

Universiteit Leiden

Autonomous Indoor Exploration and Mapping for a Nano-UAV Using Sparse Time-of-Flight Sensing

Tijs Konijn (s3197794)

Supervisors:
Mike Preuss & Aske Plaat

Master Thesis– Computer Science - Artificial Intelligence

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

June 5, 2026

Abstract

Indoor mapping and object search are important capabilities for autonomous drones operating in environments where GPS is unavailable. While modern Simultaneous Localization and Mapping (SLAM) systems typically rely on cameras or LiDAR sensors, such sensing solutions are often too heavy and power-intensive for nano-UAVs. This research investigates whether a Crazyflie 2.1 micro-drone equipped only with a Flowdeck V2 and a Multi-Ranger deck can autonomously map an indoor environment and detect newly introduced objects using lightweight onboard sensing.

A complete autonomous mapping framework was developed in ROS2 and evaluated in both simulation and real-world experiments. The system combines optical flow based odometry, sparse time of flight range measurements, a custom occupancy grid mapping algorithm, and four exploration strategies: a baseline wall follower, an improved wall follower, Frontier Exploration, and Next Best View (NBV) planning. Mapping performance was evaluated using F1 score, Structural Similarity Index (SSIM), Hausdorff distance, coverage, and path length. Object detection was implemented through occupancy grid differencing between two mapping phases.

Simulation results showed that exploration based methods significantly outperformed reactive wall-following approaches. Frontier Exploration achieved the highest overall mapping performance, reaching an F1 score of 0.765, an SSIM score of 0.546, and the lowest average Hausdorff distance of 3.92 m in the large benchmark environment. NBV produced comparable results but exhibited greater variability between runs. Object detection experiments demonstrated that newly introduced obstacles could be identified successfully through map differencing.

Real world experiments confirmed that the Crazyflie is capable of producing usable occupancy maps using only onboard sensors. However, sensor limitations, environmental conditions, and accumulated drift substantially reduced mapping accuracy compared to simulation. The results show that autonomous indoor mapping and basic object detection are feasible on a nano-UAV with minimal sensing, but robust deployment in real environments remains constrained by sparse measurements and localization errors.

Contents

1 Introduction **1**

2 Background and Related Work **2**
2.1 Simultaneous Localization and Mapping (SLAM) . . . 2
2.2 SLAM with Sparse Sensors (ToF + Optical Flow) . . . 4
2.3 Micro-UAV Constraints . . . 5
2.4 Crazyflie Platform and Research Applications . . . 6
2.5 Simulation-to-Real Considerations . . . 7
2.6 Exploration and Path Planning . . . 8

3 Methods **9**
3.1 Crazyflie2.1+ . . . 9
3.1.1 Flowdeck V2 . . . 11
3.1.2 Multiranger . . . 11
3.2 ROS2 . . . 12
3.3 Webots . . . 13
3.4 Custom Occupancy Grid Mapping . . . 14
3.5 Exploration and Path Planning . . . 15
3.5.1 Reactive Wall Following . . . 16
3.5.2 Frontier-Based Exploration . . . 16
3.5.3 A* Path Planning . . . 17
3.5.4 Pure Pursuit Path Following . . . 17
3.6 Simulation-to-Real Adaptation . . . 17

4 Experiments and Results **19**
4.1 Simulation Performance . . . 19
4.1.1 Finetuning . . . 22
4.1.2 Pathing behavior . . . 25
4.1.3 Object Detection . . . 30
4.2 Simulation-to-Real . . . 31
4.2.1 Sensor limitations . . . 31
4.2.2 SLAM in Reality . . . 32
4.2.3 Object Detection in Reality . . . 34

5 Discussion **34**

6 Conclusion **36**

7 Future Work **37**

References **43**

1 Introduction

Unmanned aerial vehicles (UAVs) have become increasingly important for inspection, monitoring, mapping, and search and rescue tasks in environments that are difficult or unsafe for humans to access. Recent developments in embedded systems and lightweight sensing technologies have enabled the emergence of nano-UAVs, aircraft weighing only a few tens of grams that are capable of autonomous flight in confined indoor environments [36]. Compared to larger aerial platforms, nano-drones can operate safely near people, navigate through narrow openings, and access spaces that would otherwise be inaccessible. These characteristics make them attractive for applications such as indoor inspection, inventory management, environmental monitoring, and rapid assessment of hazardous environments.

At the same time, the small size of nano-UAVs introduces significant technical challenges. Their limited payload capacity restricts the sensors and computational hardware that can be carried onboard. While larger UAVs frequently employ cameras, multi-beam LiDAR sensors, and powerful onboard computers for localization and mapping, such hardware is generally too heavy and power intensive for nano-scale platforms [36]. As a result, micro-UAVs often rely on lightweight sensors that provide only sparse information about the environment. This creates a challenging scenario for autonomous navigation and mapping, particularly in indoor environments where Global Positioning System (GPS) signals are unavailable.

Simultaneous Localization and Mapping (SLAM) is the standard approach for enabling autonomous operation in such GPS denied environments. SLAM allows a robot to estimate its own position while simultaneously constructing a map of its surroundings. Traditional SLAM systems commonly rely on dense visual observations or laser range measurements, allowing accurate reconstruction of indoor environments [8]. However, these approaches assume sensing capabilities that are typically unavailable on nano-UAVs. Therefore, applying SLAM to highly constrained aerial platforms remains an active research challenge. Recent work has demonstrated that meaningful localization and mapping can still be achieved using lightweight sensors such as optical flow cameras and low cost time of flight (ToF) sensors, provided that the mapping algorithms are carefully designed to compensate for the limited information available [36, 37].

This research focuses on the Crazyflie 2.1+, a widely used open-source nano-UAV platform developed for research and education. The Crazyflie has become a popular experimental platform in robotics because of its modular design, extensive software ecosystem, and support for lightweight sensing extensions. In particular, the Flowdeck V2 combines optical flow measurements with a downward-facing ToF sensor for velocity and altitude estimation, while the Multi-Ranger deck provides distance measurements in five directions using miniature ToF sensors [2, 3]. Although these sensors are lightweight enough to be carried by the platform, they provide only sparse measurements of the environment. Unlike LiDAR based systems that observe hundreds or thousands of points simultaneously, the Multi-Ranger deck provides only a small number of one dimensional range measurements at each instant. This significantly complicates the mapping process.

Previous work has demonstrated that the Crazyflie can perform basic indoor mapping using these sensors. For example, McGuire et al. showed that a combination of wall-following behaviour, optical flow odometry, and Multi-Ranger measurements can generate occupancy grid maps in both simulation and real world environments [31]. More broadly, the Crazyflie platform has been extensively used as a testbed for research in swarm robotics, autonomous exploration, collaborative mapping, and resource-constrained SLAM [36, 37]. Despite these advances, important questions

remain regarding the quality of maps that can be produced using such sparse sensor data and the extent to which these maps can support higher level tasks such as autonomous object search.

The objective of this research is therefore to investigate the feasibility of indoor mapping and object detection using a Crazyflie 2.1+ equipped only with a Flowdeck V2 and a Multi-Ranger deck. Specifically, this work examines whether reliable occupancy grid maps can be generated from sparse ToF and optical-flow measurements, how different exploration strategies influence mapping performance, and whether the resulting maps can be used to detect previously unseen objects through repeated exploration of an environment. Furthermore, the study evaluates how performance differs between simulation and real world deployments, providing insight into the practical limitations of deploying resource constrained aerial robots outside controlled simulation environments.

The remainder of this research is structured as follows. Chapter 2 reviews related work on SLAM, micro-UAV navigation, exploration strategies, and simulation-to-real transfer. Chapter 3 describes the Crazyflie platform, the proposed mapping framework, and the exploration algorithms investigated in this work. Chapter 4 presents the experimental setup and evaluates the system in both simulation and real world environments. Then, Chapter 5 discusses the results, limitations of the current approach, Finally Chapter 6 concludes the research and in Chapter 7 possible new directions for future research are discussed.

2 Background and Related Work

This section reviews the state of the art in autonomous indoor navigation, mapping, and exploration for micro-UAV platforms operating in GPS-denied environments. Particular focus is placed on nano drones such as the Crazyflie 2.1+, which rely on lightweight onboard sensing and limited computational resources. First, the fundamentals of Simultaneous Localization and Mapping (SLAM) are discussed, including graph based, filter based, visual, and LiDAR-based approaches. Subsequently, the challenges of performing SLAM with sparse sensors such as Time-of-Flight range sensors and optical flow cameras are examined. The section further discusses the physical and computational limitations of micro-UAV systems, including payload, energy, sensing, and onboard processing constraints.

In addition, previous research involving the Crazyflie platform is reviewed to position this work within the broader context of nano-UAV navigation and mapping research. The use of simulation environments for robotics development is also discussed, including the simulation-to-reality gap and methods such as domain randomization and system identification that aim to improve transferability between simulated and physical environments [58, 39]. Finally, existing approaches for autonomous exploration and path planning are summarized, including frontier based exploration, graph search planning algorithms such as A*, and trajectory-following methods including Pure Pursuit.

2.1 Simultaneous Localization and Mapping (SLAM)

Autonomous indoor navigation requires a robot to estimate its own position while simultaneously constructing a map of the surrounding environment. Since Global Positioning System (GPS) signals are unreliable or unavailable indoors, this problem is commonly addressed using Simultaneous Localization and Mapping (SLAM) techniques [15, 6]. SLAM has become one of the fundamental

problems in mobile robotics because localization and mapping are strongly interdependent: accurate localization requires a map, while accurate mapping requires knowledge of the robot pose.

In general, SLAM systems combine sensor measurements with probabilistic state estimation techniques to incrementally estimate both the robot trajectory and the environment representation. Common sensor modalities include LiDAR, RGB cameras, stereo vision systems, inertial measurement units (IMUs), and Time-of-Flight (ToF) sensors [8]. Depending on the sensing modality and optimization strategy, SLAM systems can be categorized into filter based SLAM, graph based SLAM, and visual SLAM approaches.

Filter based SLAM methods estimate the robot state recursively using probabilistic filters such as the Extended Kalman Filter (EKF) or particle filters [48]. These approaches continuously update the estimated pose and map whenever new sensor measurements become available. EKF-SLAM was one of the earliest successful probabilistic SLAM methods and remains computationally efficient for smaller environments. However, filter based approaches tend to accumulate drift over time because previous estimates are not globally optimized.

Graph based SLAM instead formulates the SLAM problem as a graph optimization task [17]. Robot poses are represented as graph nodes while spatial constraints between poses form graph edges. When the robot revisits a previously explored area, loop closures can be detected and incorporated into the graph, enabling global trajectory correction and reducing accumulated drift. Graph based approaches generally produce more accurate maps than filter based methods but are computationally more demanding.

Visual SLAM (VSLAM) approaches estimate motion and reconstruct environments using camera imagery [33]. Systems such as ORB-SLAM use visual feature extraction and feature matching to estimate camera motion and detect loop closures. Visual SLAM can generate highly detailed maps with relatively inexpensive sensors, but performance strongly depends on lighting conditions, texture availability, and camera quality. Feature poor indoor environments such as white corridors or low light rooms can significantly reduce localization accuracy.

LiDAR-based SLAM methods instead rely on laser range measurements and are generally more robust in poorly illuminated environments [8]. Two-dimensional LiDAR systems are commonly used for occupancy grid mapping, while three-dimensional LiDAR systems provide dense geometric reconstructions of complex environments. LiDAR based systems often achieve higher mapping accuracy than visual approaches, especially indoors, but the sensors are relatively expensive, heavy, and energy intensive. Therefore, they are often unsuitable for nano-UAV platforms with strict payload limitations.

For micro-UAVs, the SLAM problem becomes significantly more challenging due to computational, sensing, and power constraints [24]. Lightweight drones cannot easily carry high resolution LiDAR systems or powerful onboard computers. As a result, many nano-UAV platforms rely on sparse sensing and lightweight state estimation methods rather than dense mapping approaches.

Recent work has demonstrated that lightweight SLAM on nano scale robotic platforms is nevertheless feasible. NanoSLAM proposed a fully onboard SLAM pipeline for tiny robots by combining lightweight visual inertial odometry with optimized mapping algorithms [36]. Similarly, Karam et al. [24] demonstrated graph based indoor mapping using a microdrone equipped only with one-dimensional laser rangefinders and optical flow sensing. Their results showed that sparse sensing can still produce usable indoor maps when combined with appropriate SLAM optimization techniques. Despite these advances, robust SLAM for nano-UAVs with extremely sparse sensing remains an open research challenge. Limited field of view sensors, odometry drift, low computational

resources, and restricted payload capacity continue to constrain mapping accuracy and navigation robustness.

2.2 SLAM with Sparse Sensors (ToF + Optical Flow)

Compared to conventional indoor SLAM systems that rely on dense LiDAR scans or camera imagery, the sensor configuration used in this research provides only sparse environmental observations. The Crazyflie platform estimates motion using optical flow and altitude measurements from the Flowdeck V2, while environmental perception is obtained from the five single beam Time of Flight sensors of the Multi-Ranger deck. As a result, the drone observes only a small number of distance measurements at each time step rather than a dense point cloud or laser scan.

This limited sensing capability introduces several challenges for simultaneous localization and mapping. Most SLAM algorithms are designed for sensors that provide rich geometric information and sufficient overlap between observations. With only a handful of range measurements available at any moment, environmental observability is reduced, localization uncertainty increases, and loop-closure detection becomes significantly more difficult. Sparse measurements also make occupancy grid construction more sensitive to individual sensor errors or missed readings, since each measurement contributes a relatively large portion of the available environmental information. Therefore, conventional LiDAR based SLAM techniques often perform poorly when directly applied to sparse range data, requiring alternative mapping and exploration strategies that explicitly account for the limited sensing coverage available on nano-UAV platforms [61].

Despite these limitations, previous research has demonstrated that sparse-sensor SLAM can still produce useful indoor maps. Karam et al. [24] developed a graph based SLAM system for a microdrone equipped with only six one-dimensional laser rangefinders and optical sensing. Their system achieved mapping accuracy comparable to larger LiDAR-equipped drones in small indoor environments.

Similarly, Bitcraze demonstrated occupancy grid mapping using the Crazyflie Multi-Ranger deck together with ROS2 based SLAM toolboxes [31]. In these demonstrations, sequential range measurements are integrated over time to incrementally construct a two-dimensional occupancy grid map. The occupancy grid representation models the environment probabilistically by discretizing space into occupied, free, and unknown cells [48].

However, sparse sensing remains fundamentally limited compared to dense perception systems. Since only a small number of range measurements are available at each timestep, environmental structure can easily become ambiguous. Sparse observations also reduce loop closure reliability because revisited areas may not contain sufficiently unique geometric signatures for robust place recognition.

Additionally, flow based odometry errors accumulate over time and directly influence mapping quality. Small velocity estimation errors gradually distort the occupancy map, particularly during longer exploration trajectories. Without reliable loop closure or global optimization, these errors can lead to inconsistent maps and navigation failures.

Recent research has therefore focused on improving lightweight SLAM approaches specifically for resource constrained robotic platforms. Efficient sparse sensing graph SLAM methods [61] and ultra lightweight collaborative mapping systems for robot swarms [37] attempt to reduce computational requirements while preserving mapping accuracy. Autonomous exploration and object-search tasks using only sparse ToF sensing remain relatively underexplored in the literature. Most existing

approaches either rely on richer sensing modalities or assume access to external localization systems.

2.3 Micro-UAV Constraints

Micro-UAVs and nano-UAVs operate under significantly stricter physical and computational constraints than larger robotic platforms. These limitations strongly influence the design of sensing systems, navigation algorithms, onboard processing architectures, and autonomous behaviors [11, 24]. In contrast to larger drones that can carry high resolution LiDAR systems, stereo vision rigs, or embedded GPUs, micro-UAV platforms are constrained by limited payload capacity, battery endurance, onboard memory, and processing power.

One of the most fundamental constraints is the trade off between payload weight and flight endurance. Due to the square cube scaling law, reducing vehicle dimensions drastically reduces the available energy storage and payload capacity [56]. As a result, adding heavier sensors or more powerful processors directly impacts flight time and maneuverability. Even lightweight sensing systems can significantly alter flight dynamics and energy consumption. Consequently, micro-UAVs typically rely on lightweight and low power sensing modalities such as optical flow cameras, inertial measurement units (IMUs), and low-resolution Time of Flight (ToF) sensors rather than computationally intensive dense sensing systems [36].

Power consumption is another critical limitation. Autonomous flight requires energy not only for propulsion, but also for sensing, onboard computation, wireless communication, and stabilization [35]. Since the majority of available energy is consumed by the propulsion system, only a small fraction of the power budget remains available for autonomous navigation algorithms. This severely limits the use of computationally intensive techniques such as dense visual SLAM, deep neural networks, or high frequency three-dimensional mapping pipelines [56].

In addition to power limitations, onboard processing resources are highly constrained. Most nano-UAV platforms rely on microcontroller class processors with limited memory and computational throughput [50]. These processors are highly suitable for low latency stabilization and sensor fusion tasks, but struggle with complex optimization problems or large scale map representations. Therefore, many micro-UAV systems offload computationally demanding tasks such as SLAM optimization or path planning to external computers using wireless communication links [24]. However, offboard computation introduces additional challenges including communication latency, packet loss, and bandwidth limitations.

Sensing capabilities are similarly restricted. Due to payload and power limitations, many micro-UAV systems rely on sparse sensing configurations with narrow fields of view and limited sensing ranges [61]. Sparse sensors provide significantly less environmental information than dense LiDAR or RGB-D systems, reducing observability and increasing uncertainty during localization and mapping. Narrow field of view ToF sensors in particular suffer from what recent literature describes as *sensory myopia*, where only a very limited portion of the environment is observable at any given moment [56]. This complicates obstacle avoidance, loop closure detection, and reliable map generation.

Flight dynamics additionally become increasingly difficult to control at smaller scales. Nano-UAVs possess very low inertia and are therefore highly sensitive to aerodynamic disturbances, motor vibrations, air turbulence, and sensor noise [11]. Rapid altitude changes and vibrations can degrade inertial sensing and visual odometry accuracy, particularly during aggressive maneuvers or near obstacles. These effects further complicate state estimation and autonomous navigation.

Environmental factors also impose substantial challenges. Indoor navigation commonly occurs in GPS denied environments where localization must rely entirely on onboard sensing [24]. Variable lighting conditions, reflective surfaces, textureless walls, and transparent obstacles can degrade both optical flow estimation and range sensing accuracy. Such limitations are particularly problematic for lightweight drones that cannot compensate using richer sensing modalities.

Despite these constraints, recent research has demonstrated that autonomous navigation and mapping on highly resource constrained aerial robots is feasible. NanoSLAM demonstrated fully onboard SLAM for tiny robotic platforms by combining lightweight sensing with highly optimized mapping algorithms [36]. Similarly, recent collaborative mapping approaches have shown that even extremely constrained nano-UAV swarms can construct occupancy maps using sparse ToF sensing and lightweight graph based optimization [37]. Other work has further demonstrated low power localization using low resolution multizone ToF sensors and ultra low power parallel processors [34].

2.4 Crazyflie Platform and Research Applications

The Crazyflie platform has become one of the most widely adopted nano-UAV systems in aerial robotics research due to its open source ecosystem, low operational cost, lightweight design, and extensive software support [1, 13]. Unlike larger research UAVs, the Crazyflie enables rapid experimentation in indoor environments with relatively low safety risks, making it particularly attractive for research involving autonomous navigation, swarm coordination, and lightweight SLAM algorithms.

Over the last decade, the platform has been used extensively in swarm robotics research. One of the best known examples is the Crazyswarm framework, which demonstrated coordinated flight with large groups of nano quadrotors using centralized trajectory generation and motion capture based localization [40]. This work showed that relatively inexpensive nano-UAV platforms could reliably execute synchronized multi-agent trajectories, enabling research into formation control, trajectory optimization, and decentralized aerial coordination.

Subsequent research expanded these ideas toward increasingly autonomous swarm behaviors. Recent work such as AttentionSwarm investigated reinforcement learning based swarm coordination using Crazyflie drones in dynamic obstacle environments [46]. Their system combined attention mechanisms with Control Barrier Functions (CBFs) to enable safe multi-agent navigation and collision avoidance during dynamic drone racing and cooperative flight tasks. Such studies demonstrate that the Crazyflie platform is a suitable platform for advanced multi-agent robotics research.

The platform has additionally become popular in research on autonomous indoor navigation and obstacle avoidance. McGuire et al. [29] demonstrated reactive collision avoidance for nano-UAVs using only lightweight onboard sensing, including optical flow and sparse range measurements. Similar studies have used the Crazyflie for wall following, corridor navigation, reactive planning, and lightweight exploration strategies in GPS-denied indoor environments [24].

Another major research direction involving the Crazyflie concerns lightweight SLAM and mapping systems for resource constrained aerial robots. Since nano-UAVs cannot easily carry dense LiDAR systems or embedded GPUs, the platform is frequently used to investigate sparse sensing SLAM techniques [36]. Karam et al. [24] demonstrated graph based indoor mapping using a lightweight microdrone equipped only with one-dimensional laser rangefinders and optical sensing. More recent work such as NanoSLAM further demonstrated fully onboard SLAM on tiny robotic platforms using highly optimized lightweight algorithms [36]. Collaborative mapping using groups

of Crazyflie drones has also been explored in recent swarm SLAM research [27, 37].

The Crazyflie ecosystem has furthermore become widely used in reinforcement learning and sim-to-real robotics research. Due to the availability of simulation support through ROS, Webots, Gazebo, and PyBullet integrations, the platform enables rapid testing of learning based flight control and navigation algorithms before deployment on physical hardware [13]. Several studies have used the Crazyflie platform for trajectory learning, autonomous racing, adaptive flight control, and learning-based swarm coordination [46, 38].

2.5 Simulation-to-Real Considerations

Developing autonomous robotic systems directly on physical hardware is often expensive, time consuming, and potentially unsafe. Consequently, robotics simulators are commonly used during early development stages to accelerate experimentation, validate algorithms, and reduce the risk of hardware damage [58, 14]. Simulation environments allow researchers to test navigation, control, and mapping algorithms under reproducible conditions before deployment on real robotic platforms.

Despite these advantages, transferring algorithms from simulation to real world systems remains a major challenge in robotics research. This problem is commonly referred to as the *simulation-to-reality gap* or *sim-to-real gap* [58, 22]. The gap emerges because simulations can only approximate real world physics and sensor behavior. Factors such as imperfect aerodynamic modelling, motor nonlinearities, sensor inaccuracies, wireless communication delays, battery degradation, and environmental disturbances are difficult to replicate accurately in simulation environments.

For micro-UAV systems, these discrepancies become even more significant due to the highly sensitive flight dynamics of lightweight drones [11]. Small aerial robots possess low inertia and are therefore strongly affected by airflow disturbances, ground effect interactions, sensor noise, and rapid attitude changes. In real world indoor environments, additional factors such as changing lighting conditions, reflective surfaces, and textureless floors can further degrade optical flow estimation and Time of Flight sensor accuracy [36]. Such effects are often simplified or omitted entirely in simulation.

Previous research has shown that controllers and navigation algorithms performing well in simulation may behave differently once deployed on physical robotic platforms due to modelling inaccuracies and unmodelled environmental effects [39]. Navigation systems relying on idealized sensor measurements may become unstable when exposed to noisy or delayed observations, while controllers tuned in simulation may exhibit oscillatory or overly aggressive behavior on real UAV hardware.

Several approaches have been proposed to improve sim-to-real transferability. One widely used strategy is *domain randomization*, in which simulation parameters such as sensor noise, lighting conditions, friction coefficients, and actuator delays are randomized during training or evaluation [49]. By exposing algorithms to a broader distribution of conditions in simulation, the resulting policies become more robust to unseen real world variations. Another commonly used approach is system identification, where simulation parameters and dynamic models are tuned to closely match the measured behavior of the physical UAV platform [20].

In the context of aerial robotics, simulation environments such as Webots, Gazebo, AirSim, and PyBullet are frequently used for algorithm development and validation [14, 43]. These simulators provide realistic rigid-body physics, sensor emulation, and ROS integration while allowing rapid and reproducible experimentation. Previous work involving the Crazyflie platform has similarly

used simulation environments to validate navigation and SLAM algorithms before deployment on physical nano-UAV systems [31].

2.6 Exploration and Path Planning

Autonomous exploration and path planning are fundamental components of indoor UAV navigation systems. After a map representation has been generated through SLAM, the robot must determine where to move next while safely avoiding obstacles and efficiently covering unknown areas [44, 23]. In robotics literature, path planning generally refers to the process of generating collision free trajectories between positions, while exploration focuses on selecting informative regions that maximize knowledge about the unknown environment.

Classical path planning algorithms can broadly be divided into graph based, sampling based, and optimization based approaches [23]. Graph based methods such as Dijkstra's algorithm and A* search discretize the environment into a graph or occupancy grid and compute the shortest traversable route between nodes. Among these approaches, A* has become one of the most widely used planning algorithms in robotics due to its balance between computational efficiency and path optimality [18]. A* extends Dijkstra's algorithm by introducing heuristic cost estimation, allowing the planner to prioritize nodes closer to the target and significantly reduce search complexity.

Sampling based methods such as Rapidly exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM) instead construct feasible trajectories through random sampling of the configuration space [25]. These approaches are particularly effective in high dimensional environments and dynamic scenarios, but often require post-processing to generate smooth trajectories suitable for UAV flight. Optimization based planners additionally attempt to minimize trajectory cost functions such as travel distance, energy consumption, or collision risk [44].

For autonomous exploration in unknown environments, frontier based exploration remains one of the most commonly used strategies [54]. In frontier exploration, the robot identifies *frontiers*, defined as the boundaries between explored and unexplored regions within the occupancy map. The robot then selects one of these frontiers as the next exploration target. Frontier based methods are computationally efficient and particularly suitable for indoor mapping tasks because they naturally guide the robot toward unexplored areas while gradually increasing environmental coverage.

Several recent studies have further extended frontier exploration for UAV systems. Li et al. [28] combined frontier exploration with information gain optimization to improve exploration efficiency during active SLAM. Similarly, Zhou et al. [60] proposed a hierarchical frontier exploration framework for UAVs that combines global coverage planning with local trajectory refinement to accelerate exploration in unknown environments. Other work has additionally investigated frontier prioritization, viewpoint selection, and three-dimensional exploration strategies for aerial robots [59, 57].

Once a target location has been selected through frontier exploration, a local path following controller is required to execute the trajectory safely. In mobile robotics and autonomous driving research, Pure Pursuit is one of the most commonly used geometric path following algorithms [12]. The controller operates by selecting a lookahead point along the planned path and continuously steering the robot toward this point. Due to its relatively low computational complexity and smooth trajectory generation, Pure Pursuit is widely used in real time robotic navigation systems.

Pure Pursuit is particularly attractive for micro-UAV platforms because it does not require computationally expensive trajectory optimization or complex dynamic models [23]. The algorithm

naturally generates smooth curved motion rather than abrupt directional changes, making it suitable for lightweight aerial vehicles with limited dynamic stability. However, Pure Pursuit performance strongly depends on the chosen lookahead distance, which influences both tracking accuracy and trajectory smoothness.

3 Methods

This section describes the methodology and system architecture used throughout the experiments. The research focuses on autonomous indoor exploration using a Crazyflie 2.1+ nano quadcopter developed by Bitcraze [1]. Both simulated and real-world experiments were conducted to evaluate the performance of different navigation and exploration strategies under controlled indoor conditions.

To enable simulation-to-real transfer, a unified robotics framework was developed using ROS2 as the communication middleware and Webots as the simulation environment. The Crazyflie platform was equipped with a Flowdeck V2 and Multi-Ranger deck to provide onboard localization, obstacle detection, and autonomous navigation capabilities. Multiple exploration and path planning strategies were implemented, including reactive wall following, frontier based exploration, A* path planning, and Pure Pursuit trajectory tracking.

The following sections describe the hardware platform, sensor systems, simulation environment, communication architecture, navigation algorithms, and the adaptations required to transfer the developed algorithms from simulation to the real drone platform.

3.1 Crazyflie2.1+

The Crazyflie 2.1+ is a compact open-source nano quadcopter developed by Bitcraze for robotics research and education. Due to its lightweight design, modular architecture, and extensive software support, it has become a widely used platform for indoor autonomous navigation and swarm robotics research. The platform measures approximately 92×92×92 mm and weighs approximately 27 g without expansion decks. The Crazyflie ecosystem additionally provides extensive Python APIs and ROS interfaces, which makes integration with external controllers and simulation environments relatively straightforward. Another important advantage of the platform is its modularity. This is because expansion decks can easily be attached or replaced, reducing maintenance complexity and allowing rapid prototyping of new sensing configurations.

For indoor localization, the Crazyflie platform commonly relies on external positioning systems such as the Loco Positioning System (LPS) based on Ultra-Wideband (UWB) ranging or the Lighthouse positioning system using infrared base stations [4]. The Lighthouse system is capable of millimeter level precision within a confined environment, while the LPS provides centimeter level localization accuracy in larger indoor spaces. However, in this project no external positioning system was used during autonomous exploration. Instead, the drone relied solely on onboard sensing using the Flowdeck V2 and Multi-Ranger deck shown in Figure 1.

The Crazyflie firmware contains an onboard state estimation pipeline based on an Extended Kalman Filter (EKF). Kalman filtering is commonly used in robotics for estimating hidden system states from noisy sensor measurements [48]. Unlike the standard Kalman filter, the EKF can operate on non-linear systems by linearizing the state transition and measurement models around the current estimate. The filter recursively performs two main steps: prediction and correction.



Figure 1: Crazyflie 2.1+ drone equipped with Multi-Ranger Deck and Flow Deck V2

The prediction step estimates the next system state based on the previous state and the system dynamics:

$$\hat{\mathbf{x}}_{k|k-1} = f\left(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}\right) \tag{1}$$

where $\hat{\mathbf{x}}_{k|k-1}$ is the predicted state estimate, $f(\cdot)$ represents the non-linear motion model, and u_k denotes the control input. The covariance prediction is computed as:

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \tag{2}$$

where F_k is the Jacobian matrix of the motion model and Q_k represents process noise covariance.

During the correction step, incoming sensor measurements are fused into the state estimate:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \left(\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} \tag{3}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \left(\mathbf{z}_k - h\left(\hat{\mathbf{x}}_{k|k-1}\right)\right) \tag{4}$$

where K_k is the Kalman gain, H_k is the measurement Jacobian, R_k is the measurement noise covariance matrix, and z_k represents the sensor measurements.

Within the Crazyflie firmware, the EKF fuses measurements from the inertial measurement unit (IMU), optical flow sensor, and Time of Flight sensors to estimate the drone position, velocity, and orientation in real time [30]. The onboard estimation system significantly improves flight stability and allows the platform to operate in GPS denied indoor environments.

The Crazyflie control architecture itself is hierarchical. Low-level altitude stabilization is implemented onboard using PID controllers operating at high frequency, while high-level velocity or position commands are sent externally through ROS2 and Python interfaces. This separation allows computationally expensive navigation algorithms to run on an external computer while maintaining stable flight onboard the drone.

3.1.1 Flowdeck V2

The Flowdeck V2 shown in Figure 2 enables relative positioning and stable hovering in indoor environments where GPS signals are unavailable. The deck combines two primary sensors: a PMW3901 optical flow sensor and a VL53L1x Time of Flight (ToF) distance sensor [2]. The optical flow sensor estimates horizontal motion by tracking changes in texture between consecutive image frames captured from the ground surface. The ToF sensor measures altitude by determining the travel time of emitted infrared light pulses reflected from the ground.

The PMW3901 sensor estimates motion in the horizontal plane by computing optical flow vectors. Optical flow describes the apparent motion of visual features between consecutive frames [19]. Assuming brightness constancy, the optical flow constraint equation can be written as:

$$I_x u + I_y v + I_t = 0 \tag{5}$$

[19]

where I_x and I_y are spatial image gradients, I_t is the temporal intensity gradient, and (u, v) represents image motion velocity. The resulting motion estimate is combined with altitude information from the ToF sensor to estimate translational velocity relative to the ground.

The VL53L1x ToF sensor measures altitude by emitting infrared light pulses and measuring the round trip propagation delay. The sensor can measure distances up to approximately 4 m with millimeter-level precision under favorable conditions [2]. These measurements are fused within the onboard EKF to stabilize hovering and improve short-term odometry estimation.

The Flowdeck V2 is particularly useful in indoor navigation because it allows stable autonomous flight without requiring external motion capture systems. However, optical flow estimation depends heavily on surface texture and lighting conditions. Uniform surfaces or reflective flooring can reduce estimation accuracy and introduce drift over time.



Figure 2: Flow Deck V2 for Crazyflie

3.1.2 Multiranger

The Multi-Ranger deck shown in Figure 3 extends the spatial awareness capabilities of the Crazyflie by providing obstacle distance measurements in multiple directions. The deck contains five VL53L1x

Time of Flight sensors oriented toward the front, back, left, right, and upward directions [3]. Each sensor measures the distance to nearby obstacles using infrared ranging.

The VL53L1X sensors used in the Multi-Ranger deck operate with a relatively narrow field of view, approximately 27° according to the manufacturer specifications of the underlying sensor [45], allowing the drone to estimate nearby wall distances and avoid collisions. By combining measurements from all sensors, the platform obtains a sparse representation of the surrounding environment. The sensors can detect objects up to approximately 4 m away under optimal conditions [3].

The Multi-Ranger deck operates entirely onboard the drone and provides low latency obstacle measurements that can be used directly for reactive navigation behaviors such as wall following and collision avoidance. In this project, only the horizontal sensors were used for mapping and navigation purposes. Although the upward facing sensor could theoretically support full 3D SLAM and volumetric mapping, implementing reliable 3D reconstruction on a resource constrained nano drone significantly increases computational complexity and falls outside the scope of this research.

Since the platform does not contain a conventional RGB camera or depth camera, the range sensors provide the primary environmental perception mechanism. Therefore, the drone only observes sparse distance measurements rather than dense visual information. This limitation directly influences the navigation strategy and path planning algorithms used throughout the project.

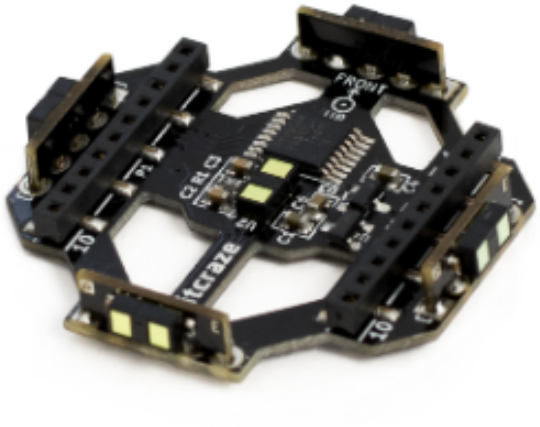


Figure 3: Multi-ranger Deck for Crazyflie

3.2 ROS2

Communication and control within the system are handled using the Robot Operating System 2 (ROS 2). ROS 2 is an open-source robotics middleware framework designed to support modular, distributed, and real-time capable robotic systems [41]. Unlike traditional massive robotics software architectures, ROS 2 decomposes robotic applications into separate nodes that communicate through standardized interfaces such as topics, services, and actions. This modular architecture simplifies development, testing, and scalability of complex robotic systems.

ROS 2 was selected for this project for several reasons. First, ROS 2 provides hardware abstraction and a standardized communication layer, allowing the same navigation algorithms to

operate both in simulation and on the physical Crazyflie platform with minimal modifications. Second, ROS 2 natively supports distributed computation, enabling computationally intensive tasks such as mapping and path planning to run on an external computer while low-level stabilization remains onboard the drone. Third, ROS 2 provides extensive support for robotics research through existing packages for localization, navigation, visualization, and sensor integration. Finally, ROS 2 improves upon the original ROS framework by introducing better real-time support, improved security, and middleware abstraction using the Data Distribution Service (DDS) communication standard [47, 53].

ROS 2 communication is primarily based on a publish-subscribe architecture. In this communication model, nodes publish data to named topics while other nodes subscribe to these topics asynchronously. This decouples individual software components and allows sensor processing, mapping, and control modules to operate independently. The underlying DDS middleware handles message serialization, discovery of nodes, and transport between processes or computers [47]. This architecture is particularly beneficial for robotics applications where multiple sensors and controllers must exchange data with low latency and high reliability.

The ROS 2 distribution used in this project is ROS 2 Kilted Kaiju. Communication with the Crazyflie drone is performed using the Crazyflie Python library (CFlib) together with ROS 2 wrapper nodes. These nodes publish sensor data and subscribe to velocity commands. For example, distance measurements from the Multi-Ranger deck are converted into `/scan` messages, while velocity commands are transmitted through the `/cmd_vel` topic. Additionally, odometry estimates generated from the Flowdeck V2 and onboard Extended Kalman Filter are published through the `/odom` topic.

An important advantage of this architecture is that both the Webots simulation environment and the physical Crazyflie platform expose identical ROS 2 interfaces. This makes it so that the same navigation stack and path planning algorithms can operate in both environments without modification. This significantly simplifies simulation-to-real transfer and enables fast validation of navigation algorithms before deployment on the real drone.

3.3 Webots

To validate navigation algorithms before deployment on the physical drone, the Webots robotics simulator was used as the primary simulation environment. Webots is an open-source multi-platform robotics simulator developed by Cyberbotics that provides a complete environment for modelling, programming, and simulating robotic systems [14, 32]. The simulator is widely used in robotics research because it supports realistic rigid-body physics, configurable sensors, ROS2 integration, and rapid prototyping of autonomous systems.

Webots was selected for this project for several reasons. First, the simulator already provides a built-in Crazyflie model together with support for common robotics sensors and actuators. Second, Webots integrates natively with ROS2, which enables the same control architecture to operate both in simulation and on the real Crazyflie platform. Third, the simulator provides deterministic and reproducible experiments, allowing navigation algorithms to be evaluated safely before deployment on real hardware. Finally, Webots supports Python controllers and custom world generation, making it suitable for rapid iterative development of indoor exploration algorithms [14, 32].

The simulation environment consists of simplified indoor apartment like environments based on the `crazyflie_apartment` example provided by Bitcraze and Cyberbotics. Within this environment,

the Crazyflie model is equipped with virtual equivalents of the Flowdeck V2 and Multi-Ranger deck sensors. The simulator models the Time-of-Flight range sensors, inertial measurements, and optical flow estimation used by the physical drone. Additionally, Webots provides a virtual RGB camera, although this camera was not used within the scope of this project because the real Crazyflie platform does not contain an onboard RGB vision system.

The underlying physics engine of Webots is based on the Open Dynamics Engine (ODE), which simulates rigid-body dynamics, collisions, friction, inertia, and actuator forces [14]. This enables realistic simulation of quadcopter motion and interactions with the environment. However, several real world effects are simplified or omitted. For example, aerodynamic turbulence, airflow interactions, battery voltage fluctuations, and motor wear are not explicitly modelled. Since the experiments were performed in relatively controlled indoor environments, the omission of wind simulation was considered acceptable for this project. Nevertheless, these simplifications contribute to the simulation-to-real gap discussed in Section 3.6.

The Crazyflie in Webots is stabilized using a cascaded PID (Proportional-Integral-Derivative) control architecture similar to the real Crazyflie firmware. PID controllers are widely used in robotics and control systems because they provide a computationally efficient method for regulating system dynamics. The controller computes a control signal according to:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt} \tag{6}$$

where $e(t)$ is the control error, K_p is the proportional gain, K_i is the integral gain, and K_d is the derivative gain. The proportional component reacts to the current error, the integral term compensates for accumulated steady-state errors, and the derivative term dampens oscillatory behavior by responding to the error rate of change.

Within the Crazyflie control stack, multiple PID controllers are typically arranged hierarchically. Low-level attitude controllers stabilize roll, pitch, and yaw angles, while higher level velocity and position controllers generate reference setpoints [5]. In the Webots simulation, the PID parameters were manually tuned to approximate the flight characteristics of the real Crazyflie platform. This tuning process aimed to reduce differences in acceleration, turning behavior, and hovering stability between simulation and reality.

The simulation environment was executed within Windows Subsystem for Linux 2 (WSL2). Running ROS2 and Webots inside WSL2 simplified package management and ensured compatibility with the Linux based ROS2 ecosystem while still allowing development on a Windows host system. Communication between Webots and ROS2 nodes occurs through the DDS middleware used by ROS2, enabling real time message exchange between the simulator and external control nodes.

Although Webots provides realistic rigid-body dynamics and sensor simulation, no artificial sensor noise or latency was initially introduced into the simulation environment. Real-world sensor measurements inevitably contain measurement uncertainty, timing jitter, and environmental disturbances. The absence of these effects results in an idealized simulation environment and partially explains performance differences observed between simulation and real-world experiments.

3.4 Custom Occupancy Grid Mapping

The Crazyflie platform does not provide sufficient sensing capabilities to directly use traditional LiDAR-based SLAM approaches. Instead, a custom occupancy grid mapping pipeline was developed

that uses only the distance measurements from the Multi-Ranger deck together with the pose estimate generated by the onboard Extended Kalman Filter and Flowdeck odometry.

The environment is represented as a two-dimensional occupancy grid in which each cell stores the probability of being occupied. Occupancy grids are a widely used representation in mobile robotics because they allow sensor measurements from multiple viewpoints to be integrated into a consistent map while naturally handling uncertainty [16].

The implemented map covers a fixed size area around the drone and is discretized into square cells. During operation, each Multi-Ranger measurement is transformed from the local drone frame into world coordinates using the current pose estimate:

$$\mathbf{p}_{world} = \mathbf{R}(\theta)\mathbf{p}_{sensor} + \mathbf{t}, \tag{7}$$

where $\mathbf{R}(\theta)$ is the rotation matrix corresponding to the drone heading θ, $\mathbf{p}_{sensor}$ is the measured obstacle position in the local sensor frame, and $\mathbf{t}$ is the current position estimate of the drone.

For every valid range measurement, the endpoint of the sensor beam is marked as occupied while all cells between the drone and the detected obstacle are marked as free. This process is repeated for all four horizontal Multi-Ranger sensors. The occupancy state of a cell is updated using a log-odds formulation

$$L_t(m_i) = L_{t-1}(m_i) + \log\left(\frac{p(m_i|z_t)}{1 - p(m_i|z_t)}\right), \tag{8}$$

where $L_t(m_i)$ denotes the log-odds value of cell m_i after incorporating measurement z_t. Occupied observations increase the log-odds value whereas free-space observations decrease it. This recursive formulation allows measurements collected over time to reinforce or weaken occupancy beliefs without requiring storage of previous sensor readings [48].

Because the Multi-Ranger provides only four horizontal range measurements instead of a dense laser scan, a complete map cannot be generated from a single observation. The drone must therefore actively explore the environment while accumulating measurements from multiple viewpoints. As new observations are received, the occupancy grid gradually converges towards a representation of the room layout.

Unknown cells are initially assigned an occupancy value of -1, free cells are stored as 0, and occupied cells as 100, matching the standard ROS 2 occupancy-grid representation. The resulting map is continuously published as a `/map` topic and visualized in RViz2. This map is subsequently used by the frontier exploration and next-best-view planners to determine unexplored regions and compute navigation goals.

3.5 Exploration and Path Planning

The implemented autonomous navigation system combines both reactive and global navigation strategies to enable exploration of unknown indoor environments using only onboard distance sensors. The navigation stack consists of multiple layers, including reactive wall following, frontier-based exploration, global path planning using the A* algorithm, and local trajectory tracking using Pure Pursuit. This layered navigation pipeline allows the Crazyflie to autonomously explore previously unknown indoor environments using only sparse onboard distance sensing.

3.5.1 Reactive Wall Following

As an initial baseline navigation strategy, a reactive wall-following algorithm was implemented. Wall following is a classical reactive navigation technique commonly used in mobile robotics for obstacle avoidance and simple autonomous exploration. The approach relies directly on local sensor measurements without requiring a complete map of the environment.

Two variations of wall-following behavior were implemented. The first implementation is based on the default Bitcraze wall-following controller provided in the Crazyflie Webots demonstration environment [31]. In this approach, the drone maintains a desired distance from nearby walls while adjusting its yaw orientation whenever the measured side distance exceeds a threshold. This results in discrete rotational corrections that allow the drone to align itself parallel to walls.

The second implementation introduces smoother trajectory behavior by continuously steering toward or away from walls using curved velocity commands rather than abrupt yaw rotations. Instead of rotating in place, the drone adjusts its lateral velocity and heading simultaneously, producing smoother trajectories and reducing oscillatory motion near corners.

Since the Crazyflie does not possess a dense depth camera or LiDAR system, navigation decisions are based entirely on sparse range measurements. This reactive wall following provides computational simplicity and robustness in narrow indoor corridors. However, purely reactive methods suffer from several limitations, including inefficient exploration paths, inability to reason globally about unexplored regions, and susceptibility to local minima [54]. Therefore, more advanced exploration strategies were also implemented.

3.5.2 Frontier-Based Exploration

To improve autonomous exploration performance, a frontier based exploration strategy was implemented. Frontier exploration was originally introduced by Yamauchi [54, 55]. In frontier based exploration, the robot incrementally constructs an occupancy grid map of the environment and identifies frontier cells, defined as the boundary between explored free space and unexplored regions.

The occupancy grid representation models the environment as a discretized probabilistic map in which each cell is classified as occupied, free, or unknown. As new Multi-Ranger measurements are received, the occupancy grid is updated to reflect newly observed obstacles and free space.

Frontiers are detected by searching for free cells adjacent to unknown cells. These frontier regions represent candidate exploration targets because moving toward them maximizes the expected information gain regarding the unknown environment [54]. Once a frontier is selected, the planner computes a collision-free trajectory toward the target.

Compared to reactive wall following, frontier exploration significantly improves exploration efficiency because the robot explicitly reasons about unexplored regions rather than merely reacting to nearby obstacles. Frontier-based methods have become one of the most widely used exploration strategies in autonomous robotics due to their simplicity and effectiveness [26].

Additionally, a Next-Best-View (NBV) strategy was investigated. In NBV exploration, the robot selects future viewpoints that maximize expected information gain while minimizing travel cost. This extends classical frontier exploration by incorporating utility based viewpoint selection rather than simply choosing the nearest frontier [10].

3.5.3 A* Path Planning

Global path planning between the current robot position and selected frontier targets is performed using the A* search algorithm. A* is one of the most widely used graph-search algorithms in robotics and autonomous navigation due to its computational efficiency and optimality properties [18].

The occupancy grid is interpreted as a weighted graph in which free cells correspond to traversable nodes. A* incrementally searches this graph by minimizing the evaluation function:

$$f(n) = g(n) + h(n) \tag{9}$$

where $g(n)$ represents the accumulated path cost from the start node to node n, and $h(n)$ is a heuristic estimate of the remaining distance to the goal. In this project, the Euclidean distance heuristic was used.

The A* algorithm guarantees an optimal path when the heuristic function is admissible [18]. Compared to uninformed search methods such as Dijkstra's algorithm, A* significantly reduces computational complexity by prioritizing nodes that are likely to lead toward the goal. The resulting path consists of a sequence of waypoints that guide the drone safely around mapped obstacles toward the selected frontier location.

3.5.4 Pure Pursuit Path Following

After generating a global path using A*, local trajectory tracking is performed using the Pure Pursuit algorithm. Pure Pursuit is a geometric path-following controller originally developed for autonomous vehicles and mobile robots [12].

The algorithm operates by selecting a look-ahead point located at a fixed distance along the planned path. Instead of directly steering toward the next waypoint, the controller continuously adjusts the drone trajectory toward this moving target point. This produces smooth motion and avoids abrupt directional changes.

The curvature required to reach the look-ahead point is computed geometrically according to:

$$\gamma = \frac{2x}{l^2} \tag{10}$$

where γ is the desired path curvature, x is the lateral displacement of the look-ahead point relative to the robot frame, and l is the look-ahead distance [12].

Pure Pursuit was selected because of its computational simplicity and robustness on resource-constrained robotic platforms. Furthermore, the algorithm naturally produces smooth trajectories, which is beneficial for quadcopter flight stability. The look-ahead distance can additionally be tuned to balance responsiveness and trajectory smoothness.

3.6 Simulation-to-Real Adaptation

One of the primary challenges in robotics research is the transfer of algorithms developed in simulation to physical robotic systems. Although simulation environments provide safe, reproducible, and computationally efficient testing conditions, simulated environments inevitably differ from real world conditions. This discrepancy is commonly referred to as the *simulation-to-real gap* or *reality gap* [9, 58]. Differences in sensing, actuation, environmental conditions, and physical modelling can

significantly influence the performance of autonomous navigation systems when transferred from simulation to physical hardware.

In this project, the same ROS2 based navigation architecture was used in both the Webots simulation environment and on the physical Crazyflie platform. Nevertheless, several parameters required adaptation when transitioning from simulation to the real drone. These modifications were necessary to compensate for differences between idealized simulation behavior and real world flight conditions.

One important limitation of the physical platform is battery capacity and voltage fluctuation. In simulation, the drone is effectively supplied with constant power, while in reality the Crazyflie battery voltage decreases during flight. This reduction in voltage directly affects motor thrust generation and flight stability, especially during aggressive maneuvers or longer exploration runs. Consequently, lower maximum velocities and more conservative acceleration parameters were used on the physical drone to improve stability and reduce battery consumption.

Another major source of discrepancy originates from sensor inaccuracies. The simulated sensors in Webots provide relatively clean and deterministic measurements, whereas the real sensors are affected by environmental noise, calibration errors, and lighting conditions. The optical flow sensor of the Flowdeck V2 is particularly sensitive to floor texture and illumination. Uniform surfaces or poor lighting conditions reduce optical flow accuracy and can introduce drift in the estimated velocity and position. Similarly, the Time of Flight sensors of the Multi-Ranger deck are influenced by reflective materials, sensor angle, and ambient infrared light sources [2, 3]. These effects are difficult to reproduce accurately in simulation.

Additionally, communication latency and timing jitter introduce further differences between simulation and reality. In simulation, ROS2 messages and control commands are processed with near deterministic timing. On the physical platform, wireless communication delays between the Crazyflie and the external computer can introduce small inconsistencies in control update frequency. Although these delays are relatively small, they can influence navigation stability during rapid maneuvers.

The physical dynamics of the drone also differ from the simplified simulation model. Webots approximates quadcopter motion using rigid-body physics and idealized actuator models. However, real quadcopters are affected by aerodynamic disturbances, motor response delays, air turbulence, vibrations, and minor manufacturing inconsistencies. These effects contribute to deviations between simulated and observed real world trajectories [9, 7]. Furthermore, the real Crazyflie experiences small variations in weight distribution depending on battery mounting and sensor deck placement, which can slightly alter flight characteristics.

To reduce the simulation-to-real gap, several practical adjustments were applied during deployment on the physical platform. First, PID controller gains and velocity constraints were manually tuned to better match real flight behavior. Second, exploration velocities were reduced to improve robustness against sensor uncertainty and communication latency. Third, obstacle avoidance thresholds were increased slightly to compensate for noisy range measurements and prevent collisions caused by inaccurate sensor readings.

4 Experiments and Results

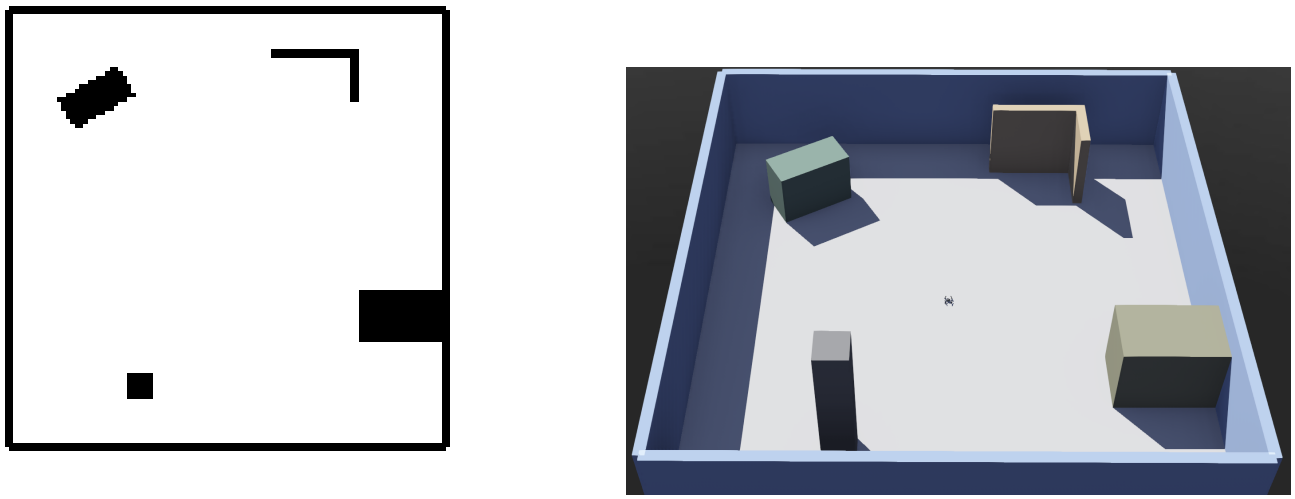
This chapter presents the experimental evaluation of the proposed mapping and object detection system. The evaluation is divided into two parts: simulation experiments and real world experiments.

The simulation experiments were conducted in the Webots simulator and served both as a development environment and as a controlled benchmark for quantitative evaluation. Two custom environments were used: a small 5×5 m room containing several obstacles and a larger, more complex environment designed to challenge the mapping and exploration capabilities of the drone. Four navigation behaviours were evaluated: a baseline wall-following controller based on the Bitcraze mapping tutorial [31], an improved wall-following strategy, Frontier Exploration, and Next Best View exploration. In addition to mapping performance, simulation experiments were also used to evaluate the proposed object detection approach.

The second part of this chapter focuses on real-world experiments using a physical Crazyflie drone. These experiments investigate the practical limitations of the onboard sensors, evaluate the mapping performance of the exploration behaviours under real operating conditions, and assess the feasibility of performing object detection using maps generated from onboard sensing alone.

4.1 Simulation Performance

The navigation and mapping algorithms were first developed and evaluated inside the Webots simulation environment before deployment on the physical Crazyflie platform. The simulation experiments were conducted in two custom indoor benchmark environments designed specifically for evaluating sparse-sensor SLAM and autonomous exploration performance. The first environment is a small benchmark room of 5×5 m containing several obstacles positioned to create narrow passages and partially occluded regions as can be seen in Figure 4.

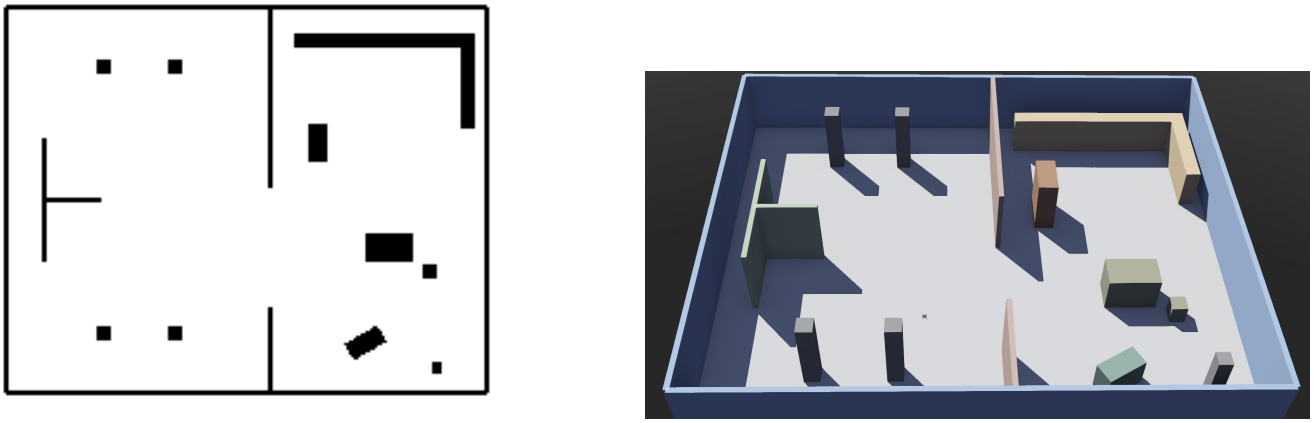


(a) Ground Truth

(b) Simulator Setup

Figure 4: Figure 4b shows the setup of the small 5×5 m benchmark room in the Webots simulator. The ground truth of that room can be seen in Figure 4a.

The second environment is a larger and more complex room intended to challenge the exploration and mapping capabilities of the drone under sparse sensing conditions. This larger benchmark room can be seen in Figure 5. Both benchmark environments were manually designed in Webots and remained identical throughout all experiments to ensure experimental consistency. The drone was always initialized at a fixed starting pose located near the center of the environment. Initial orientation was kept constant across all experiments to eliminate variance caused by randomized spawning conditions.



(a) Ground Truth

(b) Simulator Setup

Figure 5: Figure 5b shows the setup of the big benchmark room in the Webots simulator. The ground truth of that room can be seen in Figure 5a.

The simulator experiments were divided into three main stages. First, hyperparameter optimization was performed using both benchmark rooms. During this stage, the navigation and exploration parameters of each path planning behavior were tuned to maximize map coverage while maintaining stable navigation performance. Each parameter configuration was evaluated three times in both the small and large benchmark environments to reduce the effect of stochastic exploration behavior. The best performing parameter configuration was subsequently selected for the remaining experiments.

The second stage evaluated the performance of the four finetuned different exploration and path planning behaviors. Each behavior was evaluated three times in the large benchmark environment using the optimized parameter settings. During these experiments, the drone generated a two-dimensional occupancy grid map using the custom SLAM implementation described in Section 3.4. RViz2 was used throughout the experiments to visualize the occupancy grid, costmap, explored frontiers, and navigation trajectories in real time.

The final simulation experiment evaluated the object detection pipeline. In this experiment, the drone first explored and mapped the environment to create a reference occupancy grid. Subsequently, an additional object was inserted into the environment, after which the drone performed a second mapping run. The object detection algorithm compared both occupancy maps and attempted to identify structural differences between the two maps. Detection was performed using occupancy grid difference analysis combined with noise filtering techniques. Because the objective of this experiment was primarily proof of concept validation, the result was evaluated visually rather than quantitatively.

The drone will be evaluated using a few different metrics. Three evaluation metrics were used to assess mapping quality: Structural Similarity Index (SSIM), F1-score, and Hausdorff distance.

The Structural Similarity Index (SSIM) is an image similarity metric introduced by Wang et al. [51]. Unlike traditional pixel-wise error measures such as Mean Squared Error (MSE), SSIM evaluates perceived structural similarity between two images by comparing luminance, contrast, and structural information. The SSIM metric is defined as:

$$SSIM(x, y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{11}$$

where μ_x and μ_y represent the mean image intensities, σ_x^2 and σ_y^2 represent the variances, and σ_{xy} represents the covariance between the generated occupancy map and the ground-truth map. Constants C_1 and C_2 are used to stabilize the division operation for small denominators. SSIM values range from -1 to 1, where a value of 1 indicates perfect structural similarity [51, 52]. In the context of occupancy-grid mapping, SSIM measures how closely the generated map resembles the true spatial structure of the environment.

In addition to SSIM, the F1-score was used to evaluate occupancy classification accuracy. The occupancy grid was converted into a binary representation consisting of occupied and free cells. The F1-score combines precision and recall into a single metric and is defined as:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{12}$$

with:

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

where TP represents correctly identified occupied cells, FP represents falsely detected occupied cells, and FN represents missed occupied cells [42]. The F1-score provides a balanced evaluation of mapping accuracy by simultaneously penalizing false positives and false negatives.

Finally, Hausdorff distance was used to evaluate the geometric similarity between obstacle boundaries in the generated occupancy map and the corresponding ground-truth map. Unlike pixel based similarity metrics, Hausdorff distance measures the largest geometric deviation between two sets of points and is therefore particularly suitable for evaluating map accuracy and obstacle reconstruction quality [21]. Given two point sets A and B, the directed Hausdorff distance is defined as:

$$h(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\| \tag{15}$$

where $\|a - b\|$ denotes the Euclidean distance between points a and b. Intuitively, this metric measures the largest distance from any point in set A to its nearest neighbour in set B. The symmetric Hausdorff distance is then defined as:

$$H(A, B) = \max\left(h(A, B), h(B, A)\right) \tag{16}$$

which ensures that deviations are considered in both directions. In this work, the point sets correspond to obstacle boundary points extracted from the generated occupancy grid and the corresponding ground-truth map. Lower Hausdorff distances indicate that the mapped obstacles more closely match the true geometry of the environment, while larger values indicate greater localization or mapping errors [21].

4.1.1 Finetuning

The hyperparameter finetuning experiments were limited to a maximum runtime of 150 seconds. This time limit was selected because it approximately corresponds to half of the effective battery duration of the real Crazyflie platform during autonomous flight, thereby improving consistency between simulation and real-world experiments. The results of these experiments can be seen in Table 1. The values used for testing were chosen based on pre-testing and were expected to span the relevant operating range for each algorithm.

The `forward_speed` governs the cruise velocity of the drone in metres per second during autonomous navigation and is shared across all four behaviours. The `turn_speed` controls the angular velocity in radians per second used for heading corrections and wall-following turns; it appears in both the baseline yaw agent and the wall follower. The `wall_distance` is the target lateral stand-off distance the yaw agent tries to maintain from a detected wall, while `target_distance` serves the analogous role for the wall follower. The `inflation_radius` defines the minimum lethal clearance around occupied cells in the costmap, effectively setting how closely the planner is allowed to route the drone near walls; it is shared by both the frontier exploration and NBV planners as they use the same underlying A* path planner. The `frontier_min_size` sets the minimum number of cells a frontier cluster must contain to be considered a valid exploration goal: smaller values cause the planner to pursue finer boundary details, while larger values lead to earlier termination. Finally, `candidate_spacing` and `ray_count` are specific to the NBV planner. The `candidate_spacing` sets the grid resolution at which viewpoint candidates are sampled over the free space, and `ray_count` determines how many synthetic rays are cast per candidate when estimating information gain.

The results in Table 1 reveal several consistent trends across the four navigation behaviours. In general, higher mapping quality was achieved when the drone was able to cover a larger portion of the environment within the 150 second flight budget. This is reflected by the relationship between path length and mapping performance, configurations that produced longer trajectories often achieved higher F1 and SSIM scores, indicating improved environmental coverage.

For the baseline wall following behaviour, the influence of the turning speed was relatively modest. Increasing the turn speed from 0.2 to 0.4 rad/s generally improved both F1 and SSIM, particularly in the larger benchmark environment. Faster turning allowed the drone to react more effectively when approaching corners and reduced the likelihood of cutting corners too aggressively. Similarly, higher forward speeds increased the explored area within the available flight time, resulting in higher structural similarity scores. However, the highest forward speed also introduced a slight reduction in F1 score in the large environment, suggesting that increased velocity can lead to reduced sensor coverage near walls. A wall distance of 0.4 m was ultimately selected because it provided the most consistent performance across both environments while maintaining safe clearance from obstacles.

Behaviour	Parameter	Values	Small Room			Large Room		
			F1	SSIM	Path (m)	F1	SSIM	Path (m)
Baseline	turn_speed (rad/s)	0.2	0.667	0.337	**6.3**	0.612	0.247	**9.1**
		0.3	**0.722**	0.345	7.0	0.666	0.273	9.9
		0.4	0.709	**0.350**	7.5	**0.742**	**0.277**	10.6
	forward_speed (m/s)	0.1	**0.680**	0.337	**5.9**	0.636	0.242	**9.2**
		0.2	0.607	0.449	9.1	**0.675**	0.205	10.1
		0.3	0.635	**0.568**	14.3	0.556	**0.248**	13.8
	wall_distance (m)	0.4	**0.695**	0.343	**6.0**	0.597	**0.241**	9.0
		0.6	0.632	0.349	7.4	0.700	0.229	7.9
		0.8	0.631	**0.354**	7.2	**0.703**	0.223	**7.7**
Wall Follower	forward_speed (m/s)	0.1	0.756	0.506	**11.6**	0.713	0.258	**12.2**
		0.2	0.791	0.649	21.0	**0.747**	0.274	22.3
		0.3	**0.816**	**0.663**	28.3	0.685	**0.275**	31.4
	target_distance (m)	0.3	0.557	0.223	12.2	**0.769**	0.255	**11.7**
		0.5	**0.752**	**0.506**	**11.5**	0.706	0.257	11.9
		0.7	0.737	0.398	12.0	0.727	**0.264**	11.9
	turn_speed (rad/s)	0.2	0.694	0.352	**9.2**	0.636	0.233	**11.5**
		0.4	0.739	0.453	11.3	0.699	0.256	11.7
		0.6	**0.755**	**0.510**	11.5	**0.699**	**0.259**	12.0
Frontier Exploration	inflation_radius (m)	0.25	0.774	0.579	11.7	0.772	0.324	**12.9**
		0.35	**0.809**	**0.619**	14.3	**0.773**	**0.328**	13.8
		0.45	0.734	0.555	**10.3**	0.740	0.299	13.5
	forward_speed (m/s)	0.1	0.776	0.523	**9.3**	**0.790**	0.247	**9.0**
		0.2	0.818	0.579	12.4	0.786	0.353	14.4
		0.3	**0.835**	**0.636**	15.6	0.761	**0.374**	18.2
	frontier_min_size	2	**0.818**	0.605	13.0	**0.818**	0.300	13.3
		4	0.802	**0.639**	13.8	0.782	**0.320**	14.1
		8	0.751	0.486	**6.5**	0.780	0.252	**12.2**
Next Best View	candidate_spacing (m)	0.3	0.752	0.611	16.0	0.690	**0.395**	18.4
		0.4	0.732	0.392	**3.3**	**0.738**	0.391	19.8
		0.5	**0.808**	**0.649**	18.8	0.700	0.288	**14.3**
	inflation_radius (m)	0.25	**0.849**	0.637	17.0	**0.796**	**0.372**	19.4
		0.35	0.734	0.392	**3.3**	0.751	0.357	19.6
		0.45	0.807	**0.639**	19.9	0.754	0.360	**18.8**
	ray_count	18	**0.815**	**0.632**	19.3	0.772	0.358	19.8
		36	0.712	0.392	**3.3**	**0.790**	**0.376**	19.3
		72	0.749	0.470	8.3	0.779	0.371	**17.5**

Table 1: Hyperparameter finetuning results (averaged over three runs) across the small room (Figure 4) and large room (Figure 5). Bold indicates the best value per group. Drones limited to 150 s flight time. The baseline agent is based on the yaw wall follower from the Bitcraze Tutorial [31].

The wall follower exhibited substantially better mapping performance than the baseline behaviour. Unlike the baseline controller, which relies primarily on yaw corrections, the wall follower continuously steers towards the desired wall distance, resulting in smoother trajectories and improved coverage. Increasing the forward speed consistently improved both F1 and SSIM scores, with the best overall performance obtained at 0.3 m/s. Similarly, larger turn speeds improved the controller's ability to recover from deviations and negotiate corners. The selected target distance of 0.5 m provided the best compromise between maintaining wall visibility and avoiding collisions.

For frontier exploration, the results demonstrate the importance of balancing exploration efficiency and obstacle clearance. An inflation radius of 0.35 m produced the best overall mapping quality. Smaller values allowed the planner to approach walls more closely but increased the risk of

getting the drone stuck in narrow passages, while larger values reduced the amount of free space available for exploration. Increasing the forward speed improved mapping performance in the small room because more frontiers could be reached within the time limit. In the larger room, however, the highest speed slightly reduced F1 score despite increasing SSIM, indicating that the planner occasionally sacrificed local map accuracy for faster exploration. Frontier clusters smaller than four cells often represented sensor noise, while larger thresholds caused legitimate exploration goals to be ignored. Consequently, a frontier minimum size of four cells was selected.

The Next Best View planner showed the highest sensitivity to parameter selection. Several parameter combinations resulted in unusually short path lengths, indicating that the planner terminated prematurely after identifying insufficient candidate viewpoints. This behaviour is particularly visible for the candidate spacing of 0.4 m and the inflation radius of 0.35 m, where the drone travelled only approximately 3 m in the small room. Increasing the candidate spacing reduced the computational burden but also reduced the density of potential viewpoints. Likewise, excessively large inflation radiuses restricted the available free space and reduced the number of feasible viewpoints. The best overall mapping performance was achieved using a candidate spacing of 0.3 m, an inflation radius of 0.25 m, and 18 information-gain rays. Increasing the ray count beyond this value did not improve mapping quality significantly but increased computational complexity.

Based on these observations, the final parameter configurations listed in Table 2 were selected for all subsequent experiments. The chosen values do not always correspond to the absolute best score in a single benchmark. Instead, they represent the most robust configurations across both environments while maintaining stable behaviour and avoiding premature exploration termination.

Behaviour	**Parameter**	**Value**	**Unit**
Baseline	`turn_speed`	0.4	rad/s
	`forward_speed`	0.3	m/s
	`wall_distance`	0.4	m
Wall Follower	`forward_speed`	0.3	m/s
	`target_distance`	0.5	m
	`turn_speed`	0.6	rad/s
Frontier Exploration	`inflation_radius`	0.35	m
	`forward_speed`	0.3	m/s
	`frontier_min_size`	4	–
Next Best View	`candidate_spacing`	0.3	m
	`inflation_radius`	0.25	m
	`ray_count`	18	–

Table 2: Final parameter configurations selected for each mapping behaviour based on the finetuning results in Table 1. These values are used in all subsequent benchmark comparison experiments.

4.1.2 Pathing behavior

The second simulation experiment evaluated the four navigation behaviours in the large benchmark environment shown in Figure 5. In contrast to the hyperparameter tuning experiments, the flight time limit was increased to 300 s, corresponding approximately to the full operational battery life of the Crazyflie. This experiment aimed to evaluate the long term mapping performance of each behaviour using the optimized parameters listed in Table 2.

Figure 6 summarizes the average results over three runs for each behaviour. Frontier Exploration achieved the highest overall mapping performance, obtaining an average F1 score of 0.765 and an SSIM score of 0.546. Next Best View (NBV) achieved the second-best performance with an F1 score of 0.742 and an SSIM score of 0.492. The wall-following approaches performed noticeably worse, with F1 scores of 0.677 and 0.658 for the wall follower and baseline respectively. Similar trends are observed for the SSIM metric, indicating that the exploration-based methods produce occupancy maps that more closely resemble the ground-truth environment.

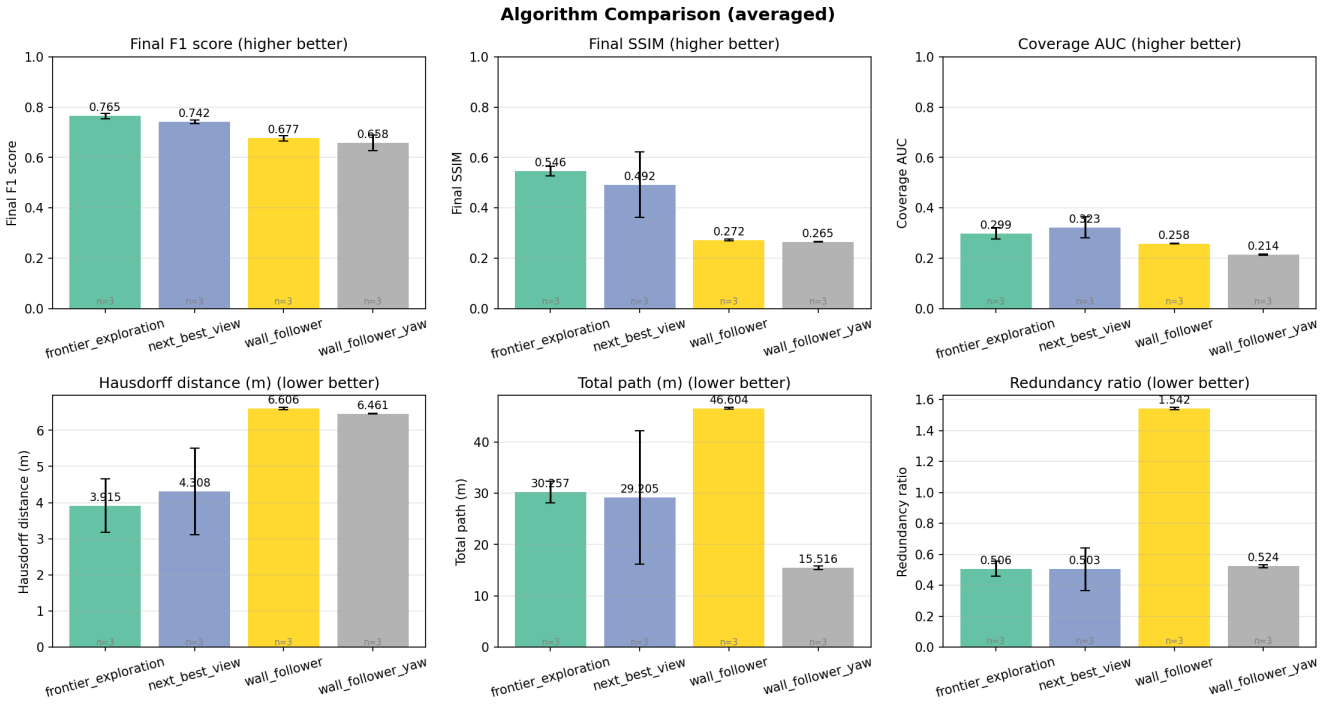


Figure 6: Average performance of the four navigation behaviours on the large benchmark environment. Results are averaged over three runs with a maximum flight time of 300 s. Error bars indicate the standard deviation between runs. The wall_follower_yaw is the baseline agent.

While the wall-following behaviours rapidly explore the immediate surroundings of the starting location, they eventually become confined to repeatedly traversing the same regions of the map. In contrast, Frontier Exploration and NBV continue to discover new unexplored areas throughout the experiment. This behaviour is reflected in their higher coverage AUC values and improved map quality scores.

Frontier Exploration achieved the lowest average Hausdorff distance of 3.915 m, indicating the closest geometric agreement with the ground truth obstacle layout. NBV achieved a similar value

of 4.308 m but exhibited greater variation between runs. The wall-following behaviours produced substantially larger errors, with Hausdorff distances exceeding 6 m. These results indicate that exploration-based strategies not only increase map coverage but also improve the geometric accuracy of the reconstructed environment.

The wall follower travelled the greatest distance on average (46.6 m), yet achieved lower coverage and map quality than both Frontier Exploration and NBV. This indicates that simply travelling farther does not necessarily improve mapping performance. Instead, the efficiency with which unexplored regions are selected plays a more important role. Frontier Exploration and NBV achieved better maps while travelling approximately 30 m, demonstrating a more effective use of the available flight time.

The wall follower achieved a redundancy ratio of 1.542, indicating that a large proportion of its travelled distance consisted of revisiting previously explored areas. In contrast, Frontier Exploration and NBV achieved ratios close to 0.5, suggesting that a much larger fraction of their trajectories contributed directly to discovering new map information.

Figure 7 provides additional insight into the mapping process over time. The coverage curves show that all behaviours initially explore their immediate surroundings at a similar rate. After approximately 150 s, however, the exploration based methods begin to outperform the wall-following approaches. Both Frontier Exploration and NBV continue to discover new regions throughout the experiment, whereas the wall-following behaviours plateau relatively early.

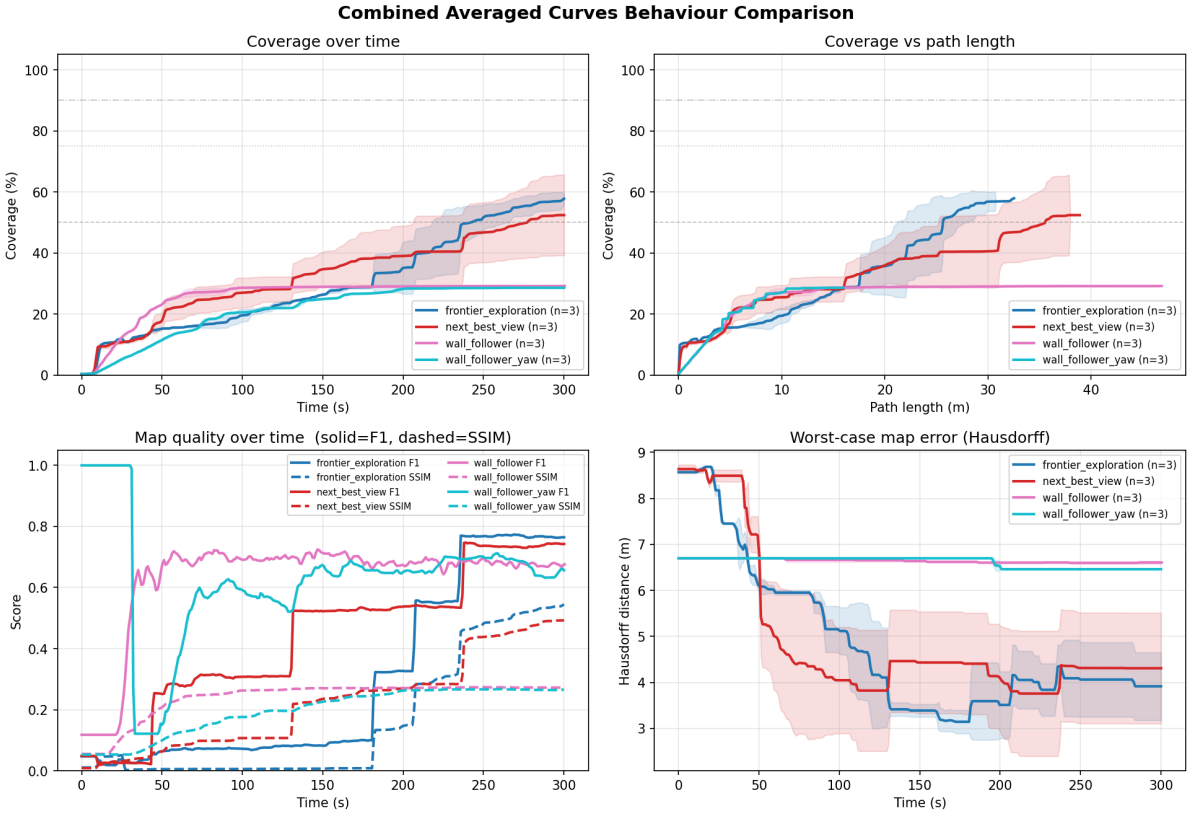


Figure 7: Comparison of coverage, map quality and Hausdorff error over time for the four navigation behaviours in the large benchmark room. Curves represent the average of three runs. The runs were limited to 300 s flight time.

The coverage versus path length plot demonstrates that all behaviours obtain a large fraction of their final coverage within the first 10 m of travel. Beyond this point, the baseline and wall follower exhibit only marginal improvements, indicating that they repeatedly traverse previously explored regions. Frontier Exploration and NBV, on the other hand, continue to gain coverage throughout the experiment.

The F1 and SSIM curves show a similar trend. At very early time steps, the baseline behaviour exhibits an artificially high F1 score because only a small number of occupied cells have been observed. As exploration progresses and additional free space and obstacle cells are incorporated into the map, the score temporarily decreases before converging towards its final value. Consequently, the first few measurements should not be interpreted as representative of the overall map quality. After approximately 50 s, the scores become more stable and meaningful for comparison.

Finally, the Hausdorff distance curves show that the wall-following behaviours achieve only minor reductions in geometric error throughout the experiment. In contrast, Frontier Exploration and NBV exhibit a substantial decrease in Hausdorff distance, reducing the worst-case map error from approximately 9 m to between 3 and 4 m. This indicates that these methods progressively improve the global structure of the occupancy map as additional regions are explored.

A closer examination of the individual behaviour plots provides additional insight into the strengths and weaknesses of each navigation strategy. The baseline behaviour (Figure 8) rapidly reaches a local optimum. After an initial adjustment period, all performance metrics stabilize relatively early and show little improvement for the remainder of the experiment. Coverage saturates after approximately 10 m of travel and the Hausdorff distance decreases only marginally, indicating that the behaviour struggles to discover new regions once the nearby environment has been explored.

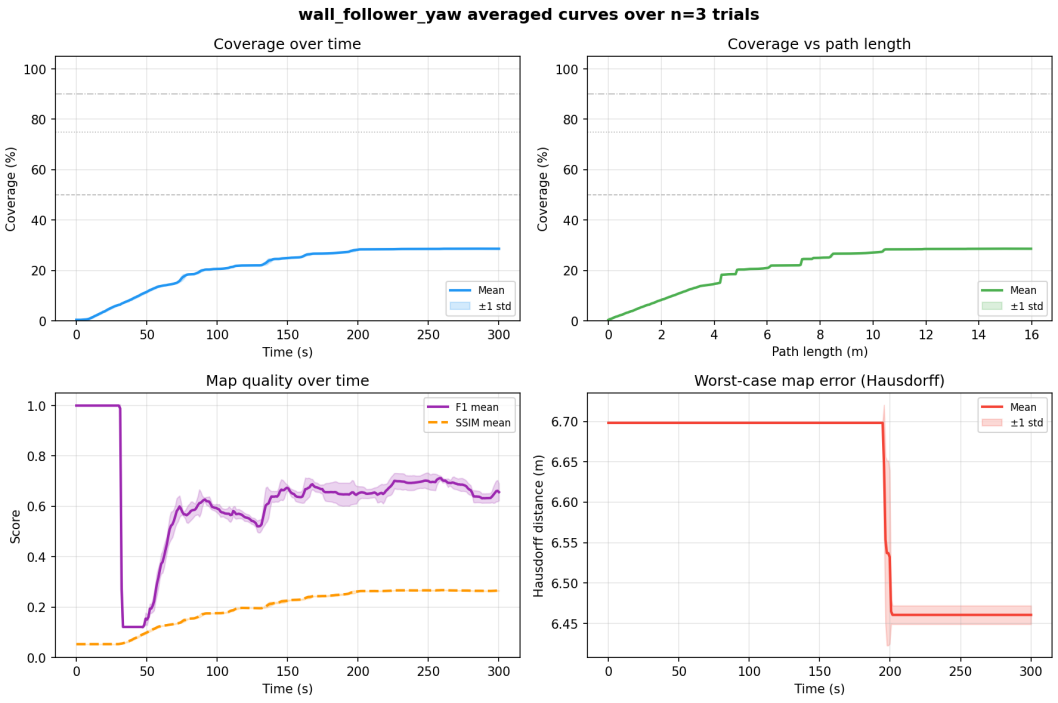


Figure 8: Average performance curves of the baseline wall-following behaviour averaged over three runs in the big benchmark room.

The wall follower (Figure 9) exhibits similar behaviour. Most improvements occur within the first 50 s, after which the coverage and map quality metrics plateau. Although the travelled distance continues to increase, little additional map information is acquired. This suggests that the drone repeatedly revisits already explored regions rather than seeking unexplored areas of the environment.

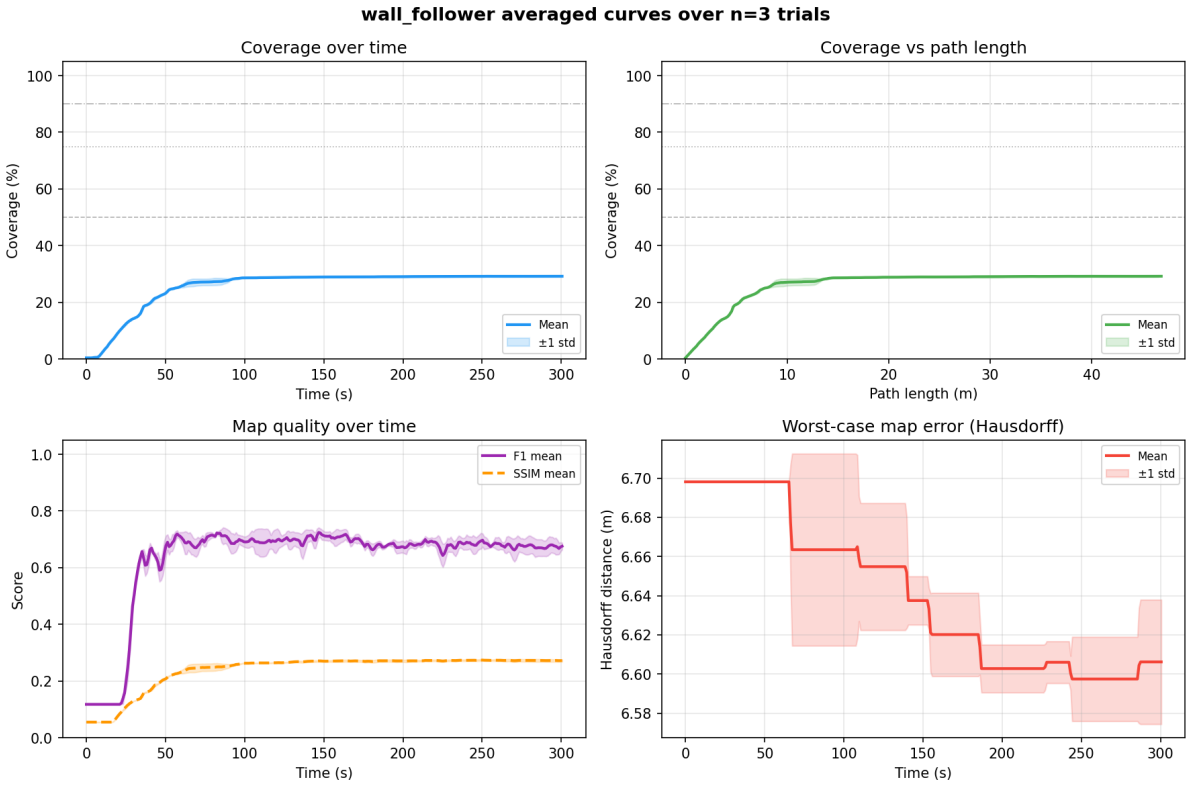


Figure 9: Average performance curves of the wall follower behaviour averaged over three runs in the big benchmark room.

Frontier Exploration (Figure 10) demonstrates a markedly different pattern. Both F1 and SSIM increase gradually throughout the experiment, reflecting the continuous acquisition of new information. The Hausdorff distance decreases substantially during the first half of the run, indicating that the global geometry of the map becomes increasingly accurate. Coverage also increases steadily and approaches 60% by the end of the experiment. These results suggest that the frontier based strategy efficiently directs the drone towards unexplored regions and avoids unnecessary revisits.

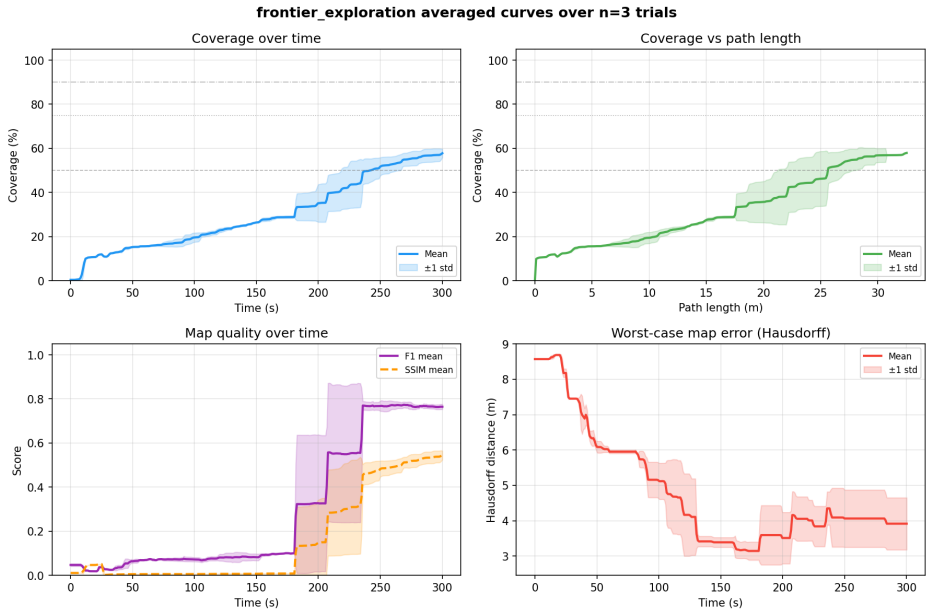


Figure 10: Average performance curves of the Frontier Exploration behaviour averaged over three runs in the big benchmark room.

The NBV behaviour (Figure 11) achieves performance comparable to Frontier Exploration but exhibits noticeably larger variance between runs. The map quality metrics improve in several distinct stages rather than gradually, suggesting that the information gain planner occasionally identifies highly informative viewpoints that result in sudden improvements to the occupancy map. Despite the higher variability, NBV consistently achieves high coverage and low geometric error.

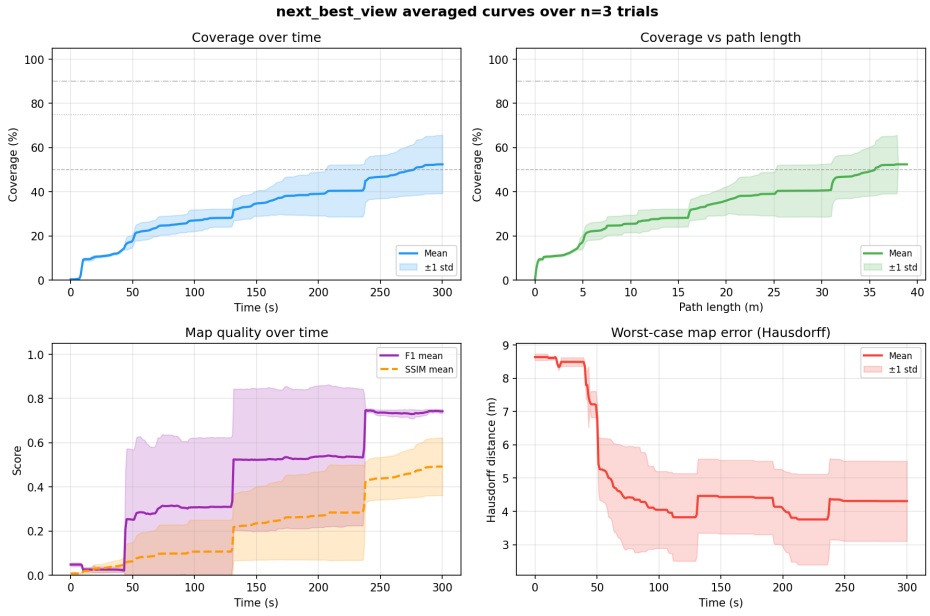


Figure 11: Average performance curves of the Next Best View behaviour averaged over three runs in the big benchmark room.

4.1.3 Object Detection

To evaluate whether the generated occupancy maps can be used for environmental change detection, a proof of concept object detection experiment was conducted in the small benchmark environment shown in Figure 4. The frontier exploration behaviour was selected for this experiment because it achieved the highest mapping performance during the benchmark evaluation.

The experiment consisted of two mapping phases. During Phase 1, the drone explored and mapped the environment without any additional objects present. After the first map had been generated, a large square obstacle was placed inside the room and the drone repeated the exploration task for Phase 2. Both phases were limited to 120 seconds to remain consistent with the battery constraints of the real Crazyflie platform while still allowing sufficient time for map generation.

The resulting occupancy grids are shown in Figure 12. The Phase 1 map closely resembles the ground truth environment presented in Figure 4a. In Phase 2, the newly introduced obstacle is clearly visible near the coordinates ($x \approx 1\,\text{m}$, $y \approx 0.5\,\text{m}$).

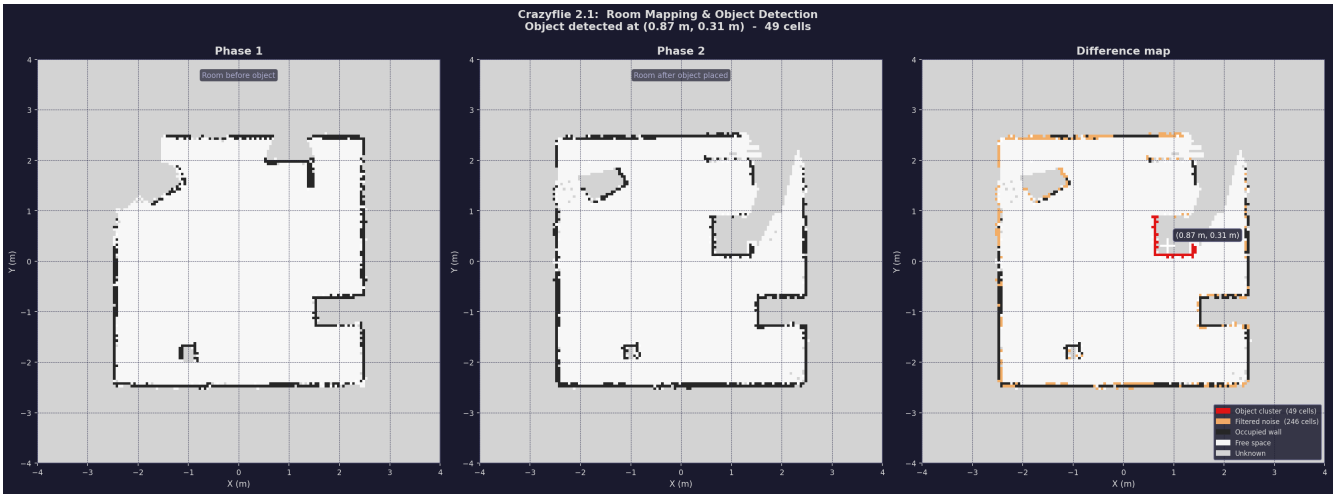


Figure 12: Object detection through occupancy-grid differencing using frontier exploration in the small benchmark environment. Phase 1 shows the occupancy map generated before an object was introduced. Phase 2 shows the map after a square obstacle was placed in the room. The difference map highlights cells that changed between both mapping sessions. Orange cells indicate differences classified as noise after filtering, while red cells indicate clusters identified as potential objects. The detected cluster corresponds to the location of the newly introduced obstacle.

To identify environmental changes, the occupancy grids from both phases were aligned and compared cell by cell. Newly occupied cells in Phase 2 that were previously free in Phase 1 were considered candidate object detections. However, occupancy grid maps generated from sparse ToF measurements are subject to small localization errors and sensor noise. As a result, direct differencing produces many false positives near existing walls and obstacle boundaries. To mitigate this effect, the algorithm first applies a proximity filter that removes differences occurring within a predefined distance of previously mapped walls. The remaining candidate cells are then grouped using connected component clustering. Small isolated clusters are classified as measurement noise, while larger clusters are retained as potential object detections.

The rightmost image in Figure 12 visualizes the output of this procedure. Orange cells represent

differences that were rejected as noise because they occurred close to existing walls or formed clusters below the minimum size threshold. Red cells correspond to clusters that satisfy the object detection criteria. A clearly identifiable cluster appears at the location of the inserted obstacle, demonstrating that occupancy grid differencing can successfully detect previously unknown objects using only the sparse range measurements.

4.2 Simulation-to-Real

4.2.1 Sensor limitations

Before evaluating SLAM performance in real world environments, it is important to establish a baseline understanding of the sensing capabilities of the Crazyflie platform. Unlike the simulator, real world operation introduces a variety of factors that can influence sensor performance, including lighting conditions, surface materials, viewing angles, and environmental noise. To characterize the reliability of the Multi-Ranger deck under practical operating conditions, two sensor validation experiments were conducted. The first experiment evaluated the effective maximum measurement range of the Time-of-Flight (ToF) sensors, while the second quantified the occurrence of measurement dropouts and anomalous range readings during flight.

Maximum Range Evaluation According to the manufacturer specifications, the VL53L1x Time-of-Flight sensors used on the Multi-Ranger deck can measure distances up to approximately 4 m under ideal conditions [3]. In practice, however, the effective measurement range depends strongly on target reflectivity, ambient illumination, sensor angle, and surface properties.

To determine the practical sensing range, the drone was placed in a large indoor corridor and commanded to fly slowly backwards while continuously recording the front-facing range sensor. As the drone moved away from the wall, the measured distance increased until the sensor began reporting out-of-range values. The drone subsequently approached the wall again to determine the maximum distance at which stable measurements could still be obtained. This procedure was repeated three times using a fully charged battery.

Metric	Mean	Std. Dev.	Minimum	Maximum
Maximum Range [m]	2.47	0.25	2.20	2.60

Table 3: Maximum effective sensing range measured for the front Multi-Ranger sensor.

Table 3 summarizes the observed maximum sensing ranges. The results indicate that the effective sensing range in a typical indoor environment is approximately 2.5 m, significantly lower than the theoretical maximum specified by the manufacturer. Consequently, obstacles located beyond approximately 2.5 m cannot reliably contribute to the occupancy map generated by the SLAM system. This limitation directly affects exploration performance because frontier detection and obstacle avoidance rely entirely on these range measurements.

Measurement Stability and Dropout Analysis A second experiment was conducted to investigate measurement reliability and sensor dropout behaviour. During this test, the drone

was manually rotated through multiple headings while continuously logging all four horizontal Multi-Ranger sensors. A total of 1411 sensor samples were collected. Out-of-range measurements were excluded from the statistical analysis because these values indicate that no valid reflection was detected.

Sensor	Mean [m]	Std [m]	Min [m]	Max [m]	Samples	Outliers
Front	1.336	0.499	0.220	2.495	1365	1.7
Left	1.066	0.538	0.529	2.645	1337	2.8
Back	1.691	0.699	0.451	2.635	1150	8.6
Right	0.908	0.425	0.405	2.427	1327	1.9

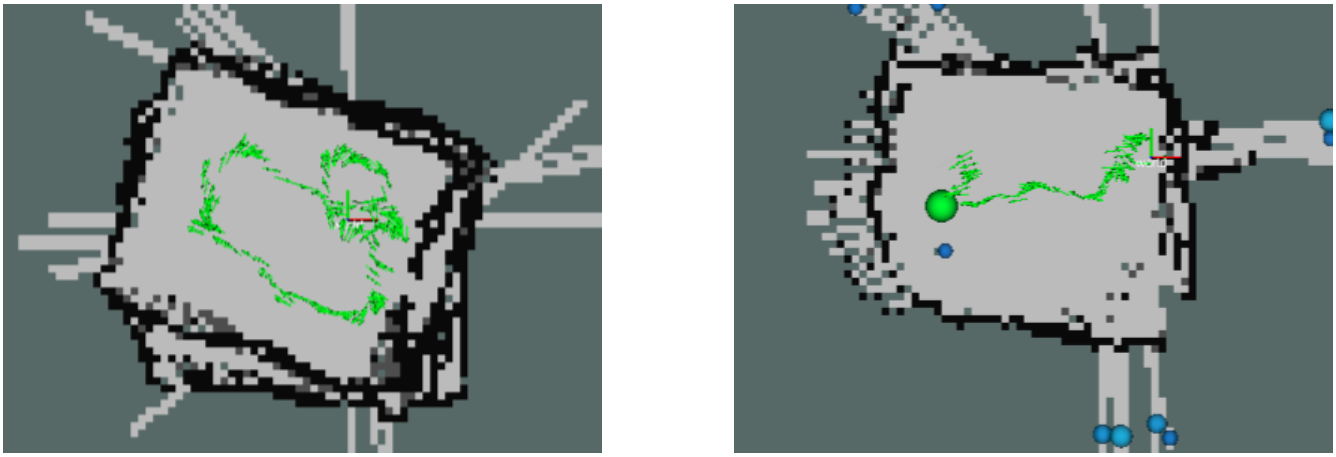
Table 4: Measurement statistics of the Multi-Ranger sensors.

Table 4 presents the statistical properties of the recorded measurements. The measurements show that the sensors generally provide stable distance estimates. Most sensors exhibited dropout rates below 3%, while the rear sensor showed a noticeably higher dropout rate of 8.6%. This difference is likely caused by sensor orientation and variations in target geometry during the experiment.

To further quantify measurement noise, the standard deviation of the range measurements was analyzed for different yaw headings. For most orientations, the observed measurement noise remained below 5 cm. Larger deviations were observed at specific angles where the sensors viewed object edges or oblique surfaces. This testing showcases the limited sensing range and occasional measurement dropouts introduce uncertainty that is not present in simulation. These effects are expected to reduce mapping quality and exploration performance during real world experiments and therefore represent an important component of the simulation-to-reality gap discussed in Section 3.6.

4.2.2 SLAM in Reality

To evaluate the performance of the SLAM system in a real-world environment, the Crazyflie was placed in a small square room without obstacles and instructed to autonomously explore the environment while constructing an occupancy map. Two navigation strategies were evaluated: the baseline wall-following behaviour and the frontier exploration approach. The resulting occupancy maps, visualized in RViz2, are shown in Figure 13.



(a) Baseline

(b) Frontier Exploration

Figure 13: Occupancy maps generated during real-world SLAM experiments and visualized in RViz2. The Crazyflie was placed in a square room without obstacles and tasked with autonomously exploring the environment while constructing an occupancy grid. Figure 13a shows the result obtained with the baseline wall-following behaviour, while Figure 13b shows the frontier exploration approach. The green trajectory represents the estimated path of the drone. In the frontier exploration example, the large green marker indicates the currently selected frontier goal, while the blue markers indicate candidate frontier locations identified by the exploration algorithm.

Figure 13a presents the map generated using the baseline wall-following behaviour. The overall room geometry was reconstructed successfully and the resulting occupancy grid closely resembles the actual room layout. However, because the wall-following algorithm has no explicit exploration completion criterion, the drone continued flying after the environment had already been mapped. As the flight duration increased, odometry drift accumulated and caused the estimated trajectory to gradually rotate with respect to the previously mapped area. This effect resulted in a second, slightly rotated representation of the room becoming visible in the occupancy grid. Remnants of the original map can still be observed below the rotated structure, indicating that the SLAM system was unable to fully compensate for the accumulated localization error.

Figure 13b shows the occupancy map generated using frontier exploration. In contrast to the wall-following approach, frontier exploration terminates when no unexplored frontiers remain. Consequently, the drone landed before substantial long term drift could accumulate, preventing the rotational distortion observed in Figure 13a. The resulting map is therefore better aligned with the true room geometry. Nevertheless, a noticeable translational drift is visible along the horizontal axis. The left and right walls appear as multiple parallel structures rather than single continuous walls. This effect suggests inconsistencies between successive range measurements and the estimated drone pose. Since the Multi-Ranger sensors provide only sparse distance measurements and the Flowdeck based odometry accumulates error over time, small localization inaccuracies can cause repeated observations of the same wall to be inserted into slightly different positions within the occupancy grid.

4.2.3 Object Detection in Reality

The final experiment aimed to evaluate the object detection pipeline in a real world environment. Although the simulation experiments demonstrated that newly introduced objects could be detected by comparing two occupancy maps, several practical challenges prevented a reliable evaluation on the physical platform.

The first challenge is the accumulated localization error of the SLAM system. As shown in Section 4.2.2, the generated occupancy maps exhibit small translational and rotational drift. Figure 13b shows that even in a relatively small room the occupancy grid is not perfectly aligned with the physical environment. While these errors remain acceptable for navigation, they become problematic for map differencing. The object detection algorithm assumes that the occupancy maps generated during Phase 1 and Phase 2 are aligned within the same coordinate frame. Any translation or rotation between the maps causes static parts of the environment to appear as differences, even when no object has been added.

A second challenge is the sensitivity of the Multi-Ranger sensors to environmental conditions. During testing it was observed that reflective surfaces and varying lighting conditions occasionally produced inconsistent range measurements. Although these effects are generally small, they introduce additional uncertainty into the occupancy grid and increase the number of false differences between two mapping sessions.

Finally, the experimental procedure itself introduces additional alignment errors. The object detection pipeline requires two separate mapping phases: an initial flight to generate a reference map and a second flight after placing an object in the environment. During testing it was observed that the take-off procedure of the second flight did not always register correctly in the occupancy map. As a result, the generated map could be slightly translated or rotated with respect to the reference map. Since the current object detection algorithm performs a direct cell by cell comparison of both occupancy grids, even small alignment errors can result in large numbers of false detections.

A possible improvement would be to introduce a map registration step before comparing the occupancy grids. Such a method could compensate for small translational and rotational offsets between the two maps, thereby increasing the robustness of the object detection algorithm. However, the implementation and evaluation of such techniques fall outside the scope of this research. As a result of these limitations, a quantitative evaluation of object detection in a real world environment was not considered sufficiently reliable within the scope of this study. Nevertheless, the simulation experiments demonstrate that the underlying concept is feasible when accurate map alignment can be maintained.

5 Discussion

This research investigated whether a Crazyflie 2.1+ equipped only with a Flowdeck V2 and Multi-Ranger deck can perform autonomous indoor mapping and object detection. The results demonstrate that meaningful occupancy grid maps can be generated using only sparse range measurements and optical flow based odometry, but they also highlight several limitations inherent to micro-UAV platforms and low cost sensing.

The simulation experiments showed clear differences between the evaluated exploration strategies. Frontier Exploration consistently produced the highest mapping performance across most evaluation metrics, achieving the best average F1 score, SSIM score and Hausdorff distance in the large

benchmark environment. The main reason for this performance is that frontier based planners explicitly direct the robot towards unexplored regions of the map. As a result, exploration effort is spent on reducing uncertainty rather than repeatedly traversing already known areas. This behaviour is visible in the coverage curves, where Frontier Exploration continues to discover new regions throughout the entire experiment while the wall-following approaches quickly plateau.

The Next Best View (NBV) planner achieved performance comparable to Frontier Exploration in several experiments but exhibited significantly larger variance between runs. The hyperparameter finetuning revealed several failure cases where the planner terminated prematurely, resulting in path lengths of only a few metres. These results suggest that NBV is highly sensitive to viewpoint selection and information gain estimation. Small changes in the occupancy map can substantially alter the ranking of candidate viewpoints, causing the planner to either continue exploring effectively or incorrectly conclude that no useful viewpoints remain. While NBV offers a theoretically attractive framework for information driven exploration, the results indicate that its implementation is less robust than Frontier Exploration in environments where the map quality is still evolving.

Both wall following approaches performed noticeably worse than the exploration based methods. Nevertheless, the custom Wall Follower consistently outperformed the Bitcraze baseline implementation. The improved performance can largely be attributed to its more stable wall distance control and more consistent environment coverage. However, both wall following methods suffer from the same fundamental limitation: they only react to local geometry. Once a wall has been discovered, the drone tends to repeatedly traverse the same areas instead of actively searching for unexplored space. This behaviour is reflected in the high redundancy ratio of the Wall Follower and the rapid saturation of its coverage curves.

An interesting observation is that SSIM and F1 scores do not always rank behaviours identically. In several parameter configurations, higher flight speeds resulted in improved SSIM values while simultaneously reducing F1 scores. This discrepancy is expected because the two metrics evaluate different aspects of map quality. F1 measures occupancy classification accuracy on a cell by cell basis and is therefore sensitive to small localization errors and incorrect obstacle placement. SSIM, in contrast, evaluates structural similarity and is more tolerant of small shifts as long as the overall geometry of the environment remains consistent. Consequently, a map can preserve the global structure of a room while still containing occupancy errors that reduce its F1 score.

The real world experiments revealed additional challenges that are not fully captured in simulation. The most significant issue was drift accumulation. Although the occupancy grid mapping approach is capable of constructing a coherent map, localization errors gradually accumulate because the system relies primarily on optical flow odometry and sparse range measurements. This effect is visible in the real world maps, where walls become duplicated or slightly rotated after prolonged flight. Unlike the simulation environment, no external positioning system was available to correct these errors. The results therefore illustrate one of the main challenges of low cost indoor SLAM: mapping performance is ultimately constrained by localization accuracy.

The sensor characterization experiments further highlighted the limitations of the Multi-Ranger deck as the primary sensing modality of the system. Although the sensors achieved a practical measurement range of approximately 2.5 m and generally exhibited low noise levels under controlled conditions, occasional dropouts and outlier measurements were observed. These errors are particularly problematic because each sensor provides only a single range measurement in a narrow field of view. Unlike a LiDAR scanner, which acquires hundreds or thousands of measurements per scan and can tolerate individual errors, a missed Multi-Ranger measurement directly results in

missing environmental information. This mean that the mapping system relies heavily on repeated observations from different viewpoints to compensate for individual sensor failures.

The experiments also demonstrated that sensor performance is strongly influenced by environmental conditions. While the simulator provides idealized measurements, the real world introduces a wide range of factors that affect the quality of the returned distance data. Ambient infrared light, especially direct sunlight, can interfere with the time of flight measurements and reduce the effective sensing range. Surface properties further influence measurement quality. Highly reflective or glossy materials can generate multipath reflections, causing unstable or incorrect distance estimates, whereas dark or absorptive surfaces may produce weak returns that lead to intermittent measurements or complete dropouts. In addition, the measured distance can vary depending on the angle at which the sensor observes a surface.

These effects became apparent during the real world experiments, where the occupancy maps contained noticeably more noise and inconsistencies than their simulated counterparts. Small measurement errors that would be insignificant for a dense sensor such as a LiDAR accumulate more rapidly when only four horizontal range measurements are available. As a result, environmental conditions that would normally have a minor impact on mapping performance can significantly influence the quality of the generated occupancy grid. This sensitivity to lighting conditions, surface materials, and viewing geometry represents one of the principal challenges of performing SLAM with extremely lightweight sensing hardware.

The object detection proof of concept demonstrated that map differencing can successfully identify newly introduced objects in simulation. By comparing occupancy grids generated before and after an object was placed, the system was able to isolate clusters of newly occupied cells while filtering out mapping noise. However, the real world experiments revealed that this approach is highly dependent on map alignment. Even small rotational or translational drift between two mapping sessions can produce large apparent differences in the occupancy grid, overwhelming the actual object signal. Therefore, reliable real world object detection would likely require either significantly improved localization accuracy or an explicit map registration step before performing the comparison.

Several limitations of this research should be acknowledged. First, the simulation environment does not model sensor noise, lighting effects or aerodynamic disturbances, meaning that real-world performance is likely to be worse than simulated performance. Second, all experiments were conducted in relatively small indoor environments with static obstacles. Larger and more complex environments would likely amplify localization drift and exploration challenges. Third, object detection was only qualitatively evaluated and therefore does not provide quantitative detection accuracy metrics. Finally, the mapping approach relies on a 2D occupancy grid and therefore cannot represent multi-level structures or fully capture the three-dimensional nature of indoor environments.

6 Conclusion

This research investigated whether a Crazyflie 2.1 nano-drone equipped only with a Flowdeck V2 and Multi-Ranger deck can autonomously perform indoor mapping and object detection. The work focused on the challenges imposed by sparse sensing, limited onboard computation, and the constraints of a lightweight aerial platform.

A complete mapping and exploration framework was developed and evaluated in both simulation and real world environments. The proposed system combines a custom occupancy grid mapping approach with four autonomous exploration strategies: a baseline wall-following controller based on the Bitcraze tutorial, an improved wall follower, frontier exploration, and next best view planning. The generated maps were evaluated using F1 score, SSIM, Hausdorff distance, coverage, and path length metrics.

The simulation results demonstrate that reliable indoor mapping is feasible using only the sparse range measurements provided by the Multi-Ranger deck. Among the evaluated exploration strategies, frontier exploration consistently achieved the highest mapping performance. It produced the most accurate occupancy grids, achieved the highest coverage, and exhibited the lowest geometric mapping error. Next best view planning achieved comparable performance but showed significantly greater variability between runs due to its dependence on viewpoint selection and information-gain estimation. The wall-following approaches were able to produce usable maps but frequently became trapped in local exploration patterns, limiting overall coverage and reducing mapping quality.

The object detection experiments further demonstrated that occupancy grid differencing can be used to identify newly introduced objects within a previously mapped environment. In simulation, the proposed method successfully detected an inserted object and distinguished it from mapping noise through cluster based filtering. This result shows that environment change detection can be performed using only the sparse sensing capabilities of a nano-drone.

The real world experiments highlighted the challenges of transferring simulation results to physical platforms. Although the Multi-Ranger sensors were capable of measuring distances up to approximately 2.5 m and generally produced stable measurements, occasional dropouts and environmental interference affected mapping performance. Reflective surfaces, varying lighting conditions, and accumulated odometry drift introduced errors that were not present in simulation. Despite these challenges, the system was still capable of producing recognizable maps of indoor environments and demonstrated that frontier-based exploration remains effective under real-world conditions.

The results therefore indicate that a Crazyflie 2.1 equipped only with a Flowdeck V2 and Multi-Ranger deck can perform basic autonomous indoor mapping and exploration without external positioning systems. However, the accuracy and robustness of the resulting maps remain strongly dependent on sensor quality, environmental conditions, and accumulated localization drift. While the proposed framework demonstrates the feasibility of low cost autonomous mapping with nano-drones, further improvements are required before such systems can reliably operate in complex real world environments.

7 Future Work

Several opportunities exist to extend the work presented in this thesis. The most immediate improvement would be the integration of a full SLAM framework with loop-closure detection and pose-graph optimization. The current implementation relies primarily on dead reckoning from the Flowdeck and occupancy grid updates from the Multi-Ranger sensors. Although this approach is computationally lightweight, accumulated drift remains one of the dominant sources of mapping error. Introducing loop closures could significantly improve long-term map consistency and reduce the rotational drift observed during real-world experiments.

Another important direction is the incorporation of additional sensors. While the Multi-Ranger deck provides a lightweight solution for obstacle detection, it only delivers five one-dimensional range measurements. Integrating lightweight vision sensors, event cameras, or compact LiDAR systems could substantially improve environmental perception while remaining within the payload constraints of small aerial platforms. Sensor fusion techniques combining range, visual, and inertial measurements may further improve localization robustness.

The current work is limited to two-dimensional occupancy grid mapping. Extending the system to three-dimensional mapping would allow the drone to operate in more realistic indoor environments containing furniture, shelves, staircases, and multiple floors. Three-dimensional occupancy grids or voxel based mapping techniques could be explored to provide richer environmental representations. Such an extension would also enable more advanced object search tasks where target objects may be located at different heights.

Multi-drone exploration represents another promising research direction. Several studies have demonstrated that teams of Crazyflies can cooperate to explore environments more efficiently than a single drone. A shared mapping framework could distribute exploration tasks among multiple agents, reducing exploration time while improving map completeness and robustness. The lightweight nature of the Crazyflie platform makes it particularly suitable for swarm based applications.

The object detection system could also be expanded beyond simple occupancy grid differencing. Machine learning based object recognition or semantic mapping approaches would allow the drone not only to detect environmental changes but also to classify and identify specific objects. This would transform the system from a mapping platform into a more general autonomous search platform.

Finally, future work could investigate competitive exploration scenarios in which the objective is not only to construct an accurate map but also to locate a target object as quickly as possible. Such experiments would provide a realistic benchmark for comparing exploration strategies and could lead to the development of planners that explicitly balance mapping quality against search efficiency. This type of task is particularly relevant for applications such as search and rescue, where rapid target localization is often more important than generating a complete map.

References

[1] Bitcraze AB. CrazyFlie 2.1+. URL: https://store.bitcraze.io/products/crazyflie-2-1-plus.

[2] Bitcraze AB. Flow deck v2. URL: https://www.bitcraze.io/products/flow-deck-v2/.

[3] Bitcraze AB. Multi-ranger deck. URL: https://www.bitcraze.io/products/multi-ranger-deck/.

[4] Bitcraze AB. Positioning Systems Overview. URL: https://www.bitcraze.io/documentation/system/positioning/.

[5] Bitcraze AB. Crazyflie firmware documentation, 2025. URL: https://www.bitcraze.io/documentation/repository/crazyflie-firmware/master/.

[6] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *Robotics Automation Magazine, IEEE*, 13:108 – 117, 10 2006. doi:10.1109/MRA.2006.1678144.

[7] Stephen Balakirsky, Stefano Carpin, George Dimitoglou, and Benjamin Balaguer. *From Simulation to Real Robots with Predictable Results: Methods and Examples*. Performance Evaluation and Benchmarking of Intelligent Systems, Springer, Norwell, MA, 2009-12-01 2009. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=902330.

[8] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, Jose Neira, Ian Reid, and John J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, December 2016. URL: http://dx.doi.org/10.1109/TRO.2016.2624754, doi:10.1109/tro.2016.2624754.

[9] Naomi Chukwurah, Abiodun Sunday Adebayo, and Olanrewaju Oluwaseun Ajayi. Sim-to-real transfer in robotics: Addressing the gap between simulation and real-world performance. *Journal of Frontiers in Multidisciplinary Research*, 5(1):33–39, 2024. doi:10.54660/.IJFMR.2024.5.1.33-39.

[10] C. Connolly. The determination of next best views. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 432–435, 1985. doi:10.1109/ROBOT.1985.1087372.

[11] Mario Coppola, Kimberly N. McGuire, Christophe De Wagter, and Guido C. H. E. de Croon. A survey on swarming with micro air vehicles: Fundamental challenges and constraints. *Frontiers in Robotics and AI*, Volume 7 - 2020, 2020. URL: https://www.frontiersin.org/journals/robotics-and-ai/articles/10.3389/frobt.2020.00018, doi:10.3389/frobt.2020.00018.

[12] Craig Coulter. Implementation of the pure pursuit path tracking algorithm. 1992. URL: https://api.semanticscholar.org/CorpusID:62550799.

[13] Rareș Crăciun and Adrian Burlacu. A scoping review of the crazyflie ecosystem: An evaluation of an open-source platform for nano-aerial robotics research. *Drones*, 10(4), 2026. URL: https://www.mdpi.com/2504-446X/10/4/261, doi:10.3390/drones10040261.

[14] Cyberbotics. Webots documentation, 2025. URL: https://www.cyberbotics.com/.

[15] Hugh Durrant-whyte and Tim Bailey. Simultaneous localization and mapping: Part i. *Robotics Automation Magazine, IEEE*, 13:99 – 110, 07 2006. doi:10.1109/MRA.2006.1638022.

[16] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. doi:10.1109/2.30720.

[17] Giorgio Grisetti, Rainer Kümmerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Transactions on Intelligent Transportation Systems Magazine*, 2:31–43, 12 2010. doi:10.1109/MITS.2010.939925.

[18] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi:10.1109/TSSC.1968.300136.

[19] Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1):185–203, 1981. URL: https://www.sciencedirect.com/science/article/pii/0004370281900242, doi:10.1016/0004-3702(81)90024-2.

[20] Ayaz Ahmed Hoshu, Liuping Wang, Shahzeb Ansari, Abdul Sattar, and Manzoor Hyder Alias Bilal. System identification of heterogeneous multirotor unmanned aerial vehicle. *Drones*, 6(10), 2022. URL: https://www.mdpi.com/2504-446X/6/10/309, doi:10.3390/drones6100309.

[21] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9):850–863, 1993. doi:10.1109/34.232073.

[22] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In Federico Morán, Alvaro Moreno, Juan Julián Merelo, and Pablo Chacón, editors, *Advances in Artificial Life*, pages 704–720, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg. doi:10.1007/3-540-59496-5_337.

[23] Michael Jones, Soufiene Djahel, and Kristopher Welsh. Path-planning for unmanned aerial vehicles with environment complexity considerations: A survey. *ACM Comput. Surv.*, 55(11), February 2023. doi:10.1145/3570723.

[24] Samer Karam, Francesco Nex, Bhanu Teja Chidura, and Norman Kerle. Microdrone-based indoor mapping with graph slam. *Drones*, 6(11), 2022. URL: https://www.mdpi.com/2504-446X/6/11/352, doi:10.3390/drones6110352.

[25] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011. arXiv:https://doi.org/10.1177/0278364911406761, doi:10.1177/0278364911406761.

[26] Matan Keidar and Gal A. Kaminka. Efficient frontier detection for robot exploration. *The International Journal of Robotics Research*, 33(2):215–236, 2014. arXiv:https://doi.org/10.1177/0278364913494911, doi:10.1177/0278364913494911.

[27] Pierre-Yves Lajoie and Giovanni Beltrame. Swarm-slam: Sparse decentralized collaborative simultaneous localization and mapping framework for multi-robot systems. *IEEE Robotics and Automation Letters*, 9(1):475–482, January 2024. URL: http://dx.doi.org/10.1109/LRA.2023.3333742, doi:10.1109/lra.2023.3333742.

[28] Peng Li, Cai yun Yang, Rui Wang, and Shuo Wang. A high-efficiency, information-based exploration path planning method for active simultaneous localization and mapping. *International Journal of Advanced Robotic Systems*, 17(1):1729881420903207, 2020. arXiv:https://doi.org/10.1177/1729881420903207, doi:10.1177/1729881420903207.

[29] K. N. McGuire, C. De Wagter, K. Tuyls, H. J. Kappen, and G. C. H. E. de Croon. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robotics*, 4(35):eaaw9710, 2019. URL: https://www.science.org/doi/abs/10.1126/scirobotics.aaw9710, arXiv:https://www.science.org/doi/pdf/10.1126/scirobotics.aaw9710, doi:10.1126/scirobotics.aaw9710.

[30] Kimberly McGuire. Explaining kalman filters with the crazyflie. URL: https://www.bitcraze.io/2024/01/explaining-kalman-filters-with-the-crazyflie/.

[31] Kimberly McGuire. Mapping. URL: https://www.bitcraze.io/tag/mapping/.

[32] Olivier Michel. Cyberbotics ltd. webots™: Professional mobile robot simulation. *International Journal of Advanced Robotic Systems*, 1(1):5, 2004. arXiv:https://doi.org/10.5772/5618, doi:10.5772/5618.

[33] Raul Mur-Artal, J. M. M. Montiel, and Juan D. Tardos. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, October 2015. URL: http://dx.doi.org/10.1109/TRO.2015.2463671, doi:10.1109/tro.2015.2463671.

[34] Hanna Müller, Nicky Zimmerman, Tommaso Polonelli, Michele Magno, Jens Behley, Cyrill Stachniss, and Luca Benini. Fully on-board low-power localization with multizone time-of-flight sensors on nano-uavs, 2022. URL: https://arxiv.org/abs/2212.00710, arXiv:2212.00710.

[35] Sri Narayan, A Kisor, Thomas Valdez, and H Manohara. Power sources for micro-autonomous vehicles-challenges and prospects. *Proceedings of SPIE - The International Society for Optical Engineering*, 05 2009. doi:10.1117/12.818779.

[36] Vlad Niculescu, Tommaso Polonelli, Michele Magno, and Luca Benini. Nanoslam: Enabling fully onboard slam for tiny robots. *IEEE Internet of Things Journal*, 11(8):13584–13607, April 2024. URL: http://dx.doi.org/10.1109/JIOT.2023.3339254, doi:10.1109/jiot.2023.3339254.

[37] Vlad Niculescu, Tommaso Polonelli, Michele Magno, and Luca Benini. Ultra-lightweight collaborative mapping for robot swarms, 2024. URL: https://arxiv.org/abs/2407.03136, arXiv:2407.03136.

[38] Daniele Palossi, Francesco Conti, and Luca Benini. An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-uavs. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, page 604–611. IEEE, May 2019. URL: http://dx.doi.org/10.1109/DCOSS.2019.00111, doi:10.1109/dcoss.2019.00111.

[39] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3803–3810, 2018. doi:10.1109/ICRA.2018.8460528.

[40] James A. Preiss, Wolfgang Honig, Gaurav S. Sukhatme, and Nora Ayanian. Crazyswarm: A large nano-quadcopter swarm. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3299–3304, 2017. doi:10.1109/ICRA.2017.7989376.

[41] Open Robotics. Different ros 2 middleware vendors, 2025. URL: https://docs.ros.org/en/rolling/Concepts/Intermediate/About-Different-Middleware-Vendors.html.

[42] Yutaka Sasaki. The truth of the f-measure. *Teach Tutor Mater*, 01 2007.

[43] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles, 2017. URL: https://arxiv.org/abs/1705.05065, arXiv:1705.05065.

[44] Tarek Sheltami, Gamil Ahmed, Mustafa Ghaleb, and Ashraf Mahmoud. Uav path planning and trajectory optimization: A comprehensive survey. *Arabian Journal for Science and Engineering*, 51, 12 2025. doi:10.1007/s13369-025-10971-8.

[45] STMicroelectronics. *VL53L1X Time-of-Flight Ranging Sensor Datasheet*, 2024. URL: https://www.st.com/resource/en/datasheet/vl53l1x.pdf.

[46] Grik Tadevosyan, Valerii Serpiva, Aleksey Fedoseev, Roohan Ahmed Khan, Demetros Aschu, Faryal Batool, Nickolay Efanov, Artem Mikhaylov, and Dzmitry Tsetserukou. Attentionswarm: Reinforcement learning with attention control barier function for crazyflie drones in dynamic environments, 2025. URL: https://arxiv.org/abs/2503.07376, arXiv:2503.07376.

[47] Dirk Thomas. Ros 2 middleware interface, 2014. URL: https://design.ros2.org/articles/ros_middleware_interface.html.

[48] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[49] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30, 2017. doi:10.1109/IROS.2017.8202133.

[50] Luca Valente, Alessandro Nadalini, Asif Hussain Chiralil Veeran, Mattia Sinigaglia, Bruno Sá, Nils Wistoff, Yvan Tortorella, Simone Benatti, Rafail Psiakis, Ari Kulmala, Baker Mohammad, Sandro Pinto, Daniele Palossi, Luca Benini, and Davide Rossi. A heterogeneous risc-v based soc for secure nano-uav navigation. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 71(5):2266–2279, May 2024. URL: http://dx.doi.org/10.1109/TCSI.2024.3359044, doi:10.1109/tcsi.2024.3359044.

[51] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi:10.1109/TIP.2003.819861.

[52] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. The ssim index for image quality assessment, 2003. URL: https://www.cns.nyu.edu/~lcv/ssim/.

[53] William Woodall. Ros on dds, 2014. URL: https://design.ros2.org/articles/ros_on_dds.html.

[54] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151, 1997. doi:10.1109/CIRA.1997.613851.

[55] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the Second International Conference on Autonomous Agents*, AGENTS '98, page 47–53, New York, NY, USA, 1998. Association for Computing Machinery. URL: https://doi-org.leidenuniv.idm.oclc.org/10.1145/280765.280773, doi:10.1145/280765.280773.

[56] Mahmud S. Zango and Jianglin Lan. Autonomous navigation at the nano-scale: Algorithms, architectures, and constraints, 2026. URL: https://arxiv.org/abs/2601.13252, arXiv:2601.13252.

[57] Weiye Zhang, Wenshuai Yu, Licong Zhuang, Xiaoyi Zhang, Zhi Zeng, and Jiasong Zhu. Go-feap: Global optimal uav planner using frontier-omission-aware exploration and altitude-stratified planning, 2023. URL: https://arxiv.org/abs/2310.15931, arXiv:2310.15931.

[58] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. *CoRR*, abs/2009.13303, 2020. URL: https://arxiv.org/abs/2009.13303, arXiv:2009.13303.

[59] Yinghao Zhao, Li Yan, Yu Chen, Hong Xie, and Bo Xu. Faep: Fast autonomous exploration planner for uav equipped with limited fov sensor, 2022. URL: https://arxiv.org/abs/2202.12507, arXiv:2202.12507.

[60] Boyu Zhou, Yichen Zhang, Xinyi Chen, and Shaojie Shen. Fuel: Fast uav exploration using incremental frontier structure and hierarchical planning, 2020. URL: https://arxiv.org/abs/2010.11561, arXiv:2010.11561.

[61] Hanzhi Zhou, Zichao Hu, Sihang Liu, and Samira Khan. Efficient 2d graph slam for sparse sensing. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6404–6411, 2022. doi:10.1109/IROS47612.2022.9981200.