Simultaneous Navigation And Mapping (SNAM) Using Collision Resilient UAV

by

Aravind Adhith Pandian Saravanakumaran

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved June 2023 by the
Graduate Supervisory Committee:

Wenlong Zhang, Chair
Jnaneshwar Das
Spring Berman

ARIZONA STATE UNIVERSITY

August 2023

ABSTRACT

Navigation and mapping in GPS-denied environments, such as coal mines or dilapidated buildings filled with smog or particulate matter, pose a significant challenge due to the limitations of conventional LiDAR or vision systems. Therefore there exists a need for a navigation algorithm and mapping strategy which do not use vision systems but are still able to explore and map the environment. The map can further be used by first responders and cave explorers to access the environments.

This thesis presents the design of a collision-resilient Unmanned Aerial Vehicle (UAV), XPLORER that utilizes a novel navigation algorithm for exploration and simultaneous mapping of the environment. The real-time navigation algorithm uses the onboard Inertial Measurement Units (IMUs) and arm bending angles for contact estimation and employs an Explore and Exploit strategy. Additionally, the quadrotor design is discussed, highlighting its improved stability over the previous design.

The generated map of the environment can be utilized by autonomous vehicles to navigate the environment. The navigation algorithm is validated in multiple real-time experiments in different scenarios consisting of concave and convex corners and circular objects. Furthermore, the developed mapping framework can serve as an auxiliary input for map generation along with conventional LiDAR or vision-based mapping algorithms.

Both the navigation and mapping algorithms are designed to be modular, making them compatible with conventional UAVs also. This research contributes to the development of navigation and mapping techniques for GPS-denied environments, enabling safer and more efficient exploration of challenging territories.

ACKNOWLEDGMENTS

I am immensely grateful to my esteemed advisor, Dr. Wenlong Zhang, for granting me the opportunity to immerse myself in the captivating and pioneering field of collision-resilient drones at RISE lab. Dr. Zhang's unwavering support, feedback, and guidance have been truly invaluable throughout my research journey, and I am profoundly thankful for his continuous assistance. I would also like to extend my heartfelt gratitude to Dr. Jnaneshwar Das for his constant encouragement, mentorship, and the chance to contribute to the exploration of Aerial and Aquatic robots and their diverse applications. Furthermore, I express my gratitude to Dr. Spring Berman for her valuable participation in my thesis committee, offering insightful comments and suggestions.

I extend my heartfelt thanks to Karishma Patnaik for her unwavering motivation, invaluable advice, and unwavering support throughout my tenure at RISE lab. Her contributions in formulating this thesis have been of utmost importance, and I am profoundly grateful for her assistance. I would also like to express my deep appreciation to all the lab members for their valuable insights into the research work and the delightful moments we shared during our time together. Working alongside each of them for the past two years has been an enriching and memorable experience that I will forever hold dear.

I would like to take this opportunity to extend my sincere appreciation to all my friends for their unwavering encouragement and support during the challenging and demanding times. Their presence has been a source of strength and motivation for me. Additionally, I am immensely grateful to my parents for their unwavering love and unwavering belief in my abilities. I attribute my accomplishments to their constant support and guidance. Meeting their expectations has been a driving force for me, and

I am committed to surpassing them and reaching even greater heights in the future. Lastly, I want to express gratitude to myself for believing in me, working hard, never giving up, being generous, striving to do what's right, and staying true to myself.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION & LITERATURE REVIEW

1.1 Introduction

Various methods have been used to achieve locomotion for autonomous robots with knowledge of the environment but navigation in unknown environments is still being explored. Exploratory path planning is a promising approach for exploring unknown environments. The primary motivation for this thesis comes from the DARPA SubTerrean Challenge [1]. Since there was a challenge in this competition is to continue exploring even when perception systems fail. Additionally, the robot should be able to synthesize an accurate map of the environment when conventional mapping techniques, such as LiDAR [2] or vision systems [3], fail. Conventionally, exploration is accomplished using Simultaneous Localization and Mapping (SLAM) techniques and one of several path planning algorithms, such as A*, Dijkstra's, RRT [4], etc. This approach enables a robot to navigate through the environment while simultaneously generating an accurate map of the environment. These algorithms are commonly used in Unmanned Aerial Vehicles (UAVs), Unmanned Surface Vehicles (USVs), and ground vehicles for autonomous navigation without human operators.

Mapping an environment using Light Detection And Ranging (LiDAR) or vision systems and one of the Simultaneous Localization and Mapping (SLAM) algorithms is an accurate method, but there are situations when these systems may fail. For instance, in environments like caves, where there is smog or particulate matter, the performance of LiDAR may be affected [5]. The particulate matter can obstruct the

light rays and produce noisy data, resulting in an inaccurate map. Similarly, vision systems may be hindered in environments with thick smog, making it challenging to perceive the environment accurately. These inaccuracies may result in a flawed path planner, causing the UAV to navigate through an undesired path, and in some cases, it may lead to crashes.

The problem of crashing can be overcome by utilizing a collision-resilient or tolerant UAV [6, 7, 8, 9], that can withstand collisions and can still continue its trajectory. There are various novel designs that utilize a protective structure around the UAV [7] or passive systems that can absorb the impact forces [6, 10]. The collision absorbing systems can be either passive [7, 11, 12] or active [6, 13] which may use springs or deformable structures. The protective structures can protect the UAV's propellers from contacting the environment[8, 14] or also consist of tactile sensors [11, 15] that can detect contact. These types of collision-resilient UAVs are able to perform the tasks of conventional UAVs but also are able to do special tasks such as contact inspection [16], collision-based motion planning [17] or to carryout mapping [8, 18]. Contact inspection can be used to conduct Non-Destructive Testing or crack inspection on structures. Apart from this as aforementioned use cases, these UAVs can be used in search and rescue missions also. Interaction and contact scenarios such as end effector force control and contact inspection are areas of growing interest and the performance of these types of UAVs is comparatively higher than conventional multirotor UAVs.

The present research on collision-resilient UAVs aims to tackle the challenge of environment navigation; however, the current approaches involve an element of randomness, resulting in uncontrolled maneuvering of the UAVs. Instead of mapping the obstacles in the environment, the drones tend to evade them. If the obstacles are mapped, the information can be shared with other autonomous vehicles. Therefore,

the emphasis of this thesis is to design a collision-resilient quadcopter and develop an exploration algorithm that can navigate through the environment while mapping all obstacles.

1.1.1 Problem Statement

During rescue operations, rescue personnel often face multiple challenges, including limited visibility, inaccessibility, unpredictable circumstances, and limited knowledge of the environment. Figure 1 illustrates a collapsed and dilapidated building, which may be a similar scenario where rescue personnel must navigate to explore or rescue people. The collapse of a building can result from natural disasters, improper construction, or the age of the building, with older buildings being more susceptible to environmental conditions that could compromise their integrity. Even if the blueprint of the building is available, the pathways and structure may become deformed and pose a risk of further collapse.

Figure 1. Collapsed[19] and Dilapidated Buildings[20]

Cave exploration and rescue is another scenario where manual exploration is risky and tedious. Figure 2 shows snippets from Tham Luang cave rescue which was carried

out in one of the caves in Thailand where a group of 12 students got stuck and it took 18 days to rescue them. One of the reasons was that the cave network was not completely mapped and rescuers had to initially find the path, before starting the rescue operation. There are approximately 38,991 abandoned mines in the USA, and the availability of the maps of these mines is uncertain. So, exploration of this cave system still is a tedious task and the need for an autonomous mapping system still exists.

Figure 2. Cave Exploration[21] and Rescue[22]

The DARPA SubTerrean Challenge [1] served as a competitive ground to come up with a solution for the mentioned problem. It aimed to advance technologies for underground navigation, mapping, and search and rescue operations. The competition involves teams from around the world who develop and deploy autonomous systems to navigate through complex underground environments, including tunnels, caves, and urban underground infrastructure. The competition is designed to test the ability of autonomous systems to operate in GPS-denied environments, which are common in underground environments. The systems must navigate through the environments, identify objects, and map the surroundings in real-time. The challenge was split into three different tracks: the Systems track, the Virtual track, and the Open Exploration

track, which involve different levels of autonomy and complexity. The Subterranean Challenge was seen as an important step towards developing technologies that can be used in disaster response scenarios and other situations where humans cannot easily access underground environments.

All the teams addressed the problem of navigating such complex environments using visual sensors such as LiDAR or computer vision systems, the problem of when these systems fail was not solved. Therefore, the proposed Navigation and Mapping Algorithm serves as a solution to this problem and has been deployed in the collision-resilient drone and validated for certain cases.

1.1.2 Challenges

Although the problem statement may appear akin to floor-cleaning robots, the difficulty of implementing it on a quadcopter presented numerous challenges due to the increased degrees of freedom that must be precisely controlled. Below, we discuss the primary difficulties that were encountered during the quadrotor design and algorithm development.

1. In-Plane propellers and Higher Centre of Gravity(CoG)

   The present configuration of the quadcopter was initially inspired by SQUEEZE[6], which had offset axes for the propellers, allowing it to navigate through small spaces. However, in the context of exploration and mapping, the quadcopter needs to have stable contact with the environment. Since the adjacent propellers were off-axis, there was a singularity where the drone would get stuck in ledges whose thickness was less than the off-axis distance. The

center of mass of the previous design was also above the propellor axis leading to the possibility of toppling upon contact.

2. Compliance to the Obstacle

   To ensure the stability and efficiency of the quadcopter, it is necessary for it to conform to obstacles it encounters rather than trying to maintain its orientation. Failure to conform can cause the drone to generate yaw about the point of impact, requiring correction from the controller and leading to instability and wasted control effort.

3. Arm Angle Estimation

   The incorporation of the torsional spring introduces a new challenge for controlling the drone and creating a map of the environment. It is essential to estimate the bending angle of each arm because it serves as an input parameter for both the mapping framework and wrench estimation.

4. External Wrench Estimation

   The measurement of the force acting on the drone is an essential parameter that allows the algorithm to identify whether the drone has encountered an obstacle or not. However, the deformable nature of the drone during a collision presents a challenge because the forces acting on the controller do not fully represent the actual force. This is because some of the force is absorbed by the torsional spring. Additionally, while sliding on an obstacle, the drone also experiences a frictional force that needs to be taken into account.

5. Concave and Convex Obstacles

   The shape of obstacles is a significant challenge for the algorithm as the quadcopter reacts differently to different curvatures and types of curvature. Therefore, the algorithm was adapted to make sure all types of obstacles can be traversed

and mapped. Convex obstacles or corners can cause the drone to yaw around the point, which needs to be controlled systematically. For concave corners, the drone can get stuck and may not be able to recover, which leads to singularity cases.

6. Point Cloud Generation

   The conventional method of generating a point cloud relies on input from the visual sensor. However, for the mapping framework to generate the necessary data, a new approach was needed. Additionally, the resulting map should accurately depict the obstacle's boundary layer so that it can be utilized by other navigation algorithms.

1.2 Literature Review

A brief description of recent work related to collision-resilient UAVs, navigation algorithms, and mapping techniques is discussed in this section. Drawbacks for each of the works mentioned are detailed, and advancements that were made to address those drawbacks are then explained. The concluding paragraph of this section describes the significance and novelty of this thesis.

1.2.1 Collision-Resilient UAVs

Collision-resilient UAVs have been developed by multiple researchers as shown in [6, 7, 11, 23, 24, 25, 26] each of it has its own pros and cons. A protective rigid structure is used to sustain collisions in [8, 11, 12] such that the propellers are protected from getting into contact with the environment. The drawback with the approach is that

the collision forces are directly transmitted to the drone which causes instability issues. Authors of [13] propose a novel design where it consists of a shock absorber that helps to take collision, the protective structure around the propeller is not complete hence leading to the possibility of collision with vertical obstacles. An inflated flap-like structure was used in [27] which was able to sense collisions and navigate across the environment, the drawback with this is that when the collision forces are high it would directly impact the inflated structure and might topple the drone. Torsional springs were used in [6] which were mounted at the joint of the propeller arm and the body assisted in absorbing the collisions and allowed the morphing of the drone to squeeze through confined spaces. Another collision-resilient UAV is shown in [26] which also utilizes a recovery controller which re-plans the post-collision trajectory by analyzing the post-collision dynamics of the UAV. Authors of [28] proposed an adaptive attitude controller for varying morphology UAV which was able to produce superior tracking performance compared to the conventional geometric controller. The design part of this thesis was inspired and developed on the design of [6], the drawback in the design was that the drone was not stable for contact-enabled tasks, and the off-plane propellers meant that there was a possibility of the drone getting stuck in between the obstacles. The new design proposed in the further chapters overcomes these problems but still uses the core design of using torsional springs for collision force mitigation. Another novel collision-resilient drone is presented in [29] which is capable of taking collisions in all six directions. This is enabled due to the body of the quadrotor being fabricated with soft actuators instead of conventional plastic or metal parts.

1.2.2 Contact-Based Navigation

Contact-based navigation is an area of research that multiple researchers have worked on and have successfully exhibited its capabilities for various applications. Contact sensing for these applications has been done using tactile sensors [11] or inertial measurement units (IMUs) by authors of [18, 27, 30]. Upon collision detection, the motion planners replan the trajectory to navigate about it. A novel contact-based navigation planner was proposed in [31] which consists of two modes, namely sliding and flying cartwheel. Unlike other approaches where obstacle avoidance is the main motivation, the planner tries to track the obstacle and tries to be in contact with the obstacle using either of those two modes. Normally upon encountering an obstacle, the drone would slide across the obstacle and traverse around. It switches to the cartwheel motion if there exists no feasible trajectory for sliding or if the drone stalls due to map inconsistencies. The cons of this planner are that it has a knowledge of the environment and the authors do state that the flying cartwheel mode has a risk of high collision forces which compromise robust state estimation. Computer vision is used for navigating man-hole-sized tubes in [27] along with flaps to sense contacts. In [32], the map knowledge is available as point cloud data, on which the authors use a motion planner to compute trajectories. In [17] a sampling bases method was used for path-planning by exploiting collisions, similar to previous articles the map knowledge is known. Navigation in an unknown environment is still fledgling. Authors of [11] propose a random exploration algorithm, which is similar to a bee's behavior. Upon collision, the drone would start moving in the direction away from the obstacle. The drawback with this approach is that due to the randomness, the coverage rate is uncertain and reproducibility of the results is hard. There is a requirement

for controlled trajectory planning while the vehicle is in constant contact with the environment.

1.2.3 Tactile-Mapping

Mapping an environment whilst exploring has served as the foundation for SLAM. The map generated is used for trajectory planning by the same vehicle or by other autonomous vehicles. Traditionally map generation is done using the concept of Structure from Motion (SfM) technique [33]. It is a computer vision technique that aims to reconstruct the three-dimensional structure of a scene or object from a series of two-dimensional images or video frames. It involves estimating the camera motion and the 3D positions of points in the scene simultaneously. By analyzing the correspondences between image features across multiple views, SfM can infer the geometry and spatial relationships of the scene, allowing for the creation of a 3D model or reconstruction. SfM can also be done using point cloud data obtained from a LiDAR, authors of [34] proposed an algorithm that utilizes both LiDAR and a stereo camera to generate a high-resolution map of the environment. The aforementioned techniques are not real-time, they are used to process the data that was previously recorded to generate maps. Eventually, as research and computation power progressed, real-time map generation was achieved along with SLAM algorithms. Map generation algorithms were proposed in [35, 36] which uses a monocular camera to achieve SLAM. A 2-axis LiDAR was used in [37] to generate a map of the environment and localize the vehicle. Sensor fusion was achieved in [38] using multiple sensors such as LiDAR, IMU, and encoders to generate a high-resolution map of the environment. Another novel multi-sensor fusion algorithm was proposed in [39] uses GPS for outdoor navigation

and LiDAR or camera when GPS data is not available to navigate across a complex environment. A novel map generation technique is presented in [18] which utilizes collision detection to place obstacle blocks in the map and re-plan the trajectory. However, the obstacle blocks are of constant dimension and do not represent the exact obstacle dimensions, also the algorithm does obstacle avoidance and moves around the environment.

1.2.4 Congregation and Docking Applications

Another application where collision-resilient UAVs can be utilized is multi-robot congregation and docking. Where multiple collision-resilient UAVs can be cohered together and carry out various tasks. Visual sensing and docking are proposed by authors of [40, 41] where the individual UAVs come together with the help of either motion capture or camera sensing systems respectively. This type of multi-robot coordination can be used to carry out various tasks. In this case, for docking purposes, the UAV's body consists of neodymium magnets to interlock with each other which adds complexity to the design and undocking is harder since the motors would saturate before generating enough thrust to detach. A multi-link robot is proposed in [27] where the multi-linked robot is used to form different shapes and the whole system is modeled which helps it to control. A similar multi-linked UAV is proposed in [42] which also uses an external wrench estimator to estimate the forces and torques acting on the body which is then utilized for collision detection. The same UAV is also used in [43, 44] for applications such as pivoting and pick & place tasks. The UAV is capable of sensing the force and the wrench estimator provides the external forces acting on the UAV. Even though the UAV is a multi-linked body it is not capable of

adding multiple links together in real-time. But a decentralized group of UAVs can come together to accomplish such tasks. If a larger force is needed then multiple UAVs together can act as a single UAV and perform the tasks. Even though this problem is not addressed in this thesis, the current proposed collision-resilient quadcopter can be employed for such tasks.

This thesis thus provides a collision-resilient quadcopter, XPLORER, and the Simultaneous Navigation And Mapping (SNAM) algorithm which can be used for environmental exploration, obstacle analysis, and other applications.

1.3 Organization of the Thesis

The thesis is divided into three modules. The first module, Module 1, provides a brief overview of the literature review on contact-based navigation and mapping techniques, as well as the drawbacks of current state-of-the-art technologies. The second module, Module 2, comprises the work conducted for the thesis. Chapter 3 discusses the novelty of the design and upgrades from the previous design, including the elimination of singularities and the modularity of the new design. Chapter 4 describes the navigation algorithm, including the state machine model and how it uses the estimated Arm angles and External Wrench. Chapter 5 explains how multiple input parameters such as the CAD model of the drone, odometry data, and arm angles data are used to generate the point cloud and the C-Space of the obstacle. The final module, Module 3, consists of the experimental results and validation of the navigation algorithm, as well as a discussion of the accuracy of the map generated and metrics for the pipeline. The thesis concludes with final remarks and an overview of future work

1.3.1 Research Objective and Contributions

The objective of this thesis is to redesign a collision-resilient UAV and come up with a novel exploratory navigation algorithm and mapping pipeline for the same with the following features -

- The design of the drone must be robust and avoid singularity cases or toppling upon impact, while also allowing for modular construction that facilitates rapid component replacement.
- The navigation algorithm should enable the drone to effectively explore obstacles with convex or concave curvature, regardless of orientation, and should be adapted for use on conventional drones equipped with propeller guards.
- The mapping pipeline's modularity enables it to be implemented with minimal modifications on any type of drone using CAD models and flight controller parameters and to generate an accurate map of the obstacle.
- The whole thesis was developed with modularity and distributed processing as the driving factor. So, that the algorithm and pipeline can be recreated by fellow people for their own drones and reduce the computational load on the onboard computer and offload it to a powerful computer.

With the above-mentioned features in mind, a reliable and robust exploratory navigation algorithm and a mapping framework were developed and presented in this thesis. The existing Collision-resilient drones address the problem of post-collision stability but the design proposed in this thesis aims to achieve exploration capabilities along with post-collision stability. The design also helps to perform tasks like contact inspection and force application as shown in [10]. The navigation algorithm allows the drone to be in contact even after collision and traverse along the obstacle, instead

of navigating around it, which conventional navigation algorithms focus on. It is analogous to a blind person moving around a room and was the motivation for this algorithm. The navigation algorithm assists the mapping framework to generate a precise map without any visual sensors. The mapping framework is not targeted to be a standalone mapping strategy but instead to assist conventional mapping methodologies when the effectiveness of the visual sensors fail and this can supplement those methods.

1.3.2 Assumptions

The major assumptions that were made in the thesis are as follows.

1. The obstacle's surface is smooth and not serrated. This was considered for experiments since the propellor guards are made of PLA and wear and tear might affect its integrity.
2. The odometry state estimation of the drone is significantly accurate. This was done to limit the extensivity of the thesis as inertial odometry itself is a separate topic to develop on.

Chapter 2

DESIGN OPTIMIZATION

In this chapter, we will discuss the improvements made to the design of the quadcopter to address the limitations of the previous design presented in [6]. The initial section will outline the rationale for the new design, followed by a description of the design modifications in subsequent sections. Lastly, we will highlight the modular features of the new design.

Figure 3. Morphing capability of SQUEEZE [6]

2.1 Drawbacks and Need for Design Improvements

Figure 3 illustrates the original design, which was utilized in [6]. This unique design features variable geometry that enables it to navigate cluttered environments, and maneuver through narrow gaps and passages. The quadcopter is compliant and

has passive morphing capabilities, achieved by incorporating torsional springs at every arm hinge to facilitate rotation propelled by external forces. The arms are specifically designed so that the adjacent motors and propellers are at varying heights, preventing any physical interference between the arms during folding (refer to arm types 1 and 2 in Figure 4), this enables the arms to bend about 90 degrees.

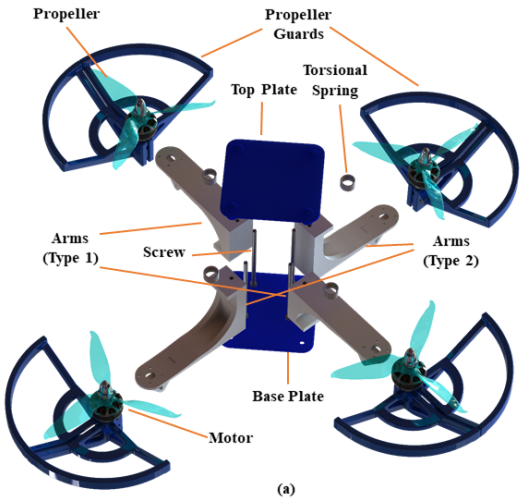


Figure 4. Exploded View of the SQUEEZE [6]

The quadcopter's design is effective for its intended purpose of navigating through narrow spaces and confined areas. However, it poses a risk of instability upon colliding with obstacles due to the varying heights of the adjacent propellers. As a result, the contact plane formed in such situations is not perpendicular to the quadcopter's body, increasing the likelihood of it getting stuck. There is a possibility of crashing if the obstacle's thickness is less than the height difference between the propellers as shown in Figure 5. Additionally, the design employs semi-circular propeller guards, creating a potential for the propellers to collide with obstacles in the sagittal plane, in this case, the propeller will hit the obstacle say the wall. Another drawback in the design

was that the propellers were above the top plane of the propeller guards which again poses the risk of crashing. These limitations are depicted in Figure 6.

The design also has a problem with the Center of Gravity (CoG) of the drone being located above one of the propeller axes. This induces a moment upon contact with an obstacle that can cause the drone to topple. While this was not a significant issue for passage pass-through applications, it can lead to instability during higher velocity impacts where the contact is only on one plane which is the focus of this research.

The limitations mentioned above necessitate a revised design that addresses the issues without adding any weight to the drone. The upcoming sections will describe how the new design effectively overcomes these limitations and outperforms its predecessor.

Figure 5. The adjacent propellers are off-axis which causes SQUEEZE to slide in and crash

2.2 Design Derivatives from Previous Design

XPLORER's design language still follows the concept of passive adaptive morphology similar to SQUEEZE, utilizing torsional springs for morphing. However, all four

(a) Propeller above the frame (b) Obstacle in the Sagittal Plane

Figure 6. Drawbacks of SQUEEZE

arms now use stiffer springs compared to the previous design, which only used two arms with these springs. The new design is set up in the 'X' configuration to enable two-point contact with obstacles, deviating from the previous '+' configuration. The new design measures 43 cm in edge-to-edge distance compared to the initial design's 37 cm, the total height is longer due to the three-stack design. Further sections will discuss the design improvements implemented.

2.3 Design Refinements

The redesigned SQUEEZE is illustrated in Figure 7. The primary modification is that all the propellers are now in the same plane, which allows for secure contact with obstacles. In addition, the propeller plane is almost coincident with the top plate, and the Center of Gravity (CoG) is below that plane, as depicted in Figure 8. This prevents the drone from toppling since upon establishing contact it would require a lot more effort to generate a moment and to topple. But since the adjacent propellers are now in the same plane, the ability to squeeze and fly is limited. The arms can now only deform up to 30 degrees before coming in contact with the adjacent one.

Furthermore, since the scope of this thesis is to explore contact-based navigation abilities and not showcase squeeze and fly abilities, we limit the linear velocities to 2m/s.

The propeller guard of SQUEEZE [6] measured 1 cm in width but the new design measures 2 cm in width but is split into two 0.5 cm rings to enable better contact with obstacles. The dual ring design enables a larger contact area and this was done to reduce the weight and also reduce the contact area to reduce friction while traversing across obstacles. From Figure 7 we can see that the propeller guards are now circular, providing protection from all directions including the sagittal plane, and aiding in stable obstacle contact.

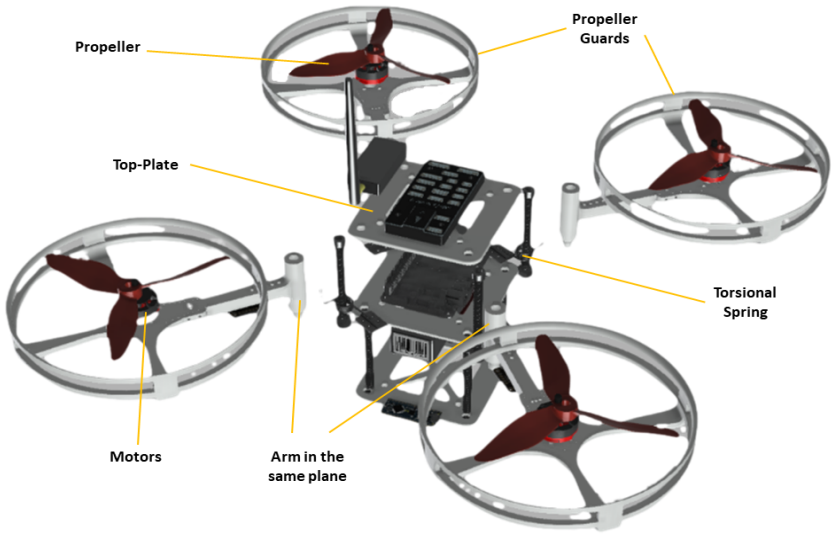


Figure 7. Exploded View of the XPLORER

2.4 Weight Optimization

From the initial stages, weight optimization was considered and incorporated into the new design. The initial design had a weight of approximately 632 grams, whereas the new design weighs approximately 635 grams without including the battery and the onboard computer. This weight was achieved by using Nylon screws and nuts instead of steel ones and by 3D printing the parts at 20% infill using PLA material. The drone's design was changed to a three-stack design from a two-stack design to protect the battery from impacts while landing and all the electronics were enclosed within a protective structure which can be seen in Figure 8. The propeller's size was also increased to 6 inches from 5 inches, resulting in increased thrust and a higher thrust-to-weight ratio compared to the previous design. Even though the propeller guards are larger the amount of material used to is less due to the design, since the previous design used a two-piece split design, unlike the new unibody design.

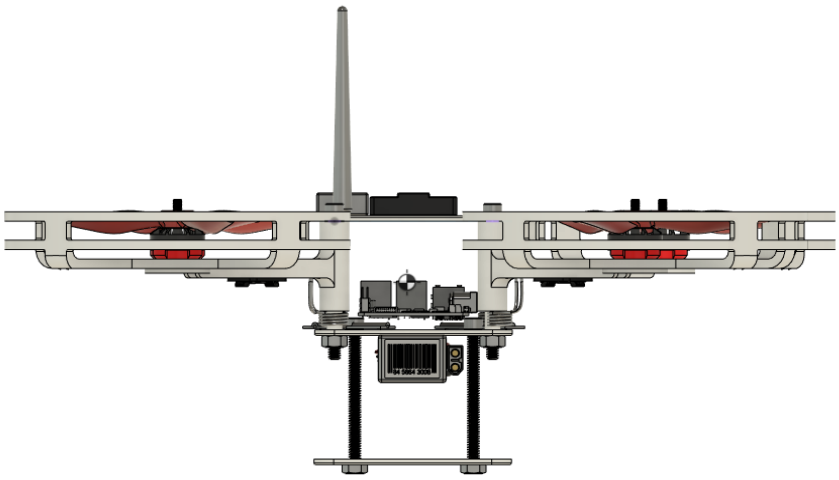


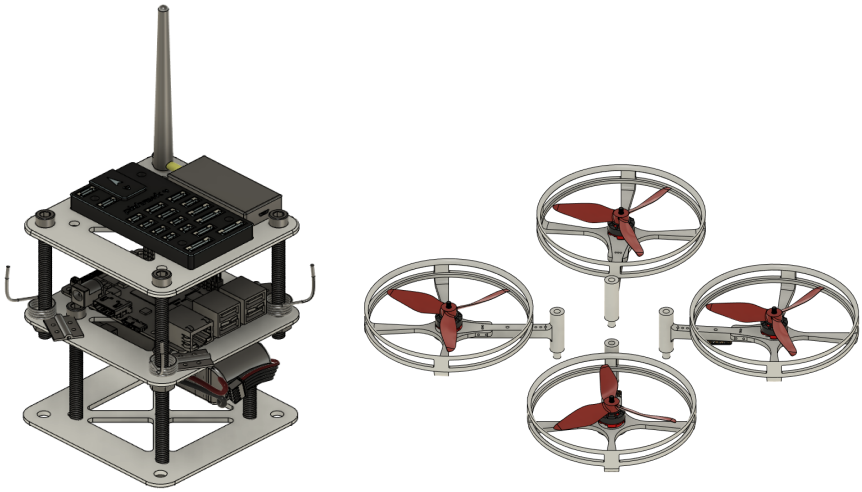
Figure 8. Lateral View of the New Design.

2.5 Modularity

The redesign prioritized modularity and repairability, and the drone can be divided into two distinct modules: the center tower and the propeller module. The center tower, shown in Figure 9(a) which is designed with a three-stack structure, encloses the electronics and battery. The dimensions of the tower allow for the Raspberry Pi and battery to align with the plate edges, and wiring can be routed within the tower to prevent dangling except for the IMUs wiring.

The propeller module shown in Figure 9(b) consists of two parts, the arm, and the propeller guard. The arm features multiple holes that allow for easy swapping of springs of different stiffness. The propeller guard can also be replaced easily since it is held by two screws and nuts. The propeller guards also have a slight amount of flex which allows them to absorb some of the collision impacts, this also possess the problem of the point of failure, and the modular design enables quick replacement.

While the propeller guard can take the forces in the lateral direction, the forces in the vertical direction were also considered. Suppose the drone undergoes a free-fall and crashes the electronics should be protected. The nylon screws act as shock absorbers in this case and thereby shielding the electronics from any impact. When the drone crashes, the screws absorb the forces and break apart, preventing the transfer of force to the electronics. Moreover, the sandwich design of the towers helps to protect the electronics from the environment, ensuring that they do not come into direct contact with it upon crash.



(a) Tower Module (b) Propeller Module

Figure 9. Modules of XPLORER.

2.6 General Specifications

In accordance with the XPLORER design [10], this section provides an overview of the electronics used. The design and assembly were performed using Autodesk Fusion 360 software. Polylactic acid (PLA) was utilized to print the parts using a Prusa i3 MK3S+ 3D Printer. The springs utilized in the design were made of spring steel and wound in a clockwise direction. Each arm has a 9-DoF BNO055 inertial measurement unit (IMU) attached to it to transmit its relative orientation with respect to the body, which will be utilized for estimating the external wrench, which in turn will be used by the navigation algorithm and mapping framework. The PIXHAWK4 is employed as the flight controller with a Raspberry Pi serving as a high-level companion computer. ROS2 is employed to establish communication between the flight controller, the Raspberry Pi, and the local position system provided by Optitrack Systems, CO.

This chapter presents a novel design of a collision-resilient UAV, specifically tailored

for exploration tasks. The proposed redesign prioritizes modularity and repairability while maintaining an optimal weight distribution. XPLORER exhibits an increased contact area with obstacles and a lowered center of gravity, resulting in enhanced contact stability during exploration missions. The modular design facilitates effortless component replacement, such as torsional springs, to adapt to varying operational requirements. Moreover, all electronic components are integrated within the frame, ensuring indirect collision and safeguarding their integrity. Although XPLORER possesses a larger footprint compared to its predecessor, SQUEEZE, the overall weight remains nearly identical, showcasing an effective weight optimization strategy.

Chapter 3

NAVIGATION ALGORITHM AND MAPPING FRAMEWORK

This section gives the formulation for the exploratory navigation algorithm and the mapping framework. It discusses the state machine model for the navigation algorithm and the mapping strategy and the specifications of the map generated.

3.1 Navigation Algorithm

In this section, a navigation algorithm is introduced for exploring the environment. The algorithm adopts a frontier-based explore-and-exploit approach, utilizing the interaction controller [10] and collision-resilient design for tactile-based navigation. To achieve this, the algorithm employs a state machine model. The necessary inputs for the state machine to transition between states based on input data will be discussed. Additionally, it will show how the algorithm overcame the challenges of effectively navigating both convex and concave obstacles.

3.1.1 Requisites for the algorithm

The state machine relies on two primary input parameters to function effectively, namely the external wrench acting on the drone and the odometry data from the flight controller. The external wrench, represented by the symbol $\hat{\boldsymbol{\tau}}$, helps to determine the direction of the collision in relation to the body frame, while the yaw speed, represented by $\dot{\psi}$, serves as a trigger for the tactile-turning state. Although the external wrench

provides both force $(\hat{\boldsymbol{f}})$ and torque $(\hat{\boldsymbol{m}})$, only the forces acting on the X-Y planes are utilized by the algorithm. The external wrench estimator from [10] is used to obtain the wrench data for this scope of the thesis, but it is not limited and any validated wrench estimation techniques can be used and given as input for the algorithm.

To utilize the forces obtained from the wrench estimator, it must first be transformed to the body frame because the trajectory generation is done in this frame. To perform the transformation, Equation (3.3) is employed, with ${}^{b}_{w}\boldsymbol{R}(\boldsymbol{\psi})$ providing the transformation from inertial to the body frame.

$$ {}^{w}_{b}\boldsymbol{R} = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1} $$

Equation (3.1) shows the transformation from the body frame to the inertial frame, while the inverse transformation can be obtained by using Equation (3.2).

$$ {}^{b}_{w}\boldsymbol{R}(\boldsymbol{\psi}) = {}^{w}_{b}\boldsymbol{R}(-\psi) = \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.2} $$

$$ ({}_{b}\hat{\boldsymbol{f}}) = [{}^{b}_{w}\boldsymbol{R}(\boldsymbol{\psi})][\hat{\boldsymbol{f}}] \tag{3.3} $$

The external wrench estimate is obtained at a rate of 50Hz from the wrench estimator, and in order to mitigate the effects of noise, we employ a moving average filter with a buffer size of 50 samples. The filtered data is subsequently utilized in the state machine, where further processing occurs. The moving average filter can be represented mathematically by Equation (3.4).

$$y[n] = (1/50) * (x[n] + x[n-1] + x[n-2] + ... + x[n-49]) \tag{3.4}$$

where y[n] is the filtered output at sample n, x[n] is the input signal at sample n and n is the current sample. The buffer is constantly updated with the First in, First out(FIFO) buffer updation. The performance of the filter can be seen in Figure 10.

Similarly, a low-pass filter is employed to eliminate sensor data noise from the yaw rate, $\dot{\psi}$. The transfer function for the low pass filter is given by the Equation (3.5) and the difference equation is given by the Equation (3.6). The time constant is chosen to be a = 0.1 as it produces optimal filtering. The performance of the filter can be seen in Figure 11.

$$H(z) = (1 - z^{(-1)})/(1 + a * (1 - z^{(-1)})) \tag{3.5}$$

$$y(n) = a * y(n-1) + (1-a) * x(n) \tag{3.6}$$

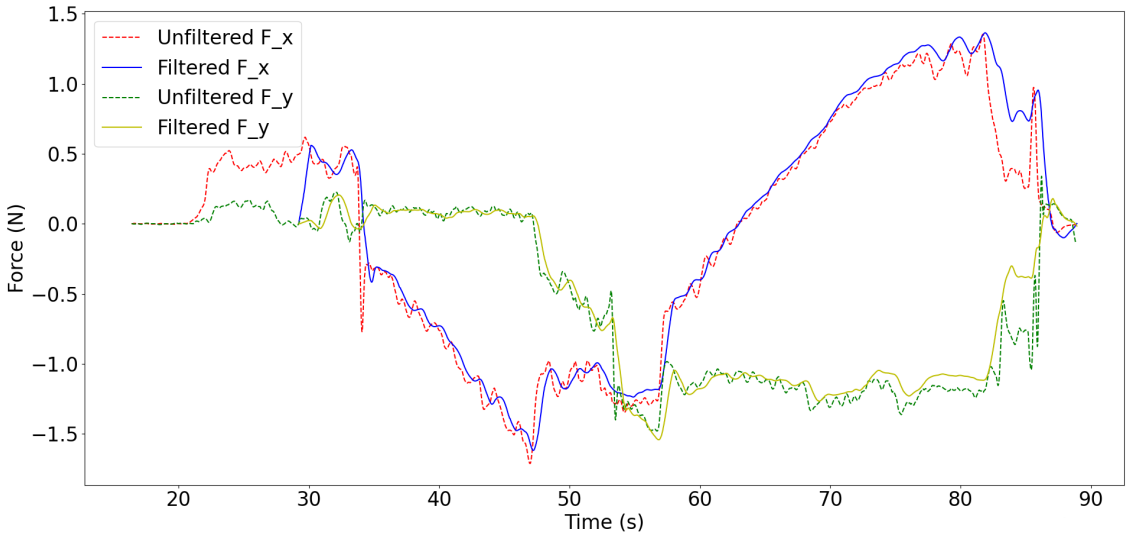


Figure 10. Moving Average Filter for Wrench Estimate

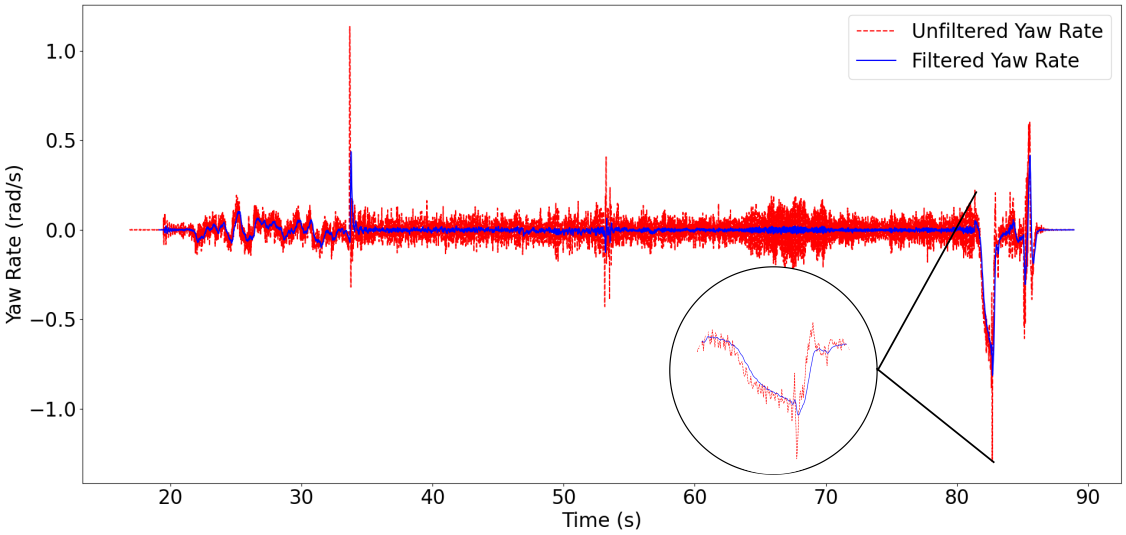


Figure 11. Low Pass Filter for Yaw Rate

The flight controller outputs yaw angles in quaternion representation, but it requires the yaw set point to be specified in Euler angles. One issue with Euler angles is that they are susceptible to a problem known as "gimbal lock" when the angle is π as it switches between $-\pi$ and π at that point. To avoid this problem, the yaw set points are first converted to the $[0, +2\pi]$ range before being published to the flight controller.

3.1.2 State Machine Model

The exploration algorithm employs a state machine to navigate across the environment. The following subsections brief on the working of the state machine and its states. The model uses an explore-and-exploit strategy, taking advantage of the collision-resilient design and the interaction controller [10] to perform tactile-based navigation Figure 12 shows the overview of the state machine. The state-machine com-

prises three primary states, namely exploration, tactile-turning, and tactile-traversal corresponding to $\Gamma \in \{1, 2, 3\}$ respectively.

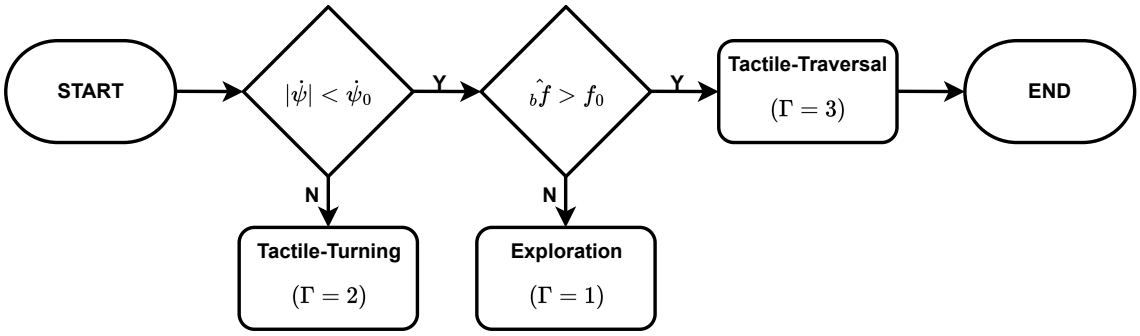


Figure 12. State Machine Model

3.1.2.1 Exploration State

The Exploration State, denoted by $\Gamma = 1$, enables the drone to move freely in the environment until it comes in contact with an obstacle. To achieve this, the drone generates a trajectory along the positive X-axis in the body frame until it reaches a threshold of $\boldsymbol{f}_0$ in either axis or exceeds the limit of ψ_0. The set points generated are then converted into the world frame using Equation (3.10) and published to the flight controller. Algorithm 1 provides an overview of the process. In this state, the yaw admittance controller[6] generates the yaw set points, and the controller uses the current yaw as input and generates the desired yaw to comply with external disturbances and is given by the Equation (3.7) where ψ is the current yaw and ψ_d is the desired yaw. This allows the drone to conform to the obstacle, which is crucial since it is always desirable for the drone to make two-point contact with the obstacle. Depending on the trigger condition of $\dot{\psi}_0$ or $\boldsymbol{f}_0$, the drone switches to the Tactile-Turning or Tactile-Traversal state, respectively.

$$M_\psi \ddot{\psi}_d + D_\psi \dot{\psi}_d + K_\psi \psi_d = \psi \tag{3.7}$$

Algorithm 1: Exploration

Input: $\dot{\psi}$ & ${}_b\hat{\boldsymbol{f}}$
Output: $[x_{sp_{body}}\ \psi_{sp}]^T$ & Γ

if $|\dot{\psi}| < \dot{\psi}_0$ & $|{}_b\hat{\boldsymbol{f}}| < \boldsymbol{f}_0$ **then**
    $X_{sp}(k) = x + d_{step}$
    $Y_{sp}(k) = Y_{sp}(k-1)$
    $\psi_{sp} = \psi_{des}(\psi)$
else if $|\dot{\psi}| > \dot{\psi}_o$ **then**
    $\Gamma = 2$
else
    $\Gamma = 3$

3.1.2.2 Tactile Turning State

The Tactile Turning State, denoted by $\Gamma = 2$ depicted in Algorithm 2, performs a controlled maneuver when its yaw rate exceeds $\dot{\psi}_0$. The algorithm continuously monitors $\dot{\psi}$ to detect when the threshold has been reached. Once this happens, the state machine generates yaw set points at a fixed rate of $\dot{\psi}_c$, allowing the drone to yaw about the point in a controlled manner and attempt to establish contact with an adjacent obstacle. The threshold $\dot{\psi}_0$ is chosen high enough such that low yaw rates do not trigger the control yaw generation. This situation can occur, for example, when the drone slides along a wall and pivots around an edge. As the drone is in yaw admittance, it can freely spin about that point and cause the yaw rate to spike.

The generation of yaw is limited to a rotation angle of approximately 180 degrees to prevent indefinite yaw generation, which would cause the drone to spin indefinitely around that point, which is not desirable. It is assumed that the drone will come into

contact with an adjacent obstacle or the same obstacle from which it got released. While in Tactile Turning State the $\hat{\boldsymbol{f}}$ is compared with $\boldsymbol{f}_{\psi_0}$ and upon reaching it, it will switch to the Tactile-Traversal state and continue exploring the environment. $\boldsymbol{f}_{\psi_0}$ is set slightly higher than $\boldsymbol{f}_0$ to ensure that the contact is registered in the direction upon switching states. The primary objective of this state is to move around corners and edges in a fluidic manner so it assists the map framework to generate a continuous map without any discontinuities.

Algorithm 2: Tactile Turning

Input: $\dot{\psi}$ & ${}_b\hat{\boldsymbol{f}}$
Output: $[x_{sp_{body}}\ \psi_{sp}]^T$ & Γ

if $|\dot{\psi}| > \dot{\psi}_o$ & $|{}_b\hat{\boldsymbol{f}}| < \boldsymbol{f}_{\psi_0}$ **then**
    $x_{sp}(k) = x_{sp}(k-1)$
    if $\dot{\psi} > \dot{\psi}_o$ **then**
        $\psi_{sp}(k) = \psi_{sp}(k-1) + \dot{\psi}_c dt$
    else
        $\psi_{sp}(k) = \psi_{sp}(k-1) - \dot{\psi}_c dt$
else
    $\Gamma = 3$

3.1.2.3 Tactile Traversal State

The Tactile-Traversal state, which enables the drone to adhere to the obstacle and move along it, is presented in Algorithm 3. The interaction controller[10] ensures that the drone maintains contact with the obstacle, while the trajectory generator guides the drone across the obstacle. This state is primarily used for traversing over flat obstacles.

When there is contact with the obstacle (i.e.) if ${}_b\hat{\boldsymbol{f}}$ is greater than either $\boldsymbol{f}_0$ or $\boldsymbol{f}_{\psi_0}$, the state-machine switches to this state. Once the threshold is reached, the

drone must move perpendicular to the obstacle in either the left or right direction. To maintain consistency, the algorithm defaults to the right-hand side of the surface normal of the obstacle. The trajectory generation relies mainly on two parameters: the Contact Normal Direction $\boldsymbol{C_n}$ and the Move Direction λ, which is updated based on the direction of the collision. $\boldsymbol{C_n}$ is a 4x1 vector that indicates the direction of contact and has four elements corresponding to the positive X, negative X, positive Y, and negative Y directions respectively in the body frame. The vector is represented by Equation (3.8).

$$\mathbf{C_n} = [+X+, -X, +Y, -Y] \tag{3.8}$$

The binary value determines the elements of $\boldsymbol{C_n}$ based on the collision direction. Subsequently, λ is updated based on $\boldsymbol{C_n}$. As previously mentioned, λ points towards the right side of the obstacle and is perpendicular to the obstacle's normal direction. λ is represented by 0, 1, 2, 3 which correspond to the movement directions positive X, negative X, positive Y, and negative Y respectively. All the directions are in the body frame since the computations are done in the body frame and then transformed into the inertial frame. The trajectory generation is divided into two parts, where the interaction controller generates trajectory set-points about the contact normal direction to maintain contact with the obstacle and apply the desired force, f_{des}, and the set-points about the λ direction are generated at intervals of d_{step} distance from the current location. The interaction controller is given by the Equation (3.9).

$$M_x\ddot{x}_{set} + D_x\dot{x}_{set} + K_x x_{set} = f_{des} - \hat{f} \tag{3.9}$$

If the drone experiences contact in two directions when the obstacle are adjacent to each other, for example, the corners of a room, the algorithm updates $\boldsymbol{C_n}$ for the

latest collision direction and then updates λ accordingly to generate the trajectory. This prevents the drone from getting stuck at a point in such corner cases.

The algorithm is an example of frontier exploration since it explores the environment by moving step by step and doesn't have knowledge beyond the point of exploration. So, we limit the trajectory generation also to the frontier region which is radially limited by distance d_{step} from the current location, which is represented by the Equation (3.11) and shown in Figure 13. Throughout the experiments, the height of the drone is fixed as this work focuses on the 2D exploration of the environment, and the constant set point is published to the flight controller.

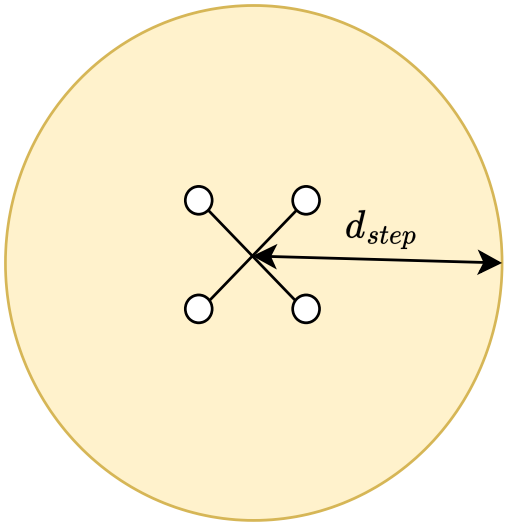


Figure 13. Frontier Representation.

3.1.2.4 Trajectory Transformation

The state machine generates the trajectory in the body frame, this is done in order to assist the frontier-based exploration. The frontier distance (d_{step}) is with respect to the body center and the new trajectory should be generated respectively. This is

Algorithm 3: Tactile Traversal

Input: $\dot{\psi}$ & ${}_b\hat{\boldsymbol{f}}$
Output: $[x_{sp_{body}}\ \psi_{sp}]^T$ & Γ

Function `CollisionNormal()`:
  if $\lambda = +X or -X$ **then**
    if ${}_b\hat{\boldsymbol{f}}_x > \boldsymbol{f}_{0x}$ **then**
      $\boldsymbol{C_n} = [1,0,0,0]$
    else if ${}_b\hat{\boldsymbol{f}}_x < -\boldsymbol{f}_{0x}$ **then**
      $\boldsymbol{C_n} = [0,1,0,0]$
  else if $\lambda = +Y or -Y$ **then**
    if ${}_b\hat{\boldsymbol{f}}_y > \boldsymbol{f}_{0y}$ **then**
      $\boldsymbol{C_n} = [0,0,1,0]$
    else if ${}_b\hat{\boldsymbol{f}}_y < -\boldsymbol{f}_{0y}$ **then**
      $\boldsymbol{C_n} = [0,0,0,1]$
  return $\boldsymbol{C_n}$

Function `MoveDirection()`:
  $\boldsymbol{C_n} \leftarrow$ ContactNormal()
  if $C_n == [1,0,0,0]$ **then**
    $\lambda = +Y$
  else if $C_n == [0,1,0,0]$ **then**
    $\lambda = -Y$
  else if $C_n == [0,0,1,0]$ **then**
    $\lambda = -X$
  else if $C_n == [0,0,0,1]$ **then**
    $\lambda = +X$
  return λ

Function `TrajectoryGeneration()`:
  $\lambda \leftarrow$ MoveDirection()
  $\psi_{sp} = \psi_{des}(\psi)$
  if $\lambda = +X$ **then**
    $X_{sp}(k) = +d_{step}$
    $Y_{sp}(k) = Y_{des}(f_{des})$
  else if $\lambda = -X$ **then**
    $X_{sp}(k) = -d_{step}$
    $Y_{sp}(k) = Y_{des}(f_{des})$
  else if $\lambda = +Y$ **then**
    $X_{sp}(k) = X_{des}(f_{des})$
    $Y_{sp}(k) = +d_{step}$
  else if $\lambda = -Y$ **then**
    $X_{sp}(k) = X_{des}(f_{des})$
    $Y_{sp}(k) = -d_{step}$

given by the Equation (3.10) and ${}^{w}_{b}\boldsymbol{R}$ is the transformation matrix from Equation (3.1).

$$[x_{sp}\ y_{sp}]^T = {}^{w}_{b}\boldsymbol{R}[x_{sp_{body}}\ y_{sp_{body}}]^T \tag{3.10}$$

As mentioned previously, the exploration algorithm is frontier based and the generated trajectory horizon should be limited by the frontier distance. This preventive check helps the drone to maintain a controlled trajectory and explore within a bounded region. This is achieved by bounding the generated set-points as shown in the Equation (3.11). Figure 13 shows the region where the set points would be generated with respect to the current position of the drone.

$$[x_{sp_{body}}\ y_{sp_{body}}]^T = [x \pm d_{step}\ y \pm d_{step}]^T \tag{3.11}$$

3.2 Mapping Framework

In this section, we discuss the mapping framework and its requirements, which enable the generation of a synthesized obstacle map by exploiting the ability of XPLORER to explore the environment through contact. The synthesized map can then be utilized for motion planning by the drone or other autonomous robots. The generated map can also be used to estimate the dimensions and location of obstacles and other applications. The generated map is almost analogous to the map generated by LiDAR, which allows data fusion to synthesize an accurate map. The novelty of the framework is it is not limited to collision-resilient drones but can be easily adapted to rigid drones also to carry out tactile-mapping.

3.2.1 Requisites for the algorithm

The mapping framework utilizes the CAD model of the drone, the wrench acting on the body, and the pose of the drone as the predominant inputs to generate the obstacle map. The CAD model of the drone is used to find the bounding box of the drone which is in turn used to generate the obstacle boundary whilst in contact. In the case of collision-resilient drones such as XPLORER, the CAD model is dynamic, (i.e.) it incorporates the arm bending angles and bounding box changes upon collision. Whereas for rigid drones the bending angles are constant throughout and don't impact the workflow. The external wrench, $\hat{\boldsymbol{\tau}}$ comprises of both the force, $\hat{\boldsymbol{f}}$ and the torque, $\hat{\boldsymbol{m}}$ but the algorithm only uses the force acting on the X-Y plane in the body frame. Similar to the navigation algorithm described in Chapter 4 the forces need to be transformed into the body frame using Equation (3.3) with ${}^{b}_{w}\boldsymbol{R}(\boldsymbol{\psi})$. The state estimate of the drone is directly obtained from the flight controller and used in the framework for the map generation, the position information and yaw of the drone are the primary states used since the scope of the work concentrates on the 2-D map generation of the environment.

The assumption for the mapping framework is that the state estimate of the drone is accurate. This is considered since inertial odometry by itself is a separate research topic and is beyond the scope of this work. The modularity of this framework allows it to be used with any state estimation algorithms as it is independent of the estimation techniques, but it is to be noted that errors in the state estimation will propagate into the generated map.

3.2.2 Point Cloud Generation

The map generation uses point cloud data to generate the obstacle map. In order to do that a predefined obstacle point cloud data is needed. The obstacle is a block of size 0.25 m $\times$ 0.08m $\times$ 0.5 m is used in this case. The block was designed in Autodesk Fusion 360 and the point cloud was generated using Cloud Compare. Based on multiple iterations it was found that 10,000 points was optimal for the map size and the resolution. This obstacle block will be used in the framework for map generation. The framework also uses another block called the corner block, the reasoning will be discussed in the later section, this block consists of 30,000 points because of the larger size but the density is almost the same as the corner block.

The framework uses the Open 3D library[45] for point cloud processing. It is a modern open-source library for 3D data processing, including 3D geometry processing, 3D visualization, and deep learning on point clouds. It is developed by a team of researchers from around the world and provides a powerful and easy-to-use platform for working with 3D data. Some of the key features of Open3D include support for a wide range of 3D data formats, powerful visualization tools for exploring 3D data, efficient algorithms for processing large-scale point clouds, and integration with popular deep learning frameworks like TensorFlow and PyTorch. Open 3D was preferred over Point Cloud Library since it has a better-integrated visualization, has better community support, and is being actively developed. The complete point cloud map generated is stored in Polygonal File Format (.ply) as per the ASCII formatting standards so that it can be utilized by other libraries and software.

3.2.3 Algorithm

The mapping framework uses a similar state-machine algorithm to generate the obstacle boundary map. Algorithm 4 shows the overview of the working. As mentioned previously the algorithm requires the pose, wrench in the body frame, and the CAD model of the vehicle, apart from that it also requires the Contact Normal Direction, $\boldsymbol{C_n}$, and the Move Direction, λ. The mapping framework runs as a separate node and subscribes to the topics from the flight controller and the Navigation State Machine.

We define the mapping wrench threshold as $\boldsymbol{f}_{map}$, if the body wrench, ${}_b\hat{\boldsymbol{f}}$ is greater than the threshold, the mapping starts. $\boldsymbol{f}_{map}$ is taken slightly higher than $\boldsymbol{f_{des}}$, this is done to ensure that mapping is only started when the drone is in firm contact with the obstacle. As an additional check is added, the mapping only initiates if the drone is armed.

Algorithm 4: Mapping Framework

Input: $[x, y, z]^T$, ψ , ${}_b\hat{\boldsymbol{f}}$, $\boldsymbol{C_n}$ & λ
Output: *Point Cloud Data*

```
if ₙf̂ > f_map then
    if λ_prev ≠ λ then
        Add Corner Block to Point Cloud
    else
        Add Obstacle to Point Cloud in C_n axis
else
    Store Point Cloud Data
```

Once the mapping is started, $\boldsymbol{C_n}$ gives the direction of the obstacle. The obstacle block is added to the point cloud in the direction of the obstacle at an offset of 0.21 m. The offset distance is derived from the CAD model, the edge of the bounding box of the XPLORER is 0.21 m away from the center of gravity as shown in Figure 14.

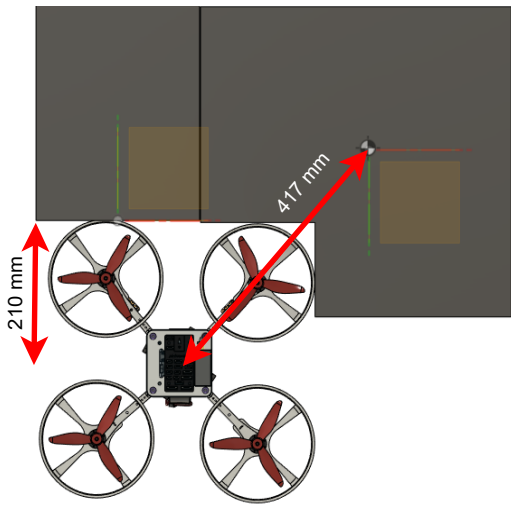


Figure 14. Offset Distance Representation.

There are cases when $\boldsymbol{C_n}$ changes to the adjacent direction, there would be a discontinuity in the map generated if only the obstacle block is used for map generation. For this particular case, we use a custom point cloud called the corner block. The corner block is a right-angled block which is a combination of multiple obstacle blocks and visually provides continuity in the map generated. As mentioned the corner block is inserted in the direction bisecting the two contact normal directions. For example, if $\boldsymbol{C_n}$ switches from positive X to positive Y, the corner block would be added in between them at an offset distance of 0.417 m, it is represented in Figure 14.

For the scope of the thesis, the map generation is done in real-time, but it is not utilized for any real-time path planning tasks. The map generated is stored upon the termination of the algorithm in the .ply format. The map is used by Collision-Aware TrAjectory plAnNer(CATAAN) [10] for generating collision-inclusive trajectories.

Chapter 4

EXPERIMENTS AND RESULTS

The Navigation algorithm and the Mapping framework explained in the previous chapter are implemented using XPLORER and tested in two different scenarios to validate it. This chapter explains the hardware used for experiments, the experimental setup for the two scenarios, and the accuracy of the generated map.

4.1 Hardware Setup

The experiment was conducted in an indoor motion capture space at Robotics and Intelligent Systems Laboratory, ASU consisting of 10 cameras (OptiTrack, NaturalPoint Inc, OR) for obtaining the localization data and 3D pose estimation of XPLORER. Raspberry Pi 4 was used as a companion computer to communicate with the PIXHAWK 4. ROS2-RTPS bridge is used to communicate between the two devices, the ROS2's DDS middleware allows offloading the computation onto a powerful computer on the network, which was configured with AMD Ryzen 5 CPU with 16 Gigabytes of RAM. The motion capture computer, the Raspberry Pi, and the powerful computer were all connected on the same network over the 5Ghz band or via Ethernet. The setup also consists of four 9-DOF IMUs (BNO055, Adafruit, New York, NY) each on one arm of XPLORER to estimate the arm angle. The overview of the hardware setup is shown in Figure 15

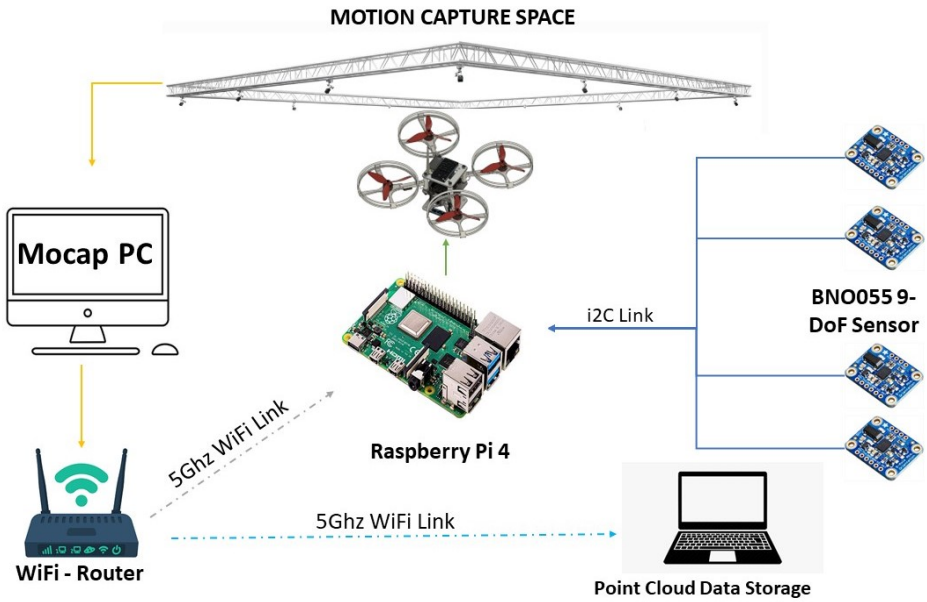


Figure 15. Hardware Setup.

4.2 Environment Setup

As aforementioned, the two distinct environments were constructed to validate the Navigation algorithm and the Mapping framework. Figure 16 & 17 show the environment with concave and convex corners and box-like structure respectively. The walls are acrylic panels, and the reason for using acrylic panels is to assist in the tracking of the drone by the motion capture cameras through the walls and also to minimize friction. The acrylics were supported using wooden structures so that it remains stable upon collisions.

The environment with concave and convex corners consists of three linear segments and two corners. The longest wall measures 1.22 m and the other two walls measure 1.00 m each, and all three are 0.90 m high. One of the corners is a right-angled concave corner and the other is at 120 degrees, which represents an obtuse-angled corner.

The environment with a box-like structure consists of four walls, each aligned

perpendicular to the adjacent wall. The walls measured 1.22 m $\times$ 1.00 m and were 0.9 m high. The adjacent walls were interconnected using masking tapes to prevent them from wobbling upon and release of contact.

Apart from these two environments, another test case was set up to validate the resolution of mapping. This environment consisted of slots between walls, this was done to study the behavior of the drone while encountering such scenarios. Three test cases were considered, one where the slots are larger than the XPLORER's width as shown in Figure 18 (a), another one where the slot size was equal to the diameter of the propeller guard as shown in Figure 18 (c), and finally a case where the slot size is smaller than the diameter of the propeller guard as shown in Figure 18 (b). These experiments help to understand the capability of the navigation algorithm and the resolution of the mapping framework.

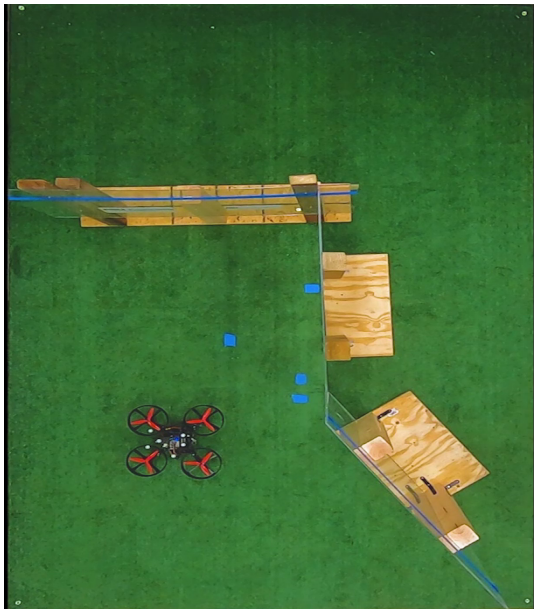


Figure 16. Environment with Concave and Convex Corners.

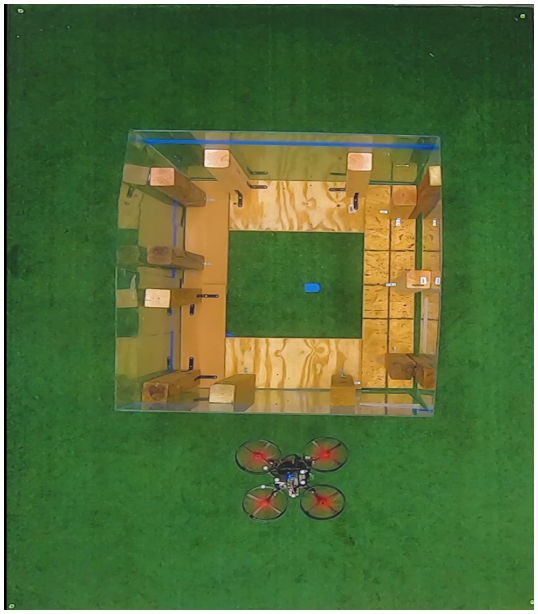


Figure 17. Environment with Box-Like structure.

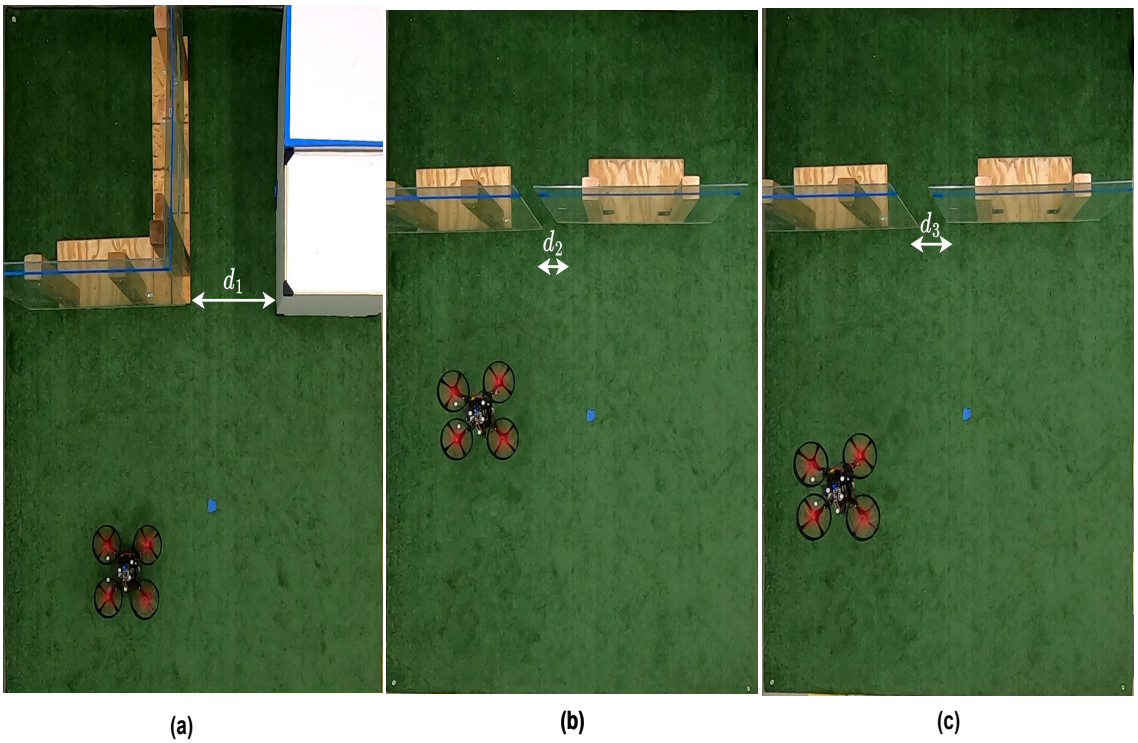


Figure 18. Environment with Slot Gaps.

4.3 Parameters Selection

There are several parameters that are used in the Navigation algorithm and the Mapping framework, these need to be fine-tuned such that the optimal performance is extracted. After extensive testing, the values for the parameters were selected and can be seen in Table 1.

Based on a series of experiments carried out, the yaw rate registered when the drone yaws about a corner is about 0.45 rad/s. So, to detect the yawing the threshold is chosen to be 0.4 rad/s. For the Tactile-Turning state, the yaw generation is done at 0.26 rad/s to generate a controlled yaw. The force threshold for contact detection and for the Tactile-Turning state is 1.5 N and 1.6 N respectively. In the Tactile-Traversal state, the force required to be applied on the obstacle to maintaining contact is taken to be 1.25 N, even though the XPLORER is capable of applying larger forces, we limit it considering the flight time. As briefed in the previous chapter, the algorithm used a frontier-based exploration approach, and the frontier distance is taken to be 0.25 m to restrict the movement in the environment. For the Mapping framework, the force required to start the mapping is taken to be 1.51 N, this is slightly higher than δ_0 to ensure contact is established with the obstacle, and thereafter mapping starts.

Table 1. Parameters & Thresholds used in Experiments

Symbol	**Threshold Value**
$\dot{\psi}_0$	0.4 rad/s
$\dot{\psi}_c$	0.26 rad/s
δ_0	1.5 N
δ_{ψ_0}	1.6 N
d_{step}	0.25 m
f_{des}	1.25 N
δ_{map}	1.51 N

4.4 Environment with Concave and Convex Corners

This environment was constructed to validate the XPLORER's ability to traverse across straight, concave, and convex walls. This scenario is almost a representation of an environment in which the drone could be deployed to carry out exploration tasks. The right-angled concave corner represents a typical corner of a room, this is important to be validated since most of the dilapidated buildings will have such corners. The other corner is an obtuse-angled one and this is used to verify if the drone is able to maintain contact in such scenarios, since it shouldn't drift away upon encountering such corners. The results of the Navigation algorithm and the maps generated for this environment are discussed in the next section.

4.4.1 Experimental Results

XPLORER takes off and starts to hover at a height of 0.7 m, thereafter the state machine switches to Exploration state($\Gamma = 1$) and the drone starts to move forward until it makes contact with the wall. The drone starts to experience a wrench in the negative X direction and when the external wrench threshold, δ_0 is reached it switched to Tactile-Traversal state($\Gamma = 3$). Also at this point once the mapping threshold, δ_{map} is reached the mapping is started. The interaction controller applies f_{des} in the positive X direction to maintain contact with the obstacle and starts to move in the positive Y direction till it reaches the right-angled corner. At that point, the wrench starts to increase in the negative Y direction, once it reaches the threshold, the move direction, λ switches to the negative X direction and the drone starts to apply force in the negative Y direction. Whilst the drone is moving in negative X direction and

reaches the obtuse-angled corner, the interaction controller ensures that the drone maintains contact in the negative Y direction. As the drone reaches the end of the environment, the flight is terminated. The motion of the XPLORER can be seen in Figure 20 (a) indicated by the white arrows and Figure 20 (b) shows the map generated for the environment. In this experiment, only two states corresponding to Γ =1 and 3 are activated as it was sufficient to navigate the environment and it is shown in Figure 19.

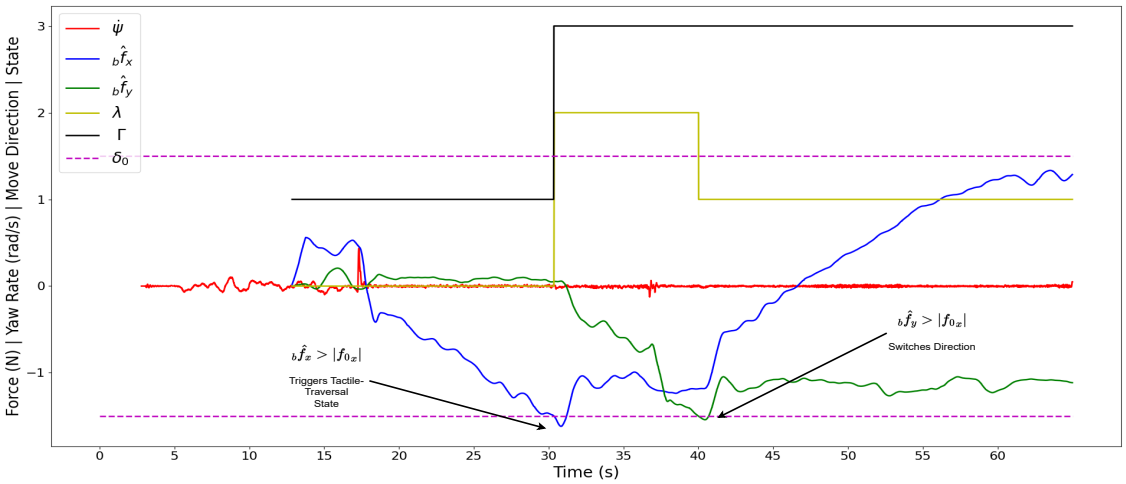


Figure 19. Experimental Results for the Environment with Concave & Convex Corners.

4.5 Environment with Box Obstacle

This environment was designed to validate the XPLORER's ability to turn across corners and continue exploration and also to achieve loop closure (i.e.) the ability to reach the starting point of the exploration. This scenario consists of right-angled corners representing a box, the motive of this environment is to trace the box completely

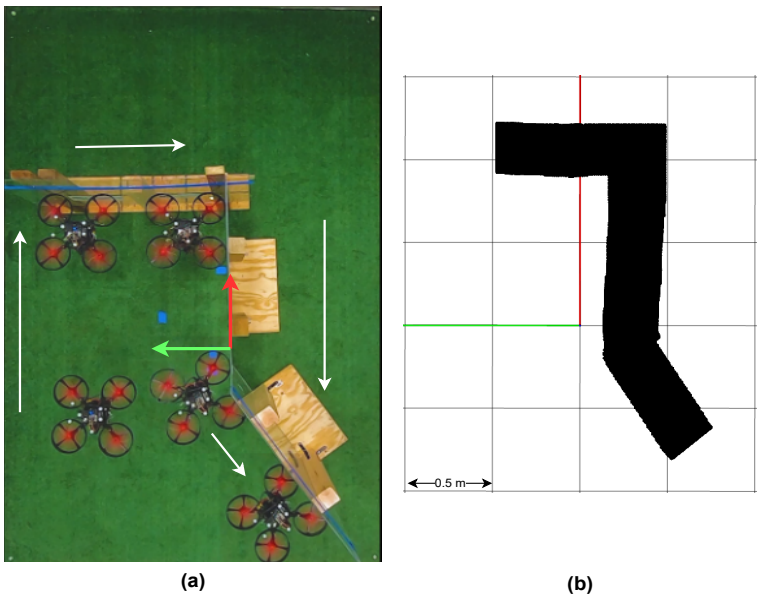


Figure 20. Top View of the Environment with Concave & Convex Corners and the Generated Map.

so that the dimensions of the box can be extracted. This also tests the ability of the navigation algorithm to trigger the tactile-turning state and maintain contact with the obstacle throughout the flight. The results of the Navigation algorithm and the maps generated for this environment are discussed in the next section. It is to be noted that the box's edges are not parallel to the axes can be seen in Figure 21, this was because while setting up the environment it was slightly difficult to exactly orient the edges parallel to the axes.

4.5.1 Experimental Results

XPLORER takes off and hovers at a height of 0.7 m. It then transitions to the Exploration state($\Gamma = 1$), moving forward until it encounters a wall. Upon contact with the wall, the drone experiences a negative X-direction wrench. When the external

wrench threshold, δ_0, is reached, it switches to the Tactile-Traversal state($\Gamma = 3$). Additionally, once the mapping threshold, δ_{map}, is reached, mapping is initiated. The interaction controller applies f_{des} in the positive X direction to maintain contact with the obstacle and begins moving in the positive Y direction until it reaches the corner. Since the interaction controller is applying force and as soon as it reaches the corner the XPLORER is released from contact and yaw about that point. The yaw rate is greater than 0.4 rad/s which triggers the Tactile-turning state($\Gamma = 2$), and the drone exhibits a controlled yaw at a rate of 0.26 rad/s. The drone yaws about that point and established contact with the adjacent wall and starts to exert force, upon reaching the yaw wrench threshold(δ_{ψ_0}) it switches back to Tactile-Turning state, and continues exploration. Similarly, the XPLORER traverses the other three corners and navigates the environment. Upon reaching back to the initial point of contact the navigation algorithm is terminated and so is the mapping. In this case, all three states corresponding to $\Gamma = 1, 2, 3$ are activated to navigate across the environment and it is shown in Figure 21. The generated map enables the measurement of the box dimensions, which are computed as 1.231 m $\times$ 1.019 m, while the actual dimensions are 1.22 m $\times$ 1.0 m. The accuracy for computing the area of the box is approximately 96.72%. For the storage metrics, the framework generated a map that was 1.1 Mb large for a 1 minute and 22 seconds flight, which corresponds to 0.0134 Mb/second. One factor contributing to the mapping error is the imperfect position estimation, which includes small errors from the motion capture system used in our experiment. These errors propagate through the state estimation process and affect the accuracy of the point cloud data.

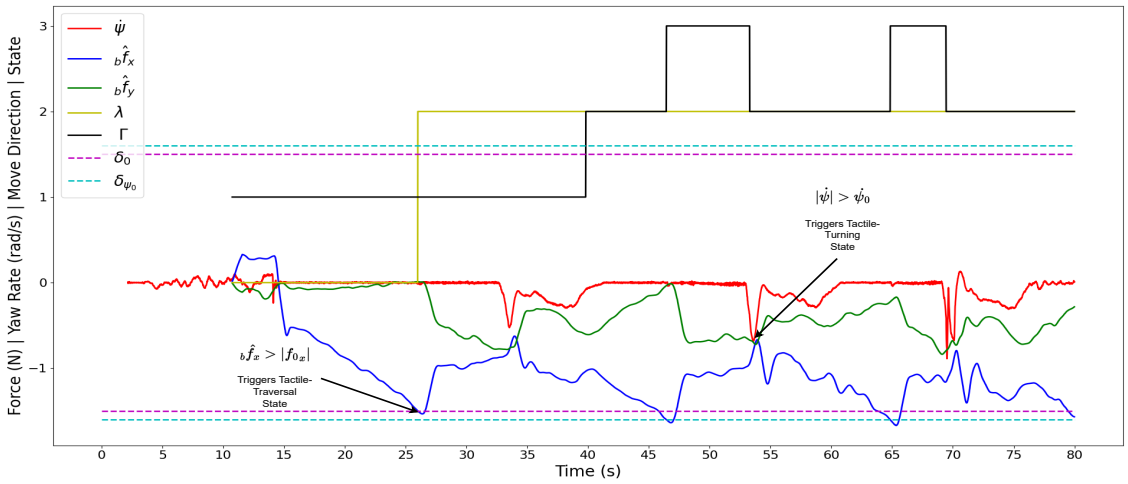


Figure 21. Experimental results for the Environment with Box-like Structure.

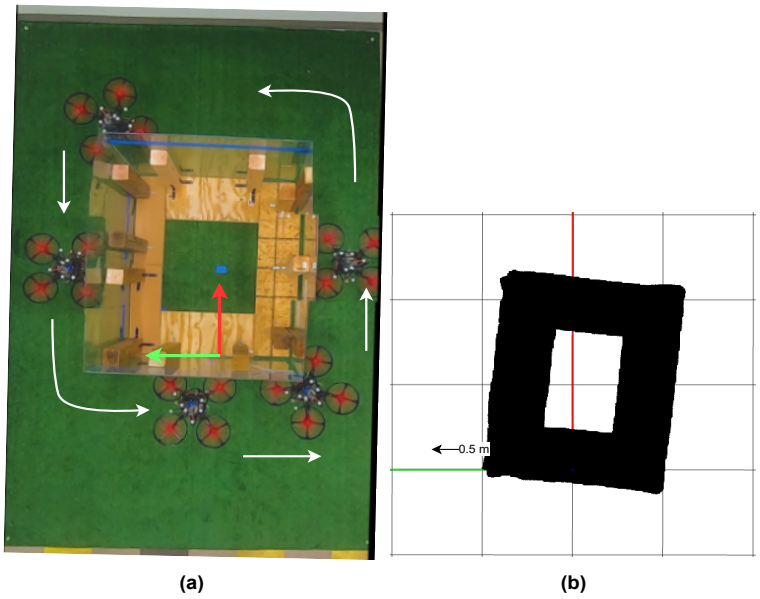


Figure 22. Top View of the Environment with Box-like Structure and the Generated Map.

4.6 Environment with Slots of Different Widths

These sets of test cases were constructed to check the ability of the navigation algorithm to traverse across gaps in the environment. For this, a parameter is

introduced named d_n, which is the width of the gap. There are three cases considered, d_1 where the width of the gap is larger than the XPLORER's width, this is a typical scenario that represents XPLORER escaping from a room or navigating a passageway. The next case d_2 is when the width is smaller than half the diameter of the XPLORER's propeller guard, this is to emulate small gaps that might present in dilapidated buildings. The final test case d_3 is where the width of the gap is equal to that of the propeller guard's diameter, this case is done to validate the behavior of the XPLORER and the navigation algorithm in such cases.

4.6.1 Experimental Results

Similar to previous experiments the XPLORER would take off at the initial location. Upon reaching the hover height it would switch to the Exploration state($\Gamma = 1$) and starts to move in the positive X-direction. Upon making contact with the wall a force is registered in the negative X-direction. Upon reaching the wrench threshold δ_0, it switches to the Tactile-Traversal state($\Gamma = 3$) and starts to move in the positive Y-direction. Based on the slot width d_n the motion of XPLORER was different and can be seen in Figure 23, 24 and 25.

For the first case where the slot width d_1 is larger than the width of the XPLORER, this scenario is almost a subset of the previous experiment. This is done to see how it would behave when encountering corners. This setup also shows a typical scenario where it would explore a room and reaches the exit passageway. The experimental results can be seen in Figure 23, where XPLORER's path is shown. Upon encountering the corner, it starts to yaw and triggers the Tactile-turning state($\Gamma = 2$) and starts to yaw about that point until it hits the yaw wrench threshold(δ_{ψ_0}). Thereafter, it

switches back to the Tactile-Traversal state($\Gamma = 3$) and starts moving in the positive Y-direction in the body frame. This case clearly shows that the navigation algorithm is capable of exiting a room into a passageway and continuing exploration. The map generated whilst exploration is also shown in Figure 23.

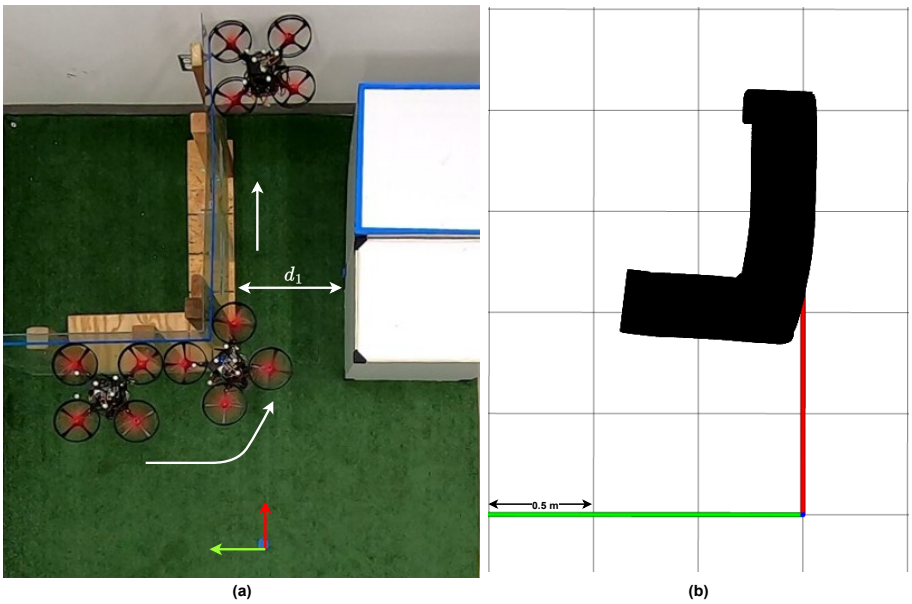


Figure 23. Slot size larger than the width of XPLORER.

The next case is where the slot width d_2 is smaller than half the diameter of the XPLORER's propeller guard. This case is to emulate small gaps that are present in environments. This would represent small gaps present in dilapidated buildings or caves. The experimental results can be seen in Figure 24. Similar to previous experiments XPLORER would take off and would move towards the wall and switches to the Tactile-Traversal state($\Gamma = 3$) and starts to traverse across the wall. When it encounters the gap, it continues to traverse since the propeller guard is larger than the slot width. Even though the gap is smaller than the propeller guard, XPLORER continues to move effortlessly. The gap is registered in the point cloud map and is

shown in Figure 24. It can be inferred from this that even though there is a small kink registered in point cloud data, it does not correspond to an actual gap in the map. This is one of the drawbacks of tactile mapping, the current mapping framework does not localize the collision instead it assumed that there is constant contact between the two propeller guards.

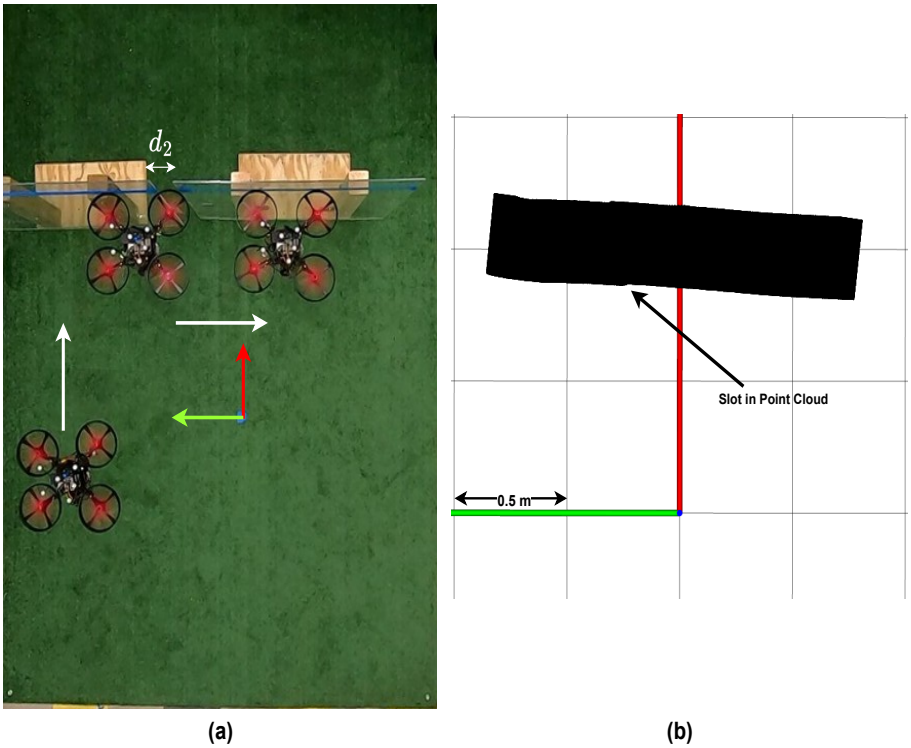


Figure 24. Slot Size Smaller than the Radius of Propeller Guard.

The final case that is considered is when the slot width d_3 is equal to the diameter of the XPLORER's propeller guard. This case was taken into account to study the behavior of XPLORER while encountering such slots. When the XPLORER switches to the Tactile-Traversal state($\Gamma = 3$) it moves across the wall. When it encounters the gap, XPLORER slides in the gap and the arm gets stuck. When it enters the gap, it triggers the Tactile-turning state($\Gamma = 2$) and XPLORER starts to yaw when it is

stuck in the slot. These experimental results are shown in Figure 25. It can be seen that XPLORER slides and yaws and gets stuck in the slot. This shows the drawback of the current navigation algorithm and this is something that needs to be developed upon.

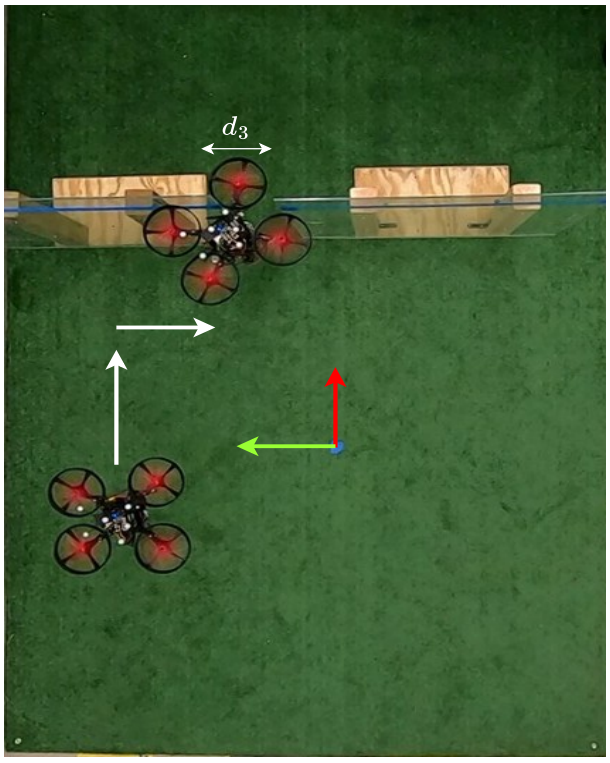


Figure 25. Slot Size Equal to Diameter of Propeller Guard.

Chapter 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion

To conclude, this thesis presents a complete package including a collision-resilient quadcopter, XPLORER, and a Simultaneous Navigation and Mapping algorithm developed for the same. The navigation algorithm and the mapping framework were deployed in XPLORER to validate and the results were discussed.

The design of the quadcopter focuses on modularity and repairability, it consists of modules that can be easily swapped out when the drone crashes or any parts need to be replaced. The use of stiffer torsional springs allows the quadcopter to undergo collisions and also does not require modification of the control allocation matrix within the flight controller since the bending of arms is limited to 30 degrees. The enclosed tower module also helps to protect the electronics from encountering direct collisions hence preventing failure.

The navigation algorithm is developed in such a way it can be deployed in rigid drones also if there is a robust wrench estimate is available. The validation experiments prove the robustness of the algorithm and address most of the corner cases. The frontier-based exploration helps the drone to navigate environments with no prior map of the same since the algorithm does not use the generated map for navigation it reduces the computational complexity.

The mapping framework is designed in such a way as to assist the conventional mapping process. The framework generates point cloud data of the boundaries of the

obstacle, which is similar to the maps generated with LiDAR. This map also helps us to carry out dimensional inspection of the objects as demonstrated in the previous chapter. The accuracy of the map generated turns out to be 96.72% and the storage consumed by the map is also minimal at 30 Hz. The above data demonstrated how the framework can be used for real-time applications.

5.2 Future Work

Currently, the Navigation algorithm and the Mapping Framework heavily rely on the odometry data from the flight controller. The current local position of the drone is obtained from the motion capture system which is significantly accurate compared to other state estimation techniques. Visual inertial odometry[46] is being extensively researched and would be the optimal approach to start with. But, the effect of collisions on such techniques still is a topic to be explored. Contact inertial odometry is done for ground vehicles in [30], a similar approach can be carried out for drones. The development of a novel collision-inclusive state estimation technique would help present a complete product solution along with the Navigation Algorithm and the Mapping Framework.

The mapping framework currently only generates the tactile map of the environment, as mentioned the goal of the framework is to supplement the current state-of-the-art mapping techniques. The generated tactile map can be utilized along with current SLAM algorithms. Authors of [39] present a multi-sensor fusion algorithm to carry out navigation in both indoor and outdoor environments, the presented mapping framework can also be added to a similar algorithm to carry out navigation. This

would be particularly useful when the performance of other visual sensors deteriorates due to the presence of fog or particulate matter.

One major drawback with the current navigation algorithm is shown in Figure 25 where XPLORER gets stuck in singularity when it traverses walls with gaps that are larger than the diameter of the propeller guard. A countermeasure for this case needs to be developed to overcome such scenarios. For instance, if XPLORER is stuck in a position for a long time then it should replan the exploration trajectory and should navigate across.

In the current state even though the Navigation algorithm and the Mapping Framework are capable of being used in any type of drone, validation experiments are required to compare the performance analytics. Since the behavior of conventional drones would be different upon collision and the parameters have to be tuned for each drone.

REFERENCES

[1] T. Rouček, M. Pecka, P. Čížek, T. Petříček, J. Bayer, V. Šalansky, D. Heřt, M. Petrlık, T. Bača, V. Spurny, et al. "Darpa subterranean challenge: Multi-robotic exploration of underground environments". In: *Modelling and Simulation for Autonomous Systems: 6th International Conference, MESAS 2019, Palermo, Italy, October 29–31, 2019, Revised Selected Papers 6*. Springer. 2020, pp. 274–290.

[2] J. Zhang and S. Singh. "LOAM: Lidar odometry and mapping in real-time." In: *Robotics: Science and Systems*. Vol. 2. 9. Berkeley, CA. 2014, pp. 1–9.

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system". In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[4] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. DOI: 10.1017/CBO9780511546877.

[5] M. Kutila, P. Pyyknen, H. Holzhüter, M. Colomb, and P. Duthon. "Automotive LiDAR performance verification in fog and rain". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2018, pp. 1695–1701.

[6] K. Patnaik, S. Mishra, S. M. R. Sorkhabadi, and W. Zhang. "Design and Control of SQUEEZE: A Spring-augmented QUadrotor for intEractions with the Environment to squeeZE-and-fly". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2020, pp. 1364–1370.

[7] P. De Petris, H. Nguyen, M. Dharmadhikari, M. Kulkarni, N. Khedekar, F. Mascarich, and K. Alexis. "Rmf-owl: A collision-tolerant flying robot for autonomous subterranean exploration". In: *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE. 2022, pp. 536–543.

[8] Y. Mulgaonkar, W. Liu, D. Thakur, K. Daniilidis, C. J. Taylor, and V. Kumar. "The tiercel: A novel autonomous micro aerial vehicle that can map the environment by flying into obstacles". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 7448–7454.

[9] P. De Petris, S. J. Carlson, C. Papachristos, and K. Alexis. "Collision-tolerant Aerial Robots: A Survey". In: *arXiv preprint arXiv:2212.03196* (2022).

[10] K. Patnaik, A. A. P. Saravanakumaran, and W. Zhang. "To Collide or Not To Collide–Exploiting Passive Deformable Quadrotors for Contact-Rich Tasks". In: *arXiv preprint arXiv:2305.17217* (2023).

[11] A. Briod, P. Kornatowski, A. Klaptocz, A. Garnier, M. Pagnamenta, J.-C. Zufferey, and D. Floreano. "Contact-based navigation for an autonomous flying robot". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 3987–3992.

[12] D. Fan, K. Guo, S. Lyu, X. Yu, L. Xie, and L. Guo. "Quadrotor UAV: Collision Resilience Behaviors". In: *IEEE Transactions on Aerospace and Electronic Systems* (2022).

[13] Z. Liu and K. Karydis. "Toward impact-resilient quadrotor design, collision characterization and recovery control to sustain flight after collisions". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 183–189.

[14] A. Briod, P. Kornatowski, J.-C. Zufferey, and D. Floreano. "A collision-resilient flying robot". In: *Journal of Field Robotics* 31.4 (2014), pp. 496–509.

[15] C. Papachristos, S. Khattak, and K. Alexis. "Haptic feedback-based reactive navigation for aerial robots subject to localization failure". In: *2019 IEEE Aerospace Conference*. IEEE. 2019, pp. 1–7.

[16] K. Alexis, G. Darivianakis, M. Burri, and R. Siegwart. "Aerial robotic contact-based inspection: planning and control". In: *Autonomous Robots* 40 (2016), pp. 631–655.

[17] J. Zha and M. W. Mueller. "Exploiting collisions for sampling-based multicopter motion planning". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 7943–7949.

[18] T. Tomic, C. Ott, and S. Haddadin. "External wrench estimation, collision detection, and reflex reaction for flying robots". In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1467–1482.

[19] T. Guardian. *Building collapses in Tibet after torrential rain.* 2017. URL: https://www.theguardian.com/global/video/2017/jul/10/building-collapses-in-tibet-after-torrential-rain-video.

[20] S. N. Material. *Visit to the Semipalatinsk Nuclear Test Site .* 216. URL: https://carlwillis.wordpress.com/2012/08/13/visit-to-the-semipalatinsk-nuclear-test-site/7_degelen-002/.

[21] W. V. Unoversity. *Beneath the surface: WVU students cave in Cuba.* 2018. URL: https://wvutoday.wvu.edu/stories/2018/11/16/beneath-the-surface-wvu-students-cave-in-cuba.

[22] BBC. *Thai cave diver relives the extraordinary rescue.* 2018. URL: https://www.bbc.com/news/av/world-asia-44822177.

[23] D. Falanga, K. Kleber, S. Mintchev, D. Floreano, and D. Scaramuzza. "The Foldable Drone: A Morphing Quadrotor That Can Squeeze and Fly". In: *IEEE Robotics and Automation Letters* 4.2 (2019), pp. 209–216. DOI: 10.1109/LRA.2018.2885575.

[24] D. Yang, S. Mishra, D. M. Aukes, and W. Zhang. "Design, planning, and control of an origami-inspired foldable quadrotor". In: *2019 American Control Conference (ACC)*. IEEE. 2019, pp. 2551–2556.

[25] N. Bucki and M. W. Mueller. "Design and control of a passively morphing quadcopter". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 9116–9122.

[26] K. Patnaik, S. Mishra, Z. Chase, and W. Zhang. "Collision recovery control of a foldable quadrotor". In: *2021 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE. 2021, pp. 418–423.

[27] P. De Petris, H. Nguyen, M. Kulkarni, F. Mascarich, and K. Alexis. "Resilient collision-tolerant navigation in confined environments". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 2286–2292.

[28] K. Patnaik and W. Zhang. "Adaptive Attitude Control for Foldable Quadrotors". In: *IEEE Control Systems Letters* (2023).

[29] P. H. Nguyen, K. Patnaik, S. Mishra, P. Polygerinos, and W. Zhang. "A soft-bodied aerial robot for collision resilience and contact-reactive perching". In: *Soft Robotics* (2023).

[30] T. Lew, T. Emmei, D. D. Fan, T. Bartlett, A. Santamaria-Navarro, R. Thakker, and A.-a. Agha-mohammadi. "Contact inertial odometry: collisions are your

friends". In: *Robotics Research: The 19th International Symposium ISRR*. Springer. 2022, pp. 938–958.

[31] N. Khedekar, F. Mascarich, C. Papachristos, T. Dang, and K. Alexis. "Contact–based navigation path planning for aerial robots". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 4161–4167.

[32] L. M. Gonzalez de Santos, E. Frias Nores, J. Martinez Sanchez, and H. Gonzalez Jorge. "Indoor path-planning algorithm for UAV-based contact inspection". In: *Sensors* 21.2 (2021), p. 642.

[33] J. L. Schonberger and J.-M. Frahm. "Structure-from-motion revisited". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4104–4113.

[34] W. Zhen, Y. Hu, H. Yu, and S. Scherer. "LiDAR-enhanced structure-from-motion". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 6773–6779.

[35] A. J. Davison. "Real-time simultaneous localisation and mapping with a single camera". In: *Computer Vision, IEEE International Conference on*. Vol. 3. IEEE Computer Society. 2003, pp. 1403–1403.

[36] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. "ORB-SLAM: a versatile and accurate monocular SLAM system". In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[37] J. Zhang and S. Singh. "LOAM: Lidar odometry and mapping in real-time." In: *Robotics: Science and Systems*. Vol. 2. 9. Berkeley, CA. 2014, pp. 1–9.

[38] D. Cattaneo, M. Vaghi, A. L. Ballardini, S. Fontana, D. G. Sorrenti, and W. Burgard. "Cmrnet: Camera to lidar-map registration". In: *2019 IEEE intelligent transportation systems conference (ITSC)*. IEEE. 2019, pp. 1283–1289.

[39] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar. "Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 4974–4981. DOI: 10.1109/ICRA.2014.6907588.

[40] D. Saldana, B. Gabrich, G. Li, M. Yim, and V. Kumar. "Modquad: The flying modular structure that self-assembles in midair". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 691–698.

[41] G. Li, B. Gabrich, D. Saldana, J. Das, V. Kumar, and M. Yim. "ModQuad-Vi: A vision-based self-assembling modular quadrotor". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 346–352.

[42] F. Shi, M. Zhao, T. Anzai, X. Chen, K. Okada, and M. Inaba. "External wrench estimation for multilink aerial robot by center of mass estimator based on distributed IMU system". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 1891–1897.

[43] F. Shi, M. Zhao, M. Murooka, K. Okada, and M. Inaba. "Aerial regrasping: Pivoting with transformable multilink aerial robot". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 200–207.

[44] T. Anzai, M. Zhao, T. Nishio, F. Shi, K. Okada, and M. Inaba. "Fully Autonomous Brick Pick and Place in Fields by Articulated Aerial Robot: Results in Various Outdoor Environments". In: *IEEE Robotics & Automation Magazine* (2023).

[45] Q.-Y. Zhou, J. Park, and V. Koltun. "Open3D: A Modern Library for 3D Data Processing". In: *arXiv:1801.09847* (2018).

[46] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. "Robust visual inertial odometry using a direct EKF-based approach". In: *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2015, pp. 298–304.