another view. The homography is a matrix 4×4 and relates the pixel coordinates in the two views if $x' = Hx$.

To find the homography H matrix, it is necessary to solve (21),

$$PH = 0 \tag{21}$$

where P is a 16×16 matrix represented in (22) and composed by two sets of points, each point set representing a view.

$$\begin{bmatrix}
-x_1 & -y_1 & -z_1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_1x'_1 & y_1x'_1 & z_1x'_1 & x'_1 \\
0 & 0 & 0 & 0 & -x_1 & -y_1 & -z_1 & -1 & 0 & 0 & 0 & 0 & x_1y'_1 & y_1y'_1 & z_1y'_1 & y'_1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -x_1 & -y_1 & -z_1 & -1 & x_1z'_1 & y_1z'_1 & z_1z'_1 & z'_1 \\
-x_2 & -y_2 & -z_2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_2x'_2 & y_2x'_2 & z_2x'_2 & x'_2 \\
0 & 0 & 0 & 0 & -x_2 & -y_2 & -z_2 & -1 & 0 & 0 & 0 & 0 & x_2y'_2 & y_2y'_2 & z_2y'_2 & y'_2 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -x_2 & -y_2 & -z_2 & -1 & x_2z'_2 & y_2z'_2 & z_2z'_2 & z'_2 \\
-x_3 & -y_3 & -z_3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_3x'_3 & y_3x'_3 & z_3x'_3 & x'_3 \\
0 & 0 & 0 & 0 & -x_3 & -y_3 & -z_3 & -1 & 0 & 0 & 0 & 0 & x_3y'_3 & y_3y'_3 & z_3y'_3 & y'_3 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -x_3 & -y_3 & -z_3 & -1 & x_3z'_3 & y_3z'_3 & z_3z'_3 & z'_3 \\
-x_4 & -y_4 & -z_4 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_4x'_4 & y_4x'_4 & z_4x'_4 & x'_4 \\
0 & 0 & 0 & 0 & -x_4 & -y_4 & -z_4 & -1 & 0 & 0 & 0 & 0 & x_4y'_4 & y_4y'_4 & z_4y'_4 & y'_4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -x_4 & -y_4 & -z_4 & -1 & x_4z'_4 & y_4z'_4 & z_4z'_4 & z'_4 \\
-x_5 & -y_5 & -z_5 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & x_5x'_5 & y_5x'_5 & z_5x'_5 & x'_5 \\
0 & 0 & 0 & 0 & -x_5 & -y_5 & -z_5 & -1 & 0 & 0 & 0 & 0 & x_5y'_5 & y_5y'_5 & z_5y'_5 & y'_5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -x_5 & -y_5 & -z_5 & -1 & x_5z'_5 & y_5z'_5 & z_5z'_5 & z'_5 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \\ h_9 \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{13} \\ h_{14} \\ h_{15} \\ h_{16} \end{bmatrix}
=
\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{22}$$

The RANSAC-Homography keypoint correspondences filter algorithm is presented in Algorithm 5.

This algorithm receives as a parameter the set of correspondences to be filtered C, the number of iterations k, and the tolerance of the maximum Euclidean distance expressed by t. A set of best correspondences BC is then defined. Afterward, in each iteration, a random set composed of five correspondences rc is obtained. From this set, the matrix P is constructed.

To solve (21) and compute the homography matrix H, the matrix P is decomposed using singular value decomposition (SVD) algorithm $S, U, V = svd(P)$ (VAN LOAN, 1976), where the last column of V matrix is the matrix H.

After for each correspondence of the input dataset, the corresponding points p_1 and p_2 are obtained. The homography is then applied to bring p_2 to the view of p_1 using $p'_2 = H * p_2$. The Euclidean distance d is then calculated between the points p_1 and p'_2. If d is less than the tolerated distance t then c is added to the set of acceptable correspondences AC. At the end of each iteration, the amount of acceptable correspondences AC is verified to be greater than the amount of best correspondences BC, if true $BC = AC$. At the end of execution only the set with the best correspondences BC is returned.

8.8 Computing the Map Transformation Matrix (MTM) Parameters

After obtaining the best set of correspondence points BC among two maps, it becomes possible to calculate the Map Transformation Matrix (MTM), used to merge the maps

Algorithm 5 RANSAC-Homography keypoint correspondences filter algorithm

procedure RANSACHOMOGRAPHYFILTER(C, k, t)
    $BC \leftarrow []$
    $i \leftarrow 0$
    while $i < k$ **do**
        $rc \leftarrow getRandomSubset(C)$
        $H \leftarrow computeHomography(rc)$
        $AC \leftarrow []$
        for c each in C **do**
            $p_1, p_2 \leftarrow c$
            $p_2' \leftarrow H * p_2$
            $d \leftarrow EuclideanDistance(p_1, p_2')$
            if $d \leq t$ **then**
                Add c in AC
            end if
        end for
        if $size(AC) > size(BC)$ **then**
            $BC \leftarrow AC$
        end if
        $i \leftarrow i + 1$
    end while
    return BC
end procedure

into a single global map. The calculations proposed here to obtain the MTM are based on (KJER; WILM, 2010).

The set of best correspondences BC is composed of two ordered point clouds represented by (23) and (24).

$$A = \begin{bmatrix} p_1^a & p_2^a & ... & p_n^a \end{bmatrix} \tag{23}$$

$$B = \begin{bmatrix} p_1^b & p_2^b & ... & p_n^b \end{bmatrix} \tag{24}$$

where p_1^a is a corresponding point of p_1^b, p_2^a is a corresponding point of p_2^b, successively.

The MTM is the composition of parameters related to translation, rotation, and scale. For the calculation of scale parameters, the technique of building a pyramid of scales to obtain these parameters is commonly used. This thesis takes into account that all maps are on the same scale, so this parameter is omitted, accelerating the computation process and allowing the use of the software in embedded hardware.

To obtain the other parameters, first, it is necessary to calculate the center of gravity (CG) between each of the point clouds. The center of gravity is equal to the average of the coordinates of all points in each point cloud and is calculated based on (25) and (26).

$$CG^a = \frac{1}{n} \cdot \sum_{i=1}^{n} p_i^a \tag{25}$$

$$CG^b = \frac{1}{n} \cdot \sum_{i=1}^{n} p_i^b \tag{26}$$

After obtaining the center of gravity of each point cloud, they are taken to the origin by subtracting their corresponding center of gravity from their coordinates, using (27) and (28).

$$A_o = A - CG^a \tag{27}$$

$$B_o = B - CG^b \tag{28}$$

From the point clouds at its center of gravity, the matrix P is obtained using (29).

$$P = A_o \cdot {B_o}^T = \sum_{i=1}^{n} p_i^a \cdot {p_i^b}^T = \sum_{i=1}^{n} \begin{bmatrix} x_i^a & y_i^a & z_i^a \end{bmatrix} \cdot \begin{bmatrix} x_i^b \\ y_i^b \\ z_i^b \end{bmatrix} \tag{29}$$

Then, the matrix P is decomposed using singular value decomposition (SVD) algorithm (VAN LOAN, 1976), according to (30).

$$S, U, V = svd(P) \tag{30}$$

The rotation matrix R is obtained by (31).

$$R = V \cdot U^T \tag{31}$$

There is a specific case of reflection that must be dealt with. When the determinant of (R) is less than zero, the reflection must be reversed by multiplying the last column of V by -1.0 and then recalculating R using (31).

The translation vector can be obtained using (32).

$$t = -R \cdot CG^a + CG^b \tag{32}$$

In a way, matrix calculation is relatively more computationally costly than performing individual calculations in line. This way the translation and rotation parameters are decomposed using (33).

$$\begin{aligned} t_x &= -t_{00} \\ t_y &= -t_{10} \\ t_z &= -t_{20} \\ \alpha &= atan2(R_{12}, R_{22}) \\ c_2 &= \sqrt{R_{00}^2 + R_{01}^2} \\ \beta &= atan2(-R_{02}, c_2) \\ s_1 &= \sin \alpha \\ c_1 &= \cos \alpha \\ \gamma &= atan2((s_1 \cdot R_{20}) - (c_1 \cdot R_{10}), (c_1 \cdot R_{11}) - (s_1 \cdot R_{21})) \end{aligned} \tag{33}$$

where t_x, t_y and t_z are the translation parameters and α, β and γ are the Euler angles (Roll, Pitch and Yaw, respectively), used for rotation.

Due to the downsampling process applied to the original maps during the preprocessing process presented in Section 8.1 the translation parameters need to be adjusted to the original proportions using (34).

$$\begin{aligned} S_x &= \boldsymbol{m}_{t\,x}^0 / (2.0 \cdot DS_r) \\ S_y &= \boldsymbol{m}_{t\,y}^0 / (2.0 \cdot DS_r) \\ S_z &= \boldsymbol{m}_{t\,z}^0 / (2.0 \cdot DS_r) \\ t_x &= (t_x + (S_x - int(S_x))) \cdot 2.0 \cdot DS_r \\ t_y &= (t_y + (S_y - int(S_y))) \cdot 2.0 \cdot DS_r \\ t_z &= (t_z + (S_z - int(S_z))) \cdot 2.0 \cdot DS_r \end{aligned} \tag{34}$$

where $\boldsymbol{m}_{t\,x}^0$, $\boldsymbol{m}_{t\,y}^0$ and $\boldsymbol{m}_{t\,z}^0$ are the original fixed local map dimensions size, DS_r is the downsample radius parameter and $int(S_x)$, $int(S_y)$ and $int(S_z)$ are the integer part of S_x, S_y and S_z. Finally, these parameters must be applied to the map corresponding to the point cloud B to enable the merging of the maps.

The calculated parameters are the best map merge parameters for the current scenario and the application settings. Perhaps, by the wrong parameters definition or low level of overlap, the calculated parameters are not adequate. To the best knowledge, there are no techniques available to adequately evaluate an MTM without any ground truth or global positioning system information. So an accurate evaluation method of MTMs still needs to be explored to improve the accuracy of the application.

8.9 Merging 3D Occupancy Grid Maps

In the process of merging occupancy grid maps by pairs, the map of a vehicle that is performing the embedded merge must be fixed as the center of the coordinate space and then the transformation parameters are discovered for the other maps. The merging process begins by applying the already computed MTM to all points of maps. After transformation, the transformed maps must be superimposed on the map identified as a virtual world center.

This process consists of merging the maps by copying all the relevant information to a single map. As cells represent occupancy states of different views of the environment, maps cannot be simply overlaid assuming the cell state of the complementary map because important information about obstacles that are part of the scene may be lost.

Table 3 demonstrates the rules used in the process of merging the occupancy grid map cells.

Table 3 – Rules for merging occupancy grid map cells, adapted from (MA *et al.*, 2016).

$\boldsymbol{W}_t$ \ $\boldsymbol{m}_t^i$	**Unknown**	**Free**	**Occupied**
Unknown	Unknown	Free	Occupied
Free	Free	Free	Occupied
Occupied	Occupied	Occupied	Occupied

The first column shows the state corresponding to the map cell fixed as world center $\boldsymbol{W}_t$ at the instant t. The first line shows the state of the cell corresponding to the transformed complementary map $\boldsymbol{m}_t^i$ of the vehicle i at the time instant t. The rest of the table shows the status that should be assigned to the new cells for each of the possible combinations in the process of merging the maps.

After the merge process, the global map $\boldsymbol{W}_t$ is incremented with the global experiences and it is available for the vehicle's navigation system and other systems.

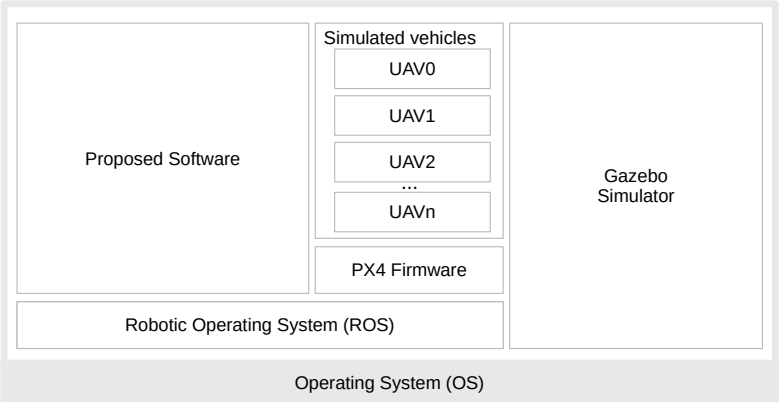
9 SOLUTION IMPLEMENTATION AND EVALUATION DESIGN

This chapter presents the implementation details (Section 9.1) and the design of the experiments (Section 9.2) performed to evaluate the presented solutions.

9.1 Implementation Details

The architecture of the proposed system is schematized and presented in Figure 37.

Figure 37 – Proposed system high-level building blocks architecture.



Source: author.

A notebook with Intel Core i7-4510U 2.00 GHz x 4 processor, 8 GB RAM, 240 GB SSD, and 2 GB dedicated GeForce 710M graphics card was used. The operating system used as the base was Ubuntu 20.04.3 LTS 64-bit.

As a basis for the proposed implementations, the Robotic Operating System (ROS) was used (QUIGLEY; GERKEY; SMART, 2015). For the simulation of three-dimensional virtual environments and their dynamics, the Gazebo Simulator was used (GAZEBO TUTORIALS, 2014). A PX4 Firmware[1] fork was used to enable simulation of multi-UAVs

[1] https://github.com/maikbasso/Firmware

(PX4 OPEN SOURCE AUTOPILOT, 2018). Each of the simulated vehicles had attached an RGB-D camera from which point clouds were collected to build the 3D occupancy grid maps.

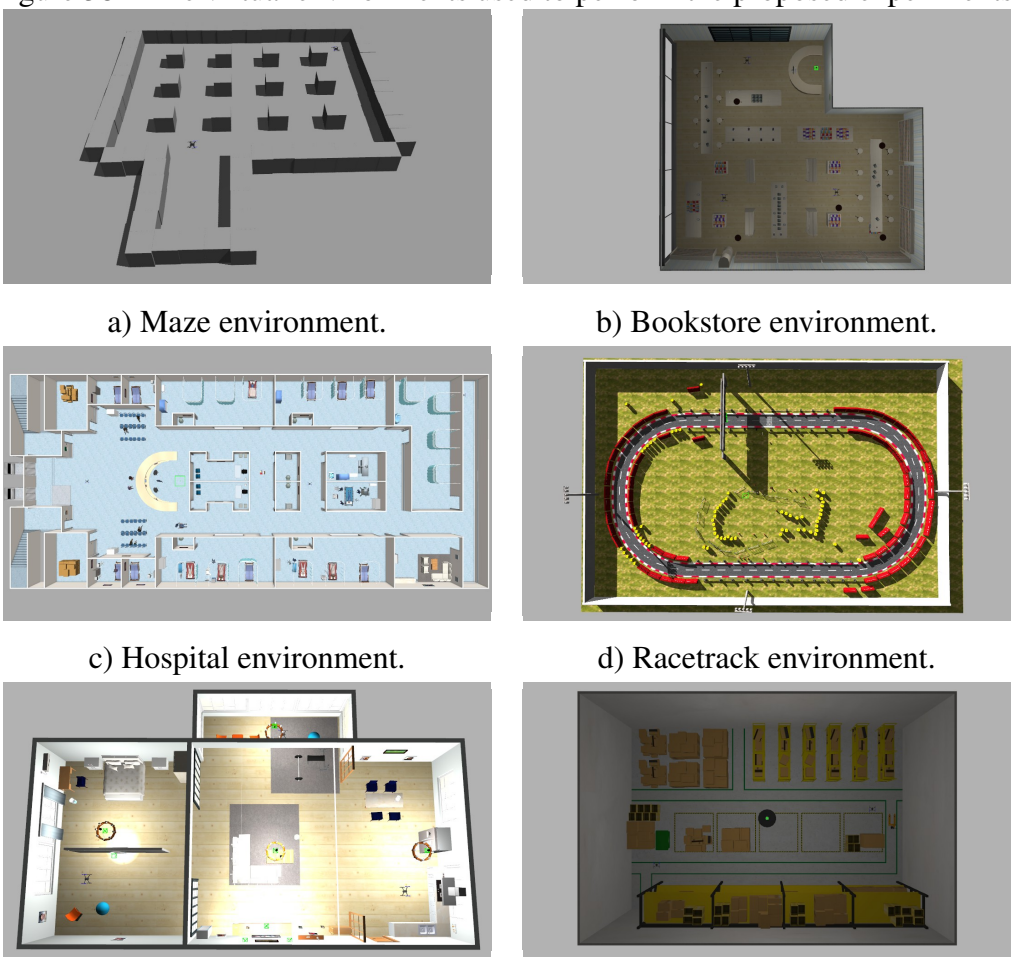
All proposed software was implemented in C++11 language using only the Eigen library for complex matrix calculations. Due to the modularity provided by ROS and the PX4 Firmware architecture, the proposed software can be used both in simulation and in real UAV embedded hardware.

The source codes for the implementations, simulations, experiments, text, and presentations are available at <https://github.com/maikbasso/phd-thesis>.

9.2 Experiments Setup

To evaluate the proposed applications in this thesis, six different scenarios were used to carry out simulations. Figure 38 shows a top view of each of the used three-dimensional simulation environments.

Figure 38 – The virtual environments used to perform the proposed experiments.



a) Maze environment. b) Bookstore environment.

c) Hospital environment. d) Racetrack environment.

e) Small house environment. f) Small warehouse environment.

Source: author.

The first environment called Maze (Figure 38a), was developed to be used in the demonstrations of the visual results proposed in this thesis, this environment has only 1 m of altitude so it allows the visualization of the data in two dimensions is suitable for inclusion and visualization in this thesis. Still, this environment represents the worst test case due to a large number of similar regions and a low level of detail in the environment structure.

The other environments used (Figure 38b to 38f), were taken from the repository of the AWS RoboMaker[2] project on GitHub. These environments represent indoor and outdoor regions, with considerable extension, level of detail, and acceptable textures for carrying out simulations.

In these environments, the UAVs were dispersed in different regions to explore partial regions but with overlapping to allow the merge of maps. In the environments represented by the Figures 38b, 38c and 38e, three distinct UAVs were used. In the environments represented in Figures 38a, 38d and 38f, two UAVs were used. In total, fifteen 3D occupancy grid maps were acquired, which were used to carry out the experiments in this thesis. The specifications of each of these acquired maps are presented in Table 4.

Table 4 – Maps Information.

ID	Environment	UAV	Resolution (m)	Axes size (cells)			Cells number	Data Size (MB)	Exploration time (seconds)
				x	y	z			
0	bookstore	0	0.1	250	250	100	6250000	5.96	149
1	bookstore	1	0.1	190	190	100	3610000	3.44	127
2	bookstore	2	0.1	190	190	100	3610000	3.44	148
3	hospital	0	0.1	730	730	100	53290000	50.82	1091
4	hospital	1	0.1	500	500	100	25000000	23.84	691
5	hospital	2	0.1	620	620	120	46128000	43.99	547
6	maze	0	0.1	230	230	100	5290000	5.04	292
7	maze	1	0.1	320	320	100	10240000	9.77	412
8	racetrack	0	0.1	670	670	100	44890000	42.81	308
9	racetrack	1	0.1	1050	1050	100	110250000	105.14	508
10	small house	0	0.1	250	250	100	6250000	5.96	336
11	small house	1	0.1	310	310	100	9610000	9.16	384
12	small house	2	0.1	230	230	100	5290000	5.04	288
13	small warehouse	0	0.1	340	340	100	11560000	11.02	267
14	small warehouse	1	0.1	270	270	100	7290000	6.95	261

The information presented in Table 4 is the size of the maps in the number of cells and megabytes, the environment name, the number of the UAV that acquired the map, the exploration time spent to build each map and the resolution of the maps are described. The resolution of the maps in this thesis was set to 0.1 m, this value was defined by the wide use in the related works evaluated.

After acquiring the maps, they were stored in files so that they could later be used to carry out the tests proposed in this thesis. Table 5 presents the identifying number of all

[2]https://github.com/aws-robotics

possible pairs of maps used to execute the proposed tests.

Table 5 – List of map pairs.

Pair ID	Map 0 ID	Map 1 ID
0	0	1
1	0	2
2	1	2
3	3	4
4	3	5
5	4	5

Pair ID	Map 0 ID	Map 1 ID
6	6	7
7	8	9
8	10	11
9	10	12
10	11	12
11	13	14

Then, the acquired maps were submitted to sharing and merging tests to obtain the data used to generate the results. All proposed tests were written in C++ language and used the *std::chrono* library to acquire the execution times for each part of the proposed application. The data obtained for each of the proposed tests were stored in CSV files for further processing.

After the CSV files data were processed using Python language in a *Jupyter Notebook* environment, thus generating a graphic presentation of the results that are discussed in Chapter 10.

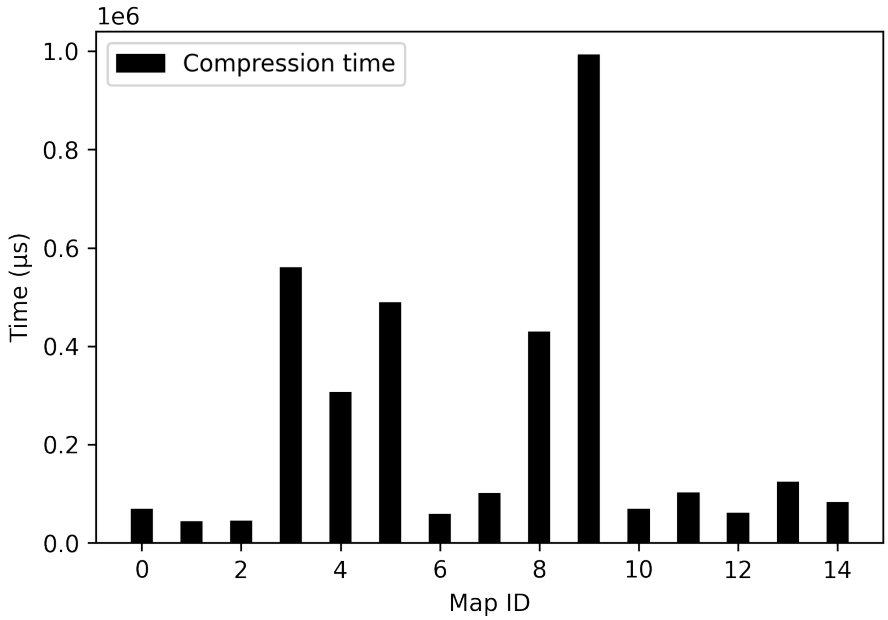
10 RESULTS AND DISCUSSIONS

This chapter presents in detail the results and discussions of the experiments to validate the main components of the proposed system. Section 10.1 present the results to evaluate the map compression component. The map transmission protocol is evaluated in Section 10.2. The complete map merging system is evaluated in Section 10.3. Finally, the application scalability is evaluated in Section 10.4.

10.1 Map Compression Evaluation

To evaluate the compression and decompression algorithms, each map in the test set was submitted to 100 compression and decompression iterations. The average time taken to compress each map is shown in Figure 39.

Figure 39 – Mean map compression time.

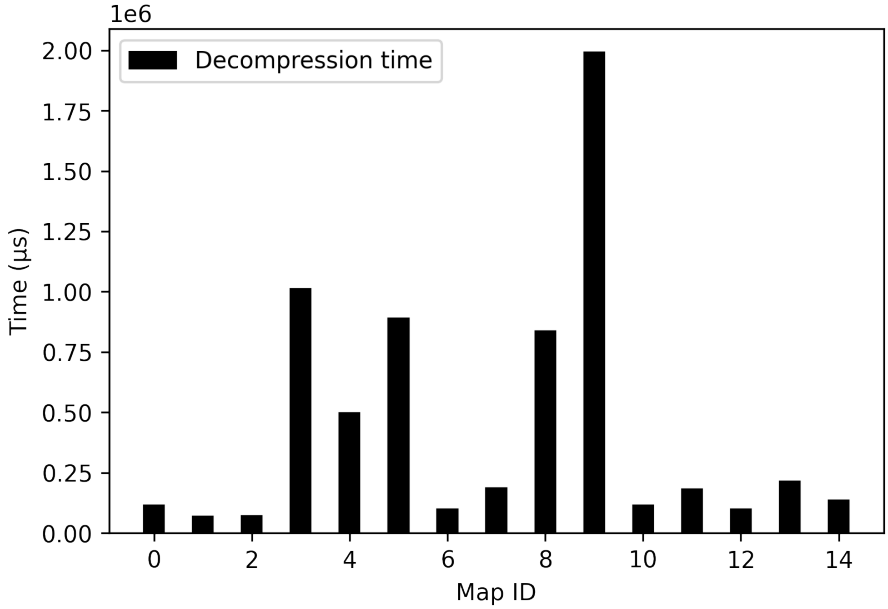


Source: author.

The average time taken to decompress each map is shown in Figure 40.

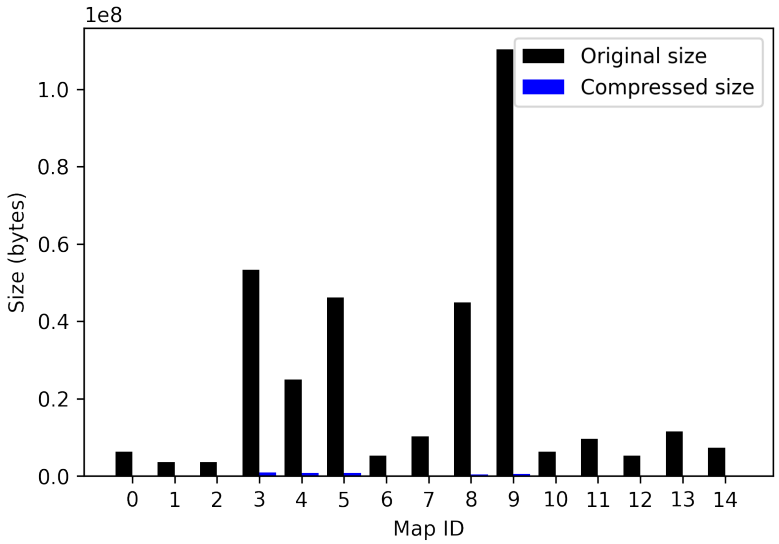
The size comparison of maps before and after the compression process is shown in Figure 41.

Figure 40 – Mean map decompression time.

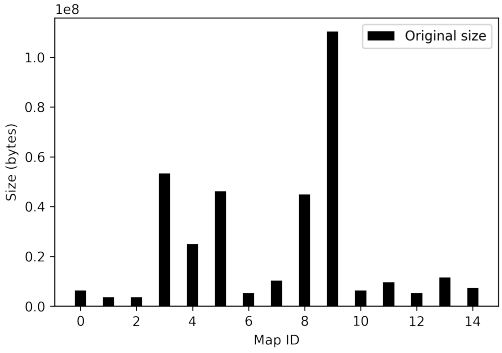


Source: author.

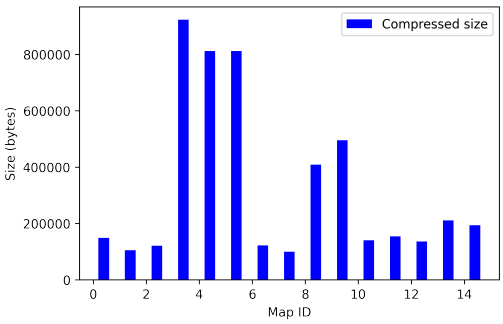
Figure 41 – Comparison of the maps' size before and after compression.



a) Comparison of original and compressed maps size.



b) Original size.



c) Compressed size.

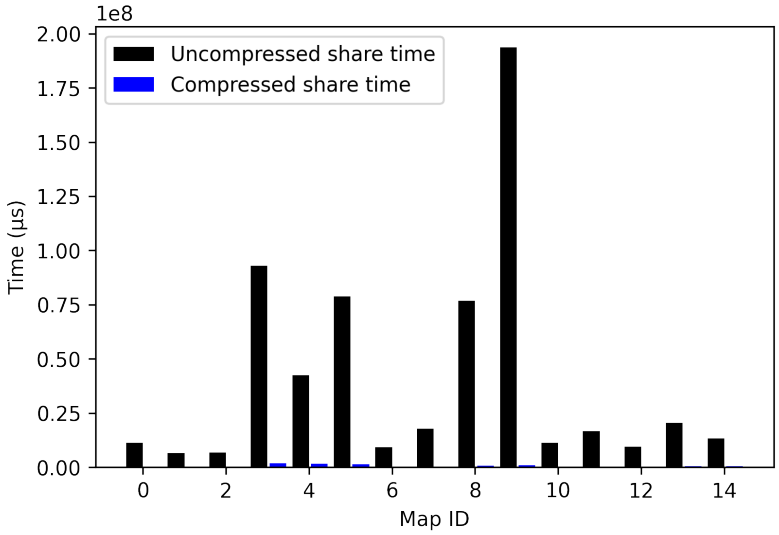
Source: author.

Observing the obtained results, it is possible to see that the maximum time spent on the compression is around 1 s and the maximum time spent on the decompression was around 2 s. Taking into account the significant reduction in the size of the maps presented, the time spent on the compression and decompression processes is justified and directly benefits the map sharing algorithm.

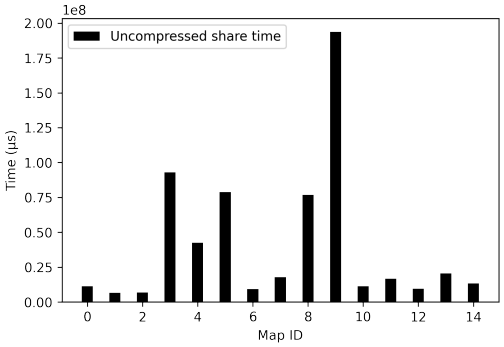
10.2 Map Transmission Protocol Evaluation

To evaluate the map transmission protocol, the test set was subjected to 100 sharing iterations for each of the maps. In this experiment, the transmission time of maps with and without compression was evaluated. The average time taken to share each of the maps is shown in Figure 42.

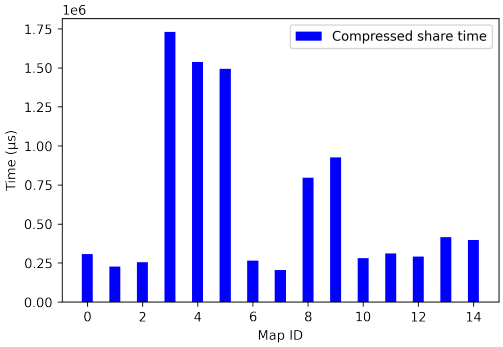
Figure 42 – Mean time to share maps with and without compression.



a) Comparison of uncompressed and compressed maps share time.



b) Sharing time of uncompressed maps.



c) Sharing time of compressed maps.

Source: author.

The maximum time spent sharing the largest map was 200 s for uncompressed maps and 1.75 s for compressed maps. Taking into account the times presented in Section 10.1 for compression and decompression, it can be concluded that the maximum time

to share compressed maps with an area of up to 100 m^2 is around 3.75 s, proving the good performance of the solutions and reducing significantly the time to share the maps presented in this thesis.

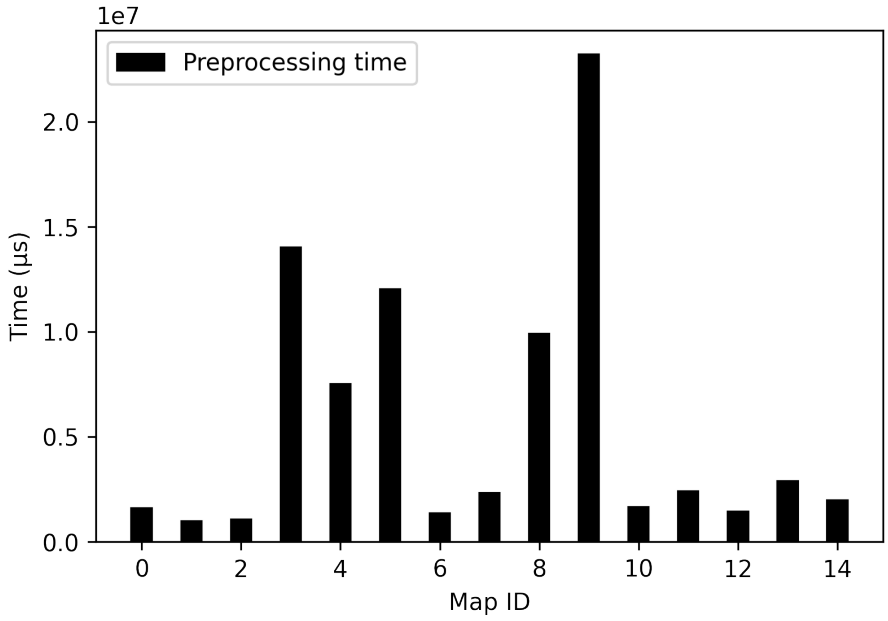
10.3 Map Merging Evaluation

This section presents in detail the results and discussions of the experiments to validate the map merging system component. Sections 10.3.1 to 10.3.6 present the results from specific components of the proposed system. The complete evaluation of the entire system for the test maps set is presented in Section 10.3.7. The visual evaluation of a map merge performed by the proposed system using maps acquired in the maze environment is presented in Section 10.3.8. Finally, the global maps obtained after fusion for all test environments used to perform simulations are presented in 10.3.9.

10.3.1 Map Preprocessing Evaluation

The average preprocessing time obtained for each of the maps is shown in Figure 43. In this experiment, 100 iterations were performed. These results were obtained considering one iteration for dilation ($D_i = 1$), erosion ($E_i = 1$), downsampling and Gaussian filter ($G_i = 1$). Also, is considered the radius measure equal to 1 for dilation ($D_r = 1$), erosion ($E_r = 1$) and downsampling ($DS_r = 1$).

Figure 43 – Mean time to preprocessing maps.



Source: author.

The results show that the preprocessing time grows considerably when the map size increases. As already described in this thesis, this step is essential to remove noise and speed up the subsequent steps of the algorithm, so one of the alternatives to reduce these

times is the introduction of more UAVs so that smaller local maps can be produced since the map that presented the longest processing time was the number 9 and it has more than 100 m in length. Another alternative would be to disable the erosion and dilation filter, but for this to be possible, the map construction algorithm needs to be more robust and produce more accurate maps with fewer continuity failures.

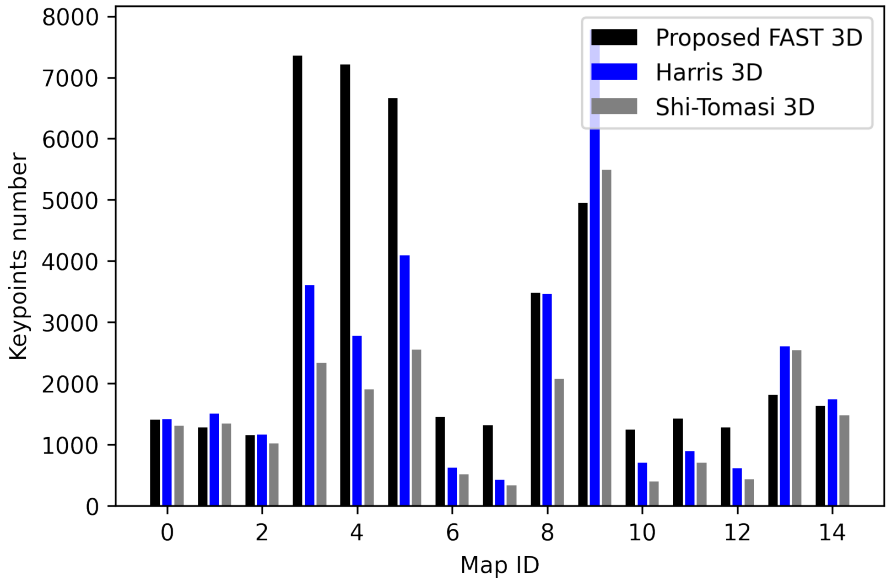
10.3.2 Keypoint Detector Evaluation

The experiments with the keypoint detector were carried out comparing the proposed detector, FAST 3D, with the Harris 3D and Shi-Tomasi 3D algorithms. The Harris 3D and Shi-Tomasi 3D algorithms had their 2D versions extended by a project for an undergraduate thesis in the laboratory where this thesis is being developed. This work is part of the research group that is related to this thesis. The work aimed to extend the 2D to 3D algorithms and apply them to 3D occupancy grid maps. The FAST 3D algorithm also emerged from this research, but this thesis received several improvements.

In this experiment the maps were subjected to keypoint detection divided into 100 iterations. The parameters used for the proposed algorithm were continuity $n = 9$, radius $r = 3$, threshold $th = 80$ and neighborhood radius $nr = 0$. The parameters used for Harris were $k = 0.05$, window size $ws = 5$ and threshold $th = 9000.0$. The parameters used for Shi-Tomasi were window size $ws = 5$ and threshold $th = 15.0$.

Figure 44 shows the average number of keypoints detected by each of the algorithms for each of the maps.

Figure 44 – Comparison between keypoint detectors in keypoints output number.



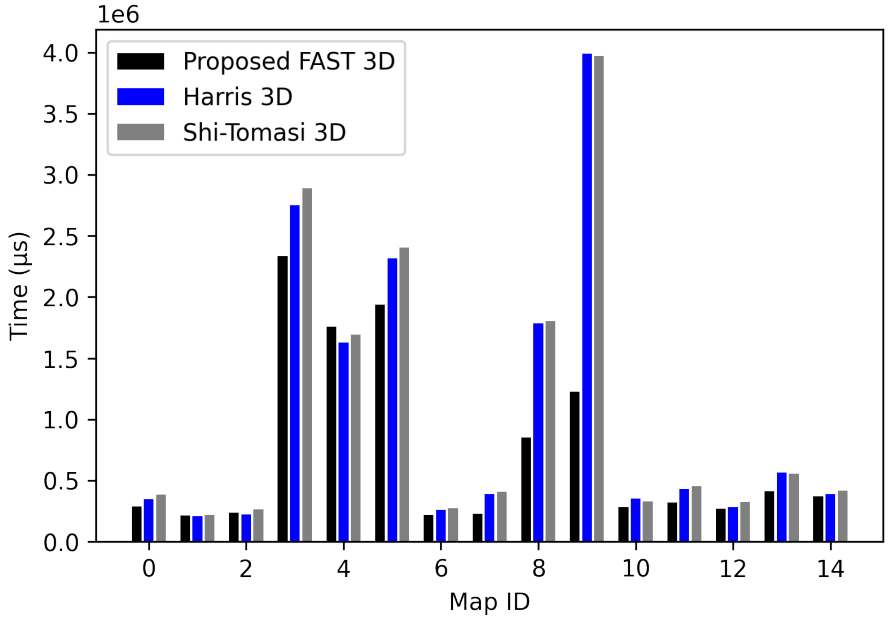
Source: author.

As the tests on the different maps were carried out without the variation of the parameters, it is clear that the proposed algorithm retrieves several keypoints similar to the other algorithms, but it is clear that it is more sensitive to internal environments, recovering a

greater number of keypoints under these conditions.

The average time taken to detect keypoints is shown in Figure 45.

Figure 45 – Comparison between keypoint detectors in mean time to detect.

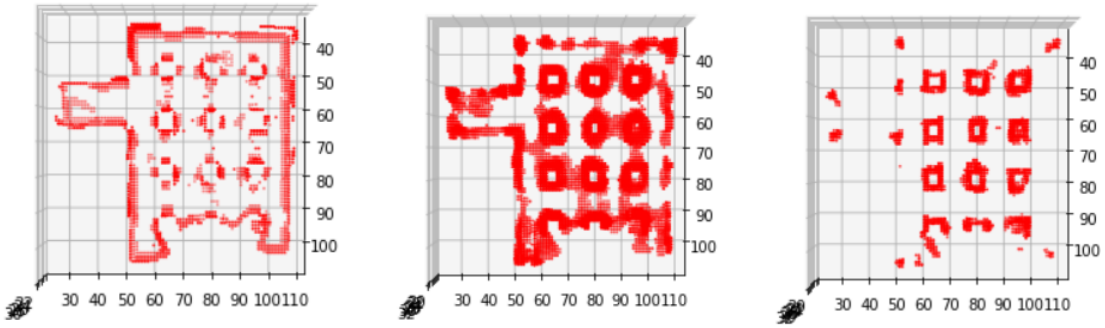


Source: author.

These results demonstrate that the proposed algorithm has a performance equal to or greater than the other solutions, even recovering a greater number of keypoints than the other algorithms.

Figure 46 presents the keypoints detected by the proposed algorithm in comparison with Harris 3D and Shi-Tomasi 3D.

Figure 46 – Visual comparison between keypoint detectors applied to the map ID = 6.



a) Proposed FAST 3D. b) Harris 3D. c) Shi-Tomasi 3D.

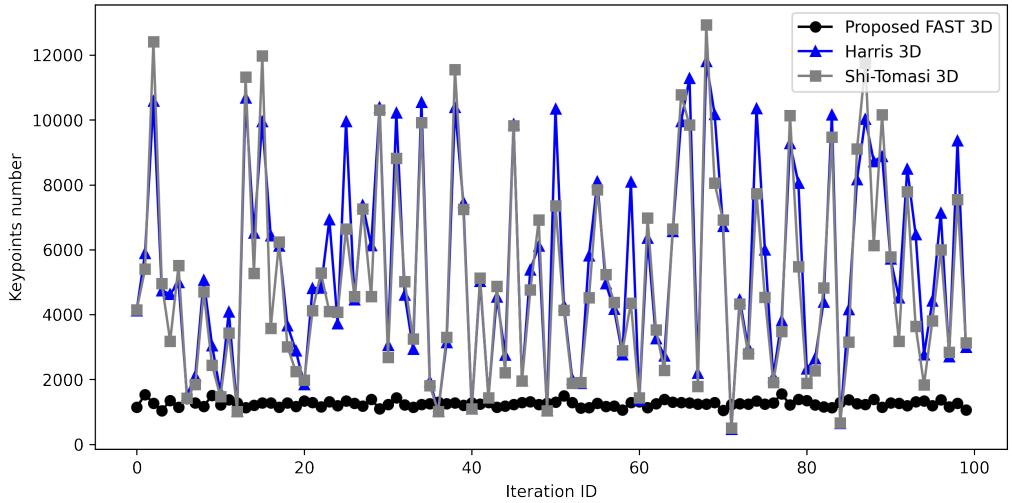
Source: author.

It is possible to observe in these visual results that the keypoints retrieved by the proposed algorithm represent all corners and edges of the environment, disregarding flat regions with low interest. The problem with Harris 3D and Shi-Tomasi 3D is that there is a tradeoff between recovering all edges and corners and reducing the number of keypoints detected in the 3D occupancy grid maps. In this example, if the parameters are refined to

retrieve a reduced number of keypoints only the most obvious corners are retrieved. Also, the concentration of keypoints in corners is very high, while in the proposed algorithm few points describe the corners and edges for each level of the map.

To prove the stability of the proposed detector, the same map of the previous experiment was submitted to 100 random transformations in the six degrees of freedom. For each of the transformations, the number of keypoints was acquired and the results are shown in Figure 47.

Figure 47 – Comparison between keypoint detectors submitted to random map transformations.



Source: author.

It is possible to observe that the proposed method retrieves an almost similar number of keypoints for each of the iterations with a small variation resulting from the error of the algorithm that applies the transformations in the maps. The other evaluated algorithms present abrupt changes in the number of detected keypoints, which can cause a sudden variation in the algorithm performance in the later stages. Thus, the proposed method proves to be more efficient for the detection of keypoints in 3D occupancy grid maps.

10.3.3 Keypoint Filter Evaluation

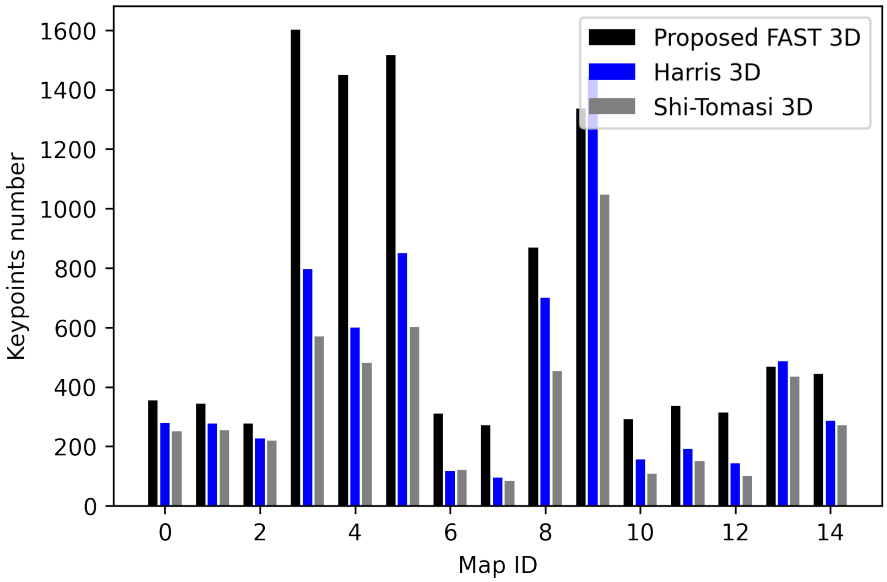
Figure 48 presents the average number of filtered keypoints obtained from the experiment presented in Section 10.3.2. The parameters used in the filter were radius $KF_r = 1$ and the minimum radius for adjacent keypoints $KF_{ar} = 1.0$.

It is possible to observe that by the number of keypoints after the filtering process compared to the total number of keypoints previously obtained that the proposed algorithm can retrieve more significant keypoints or with a higher score compared to the other algorithms, this is evident by observing the results referring to the map ID equal to 9.

The average time taken to perform the filtering process is shown in Figure 49.

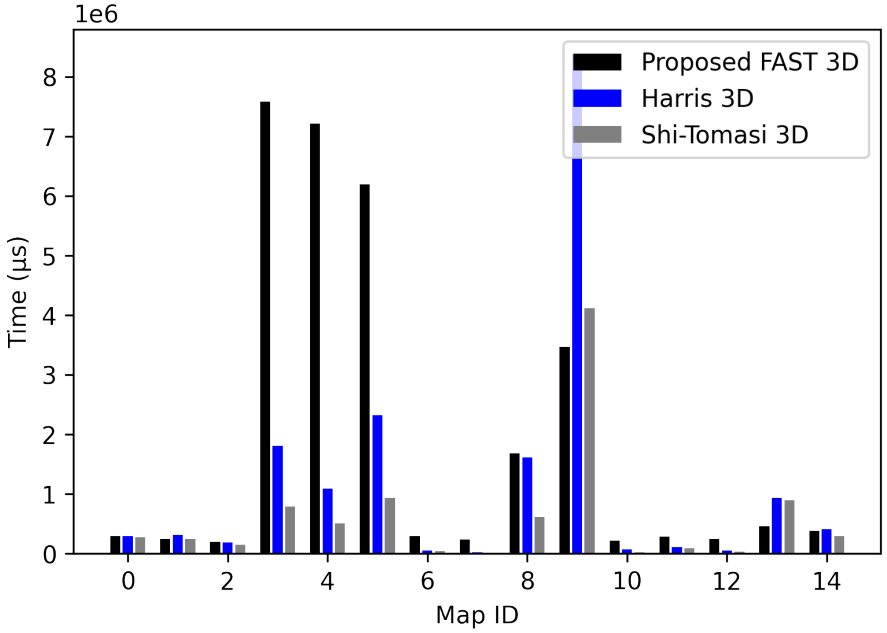
The keypoint filtering time grows exponentially to the number of detected keypoints, so the fine-tuning of the detector parameters for each type of environment makes more

Figure 48 – Mean filtered keypoints.



Source: author.

Figure 49 – Mean time to filter.



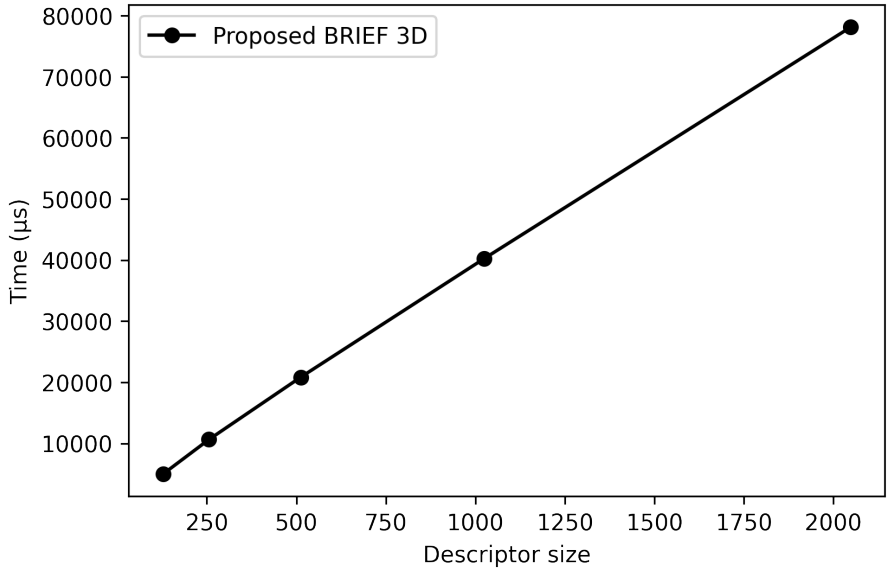
Source: author.

relevant keypoints retrieved and thus significantly reducing the filtering time. Thus, the use of this filtering process is optional in the execution of the algorithm, but its use is recommended to speed up later steps of the algorithm.

10.3.4 Keypoint Descriptor Evaluation

To test the proposed descriptor algorithm, an experiment considering a set of 100 keypoints was performed. The objective of this experiment was to verify the impact on execution time caused by the size of the descriptor selected as a parameter. The descriptor sizes evaluated were 128, 256, 512, 1024 and 2048. The other parameters used in the experiment were window size $ws = 31$, sample seed $ss = 42$ and neighboring radius $nr = 0$. Figure 50 presents the results obtained with the experiment.

Figure 50 – Mean description time by descriptor size and 100 keypoints.



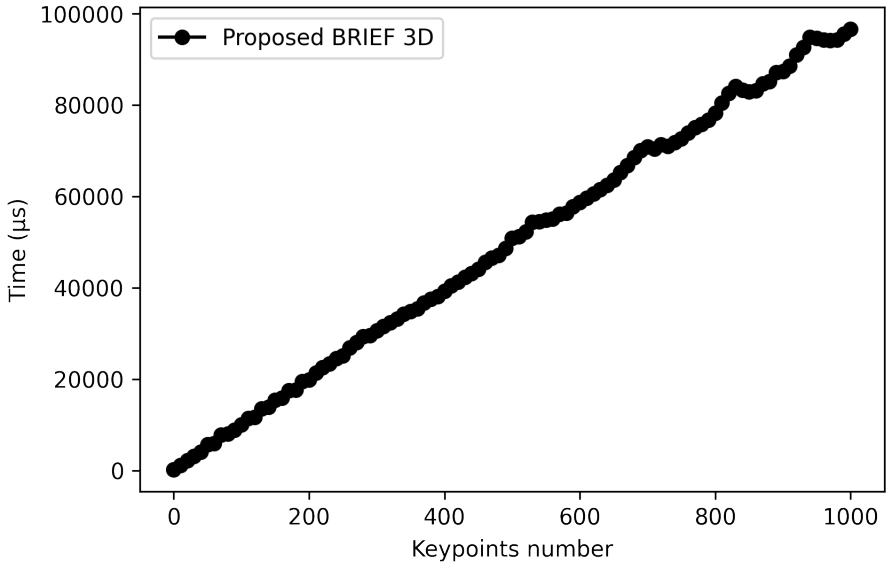
Source: author.

Observing the obtained results, it is possible to report that the time to create a descriptor depends totally on the size of the selected descriptor. In the merge tests applied in this thesis, it was observed that descriptors with size $ds > 128$ present a good result for merging maps, and descriptors with size $ds > 512$ do not present so much difference for the maps used in the tests. The selection of $ds = 256$ was adopted to perform the other map fusion tests.

Figure 51 shows the average time to create the descriptors of a set of keypoints between 0 to 1000 keypoints.

The description time for a set of keypoints increases in proportion to the number of keypoints in the set. Thus, this step can be accelerated by choosing the correct detector parameters and descriptor size. Nevertheless, the proposed descriptor presented a performance close to 0.1 second for the description of 1000 keypoints under the presented circumstances.

Figure 51 – Mean description time by number of keypoints and descriptor size $ds = 256$.

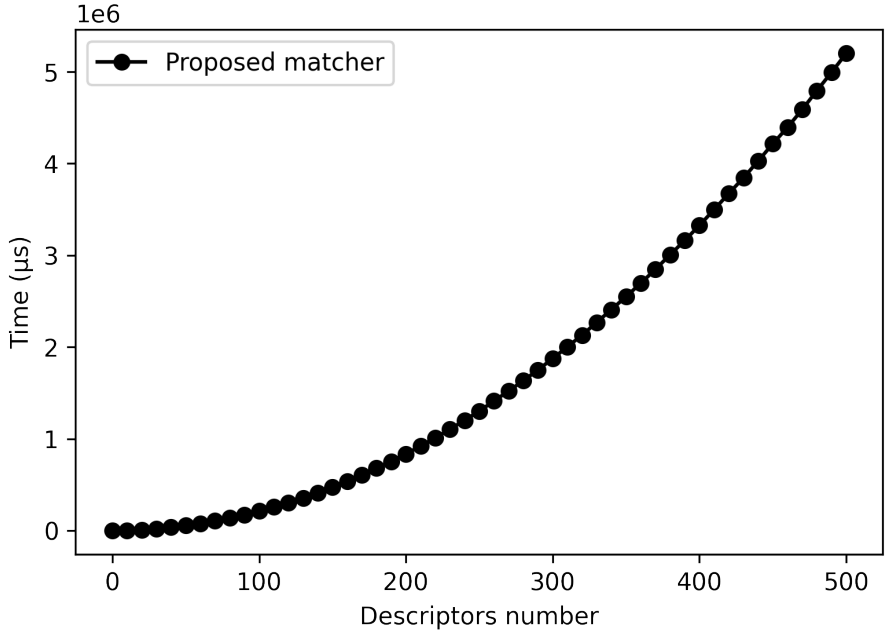


Source: author.

10.3.5 Brute Force Descriptor Matcher Evaluation

Figure 52 presents the average time to identify correspondences between two sets of descriptors of size 0 to 500 descriptors. In this experiment 100 iterations were performed, the descriptor size was set to $ds = 256$ and the maximum distance was set to $CC_d = 20$.

Figure 52 – Mean time to find correspondences by descriptors number.



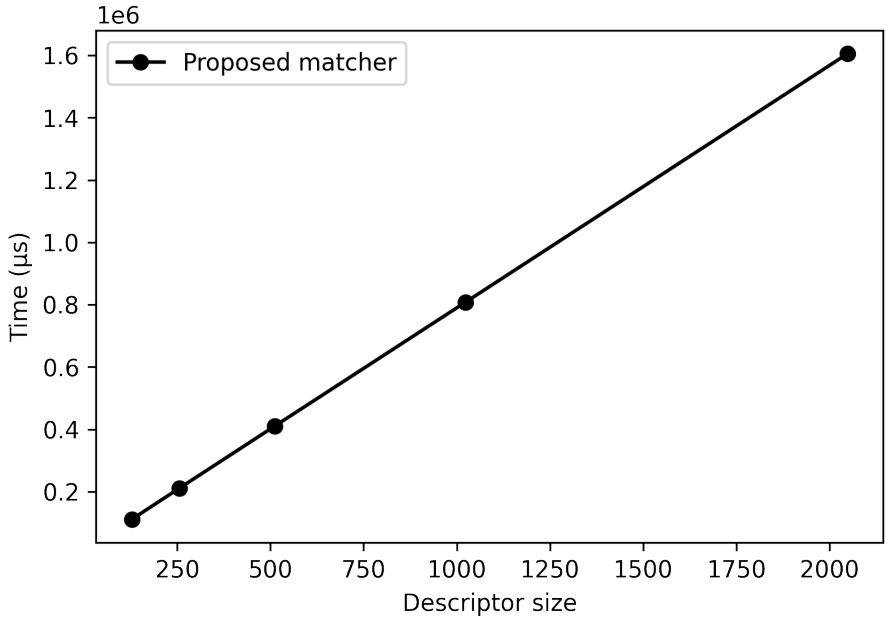
Source: author.

In this test, it is observed that the time of identification of correspondences grows exponentially with the size of the set of descriptors. As mentioned before, obtaining a smaller set of descriptors should speed up this step of the algorithm.

In Figure 53 a set of experiments with 100 repetitions were performed to identify the

impact of descriptor size on the correspondences identification algorithm for a set of 100 descriptors.

Figure 53 – Mean time to find correspondences by descriptors size and 100 descriptors.



Source: author.

As expected, the algorithm presents increasing linear time related to the size of the used descriptor. Then, based on the obtained results, the descriptor size was set at $ds = 256$.

10.3.6 Correspondences Filter Evaluation

To evaluate the proposed correspondences filter, 100 iterations were applied to filter a set of 500 correspondences. In this experiment, a random number of true correspondences ranging from 40 % to 80 % of the correspondences in the input set were inserted in each iteration. The parameters used in the proposed algorithm were number of iterations $k = 1000$ and distance threshold $t = 3.0$.

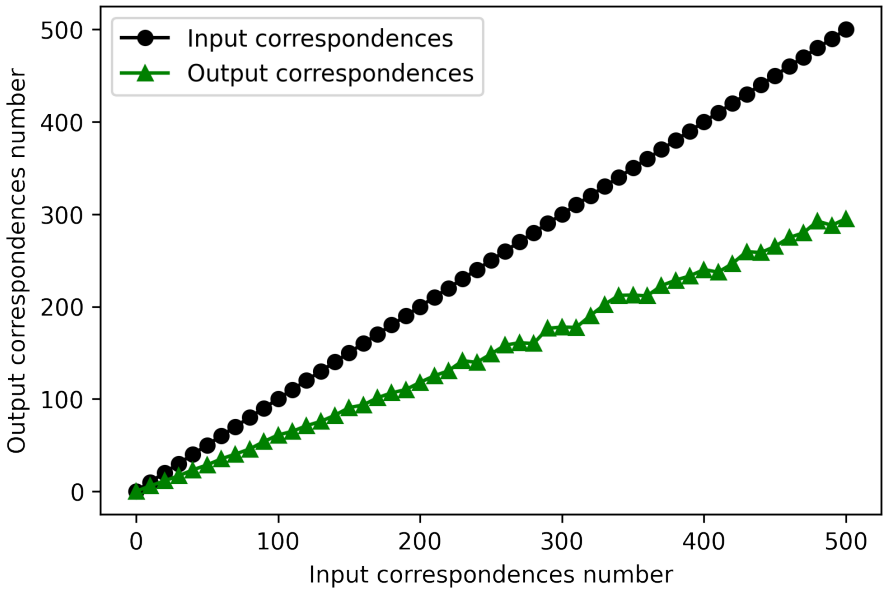
Figure 54 presents the average number of input correspondences to the average number of true correspondences applied as an input in the tests proposed in this experiment.

Figure 55 shows the number of true correspondences input into the algorithm and the number of true correspondences retrieved after filtering.

Observing the obtained result, it can be seen that the algorithm is capable of recovering almost all true correspondences applied as input to the algorithm. Figure 56 presents the average of false correspondences added as an input to this experiment, relative to the number of false correspondences present in the output of the algorithm.

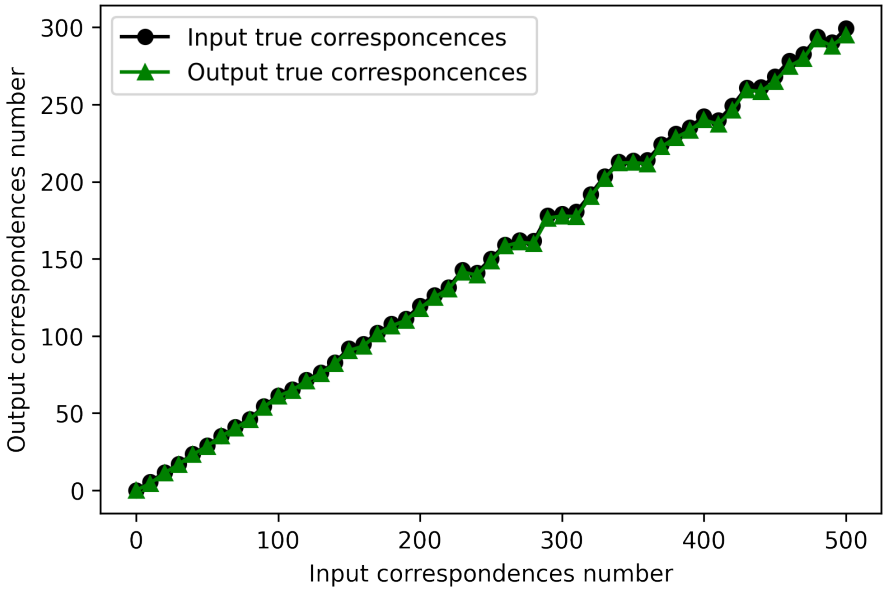
It is possible to notice that the number of retrieved false correspondences is practically null when the number of incoming correspondences is greater than 10. For the proposed filter based on RANSAC to work correctly, it is necessary to have more than 8 correspon-

Figure 54 – Mean input and output correspondences number.



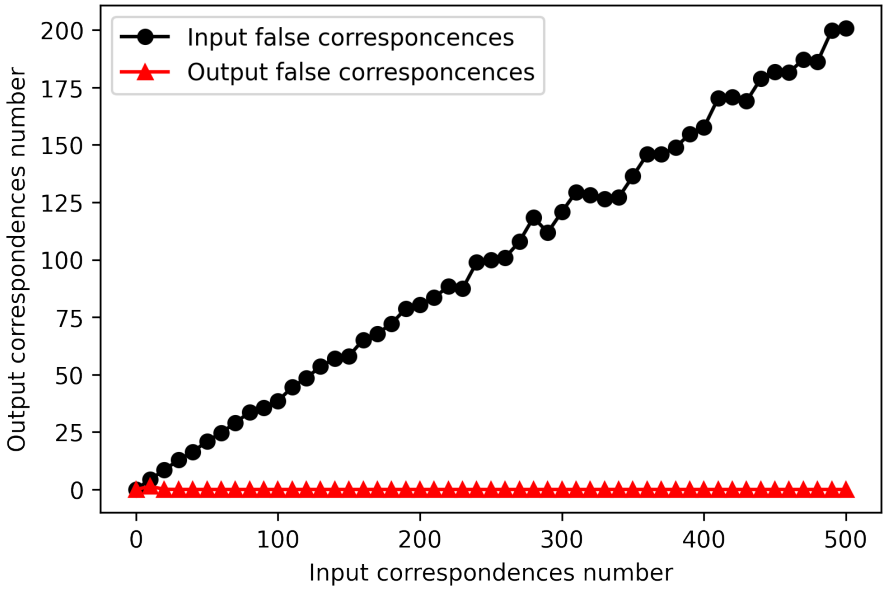
Source: author.

Figure 55 – Mean true input and output correspondences number.



Source: author.

Figure 56 – Mean false input and output correspondences number.

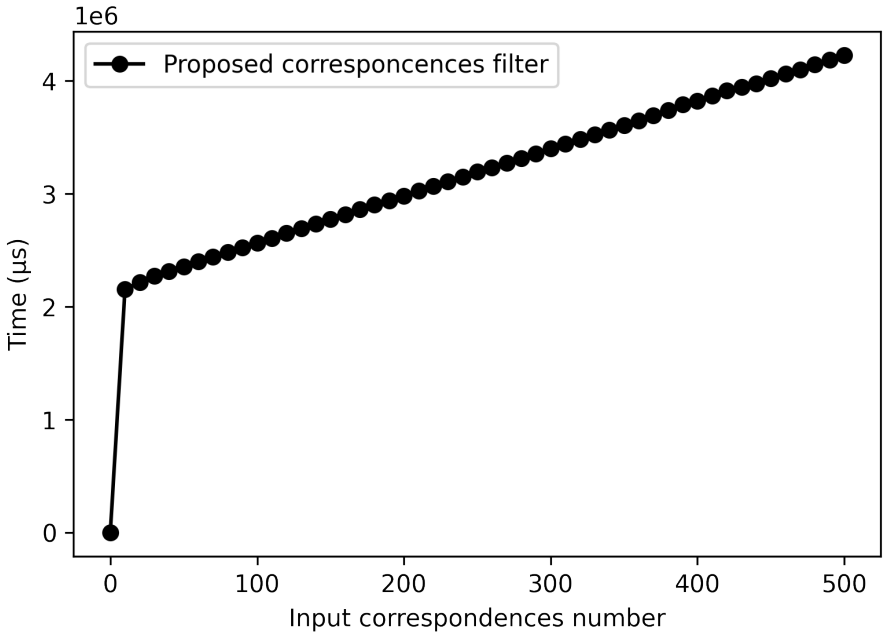


Source: author.

dences and that more than 50 % of these are true, under these conditions the filter can properly retrieve the true correspondences.

Figure 57 shows the average time for filtering the set of correspondences for this experiment.

Figure 57 – Mean time to filter correspondences.



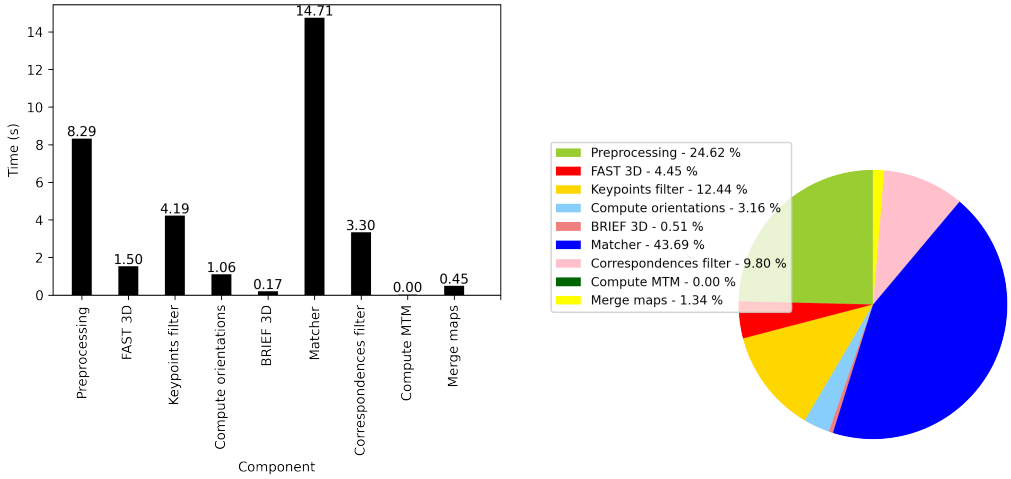
Source: author.

As expected, the filtering time grows linearly with the increase in the number of correspondences. Being close to 4 seconds for a set of 500 correspondences, when the proposed parameter set is used.

10.3.7 Pairwise Map Merging Time Evaluation

To evaluate the complete proposal of a pairwise map merging system, the set of test map pairs was submitted to merging using the parameters defined in the previous experiments. In this experiment, 100 iterations were performed. For each of the iterations, the pairs of maps are randomly ordered and the average time for the execution of each component of the proposed system during the merge can be seen in Figure 58.

Figure 58 – Mean component execution time.



a) Mean execution time in seconds. b) Mean execution time in percentage.
Source: author.

Observing the presented results, it can be seen that the preprocessing component and the matcher component are the most significant in terms of processing time. As mentioned, the preprocessing component is directly impacted by erosion and dilatation filters. The guaranteed inclusion of maps with higher continuity quality and accuracy can be a reason to disable these filters and speed up the process. As for the correspondence calculation algorithm, this thesis does not fully explore all the technologies available in this component, so the use of any solution that does not use brute force can significantly speed up the correspondence computation process.

Still, the time for calculating the orientations includes the time of iterations to build the potential field maps. In navigation systems that use these maps for exploration, this map must be available in advance so this processing time can be significantly reduced.

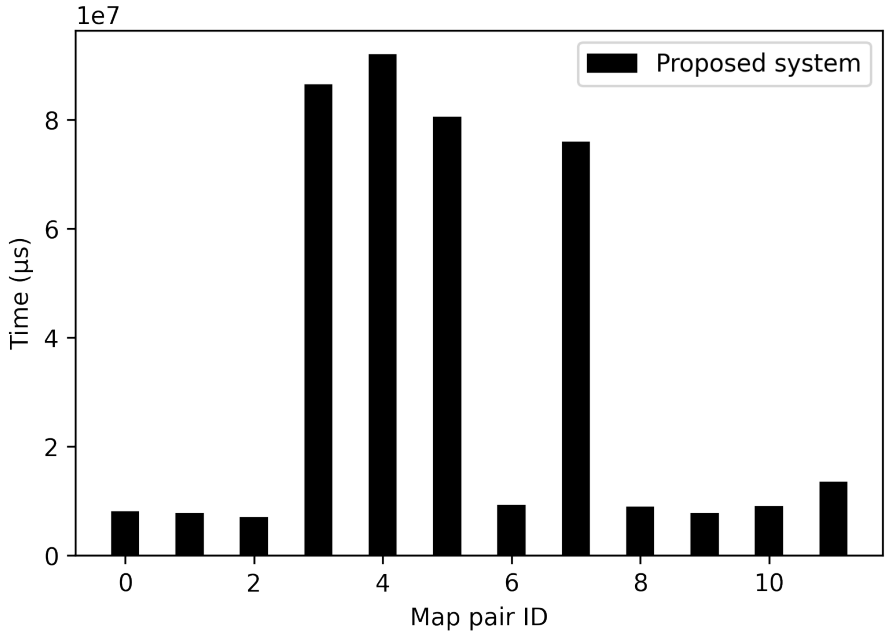
Keypoint and correspondences filters have their performance changed by the amount of input data and also by the parameters used. A refinement of the parameters for each of the maps could significantly reduce these times, but the idea of this experiment is precise to evaluate the behavior of the parameters used for different sets of maps with different characteristics.

The proposed detector and descriptor algorithms, in turn, present a good performance considering the average size of the maps used in the tests. In turn, the algorithm responsible for calculating the MTM parameters has the least representative execution time in the

proposed system. The execution time of the algorithm that merges the maps are related to the transformation time of all points and copying this information to a new global map structure.

Finally, the average merge execution time for each of the map pairs is shown in Figure 59.

Figure 59 – Mean time to merge each map pair.



Source: author.

Observing the execution times for each map pair it is possible to notice that the size of the maps has a significant impact on the system performance. Furthermore, indoor environments generally have more corners in their construction so the number of keypoints increases, and consequently the processing time becomes higher. This can be easily evidenced by comparing the hospital (indoor) with the racetrack (outdoor) results.

Taking into account all the obtained results, it is difficult to address all the problems found since all system components are interdependent. Furthermore, the mapping algorithm can produce maps in different views or ways directly affecting map merging tasks. The way vehicles are driven through the environment, that is, the way the exploration algorithm drives vehicles through the environment, also directly affects the performance of map merging.

Furthermore, the map merging problem is related to the amount of information overlapping between maps. The larger the overlap region, the greater the likelihood that the merger will take place. The problem of merging maps does not have a general solution, but adequate solutions for each situation. Even so, applications involving fusion using image processing techniques have better results than other solutions.

In this sense, the development of two auxiliary components of the proposed system can help to improve the solution's performance. The first is an algorithm capable of

evaluating the quality or error present in the MTM parameters, without taking into account any type of ground true or previous information, only the MTM parameters, and the available maps. A second component could use these measures to estimate the algorithms' input parameters.

10.3.8 Pairwise Map Merging Visual Evaluation

The two-dimensional visualization of three-dimensional information is a challenge and becomes almost unfeasible in the case of 3D occupancy grid maps. Thus, this thesis uses a map with reduced altitude for this purpose. Tables 6 and 7 present a visual evaluation of each of the steps of the algorithm.

As already mentioned, this map represents one of the worst cases for performing the merge as it has a lot of similarities between the regions of the map. Also, this map presents a reduced level of detail which makes the fusion activity an arduous activity.

As can be seen in Table 6, the original maps have small flaws and distortions that can affect the map merging. In this example, the map 1 has 180 degrees of rotation around the Z-axis and 9 m translation for the X and Y axes relative to the 0 map. To make viewing easier and more intuitive, the map 1 has been rotated.

The preprocessed maps have reduced size compared to the original maps due to the applied downsampling process. Still, in these maps, the continuity problem (edges and corners) is solved by applying the proposed filters.

Also, in Table 6 the detected keypoints can be visualized. The filtered keypoints represent the points of greatest relevance among the detected keypoints. As evidenced, the most significant keypoints are positioned across the entire length of the maps.

In Table 7 the orientations obtained by the proposed local orientation calculation method can be evidenced. Comparing the orientation obtained in similar regions between the two maps, it is possible to observe that there is a similarity between different regions of the maps.

To solve the region similarity problem, it is proposed to adjust the window size of the descriptor to accommodate the maximum size of the region that contains occupied cells in the maps. Thus, it is possible to retrieve the most significant correspondences in a prominent region on the maps.

Correspondences identified in keypoints that compose vertical or horizontal regions are relatively a problem for the correspondence calculation algorithm. Fortunately, as the requirements for the smooth functioning of the correspondences filter have been met, false matches are removed in this step.

Finally, the MTM parameters are calculated. Parameter scaling correction is applied. The map 1 is then transformed and the merge step is applied. At the end of the execution of the algorithm, the global map of this environment is available.

The final merge result still has some errors from the mapping algorithm, such as a

Table 6 – Pairwise map merging visual evaluation part 1.

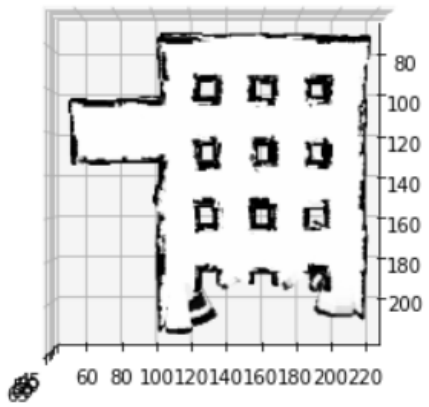
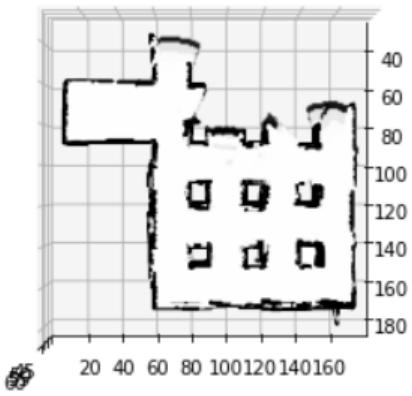
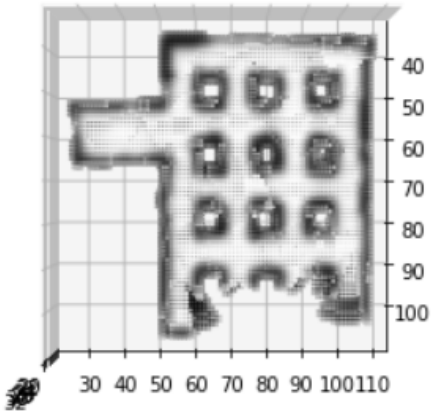
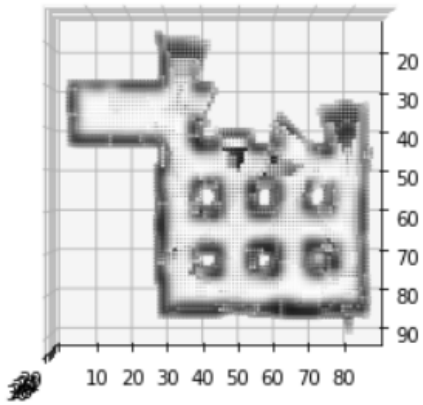
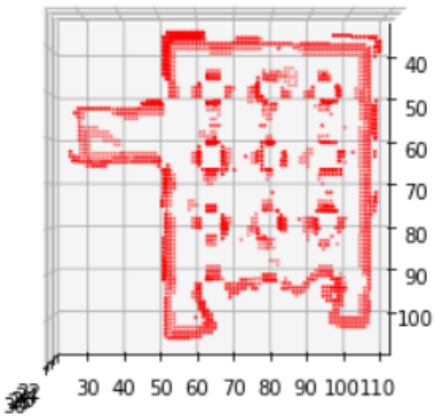
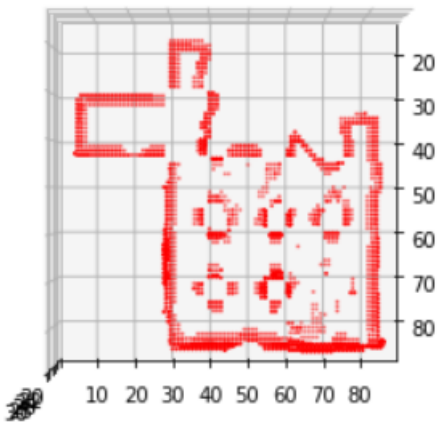
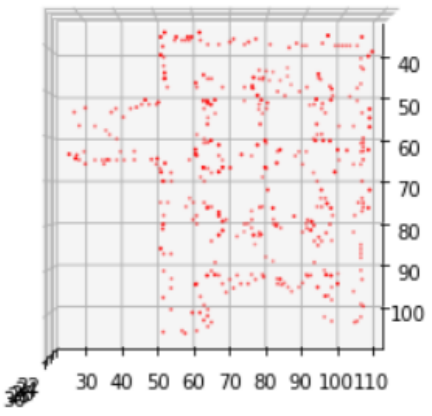
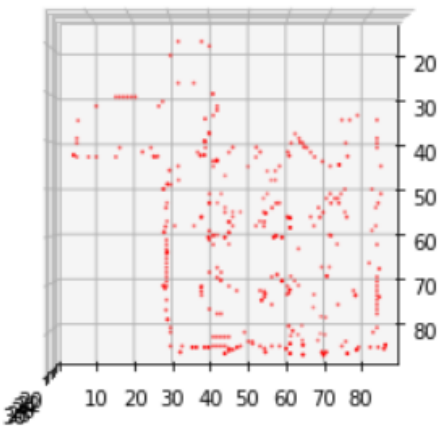
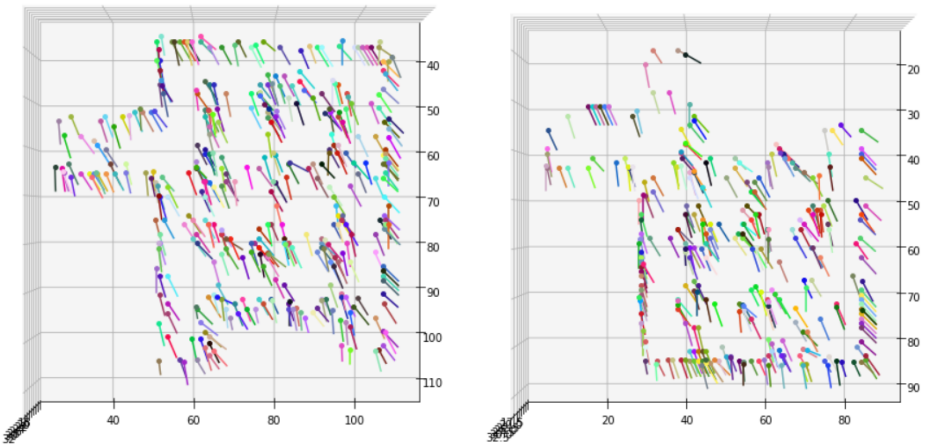
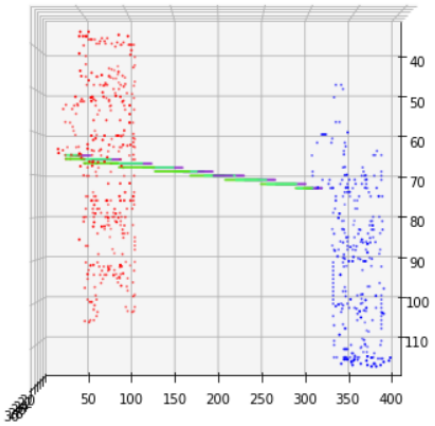
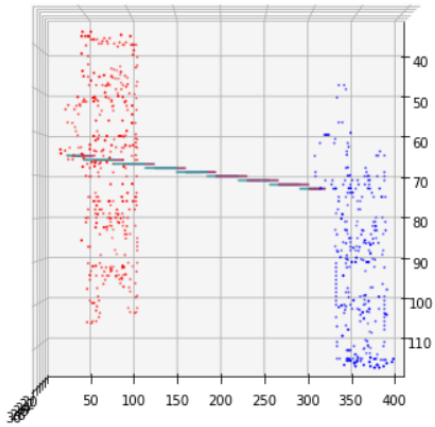
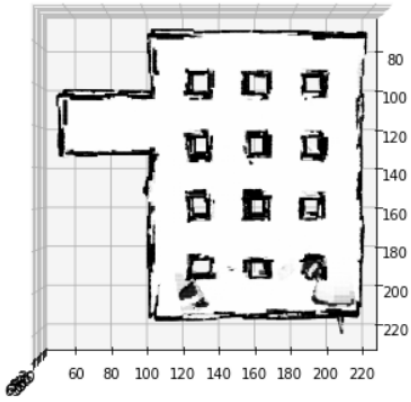
	MAP 0	MAP 1
Original maps		
Preprocessed maps		
FAST 3D keypoints		
Filtered keypoints		

Table 7 – Pairwise map merging visual evaluation part 2.

	MAP 0	MAP 1
Keypoint orientations		
Correspondences		
Filtered correspondences		
Merged maps		

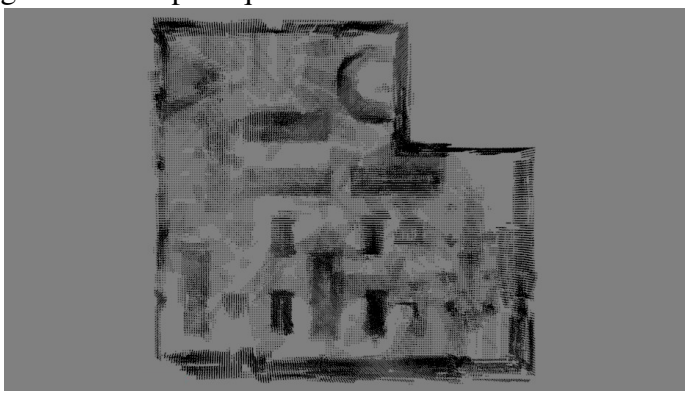
lack of continuity and distortions in some regions. In this thesis, no technical approaches for estimating corrections or distortions in maps are addressed, so such problems do not affect the obtained results.

10.3.9 Maps Visual Presentation

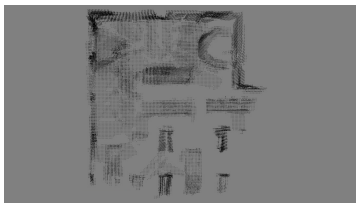
In this section, all maps obtained during the proposed experiments are presented, to provide a visual analysis of the results obtained. For ease of viewing, only occupied space cells are displayed. For each environment, the local maps obtained from each vehicle in different portions of the environment are presented, together with the global map resulting from the fusion process of these maps.

Figure 60 shows the maps obtained in the bookstore environment. In this environment, the round reception table, the corner of the wall to the right of this table, and the larger shelves were fundamental elements present in the local maps that enabled the merge process.

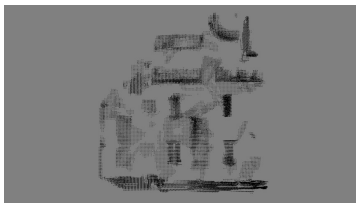
Figure 60 – Maps acquired from bookstore environment.



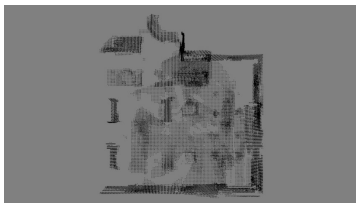
a) Global map.



b) UAV 0 map.



c) UAV 1 map.



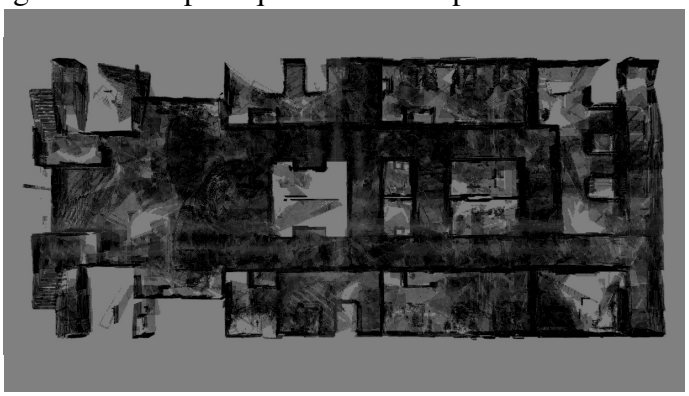
d) UAV 2 map.

Source: author.

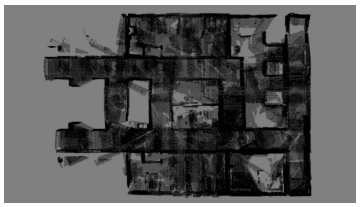
Figure 61 shows the maps obtained in the hospital environment. In this scenario, the lounge of reception and the well-defined corners of the long corridors made it possible to merge. Another important point was the position of the identified features on the ceiling. When the ceiling was present, as, in the case of this environment, the altitude reference points contribute to the correspondence filtering process.

Figure 62 shows the maps obtained in the maze environment. As already mentioned, the labyrinth environment was the most challenging case to perform the fusion due to the high similarity between the characteristics of the captured points of interest. As demon-

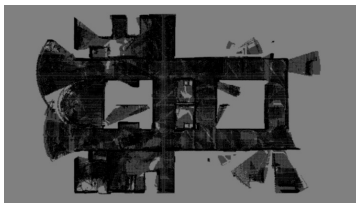
Figure 61 – Maps acquired from hospital environment.



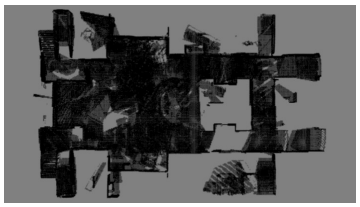
a) Global map.



b) UAV 0 map.



c) UAV 1 map.



d) UAV 2 map.

Source: author.

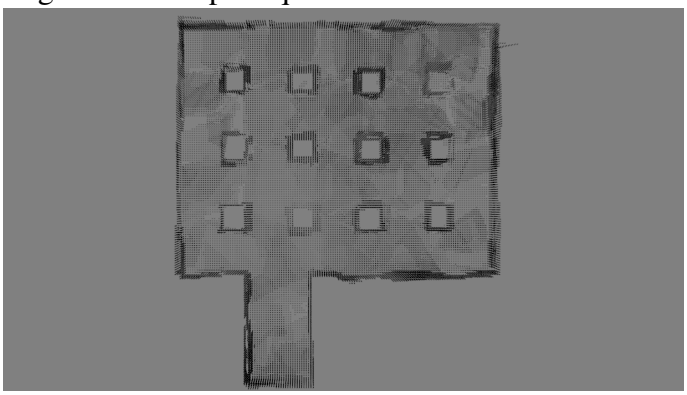
strated in Section 10.3.8, the entrance corridor to the maze was the only element that made it possible to merge the local maps in this environment. How the proposed keypoint detector retrieved the points of interest without losing vertical corner edge detail contributed to the successful merge process.

Figure 63 shows the maps obtained in the racetrack environment. In this environment, the list of keypoints detected in the straight horizontal corridors of the runway was fundamental for merging. A feature that enabled the local maps to merge in this environment was the constructive and detailed differences present in the two curves (left and right). In case these differences did not exist, the merging algorithm would probably overlap the two curves, needing more details for the correct merge process. The possibility of adjusting the descriptor window according to the dimensions of the map, as proposed, was also one of the elements that contributed to the final result obtained.

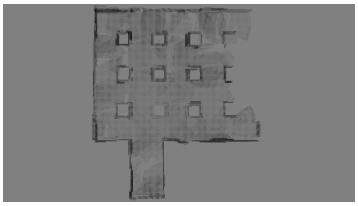
Figure 64 shows the maps obtained in the small house environment. In this environment, the elements present in the central room of the house to the edges of the walls enabled the merge of local maps. Within these elements, the sofa, the central table, and the different dimensions of the access doors to the different rooms of the house stand out. In addition, the other different elements in each room enabled were key elements that enabled the correspondences filter to eliminate false positives.

Figure 65 presents the maps obtained in the small warehouse environment. In this environment, the different types of shelves present on the upper and lower sides of the environment made it possible to align the maps horizontally. The correct altitude for vertical alignment of the maps was only obtained thanks to the edges of the boxes stacked

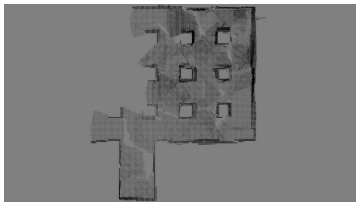
Figure 62 – Maps acquired from maze environment.



a) Global map.



b) UAV 0 map.

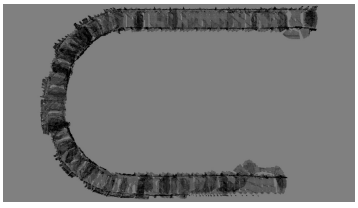


c) UAV 1 map.

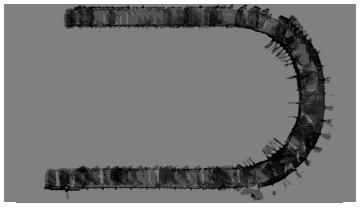
Source: author.

Figure 63 – Maps acquired from racetrack environment.

a) Global map.



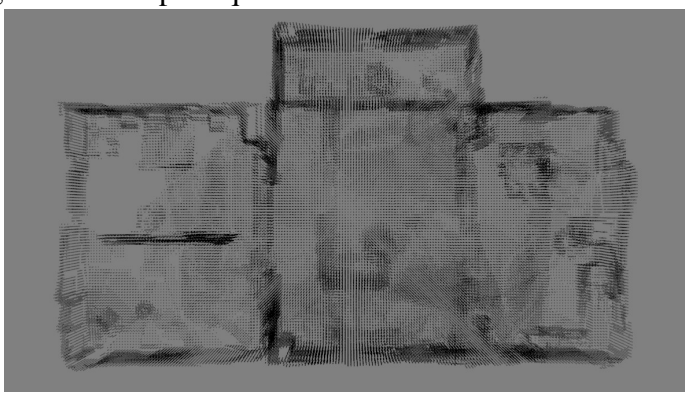
b) UAV 0 map.



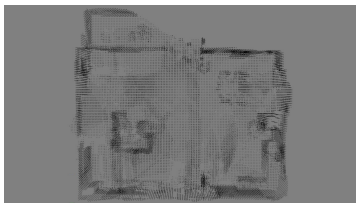
c) UAV 1 map.

Source: author.

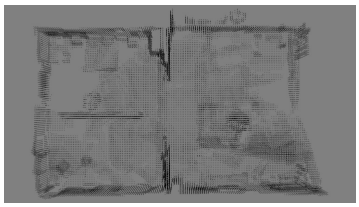
Figure 64 – Maps acquired from small house environment.



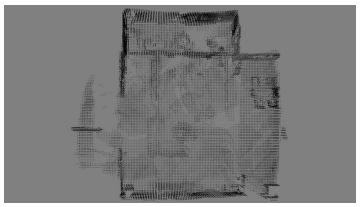
a) Global map.



b) UAV 0 map.



c) UAV 1 map.



d) UAV 2 map.

Source: author.

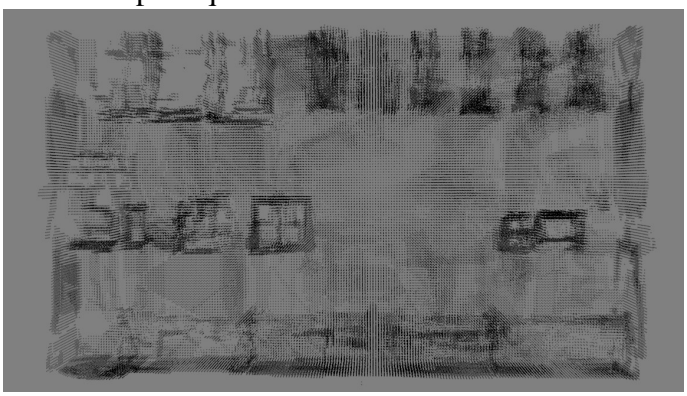
in the middle of the environment to the points of interest detected on the edges of the lower corners.

In general, map merging is a challenging problem, dependent on the characteristics of the environment in which it is being performed. The presented results were possible to be obtained because the proposed algorithms were able to recover good sets of correspondences between the presented local maps. This happens because the proposed keypoint detector can recover not only corners but also edges, vertical and horizontal, that connect these corners, composing points of interest with a greater semantic significance for the merging process. The proposed filters also helped to eliminate false positives. Without filtering processes, such as proper parametric adjustment for these filters, it is impossible to perform any occupancy grid map merging process.

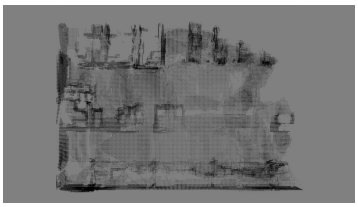
Another element that contributed to the merge process was the proper overlapping of areas on the presented local maps. Without enough overlap to capture common features, correct merging becomes impossible, which makes the local map merging process highly dependent on the way UAVs route their trajectories through the environment.

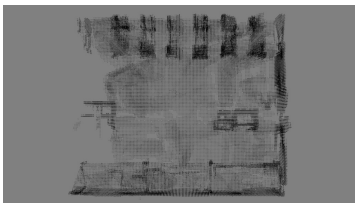
Although the activity of merging local maps is challenging and does not have a single solution, the solutions proposed in this thesis showed good performance and through them, it was possible to merge the local 3D occupancy grid maps for all environments proposed under the conditions mentioned. In contrast, the dependence on parametric adjustment was a problem not explored in this thesis, and it can be solved by concentrating research on parameter estimation through artificial intelligence or machine learning.

Figure 65 – Maps acquired from small warehouse environment.



a) Global map.





b) UAV 0 map. c) UAV 1 map.

Source: author.

10.4 Scalability Evaluation

To evaluate the scalability of the proposed system, an experiment was carried out varying the number of UAVs used in each scenario. In this experiment, the aim is to evaluate the average system execution time for an increasing number of UAVs.

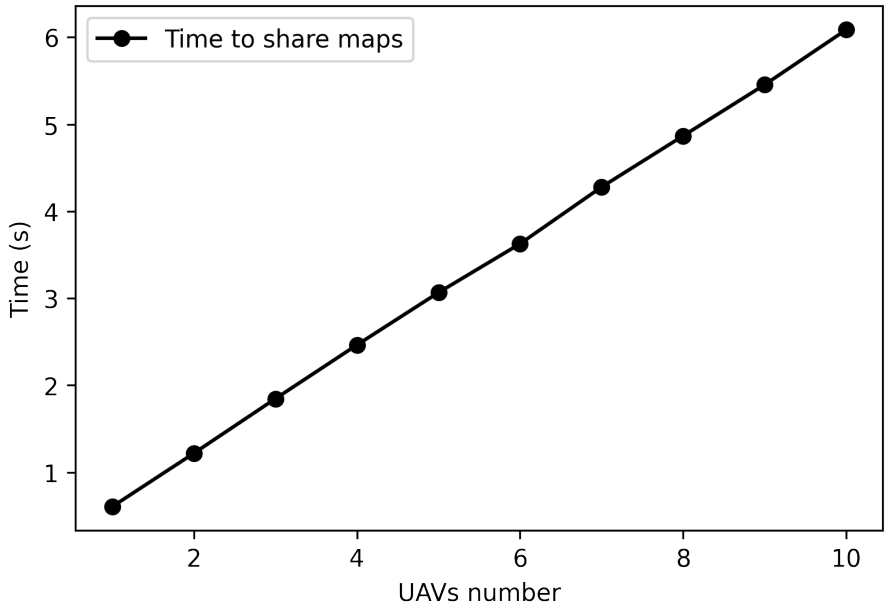
This way, it is assumed that each of the UAVs has acquired its partial map. Not necessarily all acquired maps have overlapping regions, but for each map, there is at least one overlapping region to another local map. In addition, the local maps obtained by each of the UAVs have different sizes. Also, it is assumed that all UAVs are in an adequate communication range.

From these conditions, a statistical analysis was performed to calculate the number of repetitions (samples) necessary for the experiment. Then a significance level $\alpha = 0.05$ was defined, which results in a confidence interval equal to $1 - \alpha = 0.95 = 95\ \%$. The standard deviation for the experiment was $\sigma \approx 2.92$ seconds. Afterward, the minimum number of repetitions for the experiment was calculated at $n = 11$. Then, 33 repetitions were performed taking into account different local maps and different regions distributed within the proposed environments.

In the first stage of the experiment, the UAVs shared their local maps. The average time for full sharing of all local maps among all UAVs can be seen in Figure 66.

It is observed that for a scenario containing 10 UAVs, the average time for full sharing of all local maps for all UAVs was around 6 seconds. This time in a real scenario can be relatively shorter because not all UAVs will always be in an acceptable communication

Figure 66 – Mean time to share all the maps on each UAV.

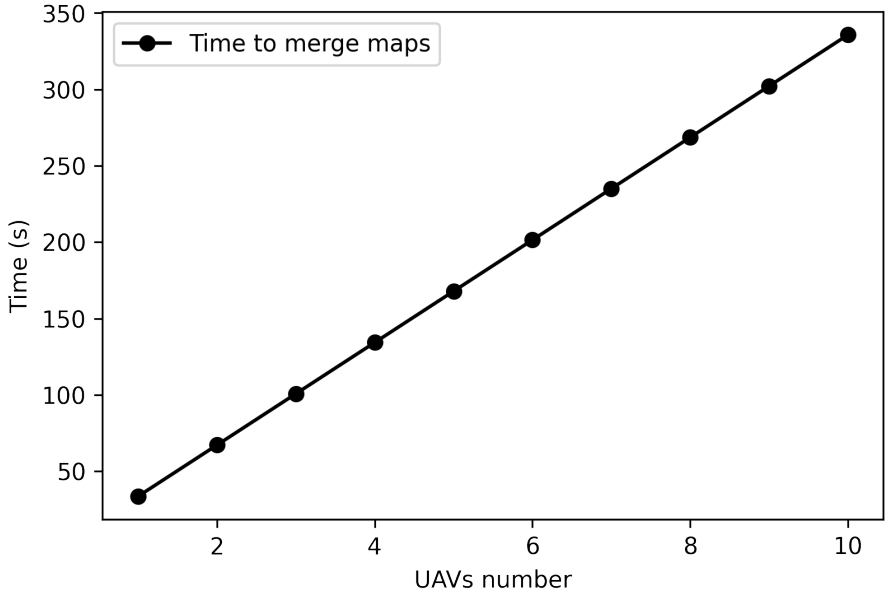


Source: author.

range, thus some sharing will occur by inheritance of global maps merged by their neighbors. In this proposed scalability test, the obtained time refers to the total sharing of all maps without considering possible shares by inheritance among neighbors.

After sharing the maps, they are merged. The average time for merging all local maps in each of the UAVs can be seen in Figure 67.

Figure 67 – Mean time to merge all the maps on each UAV.



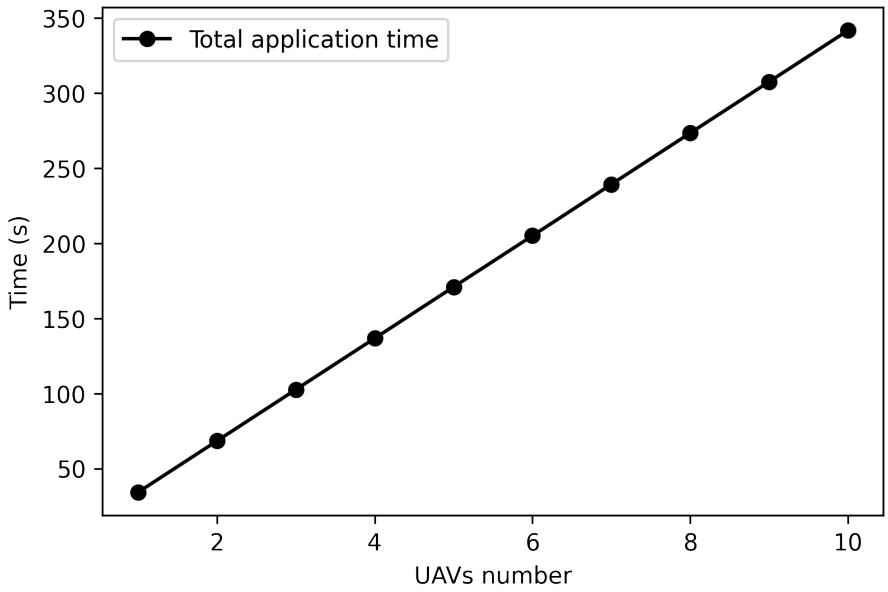
Source: author.

The average time for merging all local maps into a system containing 10 UAVs, taking into account the variability in map size and/or presence of overlapping regions, was close to 340 seconds. This time is the runtime taken for the complete generation of the global

map on each UAV, containing each of the local map representations of all the 10 UAVs.

Finally, the final average time of complete execution of the proposed system for each test scenario, taking into account the variability in the number of UAVs components of the distributed navigation system, is presented in Figure 68.

Figure 68 – Scalability test based on average time to complete application execution in each UAV.



Source: author.

The average run time of the system is an accumulated time for the transmission and merging of local maps in each of the UAVs. In this proposed experiment, the average system execution time in an application with 10 UAVs was close to 350 seconds for a scenario composed of 10 drones.

It is worth noticing that these tests represent the maximum time in a scenario where each UAV would have to carry out all the merging processes by itself. In a real scenario, the parameters obtained in the merger and the partial global maps could be shared, thus, there is no need to merge all the maps in each of the UAVs. This fact would significantly reduce the fusion time for the generation of the overall global map of the system. Thus, the results here are presented to demonstrate the worst-case situations that the system could reach.

Another important aspect to mention in the obtained results is that the linearity of the average execution times allows estimating the approximate time of execution of the system for a larger set of UAVs. Still, it demonstrates that the system scales still keeping the execution times within acceptable time limits. Despite these results, it is also important to highlight that for the type of applications this type of system is targeting, it is not expected that a very large number of UAVs is used (Schmuck; Chli, 2017; Tang *et al.*, 2019). So, a short scalability range analysis as the one here presented is enough to provide evidence that the system scales considering the scope of its desirable usage.

11 CONCLUSIONS

This chapter presents the final considerations from the performed research presented and discusses possible directions for future work.

11.1 Concluding Remarks

This thesis presented the design, development, and evaluation of a cooperative navigation system, with distributed architecture, to be used for multiple UAVs. This system is based on three basic activities: construction of three-dimensional occupancy grid maps, compressing and sharing maps between UAVs, and merging the local maps to create the global representation of the environment on each of the UAVs.

A solution based on HIMM was adopted to build the maps by reading the sensors and RGB-D camera of the simulated UAVs. These maps were stored in the 3D dynamic occupancy grid maps structures presented in this thesis.

The map compression system was specifically designed to reduce the size of maps before sharing. Then, the map transmission protocol (MTP) was proposed to identify nearby UAVs and then share the local maps as proposed.

The proposed map merging solution consists of the extension and adaptation of some components of the ORB algorithm to make them compatible for application in 3D occupancy grid maps. The proposed merging solution comprises a detector and descriptor, in addition to filters for keypoints and correspondences. In addition, keypoint orientation properties are obtained from potential field gradients. This solution also uses image processing techniques to preprocess the maps and remove noise and deformation from the mapping step to better calculate the best MTM parameters.

Several experiments were performed and the evaluation of the proposed solution was presented. The compression system proved to be efficient in reducing the size of maps. The transmission protocol can transmit maps with high dimensions in a few seconds.

To evaluate the map merging system, first, each component was evaluated and the quantitative results were presented and discussed. Then, the results of the complete pairwise map merging solution were presented and discussed. Finally, the functioning of the

proposed system is shown through a visual evaluation using a pair of 3D maps suitable to visualize the results in two dimensions. Then the local and global maps resultant from the execution of the proposed system are presented in a top view.

The main conclusion of this thesis is that the task of merging 3D occupancy grid maps is complex and does not have a trivial solution, but rather adequate solutions for each situation. Also, adding new information like colors to maps can somewhat improve the accuracy of the merging algorithms. In this area of research, there are still many other open problems to be explored.

11.2 Future Works

This section presents the possible future works. The future works regarding the mapping activity are presented in Section 11.2.1. Section 11.2.2 presents possible future works regarding the map sharing system. Finally, Section 11.2.3 presents the possible advances and ideas that can become future research involving 3D occupancy grid map merging systems.

11.2.1 3D Occupancy Grid Mapping

In this thesis, maps are generated based on point clouds obtained from RGB-D cameras. These are practical sensors to be integrated into the system and allow it to capture the details of the environment. However, its use in the construction of maps requires the vehicle to rotate around its central axis, during the exploration, to obtain readings from several different positions and orientations. With this, it becomes common to get an increased error in the construction of maps. Also, as this rotational movement causes the vehicle to waste a little time in exploration, if this is a system requirement it can be a problem. Therefore, in future works in the construction of maps, there is the addition of multiple RGB-D sensors to each vehicle or also the use of laser sensors such as 3D laser, to improve the estimates and obtain a total view of the environment in which the vehicle is inserted. Sensor fusion of readings from all these sensors is also an alternative to these problems.

11.2.2 Map Sharing

The map transmission protocol proposed in this thesis takes into account only the map data to decide when to share information between vehicles. For this reason, this is a high-level protocol. In future work, there is the integration of the protocol with information from the network layers in the system, intending to contribute to a better distribution of packets in the network, avoiding retransmission and congestion of this channel when the number of vehicles becomes higher. This integration would also allow information to travel along defined routes, avoiding retransmission of information already transmitted.

In this thesis, TCP/IP was used to guarantee the delivery of messages between UAVs. Another point of study could be the utilization of UDP to speed up the information transmission process, perhaps creating a stream for the distribution of maps.

11.2.3 3D Occupancy Grid Map Merging

In the proposed map merging system, the descriptor needs to be improved so the system can be used in distinct environments without the need to adjust parameters. The use of machine learning can be employed to solve this problem and estimate the input parameters. The algorithm for calculating the correspondences can be improved to reduce the computational cost. Furthermore, a robust way of evaluating MTMs needs to be developed, without taking into account previous information from the environment or any kind of ground truth. Furthermore, it is believed that the inclusion of global location systems in the application can contribute to reducing the need for overlapping between map regions. An efficient system capable of evaluating the availability of processing hardware in each of the UAVs to better distribute the merge tasks can contribute to reducing the energy consumption of the application.

REFERENCES

AL-KAFF, A. *et al.* Survey of computer vision algorithms and applications for unmanned aerial vehicles. **Expert Systems with Applications**, [S.l.], v. 92, n. Supplement C, p. 447 – 463, 2018.

ARTIEDA, J. *et al.* Visual 3-D SLAM from UAVs. **Journal of Intelligent and Robotic Systems**, [S.l.], v. 55, n. 4, p. 299, Jan 2009.

VETRELLA, A. R. *et al.* **Autonomous Flight in GPS-Challenging Environments Exploiting Multi-UAV Cooperation and Vision-aided Navigation**. [S.l.: s.n.], 2017.

BACHRACH, A. *et al.* RANGE–Robust autonomous navigation in GPS-denied environments. **Journal of Field Robotics**, [S.l.], v. 28, n. 5, p. 644–666, 2011.

BAILEY, T.; DURRANT-WHYTE, H. Simultaneous localization and mapping (SLAM): part ii. **IEEE robotics & automation magazine**, [S.l.], v. 13, n. 3, p. 108–117, 2006.

BASSO, M.; DE FREITAS, E. P. A UAV Guidance System Using Crop Row Detection and Line Follower Algorithms. **J. Intell. Robotic Syst.**, [S.l.], v. 97, n. 3, p. 605–621, 2020.

Birk, A.; Carpin, S. Merging Occupancy Grid Maps From Multiple Robots. **Proceedings of the IEEE**, [S.l.], v. 94, n. 7, p. 1384–1397, July 2006.

BOKOVOY, A.; MURAVIEV, K.; YAKOVLEV, K. Map-Merging Algorithms for Visual SLAM: feasibility study and empirical evaluation. *In*: ARTIFICIAL INTELLIGENCE, 2020, Cham. **Proceedings [. . .]** Springer International Publishing, 2020. p. 46–60.

BONANNI, T. M.; DELLA CORTE, B.; GRISETTI, G. 3-D Map Merging on Pose Graphs. **IEEE Robotics and Automation Letters**, [S.l.], v. 2, n. 2, p. 1031–1038, 2017.

BONIN-FONT, F.; ORTIZ, A.; OLIVER, G. Visual Navigation for Mobile Robots: a survey. **Journal of Intelligent and Robotic Systems**, [S.l.], v. 53, n. 3, p. 263, May 2008.

BORENSTEIN, J.; KOREN, Y. *et al.* The vector field histogram-fast obstacle avoidance for mobile robots. **IEEE transactions on robotics and automation**, [S.l.], v. 7, n. 3, p. 278–288, 1991.

BORENSTEIN, J.; KOREN, Y. *et al.* Histogramic in-motion mapping for mobile robot obstacle avoidance. **IEEE Transactions on robotics and automation**, [S.l.], v. 7, n. 4, p. 535–539, 1991.

BRESENHAM, J. E. Algorithm for computer control of a digital plotter. **IBM Systems Journal**, [S.l.], v. 4, n. 1, p. 25–30, 1965.

Cadena, C. *et al.* Past, Present, and Future of Simultaneous Localization and Mapping: toward the robust-perception age. **IEEE Transactions on Robotics**, [S.l.], v. 32, n. 6, p. 1309–1332, Dec 2016.

CALONDER, M. *et al.* Brief: binary robust independent elementary features. *In*: EUROPEAN CONFERENCE ON COMPUTER VISION, 2010. **Proceedings [...]** [S.l.: s.n.], 2010. p. 778–792.

CHOUDHARY, S. *et al.* Multi Robot Object-Based SLAM. *In*: INTERNATIONAL SYMPOSIUM ON EXPERIMENTAL ROBOTICS, 2016., 2017, Cham. **Proceedings [...]** Springer International Publishing, 2017. p. 729–741.

CHOWDHARY, G. *et al.* GPS-denied Indoor and Outdoor Monocular Vision Aided Navigation and Control of Unmanned Aircraft. **Journal of Field Robotics**, [S.l.], v. 30, n. 3, p. 415–438, 2013.

CHUM, O.; PAJDLA, T.; STURM, P. The geometric error for homographies. **Computer Vision and Image Understanding**, [S.l.], v. 97, n. 1, p. 86–102, 2005.

VALAVANIS, K. P.; VACHTSEVANOS, G. J. (Ed.). **Cooperative Mission Planning for Multi-UAV Teams**. Dordrecht: Springer Netherlands, 2015. p. 1447–1490.

Dapper e Silva, T. *et al.* STFANET: sdn-based topology management for flying ad hoc network. **IEEE Access**, [S.l.], v. 7, p. 173499–173514, 2019.

DE MORAES, R. S.; DE FREITAS, E. P. Multi-UAV Based Crowd Monitoring System. **IEEE Transactions on Aerospace and Electronic Systems**, [S.l.], v. 56, n. 2, p. 1332–1345, 2020.

Dinnissen, P.; Givigi, S. N.; Schwartz, H. M. Map merging of Multi-Robot SLAM using Reinforcement Learning. *In*: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS (SMC), 2012., 2012. **Proceedings [...]** [S.l.: s.n.], 2012. p. 53–60.

Drumheller, M. Mobile Robot Localization Using Sonar. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v. PAMI-9, n. 2, p. 325–332, March 1987.

Elfes, A.; Matthies, L. Sensor integration for robot navigation: combining sonar and stereo range data in a grid-based representataion. *In*: IEEE CONFERENCE ON DECISION AND CONTROL, 26., 1987. **Proceedings [...]** [S.l.: s.n.], 1987. v. 26, p. 1802–1807.

ENDRES, F. *et al.* An evaluation of the RGB-D SLAM system. *In*: ICRA, 2012. **Proceedings [...]** [S.l.: s.n.], 2012. v. 3, n. c, p. 1691–1696.

EVERS, L. *et al.* Robust UAV mission planning. **Annals of Operations Research**, [S.l.], v. 222, n. 1, p. 293–315, Nov 2014.

FERRãO, V. T.; VINHAL, C. D. N.; DA CRUZ, G. An Occupancy Grid Map Merging Algorithm Invariant to Scale, Rotation and Translation. *In*: BRAZILIAN CONFERENCE ON INTELLIGENT SYSTEMS (BRACIS), 2017., 2017. **Proceedings [...]** [S.l.: s.n.], 2017. p. 246–251.

FISCHLER, M. A.; BOLLES, R. C. Random Sample Consensus: a paradigm for model fitting with applications to image analysis and automated cartography. **Commun. ACM**, New York, NY, USA, v. 24, n. 6, p. 381–395, June 1981.

Fox, D. *et al.* Distributed Multirobot Exploration and Mapping. **Proceedings of the IEEE**, [S.l.], v. 94, n. 7, p. 1325–1339, July 2006.

FRANZ, M. O.; MALLOT, H. A. Biomimetic robot navigation. **Robotics and Autonomous Systems**, [S.l.], v. 30, n. 1, p. 133 – 153, 2000.

GALLISTEL, C. R. **The organization of learning.** [S.l.]: The MIT Press, 1990.

GAZEBO Tutorials. [Online; accessed 22-November-2019], `http://gazebosim.org/tutorials/`.

GONZALEZ, R. **Digital image processing**. 3. ed. Upper Saddle River, N.J: Prentice Hall, 2008.

GUO, K. *et al.* Ultra-wideband based cooperative relative localization algorithm and experiments for multiple unmanned aerial vehicles in GPS denied environments. **International Journal of Micro Air Vehicles**, [S.l.], v. 9, n. 3, p. 169–186, 2017.

HAMMING, R. W. Error detecting and error correcting codes. **The Bell System Technical Journal**, [S.l.], v. 29, n. 2, p. 147–160, 1950.

HAN, L. *et al.* Circular formation tracking control for time-delayed second-order multi-agent systems with multiple leaders. *In*: IEEE CHINESE GUIDANCE, NAVIGATION AND CONTROL CONFERENCE (CGNCC), 2016., 2016. **Proceedings [...]** [S.l.: s.n.], 2016. p. 1648–1653.

HARRIS, C.; STEPHENS, M. A combined corner and edge detector. *In*: IN PROC. OF FOURTH ALVEY VISION CONFERENCE, 1988. **Proceedings [...]** [S.l.: s.n.], 1988. p. 147–151.

HESTER, T.; LOPES, M.; STONE, P. Learning exploration strategies in model-based reinforcement learning. *In*: AUTONOMOUS AGENTS AND MULTI-AGENT SYSTEMS, 2013., 2013. **Proceedings [...]** [S.l.: s.n.], 2013. p. 1069–1076.

HORNUNG, A. *et al.* OctoMap: an efficient probabilistic 3d mapping framework based on octrees. **Autonomous robots**, [S.l.], v. 34, n. 3, p. 189–206, 2013.

HOWARD, A.; PARKER, L. E.; SUKHATME, G. S. The SDR experience: experiments with a large-scale heterogeneous mobile robot team. In: **Experimental Robotics IX**. [S.l.]: Springer, 2006. p. 121–130.

2RD (Ed.). **Robust statistics**. [S.l.]: Hoboken, NJ: Wiley, 2009.

JIAN, L. *et al.* Vision Feature Extraction Algorithm for Occupancy Grid Maps Merging. *In*: INTERNATIONAL CONFERENCE ON COMMUNICATION AND INFORMATION SYSTEMS, 2017., 2017, New York, NY, USA. **Proceedings [...]** Association for Computing Machinery, 2017. p. 290–293. (ICCIS 2017).

JIANG, Z. *et al.* Simultaneous Merging Multiple Grid Maps Using the Robust Motion Averaging. **Journal of Intelligent & Robotic Systems**, [S.l.], v. 94, n. 3, p. 655–668, Jun 2019.

JONKER, P. P. Morphological Operations on 3D and 4D Images: from shape primitive detection to skeletonization. *In*: DISCRETE GEOMETRY FOR COMPUTER IMAGERY, 2000, Berlin, Heidelberg. **Proceedings [...]** Springer Berlin Heidelberg, 2000. p. 371–391.

KANELLAKIS, C.; NIKOLAKOPOULOS, G. Survey on Computer Vision for UAVs: current developments and trends. **Journal of Intelligent & Robotic Systems**, [S.l.], v. 87, n. 1, p. 141–168, Jul 2017.

KJER, H. M.; WILM, J. **Evaluation of surface registration algorithms for PET motion correction**. 2010. B.S. thesis — Citeseer, 2010.

KOCH, P. *et al.* Multi-Robot Localization and Mapping Based on Signed Distance Functions. **Journal of Intelligent & Robotic Systems**, [S.l.], v. 83, n. 3, p. 409–428, Sep 2016.

Koenig, N.; Howard, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. *In*: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS) (IEEE CAT. NO.04CH37566), 2004., 2004. **Proceedings [...]** [S.l.: s.n.], 2004. v. 3, p. 2149–2154 vol.3.

KOREN, Y.; BORENSTEIN, J. Potential field methods and their inherent limitations for mobile robot navigation. *In*: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 1991., 1991. **Proceedings [...]** [S.l.: s.n.], 1991. p. 1398–1404.

KRUSE, T. *et al.* Human-aware robot navigation: a survey. **Robotics and Autonomous Systems**, [S.l.], v. 61, n. 12, p. 1726 – 1743, 2013.

LEE, H.; ROH, B.; LEE, B. Multi-hypothesis map merging with sinogram-based PSO for multi-robot systems. **Electronics Letters**, [S.l.], v. 52, n. 14, p. 1213–1214, 2016.

LEE, M.-J. *et al.* Adaptive row major order: a new space filling curve for efficient spatial join processing in the transform space. **Journal of Systems and Software**, [S.l.], v. 78, n. 3, p. 257–269, 2005.

LEERINK, L.; SCHULTZ, S. R.; JABRI, M. A. A reinforcement learning exploration strategy based on ant foraging mechanisms. *In*: SIXTH AUSTRALIAN CONFERENCE ON NEURAL NETWORKS, 1995. **Proceedings [...]** [S.l.: s.n.], 1995. p. 217–220.

LENAC, K. *et al.* Fast Active SLAM for Accurate and Complete Coverage Mapping of Unknown Environments. *In*: INTELLIGENT AUTONOMOUS SYSTEMS 13, 2016, Cham. **Proceedings [...]** Springer International Publishing, 2016. p. 415–428.

LEONARD, J. J.; DURRANT-WHYTE, H. F. Mobile robot localization by tracking geometric beacons. **IEEE Transactions on robotics and Automation**, [S.l.], v. 7, n. 3, p. 376–382, 1991.

LEVITT, T. S.; LAWTON, D. T. Qualitative navigation for mobile robots. **Artificial Intelligence**, [S.l.], v. 44, n. 3, p. 305 – 360, 1990.

LOWE, D. Object recognition from local scale-invariant features. *In*: SEVENTH IEEE INTERNATIONAL CONFERENCE ON COMPUTER VISION, 1999. **Proceedings [...]** IEEE, 1999.

Lu, Y.; Song, D. Visual Navigation Using Heterogeneous Landmarks and Unsupervised Geometric Constraints. **IEEE Transactions on Robotics**, [S.l.], v. 31, n. 3, p. 736–749, June 2015.

Lázaro, M. T. *et al.* Multi-robot SLAM using condensed measurements. *In*: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 2013., 2013. **Proceedings [...]** [S.l.: s.n.], 2013. p. 1069–1076.

MA, L. *et al.* Merging grid maps of different resolutions by scaling registration. **Robotica**, [S.l.], v. 34, n. 11, p. 2516–2531, 2016.

Makarenko, A. A. *et al.* An experiment in integrated exploration. *In*: IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 2002. **Proceedings [...]** [S.l.: s.n.], 2002. v. 1, p. 534–539 vol.1.

MATHWORKS, I. **Mathworks Examples**. Accessed: 2019-09-09, `https://www.mathworks.com/help/robotics/examples/`.

Meng, W. *et al.* Decentralized Multi-UAV Flight Autonomy for Moving Convoys Search and Track. **IEEE Transactions on Control Systems Technology**, [S.l.], v. 25, n. 4, p. 1480–1487, July 2017.

MICHAEL, N. *et al.* Collaborative mapping of an earthquake-damaged building via ground and aerial robots. **Journal of Field Robotics**, [S.l.], v. 29, n. 5, p. 832–841, 2012.

NASH, A.; KOENIG, S.; TOVEY, C. Lazy Theta*: any-angle path planning and path length analysis in 3d. *In*: TWENTY-FOURTH AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2010. **Proceedings [...]** [S.l.: s.n.], 2010.

NONAMI, K. *et al.* Autonomous control systems and vehicles. **Intelligent Systems, Control and Automation: Science and Engineering**, [S.l.], v. 65, 2013.

OPPENHEIM, A. V.; SCHAFER, R. W.; BUCK, J. R. **Discrete-time signal processing**. 2. ed. Upper Saddle River, N.J.: Prentice Hall, 1999. 168-169 p.

ORFANUS, D.; DE FREITAS, E. P.; ELIASSEN, F. Self-Organization as a Supporting Paradigm for Military UAV Relay Networks. **IEEE Communications Letters**, [S.l.], v. 20, n. 4, p. 804–807, April 2016.

PARK, J. *et al.* Map merging of rotated, corrupted, and different scale maps using rectangular features. *In*: IEEE/ION PLANS 2016, 2016. **Proceedings [...]** [S.l.: s.n.], 2016. p. 535–543.

PEREZ-GRAU, F. J. *et al.* An architecture for robust UAV navigation in GPS-denied areas. **Journal of Field Robotics**, [S.l.], v. 35, n. 1, p. 121–145, 2018.

Pi, S. *et al.* Stereo visual SLAM system in underwater environment. *In*: OCEANS 2014 - TAIPEI, 2014. **Proceedings [...]** [S.l.: s.n.], 2014. p. 1–5.

PWC. **Global market for commercial applications of drone technology valued at over $127bn**. [Online; accessed 24-December-2019], https://pwc.blogs.com/.

PX4 Open Source Autopilot. [Online; accessed 22-November-2019], `https://px4.io/`.

Qin, H. *et al.* Autonomous Exploration and Mapping System Using Heterogeneous UAVs and UGVs in GPS-Denied Environments. **IEEE Transactions on Vehicular Technology**, [S.l.], v. 68, n. 2, p. 1339–1350, Feb 2019.

QUIGLEY, M.; GERKEY, B.; SMART, W. **Programming Robots with ROS**: a practical introduction to the robot operating system. [S.l.]: O'Reilly Media, 2015.

RAMIREZ-ATENCIA, C. *et al.* Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms. **Soft Computing**, [S.l.], v. 21, n. 17, p. 4883–4900, Sep 2017.

RANGANATHAN, A.; DELLAERT, F. Online probabilistic topological mapping. **The International Journal of Robotics Research**, [S.l.], v. 30, n. 6, p. 755–771, 2011.

ROSIN, P. L. Measuring corner properties. **Computer Vision and Image Understanding**, [S.l.], v. 73, n. 2, p. 291–307, 1999.

ROSTEN, E.; DRUMMOND, T. Machine Learning for High-Speed Corner Detection. *In*: COMPUTER VISION – ECCV 2006, 2006, Berlin, Heidelberg. **Proceedings [...]** Springer Berlin Heidelberg, 2006. p. 430–443.

RUBLEE, E. *et al.* ORB: an efficient alternative to sift or surf. *In*: INTERNATIONAL CONFERENCE ON COMPUTER VISION, 2011., 2011. **Proceedings [...]** [S.l.: s.n.], 2011. p. 2564–2571.

SAEEDI, S. *et al.* Map merging for multiple robots using Hough peak matching. **Robotics and Autonomous Systems**, [S.l.], v. 62, n. 10, p. 1408 – 1424, 2014.

SASAKI, S. A practical computational technique for mobile robot navigation. *In*: IEEE INTERNATIONAL CONFERENCE ON CONTROL APPLICATIONS (CAT. NO.98CH36104), 1998., 1998. **Proceedings [...]** [S.l.: s.n.], 1998. v. 2, p. 1323–1327 vol.2.

Schmuck, P.; Chli, M. Multi-UAV collaborative monocular SLAM. *In*: IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2017., 2017. **Proceedings [...]** [S.l.: s.n.], 2017. p. 3863–3870.

SCHMUCK, P.; SCHERER, S. A.; ZELL, A. Hybrid Metric-Topological 3D Occupancy Grid Maps for Large-scale Mapping. **IFAC-PapersOnLine**, [S.l.], v. 49, n. 15, p. 230–235, 2016. 9th IFAC Symposium on Intelligent Autonomous Vehicles IAV 2016.

SHI, J.; TOMASI. Good features to track. *In*: PROCEEDINGS OF IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 1994., 1994. **Proceedings [...]** [S.l.: s.n.], 1994. p. 593–600.

SIEGWART, R.; NOURBAKHSH, I. R.; SCARAMUZZA, D. **Introduction to autonomous mobile robots**. [S.l.]: MIT press, 2011.

SINGLA, N.; GARG, D. String matching algorithms and their applicability in various applications. **International journal of soft computing and engineering**, [S.l.], v. 1, n. 6, p. 218–222, 2012.

SIPIRAN, I.; BUSTOS, B. Harris 3D: a robust extension of the harris operator for interest point detection on 3d meshes. **The Visual Computer**, [S.l.], v. 27, p. 963–976, 11 2011.

STACHNISS, C. **Exploration and mapping with mobile robots**. 2006. Tese (Doutorado em Engenharia Elétrica) — Citeseer, 2006.

STACHNISS, C.; LEONARD, J. J.; THRUN, S. Simultaneous Localization and Mapping. In: SICILIANO, B.; KHATIB, O. (Ed.). **Springer Handbook of Robotics**. Cham: Springer International Publishing, 2016. p. 1153–1176.

Tang, Y. *et al.* Vision-Aided Multi-UAV Autonomous Flocking in GPS-Denied Environment. **IEEE Transactions on Industrial Electronics**, [S.l.], v. 66, n. 1, p. 616–626, Jan 2019.

THRUN, S. *et al.* Robotic mapping: a survey. **Exploring artificial intelligence in the new millennium**, [S.l.], v. 1, n. 1-35, p. 1, 2002.

THRUN, S. *et al.* **Probabilistic Robotics**. [S.l.]: MIT Press, 2005.

TRUJILLO, A. C. *et al.* Using Natural Language to Enable Mission Managers to Control Multiple Heterogeneous UAVs. *In*: ADVANCES IN HUMAN FACTORS IN ROBOTS AND UNMANNED SYSTEMS: PROCEEDINGS OF THE AHFE 2016 INTERNATIONAL CONFERENCE ON HUMAN FACTORS IN ROBOTS AND UNMANNED SYSTEMS, JULY 27-31, 2016, WALT DISNEY WORLD®, FLORIDA, USA, 2017, Cham. **Proceedings [...]** Springer International Publishing, 2017. p. 267–280.

TRUJILLO, J.-C. *et al.* Cooperative Monocular-Based SLAM for Multi-UAV Systems in GPS-Denied Environments. **Sensors**, [S.l.], v. 18, n. 5, 2018.

URZUA, S.; MUNGUíA, R.; GRAU, A. Vision-based SLAM system for MAVs in GPS-denied environments. **International Journal of Micro Air Vehicles**, [S.l.], v. 9, n. 4, p. 283–296, 2017.

VAN LOAN, C. F. Generalizing the singular value decomposition. **SIAM Journal on numerical Analysis**, [S.l.], v. 13, n. 1, p. 76–83, 1976.

Velásquez Hernández, C. A.; Prieto Ortiz, F. A. A real-time map merging strategy for robust collaborative reconstruction of unknown environments. **Expert Systems with Applications**, [S.l.], v. 145, p. 113109, 2020.

Vetrella, A. R.; Fasano, G.; Accardo, D. Cooperative navigation in GPS-challenging environments exploiting position broadcast and vision-based tracking. *In*: INTERNATIONAL CONFERENCE ON UNMANNED AIRCRAFT SYSTEMS (ICUAS), 2016., 2016. **Proceedings [...]** [S.l.: s.n.], 2016. p. 447–456.

BASSO, M.; DE FREITAS, E. P. **Vision in indoor and outdoor drones**. [S.l.]: Institution of Engineering and Technology, 2020. p. 261–280. (Control, Robotics & Sensors).

WANG, C.-L. *et al.* Bearing-only Visual SLAM for Small Unmanned Aerial Vehicles in GPS-denied Environments. **International Journal of Automation and Computing**, [S.l.], v. 10, n. 5, p. 387–396, Oct 2013.

YAMAUCHI, B. A frontier-based approach for autonomous exploration. *In*: CIRA, 1997. **Proceedings [...]** [S.l.: s.n.], 1997. v. 97, p. 146.

Yue, Y. *et al.* Hierarchical Probabilistic Fusion Framework for Matching and Merging of 3-D Occupancy Maps. **IEEE Sensors Journal**, [S.l.], v. 18, n. 21, p. 8933–8949, 2018.

YUE, Y. *et al.* Probabilistic Fusion Framework for Collaborative Robots 3D Mapping. *In*: INTERNATIONAL CONFERENCE ON INFORMATION FUSION (FUSION), 2018., 2018. **Proceedings [...]** [S.l.: s.n.], 2018. p. 488–491.

ZHAO, W.; LI, R.; ZHANG, H. Finite-time distributed formation tracking control of multi-UAVs with a time-varying reference trajectory. **IMA Journal of Mathematical Control and Information**, [S.l.], p. dnx028, 2017.

ZHAO, Z. *et al.* Software-defined unmanned aerial vehicles networking for video dissemination services. **Ad Hoc Networks**, [S.l.], v. 83, p. 68 – 77, 2019.

Zhou, H. *et al.* StructSLAM: visual slam with building structure lines. **IEEE Transactions on Vehicular Technology**, [S.l.], v. 64, n. 4, p. 1364–1375, April 2015.