Autonomous quadcopter for rapid underground mine mapping and exploration

Nico Epler[1*], and *Callen* Fisher[2]

[1]Department of Electrical and Electronic Engineering, University of Stellenbosch, South Africa, 23910712@sun.ac.za
[2]Department of Electrical and Electronic Engineering, University of Stellenbosch, South Africa, cfisher@sun.ac.za

Abstract. Mine surveying and mapping are time-consuming and hazardous tasks that are critical to most underground mining operations. Recently, the introduction of robotics for underground mine mapping has reduced these safety risks for humans. This project aims to develop an autonomous quadcopter equipped with dedicated sensors capable of performing 3D Simultaneous Localization and Mapping (SLAM) in Global Navigation Satellite System (GNSS) denied environments, such as underground mine sites. Various localization and mapping algorithms were compared and tested, and the SpectacularAI package proved to be a computationally efficient SLAM method that can run on the quadcopter's onboard computer, generating dense and detailed point cloud maps. Additionally, autonomous exploration and path planning methods for three dimensional spaces were studied. Appropriate algorithms were implemented and optimised to develop a software suite that ensures the platform's autonomy in communication-limited subterranean environments. Finally, the SLAM module was tested in a cluttered indoor environment, while the autonomous exploration framework was developed and evaluated in a subterranean simulation environment. Future work will focus on deploying and testing both modules in real-world underground mine scenarios.

1 Introduction

Given the significant human health and safety hazards in underground mines—including fires, rock falls, gas leaks, and floods [1]—robots have recently been deployed to assist with surveying, environmental monitoring, and exploration. Such robotic platforms include autonomous rovers for underground 3D mapping using lidar sensors [2] and composite systems like the Rhino, which utilise lidar and RGB-D fusion for underground localisation and mapping [3].

More recent approaches extend these capabilities through multi-robot exploration, mapping, and terrain perception by networking diverse platforms such as quadrupeds, rovers, and Unmanned Aerial Vehicles (UAVs), as depicted in **Fig. 1** [4]. UAVs, in particular, serve

* Corresponding author: 23910712@sun.ac.za
* This research is funded by Enaex Africa

as rapid exploration tools that enable fast data collection and extensive coverage, with most existing UAV platforms relying on lidar sensors to perform mapping and exploration tasks in unknown environments [5].

SLAM processes primarily focus on synchronously mapping environments and accurately localising the robotic platform within these maps. In some scenarios only two-dimensional (2D) SLAM operations are required, such as in warehouse navigation where robotic platforms commonly manoeuvre in 2D planes [6]. However, in environments such as underground mines, it is imperative for quadcopters to utilise comprehensive three-dimensional (3D) SLAM techniques. This not only provides critical information about ground clearance, roof clearance, and obstacles that pose collision risks in three-dimensional space, but also generates more informative replicas of the environment, which are essential for mine surveying and environmental monitoring missions.

Fig. 1. A multirobot team capable of performing exploration, mapping and terrain perception consisting of (A-C) multiple ANYmal C quadrupeds, (D) a tethered rover acting as a communication hub, (E) a medium sized UAV called Alpha Aerial Scout and (F) a small-scale UAV named RMF-Owl [4].

To successfully accomplish exploration tasks, platforms require strategically determined waypoints that guide them to unexplored regions. This can be achieved through teleoperation by an operator, but more modern approaches rely on ground control stations, equipped with autonomous exploration algorithms, to identify unexplored areas. These exploration algorithms work in conjunction with autonomous path-planning techniques to generate waypoint trajectories, which are sent to the UAV for execution [7].

However, subterranean environments present a significant need for autonomy due to limited communication capabilities and intricate layouts that demand low-latency and real-time decision-making [4]. As a result, it is often more practical to perform autonomous exploration and path-planning tasks onboard the robot, eliminating the need for external communication networks. Despite this advantage, quadcopters are subject to payload limitations, which constrain both battery capacity and the size of onboard compute modules. Since extended flight time is critical for autonomous exploration, the systems overall weight must be minimised. This necessitates the use of lightweight, power-efficient computing hardware, which introduces a major limitation in computational power.

The successful implementation of an autonomous exploration UAV is highly valuable, as UAVs can navigate extreme slopes, water pools, tight spaces, and vertical shafts where quadrupeds and rovers often encounter difficulties [5]. Navigating these scenarios requires the UAV to utilise the full three-dimensional space, highlighting the need for advanced 3D exploration and path-planning algorithms. While 2D path-planning techniques have been

widely studied, research on 3D path planning remains relatively limited and is associated with significantly higher algorithmic complexity and computational cost [8]. Similarly, 2D autonomous exploration frameworks are more widespread, due to their broad application in platforms like Unmanned Ground Vehicles (UGVs) and quadrupeds, whereas 3D exploration frameworks are less common [9]. Additionally, most existing 3D exploration algorithms for UAVs have been developed for the ROS1 (Robot Operating System) framework [10-12].

ROS2 introduced substantial improvements over ROS1, including compatibility with embedded systems, superior communication capabilities across diverse networks, improved runtime efficiency for real-time computing and greater scalability, for multi-robot operations [13-14]. Despite these advantages and ROS1 reaching its end-of-life (EOL) in May 2025, a recent study conducted in 2024 revealed that migration to ROS2 remains slow, particularly in academia, highlighting the need for new systems to be developed natively in ROS2 [15].

The aim of this research is the end-to-end design and development of a small-scale, agile quadcopter for underground mine surveying, utilising modern architectures such as ROS2 to address the previously identified shortcomings. The primary objective is to establish a functional skeleton framework for a UAV platform capable of performing onboard SLAM and autonomous exploration, ultimately delivering a comprehensive 3D map of the environment. This modular framework will facilitate future research, allowing for iterative refinements of subsystems and the potential integration into multi-robot fleets, to perform collaborative exploration tasks in complex underground mines.

2 Related work

Recent advances in aerial robotics have demonstrated that drones can reliably navigate and accurately map confined and GPS-denied environments. The Flyability Elios 3, a commercially available drone controlled remotely by an offboard operator, is specifically designed for underground mine inspections and surveying tasks. Equipped with a modular payload bay capable of accommodating various sensor packages, the system has shown a robust performance in generating accurate 3D reconstructions of complex mining tunnels [16]. Furthermore, advancements in photogrammetry have increasingly leveraged offboard processing, enabling the utilisation of more computationally intensive software such as AgiSoft Photoscan (Metashape), PIX4DMapper, and DJI Terra [17]. These tools produce high-resolution point clouds for diverse applications, like mapping forest canopies and industrial sites, and can achieve sub-centimetre accuracy through optimised image capturing techniques and dense matching algorithms.

The introduction of autonomous exploration planners has greatly reduced the need for trained drone operators in exploration missions. FUEL (Fast UAV Exploration) is a ROS1-based framework that implements a Frontier Information Structure (FIS), which incrementally updates and conserves frontier information critical for efficient exploration planning. Leveraging this FIS, a hierarchical planner extracts global exploration paths, after which sets of local viewpoints are refined and minimum-time trajectories are being generated in sequence. This architecture enables FUEL to achieve exploration rates up to 3–8 times faster than conventional greedy methods [10]. Building upon FUEL, RACER (RApid Collaborative ExploRation) scales the approach to a fully decentralised multi-UAV team, while maintaining FUEL's core components [18]. It achieves this by decomposing the map into an H-grid structure, assigning distinct exploration regions to individual UAVs, and balancing workloads using a Capacitated Vehicle Routing Problem (CVRP) formulation. On the other hand, Peacock Exploration introduces a lightweight local planner based on precomputed minimum-snap trajectories, which fan out in a pattern resembling a peacock's tail [12]. These trajectories, combined with an OctoMap, a receding horizon strategy, and heuristic scoring criteria, form an efficient exploration framework with a computational

complexity of only O(log N). ExplorationRRT focuses on a tree-based exploration planning approach that employs random sampling to generate exploration paths, aiming to maximise information gain while minimising travel distance and actuator effort [11].

In literature, OctoMap is a widely used tool for simplifying dense and complex point cloud data to reduce memory usage while still preserving critical environmental information for 3D exploration and path planning activities [12]. This probabilistic mapping framework makes use of an octree data structure to generate memory-efficient 3D occupancy grids, commonly referred to as voxel maps [19]. It explicitly represents every voxel of a predefined size within the 3D space as either free, occupied, or unknown. Additionally, OctoMap supports dynamic updates, making it well-suited for real-time exploration and navigation tasks.

In the work of [20], a 3D voxel map was further utilised to extract the full navigable free space and project it onto a horizontal plane, rather than relying on a single fixed-height slice to obtain a 2D representation of the environment. During this 3D-to-2D map conversion process, safety checks are applied to remove free-space regions smaller than a predefined robot safety margin. Additionally, for each cell in the 2D map, information about the navigable height in 3D and an approximation of the floor slope is retained. This approach is particularly useful for converting 2D paths, planned on the generated 2D map, back into 3D trajectories for UAVs, while also allowing steep slopes to be marked as untraversable in UGV applications. The framework enables computationally efficient navigation by extending algorithms designed for 2D maps to 3D environments and supports faster, more efficient map sharing between agents in bandwidth-constrained scenarios [20].

To generate 2D trajectories many studies, including [11], have employed improved variants of the RRT (Rapidly-exploring Random Tree) algorithm, originally introduced in [21]. The basic RRT algorithm is a sampling-based path planner that incrementally grows a tree by randomly selecting points within the free space of the map and connecting each new point to the nearest node in the existing tree. This approach enables efficient exploration, even in high-dimensional environments [21]. Furthermore, RRT* extends this concept by introducing a rewiring step, where after a new sample point is added to the tree, the algorithm checks whether nearby nodes can be "rewired" through this new point to reduce their total path cost from the origin [22]. This ensures that as the number of iterations approaches infinity, the algorithm converges to the global optimal path. During frontier-based exploration, the rewiring mechanism allows RRT* to rapidly generate an initial feasible path, similar to its predecessor, and then incrementally refine it into shorter, higher-quality trajectories with only a minimal increase in computation per iteration [22].

Frontier-based exploration is a classic method used for autonomous exploration in robotics [10]. Frontiers are defined as the boundaries between known free space and unexplored, unknown regions. Frontier exploration focuses on detecting these boundary areas within the map and navigating toward them to incrementally expand the known environment [23]. This process is repeated until no further frontiers remain, indicating that the environment, bounded by obstacles, has been fully explored.

An example of work with a similar focus on the development of a complete end-to-end UAV system for autonomous exploration is presented in [5]. The RMF-Owl, used by Team CERBERUS during the DARPA Subterranean Challenge, is equipped with a 3D LiDAR and an RGB camera. It presents a fully autonomous exploration system capable of operating in unknown underground environments by computing waypoints onboard, eliminating the need for external operators and reliance on potentially unreliable communication networks. However, some limitations of this system include its use of a relatively heavy, power-intensive, and costly LiDAR sensor, as well as its reliance on ROS1, which limits the systems future scalability and modernisation.

3 Methodology and system design

To develop the proposed platform, a systematic approach was adopted, divided into two primary stages: a simulation setup and a real-world quadcopter implementation. An initial prototype was developed and incrementally refined throughout the development process. Following a successful manual flight control test to verify the platform's basic stability and mapping capabilities, all subsequent subsystems were first validated in simulation, with future work focusing on deploying them onto the physical UAV. This staged methodology ensures a controlled and reliable transition from virtual testing to real-world operation.

3.1 Physical system design

The developed platform, named Void Raven, is a custom-built 5-inch quadcopter designed with components specifically selected to optimise performance, efficiency, and payload capacity, thereby maximising exploration time. As illustrated in **Fig. 2A**, the platform was powered by four iFlight R5 2207 2050KV brushless motors, chosen for their high efficiency at lower RPMs and substantial thrust output [24]. The motors were controlled by a Holybro Tekko32 F4 4-in-1 60A Electronic Speed Controller (ESC) and paired with Gemfan LR 5126 two-blade polycarbonate propellers. Two-blade propellers were selected for their superior aerodynamic efficiency and reduced drag, but at the expense of some added thrust, grip, and flight stability typically provided by three-blade propellers [25].

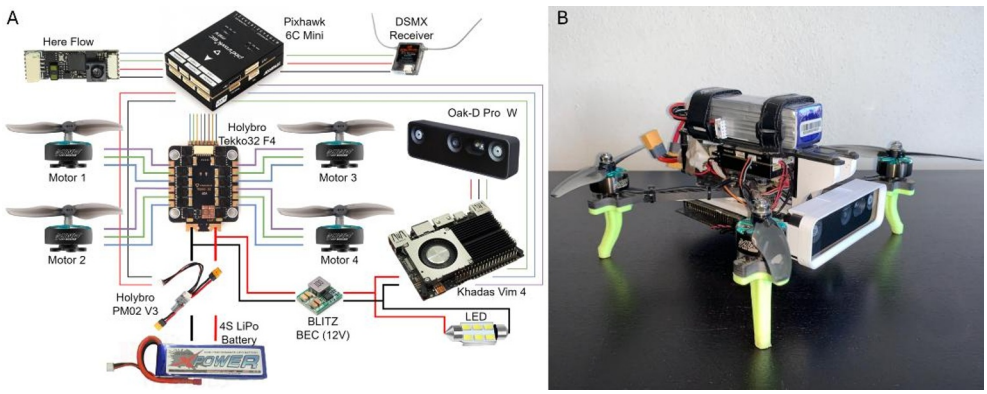

Fig. 2. A) An overview of the Void Raven electronics and **B)** a photograph of the physical development platform.

The quadcopter was equipped with a Pixhawk 6C Mini flight controller running PX4, which handled low-level flight control and provided real-time position estimation using its built-in Extended Kalman Filter (EKF). The flight controller was powered via a Holybro PM02 V3 power module, which received input from a X-Power 2200 mAh 4S1P LiPo battery, with the flexibility to upgrade to a larger battery if needed. Mapping and autonomous navigation were handled by an onboard Khadas VIM4 single-board computer, running Ubuntu 22.04 and ROS2 Humble.

An ARM64-based microcontroller was selected due to its lightweight design, lower power consumption, and higher power efficiency compared to more powerful x86_64-based alternatives [26]. The Khadas VIM4, in particular, was chosen for its relatively powerful CPU and GPU compared to other ARM64 devices, as well as its additional features, including 8 GB of RAM, a fast USB 3.0 port for interfacing with the RGB-D camera, and accessible GPIO pins for easy communication with the flight controller. The onboard computer was

powered through a BLITZ BEC Module V1.1, which offers a wide input voltage range of 6–26 V and was configured to output 12 V at a continuous current of 2 A, with a peak capability up to 3 A. This buck converter was selected to meet the 24W power requirement specified by the manufacturer [27].

To aid with position control in GNSS-denied underground environments, a downward-facing Here Flow optical flow sensor was integrated into the system. Additionally, an OAK-D Pro W RGB-D camera was mounted to provide real-time depth information, generating point clouds for onboard SLAM computation and autonomous exploration. An RGB-D camera was selected over the more commonly used LiDAR due to its significantly lower cost, reduced size and weight, lower power consumption, and its ability to capture additional visual information, including surface texture and colour.

The OAK-D Pro W variant was specifically chosen for its enhanced features, as the "Pro" series includes active stereo dot projection, which improves depth perception on low-texture surfaces, and a built-in infrared illumination LED (Light Emitting Diode), enabling operation in low-light or no-light environments. Furthermore, the "W" (Wide) version offers a broader field of view at 127°, compared to the standard 89.5°, allowing for improved situational awareness. To further support operation in low-light underground conditions, particularly to ensure consistent performance of the optical flow sensor, the quadcopter was also equipped with a 12V LED. A photograph of the integrated platform is shown in **Fig. 2B**.

3.2 Implementation of SLAM module

Careful consideration was required when selecting a suitable SLAM algorithm due to the Khadas VIM4's limited computational power and ARM64 architecture, as most state-of-the-art SLAM solutions are optimised for high-performance AMD64 platforms. After thorough evaluation and testing, the SpectacularAI SDK was selected to perform onboard SLAM. SpectacularAI is a proprietary Visual-Inertial SLAM (VISLAM) package, loosely based on the HybVIO module, and incorporates an integrated loop closure mechanism to enhance long-term localisation accuracy [28]. The SpectacularAI ROS2 wrapper was modified so that its native VIO output, which fuses visual odometry with Inertial Measurement Unit (IMU) data to estimate the camera's position, was published to a dedicated ROS2 topic. This made the data accessible to the flight controller, allowing it to be integrated into the onboard EKF and thereby contributing to a more robust and consistent global position estimate.

After observing that the package was still consuming a significant portion of the onboard computer's computational resources, further investigation revealed that the VIO module was running at an unnecessarily high frame rate of 30 fps. Additionally, every 6th frame was used for point cloud generation, resulting in a relatively high mapping frequency of 5 Hz, with processed images having a resolution of 640×400 px. It was further noted that a significant portion of system computational power was being consumed by the continuous publishing of the full, unfiltered point cloud to ROS2 topics. To overcome this computational burden, several parameters had to be changed in the ROS2 wrapper, which instantiates the SpectacularAI API.

After some testing, it was found that a frame rate of 10 fps was sufficient for stable and responsive SLAM performance and was therefore implemented. This resulted in a satisfactory point cloud mapping frequency of 1.66 Hz. To further optimise performance, the number of points published to the ROS2 topic was reduced. This was achieved by converting the point cloud generated by the SpectacularAI VISLAM package into an Open3D point cloud format, applying voxel down sampling to reduce the point cloud's density, and performing radius-based outlier removal to eliminate isolated noise points. The processed point cloud was then converted back into a NumPy array before being published to the ROS2 topic. This change reduced the number of points published by the module from approximately

9 348 points per message to around 5 074 points per message. Considering the reduced mapping frequency, the total number of points published per second dropped significantly, from about 46 740 points per second to only about 8 457 points per second.

3.3 Simulation setup

Due to the complexity of autonomous navigation and exploration tasks, the high likelihood of severe crashes during physical testing, and the need for faster prototyping, a simulation environment was set up for the development of the autonomous navigation module. Gazebo Sim Garden, running on ROS2 Humble, was selected as the simulation environment, primarily because of its advanced and seamless integration with the PX4-Autopilot software suite. During the setup, most attention was focused on the functionality of the software stack, and since significant sim-to-real transfer differences were expected, it was decided not to create a new simulation model, but rather to use the existing X500_depth quadcopter model available through PX4's Gazebo Sim integration.

To implement a realistic simulation environment, an underground cave setting was required. For this purpose, the Gazebo Fuel website was consulted, and cave models previously used in the DARPA Subterranean Challenge were selected [29]. One of the available underground cave systems, which includes a designated staging area, with the spawned X500_depth quadcopter model is shown in **Fig. 3.**

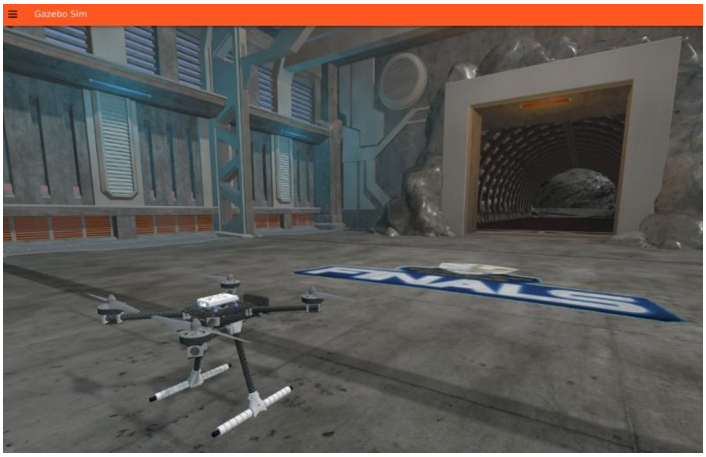

Fig. 3. Image of the standard X500_depth model located inside the staging area of an underground tunnel simulation environment.

By default, the point cloud generated from the OAK-D camera mounted on the X500_depth model was not visible in RViz, as it was only published as a native Gazebo topic. To enable visualisation and further processing within the ROS2 environment, the ros_gz_bridge package was installed and configured, allowing the point cloud to successfully be bridged and published as a ROS2 topic.

Another challenge encountered was the integration of the SpectacularAI SDK into the simulation environment. Since the SDK expects a physical OAK-D camera to be connected to self-instantiate a camera pipeline, it was not possible to redirect the input from a ROS2 topic, without obtaining a commercial license. For this reason and given the straightforward integration of RTAB-Map (Real-Time Appearance-Based Mapping) into ROS2, as well as the fact that both frameworks ultimately produce comparable point cloud maps, RTAB-Map was incorporated into the simulation environment. Only functionality equivalent to that

offered by the SpectacularAI SDK, namely the VIO and SLAM module, was utilised to simulate a close resemblance to the real-world setup.

RTAB-Map is a graph-based SLAM approach that performs incremental loop closure detection. To successfully integrate RTAB-Map into the simulation, several adjustments were required, including defining static transforms between the simulated camera sensor and the quadcopter base, modifying the camera settings, and tuning the simulated camera model to adjust the intrinsic camera parameters accordingly.

The final simulation setup, shown in **Fig. 3**, was then ready to be used for the development and testing of an autonomous navigation suite, which would later be integrated into a complete autonomous exploration stack. Future work will focus on integrating this autonomous exploration stack onto the actual quadcopter.

3.4 Autonomous navigation and exploration stack development

Due to the limited availability of 3D navigation and exploration frameworks for ROS2, it was decided to develop a custom framework, providing a stepping stone for transitioning 3D exploration algorithms from ROS1 to ROS2. An overview of the proposed 3D navigation and exploration system is presented in **Fig. 4.** This figure shows that the dense point cloud from the SLAM module is first converted into an OctoMap, which is a probabilistic volumetric occupancy grid. This conversion simplifies the environment into a coarse three-dimensional grid, where each voxel is classified as free, occupied, or unknown, and stored in a memory-efficient octree structure. This organisation significantly reduces the computational burden on downstream algorithms.

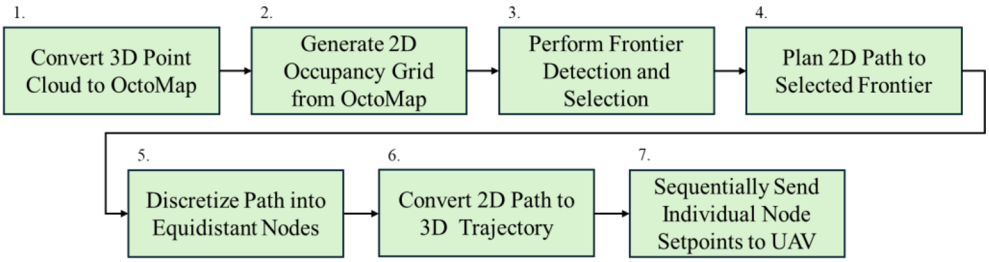

Fig. 4. Flowchart of navigation and exploration algorithm.

Since direct 3D frontier exploration proved unsuccessful, as the quadcopter sometimes attempted to navigate toward frontiers that could cause collisions, such as unobservable regions above the drone, an alternative approach was adopted. In this implementation, the 3D OctoMap was first flattened into a 2D occupancy grid. Frontier detection was then performed on this simplified 2D map, and once a frontier was identified, a 2D path was planned toward it. This 2D path was subsequently converted back into a 3D trajectory by referencing the states of OctoMap cells at varying altitudes along the path. This 3D-to-2D-to-3D mapping strategy not only simplified frontier detection but also significantly reduced computational requirements, making it well-suited for the proposed quadcopter platform, which only had limited onboard processing power.

Since RTAB-Map's built in 3D map to 2D occupancy grid module vertically projected all points onto a plane, it caused overhead structures such as the roof to be misclassified as obstacles, making the occupancy grid seem untraversable. It was therefore not appropriate for roofed environments such as underground mines and the map conversion package from [20] was used instead. This package was particularly well-suited, as it not only flattened the 3D map but also retained valuable information about the original OctoMap structure. In

combination with a path conversion module from the same package, this information could then be used to convert the generated 2D paths back into three-dimensional trajectories.

After obtaining a 2D occupancy grid, the next step involved deploying a path planning algorithm. For this purpose, the RRT* algorithm was chosen due to its low memory and computational requirements during sparse exploration and its capability of quickly finding feasible paths, even though they might initially be suboptimal [22]. After initial testing, the following parameter configuration was found to provide an effective performance in the simulated underground environment, though differently sized or structured environments may require further tuning: a step size of 0.5 m, a goal sampling bias of 0.1, a rewiring search radius of 1 m, a maximum of 1,500 iterations before termination, an obstacle buffer of 1 grid cell, and a discretised path resolution of 0.3 m.

To enhance the robustness, several additional features were incorporated alongside the core RRT algorithm. These included shifting the starting position to the next available unoccupied cell in cases where the quadcopter was positioned directly above unexplored areas, such as immediately after take-off, preventing invalid or unsafe planning attempts. Another improvement involved planning a path once and committing to it, rather than continuously replanning, which helped reduce computational overhead. To address situations where paths were generated dangerously close to obstacles, a buffer zone was introduced, ensuring that the quadcopter maintained a safe, predefined distance from surrounding structures.

A fail-safe mechanism was also implemented to handle cases where no odometry message was received, which is essential for defining the root of the planning tree. Instead of relying on outdated odometry data, the system was defined to halt all planning processes until new, valid odometry information becomes available. Finally, the generated RRT* path was discretised into a series of equidistant nodes, which would later simplify the path following process

During initial testing, goal poses were set manually using RViz2's "2D Goal Pose" tool, which publishes a pose to a ROS2 topic. This goal pose was then sent to the RRT* planner, where a path was generated and then converted into a 3D trajectory using the path conversion module mentioned earlier. This manual goal-setting process was later replaced by the autonomous exploration submodule, which was responsible for detecting frontiers and automatically assigning a goal pose to the selected frontier.

A custom frontier detection module was developed for this purpose. It operated on the 2D occupancy grid, which was previously derived from the 3D OctoMap, and identified frontiers, defined as the boundary between free space and unexplored regions. These frontiers were marked on the 2D map, and a random frontier was selected for exploration. The frontier selection logic is modular and customisable, allowing for different decision-making strategies to be implemented. These can include frontier selection based on proximity, path length, or expected information gain from visiting the frontier. The module also included logic to ensure that a new frontier is selected once the previous frontier had either been observed by the onboard camera or physically reached by the quadcopter. This was determined by comparing the frontier's position with the current pose of the quadcopter.

After the RRT* generated 2D path had been converted to a 3D trajectory, the individual node positions were sequentially sent to the quadcopter for path following. For the initial implementation, it was decided that a dedicated path-following algorithm was not required. Instead, since the nodes were already spaced equidistantly, the quadcopter was commanded to follow the path by publishing position setpoints at fixed time intervals. A custom ROS2 node was developed to manage this process. This piece of software kept the quadcopter in offboard mode and enabled position control, allowing the node to transmit position setpoints directly to the flight controller. The controller's internal cascaded control architecture was effectively used to execute the path-following task [30].

4 Results

This section presents the results obtained from both the real-world quadcopter platform and the developed autonomous navigation and exploration suite, which was extensively tested in the simulation environment.

4.1 Real world results

After applying the framerate reduction, voxelization and outlier removal changes to the SpectacularAI ROS2 wrapper, a series of tests were conducted to determine the optimal parameters for voxel down sampling and radius-based outlier filtering. When increasing the voxel size, more individual points to be merged into a single voxel, reducing the visual resolution of the point cloud while significantly freeing up computational power, as fewer points need to be processed and published. Therefore, a good balance must be struck.

Similarly, during radius outlier filtering, both the radial distance and the required number of inlier points must be carefully adjusted to ensure that excessive noise was removed without filtering out meaningful visual information. After extensive testing, the optimal parameter values were determined to be a voxel size of V = 0.03, a minimum inlier count of n = 10, and a radius of r = 0.15m. **Fig. 5** illustrates the impact of these filtering techniques, where the raw point cloud is shown in **Fig. 5A**, and the corresponding down sampled and filtered result is shown in **Fig. 5B**. Additionally, **Fig. 5C** provides an image of the scene that was used during the comparison. The Figure proved that during the filtering process critical information was successfully maintained, while most noisy outliers were removed from the point cloud.

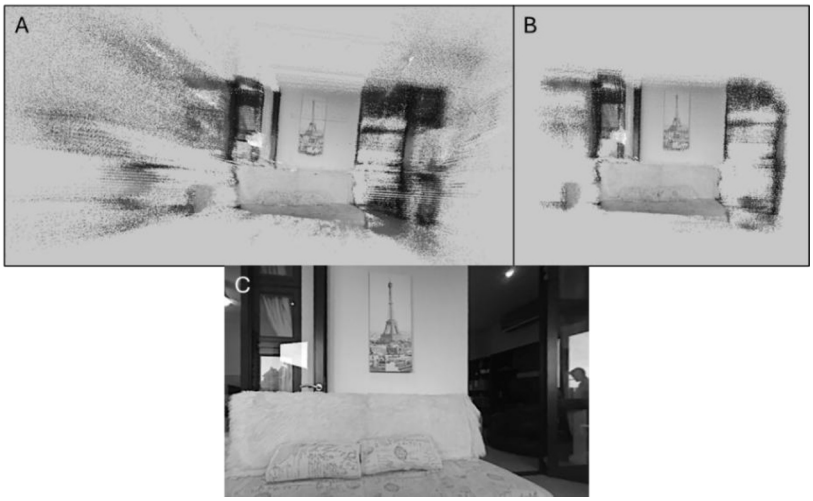

Fig. 5. A) An example of a point cloud before filtering, **B)** after filtering and **C)** a picture of the actual scene.

To demonstrate that the modified SpectacularAI module operates reliably on the developed quadcopter platform, a test was conducted in an indoor, GNSS-denied environment. During this test, the UAV was teleoperated through an apartment. The resulting map, shown in **Fig. 6**, demonstrated the performance of the SLAM module. Boundaries between individual rooms are visible with no obvious stretching or warping seen, which indicates minimal drift and good point cloud alignment during mapping. The complete coverage of the mapped apartment as well as the high-density nature of the point cloud furthermore suggested that the algorithm is well-suited for autonomous navigation tasks.

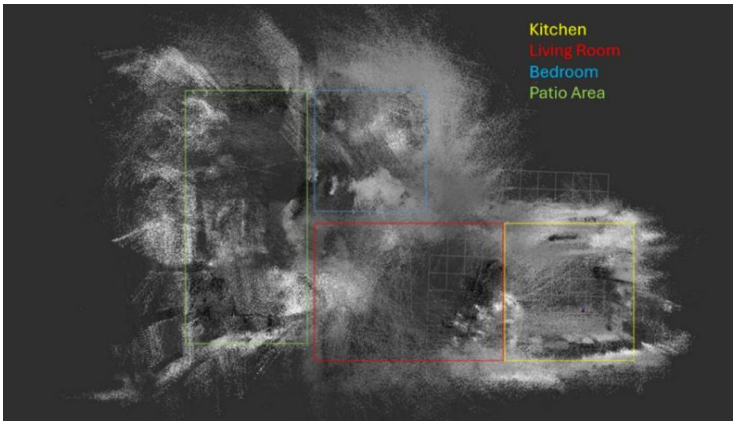

Fig. 6. Top view of a Map generated using the SpectacularAI SLAM module, running onboard the developed quadcopter.

4.2 Simulated results

While future work focuses on the implementation of the developed autonomous exploration framework on the physical quadcopter, extensive testing has been conducted in simulation, to verify the functionality of the software. Since the simulation makes use of RTAB-Map, as a replacement for the SpectacularAI suite used on the actual quadcopter, this SLAM modules functionality needed to be tested. To keep it as realistic as possible, the RTAB-Map parameters were changed to make is use as little computational power as possible while keeping it reliable and accurate. **Table 1** gives an overview of the most important parameters set for RTAB-Map to emulate the SpectacularAI SLAM module.

Table 1. RTAB-Map parameters set to emulate SpectacularAI SLAM Module.

Parameter	**Value**	**Description**
Rtabmap/DetectionRate	2	Mapping/Loop Closure Frequency
Reg, Strategy	2 (VisICP)	Method used for point registration
Reg/Force3DoF	False (Use 6DoF)	Use 6 DoF for point registration
Odom/Strategy	5 (ORB SLAM2)	Select Core VO algorithm
OdomORBSLAM/ Fps	10	Camera FPS used for VO

Using these parameters, a simulation test was conducted, and the resulting SLAM map along with the VIO output is shown in **Fig. 7**. As seen in the figure, a nearly complete map was generated, with only a few minor gaps and holes. A high level of detail is evident, which can particularly be seen in the tunnel texture. Additionally, the alignment of individual point clouds appears accurate with only minor misalignments, as indicated by the strong structural similarity between the generated map and the actual simulation environment. However, a minor mis alignment can be seen in the bottom left corner of the generated map. The visualised odometry also closely follows the UAV's true flight path. This mapping test, which produced a realistic reconstruction of the simulated environment, confirmed that RTAB-Map is a suitable foundation for the development of the autonomous exploration suite.

Fig. 7. The SLAM map generated by the optimised RTAB-Map algorithm, including odometry from its internal VIO module (visualised as the blue line), is shown on the left, while the ground truth simulation environment is depicted on the right.

After verifying the acceptable performance of the SLAM module, the autonomous exploration suite was tested. During the first tests, each building block of the proposer framework, depicted in **Fig. 4** was individually tested in the simulation environment shown in **Fig.3**. **Fig. 8** Shows the results obtained from this initial test, verifying the functionality of each individually developed and implemented submodule. It is worth noting that the submodules were tested individually, and the images generated therefore are not dependent on each other.

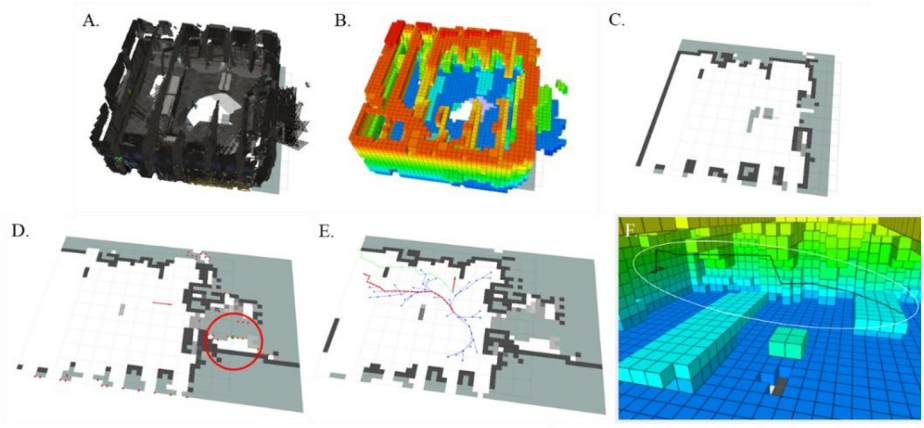

Fig. 8. Testing of autonomous exploration framework submodules. **A)** A point Cloud map is converted into **B)** an OctoMap which is then Flattened to provide **C)** a 2D Occupancy Grid. **D)** Frontier points are identified (red dots), a frontier is selected (encircled green dot) and **E)** a 2D path is planned towards it before being converted back into **F)** a 3D trajectory.

After integrating all individual submodules, final tests were conducted to verify the functionality of the complete autonomous exploration suite. **Fig. 9A** shows the map generated during an autonomous exploration mission, in which the drone was launched from the staging area of the simulation environment shown in **Fig. 7**. The exploration run lasted approximately 3 minutes and 30 seconds. By comparing the generated map in **Fig. 9A** with the ground truth simulation environment in **Fig. 7**, it is evident that the entire room was successfully mapped, demonstrating the strong performance of the exploration suite. Additionally, the blue line represents the trajectory setpoints generated by the suite, which trace the locations where

frontier points were detected during the operation, while the green line indicates the estimated position of the quadcopter as calculated by its onboard EKF.

The results of a second test, in which the quadcopter was launched inside the simulated tunnel and operated for approximately 4 minutes and 20 seconds, are shown in **Fig. 9B**, with a clearer visualisation of the quadcopter's setpoint trajectory, pose estimate, and 2D occupancy grid provided in **Fig. 9C**. Once again, these results demonstrate the vehicle's capability to autonomously map GNSS-denied environments and highlight the functionality and effectiveness of the developed autonomous exploration module.

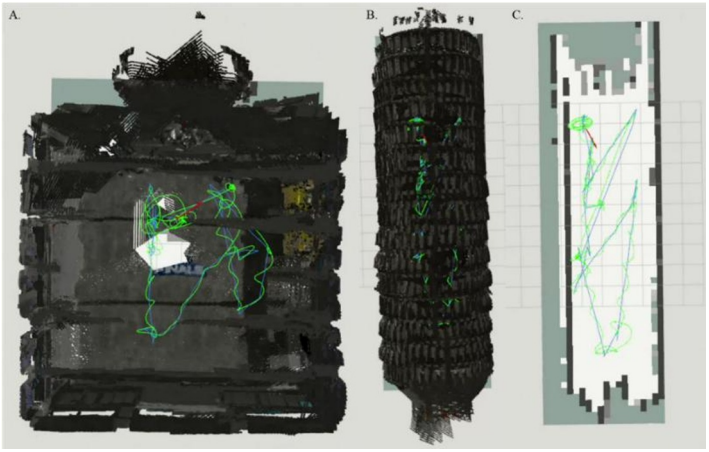

Fig. 9. A) An image depicting the resultant map, setpoint path (blue line) and vehicle position (green line) of an autonomous exploration mission started in the Simulated staging area and **B)** started in the simulated tunnel area, where **C)** provides a clearer visualisation of the 2D occupancy grid, the setpoint path and the vehicles position during the tunnel exploration.

While the proposed autonomous exploration suite has proven capable of fully autonomous exploration and navigation, it serves as a foundational framework for GNSS-denied exploration, with future work aimed at refining the individual subsystems to improve exploration efficiency and reduce overall mission time. Due to the framework's modular design and the use of ROS2, integrating improved modules will be straightforward. Additionally, future work will focus on testing the autonomous exploration suite on a real quadcopter to further validate its capabilities and functionality in real-world scenarios.

5 Conclusion

A quadcopter capable of autonomously exploring and mapping GNSS-denied environments, such as underground mines, was successfully developed using cost-effective, energy-efficient components. The mapping algorithm was deployed on the physical platform, and preliminary testing validated the system's capability to perform accurate onboard three-dimensional mapping of cluttered environments. A ROS2-based autonomous exploration suite was developed with onboard computational constraints in mind and tested in a subterranean simulation environment, demonstrating successful autonomous exploration and mapping capabilities in underground mine-like conditions.

The continuation of this work involves deploying the developed suite onto the real quadcopter, conducting physical test flights and adjustments to the algorithm, tailored for the actual hardware platform. As the objective of this project was the end-to-end development of a UAV system for autonomous underground exploration, rather than the specialised optimisation of individual subsystems, future work may focus on improving these subsystems. Furthermore, features such as precision landing and data sharing capabilities

could be integrated for interaction with other platforms, enabling coordinated multi-agent exploration. The proposed design and methodology establish a flexible foundational robot for advancing intelligent, autonomous UAV systems in subterranean mine surveying, promoting safer, more efficient data acquisition in challenging environments.

References

1. Q. Han, Y. Bao, S. Pasricha, Improving Safety in Cyber Enabled Underground Mines, in Proceedings of the 8th International Conference on Networking, Systems and Security, New York, USA, 21-23 December, 21 December (2021)
2. T. Neumann, A. Ferrein, S. Kallweit, I. Scholl, Towards a Mobile Mapping Robot for Underground Mines, in Proceedings of the 2014 PRASA, RobMech and AfLaT International Joint Symposium, Cape Town, South Africa, 27-28 November, 27 November (2014)
3. C. Tatsch, J. A. Bredu, D. Covell, I. B. Tulu, Y. Gu, Rhino: An Autonomous Robot for Mapping Underground Mine Environments, in Proceedings of the 26th IEEE International Conference on Advanced Intelligent Mechatronics, Seattle, USA, 28-30 June, 28 June (2023)
4. M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, K. Alexis, CERBERUS in the DARPA Subterranean Challenge, Sci. Robot. **7**, 66 (2022). https://doi.org/10.1126/scirobotics.abp9742
5. P. De Petris, H. Nguyen, M. Dharmadhikari, M. Kulkarni, N. Khedekar, F. Mascarich, K. Alexis, Rmf-owl: A collision-tolerant flying robot for autonomous subterranean exploration, in Proceedings of the International Conference on Unmanned Aircraft Systems, Dubrovnik, Croatia, 21-24 June, 21 June (2022)
6. A. Pålsson, M. Smedberg, Investigating Simultaneous Localization and Mapping for AGV systems, Master Thesis, Chalmers University of Technology and University of Gothenburg, Department of Computer Science and Engineering, (2017)
7. M. Zimmermann, Trajectory (re)planning of a quadcopter in indoor environments with unknown obstacles, Diploma Thesis, TU Wien, Faculty of Electrical Engineering and Information Technology, (2022)
8. F. Li, S. Zlatanova, M. Koopman, X. Bai, A. Diakité, Universal path planning for an indoor drone, Autom. Constr. **95**, 275 (2018). https://doi.org/10.1016/j.autcon.2018.07.025
9. A. Farooq, C. Laoudias, P. S. Kolios, T. Theocharides, Quantitative and Qualitative Assessment of Indoor Exploration Algorithms for Autonomous UAVs, in Proceedings of the International Conference on Unmanned Aircraft Systems, Dubrovnik, Croatia, 21-24 June, 21 June (2022)
10. B. Zhou, Y. Zhang, X. Chen, S. Shen, FUEL: Fast UAV Exploration Using Incremental Frontier Structure and Hierarchical Planning, IEEE robot. autom. lett. **6**, 779 (2021). https://doi.org/10.1109/LRA.2021.3051563
11. B. Lindqvist, A. Patel, K. Löfgren, G. Nikolakopoulos, A Tree-Based Next-Best-Trajectory Method for 3-D UAV Exploration, IEEE Trans. Robot. **40**, 3496 (2024). https://doi.org/10.1109/TRO.2024.3422052
12. E. M. Lee, D. Choi, H. Myung, Peacock Exploration: A Lightweight Exploration for UAV Using Control-Efficient Trajectory, in Proceedings of the International

Conference on Robot Intelligence Technology and Applications, Singapore, 11-13 December, 11 December (2020)

13. A. S. Al-Batati, A. Koubaa, M. Abdelkader, ROS 2 Key Challenges and Advances: A Survey of ROS 2 Research, Libraries, and Applications, preprints202410.1204 (2024)
14. M. Casuccio, ROS2-Based AMR System for Mapping and Navigation in Unknown Indoor Environments, Master Thesis, Polytechnic University of Turin, Italy, (2024)
15. D. Portugal, R. P. Rocha, J. P. Castilho, Inquiring the robot operating system community on the state of adoption of the ROS 2 robotics middleware, Int. J. Intell. Robot. Appl. **9**, 454 (2024). https://doi.org/10.1007/s41315-024-00393-4
16. Flyability, Elios 3, flyability.com
17. S. Jarahizadeh, B. Salehi, A Comparative Analysis of UAV Photogrammetric Software Performance for Forest 3D Modeling: A Case Study Using AgiSoft Photoscan, PIX4DMapper, and DJI Terra, Sensors **24**, 286 (2024). https://doi.org/10.3390/s24010286
18. B. Zhou, H. Xu, S. Shen, RACER: Rapid Collaborative Exploration With a Decentralized Multi-UAV System, IEEE Trans. Robot. **39**, 1816 (2023). https://doi.org/10.1109/TRO.2023.3236945
19. A Hornung, K. M.Wurm, M. Bennewitz, C. Stachniss, W. Burgard, OctoMap: an efficient probabilistic 3D mapping framework based on octrees, Auton. Robot. **34**, 189 (2013). https://doi.org/10.1007/s10514-012-9321-0
20. S. Fredriksson, A. Saradagi, G. Nikolakopoulos, Voxel Map to Occupancy Map Conversion Using Free Space Projection for Efficient Map Representation for Aerial and Ground Robots, IEEE robot. autom. lett. **9**, 11625 (2024). https://doi.org/10.1109/LRA.2024.3495575
21. S. M. LaValle, J. J. Kuffner, Randomized Kinodynamic Planning, in Proceedings of the IEEE International Conference on Robotics and Automation, Detroit, MI, USA, 10-15 May (1999), 06 August (2002). https://doi.org/10.1109/ROBOT.1999.770022
22. S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, Int. J. Robot. Res. **30**, 846 (2011). https://doi.org/10.1177/0278364911406761
23. S. Yamauchi, A Frontier-Based Approach for Autonomous Exploration, in Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation, Monterey, CA, USA, 10-11 July (1997), 06 August (2002)
24. iFlight, R5 2207 2050KV FPV Motor Pro, shop.iflight.com (accessed March 21, 2024)
25. O. Liang, Propellers: The Complete Guide, oscarliang.com (accessed March 22, 2024)
26. J. Kalyanasundaram, Y. Simmhan, ARM Wrestling with Big Data: A Study of Commodity ARM64 Server for Big Data Workloads, in Proceedings of the 24th IEEE International Conference on High Performance Computing, Jaipur, India, 18-21 December (2017), 08 February (2018)
27. Khadas Documentation Team, “VIM4 Getting Started”, docs.khadas.com (accessed May 05, 2024)
28. O. Seiskari, P. Rantalankila, J. Kannala, J. Ylilammi, E. Rahtu, A. Solin, HybVIO: Pushing the limits of real-time visual-inertial odometry, in Proceedings of the IEEE Winter Conference on Applications of Computer Vision, New Orleans, USA, 18-24 June (2022), 18 June (2022)
29. Open Robotics Foundation, “SubT Tech Repo,” app.gazebosim.org
30. PX4 Developer Team, PX4 Flight Stack – Multicopter Control Architecture, docs.px4.io (accessed September 15, 2024)