



**UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA**

LAURA RIBEIRO

**SIMULTANEOUS LOCALIZATION AND MAPPING TECHNIQUES USING A
NANO QUADCOPTER**

Uberlândia

2024

LAURA RIBEIRO

**SIMULTANEOUS LOCALIZATION AND MAPPING TECHNIQUES USING A
NANO QUADCOPTER**

Dissertation presented to the Faculty of Electrical Engineering at the Federal University of Uberlândia as a partial requirement for obtaining the **Master of Science** degree.

Concentration Area: Control and Automation

Advisor: Prof. Dr. Gabriela Vieira Lima

Co-advisor: Prof. Dr. Aniel Silva de Morais

Federal University of Uberlândia
Faculty of Electrical Engineering

Uberlândia
2024

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

R484 2024	<p>Ribeiro, Laura, 1998- Simultaneous Localization and Mapping Techniques Using a Nano Quadcopter [recurso eletrônico] / Laura Ribeiro. - 2024.</p> <p>Orientadora: Gabriela Vieira Lima. Coorientador: Aniel Silva de Moraes. Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Engenharia Elétrica. Modo de acesso: Internet. Disponível em: http://doi.org/10.14393/ufu.di.2024.219 Inclui bibliografia. Inclui ilustrações.</p> <p>1. Engenharia elétrica. I. Lima, Gabriela Vieira, 1990- , (Orient.). II. Moraes, Aniel Silva de, 1979- , (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Engenharia Elétrica. IV. Título.</p> <p style="text-align: right;">CDU: 621.3</p>
--------------	--

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074



ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Engenharia Elétrica				
Defesa de:	Dissertação de Mestrado, 792, PPGEELT				
Data:	Quinze de março de dois mil e vinte e quatro	Hora de início:	14:00	Hora de encerramento:	15:50
Matrícula do Discente:	12212EEL001				
Nome do Discente:	Laura Ribeiro				
Título do Trabalho:	Simultaneous Localization and Mapping Techniques Using a Nano Quadcopter				
Área de concentração:	Sistemas de Energia Elétrica				
Linha de pesquisa:	Eletrônica Industrial, Sistemas e Controles Eletrônicos				
Projeto de Pesquisa de vinculação:	Coordenador do projeto: Aniel Silva de Moraes. Título do projeto: Controle Cooperativo de Enxame de Veículos Aéreos não Tripulados Especializados no Monitoramento e Detecção de Falhas na Indústria 4.0. Agência financiadora: CNPq. Número do processo na agência financiadora: 403054/2021-4. Vigência do projeto: 17/03/2022 - 31/03/2025.				

Reuniu-se por meio de videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Engenharia Elétrica, assim composta:

Professores Doutores: Daniel Costa Ramos (UFU), Danielli Araújo Lima (IFTM), e Gabriela Vieira Lima, orientadora da discente.

Iniciando os trabalhos o presidente da mesa, Dr^a. Gabriela Vieira Lima, apresentou a Comissão Examinadora e a candidata, agradeceu a presença do público, e concedeu à discente a palavra para a exposição do seu trabalho. A duração da apresentação da discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir a candidata. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando a candidata:

APROVADA.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre. O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme, foi assinada pela Banca Examinadora.



Documento assinado eletronicamente por **Gabriela Vieira Lima, Professor(a) do Magistério Superior**, em 15/03/2024, às 15:56, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Daniel Costa Ramos, Professor(a) do Magistério Superior**, em 15/03/2024, às 15:56, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



Documento assinado eletronicamente por **Danielli Araújo Lima, Usuário Externo**, em 15/03/2024, às 15:58, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do [Decreto nº 8.539, de 8 de outubro de 2015](#).



A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5172841** e o código CRC **25569DA5**.

LAURA RIBEIRO

**SIMULTANEOUS LOCALIZATION AND MAPPING TECHNIQUES USING A
NANO QUADCOPTER**

Dissertation presented to the Faculty of Electrical Engineering at the Federal University of Uberlândia as a partial requirement for obtaining the **Master of Science** degree.

Concentration Area: Control and Automation

Examination Board:

Prof. Dr. Gabriela Vieira Lima (Advisor) – UFU
Prof. Dr. Aniel Silva de Morais (Co-advisor) – UFU
Prof. Dr. Daniel Costa Ramos – UFU
Prof. Dr. Danielli Araújo Lima – IFTM

Uberlândia, Brazil

2024

*Aos seres que me amparam e
acompanham-me nessa jornada
contínua, a existência.*

ACKNOWLEDGMENTS

Forgive me, reader. This section will be in Portuguese, my native language. Thank you for your sympathy.

-

Agradeço à engenharia divina e aos seres que diretamente ou indiretamente contribuíram com a minha jornada, até o presente momento, e que me auxiliam continuamente na minha evolução.

Um agradecimento especial à minha orientadora, uma inspiração pessoal, Prof. Gabriela pela paciência, confiança e orientação. Principalmente, nas etapas incertas desse percurso árduo que é a pesquisa e a pós-graduação, mas ainda assim, recompensador.

Agradeço ao Prof. Aniel, meu coorientador, por todas as oportunidades, pelo auxílio, apoio e suporte, incentivando a permanência na pesquisa, há tanto tempo em minha caminhada.

Um agradecimento aos demais professores que de certa maneira, contribuíram com o meu progresso nessa jornada na ciência. Em especial, ao Prof. Éder, que sempre encontrava tempo para direcionar palavras de apoio.

Agradeço à minha mãe e ao meu pai, pelo amparo e esperança nos dias difíceis. Agradeço aos familiares e amigos. Ao meu companheiro, também cientista, por toda motivação. Obrigada a todos pelo carinho, risos e outros tantos sentimentos bons.

Por fim, mas não menos importante, agradeço ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal de Uberlândia, pela oportunidade, à CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) e todos os trabalhadores que compõem a universidade, pela atenção e suporte em mais uma etapa.

ABSTRACT

Simultaneous Localization and Mapping (SLAM) is a critical challenge in autonomous mobile robotics. One of the biggest challenges in engineering is to create solutions that are applicable in the real world, dealing with uncertainties, forces, and possibilities. This research navigates these seas. Therefore, this project aims to implement a system using a nano quadcopter with autonomous flight and laser sensors capable of simultaneous localization and mapping. The research aspires to present diverse techniques for solving SLAM, covering methods for reconstructing 2D and 3D maps using an open-source system with Python programming language. To this end, employing materials developed by the Bitcraze company, the system is composed of the Crazyflie 2.1 nano quadcopter and two expansion decks to enhance the drone's capabilities, Multi-ranger Deck (measurement) and the Flow Deck v2 (pose). Furthermore, it was developed a 3D map of the environment using the point cloud, and a 2D map with the occupancy grid, using distinct techniques. For the development of the methods, the 3D map and 2D map algorithms, the libraries used are Open3D, VisPy, and Matplotlib. Accomplishing the results through experiments conducted in physical environments, with diverse scenarios, four of which applied to the two strategies of flights, Scenario 1 to 4, and two of which were individual cases, Levels, and Real-Time Visualization. All the methods and solutions developed operate satisfactorily and run efficiently, with their respective characteristics and performances.

Keywords. SLAM, Nano Quadcopter, LiDAR sensor, Point Cloud, Occupancy Grid, Crazyflie.

RESUMO

A Localização e Mapeamento Simultâneos (SLAM) é um desafio crítico na robótica móvel autônoma. Um dos maiores desafios da engenharia é criar soluções que sejam aplicáveis no mundo real, lidando com incertezas, forças e possibilidades. Assim, a referente pesquisa navega por esses mares. Dessa forma, este projeto visa a implementação de um sistema utilizando um nano quadricóptero com voo autônomo e sensores laser capazes de localização e mapeamento simultâneos. Além disso, uma das ambições da pesquisa é apresentar diversas técnicas para métodos de reconstrução de mapas 2D e 3D, usando um sistema open-source (software livre), escrito na linguagem de programação Python. Para esse fim, empregando materiais desenvolvidos pela empresa Bitcraze, o sistema é composto pelo Cazyflie 2.1, nano quadricóptero, e dois sensores para aprimorar a capacidade do drone, sendo o Multi-ranger (para medidas), e o Flow-Deck v2 (para a pose). Ademais, foram desenvolvidos mapas dos ambientes, sendo que o mapa 3D utiliza a nuvem de pontos e o mapa 2D com a grade de ocupação, usando técnicas distintas. Ainda, para o desenvolvimento dos métodos, os algoritmos dos mapas 3D e 2D, as bibliotecas que foram utilizadas são a Open3D, VisPy e Matplotlib. Os resultados foram obtidos por meio de experimentos conduzidos em ambientes físicos, sendo cenários diferentes, onde aplica-se duas estratégias de voos em quatro deles, Cenários 1 ao 4, e dois sendo casos individuais, Níveis e Visualização em Tempo Real. Todos os métodos e as soluções desenvolvidas operam de maneira satisfatória e eficiente, com suas respectivas características e desempenho.

Palavras-chaves. *SLAM, Nano Quadricóptero, Sensor LiDAR, Nuvem de Pontos, Grade de Ocupação, Crazyflie.*

LIST OF FIGURES

Figure 1.1 - General System for a Autonomous Robot.	2
Figure 1.2 – Examples of Nano Aerial Vehicles (NAVs).	4
Figure 1.3 – Project System Representation.	5
Figure 2.1 – The Fundamental Challenges of Robots.	8
Figure 2.2 – The Location and Observation of the Robot.	10
Figure 3.1 - Schematic Diagram of the Nano Quadcopter.	19
Figure 3.2 – Nano Quadcopter with Multi-ranger and Flow Deck v2.	20
Figure 3.3 – Mechanical Design of the Multi-ranger Deck.	22
Figure 3.4 – Measurement Error (mm) in Stationary Test.	24
Figure 3.5 – Mechanical Design of the Flow Deck.	25
Figure 3.6 – Process of the EKF using Flow Deck v2.	26
Figure 3.7 – Process of the EKF.	27
Figure 3.8 – Graphical Mapping Model Occupancy Grid.	32
Figure 3.9 – General Methodology Process.	40
Figure 4.1 – Architecture of the Scenario 1.	44
Figure 4.2 – Photo of the Scenario 1.	45
Figure 4.3 – Open3D Point Cloud in Scenario 1 Using Manual Path.	46
Figure 4.4 – Open3D Surface in Scenario 1 Using Manual Path.	46
Figure 4.5 – VisPy Point Cloud in Scenario 1 Using Manual Path.	47
Figure 4.6 – Occupancy Grid in Scenario 1 Using Manual Path.	47
Figure 4.7 – Open3D Point Cloud in Scenario 1 Using Wall Following.	48
Figure 4.8 – Open3D Surface in Scenario 1 Using Wall Following.	48
Figure 4.9 – VisPy Point Cloud in Scenario 1 Using Wall Following.	49
Figure 4.10 – Occupancy Grid in Scenario 1 Using Wall Following.	49

Figure 4.11 – Architecture of the Scenario 2.....	50
Figure 4.12 – Photo of the Scenario 2.	51
Figure 4.13 – Open3D Point Cloud in Scenario 2 Using Manual Path.....	52
Figure 4.14 – Open3D Surface in Scenario 2 Using Manual Path.....	53
Figure 4.15 – VisPy Point Cloud in Scenario 2 Using Manual Path.....	53
Figure 4.16 – Occupancy Grid in Scenario 2 Using Manual Path.	54
Figure 4.17 – Open3D Point Cloud in Scenario 2 Using Wall Following.....	54
Figure 4.18 – Open3D Surface in Scenario 2 Using Wall Following.	55
Figure 4.19 – VisPy Point Cloud in Scenario 2 Using Wall Following.....	55
Figure 4.20 – Occupancy Grid in Scenario 2 Using Wall Following.	56
Figure 4.21 – Architecture of the Scenario 3.....	57
Figure 4.22 – Photo of the Scenario 3.	57
Figure 4.23 – Open3D Point Cloud in Scenario 3 Using Manual Path.....	58
Figure 4.24 – Open3D Surface in Scenario 3 Using Manual Path.....	59
Figure 4.25 – VisPy Point Cloud in Scenario 3 Using Manual Path.....	59
Figure 4.26 – Occupancy Grid in Scenario 3 Using Manual Path.	60
Figure 4.27 – Open3D Point Cloud in Scenario 3 Using Wall Following.....	60
Figure 4.28 – Open3D Surface in Scenario 3 Using Wall Following.	61
Figure 4.29 – VisPy Point Cloud in Scenario 3 Using Wall Following.....	61
Figure 4.30 – Occupancy Grid in Scenario 3 Using Wall Following.	62
Figure 4.31 – Architecture of the Scenario 4.....	63
Figure 4.32 – Photo of the Scenario 4.	63
Figure 4.33 – Open3D Point Cloud in Scenario 4 Using Manual Path.....	65
Figure 4.34 – Open3D Surface in Scenario 4 Using Manual Path.....	65
Figure 4.35 – VisPy Point Cloud in Scenario 4 Using Manual Path.....	66
Figure 4.36 – Occupancy Grid in Scenario 4 Using Manual Path.	66

Figure 4.37 – Open3D Point Cloud in Scenario 4 Using Wall Following.....	67
Figure 4.38 – Open3D Surface in Scenario 4 Using Wall Following.....	67
Figure 4.39 – VisPy Point Cloud in Scenario 4 Using Wall Following.....	68
Figure 4.40 – Occupancy Grid in Scenario 4 Using Wall Following.....	68
Figure 4.41 – Photo of the Levels Scenario.....	69
Figure 4.42 – Open3D Surface in Levels Scenario.....	70
Figure 4.43 – Open3D Surface in Levels Scenario.....	70
Figure 4.44 – Real-Time Visualization Scenario.....	71

LIST OF TABLES

Table 3.1 – Percentage Errors for the Sensors.....	24
Table 3.2 – Mean and Standard Deviation Values for the Sensors Measurements.....	24
Table 3.3 – Algorithm 1.	28
Table 3.4 – Algorithm 2.	34
Table 3.5 – Algorithm 3.	35
Table 3.6 – Algorithm 4.	36
Table 3.7 – Comparison of Flight Strategies.....	40
Table 3.8 – Overview of the Methods.	41
Table 4.1 – Summary of the Experiments.	43
Table 4.2 – General Definitions for Scenario 1 to 4.....	43
Table 4.3 – Color Legend (Open3D).....	44
Table 4.4 – Scenario 1 Trajectory MP.....	45
Table 4.5 – Scenario 2 Trajectory MP.....	51
Table 4.6 – Scenario 3 Trajectory MP.....	58
Table 4.7 – Scenario 4 Trajectory MP.....	64
Table 4.8 – Parameters of the Levels Scenario.	70
Table 4.9 – Number of Scans MP and WF (2D Map).....	72

LIST OF ABBREVIATIONS

AGR	<i>Autonomous Ground Robots</i>
API	<i>Application Programming Interface</i>
AUV	<i>Autonomous Underwater Vehicle</i>
CML	<i>Concurrent Mapping and Localization</i>
CNN	<i>Convolutional Neural Network</i>
DoF	<i>Degrees of Freedom</i>
EKF	<i>Extended Kalman Filter</i>
GPS	<i>Global Positioning System</i>
GUI	<i>Graphical User Interface</i>
IMU	<i>Inertial Measurement Unity</i>
KF	<i>Kalman Filter</i>
LiDAR	<i>Light Detection and Ranging</i>
LSD	<i>Large-Scale Direct</i>
LTVS	<i>Linear Time-Varying Systems</i>
MAV	<i>Micro Aerial Vehicles</i>
MMS	<i>Mobile Mapping System</i>
NAV	<i>Nano Aerial Vehicles</i>
NTVS	<i>Nonlinear Time-Varying Systems</i>
PF	<i>Particle Filters</i>
RGB	<i>Red Green Blue</i>
ROS	<i>Robot Operating System</i>
SA	<i>Situational Awareness</i>
SAR	<i>Search and Rescue</i>

SLAM	<i>Simultaneous Localization and Mapping</i>
SMC	<i>Sequential Monte Carlo</i>
TLS	<i>Terrestrial Laser Scanner</i>
ToF	<i>Time of Flight</i>
UAV	<i>Unmanned Aerial Vehicle</i>
UGV	<i>Unmanned Ground Vehicles</i>
UMS	<i>Underactuated Mechanical System</i>

CONTENTS

1. INTRODUCTION	01
1.1. Justification	03
1.2. Objectives	06
<i>1.2.1. Specific Objectives</i>	06
1.3. Chapter Organization	06
2. SIMULTANEOUS LOCALIZATION AND MAPPING	08
2.1. The Problem	09
<i>2.1.1. Approaches</i>	11
2.2. SLAM and UAVs	14
2.3. Considerations	16
3. MATERIALS AND METHODS	18
3.1. Crazyflie 2.1	18
<i>3.1.1. Dynamic Modeling</i>	20
3.2. Multi-ranger Deck	21
<i>3.2.1. Error Measurement</i>	22
3.3. Flow Deck v2	25
<i>3.3.1. Understanding the Measurement</i>	26
3.4. Algorithms – 3D Map	27
<i>3.4.1. Visualization Approachs</i>	29
3.5. Algorithms – 2D Map	30
<i>3.5.1. Bresenham Algorithm</i>	35
3.6. Flight Strategies	37
<i>3.6.1. Manual Path</i>	37
<i>3.6.2. Wall Following</i>	38
3.7. Considerations	39
4. RESULTS AND DISCUSSION	42
4.1. Definitions	42
4.2. Scenario 1	44

4.2.1. <i>Manual Path – Scenario 1</i>	45
4.2.2. <i>Wall Following – Scenario 1</i>	48
4.3. Scenario 2	50
4.3.1. <i>Manual Path – Scenario 2</i>	51
4.3.2. <i>Wall Following – Scenario 2</i>	54
4.4. Scenario 3	56
4.4.1. <i>Manual Path – Scenario 3</i>	57
4.4.2. <i>Wall Following – Scenario 3</i>	60
4.5. Scenario 4	62
4.5.1. <i>Manual Path – Scenario 4</i>	64
4.5.2. <i>Wall Following – Scenario 4</i>	66
4.6. Individual Cases	69
4.6.1. <i>Levels</i>	69
4.6.2. <i>Real-Time Visualization</i>	71
4.7. Considerations	72
5. CONCLUSIONS	75
5.1. Future Work	76
BIBLIOGRAPHY	78

Chapter 1

INTRODUCTION

Technology is constantly progressing, and its direct effect on various areas of society is remarkable. Science advances proceed closely with technology. Scientists raise questions and create solutions. Scientists come across questions that involve all areas of humanity. From questions that begin with "What is the meaning of life?" to "What will be the future of autonomous robotics, and how much effect will it have on society? On the daily lives of human beings and life?".

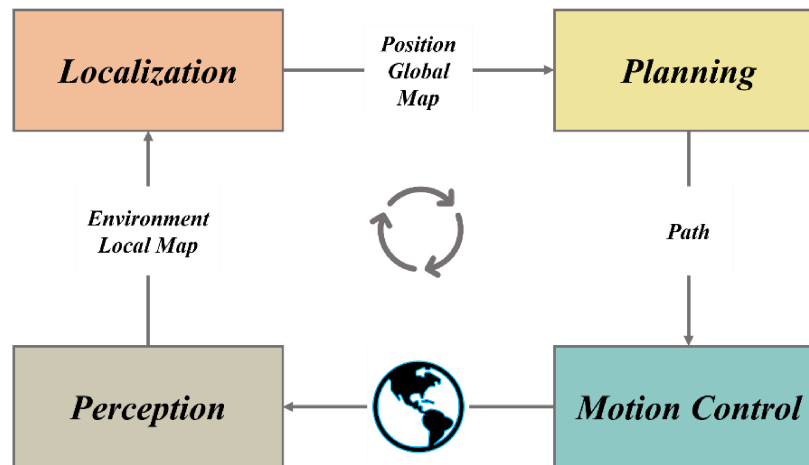
Unfortunately, this work will not be able to answer these questions. However, with the research proposed, some questions surrounding the area of autonomous robots will have a new contribution and new perspectives. The principal questions are "Where am I [robot]? Where is my [robot] position?" and "How is the world around me [robot]?". The work aims to answer these questions, motivated by the proposed solution to the problem.

According to the authors Siegwart, Nourbakhsh and Scaramuzza (2011), the concept of a successful and complete mobile robot has a design involving an interdisciplinary field, and it has to comprehend mechanism and dynamics, control and probability theory, computer vision and algorithms, and the artificial intelligence.

Mobile robots have many capabilities and applications, with the autonomous capability being the most important. Four fundamental fields are part of this ability: locomotion, perception, cognition, and navigation. In addition, examples of these mobile robots include home robots, transportation robots, aerial, underwater, polar, humanoid, pets, unmanned robots, and others (Rubio; Valero; Llopis-Albert, 2019). Figure 1.1 illustrates a general system with the relevant knowledge for an autonomous robot.

Autonomous robots are progressing exponentially, with various uses in different areas. Therefore, it is possible to notice that aerial robots, in this case, aerial vehicles, also have an essential role in this evolution. One of these vehicles is UAVs (Unmanned Aerial Vehicles), and, not different, the applications with UAVs have grown significantly.

Figure 1.1 - General System for a Autonomous Robot.



Source: The author. Adapted from Siegwart, Nourbakhsh and Scaramuzza (2011).

UAVs, also known as drones, are now employed across numerous fields. From search and rescue, agriculture, construction, and disaster management, revolutionizing industries, and expanding possibilities, UAVs have taken advantage of and found utility in numerous domains (Balestrieri *et al.*, 2021). As the name explains, UAVs can fly without the presence of an individual, and they offer a safer alternative for operating in indoor spaces that might be inaccessible to human operators due to safety concerns, such as hazardous locations. By eliminating the need for direct human presence in the target area, drones can provide safety improvement and enable operations in confined indoor environments (Karam, S. *et al.*, 2022).

Among UAVs, the quadcopter is one of the configurations that stands out compared to conventional vehicles such as helicopters. Belonging to the class of these vehicles, it has a system with four sets of propellers and rotating wings, and it is not necessary to use a tail rotor (Lima, 2019).

In addition, the quadcopter system has six degrees of freedom (6-DoF), three of which are related to the three-dimensional position of the center of gravity (x, y, z) and three to attitude (ϕ, θ, ψ), and has only four control variables related to the rotational speeds of the actuators (Souza, 2022), also characterizing it as a UMS, an underactuated mechanical system. Beyond the complexity of the UMS, quadcopters are nonlinear time-varying systems (NTVS) that can be affected by uncertainties in the parameters surrounding them, such as aerodynamic disturbances and sensory noise (Lima, 2019; Raffo, 2011).

Furthermore, considering the development of UAVs, Micro Aerial Vehicles (MAVs) have been highlighted, with the creation of even smaller vehicles, such as Nano Aerial Vehicles (NAVs). The capability of operations in indoor environments is fascinating, but they still face

several challenges, such as the need for positioning systems and limitations on the types of sensors attached (López *et al.*, 2017). Regardless, the advantage of these NAVs being able to be replaced or repaired more quickly and easily stands out.

As if autonomously flying was not complex enough, these aerial robots are crossing an unknown environment, and knowing the poses is challenging for autonomous flight. The drone must be capable of identifying and avoiding obstacles while mapping and sensing the environment. In this way, there is often insufficient information, regarding the environment, which can cause significant challenges for these robots.

Simultaneous Localization and Mapping (SLAM) is a critical challenge in autonomous mobile robotics. It involves the complex task of reconstructing unknown environments while concurrently establishing the robot's own position within the generated map (Lai, 2022). Considering Takleh (2018), the main question is whether a robot could have the ability to map its surroundings and simultaneously determine its location in that environment.

Therefore, the development of this research aims to contribute to a solution to this challenge, using an innovative system with sensors attached to a nano quadcopter, with autonomous flights and decision-making, and provide a new approach to the questions surrounding the localization and mapping of autonomous robots.

1.1. Justification

One of the biggest challenges in engineering is to create solutions that are applicable in the real world, dealing with uncertainties, forces, and possibilities. This research navigates these seas.

NAVs can operate in various applications, such as inspections, observations, and search and rescue operations (McGuire, 2019). Due to its various characteristics, such as its low mass and small size, being supported by four small rotors, and despite its simpler mechanics, the nano quadcopter still maintains its total capacity. The nano quadcopter is an excellent alternative for indoor spaces: it can fly close to humans (due to its size) and does not pollute the environment (due to electric motors) (Lima, 2019; McGuire, 2019). In Figure 1.2 can be seen two different examples of NAVs.

Figure 1.2 – Examples of Nano Aerial Vehicles (NAVs).



Source: From McGuire (2019).

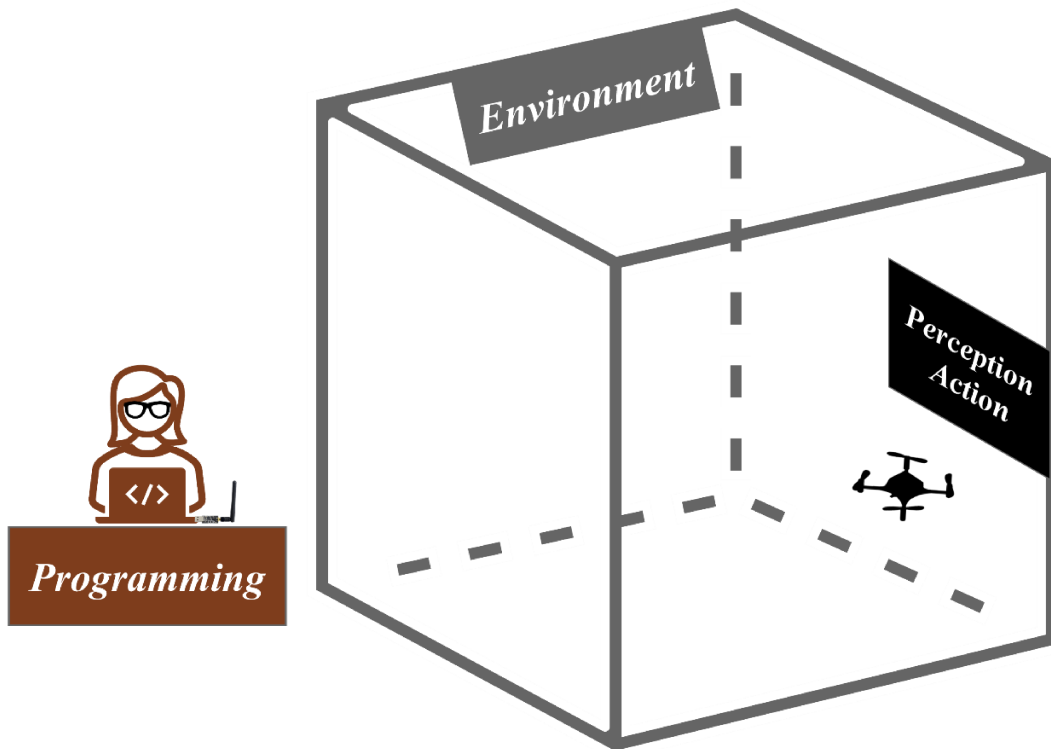
On these terms, the Crazyflie 2.1 is known for its affordability, which offers expansion capabilities through extension ports (Bitcraze, 2022a). The multi-ranger deck, a laser sensor, is one of the supported hardware extensions for the Crazyflie drone. The combination of autonomous flight with the ability to map the environment allows the drone to operate effectively in dynamic, unknown, or hard-to-reach environments.

SLAM plays an indispensable role across various robotics applications and is a promising solution for environment mapping (Xia *et al.*, 2022). Indoor environments are one of the most significant application scenarios of SLAM. Solving this type of problem yields significant benefits for navigation-related tasks, including path planning, robot re-localization, and human decision-making (Zhou *et al.*, 2022).

By addressing the challenges of working in an unmapped environment, this project represents a new step in the development of safe and effective drone-based solutions. Techniques are employed to process and visualize the data, enabling surface reconstruction.

Regarding the fundamental paradigms of robotics, the three related are sense, plan, and act (Murphy, 2000). Figure 1.3 shows the project system representation, where the robot interacts with these three categories. It illustrates that the nano quadcopter, with its sensors (sense), will interact (plan) through perception and action (act) with the environment. The human icon of a woman is a symbolic representation of the author of this dissertation with programming.

Figure 1.3 – Project System Representation.



Source: The author.

The work is divided into three significant parts to achieve the goal of autonomous flight, space localization, and environment mapping. In an autonomous flight, the code allows the drone to recognize its location in three-dimensional space and map the territory with points detected by the laser sensor (Multi-ranger Deck). To enable autonomous movement, the other sensor used was Flow Deck. All the items used are from *Bitcraze* company, well-known and utilized in numerous research by the master's and PhD students of the Automation, Electronic Systems, and Control Laboratory (LASEC), a Federal University of Uberlândia's laboratory.

Moreover, this project is motivated by the opportunity to contribute and add to studies in the diverse field of robotic science. The work addresses new techniques for simultaneous mapping and localization in indoor environments, being able to generate more didactic and innovative solutions using a nano quadcopter. Hopefully, the research will add and promote discussions beyond academia, given that SLAM projects are present in countless areas and applications.

1.2. Objectives

This project aims to implement a system using a nano quadcopter with autonomous flight and laser sensors capable of simultaneous localization and mapping. The system will be composed of Crazyflie 2.1 and two different decks, Multi-ranger Deck, and Flow Deck v2. In addition, the work aspires to present diverse techniques for solving SLAM, covering methods for reconstructing 2D and 3D maps, using an open-source system, with Python programming language. Moreover, this work's ambition is to help disseminate research and concepts, and the methodology that will be present offers a new perspective for creating a practical nano quadcopter system, developed in a computational and scientific language, using open-source materials and available for general access.

1.2.1. Specific Objectives

The specific objectives are described as follows.

- Implementation of the nano quadcopter "SLAM approach" with two different decks.
- Formulation of a script for communication between a central control node and the drone.
- Explore different techniques for plotting the maps generated by the nano quadcopter - considering point cloud maps.
- Implementing a probabilistic model for plotting a 2D map.
- Development of experimental tests for static environments and implementing experimental tests in real-time visualization.

1.3. Chapter Organization

The structure of this dissertation consists of five chapters, which are listed as follows.

This first chapter, Chapter 1 – Introduction, presents an introduction of the work proposed. In addition, the chapter highlights the motivations and justifications for choosing the topic, the general and specific objectives, and how the document is structured.

Chapter 2 – Simultaneous Localization and Mapping, as the name illustrates, highlights the challenge of this work, the problem. Based on the problem, the chapter also presents the solutions using UAVs and significant related work, finishing with considerations.

Chapter 3 - Materials and Methods, is a chapter composed of two main parts: materials and methods. The first part describes all the components, and the second, the approaches and techniques used and involved in the proposed solution. Furthermore, the chapter lists the software tools that allowed the project development.

Results and Discussion is the most important chapter. Chapter 4 explains all the results achieved, considering all the proposed techniques and scenarios. There is also a discussion and analysis of the results.

Finally, there is the Chapter 5. Chapter 5 - Conclusions, presents the final considerations and the ending. There are also details for future work.

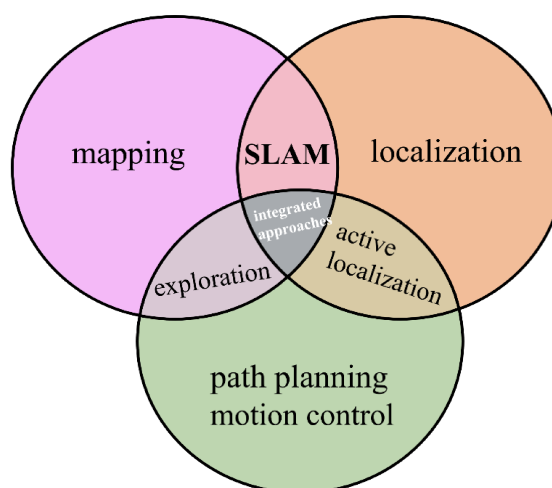
SIMULTANEOUS LOCALIZATION AND MAPPING

The second chapter highlights the challenge of the Simultaneous Localization and Mapping problem. Furthermore, it presents a historical narrative of the problem, the solution advances, and finally, the relevant applications using UAVs.

There are three fundamental challenges that an autonomous robot is supposed to achieve for autonomous behavior: mapping, localization, and path planning. According to Cyrill Stachniss (2006), the mapping problem is building the representation given the information collected by the robot's sensors, as opposed to localization, which is the problem of the robot's estimated pose within that map. Meanwhile, path planning would be to have the best trajectory for the robot to move along, leaving one point and arriving at the chosen destination.

These three challenges are connected, and the relationship between them can be seen in Figure 2.1. The SLAM is a combination of the mapping problem and the localization problem.

Figure 2.1 – The Fundamental Challenges of Robots.



Source: The author. Adapted from Stachniss (2009).

Consequently, a robot's Situational Awareness (SA) system is capable of continuously acquiring new observations, understanding the elements, performing complex reasoning,

project the state of the world in order to make decisions and execute actions that would let it accomplish its purposes (Bavle *et al.*, 2023).

2.1. The Problem

During a conference in 1986, the first work published represents the beginning of the probabilistic methods applied to robotics problems (Rueda Ramos, 2012). They joined at events, talked, and discussed these possibilities, and as a result, the work by the author's Smith and Cheesman (1986) and Durrant-Whyte (1988) established a statistical basis that describes the relationship between geometric uncertainty and points/landmarks (Nemra, 2010).

Significant steps in the history of SLAM began in 1986 with discussions of the problem, sequentially, moving on to the years between 1990 and 1995 with Kalman Filter-based approaches and finally, in 1995, at the International Symposium on Robotics Research, the result of the convergence and the creation of the acronym SLAM were presented for the first time by (Durrant-Whyte; Rye; Nebot, 1996). The work focuses on improving computational efficiency and solving data association problems. Consequently, over the years, there has been an interest in the area, which has resulted in conferences partly or entirely dedicated to SLAM (Nemra, 2010; Rueda Ramos, 2012).

With the problem defined and interest growing, different approaches, applications, techniques, and methodologies began to surface. One example is the work developed by Moravec and Elves (1985), which deals with the representation of occupancy grids for probabilistic maps. Soon after, the authors Thrun, Burgard and Fox (1998) published the first probabilistic approach to Concurrent Mapping and Localization (CML) for mobile robots. To this day, these maps are famous in mapping solutions. So much so that the proposal in this paper also includes a SLAM technique using the occupancy grid.

SLAM algorithms continue to be a promising and exciting area of research, as mapping and navigation are the keys to allowing robots to interact autonomously with the real world (Lai, 2022). SLAM technology involves several areas, such as sensing, perception, localization, mathematics, logic, and the kinematics of the robot mechanism, among others (Dai; Wu; Wang, 2023).

Following these observations, considering that mobile robots are robots capable of moving around their environment, there are three fundamental systems: locomotion, control, and sensory. And the classification of these mobile robots varies according to anatomy or purpose, type of control, functionality, and movement (Ramos, 2012).

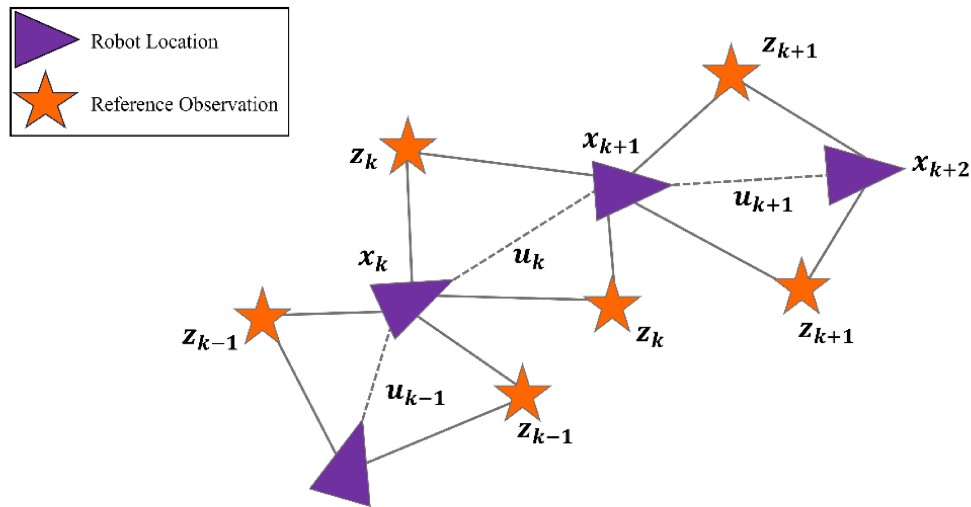
Basically, the robot will have three main structures interacting with the environment: actuators for action, sensors for perception, and logic for decision-making. The sensors are essential to solving the SLAM problem since they can provide the position and the information the robot perceives in the environment in order to represent it. And with this acquisition of information, the more knowledge it has, the better the decision-making will be.

By the definitions mentioned above, given a SLAM problem and map (m) of the environment:

- The control inputs are: $u_{1:T} = \{u_1, u_2, \dots, u_T\}$,
- The observations and/or readings from the robot: $z_{1:T} = \{z_1, z_2, \dots, z_T\}$,
- The path: $x_{0:T} = \{x_0, x_1, x_2, \dots, x_T\}$.

Figure 2.2 illustrates the problem of locating the robot, the trajectory, and its observations.

Figure 2.2 – The Location and Observation of the Robot.



Source: The author. Adapted from (Yuan; Wang; Xie, 2021).

Based on this, there are two general implementations: Full SLAM and Online SLAM (Thrun; Burgard; Fox, 2005). Online SLAM estimates only the current vehicle pose while generating the map and updating it with recent measurements. On the other hand, in the case of the Full SLAM implementation, the entire trajectory of the vehicle is updated together with the entire map (Zheng *et al.*, 2023). In other words, Full SLAM has an estimate of the complete path traveled as if it were keeping a history, unlike Online SLAM, which only has an estimate of the most recent/current position.

Furthermore, it is important to mention that in the case of the autonomous vehicle, as they move from one position to the next, it will continuously take measurements of the environment, and these measurements can be used in the Theorem of Bayes to estimate the probability of the next position of the vehicle concerning the map. Also, estimating the next pose probability estimation in Bayes' Theorem needs to consider previous measurements and control inputs (Takleh *et al.*, 2018).

Thus, considering the measurements, it is necessary to understand the sensors. Sensors can be classified into two types: proprioceptive or exteroceptive and passive or active. Proprioceptive sensors measure values internal to the system (robot), such as motor speed, orientation, and battery, whereas exteroceptive sensors measure values external to the system (environment), such as distance to objects and light intensity. Passive sensors are based on energy into the sensor from the environment, i.e., they are dependent, such as cameras, microphones, and compasses. Active sensors, on the other hand, emit their own energy and are also able to measure this result, generally performing better, such as lasers, ultrasonics, and radars (Siegwart; Nourbakhsh; Scaramuzza, 2011). It is important to remember that sensors have uncertainties, that is, errors associated with the measurements.

Then, once the perception model is known, there is the location, which indicates the planning or the navigation on that map. The map is the spatial representation of the environment based on the readings and perception of different sensors. The mapping process is the construction or building of a representation (map) of the environment with the information collected by the sensors. There are many types of representation. For example, maps can be continuous and metric, discrete and metric, topological or semantic.

In consequence, if there are robots with distinctive characteristics, models and representations, functionalities, and functions, robots with different systems, structures, and sets of sensors used, there are different algorithms for the map representation. With different algorithms existing, there are numerous approaches and research for solving the SLAM problem in the literature. As follows, this chapter and its sections will mention some of these works.

2.1.1. Approaches

This section will cite, and mention papers, works, and similarities as examples of approaches for SLAM.

A variety of approaches for SLAM exist because of diverse factors, for example, spatial representation, the structure and dynamics of the environment, and employed sensors (Sadeghzadeh-Nokhodberiz *et al.*, 2021). Still, robot sensors directly impact the algorithm in SLAM, and over the years, researchers have focused on 2D laser range finders, 3D LiDAR's, and vision sensors (López *et al.*, 2017).

Based on Lai (2022), the evolution of various SLAM approaches has been proposed, with the development of different algorithms based on the traditional ones, such as FastSLAM, EKF, Visual SLAM (V-SLAM), Large-Scale Direct (LSD) SLAM, ORB-SLAM (Monocular, Stereo, and RGB-D cameras), 2D-LiDAR SLAM, GRAPH-SLAM, PF SLAM, Direct Sparse Mapping. In accordance with Gaia *et al.* (2023), that in the work discussed survey studies on SLAM focus on the areas of theoretical issues, multi-robot SLAM, LiDAR (Light Detection and Ranging) and camera-based SLAM, autonomous driving, dynamic SLAM, and Deep Learning for SLAM.

According to Aulinas *et al.* (2008) the Kalman Filters (KF) and Particle Filters (PF) probabilistic approaches have been prevalent in SLAM since the '90s. Kalman Filters are Bayes Filters that represent posteriors using Gaussians and rely on the assumption that the state transition and the measurement functions are linear with added Gaussian noise. The Extended Kalman Filter (EKF) is a variation of the KF, and its related accommodates the nonlinearities approximating the robot motion model using linear functions from the environment. On the other hand, Particle Filters, or the method Sequential Monte Carlo (SMC), is a recursive Bayesian filter implemented in Monte Carlo simulations. PF represents the distribution by a set of samples removed from this distribution, able to take highly nonlinear sensors and non-Gaussian noise.

Conforming to the work of Thrun *et al.* (2002), FastSLAM is an algorithm that integrates particle filters and Extended Kalman Filters, exposing a structural property where correlations in the uncertainty among different map features arise only through robot pose uncertainty. The algorithm uses a particle filter to sample the robot's trajectories and an extended Kalman filter to represent the maps acquired by the robot. Soon after, the authors Montemerlo *et al.* (2003) published a new paper with version 2.0 of FastSLAM. In this new version, when proposing a new robot pose, the distribution relies not only on the motion estimate but also on the most recent sensor measurement.

The LiDAR sensors are promissors in robotics technologies. These sensors can be combined with other distinct sensors, for example, IMU (Inertial Measurement Unit), cameras, GPS (Global Positioning System), sonar, and ToF (Time of Flight) sensors. Moreover, the 2D

LiDAR and 3D LiDAR sensors have a simple difference, one records x and y parameters using a single axis, whereas the other records x and y measurements around the z -axis (Khan *et al.*, 2021).

Considering the work of Huang (2021), Visual SLAM, or V-SLAM, treats the camera as the main sensor and the image as the source of information, while LiDAR-SLAM can carry out its localization and mapping in more diverse environmental conditions. In other words, is a set of SLAM techniques that use only images to map the environment and determine the robot's position.

Examples of recent applications using V-SLAM include the control of humanoid robots, unmanned aerial vehicles, unmanned ground vehicles, autonomous underwater vehicles, and lunar vehicles. As well as augmented reality applications (Covolan; Sementille; Sanches, 2020), which can use a variety of cameras, such as monocular RGB (Red Green Blue), stereo (two RGB cameras), and RGB-D (in addition to RGB images, pixel depth information).

In (Chan; Wu; Fu, 2018), was introduced a novel method fusing two kinds of laser-based SLAM and monocular camera-based SLAM, for laser SLAM and visual SLAM fusion to provide robust localization. The first part of their experiment used a Pioneer 3-DX (a driving robot) equipped with a webcam and laser rangefinder, and the second part of the experiment used Pepper as the robot platform.

It is already known that in the mobile robots and aerial vehicles domain, SLAM can assist in positioning, path planning, autonomous exploration, obstacle avoidance, and navigation. Besides, SLAM technology also can be used in underwater operations, medical treatment, unmanned driving, and other fields (Dai; Wu; Wang, 2023).

Following (Malinverni *et al.*, 2018), a system can deal with the comparison of dense point clouds obtained with different acquisition techniques: Terrestrial Laser Scanner (TLS) and Mobile Mapping System (MMS). It wants to generate a 3D map incorporating LiDAR and odometry (wheel encoders) or inertial navigation systems to determine self-motion to solve the SLAM problem.

Furthermore, this study centers its efforts on developing techniques that utilize LiDAR methodology. This preference stems from the mention of its excellent potential in comparison to alternative sensor approaches.

Beyond this, there are various possibilities for applying SLAM, as mentioned above. One of the possibilities lies in surgery (Scaradozzi; Zingaretti; Ferrari, 2018). Further, it also can take a different approach, as can be seen in the work of Altinpinar *et al.* (2022), who proposed comparing the environmental maps generated through measurements from a laser

scan placed on different parts of a wheelchair transformed into an autonomous wheelchair. Along with Lee *et al.* (2013), who experimented with graph-based SLAM, and vision-based localization using an AUV, an Autonomous Underwater Vehicle.

There is also a possibility of including ROS (Robot Operating System) in applications of SLAM. Another possibility is a cooperative SLAM, and cooperative operations of autonomous ground robots (AGR) and quadrotors are a reality. For example, Unmanned Aerial Vehicles (UAVs) and Unmanned Ground Vehicles (UGVs) are the most commonly used robots for civil infrastructure-related applications (Hu; Assaad, 2023). An example is given by (Lin; Tsai; Tai, 2019), where they propose a cooperative SLAM method that allows one quadrotor to fly more flexibly and efficiently with an AGR in indoor environments.

2.2. SLAM and UAVs

The potential of UAVs and their applications is surprising, and several applications are using UAVs. In this section, examples of approaches for the SLAM problem will be mentioned and cited in papers, works, and similarities.

Numerous autonomous indoor applications require the use of small aerial robots capable of navigating narrow and confined spaces (Bavle *et al.*, 2020). Although many studies have a preference for and may require a relatively large drone, small drones are advantageous for passing through small openings in damaged buildings and providing data for exploration purposes (Karam, S. *et al.*, 2022).

While quadcopters are relatively simpler compared to helicopters, they still face challenges when it comes to autonomous navigation. These challenges arise due to several limitations, including non-linearity, fast dynamics, limited payload capacity, and vibration. The primary objective of an autonomous flying robot is to achieve a desired destination without the need for human intervention (Saeedi *et al.*, 2017).

Diverse types of UAVs in the literature are found, such as fixed-wing, single-rotor, fixed-wing hybrid, and multirotor, with notable applications (Mohsan *et al.*, 2023). The UAVs can have better performance in maneuvering flexibility, high scalability, portability and mobility (Ahmed *et al.*, 2022).

Noting that UAVs have many advantages, including high speed, aerial point of view, and high altitudes, and, for all that, the nano drones demonstrate the potential to be in missions of dangerous indoor exploration. However, there are some limitations to using these vehicles,

such as the battery, vibrations, external forces, lost control, and easy damage (Rueda Ramos, 2012; Zhou *et al.*, 2022).

According to Mohsan *et. al* (2023), for example, the applications can be in disaster management and Search and Rescue (SAR) where the UAV visits unsafe regions for humans. Also, an application is UAVs remotely sensed, used for inspection of water quality, gas and oil yield, famine and geological monitoring, and others. Recently, UAVs have been used in precision agriculture to obtain information through the installed sensors and, of course, in space exploration, offering the potential to be in space missions, such as studying the surface and atmosphere of the planet.

Considering mobility, MAVs can be applicable in missions in confined buildings with multiple floors, where stairwells or clutter prohibit the motion of ground vehicles, for example. Shen, Michael and Kumar (2013) discussed this, where the work presents a system design and methodology that enables autonomous navigation with real-time performance using a micro vehicle with onboard sensors. Including the results of a generated 3D mapping of multiple floors with loop closure.

Focusing on emergency response mapping in dangerous places, Karam *et. al* (2022), used a quadcopter microdrone equipped with six laser rangefinders and an optical sensor for mapping and positioning, designed to map indoor spaces with planar structures through graph optimization. The system is composed of a customized Crazyflie, multi-ranger deck, and flow deck, and the work proposed a mapping system based on a combination of 1D scanners.

One of the possibilities for multirotor aerial robots using LiDAR is in a situation awareness system. Sanchez-Lopez (2019) used deep learning in the design and development of a semantic situation awareness system for multirotor aerial robots based on 2D LIDAR measurements. The algorithm includes LiDAR for measurements, a Convolutional Neural Network (CNN) computing the radius and the position of the individual, and an indirect EKF providing an estimate of the semantic map.

For autonomous drone exploration, the author Von Stumberg *et al.* (2017) proposes a method for an autonomous MAV navigation and exploration with vision-based, LSD-SLAM (Large-Scale Direct Simultaneous Localization and Mapping) and a Parrot Bebop MAV. Considering that LSD-SLAM only determines depth at high gradient pixels, texture-less areas are not directly observed, so the approach considers semi-dense monocular depth measurement based on motion parallax. The semi-dense depth maps reconstructed with LSD-SLAM create the 3D occupancy map.

On the other side, Kuang *et al.* (2020), propose a real-time UAV path planning system, estimating the building scene and details through a SLAM framework for autonomous urban scene reconstruction. The method combines information on reconstructed point clouds and possible coverage areas, predicting the coverage of the scene. It is a successful and effective method, but the shadows and power parts of the buildings impact the results.

Following the cases of quadrotor UAVs in SLAM, the work of Fink *et al.* (2017), proposes two observer designs for Visual Inertial Localization Mapping (VISLAM) that combine measurements from an IMU with the existing monocular VSLAM to estimate position, linear velocity, and accelerometer bias. The observers are based on state transformation and a Linear Time-Varying (LTV) observer design. Compared with EKF, the observers have better performance in experiments, but in simulations, the EKF converges faster.

Therefore, applications using nano quadcopters or nano drones are not entirely recent. As seen in (Niculescu *et al.*, 2023), the authors propose a NanoSLAM, a lightweight and optimized SLAM approach designed to operate on palm-size UAVs, enabling fully onboard mapping. The results demonstrate the mapping capabilities in real-world scenarios with a nano drone (Crazyflie) equipped and a novel commercial RISC-V low-power parallel processor.

The works of Zhou *et al.* (2022) and Jeong *et al.* (2022) also used the Crazyflie nano quadcopter. In Efficient 2D Graph SLAM for Sparse Sensing (Zhou *et al.*, 2022), two graph approaches are incorporated, landmark and pose, presenting a graph-based system approach, incorporating a novel tailored for sparse sensing and an improved loop closure detection algorithm. Meanwhile, in (Jeong *et al.*, 2022), the work presents a 3D structure mapping algorithm with an accumulated point cloud that evaluates the proposed method in various Manhattan world structures and compares the results to the ROS Gmapping algorithm, which uses a 2D laser scanner.

As can be seen in (López *et al.*, 2017), in drone flights sensors must be carefully selected due to their limitations. The potential that nano drones have to explore unknown and complex environments is incredible. Their size and agility make them a safe option for operation close to humans, and their applications are an extensive area (Müller *et al.*, 2023).

2.3. Considerations

This chapter presented a brief theoretical background on topics relevant to this dissertation. It is a fundamental phase since when choosing the approaches for this work, it is also necessary to find similar works for guidance and a certain degree of novelty.

Considering all the papers included in the analyses, it is observable that the approach is differentiated, owing to the conjunct of the sensors utilized and the SLAM techniques designed. These techniques are in the 3D and 2D domain, the 3D being based on point cloud, and the 2D, on the famous probabilistic map, the occupancy grid.

Therefore, this project aims to implement a simultaneous localization and mapping system using a nano quadcopter with autonomous flight and laser sensors, work with data collected from experiments (physical and determined environments) and apply different techniques to the SLAM problem.

MATERIALS AND METHODS

This chapter is composed of two main parts: materials and methods. The first part describes all the components, and the second, the approaches and techniques used and involved in the proposed solution. Furthermore, the chapter lists the software tools that allowed the project development. The materials listed include the nano quadcopter (Crazyflie 2.1), the Multi-ranger, and Flow Deck v2. Additionally, a computer and Crazyradio PA are necessary for operation. To implement the described methods, proficiency in programming, particularly in Python, is essential.

3.1. Crazyflie 2.1

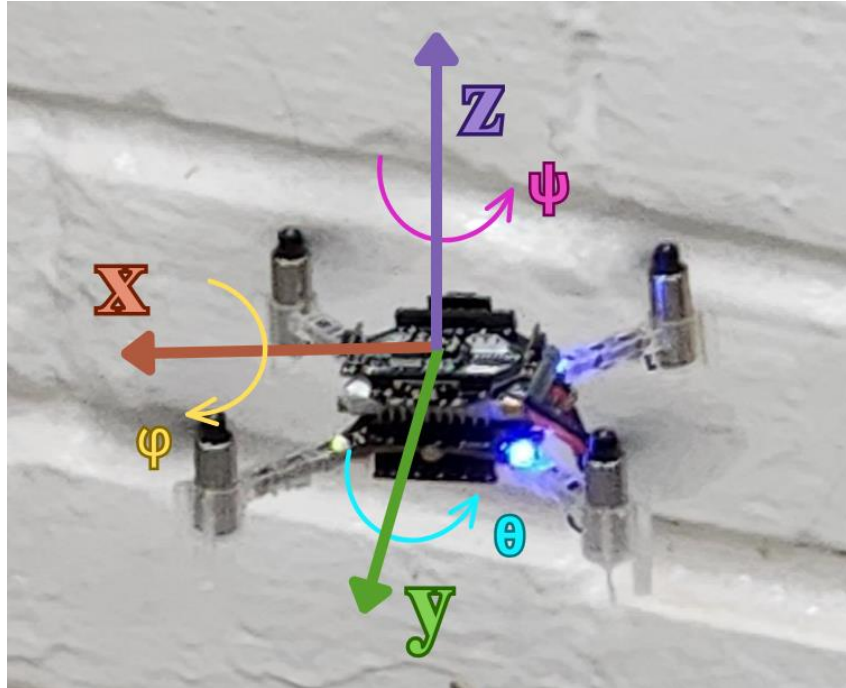
The nano quadcopter, or nano drone, chosen for this work is Crazyflie 2.1, produced by Bitcraze company (Bitcraze, 2022a). Crazyflie 2.1 has several advantages, as can be seen in the previous sections. It costs much less than other nano quadcopters. The support provided by the company is excellent, and it is an open-source and open-hardware platform.

The nano quadrotor's size is ideal for indoor applications, and the low inertia causes just a few parts to break after a crash. It can communicate with a phone, PC, and a USB dongle called Crazyradio (lower latency). The main microcontroller (STM32) is for processing, and the second microcontroller (nRF51) is for wireless communication (Hönig; Ayanian, 2017). The quadrotor has an IMU onboard consisting of three-axis gyroscopes, accelerometers, magnetometers model MPU-9250, and a pressure sensor model LPS25H (Souza *et al.*, 2022). Previously, (Landry, 2015) and (Förster, 2015) discussed mathematical models and system identification of essential parameters, later used by so many other works.

The Crazyflie is a quadcopter with four rotors, without a tail rotor. The quadcopter has six degrees of freedom, where three position coordinates in space (x, y, z) and three orientation angles roll, pitch, and yaw (φ, θ, ψ). The angles roll (φ), and yaw (ψ) are clockwise rotating around the axis. The pitch (θ) angle is counterclockwise rotating around the axis.

Figure 3.1 was captured on one of the flights of the drone and adapted for showing the angles and the coordinates.

Figure 3.1 - Schematic Diagram of the Nano Quadcopter.



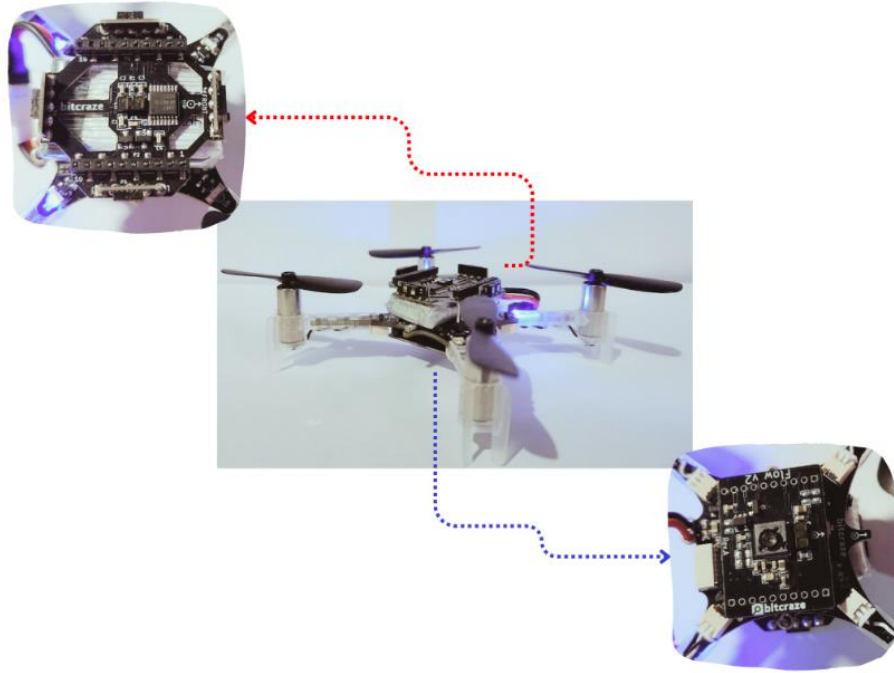
Source: The author.

The nano quadcopter was designed with maximum flexibility, allowing easy customization and modification. In addition to its standard functionality, the aircraft features a versatile expansion interface that supports a range of expansion decks.

These decks can be attached either on the top or bottom of the Crazyflie 2.1, providing enhanced capabilities. Among the available expansion decks, two options were selected for this project: the Multi-ranger Deck and the Flow Deck v2.

Figure 3.2 illustrates a schematic of the final system. The figure shows the nano quadcopter with the two decks attached to it in the center. Note that the multiranger is at the top, and the flow deck is at the bottom, which are the ideal positions for the decks to avoid any kind of influence on their capabilities. The system operates efficiently.

Figure 3.2 – Nano Quadcopter with Multi-ranger and Flow Deck v2.



Source: The author.

3.1.1. Dynamic Modeling

An understanding of the dynamic model of the nano quadcopter is required to implement the algorithms. Specifically, the complete rotation matrix of the system is essential for this endeavor (Lima, 2015). The following rotation matrices represented the dynamic modeling.

Consequently, the first rotation matrix refers to the rotation around the x -axis by the roll angle φ , shown in Equation 3.1.

$$R(x, \varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \varphi & -\sin \varphi \\ 0 & \sin \varphi & \cos \varphi \end{bmatrix} \quad 3.1$$

The matrix in Equation 3.2 represents the rotation matrix around the y -axis and the pitch angle θ .

$$R(y, \theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad 3.2$$

Finally, Equation 3.3 shows the rotation matrix around the z -axis and the yaw angle ψ .

$$R(z, \psi) = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad 3.3$$

The rotation of these angles is delimited by roll angle ($-\pi < \varphi < \pi$), pitch angle ($-\frac{\pi}{2} < \theta < \frac{\pi}{2}$), and yaw angle ($-\pi < \psi < \pi$).

The complete rotation matrix, Equation 3.4, is known as the Direction Cosine Matrix. Defines the coordinate system orientation in relation to the inertial system I.

$$R_I = R(z, \psi)R(y, \theta)R(x, \varphi)$$

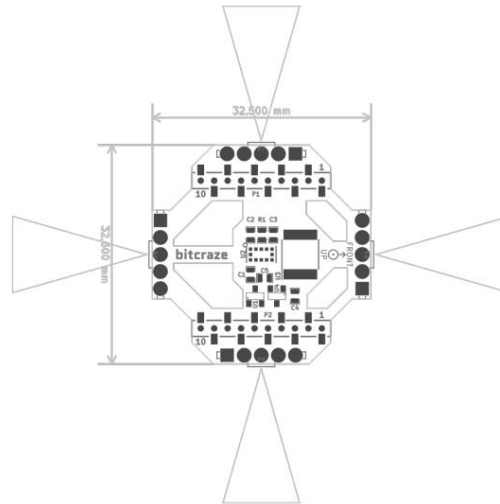
$$R_I = \begin{bmatrix} \cos \psi \cos \theta & \cos \psi \sin \theta \sin \varphi - \sin \psi \cos \varphi & \cos \psi \sin \theta \cos \varphi + \sin \psi \sin \varphi \\ \sin \psi \cos \theta & \sin \psi \sin \theta \sin \varphi + \cos \psi \cos \varphi & \sin \psi \sin \theta \cos \varphi - \cos \psi \sin \varphi \\ -\sin \theta & \cos \theta \sin \varphi & \cos \theta \cos \varphi \end{bmatrix} \quad 3.4$$

3.2. Multi-ranger Deck

The multi-ranger deck is a laser ranging for measuring distances to objects around. So, the deck enables it to detect objects in its surroundings. According to specification, it can measure from four meters up to a few millimeters, varying with light and object surface, and it can measure distance in all five directions, up, left, right, front, and back (Bitcraze, 2022c). The default measurement cone of the VL53L1 is 25 degrees. Figure 3.3 illustrates the mechanical design of the multi-ranger, taken from the deck's datasheet.

The design of the multi-ranger deck is composed of five VL53L1x ToF sensors to measure the distance up to 4 m within a few millimeters (Bitcraze, 2022c). The laser rangefinder is a ToF (Time-of-Flight) sensor, often referred to as a LiDAR (Light Detection and Ranging) sensor, that achieves significant improvements. In other words, LiDAR is a virtual perception device capable of checking exterior surfaces and is a category of ToF sensors (Khan *et al.*, 2021; Siegwart; Nourbakhsh; Scaramuzza, 2011). These type of sensors is indicated for tiny vehicles, and they can sense an object that appears along their line of sight (Coppola *et al.*, 2020).

Figure 3.3 – Mechanical Design of the Multi-ranger Deck.



Source: Multi-ranger Datasheet (2020).

Based on (Siegwart; Nourbakhsh; Scaramuzza, 2011), the Equation 3.5 represents a way to measure the ToF for the light beam, using a pulsed laser and then measuring the elapsed time directly, like an ultrasonic sensor. Where d is the distance of the object causing reflection, the propagation speed of sound c , and t is the time of flight.

$$d = \frac{c \cdot t}{2} \quad 3.5$$

Multi-ranger has been proven to be sufficient in the environment, but with some limitations, for example, delicate objects are hardly seen (McGuire *et al.*, 2019). Considering that, it was necessary to understand the uncertainties and the error measurement (*Section 3.2.1*).

It is highly recommended to pair the Multi-ranger Deck with the Flow Deck to increase its potential. By combining these decks, the Crazyflie is capable of measuring ground movement and distance, allowing it to perceive and react to its environment and detect and avoid obstacles when they are close. Also, allows to navigate and map its surroundings with improved space understanding. Furthermore, this powerful combination opens the possibilities for SLAM algorithms.

3.2.1 Error Measurement

The environment can be challenging for a sensor and introduce errors to the measurements, which might increase over time. So, during the stationary test, a comparison

was made between the measurements obtained from the multi-ranger sensors and the directly measured real-world distances. Also, note that during the test, a white, flat wall was used (although it was not completely flat), with good lighting in the environment.

With the script developed, it is possible to change it internally to point to the correct data files and the physical measurements of the environment. It will read the files and calculate the error in terms of percentiles. Taking over sample data, for example, a percentile p_{90} of 90 (p_{90}) returns a single value (v), which means 90% of the sample data is below this value. Considering that, the percentile is meaningful since it is possible to check the measurement error. For example, if the $p_{90} = v$ [mm] is equal to say that 90% of measurement errors are less than v mm.

A single sample is calculated by subtracting the sensor measurement from the real-world value and using the absolute error value. This process was executed for each measurement and each sensor, resulting in an array of error values. The error values are sorted in ascending order to extract the percentile from the sample (S). Considering the Numpy library and (Hyndman; Fan, 1996), first need to locate the position which contains p_{90} of S . Equation 3.6 is the locator formula, where L_p is the position locator.

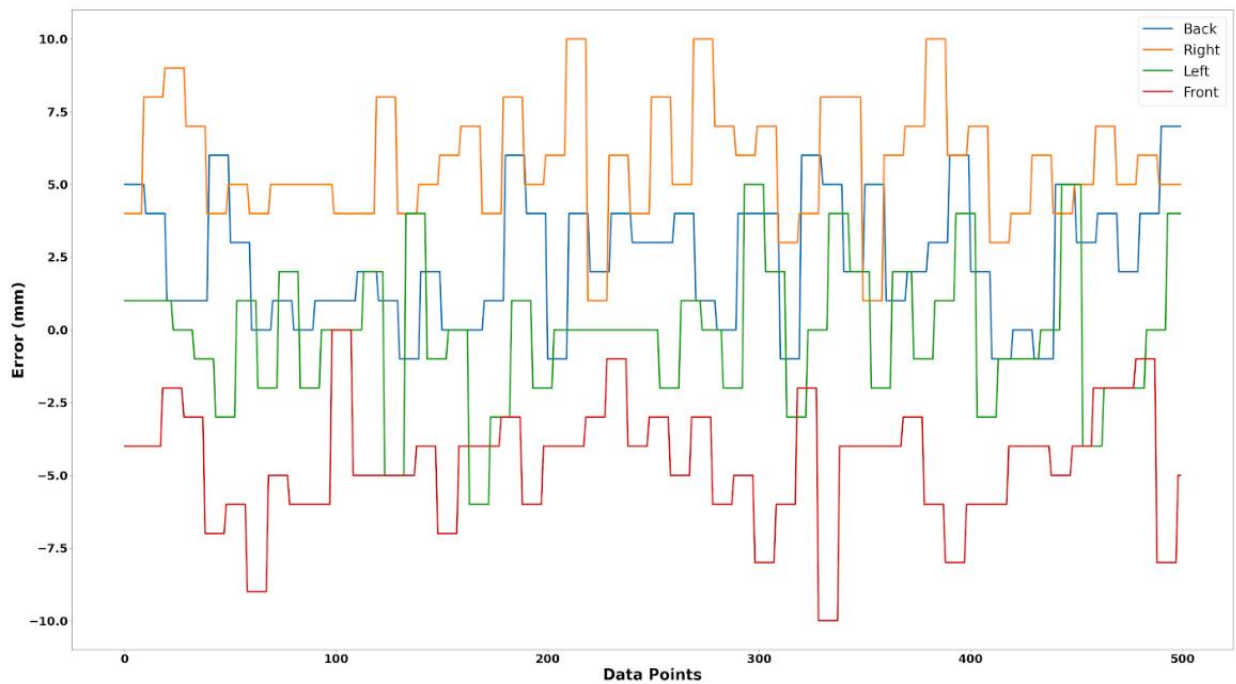
$$L_p = \frac{p}{100} * (|S| - 1) + 1 \quad 3.6$$

Thus, L_p has a floating point number. Utilizing the decimal part of L_p to access the sample at the position in S . At the end, this returns a final percentile's value (p_{90}).

The results are presented in Figure 3.4, which shows the individual errors of each sensor across all samples. Each test execution involves the collection of 500 data points (samples) to ensure robustness and consistency in the results.

In Table 3.1, the values of the percentage errors for 50%, 75%, 95%, and 99% are presented for each sensor, namely back, right, left, and front. To gain a comprehensive understanding, the analysis considers the 500 values examined for each sensor. It can be observed that 50% of these values exhibit errors less than 2.0 mm for the back sensor, 5.5 mm for the right sensor, and so forth. The same analysis can be applied to each sensor and for each error percentile. In addition, Table 3.2 shows the mean values and standard deviation of the measurements for each sensor. The distance between the sensor and the wall is 145.00 cm (1450 mm).

Figure 3.4 – Measurement Error (mm) in Stationary Test.



Source: The author.

Table 3.1 – Percentage Errors for the Sensors.

Percentile	Sensor Back	Sensor Right	Sensor Left	Sensor Front
50 %	2.0 mm	5.5 mm	2.0 mm	4.0 mm
75 %	4.0 mm	7.0 mm	3.0 mm	6.0 mm
95 %	6.0 mm	10.0 mm	5.0 mm	8.0 mm
99 %	7.0 mm	10.0 mm	6.0 mm	10.0 mm

Source: The author.

Table 3.2 – Mean and Standard Deviation Values for the Sensors Measurements.

Sensor	Mean	Standard Deviation
Back	144.80 cm	0.217 cm
Right	144.45 cm	0.203 cm
Left	145.00 cm	0.239 cm
Front	145.40 cm	0.204 cm

Source: The author.

Thus, starting with the most critical case, $p_{99\%}$, the percentile for the back sensor is less than 7 mm, the right is less than 10 mm, the left is less than 6 mm, and the front is less than 10 mm. Considering the mean, although the errors are relatively small, the sensors with considerable uncertainties are the right and front.

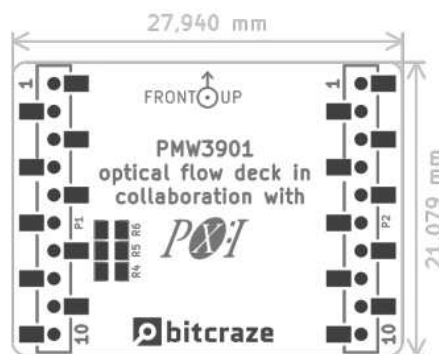
Even in stationary executions, the sensor measurements exhibit fluctuations of approximately 1 cm, which may be deemed acceptable for certain drone use cases. Interestingly, the scenario with the lowest error across all sensors was observed under optimal lighting conditions and with white boards present. The assumption for this scenario was to serve as a baseline, where the drone has good conditions to make the best measurement with as little interference as possible.

3.3. Flow Deck v2

As seen above, using the multi-ranger and flow deck combination increases Crazyflie's potential. The Flow Deck v2 enhances the capabilities of Crazyflie 2.1 by providing it with a comprehensive understanding of its movement in all directions. This advanced expansion deck combines two essential sensors: the VL53L1x ToF sensor, which measures ground distance with high precision, and the PMW3901 optical flow sensor, which tracks movement relative to the ground. Considering that, the next section (*Section 3.3.1*) will present some of the important concepts for the functionality of this powerful deck.

The Crazyflie 2.1 with the flow deck transforms into a versatile 3D flying robot. It becomes capable of executing pre-programmed flight paths with precision in any direction (Bitcraze, 2022b). Figure 3.5 illustrates the mechanical design of the Flow Deck v2, taken from the deck's datasheet.

Figure 3.5 – Mechanical Design of the Flow Deck.



Source: Flow Deck v2 Datasheet (2020).

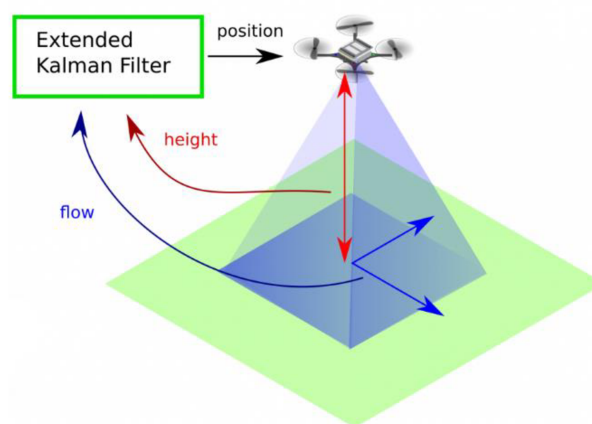
The possibility that the optical flow and laser-based altitude measurements give to the quadcopters is to position themselves within an indoor environment (Kefferputz; McGuire, 2022). Additionally, the Flow Deck v2 offers a significantly stable flying platform for learning and experimentation. The measures absolute range up to four meters. The ToF measurements of the flow-deck are also subject to a simple outlier rejection scheme, allowing the Crazyflie to pass over ground obstacles without causing height jumps.

Unfortunately, it has some relevant limitations. For example, using the flow deck, some types of floor textures, dark colors and low light conditions, can be challenging for the stability and flight of the drone (McGuire *et al.*, 2019).

3.3.1 Understanding the Measurement

Based on Bitcraze (2023) the flow deck gives a relative position, which means wherever the Crazyflie starts in the environment is the (0,0,0) position, the origin. The Extended Kalman Filter (EKF) is capable of processing information, in this case, flow and height, to determine velocity, and with this, it is possible to estimate the position with dead reckoning. Figure 3.6 represents this process of the EKF, using Flow Deck v2. The optical flow sensor calculates pixel flow per frame, and the range sensor measures height up to four meters.

Figure 3.6 – Process of the EKF using Flow Deck v2.

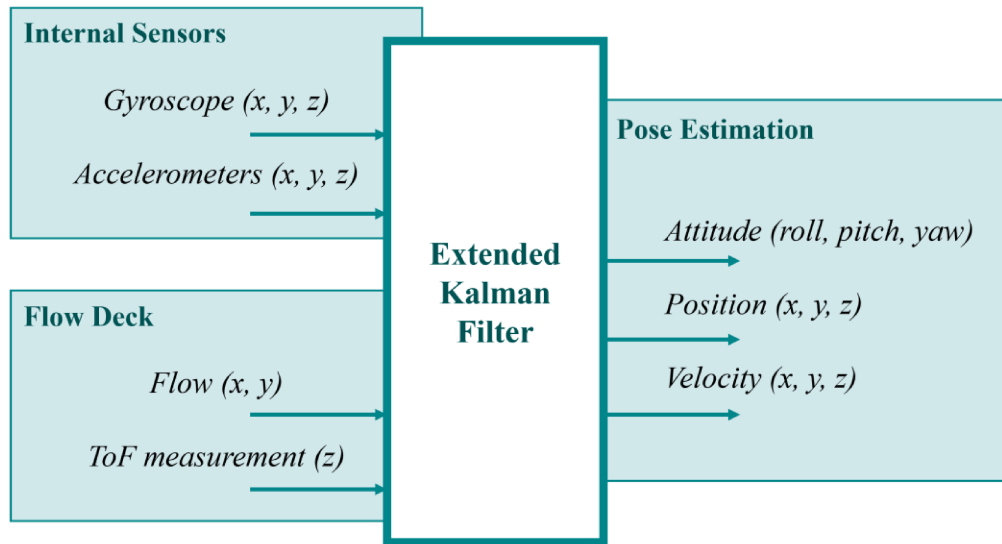


Source: Bitcraze (2023).

A state estimator turns sensor signals into an estimate of the state, an essential part of a stabilizing system. The EKF can accept more outputs from internal and external sensors and provides information for pose estimation (position, velocity, and attitude), as shows Figure 3.7.

Being a recursive filter, the EKF can estimate the current state of Crazyflie based on the measurements received, taking into account the measurement model and the model of the system itself (Bitcraze, 2022d).

Figure 3.7 – Process of the EKF.



Source: The author. Adapted from Bitcraze (2022d).

The EKF used in state estimation is a recommended method for non-linear dynamic systems and is suitable for sensor fusion, where the information comes from different types of sensors (Lima *et al.*, 2019). For many IMU-driven navigation systems, the EKF cost is from the inversion and matrix operations of the gain computation (Greiff, 2017).

3.4. Algorithms – 3D Map

For the algorithm approaches that create 3D maps, it is necessary to consider the points in (x, y, z) , and the source used is a point cloud map. The point cloud is an environment representation that utilizes discrete Cartesian points, and in a 3D environment, the point cloud is generated from a set of 3D coordinates of points.

A fundamental stage in the creation of maps is data collection. Therefore, the first step in creating a map of points is to collect data and information. As seen above, the multi-ranger deck can provide information about obstacles using points.

Utilizing the multi-ranger deck with the available Crazyflie libraries, it is possible to retrieve each sensor measurement in millimeters. The measurement is the distance between one of the sensors and an obstacle, with one value available for each sensor. The measurement becomes even more beneficial when related to the quadcopter's position information. Utilizing the data collected by the flow deck with the measurements captured by the multi-ranger deck, it is possible to create a 3D set of Cartesian points of the environment. Algorithm 1 shows the basic idea behind generating the data for the point cloud.

The procedure executes for each new measurement v at time t , requiring both the sensor data and the quadcopter pose p at time t . The pose of the quadcopter includes the position (x, y, z) and orientation (φ, θ, ψ) (Siegwart; Nourbakhsh; Scaramuzza, 2011). The procedure $3D_rotation_matrix(v, p_t)$ receives a single measurement v in which v is a non-zero integer value and the nano quadcopter's pose. The return of the procedure is the 3D coordinate of the measurement. The measurement coordinate is passed into an output. This output is a representation that, after reading, is transformed using the rotation matrix, returning the modified coordinates to the output.

Table 3.3 – Algorithm 1.

Algorithm 1. Generate the Point Cloud Data.

```

1: procedure PROCESS_DATA_FRAME( $v_t, p_t$ )
2:   for all  $v$  in  $S_t$  do ◇ For each sensor.
3:      $c \leftarrow 3d\_rotation\_matrix(v, p_t)$ 
4:     add_output( $x^*, y^*, z^*$ )

```

Source: The author.

Considering line 3 in Algorithm 1, the idea is that the first step generates a vector from the nano quadcopter's position (x, y, z) with the measurement point (sensor reading). In this case, for example, utilizing the multi-ranger left sensor reading l and the vector \vec{o} as the actual coordinate, the vector left is created as $\vec{l} = (o_x, o_y + l, o_z)$. In other words, a measurement from the left sensor has the same x , a positive offset in the y , and the same z . It is valid for all the sensor readings of the four sensors.

As follows, based on (Goldstein; Poole; Safko, 2011), a list of the equations that represent how all vectors are generated for each of the sensors (\vec{b} for the back, \vec{f} for the front, \vec{l} for the left, and \vec{r} for the right), with the same 3D coordinate of the nano quadcopter.

$$\vec{b} = (o_x - b, o_y, o_z) \quad 3.7$$

$$\vec{f} = (o_x + f, o_y, o_z) \quad 3.8$$

$$\vec{l} = (o_x, o_y + l, o_z) \quad 3.9$$

$$\vec{r} = (o_x, o_y - r, o_z) \quad 3.10$$

With the adequate vector for each measurement of the four sensors, the Direction Cosine Matrix described in *Section 3.1.1* is utilized. Each final value (measurement) is a 3D coordinate, as can be seen in the following equations. The final calculated values (v_{sensor}) are added to the output to generate the point cloud map. Where sensor: b, f, l , and r .

$$v_b = (R_l \cdot \vec{b}) + \vec{o} \quad 3.11$$

$$v_f = (R_l \cdot \vec{f}) + \vec{o} \quad 3.12$$

$$v_l = (R_l \cdot \vec{l}) + \vec{o} \quad 3.13$$

$$v_r = (R_l \cdot \vec{r}) + \vec{o} \quad 3.14$$

This computation process continues for each subsequent measurement. Because of the computational cost, the calculation is delegated to another thread to avoid blocking threads responsible for managing the nano quadcopter.

The calculations for the rotation matrices utilize the NumPy library (Harris *et al.*, 2020). NumPy is a library for scientific computing in Python. This avoids the need to implement something already well-established, well-optimized, and tested by the scientific community. The current work utilizes NumPy for the array operations and the matrices dot products, highly simplifying the code.

3.4.1 Visualization Approaches

There are different techniques for the visualization of the point cloud data. This work explores three approaches using two different libraries available in the Python programming language.

The first approach is the option to visualize using the VisPy library (Campagnola *et al.*, 2023). This first choice is motivated by the Bitcraze company. VisPy is a powerful interactive data visualization library designed to leverage the immense computational capabilities of

modern GPUs, Graphics Processing Units, for 2D and 3D. Utilizing the OpenGL library, VisPy can display and handle extensive datasets, and users can explore and interact with data in real-time. VisPy stands out for focusing on performance and delivering fast and smooth visualizations, even when dealing with complex data.

Sequentially, the second library is the Open3D (Zhou; Park; Koltun, 2018). The Open3D is an excellent open-source library designed to facilitate the rapid development of software dealing with 3D data, offering a comprehensive set of meticulously selected data structures and algorithms in the front end. On the other hand, the back end is highly optimized and geared towards efficient parallelization, providing optimal performance. Open3D focuses on simplicity, and its code is clean and versatile, offering easy setup and compilation from sources across various platforms, maintained through a robust code review mechanism, embracing contributions from the open-source community and research projects.

Once all the data generated by the nano quadcopter using the multi-ranger deck has been processed and the 3D coordinates have already been generated, the visualization using these techniques becomes a matter of drawing the points on the screen. In each measurement, the values of the four sensors are captured, and four 3D points are added to the point cloud dataset. In addition to visualization, Open3D allows you to manipulate data for surface reconstruction.

3.5. Algorithms – 2D Map

The third mapping technique used in this work is a map representation based on probabilities in 2D form. The grid maps discretize the environment into grid cells, where each cell fills in information about the environment, and a famous algorithm is occupancy grid mapping. For each cell the algorithm determines a single value, considering the probability (Stachniss, 2009). Basically, the value of this cell can vary in three modes, one being free/unoccupied, occupied, and uncertain/unknown, represented by the colors white, black, and grey, respectively. The Matplotlib library (Hunter, 2007) was used to visualize the occupancy grid map (2D map).

Addressing the problem of generating consistent maps, occupancy grid treats noisy and uncertain measurement data, with the assumption that the robot's pose is known (Thrun; Burgard; Fox, 2005). There are some advantages to using this algorithm, such as providing the ability to model unknown areas and using the values directly to update the values of the cell (Siegwart; Nourbakhsh; Scaramuzza, 2011; Stachniss, 2009).

Therefore, it is necessary to understand the dynamics of this map. As follows, all the equations and the potential analyses mentioned are based on the book Probabilistic Robotics (2005), from the authors Thrun, Burgard and Fox.

It is comprehended that each cell i in the grid is a binary random variable that models the occupancy (obstacle) in the position associated with the cell. In this way, the probability can be indicated by the set of (independent) probabilities. Initially, all cells have a value to indicate the unknown. The principal algorithm for calculating the belief in robotics is the Bayes filter. The Bayes filter is recursive, regarding the calculated belief at time t is from the belief at time $t - 1$. However, it is not adequate for learning maps due to its size, so the approach is to divide, break the map, and estimate the cells individually.

The fundamental principle of the occupancy grid mapping algorithm is to represent the map m as the union of all cells (Equation 3.15) and to calculate the posterior probability distribution over maps based on the available data (Equation 3.16). Where m is the map, $z_{1:i}$ is the measurements (sensors reading) up to time t , and $x_{i:t}$ is the path of the nano quadcopter (pose or state).

$$m = \sum_i m_i \rightarrow p(m) = \prod_i p(m_i) \quad 3.15$$

$$p(m | z_{1:i}, x_{i:t}) \quad 3.16$$

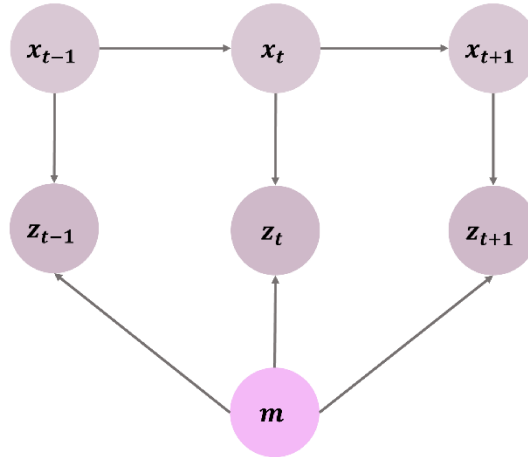
The map can quickly become unmanageable due to its size, for example, a 100x100 grid has $2^{10,000}$ possible binary maps. The solution to this problem is an approximation using marginal probability, which can be seen in Equation 3.17.

$$p(m | z_{1:i}, x_{i:t}) \cong \prod_i p(m_i | z_{1:i}, x_{i:t}) \quad 3.17$$

Also, Figure 3.8 illustrates a graphical mapping model with known poses using the occupancy grid mapping technique.

Now, there is a problem with the calculation of $p(m | z_{1:i}, x_{i:t})$. To solve the Equation 3.17, it is necessary to use the Conditional Bayes Theorem with a Markov assumption. The Bayes filter makes a Markov assumption. The Markov assumption implies that the belief is adequate to define the history of the robot, like an approximation. Thus, the Markov assumption postulates that past, and future are independent if one knows the current pose x_t .

Figure 3.8 – Graphical Mapping Model Occupancy Grid.



Source: The author. Adapted from (Thrun; Burgard; Fox, 2005)

Considering all the concepts presented, Equation 3.18 indicates the application of Bayes Theorem, and it represents the probability of the occupied cell given that a certain reading has been considered and the robot is in a certain position.

$$p(m_i | z_{1:t}, x_{i:t}) = \frac{p(z_t | m_i, z_{1:t-1}, x_{1:t})p(m_i | z_{1:t-1}, x_{1:t})}{p(z_t | z_{1:t-1}, x_{1:t})} \quad 3.18$$

In this way, Equation 3.19 shows the changes that using Markov causes to Equation 3.18, assuming that the current measurement only depends on the current state and the map cell. Furthermore, the current state without the current measurement does not provide any additional information about the occupancy of the cell since the events are independent.

$$p(m_i | z_{1:t}, x_{i:t}) = \frac{p(z_t | m_i, x_t)p(m_i | z_{1:t-1}, x_{1:t-1})}{p(z_t | z_{1:t-1}, x_{1:t})} \quad 3.19$$

A further simplification can be obtained, finally reaching Equation 3.20. The current map depends on the sensor reading and how the environment is, always considering that the past and future are independent, knowing only the present state or pose of the robot, in this case, the nano quadcopter.

$$p(m_i | z_{1:t}, x_{i:t}) = \frac{p(m_i | z_t, x_t)p(z_t | x_t)p(m_i | z_{1:t-1}, x_{1:t-1})}{p(m_i)p(z_t | z_{1:t-1}, x_{1:t})} \quad 3.20$$

Unfortunately, the direct manipulation of probabilities can be a complicated task. One resource that can be used to simplify things is to use the ratio of probabilities, which is also a way of associating the events occurring for occ (occupied) and free (empty) for cells. This probability ratio can be called odds, which are the chances. Equation 3.21 illustrates a generic representation of the chance of event A happening.

$$\text{odds}(A) = o(A) = \frac{p(A)}{p(\neg A)} = \frac{p(A)}{1 - p(A)} \quad 3.21$$

Thus, Equation 3.20 is manipulated to find the odds based on Equation 3.21 and then obtained Equation 3.22.

$$\begin{aligned} & \frac{p(m_i | z_{1:t}, x_{i:t})}{1 - p(m_i | z_{1:t}, x_{i:t})} = \\ & = \frac{p(m_i | z_t, x_t)}{1 - p(m_i | z_t, x_t)} \frac{p(m_i | z_{1:t-1}, x_{1:t-1})}{1 - p(m_i | z_{1:t-1}, x_{1:t-1})} \frac{1 - p(m_i)}{p(m_i)} \end{aligned} \quad 3.22$$

Avoiding rounding problems when multiplying probabilities and considering the efficiency, it is necessary to use the famous log odds. In other words, it is the log of the odds and helps with efficiency and numerical instability. Equation 3.23 illustrates the log odds of event A (generic).

$$\log \text{odds}(A) = l(A) = \log(o(A)) = \log\left(\frac{p(A)}{p(\neg A)}\right) \quad 3.23$$

By manipulating Equation 3.22 and Equation 3.23, we finally have Equation 3.24, which is the elemental equation of the algorithm.

$$l_{t,i} = l(m_i | z_{1:t}, x_{i:t}) = \log\left(\frac{p(m_i | z_{1:i}, x_{i:t})}{1 - p(m_i | z_{1:i}, x_{i:t})}\right) \quad 3.24$$

It is also possible to recover the probability from the log odds, as seen in Equation 3.25.

$$p(m_i | z_{1:t}, x_{i:t}) = 1 - \frac{1}{1 + \exp(l_{t,i})} \quad 3.25$$

Algorithm 2 shows the algorithm to update the map on each new measurement. In this way, on each new measurement, the algorithm loops through all the cells in the view of the sensor field. Cells outside the field of view remain unchanged. Cells in the field of view are updated utilizing the prior value and the inverse sensor model. The last term is a constant of the cell value at time 0.

Table 3.4 – Algorithm 2.

Algorithm 2. Occupancy Grid Algorithm.

```

1: procedure OCCUPANCY_GRID_MAPPING( $m, x_t, z_t$ )
2:   for all  $c$  in  $m$  do
3:     if  $c$  in perceptual field of  $z_t$  then
4:        $m_{t,c} = m_{t-1,c} + \text{inverse\_sensor\_model}(c, x_t, z_t) - m_{c,0}$ 
5:     else
6:        $m_{t,c} = m_{t-1,c}$ 

```

Source: The author.

Algorithm 3 shows the procedure for the inverse sensor model utilized to update each cell. This procedure has two well-defined constants L_{occ} and L_{free} . These constants represent the probability of an occupied and free cell, respectively. The arguments are the cell to update, the pose, and the measurement.

Cells outside the field of view remain at the same value as they are unexplored. Cells at the edge of the field mean that the measurement has found an obstacle, so the cell is updated as likely to be occupied. Cells are updated as the probability of being free within the field of view that has not reached an obstacle. The cell values are then updated using log odds.

The implementation of Algorithm 2 and Algorithm 3 are complete. Now, the effort becomes to identify the cells within the multi-ranger's field of view. With the drone pose x_t known and the point reading by the sensor z_t , a straight line is drawn between the two pieces of information. The cell in which x_t occupies is updated as probably free, that is, updating as probably free all the cells in the path between x_t and z_t .

Table 3.5 – Algorithm 3.

Algorithm 3. An algorithm for the inverse sensor model.

```

1: procedure INVERSE_SENSOR_MODEL( $c, x_t, z_t$ )
2:   if  $c$  is not in perceptual field of  $z_t$  then                                 $\diamond$  Unknown.
3:     return  $c$ 
4:   if  $c$  at the edge of view of  $z_t$  then                                     $\diamond$  Cell is occupied.
5:     return  $L_{occ}$ 
6:   if  $c$  within the field of view of  $z_t$  then                                 $\diamond$  Cell is free.
7:     return  $L_{free}$ 

```

Source: The author.

Moreover, note that the characteristics of a sensor are invariant to the absolute coordinates of the nano quadcopter or grid cell when assuming a measurement. Thus, if the quadcopter pose can be denoted by x_t and the grid cell by m_i , the grid cell coordinates are transferred to the quadcopter local reference frame using Equation 3.26.

$$\begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x_{m_i} - x \\ y_{m_i} - y \end{pmatrix} \quad 3.26$$

The cell identified by x_t is updated as probably occupied. Each cell is updated utilizing the log odds with the defined constant L_{occ} and L_{free} , and the values are $L_{occ} = 0.9$ and $L_{free} = 0.1$. The constant of the map at time zero is 0.5, initializing all cells with this value, meaning that there is not enough information available to know the situation.

Since this is a grid with cells, the application of the Bresenham algorithm is to identify the cells in the path between the points x_t and z_t . More about the Bresenham algorithm will be clarified in *Section 3.5.1*.

3.5.1 Bresenham Algorithm

For the Occupancy Grid, it is not necessary to go through the entire map with each new sensor reading while only updating the cells associated with the readings. Therefore, it is the justification for the Bresenham algorithm use. Algorithm 4 presents the complete algorithm with all the steps used in the current work.

Table 3.6 – Algorithm 4.

 Algorithm 4. Bresenham Algorithm.

```

1: procedure BRESENHAM_ALGORITHM((  $x_1, y_1$ )(  $x_2, y_2$ ))
2:   Coordinates  $\leftarrow \langle \rangle$ 
3:    $dx \leftarrow x_2 - x_1$ 
4:    $dy \leftarrow y_2 - y_1$ 
5:    $p \leftarrow 2 * dy - dx$ 
6:    $p2 \leftarrow 2 * dy$ 
7:    $xy2 \leftarrow 2 * (dy - dx)$ 

8:   if  $x_1 > x_2$  then
9:      $x \leftarrow x_2$ 
10:     $y \leftarrow y_2$ 
11:     $x_f \leftarrow x_1$ 
12:  else
13:     $x \leftarrow x_1$ 
14:     $y \leftarrow y_1$ 
15:     $x_f \leftarrow x_2$ 

16:  Coordinates  $\|(x, y)$ 
17:  while  $x < x_f$  do
18:     $x \leftarrow x + 1$ 
19:    if  $p < 0$  then
20:       $p \leftarrow p + p2$ 
21:    else
22:       $y \leftarrow y + 1$ 
23:       $p \leftarrow p + xy2$ 
24:    Coordinates  $\|(x, y)$ 

25:  return Coordinates

```

Source: The author.

The choice of the Bresenham algorithm (Bresenham, 1965) is moved by its simplicity and efficiency in computer graphics, basically for drawing lines. The algorithm effectively

avoids division, only using integer values. In addition, the algorithm is efficient in terms of calculation speed and memory consumption (Bresenham, 1965).

Based on the work of Bresenham (1965), the algorithm assumes a mesh with contiguous cells. In this manner, given two data points $D_1(x_1, y_1)$ and $D_2(x_2, y_2)$ the algorithm follows in the mesh going to either $(x_1 + 1, y_1)$ or $(x_1 + 1, y_1 + 1)$ until it reaches D_2 . The direction step might vary with the octant the coordinates are in.

Regarding the first octant, the coordinate selection movements have a recursive relationship, as seen in Equation 3.27.

Where $\Delta a = x_2 - x_1$ and $\Delta b = y_2 - y_1$.

$$\left\{ \begin{array}{l} \nabla_1 = 2\Delta b - \Delta a \\ \nabla_{i+1} = \begin{cases} \nabla_i + 2\Delta b - 2\Delta a & \text{if } \nabla_i \geq 0 \\ \nabla_i + 2\Delta b & \text{if } \nabla_i < 0 \end{cases} \end{array} \right. \quad 3.27$$

3.6. Flight Strategies

After all the concepts presented above, it is still necessary to consider how the nano quadcopter will travel through the environment. In this way, there are two different styles developed in this project: a manual path, where the path the drone should take is defined, and the other, wall following, an autonomous flight where the robot makes its own decisions according to previously established settings and the environment.

3.6.1 Manual Path

The application for the manual path utilizes the MotionCommander blocking APIS and the logging framework. This approach requires prior information about the environment. Knowledge of when to turn, how many meters should the drone fly in some direction, and so on. The distances must be accurate and optimally include a distance interval from the wall to tolerate errors and avoid crashes.

The callbacks registered in the logging framework utilize the buffer approach to persist data. The application sends the commands through the MotionCommander serially, and after the predefined path finishes, the drone lands and flushes the data to disk before finishing

execution. The applications developed to utilize the manual path have a more straightforward implementation. The scripts are usually easier to understand since commands execute serially.

This utilization of the manual path, along with the blocking API, makes a perfect case for experimentation with the drone. This simplification is powerful and especially useful when testing the other decks attached. For instance, it enables verifying how to collect data utilizing the logging framework and persist it to the disk. Because of this facilitation in the experiments, the current work found that the buffered approach improved the real-world results, making it evident that the callbacks registered in the logging framework should all be non-blocking.

However, given the simplification in the utilization, it does not come for free. As mentioned, the MotionCommander API blocks for an established period, expecting the drone to have completed the movement by then. Unfortunately, this approach is not capable of dealing with unexpected circumstances or possible errors properly, as it continues executing a command until the end.

One such example of a restriction of movement happens when the drone directly crashes against an obstacle. Since the command must execute to the end before returning control to the caller, it is not possible to recover from a crash or try to move away from the wall. This restriction renders the utilization, both of the manual path and blocking APIs, unfit for developing an application for autonomous flight.

3.6.2 *Wall Following*

The wall following utilizes the MotionCommander with non-blocking commands and the logging framework to provide autonomous flight. Given the utilization of non-blocking commands, the drone reacts in each measurement to avoid crashing against obstacles. The wall following is an application to traverse an unknown environment autonomously.

Oversimplifying how the wall following works, the drone starts at an arbitrary position and takes flight. The first objective is to be parallel to a wall, so the drone flies to find a wall and then readjusts its position. The drone continues flying parallel to the wall until finding a corner, and the adjustment process begins again. This process continues until manually stopping the application where the drone lands and flushes data to disk.

As expected, comprehending the performance and operation is equally complex, given the complexity of the code for this behavior. The application starts and executes until the drone lands, where the core algorithm for wall following is non-blocking. After receiving a

measurement from the LogConfig, the drone readjusts accordingly with the new readings. The adjustment loop executes for each new measurement.

The reaction movement to readjust the drone involves sending a command with the MotionCommander non-blocking API. The application utilizes the buffered approach to persist the measurements and avoid blocking. To complete the non-blocking requirements, the default class to listen in the logging framework, the SyncLogger, has to be non-blocking. Extending the implementation to remove the blocking calls to the events queue was enough to solve the issue. All these aspects together make the wall following application highly responsive and non-blocking.

In the wall following algorithm, the drone keeps following the wall indefinitely (wall-following behavior). Therefore, a stop mechanism is a requirement. This mechanism should safely land the drone and flush the data to disk. The stop mechanism must happen remotely through the application and not require external interference applied to the drone, which would affect the data.

The application developed for the wall following wait for the SIGTERM signal to trigger the stop procedure. This way, the control node receives the SIGTERM signal and initiates a graceful shutdown process, which is a two-fold mechanism, also acting as an "emergency button" if necessary. As the entire procedure is remote, the data is not interfered with, which would not be the case if physical contact with the drone were to occur.

Another benefit of the non-blocking application is its responsiveness. The application promptly handles the SIGTERM signal since there is no one blocking to handle the signal. This responsiveness is essential during experiments, and a force stop is needed. In contrast with the manual path and its blocking commands, it would not be capable of handling the SIGTERM signal until the command finished execution.

The autonomous application developed for the wall following is much more complex than the manual path. As such, it covers all the weak points mentioned in adaptability and needing previous knowledge of the environment. The drone is capable of autonomously traversing the environment by following its walls. The drone can adapt and readjust its position based on the newest readings.

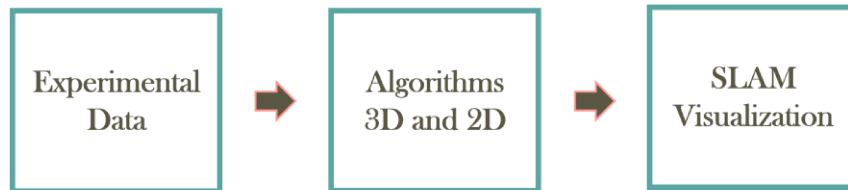
3.7. Considerations

In this chapter, we present the system used in this work, with all its fundamental parts, as an entirety to create the solution to the SLAM problem. In this way, the chapter covers the

physical materials, such as the Crazyflie nano quadcopter, the multi-ranger deck for measurement, and the flow deck for localization, with the main concepts mentioned.

Figure 3.9 summarizes the process (the methodology applied), with the first stage being data collection through experiments, data processing, and visualization of the solution to the problem.

Figure 3.9 – General Methodology Process.



Source: The author.

Then, there are the methods described. The methods describe the techniques used to develop this current work, focusing on mapping, and will be the basis for the results in the next chapter. The methods are 2D and 3D style maps, with 3D using two different libraries as a technique and 2D using occupancy grid, a probability map technique.

For general consideration, Table 3.7 illustrates a direct comparison between the two flight strategies (manual path and wall following) of their main aspects and characteristics. As a summary, Table 3.8 illustrates the methodologies discussed in this chapter, all of which will be used for the validation and experiments (Chapter 4). The table provides a concise overview of the methods employed in the work.

Table 3.7 – Comparison of Flight Strategies.

Characteristics	Manual Path (MP)	Wall Following (WF)
Prior Knowledge	Maximal	Minimal
Trajectory	Predefined	Autonomous
Implementation	Straightforward	Complex
API	Blocking	Non-Blocking

Source: The author.

Table 3.8 – Overview of the Methods.

Methods		Library
3D	Point Cloud	Open 3D, VisPy
3D	Surface	Open 3D
2D	Occupancy Grid	Matplotlib

Source: The author.

RESULTS AND DISCUSSION

Chapter 4 explains all the results achieved, considering all the proposed techniques and scenarios. There is also a discussion and analysis of the results. Overall, the first section (4.1) presents the initial and principal definitions (settings) for the experiments. The second section (4.2) refers to Scenario 1, the third (4.3) to Scenario 2, and so on until Scenario 4. The sixth section (4.6) presents different approaches dealing with some individual cases. In addition, there are two path approaches that the nano quadcopter, must follow: wall following and manual path. For each path, there were 2D and 3D maps constructed. However, the individual cases use the paths approach according to each case.

4.1. Definitions







This section will elucidate important definitions, offering clarity and facilitating a deeper understanding of each subsequent section. It has been a long journey to achieve the results described here and conducted numerous experiments over these two years. There was a lot of trial and error until we arrived at concise solutions to the SLAM problem because we were dealing with the real world.

Initially, experiments were conducted to ensure that all the materials (nano quadcopter and its decks) were operational and properly functioning. The primary questions about these experiments were how the decks work, how to extract information, and how to deal with it. Then there were the techniques. These techniques need several adjustments on the code to work as expected, always regarding the environment representation and mapping. The nano quadcopter location is not global. It is relative and influences directly on the localization problem.

In order to validate the experiment results, it is necessary to establish the general parameters for the scenarios. Table 4.1 illustrates a summary of the experiments conducted

across different environments (scenarios) using the respective maps and flight strategies. MP is for Manual Path, and WF is for Wall Following.

Table 4.1 – Summary of the Experiments.

Experiment	Map	Flight
 Scenario 1	3D, 2D	MP, WF
 Scenario 2	3D, 2D	MP, WF
 Scenario 3	3D, 2D	MP, WF
 Scenario 4	3D, 2D	MP, WF
 Levels	3D	MP
 Real-Time Visualization	2D	WF

Source: The author.





Table 4.2 shows the general settings for the Scenario 1 to 4. Table 4.3 delineates the color legend for the 3D maps (Open3D Library) to enhance comprehension of the results. In other words, it shows the representations of the measurements of each sensor by different colors. For 3D maps using VisPy, there are only two colors presented by points (dots) on the map: red (pose) and blue (measurements).

Table 4.2 – General Definitions for Scenario 1 to 4.

Settings	Scenario 1 to 4	
	Manual Path	Wall Following
Velocity	0.1 m/s	0.1 m/s
Period	100 ms	100 ms
Default Height	0.3 m	0.3 m
Distance from Wall	-	0.15 m

Source: The author.

Table 4.3 – Color Legend (Open3D).

	Colors	Measurement
	Rajah	Sensor Back
	Comet	Sensor Front
	Perano	Sensor Left
	De York	Sensor Right

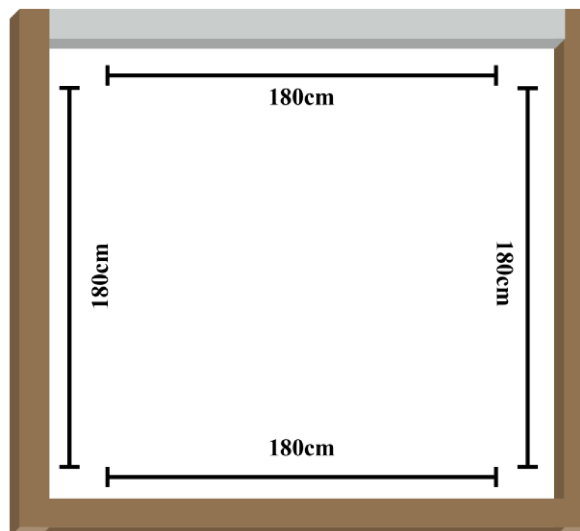
Source: The author.

4.2. Scenario 1

The first scenario is a simple one. The map should resemble a square since the map is a simple square with no obstacles. The justification of the environment as the first case is because of its simplicity.

Setting up the environment is an initial step, using the laboratory space available in block 1C at the Federal University of Uberlândia. In addition to the wall of the room, cardboard was positioned to form the other walls. Figure 4.1 shows the architecture of the environment with its measurements. Figure 4.2 is a photo taken of the first environment.

Figure 4.1 – Architecture of the Scenario 1.



Source: The author.

Figure 4.2 – Photo of the Scenario 1.



Source: The author.

In this way, the following sections will show the results of each technique for this environment. *Sections 4.2.1* and *4.2.2* will present and discuss the results of each SLAM technique. The results consider different techniques, and these techniques are presented in the methods in Chapter 3. There are two forms of behavior (predefined path and wall-following) for each scenario, with their respective SLAM results.

4.2.1 Manual Path – Scenario 1

Considering the behavior of following a defined path, the commands were predetermined prior to the experiment. In Table 4.4, these commands are detailed. It was considered the primary settings from Table 4.2 (*Section 4.1*), and the commands in a sequence of forward, sleep, turn left, and sleep (four times) and finally, land at a velocity of 0.1 m/s.

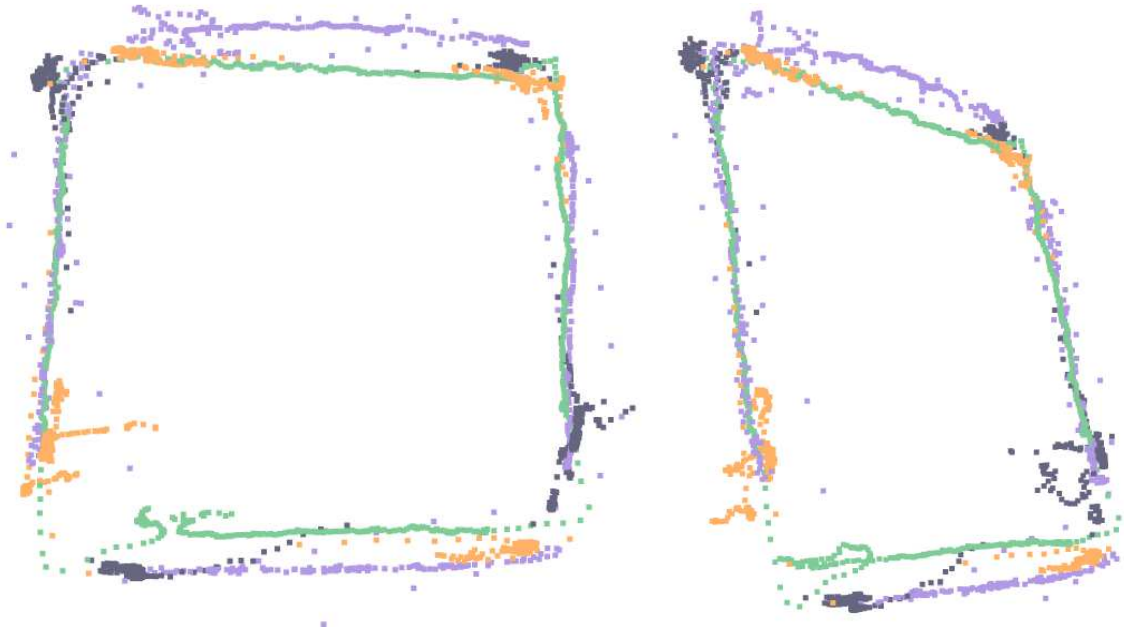
Table 4.4 – Scenario 1 Trajectory MP.

Commander		
	Forward	1.1 m
4*	Sleep	1 s
	Turn Left	90°
	Sleep	1 s
-	Land	0.1 m/s

Source: The author.

Figures 4.3 to 4.6 show the manual path results, in other words, the nano quadcopter's behavior and path results. For a better visualization, some of the Figures have the same plot, despite the two different ways of positioning the image.

Figure 4.3 – Open3D Point Cloud in Scenario 1 Using Manual Path.



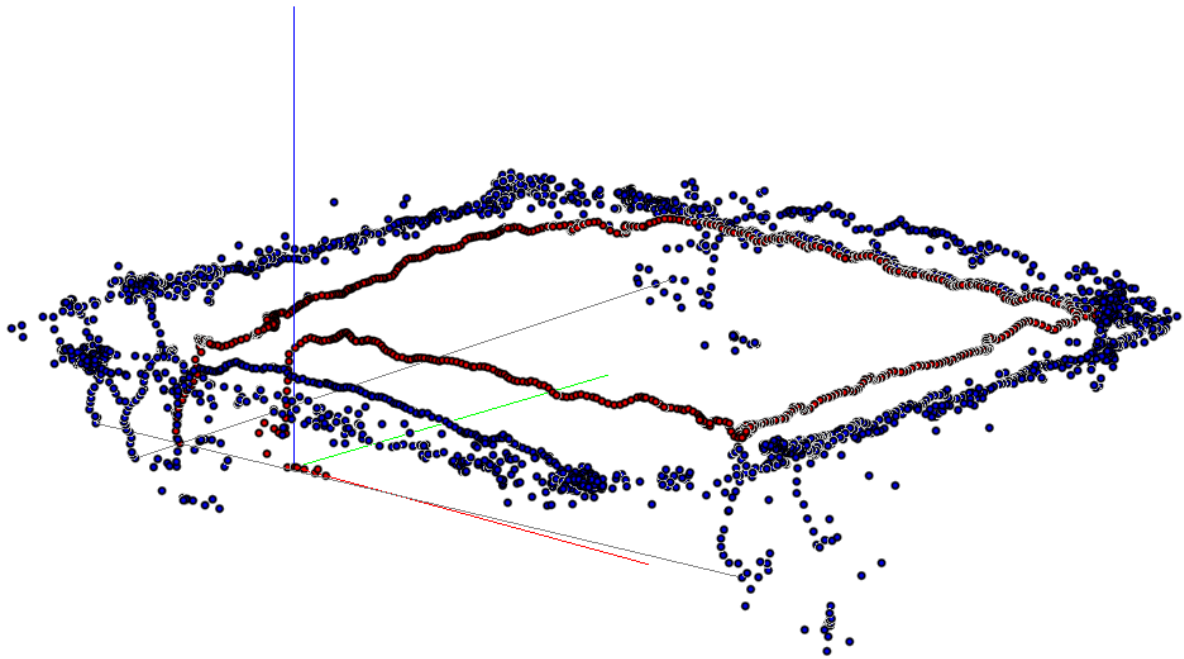
Source: The author.

Figure 4.4 – Open3D Surface in Scenario 1 Using Manual Path.



Source: The author.

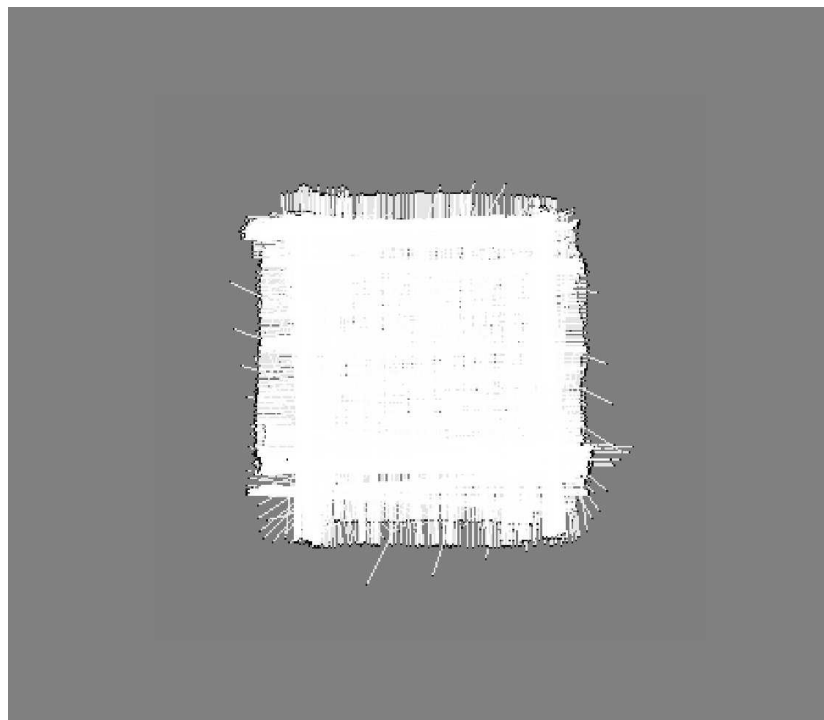
Figure 4.5 – VisPy Point Cloud in Scenario 1 Using Manual Path.



Source: The author.

In *Section 4.7*, final considerations will include some comparative information regarding the occupancy grid map (2D map) and the others 3D maps.

Figure 4.6 – Occupancy Grid in Scenario 1 Using Manual Path.

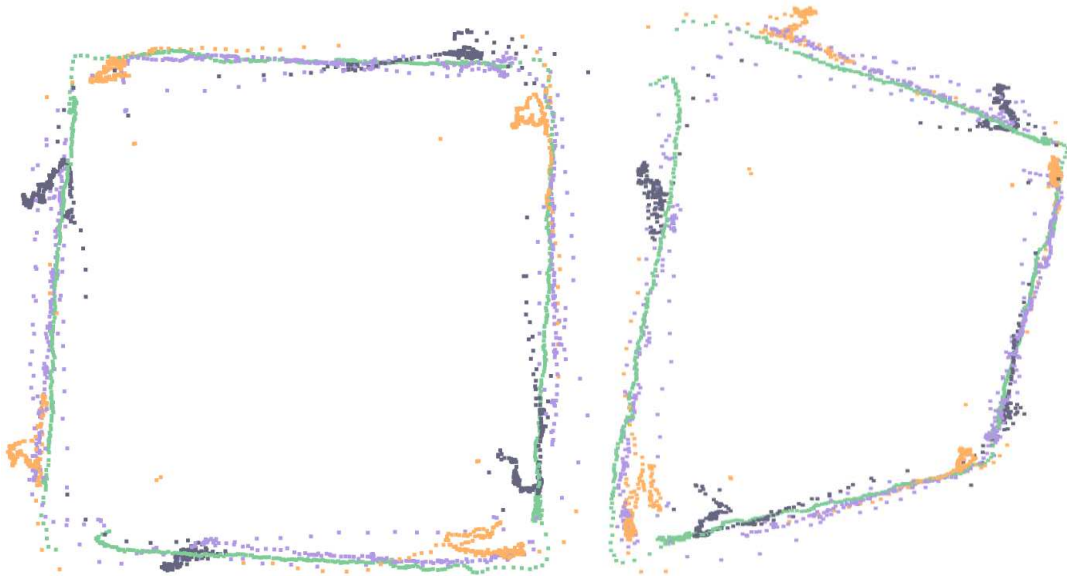


Source: The author.

4.2.2 Wall Following – Scenario 1

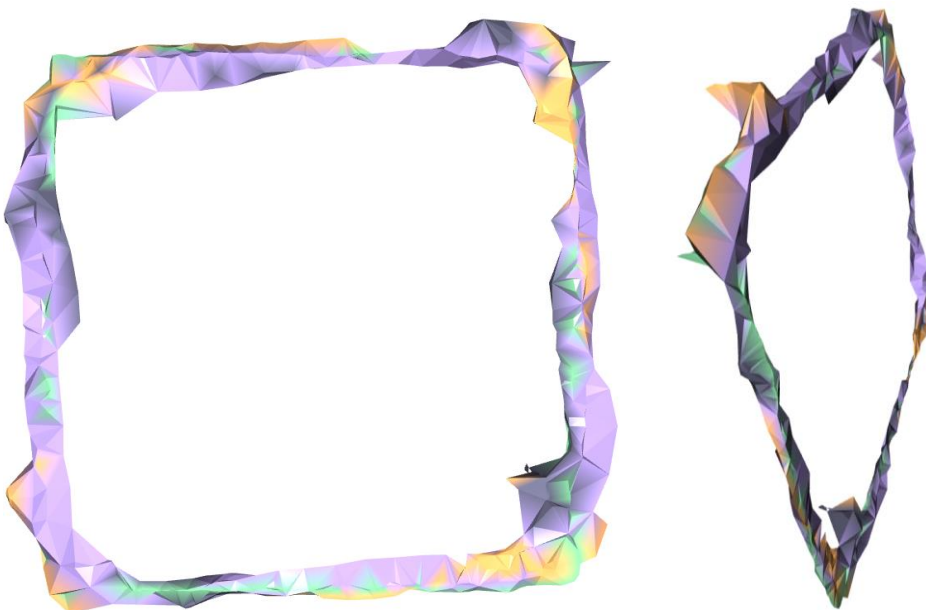
As the behavior is autonomous, setting the trajectory commands is unnecessary. For this purpose, the necessary preliminary information is also available in Table 4.2. Figures 4.7 to 4.10 show the wall following results for the Scenario 1.

Figure 4.7 – Open3D Point Cloud in Scenario 1 Using Wall Following.



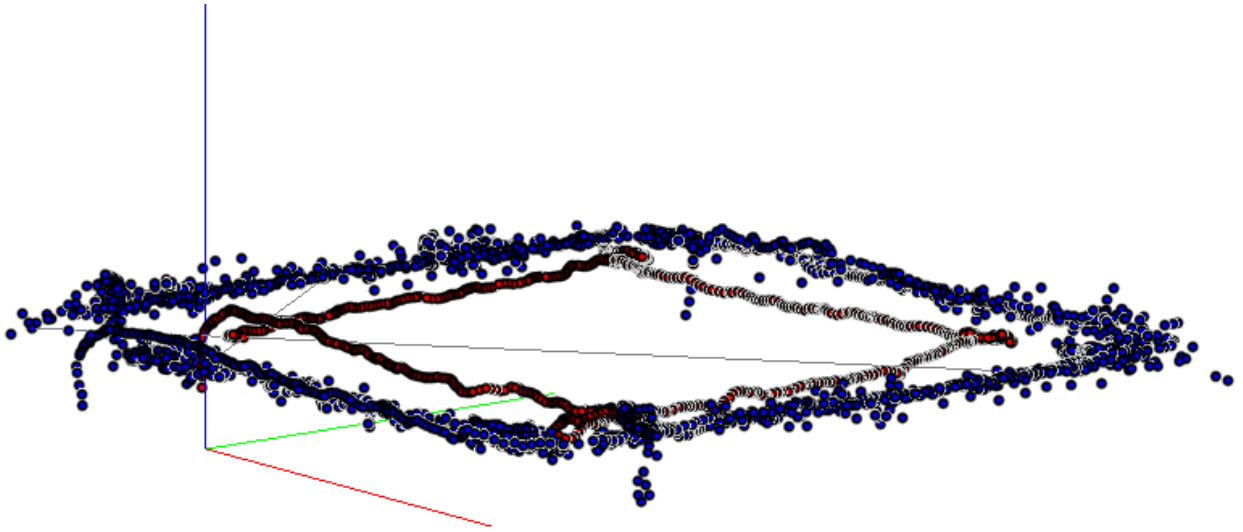
Source: The author.

Figure 4.8 – Open3D Surface in Scenario 1 Using Wall Following.



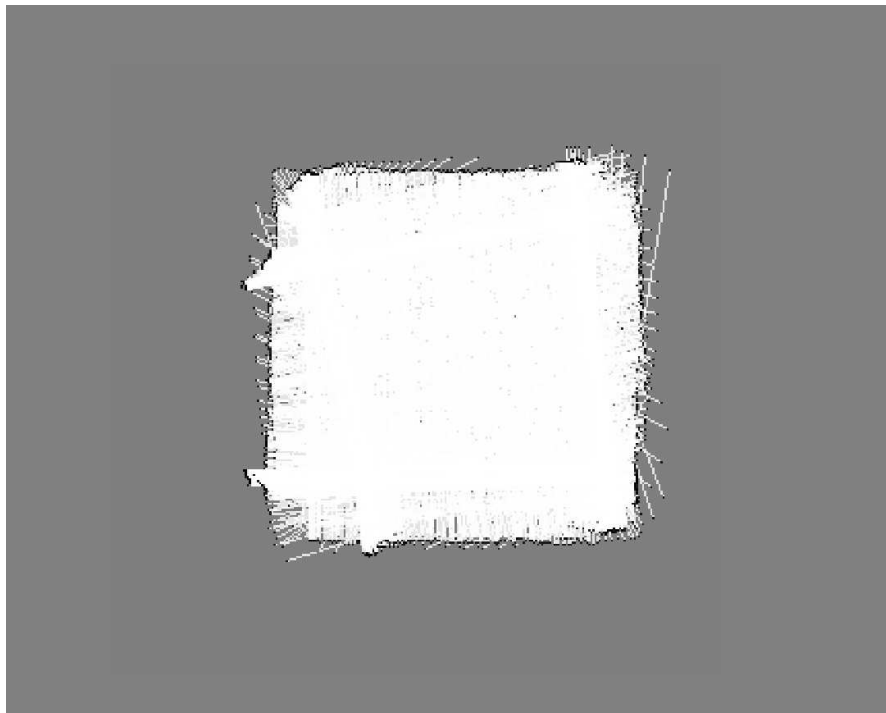
Source: The author.

Figure 4.9 – VisPy Point Cloud in Scenario 1 Using Wall Following.



Source: The author.

Figure 4.10 – Occupancy Grid in Scenario 1 Using Wall Following.



Source: The author.

In *Section 4.7*, final considerations will include some comparative information regarding the occupancy grid map (2D map) and the others 3D maps.

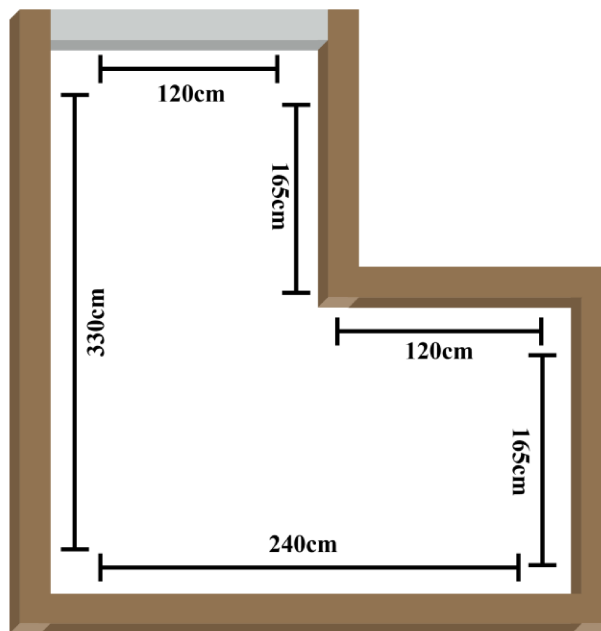
4.3. Scenario 2

The Scenario 2 has a more rectangular shape and was created with the aim of having a map with more geometric shapes, and resemble the letter L.

The process for building this scenario was the same as in Scenario 1. An initial step is setting up the environment using the laboratory space available in block 1C at the Federal University of Uberlândia. Also, in addition to the wall of the room, cardboard was positioned to form the other walls. Figure 4.11 shows the architecture of the environment with its measurements. Figure 4.12 is a photo taken of the second scenario.

In this way, the following sections will show the results of each technique for this environment. *Sections 4.3.1* and *4.3.2* will present and discuss the results of each SLAM technique.

Figure 4.11 – Architecture of the Scenario 2.



Source: The author.

Figure 4.12 – Photo of the Scenario 2.



Source: The author.

4.3.1 Manual Path – Scenario 2

Considering a defined path, the commands were predetermined prior to the experiment. In Table 4.5, these commands are demonstrated.

Table 4.5 – Scenario 2 Trajectory MP.

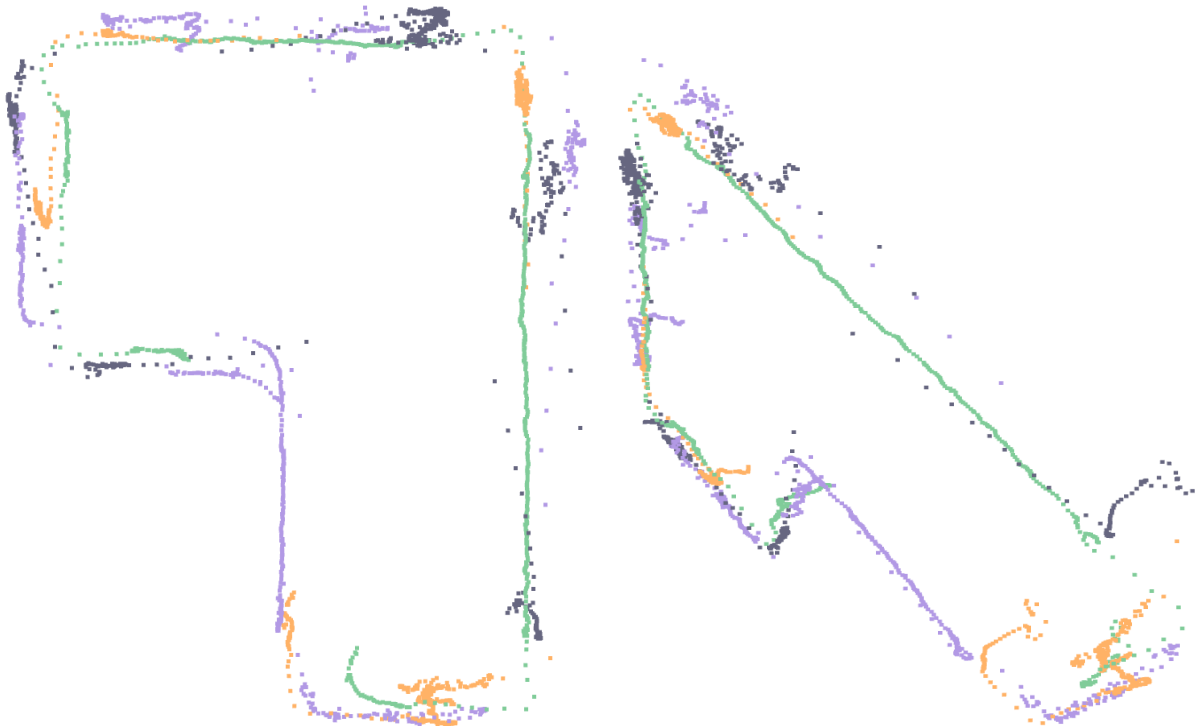
Commander		
1:	Forward	0.2 m
2:	Turn Left	90°
3:	Forward	2.4 m
4:	Turn Left	90°
5:	Forward	1.4 m
6:	Turn Left	90°
7:	Forward	0.3 m
8:	Turn Left	90°
9:	Forward	0.3 m
10:	Turn Left	90°
11:	Land	0.1 m/s

Source: The author.

It was considered the primary setting from Table 4.2 (*Section 4.1*). The drone pursues a sequence of forward, sleep, turn left, and sleep. Finally, the drone landed at a velocity of 0.1 m/s. The command sleep (for one second = 1 s) is not in Table 4.5 but exists in the code, as the sequence explained. The distance for the "forward" differs due to the distance of the environment.

Figures 4.13 to 4.16 show the manual path results, in other words, the nano quadcopter's behavior and path results. For enhanced visual comprehension, it is worth noting that several Figures contain identical plots, each positioned differently to illustrate alternative display methods and improve overall clarity.

Figure 4.13 – Open3D Point Cloud in Scenario 2 Using Manual Path.



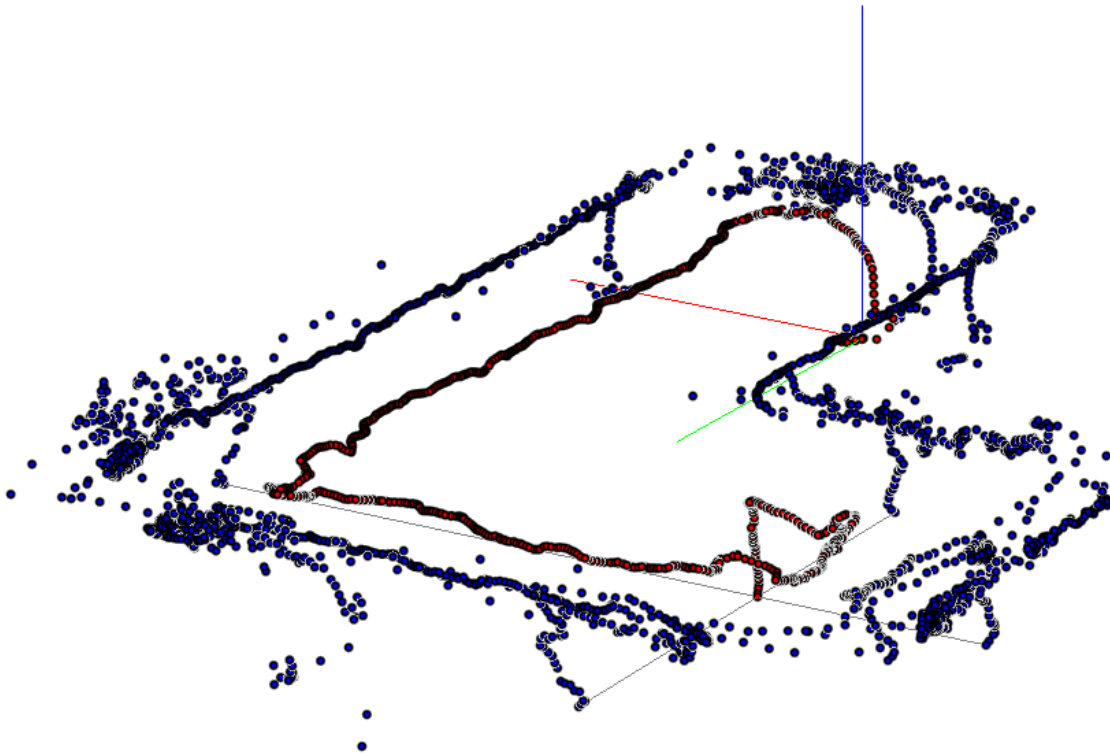
Source: The author.

Figure 4.14 – Open3D Surface in Scenario 2 Using Manual Path.



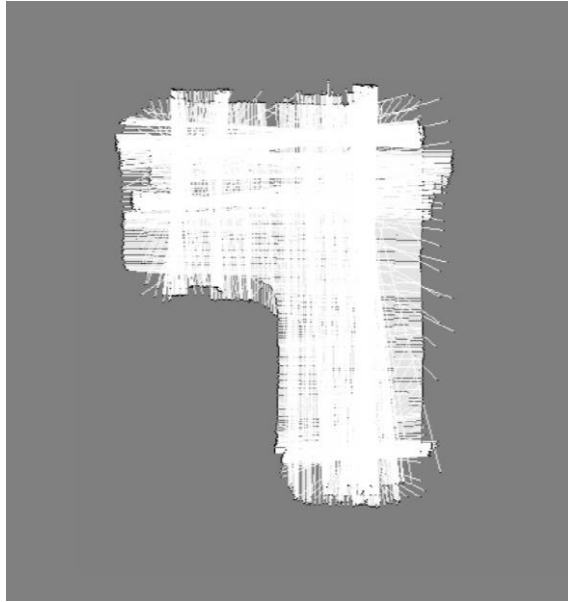
Source: The author.

Figure 4.15 – VisPy Point Cloud in Scenario 2 Using Manual Path.



Source: The author.

Figure 4.16 – Occupancy Grid in Scenario 2 Using Manual Path.

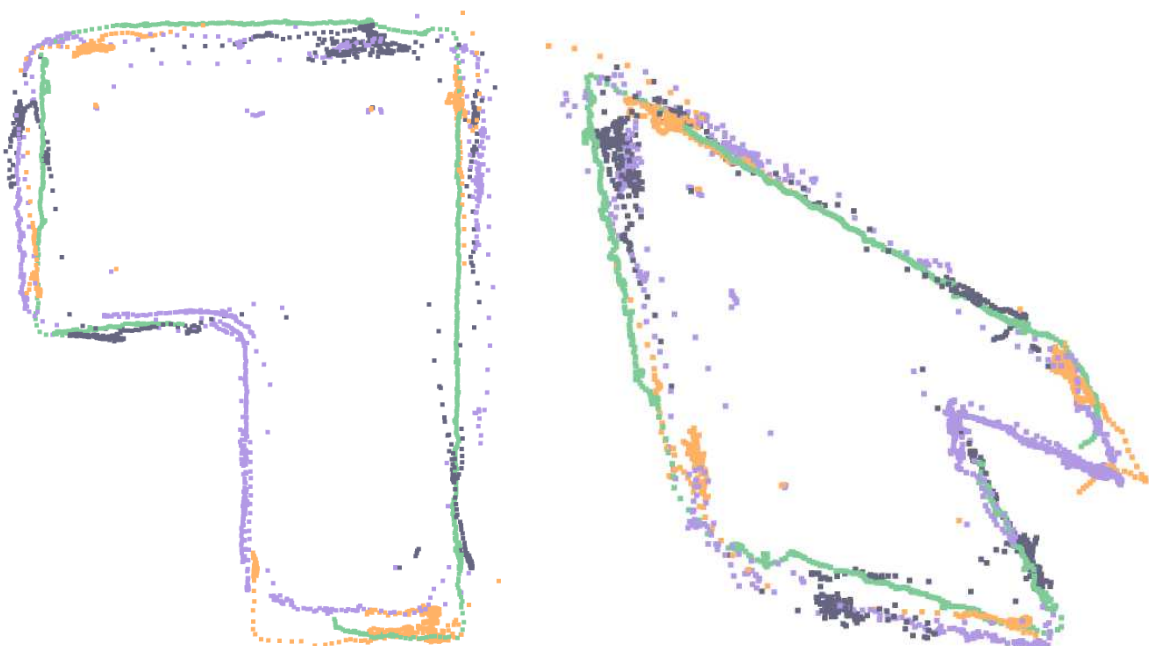


Source: The author.

4.3.2 Wall Following – Scenario 2

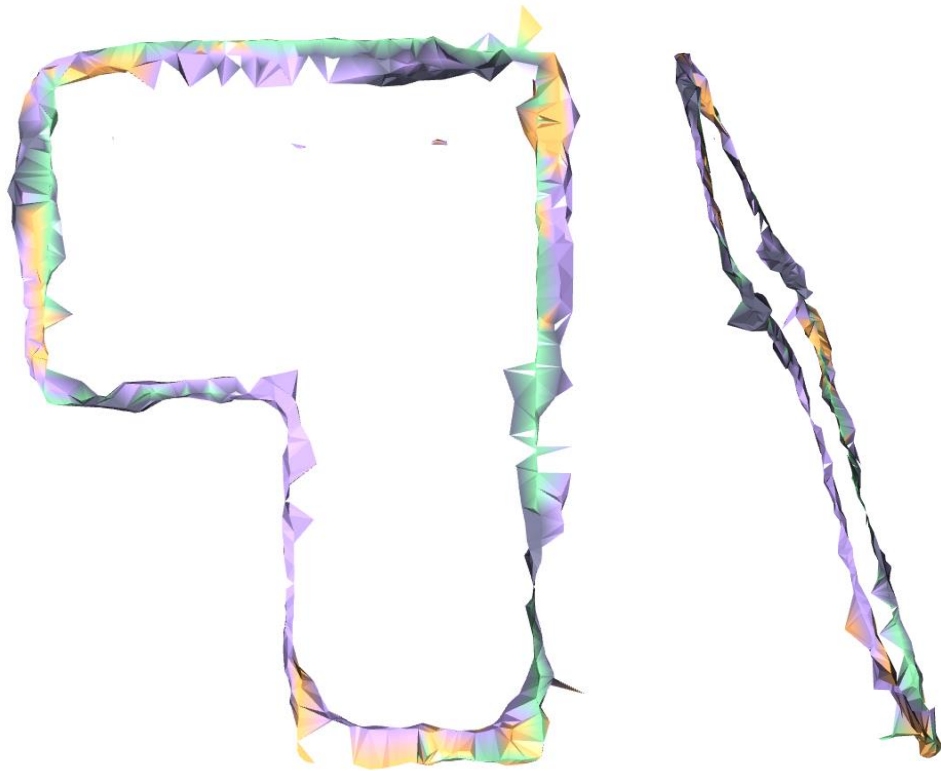
The behavior is autonomous in the wall following process, so setting the trajectory commands is nonessential. The necessary preliminary information is also available in Table 4.2. Figures 4.17 to 4.20 illustrate the results for Scenario 2.

Figure 4.17 – Open3D Point Cloud in Scenario 2 Using Wall Following.



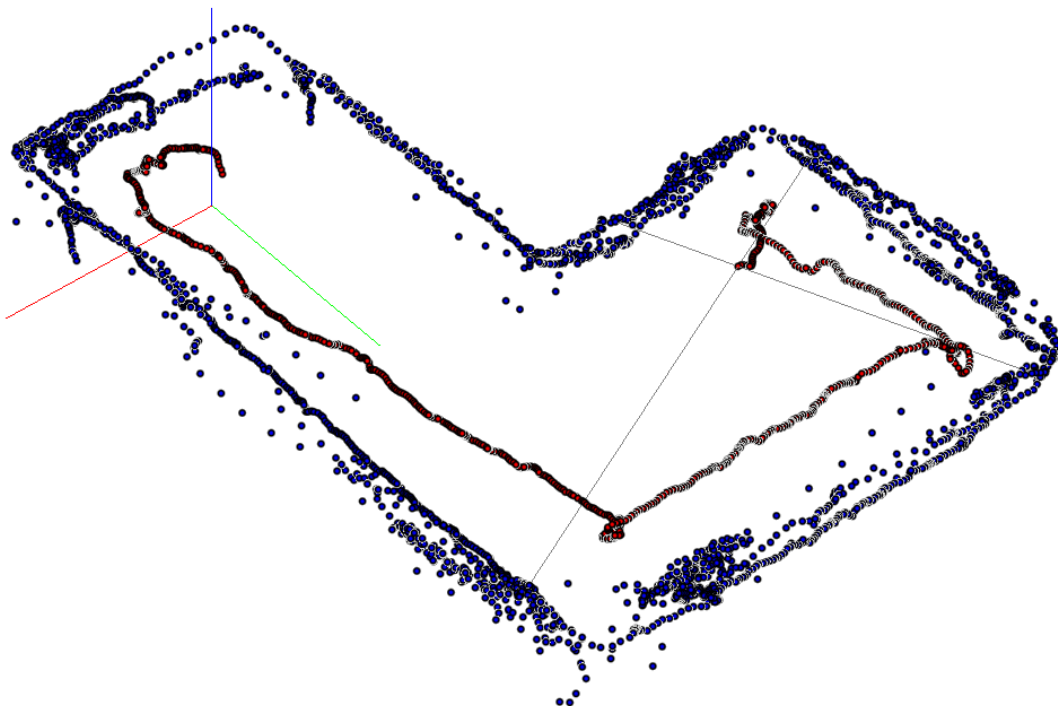
Source: The author.

Figure 4.18 – Open3D Surface in Scenario 2 Using Wall Following.



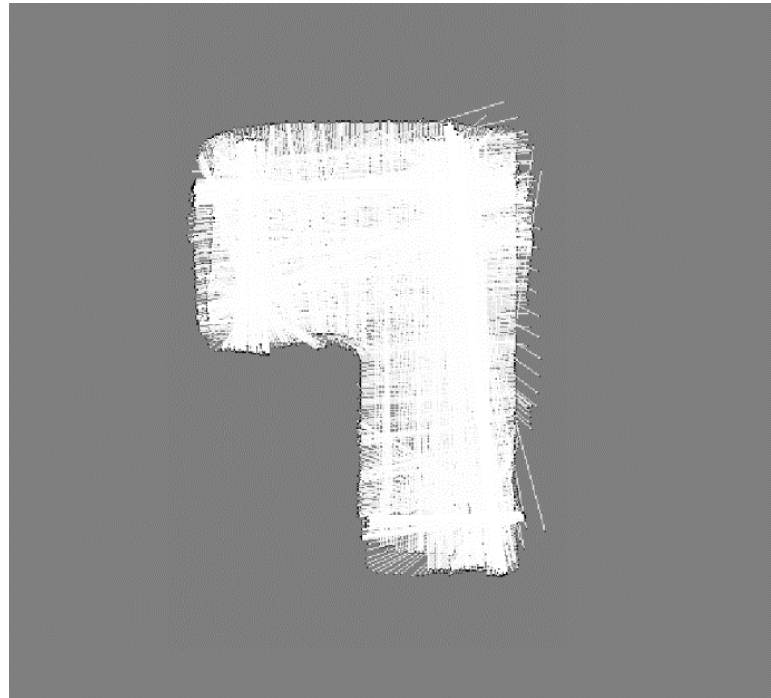
Source: The author.

Figure 4.19 – VisPy Point Cloud in Scenario 2 Using Wall Following.



Source: The author.

Figure 4.20 – Occupancy Grid in Scenario 2 Using Wall Following.



Source: The author.

As previously mentioned, in *Section 4.7*, the final considerations will include some comparative information regarding the occupancy grid map (2D map) and the other 3D maps (for both trajectories).

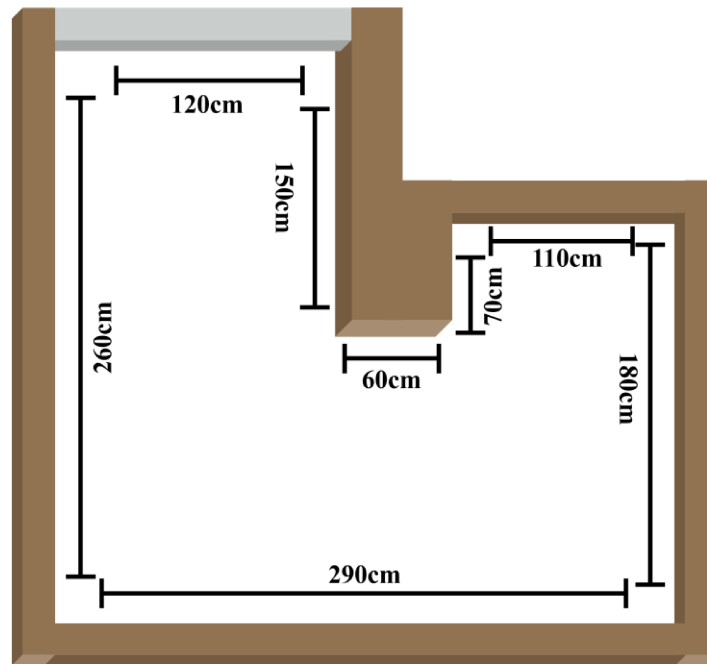
4.4. Scenario 3

The third scenario is an adaptation of the second scenario. Scenario 3 also has a more rectangular shape, but it has an extra corridor, created with the aim of having a map with more geometric shapes, with the drone dealing with asymmetrical spaces.

The process for building this scenario was the same as described in Scenarios 1 and 2. The first step is setting up the environment using the laboratory space available in block 1C at the Federal University of Uberlândia. The environment used the wall of the room, and the cardboard was positioned to form the other walls.

Figure 4.21 shows the architecture of the environment with its measurements. Figure 4.22 is a photo taken of the second scenario. The following sections will show the results of each technique for this environment. *Sections 4.4.1* and *4.4.2* will present and discuss the results of each SLAM technique.

Figure 4.21 – Architecture of the Scenario 3.



Source: The author.

Figure 4.22 – Photo of the Scenario 3.



Source: The author.

4.4.1 Manual Path – Scenario 3

As earlier mentioned, the commands were predetermined for the experiment, represented in Table 4.6, also it was considered the settings from Table 4.2 (*Section 4.1*). The drone pursues a sequence of forward, sleep, turn left, and sleep. The drone landed at a velocity

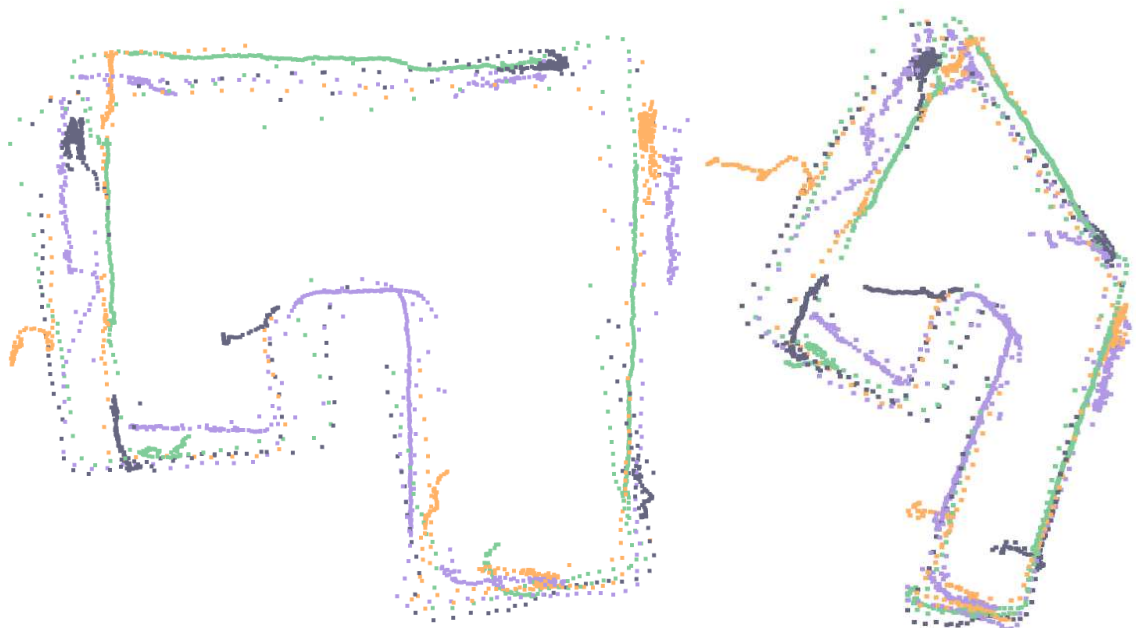
of 0.1 m/s. The command sleep (for one second = 1 s) is not in Table 4.6, but it does exist in the code, as the sequence explained. The distance for the "forward" differs due to the distance of the environment. Despite being an environment containing more diverse geometric shapes, the number of commands is the same as in the previous scenario. Figures 4.23 to 4.26 show the manual path results.

Table 4.6 – Scenario 3 Trajectory MP.

Commander		
1:	Forward	0.2 m
2:	Turn Left	90°
3:	Forward	1.8 m
4:	Turn Left	90°
5:	Forward	2 m
6:	Turn Left	90°
7:	Forward	0.9 m
8:	Turn Left	90°
9:	Forward	0.1 m
10:	Turn Left	90°
11:	Land	0.1 m/s

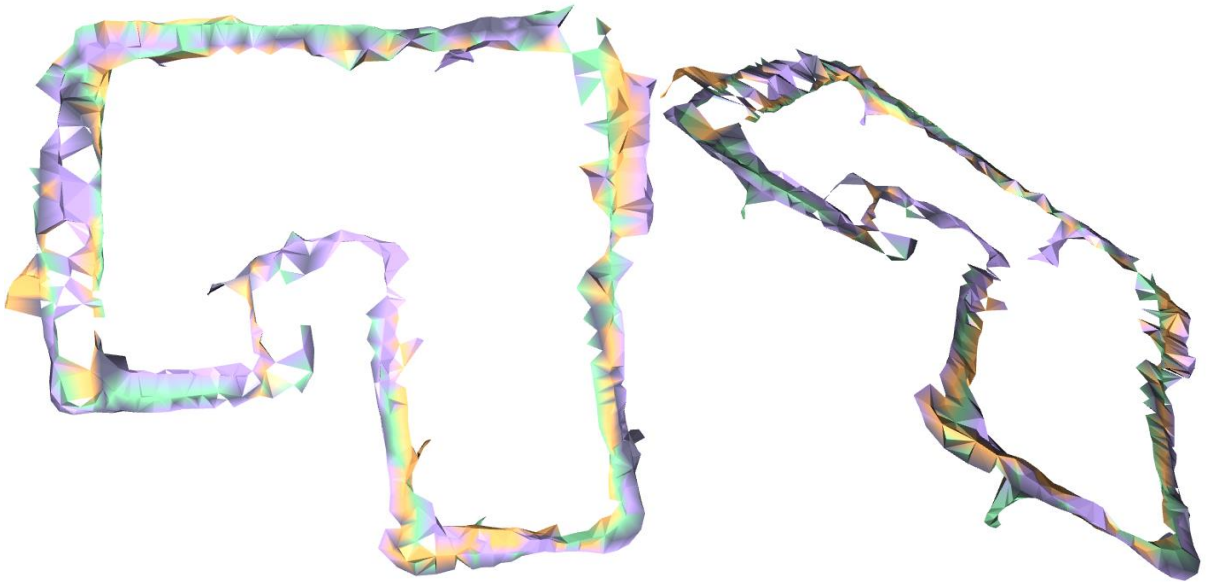
Source: The author.

Figure 4.23 – Open3D Point Cloud in Scenario 3 Using Manual Path.



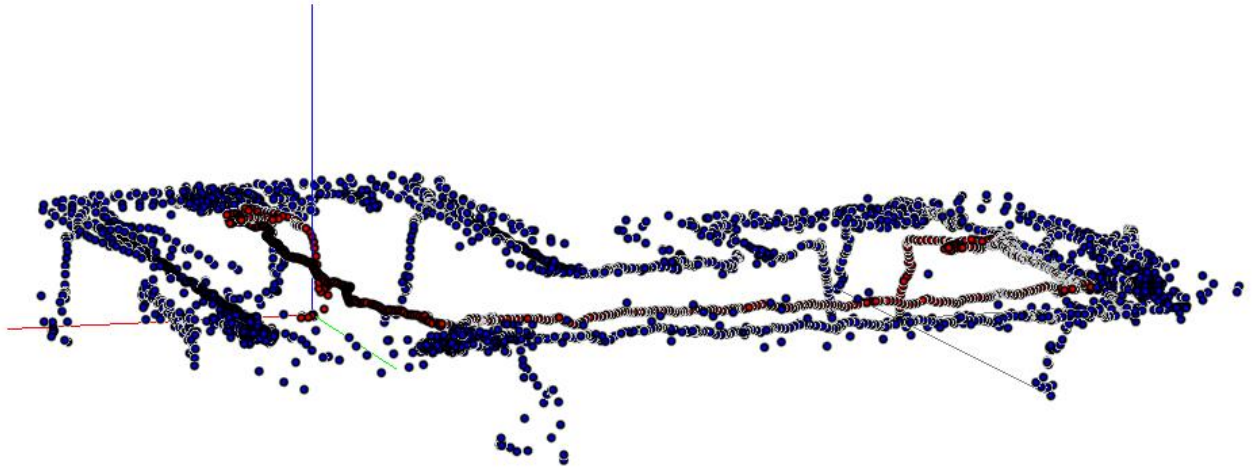
Source: The author.

Figure 4.24 – Open3D Surface in Scenario 3 Using Manual Path.



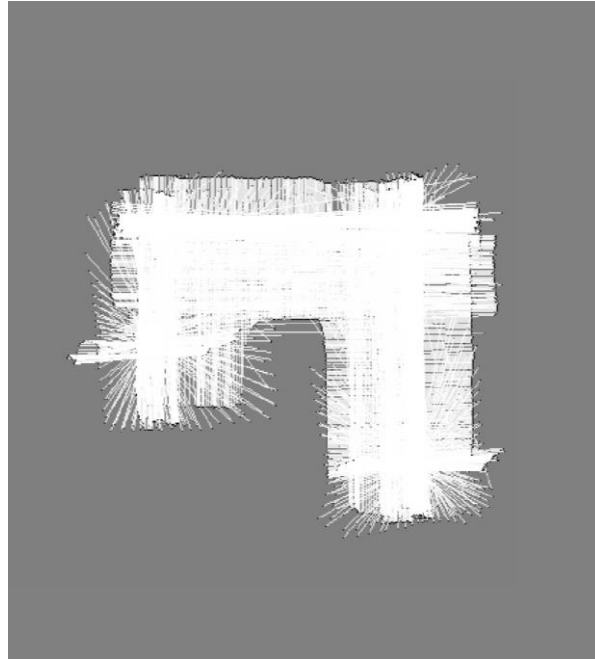
Source: The author.

Figure 4.25 – VisPy Point Cloud in Scenario 3 Using Manual Path.



Source: The author.

Figure 4.26 – Occupancy Grid in Scenario 3 Using Manual Path.

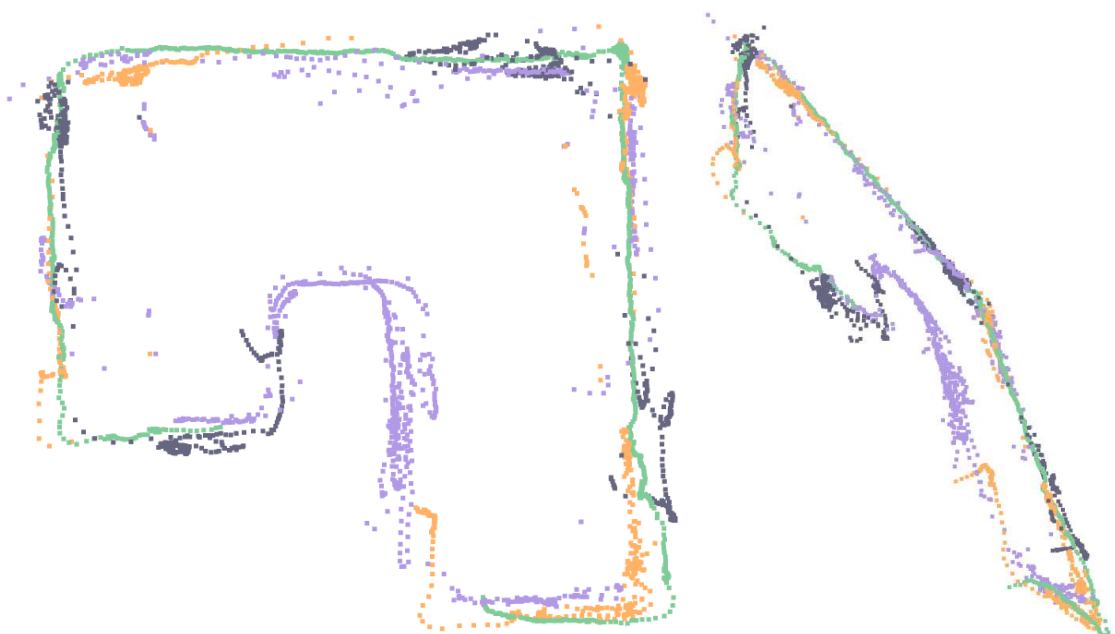


Source: The author.

4.4.2 Wall Following – Scenario 3

Figures 4.27 to 4.30 show the wall following, in other words, the nano quadcopter's behavior and path results. For a better visualization, some of the Figures have the same plot, despite the two different ways of positioning the image.

Figure 4.27 – Open3D Point Cloud in Scenario 3 Using Wall Following.



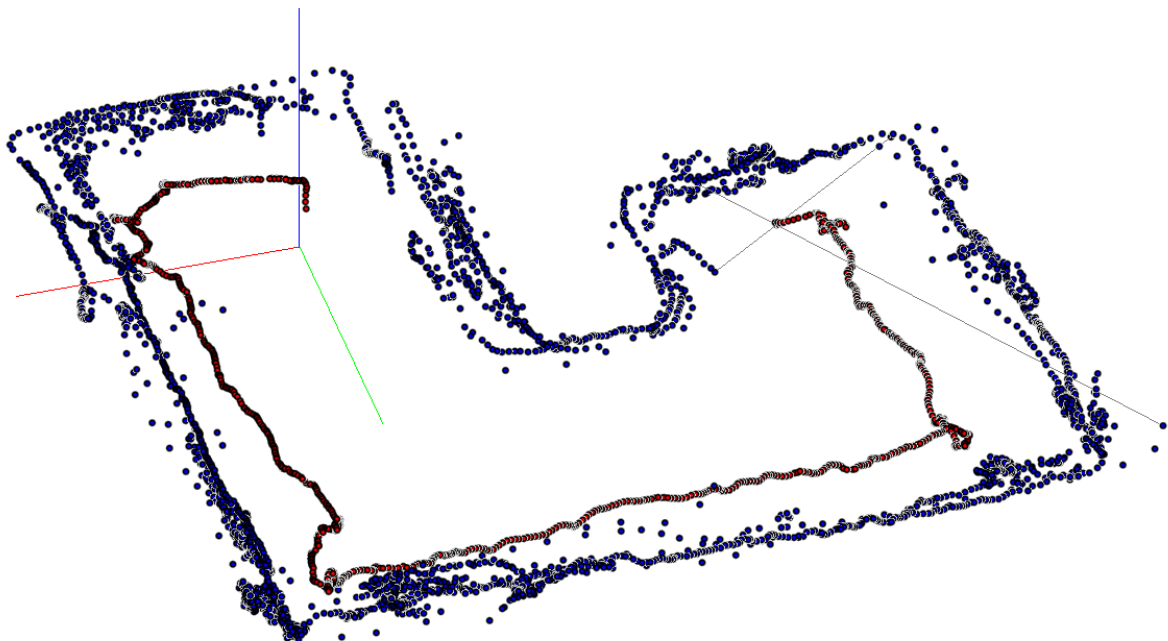
Source: The author.

Figure 4.28 – Open3D Surface in Scenario 3 Using Wall Following.



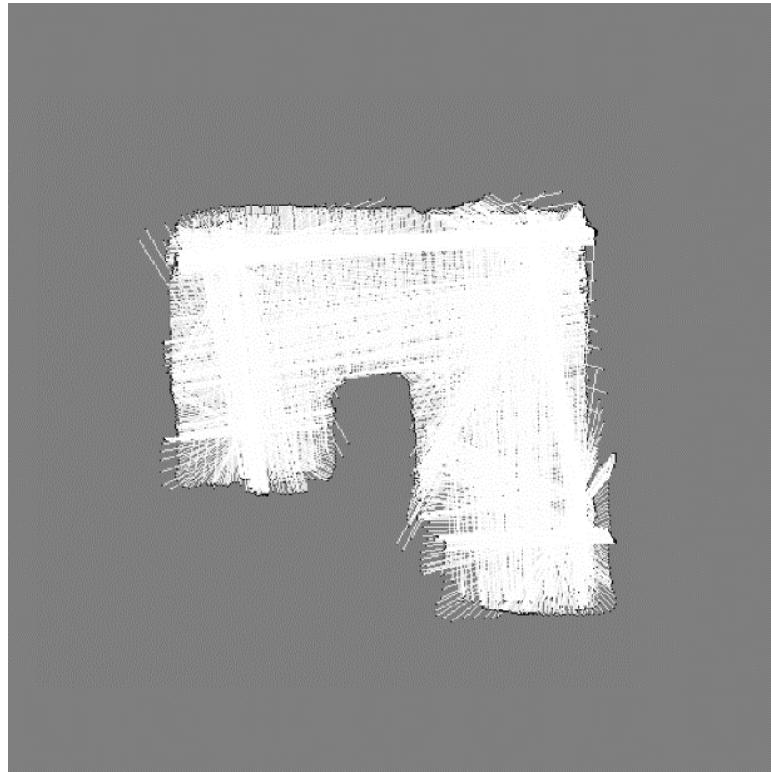
Source: The author.

Figure 4.29 – VisPy Point Cloud in Scenario 3 Using Wall Following.



Source: The author.

Figure 4.30 – Occupancy Grid in Scenario 3 Using Wall Following.



Source: The author.

As mentioned above, in *Section 4.7*, the final considerations will include some comparative information regarding the occupancy grid map (2D map) and the other 3D maps - both trajectories.

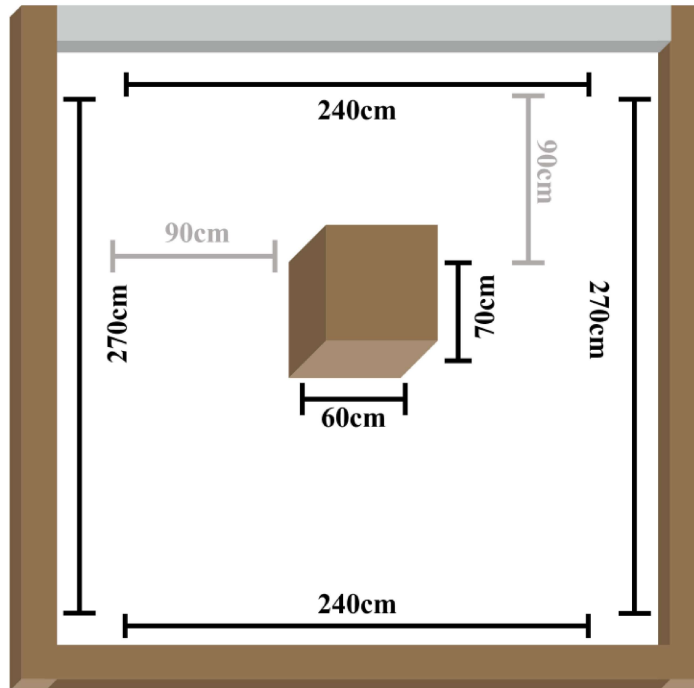
4.5. Scenario 4

The fourth scenario, and the last in this sequence, is an adaptation of the first scenario. Scenario 4 has a square shape but has an obstacle in the middle, which is a significant difference, apart from the different measurements.

This scenario is intended to have map results considering a static obstacle in the center. The process for building this scenario was the same as in the previous scenarios. The environment was set up in the laboratory space available in block 1C at the Federal University of Uberlândia. The environment used the wall of the room, and the cardboard was positioned to form the other walls. Figure 4.31 shows the architecture of the environment with its measurements. Figure 4.32 is a photo taken of the second scenario.

In this way, the following sections will show the results of each technique for this environment. *Sections 4.5.1* and *4.5.2* will present and discuss the results of each SLAM technique.

Figure 4.31 – Architecture of the Scenario 4.



Source: The author.

Figure 4.32 – Photo of the Scenario 4.



Source: The author.

4.5.1 Manual Path – Scenario 4

As previously mentioned, the commands were predetermined for the experiment, represented in Table 4.7, also it was considered the settings from Table 4.2 (*Section 4.1*).

The nano quadcopter pursues a sequence of forward, sleep, turn left, and sleep. The nano quadcopter landed at a velocity of 0.1 m/s. The starting point affects the nano quadcopter, so for two sides (first and fifth command), the distance covered differs from one to another. The command sleep (for one second = 1 s) is not in Table 4.7, but it does exist in the code, as the sequence explained.

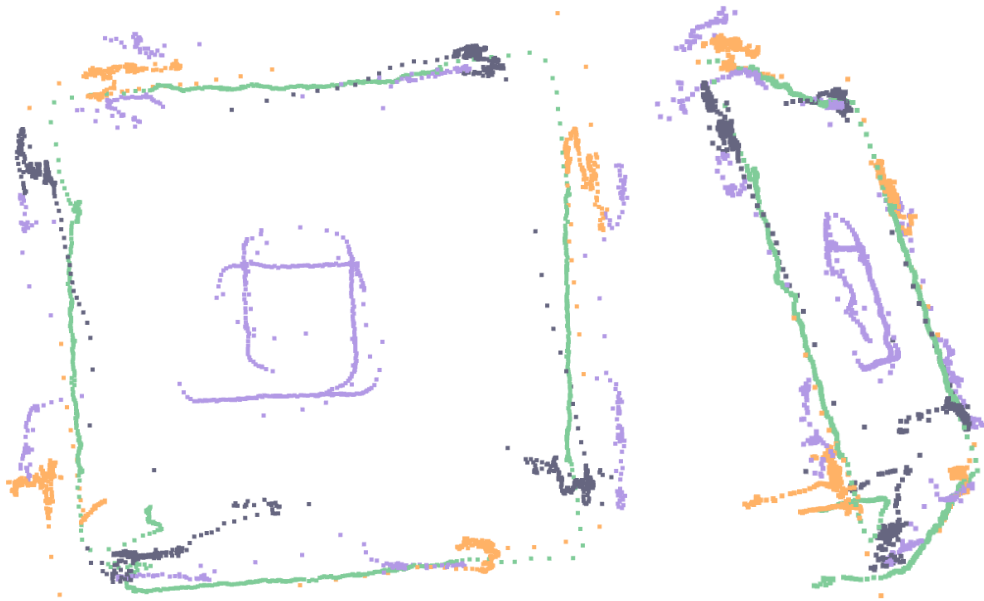
Table 4.7 – Scenario 4 Trajectory MP.

Commander		
1:	Forward	1.7 m
2:	Turn Left	90°
3:	Forward	1.35 m
4:	Turn Left	90°
5:	Forward	1.45 m
6:	Turn Left	90°
7:	Forward	1.35 m
8:	Turn Left	90°
9:	Land	0.1 m/s

Source: The author.

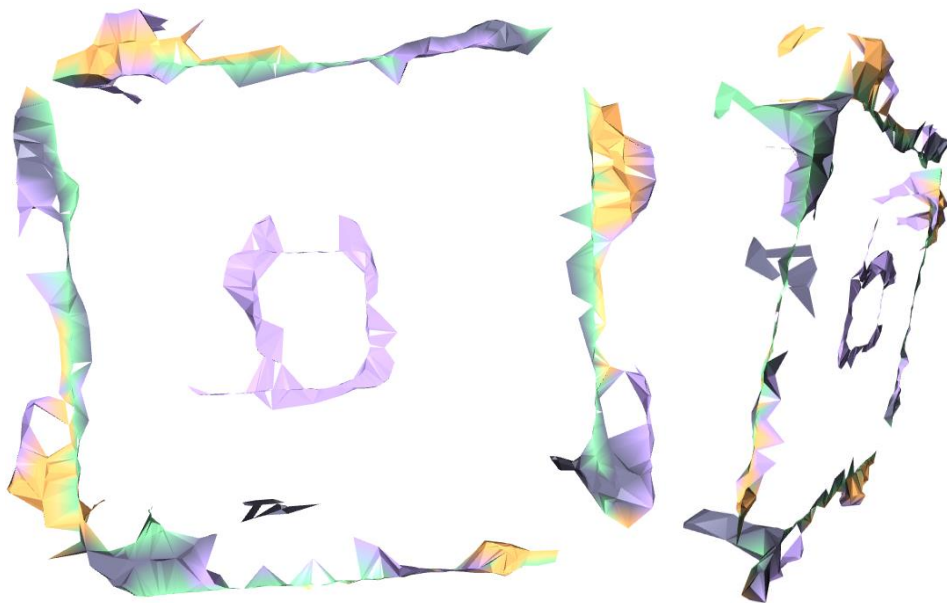
Figures 4.33 to 4.36 show the manual path results, in other words, the nano quadcopter's behavior and path results. For a better visualization, some of the Figures have the same plot, despite the two different ways of positioning the image.

Figure 4.33 – Open3D Point Cloud in Scenario 4 Using Manual Path.



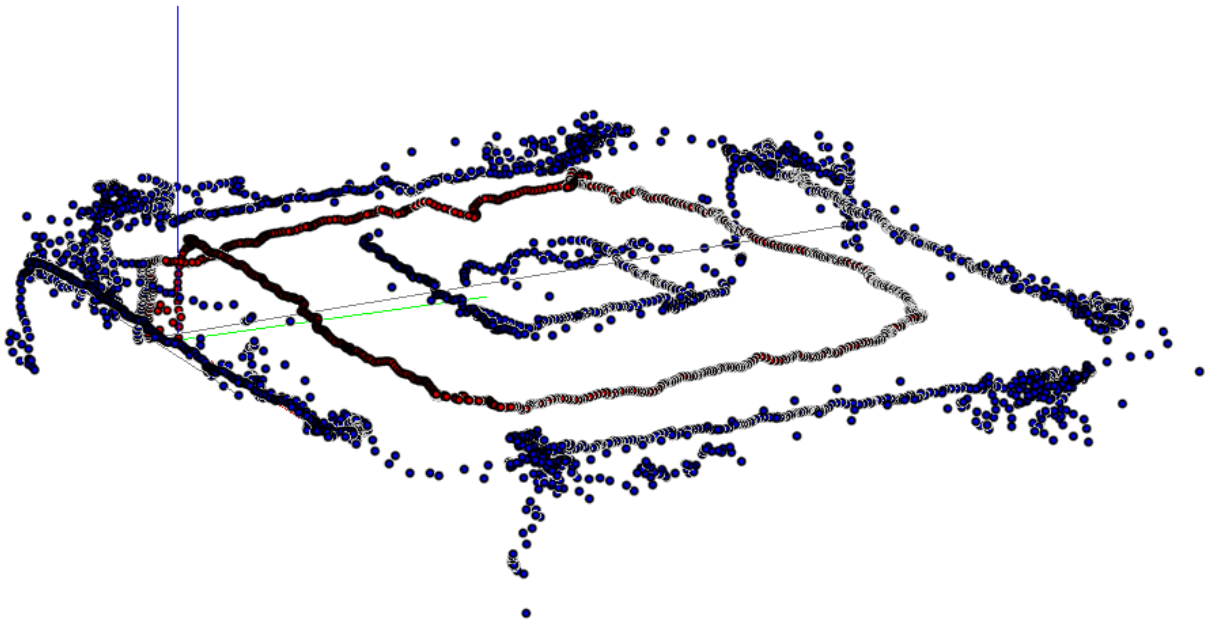
Source: The author.

Figure 4.34 – Open3D Surface in Scenario 4 Using Manual Path.



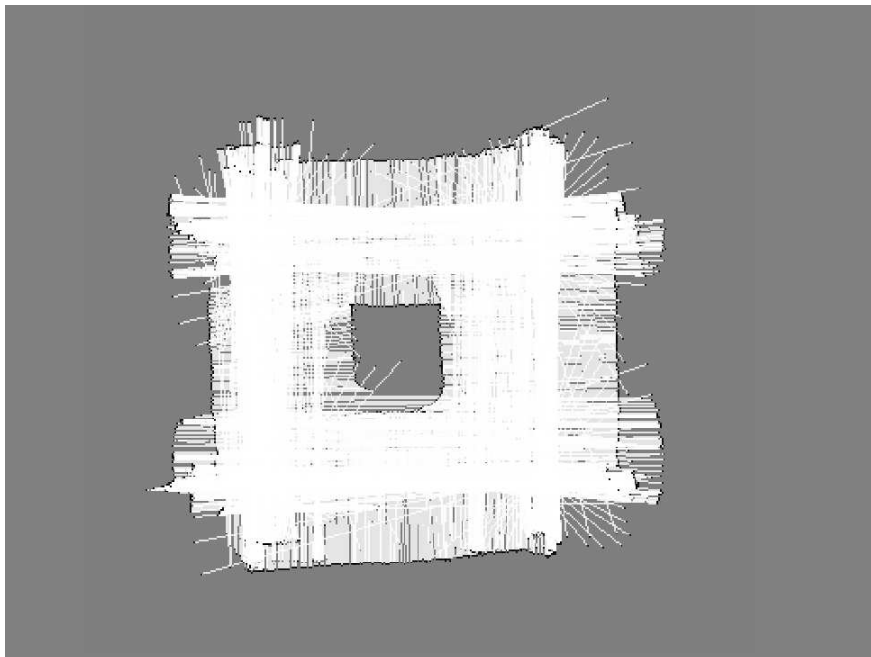
Source: The author.

Figure 4.35 – VisPy Point Cloud in Scenario 4 Using Manual Path.



Source: The author.

Figure 4.36 – Occupancy Grid in Scenario 4 Using Manual Path.

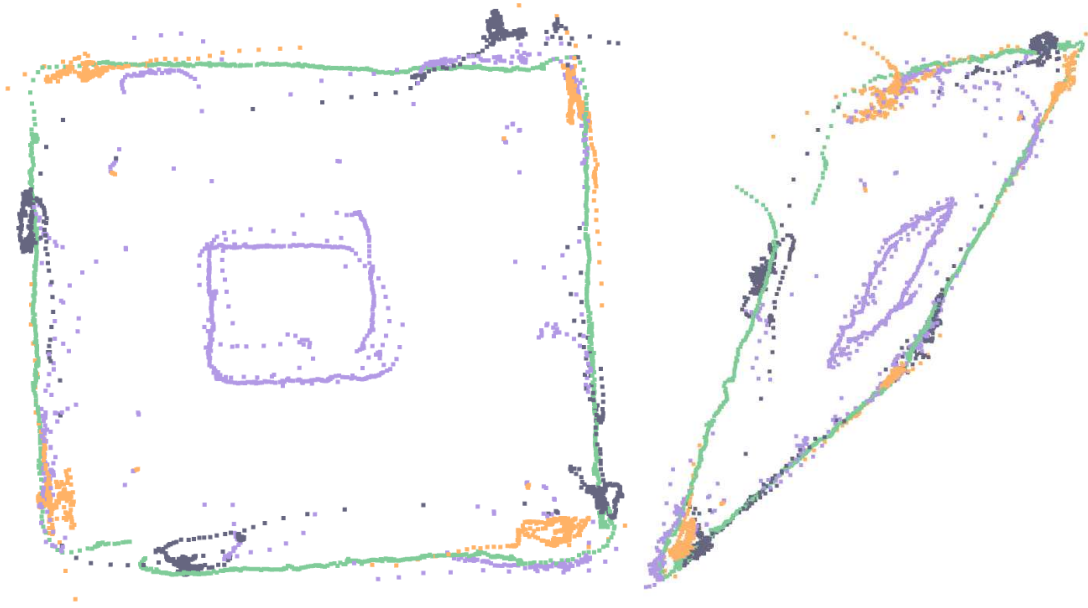


Source: The author.

4.5.2 Wall Following – Scenario 4

Figures 4.37 to 4.40 show the wall following, in other words, the nano quadcopter's behavior and path results. For a better visualization, some of the Figures have the same plot, despite the two different ways of positioning the image.

Figure 4.37 – Open3D Point Cloud in Scenario 4 Using Wall Following.



Source: The author.

Figure 4.38 – Open3D Surface in Scenario 4 Using Wall Following.



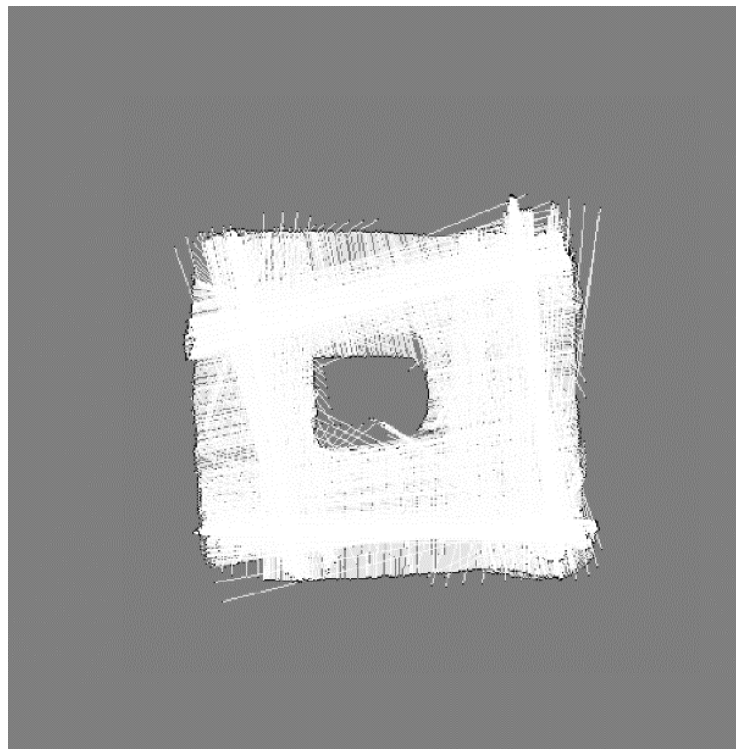
Source: The author.

Figure 4.39 – VisPy Point Cloud in Scenario 4 Using Wall Following.



Source: The author.

Figure 4.40 – Occupancy Grid in Scenario 4 Using Wall Following.



Source: The author.

As previously mentioned, in *Section 4.7*, the final considerations will include some comparative information regarding the occupancy grid map (2D map) and the other 3D maps (for both trajectories).

4.6. Individual Cases

In this section, two individual cases will be presented. These cases are specific experiments that have been discussed and contribute with extra analysis. These cases are called Levels (*Section 4.6.1*) and Real-Time Visualization (*Section 4.6.2*).

4.6.1 Levels

Figure 4.41 illustrates the photo taken of the Levels scenario, and in Table 4.8 are the parameters of this scenario. For the results, only Open 3D Surface and VisPy Point Cloud were used, as seen in Figure 4.42 and Figure 4.43.

For this case, the drone is positioned in the center of an area with similarity to a square, takes off to an altitude of 30 cm, performs a gradual 360° rotation, climbs another 20 cm, performs a 360° rotation, goes up more 20cm and performs a 360° rotation, and finally lands at a speed of 0.1 m/s to the ground. Therefore, the manual path was used.

Figure 4.41 – Photo of the Levels Scenario.



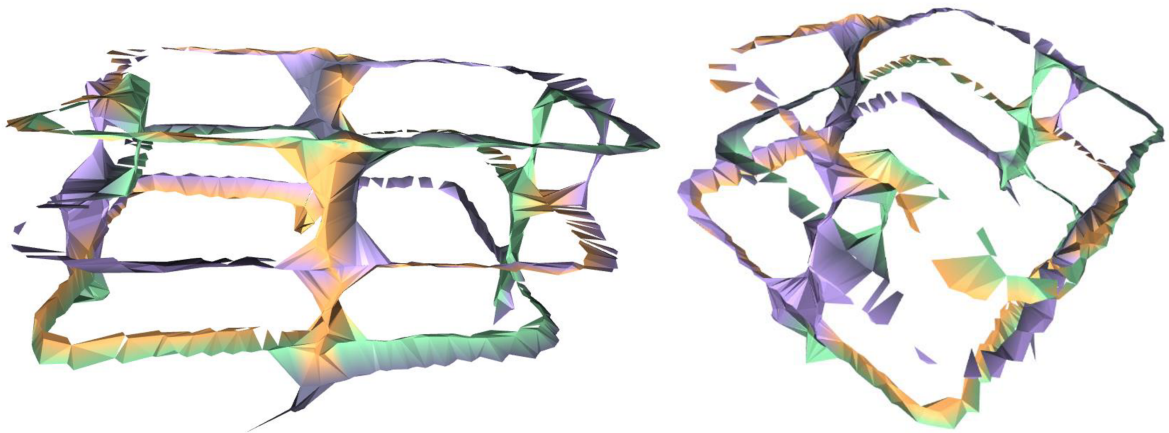
Source: The author.

Table 4.8 – Parameters of the Levels Scenario.

Parameters	Scenario
Height	0.9 m
Width	1.5 m
Length	1.8 m

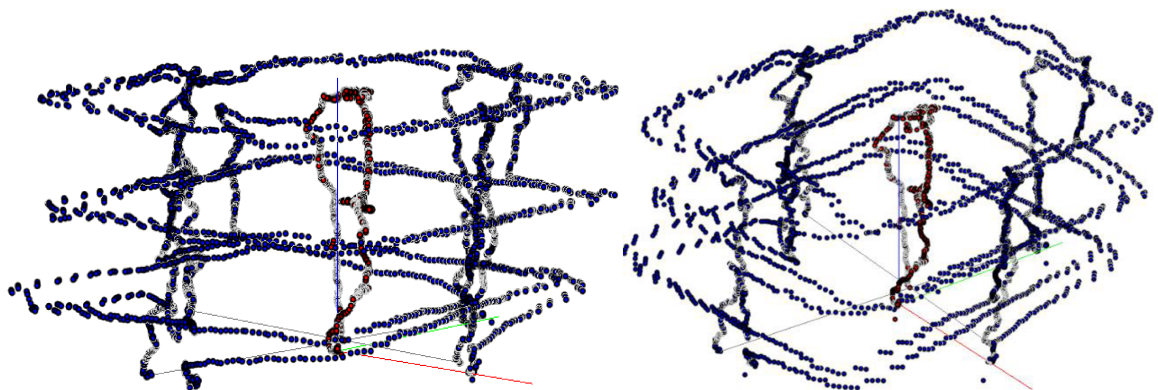
Source: The author.

Figure 4.42 – Open3D Surface in Levels Scenario.



Source: The author.

Figure 4.43 – Open3D Surface in Levels Scenario.



Source: The author.

This experiment aimed to evaluate the system's performance in the third dimension across varying levels of data collection. The surface method was selected to observe to see if the experiment could generate a flat surface by reconstructing the walls with the different levels of measurements collected. Unfortunately, it was reconstructed only on the level surface. By

the point cloud method (using VisPy), it is possible to observe the trajectory of the nano quadcopter.

4.6.2 Real-Time Visualization

Another major challenge of using SLAM lies in its real-time visualization. This challenge was carried out in a simple scenario, the same scenario described in Scenario 1, in an environment in a simple square shape. Figure 4.44 illustrates the Scenario Real-Time Visualization representation as can be seen by the user.

Figure 4.44 – Real-Time Visualization Scenario.



Source: The author.

For the Real-Time Visualization Scenario, it was developed a stream abstraction to avoid heavy work on the LogConfig callback. The base stream registers the LogConfig to receive data from the flow deck and multiranger deck.

Once the complete measurements are received, with both the nano quadcopter's pose and all four sensor measurements, the data is flushed down the stream to subscribers. The stream accepts multiple subscribers, which are notified on each measurement, meaning that the actions could happen independently of each other and avoid blocking.

For example, after flushing the measurement, there is a buffer registered in the stream to later flush the measurement to disk. On the other hand, another stream subscriber calculates

the point with the rotation matrix and draws it into a canvas for live preview. This setup gave a better separation of concerns, with each step in the stream being unaware of the existence of other subscribers. All of this utilizes stream abstraction. This design is based on the reactive approach (ReactiveX, [s. d.]), although in a much-simplified form.

4.7. Considerations

Generally, the maps for the environments exhibit that the manual path has more distributed and even measurements that seem like random points, and as a consequence, it has more uncertainties. However, the wall following has more accurate and accumulated points, which can indicate fewer uncertainties in the data collected.

Furthermore, for the MP, the surface method has more empty spaces presented in the figures. For the WF, the maps (in figures) indicate more satisfactory performance in surface reconstruction, with clear and intense maps and few gaps.

Considering the experimental data, the surface method used in both flight techniques is one way of manipulating the data. The reconstruction with more grouped data is much more perceptive, with a surface much better defined, which is the case with wall-following behavior.

For the occupancy grid map, the results using the wall following behavior are more promising in the first three environments (Scenario 1 to 3). In the fourth environment (Scenario 4), the WF is not performing well, though the manual path is even more unsatisfactory.

Table 4.9 shows the number of scans for building the 2D map respectively in manual path and wall following. The scan represents a total data set of the quadcopter's information (measurement and pose).

Table 4.9 – Number of Scans MP and WF (2D Map).

Scenario	Scan MP	Scan WF
1	639	705
2	637	798
3	774	896
4	749	952

Source: The author.

For the maps of Scenarios 1 to 4, it is observed that for the MP, the occupancy grid plots show this accumulation, where the plot gets more white lines where the drone flew, whereas others still have some small gray area. The corners of the walls in the occupancy grid highlight this behavior. The corners have a few white lines, but it is mostly gray. As the drone knows how many degrees to turn, it quickly turns the necessary degrees, causing less information to be collected.

In contrast, since the wall following utilizes the most recent reading to take action, it covers more of the environment. Take the wall corners as an example again. The drone needs to have the measurements to adjust itself to the wall. The inherent way the algorithm works allows it to collect more details about the environment. This adaptive behavior of the wall following algorithm leads to more detailed plots.

For the 3D maps, point cloud is another evidence of this behavior. The points collected by the front and back sensors of the multi-ranger have clusters where the drone flights. There is even less information collected at the corners when the drone makes a turn. In contrast with the point cloud from the wall following data, the plot has many more points distributed across the environment. And, as the drone needs to adjust at the wall corners, the corners have a much denser quantity of points.

The manual path provides the Crazyflie with a simplified version for experiments. The development of an application for experimenting is straightforward. However, the more rigid, the less responsive the nano quadcopter is, making the utilization unfit for developing an application for autonomous flight. The plots also evidence this behavior, where the occupancy grid is less detailed, the point cloud has a cluster of points where the drone flies, leaving other regions uncovered, and the surface reconstruction is incomplete.

By contrast, the wall-following behavior provides the Crazyflie with autonomous flight capability. The nano quadcopter crosses an unknown environment by following its walls. It takes action and adjusts its trajectory with measurements captured with the multi-ranger deck. The data collected also enables the construction of a very detailed occupancy grid map. The plots generated with the data successfully represent the forms and obstacles from the real-world environment. The point cloud has a more even distribution of the points, and the surface reconstruction has few gaps.

Furthermore, it can be seen that all the techniques are efficient, despite the aspects discussed above. Using Open3D there is manipulation of the data into another type of representation, and the VisPy is possible to easily insert the location and trajectory of the nano quadcopter into the representation (the red dots). Furthermore, it can be noted that in the

occupancy grid technique, many uncertainties are dealt with in the algorithm, giving an even better representation of the environment.

CONCLUSIONS

Regarding the aims of this project, the research purpose is to implement a system using a nano quadcopter with autonomous flight and laser sensors capable of simultaneous localization and mapping, with diverse techniques for solving SLAM, covering methods for reconstructing 2D and 3D maps, using an open-source system, with Python programming language. In addition to all the justifications mentioned above, the ambition is to help disseminate research and concepts in this complex and arduous area of engineering, control, and automation whilst still brilliant.

The methodology presented offers a new perspective for creating a practical nano quadcopter system, developed in a computational and scientific language, using open-source materials and available for general access.

The implementation of the nano quadcopter was operating well with the two different decks, with Cazyflyie 2.1, Multi-ranger, and Flow-Deck v2, and all the methods solutions chosen for the SLAM problem are working well, operating satisfactory, and running efficiently, with their respective characteristics, performances, qualities, and issues for improvement. Considering the results, accomplishing all the objectives of the project with success.

The data collected through the persistence layer allows different visualization techniques. The current work developed a 3D visualization of the environment utilizing point cloud maps and a 2D visualization with the occupancy grid map. Since the persistence layers offer the durability necessary, it is possible to manipulate the data without requiring more flights. The surface reconstruction is an example of the experimentation possible because of the persistence layer.

Therefore, the experimental results present that the nano quadcopter system can successfully map the environment and identify obstacles in its path, as well. However, there are still improvements to make since the mapping has many out-of-bounds points and irregular curves. The current work conducted several real-world experimental tests to ensure the system's robustness. These tests utilize different techniques for flying the nano quadcopter.

One of the techniques uses a predefined route through the environment, avoiding obstacles. As such, this technique requires *a priori* knowledge of the environment. The development utilizes the basic movement primitives offered by the `cflib` Python library. The development is more straightforward and restricted, a good fit for experimenting with the Crazyflie.

A second, more complete technique, called wall following, enables the Crazyflie to fly in an unknown environment autonomously. The drone takes measurements utilizing the extension decks to adjust its trajectory and fly parallel to a wall. Inherently, this technique requires more information about the environment before taking action. The current work implements the wall following algorithm in a highly responsive, non-blocking application capable of gracefully terminating without losing data.

The current work verified the different scenarios with both techniques, collecting the data and generating the plots for a qualitative visual analysis. Overall, the wall following resulted in more detailed plots compared to the manual path. The measurements are more distributed in the environment since the algorithm requires the data collection to move.

On the other hand, the manual path has a defined path, so it can fly directly without requiring more information. The data collected by the manual path ends up being denser in the region where the drone flies, while others are not fully covered.

As previously described and noted, all map representations are uncertain since the pose and the location (relative position) have uncertainty, and so do the sensor readings.

To conclude, the current work also experiments with a live visualization of the drone flying with the occupancy grid map. This approach has a restriction that the map size needs to be defined *a priori* to construct the grid. A size larger than the real (actual map) means expending computational resources. A size smaller than the actual map world means not mapping some parts of the environment. A more advanced and robust implementation could apply computer graphics algorithms to expand the grid on demand.

5.1. Future Work

The current work still has some room for improvement. There are different lines of future work to improve. There is the data analysis part. There is room for developing tools to export the data to other visualization or scientific tools, such as MATLAB. The development of a single tool with all the visualization techniques utilized. On the other hand, it is also

possible to employ the framework of the current work for persistence and visualization to apply different SLAM techniques.

The principal improvement is the buffered output keeping all the measurements in memory, and it could completely interfere with autonomous flight over long periods. The ideal solution would batch the writes to disk, delegating each write to another thread and avoiding any blocking operations on the callbacks.

Additional improvements could be applied to the data post-processing. Currently, the persisted data does not follow any industry standard in the file format. It is a proprietary format design for the current work. An improvement would be to create a converter to other accepted formats, and this would open room to analyze the data more easily in other software.

One of the possibilities for solving map optimization would be the use of multiple robots. One of these approaches was addressed in recent work (Friess et al., 2023) using a swarm of nano quadcopters (Crazyflie 2.1), which could be a possibility for future work.

It can be observed that the difference in the measurements of each sensor directly influences the point values obtained. In addition, the current work does not have a closed loop, and the algorithm does not know that it has already visited such a place in the environment. According to (Taheri; Xia, 2021), loop closure would be a solution to this, as well as being able to optimize the robot's pose better and reduce drifts and odometry errors.

For future work, besides improving the points observed, there is the need to implement the mapping with a simulation in a ROS environment and decision-making on the path, and another possibility of improvement is the GUI (Graphical User Interface) implementation.

Bibliography

REFERENCES

AHMED, F. *et al.* Recent Advances in Unmanned Aerial Vehicles: A Review. *Arabian Journal for Science and Engineering*, [s. l.], v. 47, n. 7, p. 7963–7984, 2022. <https://doi.org/10.1007/s13369-022-06738-0>

ALTINPINAR, O. V. *et al.* Comparison of Autonomous Robot's Mapping Performance Based on Number of Lidars And Number of Tours. *In: 2022 INNOVATIONS IN INTELLIGENT SYSTEMS AND APPLICATIONS CONFERENCE (ASYU)*, 2022, Antalya, Turkey. 2022 Innovations in Intelligent Systems and Applications Conference (ASYU). Antalya, Turkey: IEEE, 2022. p. 1–6. Available in: <https://ieeexplore.ieee.org/document/9925444/>. <https://doi.org/10.1109/ASYU56188.2022.9925444>

AULINAS, J. *et al.* The SLAM problem: a survey. *Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*, [s. l.], p. Pages 363-371, 2008. <http://dx.doi.org/10.3233/978-1-58603-925-7-363>

BALESTRIERI, E. *et al.* Sensors and Measurements for UAV Safety: An Overview. *Sensors*, [s. l.], v. 21, n. 24, p. 8253, 2021. <https://doi.org/10.3390/s21248253>

BAVLE, H. *et al.* From SLAM to Situational Awareness: Challenges and Survey. *Sensors*, [s. l.], v. 23, n. 10, p. 4849, 2023. <https://doi.org/10.3390/s23104849>

BAVLE, H. *et al.* VPS-SLAM: Visual Planar Semantic SLAM for Aerial Robotic Systems. *IEEE Access*, [s. l.], v. 8, p. 60704–60718, 2020. <https://doi.org/10.1109/ACCESS.2020.2983121>

BITCRAZE, A. Crazyflie 2.1 | Bitcraze. [S. l.], 2022a. Available in: <https://www.bitcraze.io/products/crazyflie-2-1/> . .

BITCRAZE, A. Flow deck v2 | Bitcraze. [S. l.], 2022b. Available in: <https://www.bitcraze.io/products/flow-deck-v2/>.

BITCRAZE, A. Go with the Flow: Relative Positioning with the Flow deck. [S. l.], 2023. Available in: <https://www.bitcraze.io/2023/11/go-with-the-flow-relative-positioning-with-the-flow-deck/>.

BITCRAZE, A. Multi-ranger deck | Bitcraze. [S. l.], 2022c. Available in: <https://www.bitcraze.io/products/multi-ranger-deck/>.

BITCRAZE, A. State Estimation. [S. l.], 2022d. Available in: https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/functional-areas/sensor-to-control/state_estimators/#flowdeck-measurement-model.

BRESENHAM, J. E. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, [s. l.], v. 4, n. 1, p. 25–30, 1965. <https://doi.org/10.1147/sj.41.0025>

CAMPAGNOLA, L. *et al.* Vispy/vispy: Version 0.14.1. Versão v0.14.1. [S. l.]: [object Object], 2023. Available in: <https://zenodo.org/record/592490>.

CHAN, S.-H.; WU, P.-T.; FU, L.-C. Robust 2D Indoor Localization Through Laser SLAM and Visual SLAM Fusion. *In: 2018 IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN, AND CYBERNETICS (SMC), 2018, Miyazaki, Japan. 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Miyazaki, Japan: IEEE, 2018. p. 1263–1268.* Available in: <https://ieeexplore.ieee.org/document/8616217/>. <https://doi.org/10.1109/SMC.2018.00221>

COPPOLA, M. *et al.* A Survey on Swarming With Micro Air Vehicles: Fundamental Challenges and Constraints. *Frontiers in Robotics and AI*, [s. l.], v. 7, p. 18, 2020. <https://doi.org/10.3389/frobt.2020.00018>

COVOLAN, J. P. M.; SEMENTILLE, A. C.; SANCHES, S. R. R. A mapping of visual SLAM algorithms and their applications in augmented reality. *In: 2020 22ND SYMPOSIUM ON VIRTUAL AND AUGMENTED REALITY (SVR), 2020, Porto de Galinhas, Brazil. 2020 22nd Symposium on Virtual and Augmented Reality (SVR). Porto de Galinhas, Brazil: IEEE, 2020. p. 20–29.* Available in: <https://ieeexplore.ieee.org/document/9262630/>. <https://doi.org/10.1109/SVR51698.2020.00019>

DAI, Y.; WU, J.; WANG, D. A Review of Common Techniques for Visual Simultaneous Localization and Mapping. *Journal of Robotics*, [s. l.], v. 2023, p. 1–21, 2023. <https://doi.org/10.1155/2023/8872822>

DURRANT-WHYTE, H. F. Uncertain geometry in robotics. *IEEE Journal on Robotics and Automation*, [s. l.], v. 4, n. 1, p. 23–31, 1988. <https://doi.org/10.1109/56.768>

DURRANT-WHYTE, H.; RYE, D.; NEBOT, E. Localization of Autonomous Guided Vehicles. *In: GIRALT, G.; HIRZINGER, G. (org.). Robotics Research*. London: Springer London, 1996. p. 613–625. Available in: http://link.springer.com/10.1007/978-1-4471-1021-7_69.

FINK, G. *et al.* Observer design for visual inertial SLAM scale on a quadrotor UAV. *In: 2017 INTERNATIONAL CONFERENCE ON UNMANNED AIRCRAFT SYSTEMS (ICUAS), 2017, Miami, FL, USA. 2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. Miami, FL, USA: IEEE, 2017. p. 830–839. Available in: <http://ieeexplore.ieee.org/document/7991497/>. <https://doi.org/10.1109/ICUAS.2017.7991497>

FÖRSTER, J. System Identification of the Crazyflie 2.0 Nano Quadcopter. [s. l.], p. 147 p., 2015. <https://doi.org/10.3929/ethz-b-000214143>

FRIESS, C. *et al.* Fully Onboard SLAM for Distributed Mapping with a Swarm of Nano-Drones. [S. l.]: arXiv, 2023. Available in: <http://arxiv.org/abs/2309.03678>.

GAIA, J. *et al.* Mapping the Landscape of SLAM Research: A Review. *IEEE Latin America Transactions*, [s. l.], v. 21, n. 12, p. 1313–1336, 2023. <https://doi.org/10.1109/TLA.2023.10305240>

GOLDSTEIN, H.; POOLE, C. P.; SAFKO, J. L. *Classical mechanics*. Third editioned. United States: Pearson, 2011.

GREIFF, M. *Modelling and Control of the Crazyflie Quadrotor for Aggressive and Autonomous Flight by Optical Flow Driven State Estimation*. 2017. 153 f. Dissertation - Lund University, Department of Automatic Control, 2017. Available: <https://lup.lub.lu.se/student-papers/search/publication/8905295>.

HARRIS, C. R. *et al.* Array programming with NumPy. *Nature*, [s. l.], v. 585, n. 7825, p. 357–362, 2020. <https://doi.org/10.1038/s41586-020-2649-2>

HÖNIG, W.; AYANIAN, N. Flying Multiple UAVs Using ROS. *In: KOU BAA, A. (org.). Robot Operating System (ROS)*. Cham: Springer International Publishing, 2017. (Studies in Computational Intelligence). v. 707, p. 83–118. Available in: http://link.springer.com/10.1007/978-3-319-54927-9_3. https://doi.org/10.1007/978-3-319-54927-9_3

HU, X.; ASSAAD, R. H. The use of unmanned ground vehicles (mobile robots) and unmanned aerial vehicles (drones) in the civil infrastructure asset management sector: Applications, robotic platforms, sensors, and algorithms. *Expert Systems with Applications*, [s. l.], v. 232, p. 120897, 2023. <https://doi.org/10.1016/j.eswa.2023.120897>

HUANG, L. Review on LiDAR-based SLAM Techniques. *In: 2021 INTERNATIONAL CONFERENCE ON SIGNAL PROCESSING AND MACHINE LEARNING (CONF-SPML)*, 2021, Stanford, CA, USA. 2021 International Conference on Signal Processing and Machine Learning (CONF-SPML). Stanford, CA, USA: IEEE, 2021. p. 163–168. Available in: <https://ieeexplore.ieee.org/document/9707054/>. <https://doi.org/10.1109/CONF-SPML54095.2021.00040>

HUNTER, J. D. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, [s. l.], v. 9, n. 3, p. 90–95, 2007. Available in: <https://ieeexplore.ieee.org/document/4160265>. <https://doi.org/10.1109/MCSE.2007.55>

HYNDMAN, R. J.; FAN, Y. Sample Quantiles in Statistical Packages. *The American Statistician*, [s. l.], v. 50, n. 4, p. 361–365, 1996. <https://doi.org/10.1080/00031305.1996.10473566>

JEONG, E. *et al.* Parsing Indoor Manhattan Scenes Using Four-Point LiDAR on a Micro UAV. *In: 2022 22ND INTERNATIONAL CONFERENCE ON CONTROL, AUTOMATION AND SYSTEMS (ICCAS)*, 2022, Jeju, Korea, Republic of. 2022 22nd International Conference on Control, Automation and Systems (ICCAS). Jeju, Korea, Republic of: IEEE, 2022. p. 708–713. Available in: <https://ieeexplore.ieee.org/document/10003969/>. <https://doi.org/10.23919/ICCAS55662.2022.10003969>

KARAM, S. *et al.* MICRO AND MACRO QUADCOPTER DRONES FOR INDOOR MAPPING TO SUPPORT DISASTER MANAGEMENT. *ISPRS Annals of the*

Photogrammetry, Remote Sensing and Spatial Information Sciences, [s. l.], v. V-1-2022, p. 203–210, 2022. <https://doi.org/10.5194/isprs-annals-V-1-2022-203-2022>

KARAM, Samer *et al.* Microdrone-Based Indoor Mapping with Graph SLAM. *Drones*, [s. l.], v. 6, n. 11, p. 352, 2022. <https://doi.org/10.3390/drones6110352>

KEFFERPUTZ, K.; MCGUIRE, K. Error-State Unscented Kalman-Filter for UAV Indoor Navigation. *In: 2022 25TH INTERNATIONAL CONFERENCE ON INFORMATION FUSION (FUSION), 2022, Linköping, Sweden. 2022 25th International Conference on Information Fusion (FUSION). Linköping, Sweden: IEEE, 2022. p. 01–08. Available in: https://ieeexplore.ieee.org/document/9841385/. https://doi.org/10.23919/FUSION49751.2022.9841385*

KHAN, M. U. *et al.* A Comparative Survey of LiDAR-SLAM and LiDAR based Sensor Technologies. *In: 2021 MOHAMMAD ALI JINNAH UNIVERSITY INTERNATIONAL CONFERENCE ON COMPUTING (MAJICC), 2021, Karachi, Pakistan. 2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC). Karachi, Pakistan: IEEE, 2021. p. 1–8. Available in: https://ieeexplore.ieee.org/document/9526266/. https://doi.org/10.1109/MAJICC53071.2021.9526266*

KUANG, Q. *et al.* Real-Time UAV Path Planning for Autonomous Urban Scene Reconstruction. *In: 2020 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA), 2020, Paris, France. 2020 IEEE International Conference on Robotics and Automation (ICRA). Paris, France: IEEE, 2020. p. 1156–1162. Available in: https://ieeexplore.ieee.org/document/9196558/. https://doi.org/10.1109/ICRA40945.2020.9196558*

LAI, T. A Review on Visual-SLAM: Advancements from Geometric Modelling to Learning-Based Semantic Scene Understanding Using Multi-Modal Sensor Fusion. *Sensors*, [s. l.], v. 22, n. 19, p. 7265, 2022. <https://doi.org/10.3390/s22197265>

LANDRY, B. Planning and Control for Quadrotor Flight through Cluttered Environments. 2015. - Massachusetts Institute of Technology, [s. l.], 2015. Available in: <http://hdl.handle.net/1721.1/100608>.

LEE, D. *et al.* Experiments on localization of an AUV using graph-based SLAM. *In: 2013 10TH INTERNATIONAL CONFERENCE ON UBIQUITOUS ROBOTS AND AMBIENT*

INTELLIGENCE (URAI), 2013, Jeju, Korea (South). 2013 10th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI). Jeju, Korea (South): IEEE, 2013. p. 526–527. Available in: <http://ieeexplore.ieee.org/document/6677329/>.
<https://doi.org/10.1109/URAI.2013.6677329>

LIMA, G. Modelagem dinâmica e controle para navegação de um veículo aéreo não tripulado do tipo quadricóptero. 2015. - Universidade Federal de Uberlândia, [s. l.], 2015. Available in: <https://repositorio.ufu.br/handle/123456789/14608>.

LIMA, G. Planejamento de trajetórias para quadricópteros em tarefas de perseguição. 2019. - Universidade Federal de Uberlândia, [s. l.], 2019. Available in: <https://repositorio.ufu.br/handle/123456789/27532>.

LIMA, G. V. *et al.* Stabilization and Path Tracking of a Mini Quadrotor Helicopter: Experimental Results. IEEE Latin America Transactions, [s. l.], v. 17, n. 03, p. 485–492, 2019. <https://doi.org/10.1109/TLA.2019.8863319>

LIN, X.-C.; TSAI, C.-C.; TAI, F.-C. Cooperative SLAM of an Autonomous Indoor Quadrotor Flying Together with an Autonomous Ground Robot. [s. l.], 2019. Available in: <https://ieeexplore.ieee.org/document/8765035>.

LÓPEZ, E. *et al.* A Multi-Sensorial Simultaneous Localization and Mapping (SLAM) System for Low-Cost Micro Aerial Vehicles in GPS-Denied Environments. Sensors, [s. l.], v. 17, n. 4, p. 802, 2017. <https://doi.org/10.3390/s17040802>

MALINVERNI, E. S. *et al.* Evaluating a Slam-Based Mobile Mapping System: a Methodological Comparison for 3D Heritage Scene Real-Time Reconstruction. In: 2018 METROLOGY FOR ARCHAEOLOGY AND CULTURAL HERITAGE (METROARCHAEO), 2018, Cassino FR, Italy. 2018 Metrology for Archaeology and Cultural Heritage (MetroArchaeo). Cassino FR, Italy: IEEE, 2018. p. 265–270. Available in: <https://ieeexplore.ieee.org/document/9089765/>.
<https://doi.org/10.1109/MetroArchaeo43810.2018.13684>

MCGUIRE, K. N. Indoor swarm exploration with Pocket Drones. 2019. - Delft University of Technology, [s. l.], 2019. <https://doi.org/10.4233/uuid:48ed7edc-934e-4dfc-b35c-fe04d55caee1>

MCGUIRE, K. N. *et al.* Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics*, [s. l.], v. 4, n. 35, p. eaaw9710, 2019. <https://doi.org/10.1126/scirobotics.aaw9710>

MOHSAN, S. A. H. *et al.* Unmanned aerial vehicles (UAVs): practical aspects, applications, open challenges, security issues, and future trends. *Intelligent Service Robotics*, [s. l.], 2023. Available in: <https://link.springer.com/10.1007/s11370-022-00452-4>. <https://doi.org/10.1007/s11370-022-00452-4>

MONTEMERLO, M. *et al.* FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. [s. l.], 2003.

MORAVEC, H.; ELFES, A. High resolution maps from wide angle sonar. *In: 1985 IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION*, 1985, St. Louis, MO, USA. Proceedings. 1985 IEEE International Conference on Robotics and Automation. St. Louis, MO, USA: Institute of Electrical and Electronics Engineers, 1985. p. 116–121. Available in: <http://ieeexplore.ieee.org/document/1087316/>. <https://doi.org/10.1109/ROBOT.1985.1087316>

MÜLLER, H. *et al.* Robust and Efficient Depth-Based Obstacle Avoidance for Autonomous Miniaturized UAVs. *IEEE Transactions on Robotics*, [s. l.], v. 39, n. 6, p. 4935–4951, 2023. <https://doi.org/10.1109/TRO.2023.3315710>

MURPHY, R. Introduction to AI robotics. Cambridge, Mass.: MIT Press, 2000. (Intelligent robotics and autonomous agents).

NEMRA, A. Robust Airborne 3D Visual Simultaneous Localisation And Mapping. 2010. - Cranfield University, [s. l.], 2010. Available in: <http://dspace.lib.cranfield.ac.uk/handle/1826/6157>.

NICULESCU, V. *et al.* NanoSLAM: Enabling Fully Onboard SLAM for Tiny Robots. *IEEE Internet of Things Journal*, [s. l.], p. 1–1, 2023. <https://doi.org/10.1109/JIOT.2023.3339254>

RAFFO, G. V. Robust Control Strategies for a QuadRotor Helicopter. An Underactuated Mechanical System. 2011. - Universidad de Sevilla, [s. l.], 2011.

RAMOS, D. C. Aplicação de técnicas de fusão sensorial para mapeamento e localização simultâneos para robôs terrestres. 2012. - Universidade Federal de Santa Catarina, [s. l.], 2012. Available in: <http://repositorio.ufsc.br/xmlui/handle/123456789/100770>.

REACTIVEX. ReactiveX: Reactive Extensions for Async Programming. [S. l.], [s. d.]. Available in: <https://github.com/ReactiveX>. .

RUBIO, F.; VALERO, F.; LLOPIS-ALBERT, C. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, [s. l.], v. 16, n. 2, p. 172988141983959, 2019. <https://doi.org/10.1177/1729881419839596>

RUEDA RAMOS, J. M. G. de. SLAM for drones: simultaneous localization and mapping for autonomous flying robots. 2012. - Universidad Carlos III de Madrid, [s. l.], 2012. Available in: <http://hdl.handle.net/10016/16099>.

SADEGHZADEH-NOKHODBERIZ, N. *et al.* Dynamics-Based Modified Fast Simultaneous Localization and Mapping for Unmanned Aerial Vehicles With Joint Inertial Sensor Bias and Drift Estimation. *IEEE Access*, [s. l.], v. 9, p. 120247–120260, 2021. <https://doi.org/10.1109/ACCESS.2021.3106864>

SAEEDI, S. *et al.* 3D Mapping for Autonomous Quadrotor Aircraft. *Unmanned Systems*, [s. l.], v. 05, n. 03, p. 181–196, 2017. <https://doi.org/10.1142/S2301385017400064>

SANCHEZ-LOPEZ, J. L. *et al.* Deep learning based semantic situation awareness system for multicopter aerial robots using LIDAR. *In: 2019 INTERNATIONAL CONFERENCE ON UNMANNED AIRCRAFT SYSTEMS (ICUAS), 2019, Atlanta, GA, USA. 2019 International Conference on Unmanned Aircraft Systems (ICUAS). Atlanta, GA, USA: IEEE, 2019. p. 899–908.* Available in: <https://ieeexplore.ieee.org/document/8797770/>. <https://doi.org/10.1109/ICUAS.2019.8797770>

SCARADOZZI, D.; ZINGARETTI, S.; FERRARI, A. Simultaneous localization and mapping (SLAM) robotics techniques: a possible application in surgery. *Shanghai Chest*, [s. l.], v. 2, p. 5–5, 2018. <https://doi.org/10.21037/shc.2018.01.01>

SHEN, S.; MICHAEL, N.; KUMAR, V. Obtaining Liftoff Indoors: Autonomous Navigation in Confined Indoor Environments. *IEEE Robotics & Automation Magazine*, [s. l.], v. 20, n. 4, p. 40–48, 2013. <https://doi.org/10.1109/MRA.2013.2253172>

SIEGWART, R.; NOURBAKHSI, I. R.; SCARAMUZZA, D. Introduction to autonomous mobile robots. 2nd eded. Cambridge, Mass: MIT Press, 2011. (Intelligent robotics and autonomous agents).

SMITH, R. C.; CHEESEMAN, P. On the Representation and Estimation of Spatial Uncertainty. *The International Journal of Robotics Research*, [s. l.], v. 5, n. 4, p. 56–68, 1986. <https://doi.org/10.1177/027836498600500404>

SOUZA, R. M. J. A. *et al.* Modified Artificial Potential Field for the Path Planning of Aircraft Swarms in Three-Dimensional Environments. *Sensors*, [s. l.], v. 22, n. 4, p. 1558, 2022. <https://doi.org/10.3390/s22041558>

SOUZA, R. Sistema de identificação de fonte de gás utilizando nanoquadricóptero. 2022. - Universidade Federal de Uberlândia, [s. l.], 2022. Available in: <https://repositorio.ufu.br/handle/123456789/36129>.

STACHNISS, C. Exploration and mapping with mobile robots. 2006. - Universitätsbibliothek Freiburg, [s. l.], 2006. Available in: <https://freidok.uni-freiburg.de/data/2440>.

STACHNISS, C. *Robotic Mapping and Exploration*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009. (Springer Tracts in Advanced Robotics). v. 55 Available in: <http://link.springer.com/10.1007/978-3-642-01097-2>. <https://doi.org/10.1007/978-3-642-01097-2>

TAHERI, H.; XIA, Z. C. SLAM; definition and evolution. *Engineering Applications of Artificial Intelligence*, [s. l.], v. 97, p. 104032, 2021. <https://doi.org/10.1016/j.engappai.2020.104032>

TAKLEH, T. T. O. *et al.* A Brief Survey on SLAM Methods in Autonomous Vehicle. *International Journal of Engineering & Technology*, [s. l.], v. 7, n. 4.27, p. 38, 2018. <https://doi.org/10.14419/ijet.v7i4.27.22477>

THRUN, S. *et al.* FastSLAM: An Efficient Solution to the Simultaneous Localization And Mapping Problem with Unknown Data Association. [*s. l.*], 2002.

THRUN, S.; BURGARD, W.; FOX, D. A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. *Machine Learning*, [*s. l.*], v. 31, n. 1/3, p. 29–53, 1998. <https://doi.org/10.1023/A:1007436523611>

THRUN, S.; BURGARD, W.; FOX, D. Probabilistic robotics. Cambridge, Mass: MIT Press, 2005. (Intelligent robotics and autonomous agents).

VON STUMBERG, L. *et al.* From monocular SLAM to autonomous drone exploration. *In: 2017 EUROPEAN CONFERENCE ON MOBILE ROBOTS (ECMR), 2017, Paris. 2017 European Conference on Mobile Robots (ECMR). Paris: IEEE, 2017. p. 1–8. Available in: <http://ieeexplore.ieee.org/document/8098709/>. <https://doi.org/10.1109/ECMR.2017.8098709>*

XIA, L. *et al.* A point-line-plane primitives fused localization and object-oriented semantic mapping in structural indoor scenes. *Measurement Science and Technology*, [*s. l.*], v. 33, n. 9, p. 095017, 2022. <https://doi.org/10.1088/1361-6501/ac784c>

YUAN, S.; WANG, H.; XIE, L. Survey on Localization Systems and Algorithms for Unmanned Systems. *Unmanned Systems*, [*s. l.*], v. 09, n. 02, p. 129–163, 2021. <https://doi.org/10.1142/S230138502150014X>

ZHENG, S. *et al.* Simultaneous Localization and Mapping (SLAM) for Autonomous Driving: Concept and Analysis. *Remote Sensing*, [*s. l.*], v. 15, n. 4, p. 1156, 2023. <https://doi.org/10.3390/rs15041156>

ZHOU, H. *et al.* Efficient 2D Graph SLAM for Sparse Sensing. *In: 2022 IEEE/RSJ INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS), 2022, Kyoto, Japan. 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Kyoto, Japan: IEEE, 2022. p. 6404–6411. Available in: <https://ieeexplore.ieee.org/document/9981200/>. <https://doi.org/10.1109/IROS47612.2022.9981200>*

ZHOU, Q.-Y.; PARK, J.; KOLTUN, V. Open3D: A Modern Library for 3D Data Processing. [*s. l.*], 2018. Available in: <https://arxiv.org/abs/1801.09847>. <https://doi.org/10.48550/arXiv.1801.09847>