Perception and Navigation for an Autonomous Quadrotor in GPS-denied Environments

S. Saeedi, A. Nagaty, C. Thibault, M. Trentini, and H. Li

Abstract

Robots need autonomous navigation to complete their tasks efficiently. This becomes more challenging in GPS-denied environments where positioning information is not available. For small flying robots and quadrotors, inherent limitations such as limited sensor payload and small system time-constant add another layer of challenge to the problem.

This paper presents a solution for autonomous navigation of a quadrotor rotorcraft by performing autonomous behaviors, such as exploration, returning home, or following waypoints. The solution relies on accurate localization, mapping, and navigation between waypoints. Multiple tests were performed in simulated and real-world environments to show the effectiveness of the proposed solution.

1 Introduction

In robotics, perceiving the environment and representing the acquired knowledge has always been a challenging problem. By increasing robot complexity and degrees of freedom, the environment perception problem becomes more complicated. Map learning is a solution for this problem which involves localization, mapping, and path planning. For a mobile robot, map learning is an essential requirement by which the robot understands its unknown environment and attempts to perform autonomous tasks. The developed map and robot trajectory represent all the knowledge that the robot has perceived using its sensors. The acquired knowledge through the map learning process can be utilized to perform various autonomous behaviors, such as exploring the environment, localizing targets, and following waypoints.

In recent years, researchers have intensively worked on the development of flying machines capable of performing complicated missions with little human supervision. These vehicles are commonly known as unmanned aerial vehicles (UAVs). UAVs have no crew onboard and are flown either remotely by a pilot at a ground control station or autonomously through a pre-planned program. Rotorcraft, vehicles with multiple rotors, have attracted more attention recently, with quadrotors being the most popular. Multiple-rotor vehicles, also known as flying robots, have outstanding maneuverability. Furthermore, they are less complex compared with other rotorcraft such as helicopters. The previously mentioned advantages have made quadrotors more suitable for civil and military applications.

With the main objective of developing efficient quadrotors, capable of accomplishing complicated tasks autonomously, a significant amount of work has been dedicated

to designing and building efficient sensing suites to perceive the environment and estimate the state of the vehicle. Knowledge of the surrounding world and variables representing the vehicle in the world are used for vehicle stabilization. Once the vehicle is stable, it can perform more difficult and complicated tasks autonomously.

Early prototypes of UAVs were dependent on inertial dead reckoning aided by global positioning system (GPS). However, GPS signals have poor performance in urban settings and inclement weather conditions. Moreover, there is no GPS reception in most indoor environments and enclosed spaces which are referred to as GPS-denied environments. To obtain position and orientation estimates in these environments, an alternative solution is to equip vehicles with inertial, optical, and ranging sensors.

This paper studies the possibility of using state-of-the-art sensing technologies to perform perception in GPS-denied environments and achieve autonomy with minimum human intervention. The rest of the work is organized as follows: Section 2 presents background information to quadrotor and autonomous navigation requirements. Section 3 introduces the proposed solution for perception and navigation of an autonomous quadrotor. Section 4 presents experimental results in simulated and real-world environments. Finally, Section 5 summarizes the work.

1.1 Contribution

The contribution of this work is an integrated autonomy solution, which includes all necessary elements for quadrotor autonomy. These elements include mission planning, map learning, and state estimation. Furthermore, map learning consists of localization, mapping, and path planning components. The selected sensor suite enables the quadrotor to operate in GPS-denied environment. The proposed solution with the sensor suite can also be used on ground robots or other types of rotorcraft in GPS-denied environments. The contributions are summarized as follows:

- At the mission level, a set of behaviors are developed and structured to suit the goals of autonomous navigation.
- Main building blocks of the autonomy solution are implemented to ensure that the mission goals are met.
- Localization and mapping are performed using mounted sensors considering 3D motion of the robot.
- To guide the robot between waypoints, a path planning algorithm is developed.
- The robot has been enabled to explore and map unknown environments using the path planning, localization, and mapping capabilities.
- The proposed work is designed in such a way that it can be utilized on ground robots as well.

2 Background and Literature Review

This section presents the background for rotorcraft perception. First, some information about small unmanned flying robots, more specifically quadrotors, is presented. Then, background to navigation which includes control, state estimation, path planning, localization, mapping, and high level mission planning is presented.

2.1 Quadrotor

A quadrotor, also called quad rotorcraft, is a rotorcraft that is lifted, controlled, and propelled by four rotors. The quadrotor can take off and land vertically. It can fly in any direction without changing its heading. This section presents background to its history, motivation, definition, advantages, and disadvantages.

With the advances in electromechanical systems and sensing technologies, recently quadrotors have become popular in unmanned systems research. Their small size and maneuverability make it possible to fly indoors and outdoors. Compared with other rotorcraft, quadrotors are less complicated and easier to control. They have many different applications, including border patrol, search and rescue [1], pipeline monitoring, wildfire monitoring, traffic monitoring, land surveys, and agile load transportation [2].

Small-sized quadrotors have significant advantages as unmanned systems. Compared with ground robots, they can move in any direction and therefore are not limited by obstacles. Compared to other rotorcraft, such as helicopters, quadrotors have less complex dynamics; however, for small-size autonomous flying robots, there exists a number of challenges. These challenges impose limitations on flight time, autonomy, and maneuverability of the rotorcraft and include the following:

- **Nonlinearity:** Quadrotor models have nonlinear kinematics and dynamics. These models are used for control, stabilization, and state estimation purposes. Most techniques of control or state estimation require linearization of the system model. Linearization is an approximation and can affect the performance of the controller.
- **Fast Dynamics:** The small time-constant of the quadrotor, which is because of fast dynamics, requires a robust and reliable controller.
- **Limited Payload:** Due to limited payload, limited number of sensors can be carried onboard. Moreover, lightweight batteries must be used which limit the flight time.
- **Vibration:** Vibration caused by high speed rotors affects the onboard sensor measurements. To minimize the vibration effects, efficient noise removal filtering and vibration damping must be used.

Fig. 1 shows a simplified sketch of the quadrotor, its coordinate systems, and the free body diagram. The frame tagged with $X-Y-Z$ shows the body frame. The frame tagged with $N-E-D$ shows the inertial frame, based on the North-East-Down standard. F_i and M_i (i=1,..4) are the aerodynamic forces and moments produced by the i^{th} rotor, respectively. L is the moment arm, the distance from the center of the

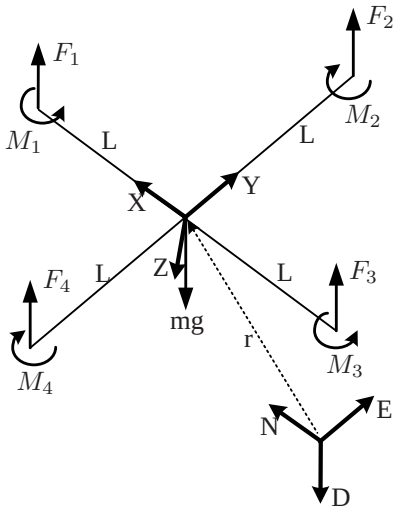


Figure 1: The coordinate systems and the free body diagram of the quadrotor with acting forces. The frame shown by $X - Y - Z$ is the body frame. The frame shown by $N - E - D$ is the inertial frame, following the North-East-Down standard.

quadrotor to the axis of rotation of the rotors. g is the acceleration due to gravity and m is the mass of the quadrotor. The pose of the robot in this work includes the 3D coordinates of the system and the orientation of the system. The kinematic and dynamic models of the quadrotor using these parameters are explained in [3].

2.2 Quadrotor Navigation

The main problem for an autonomous flying robot is reaching a desired location without human supervision and intervention. In the literature and the robotics community, this problem is referred to as *navigation* or *guidance*. In this section, quadrotor navigation is investigated.

To address the navigation problem, a set of tasks must be addressed. These tasks include the following:

- Mission planning,
- Map learning: simultaneous planning, localization, and mapping,
- State estimation, and
- Control.

Fig. 2 depicts these tasks and their dependency on each other. Each task in each level operates based on the information received from the next higher level. *Mission planning* determines the general behavior of the robot. For example, a mission planning algorithm may decide that the robot will explore an unknown environment, then it will

Figure 2: Navigation stack. This figure shows the required tasks to accomplish navigation. Each level relies on information received from the next higher level.

return to the start position and will deliver an exploration report such as a map. *Map learning* is the task of modeling the world, which requires mapping, localization, and planning a path between waypoints. *State estimation* is the process of fusing data from all available sources. For instance, using an optical sensor, a robot can estimate 2D pose and heading through localization and mapping. The 2D pose and the heading can be fused by odometry or inertial measurements to provide better estimates. The estimated state is used by the *Control* layer to generate proper control signals to stabilize the robot towards a desired state. In the rest of this section, each of these problems in the context of the quadrotor is briefly introduced.

2.2.1 Control

Compared to other types of rotorcraft, quadrotors are mechanically simpler and easier to control. However, controlling a quadrotor is still a challenging problem due to its system nonlinearities, cross couplings of the gyroscopic moments and underactuation [3], [4]. Additionally, controlling and coordinating a fleet of rotorcraft requires dealing with another layer of problems which are inherent to the multi-agent systems [5]. For more information on controlling a quadrotor, see [6], [7], [8], and [9].

2.2.2 State Estimation

Any control algorithm needs to have access to the real state of the system. The state of a system is a set of variables which describe enough about the system so that one can determine and analyze its future behavior. In practical applications, state variables are affected by noise and might not be observable directly. To solve these problems, state estimation techniques are used. Accurate state estimation directly affects the performance of the controller and therefore the overall performance of the system. Typically, the state of a quadrotor includes its orientation, position, and velocities.

In 2006, S. Thrun et al. [10] were among the first researchers who did real-time state estimation. They performed state estimation with a helicopter in an outdoor environment. In 2009, Grzonka et al. [11] from the University of Freiburg and Bachrach et al. [12] from Massachusetts Institute of Technology (MIT) performed state estimation for a quadrotor, independently. MIT's work won the 2009 International Aerial Robotics Competition (IARC) held by the Association for Unmanned Vehicle Systems International (AUVSI). Since then, researchers have investigated different configurations for state estimation, but one thing which is common in all solutions is the use of Kalman filtering. Kalman filtering, if tuned properly, can provide optimal results for linear systems.

2.2.3 Map Learning

Deploying an autonomous robot in a real-world scenario requires learning maps. Learning maps is different than only mapping. In mapping it is assumed that the pose of the mapper is known, but in learning maps, generally, pose is not known. To learn maps, a number of key problems should be addressed, including

- Mapping,
- Localization, and
- Path planning.

Fig. 3 depicts these key problems and their relationships. *Mapping* is the process of modeling a robot's world given sensory measurements, while *localization* calculates the position of the robot within the map. To move between two given points in the world, a *path planning* or *motion planning* algorithm is required.

There is a tight dependency between these components. In an unknown environment, mapping and localization can not be decoupled. This is because to make a map, the robot needs to know its current position and to know its current position, it needs to have a map. *Simultaneous localization and mapping (SLAM)* addresses this issue for one robot [13] or a team of robots [14], [15], [16], [17], [18], [19], [20], and [21]. In cases where the map is known in advance, *active localization* algorithms are used to improve the pose of the robot by selecting control actions that will reduce the uncertainty of the robot's pose estimate. If the pose of the robot is known, *exploration* algorithms are used to find waypoints which lead the robot to the unexplored parts of the environment. The combination of mapping, localization, and path planning is often referred to as simultaneous planning, localization, and mapping (SPLAM); integrated autonomy solutions; or autonomy packages [22]. Complete information on different methods for components of map learning in relation to mapping and localization can be found in [23], [24], and [25], and for path planning in [26] and [27]. SLAM and path planning for indoor quadrotors were first performed in 2009 by Grzonka et al. [11] from the University of Freiburg and Bachrach et al. [12] from MIT, independently. Later, other researchers continued the same trend to improve on the previous results. In 2010, Dryanovski et al. [28] at the City College of New York presented their approach for quadrotor SLAM. It was similar to the work proposed by Grzonka et al.

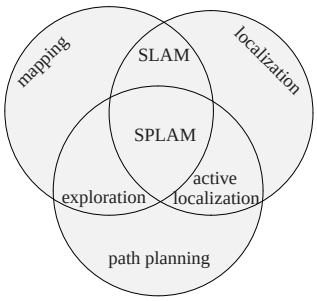


Figure 3: Tasks that need to be accomplished towards map learning [22]. An autonomous robot needs mapping, localization, and path planning to complete its tasks efficiently.

[11]. In 2011, Kohlbrecher et al. [29] from the Darmstadt University of Technology proposed their fast approach for quadrotor SLAM. The latter two solutions had no path planning involved. All these solutions are using scanning laser rangers as a key perception sensor to perform SLAM. Other researchers at the University of Pennsylvania [30] and the Swiss Federal Institute of Technology in Zurich (ETH Zurich) [31] published similar results for SLAM or path planning with the quadrotor.

2.2.4 Mission Planning

Mission planning for an autonomous flying robot is defined as a set of actions which should be taken at different times. It is part of the navigation stack and like a state machine, it tells the robot what to do. For example, a simple mission for an autonomous flying robot which is patrolling over borders between two countries would be: fly over the curve of the border at a given altitude with a given velocity. If a moving object is approaching the border, the flying robot will calculate velocity and position of the object and will report them to headquarters.

An artificial intelligence based solution for this task of navigation is to define a set of navigational behaviors. Each behavior is a process or control law that achieves and/or maintains a goal [32], [33]. As an example, *Obstacle avoidance* behavior maintains the goal of preventing collisions and *follow me* behavior achieves the goal of following the position of a person. Some behaviors are prerequisites for other behaviors. For instance, if a robot wants to perform *follow me* behavior, knowledge of the current position of the robot is required. Therefore, *localization* behavior is required for following a person.

Path planning is a another important behavior that most other behaviors such as *follow me* and *return home* rely on. For these behaviors, a path is generated between the goal point and the current position of the robot. For more information on behaviors, see [32].

3 Proposed Perception and Autonomy Solution

In this section, the problem of developing an autonomous quadrotor is defined formally. Advantages and disadvantages of previous methods are reviewed. Then, the proposed solution is explained in detail. Finally, some discussions followed by contributions of the work are presented.

3.1 System Overview

To enable a quadrotor to navigate autonomously, a set of interdependent requirements must be addressed, including control, state estimation, map learning, and mission planning. As mentioned, autonomous navigation requires stabilization of the vehicle. To stabilize the vehicle, the state of the vehicle must be known. In GPS-denied environments, state estimation is calculated using simultaneous localization and mapping algorithms. Once the state of the vehicle is known, depending on the planned mission, the vehicle can move from one point to another using a path planning algorithm. This section proposes an integrated solution for all these requirements.

To perceive the environment, sensors play a key role. For the autonomous quadrotor, a sensor suite including inertial, laser ranging, and optical sensors are used to provide exteroceptive and proprioceptive measurements. All four levels of the autonomy (Fig. 2) rely on information from these sensors, directly or indirectly. The state estimation is designed so that if GPS signals are available, they can be integrated with the solution. The sensor suite with the developed algorithms are independent of the hardware platform. In other words, the proposed autonomy solution with the sensor suite can easily be ported to other robots, such as ground robots. The only necessary change would be modifying the control level, in the navigation stack (see Fig. 2), to apply the proper control signals to the specific actuators of the robot. As will be shown in the experimental results, portability of the autonomy solution and the sensor suite have been demonstrated in real-world experiments on a ground robot and a quadrotor.

3.2 Problem Statement

Developing an autonomous quadrotor for GPS-denied environments is considered to be a challenging problem. Formally and briefly, this problem can be defined as *to design and develop a quadrotor capable of performing a set of tasks in GPS-denied environments, autonomously*. An autonomous quadrotor needs the following tasks to navigate: control, state estimation, map learning, and mission planning. Each of these tasks is required to achieve autonomy. More specifically, map learning involves path planning, localization, and mapping. The mission might be designed by a human, but the robot must be able to perform transition between different stages of the mission. The main challenge is that all these tasks must be performed in spite of the characteristics and limitations of a quadrotor, such as nonlinear dynamics, small time-constant, limited payload, vibration effect, and 3D motion. The sensor suite should enable the robot to acquire enough data about the environment for map learning, while being designed with consideration of the limitation of the quadrotor. Moreover, the developed solution must run in real time.

3.3 Problems with Existing Methods and Motivations

Most research that demonstrates outstanding results with quadrotors, such as the work by Nathan et al. [7] and Juggling Quadrotors by ETH [34], use the Vicon system. Vicon provides very accurate positioning information and avoids dealing with the challenging problem of map learning; however, this is not a problem that can be solved by Vicon in every desired indoor environment: Vicon is an expensive system and needs pre-installation. Therefore, relying on Vicon provides spatially limited autonomy. On the other hand, compared with Vicon, alternative sensing devices such as vision, laser, and inertial sensors are very inexpensive; however, these sensors require further processing which is the main challenge in real-world applications.

Some studies lack path planning and exploration as an important part of map learning. For instance, in [28] and [29], a pilot is flying the quadrotor, and the robot only performs mapping and localization.

Planning a mission is another important task that most researchers ignore. The quadrotor should be able to interact with the environment based on the priority of tasks and transition from one task to another autonomously.

To enable the quadrotor to operate autonomously in every indoor environment, map learning, as an alternative solution for GPS and Vicon, is the main focus of this work. Moreover, path planning and exploration are integral to the solution. Mission planning and transition between different behaviors and tasks are also important parts of the work. All these elements provide a complete autonomy package for a quadrotor.

3.4 Description of the Method

An overview of the proposed autonomy system is shown in Fig. 4. The proposed solution is composed of two main modules: *Mission Planner* and *Autonomy*. Each module consists of several blocks. *Mission Planner* takes care of sequencing the autonomous behaviors. The *Autonomy* module accepts behaviors from *Mission Planner* and takes actions to achieve or maintain the goal of the behavior.

The sensor suite includes an IMU, a scanning laser rangefinder with a horizontal scan, a second scanning laser rangefinder with a vertical scan, a Kinect camera, an altimeter, and a GPS. Sensors connected by a dashed line to the autonomy blocks are optional sensors. In *2D SLAM*, a 2D view-based SLAM is performed using the horizontal scanning laser rangefinder and the IMU. This block outputs 2D Cartesian coordinates and yaw angle. In the *rgbdSLAM* block, a Kinect camera is used to do 3D SLAM. Using the accelerometers of the IMU, in the *KF Fusion* block, the results of these two blocks and the measurement from the altimeter are fused by a Kalman filter. If there exists any GPS measurement, it is fused with the estimated pose to provide an estimation with bounded error. The estimated pose is then used by the *Planner* block to find waypoints and plan paths between waypoints. The calculated waypoints and position feedback are passed to the *controller* block to drive the quadrotor. *Obstacle Avoidance* block has the highest priority and directly uses the laser ranger to avoid obstacles. Although in the *Planner* block, the path planning algorithm makes plans taking into account the obstacles; however, dynamic obstacles or situations caused by disturbances or controller inaccuracies require an obstacle avoidance plan with a higher

laser scan rate. In the *voxel mapping* block, a 3D volumetric map is generated by the vertical scanning laser rangefinder using the filtered pose.

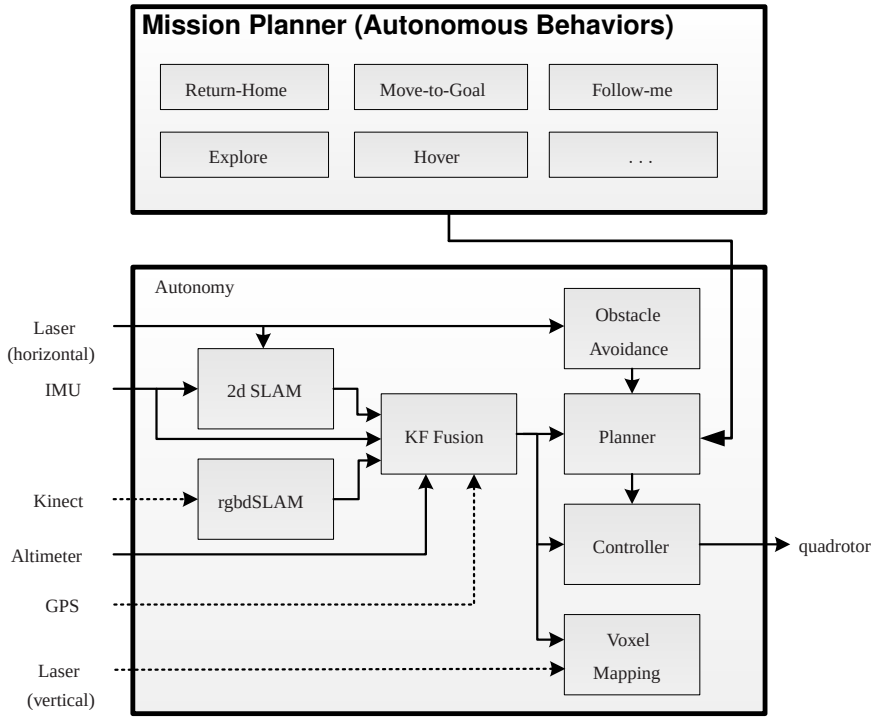


Figure 4: Proposed autonomous navigation in GPS-denied environments is composed of two main modules: *Mission Planner* and *Autonomy*. *Mission Planner* takes care of organizing and sequencing the autonomous behaviours. The *Autonomy* block accepts these behaviours and takes proper actions.

3.4.1 Autonomous Behaviors

Autonomous behaviors are a set of behaviors designed to navigate the robot efficiently. These behaviors rely on exploration, path planning, and obstacle avoidance behaviors. Behaviors such as *follow-me*, *go-home*, *move-to-goal* and *return-to-me* are based on the *path planning* behaviour between two given points and the *follow-path* behaviour. *Obstacle avoidance*, a behavior with the highest priority, is performed at two levels. First, it is performed at the map level where occupied cells are dilated and an optimal path is designed. Second, it is performed using the laser ranger and keeping a safe distance from instantaneous detected obstacles which are closer than a pre-specified threshold.

Any complicated mission can be represented as a set of behaviors. A sample mission composed of the aforementioned behaviors is presented in Fig. 5. In this mission

(mapping, localization, and path planning are not shown for clarity), first the robot explores the environment (*explore* behavior). Once exploration is complete, it moves to a given goal point (*move-to-goal* behavior). Once it reaches the destination, it returns to the start point where the mission was started (*go-home* behavior).

Figure. 5 depicts the state machine of the navigation. Two behaviors, obstacle avoidance and hold (as an emergency stop), have a higher priority than others.

The transition between behaviours is determined by their priorities and also by their state of accomplishment. For behaviors such as *go-home*, *move-to-goal*, and *follow-path*, the accomplishment criteria is the distance of the robot to the goal. If the distance is less than a threshold, the behaviour is assumed to be done and the next behavior starts. A behavior can also be called while another behavior is running, depending on the priority of the behaviors. For example, if the robot is exploring an environment, but suddenly a dynamic object appears in the field of view, then the new behavior is *avoiding obstacle*. Once there is no risk of collision, the robot retrieves its previous behaviour, and the exploration behaviour continues until there is nothing left to explore.

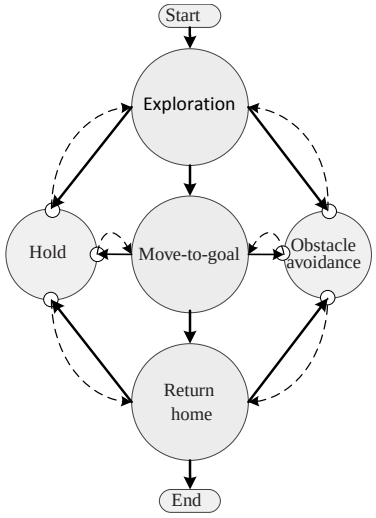


Figure 5: A sample mission composed of basic navigation behaviours. First, the robot explores an unknown environment, then it moves to a given goal. Once it arrives at the goal, it returns home. While performing these behaviours, obstacle avoidance and hold (as an emergency stop) are running at a higher priority than others. Once any of these high priority tasks ends, the previous state of the mission is resumed. This transition to the previous behavior is shown by dashed arrows.

3.4.2 2D SLAM

The quadrotor can fly at a fixed altitude; therefore, a 2D grid map is generated and used for navigation. The horizontal scanning laser ranger is used to perform 2D SLAM and generate an occupancy grid map. The method used for 2D SLAM is adopted from [29]

which is accurate and fast. In this method the scans are transformed into the stabilized local frame given the roll and pitch angles. These angles are estimated through an attitude and heading reference system (AHRS). Utilizing bilinear filtering, scans are matched against the current map at the rate of $40Hz$.

The mapping process seeks to calculate a transformation between a current scan and the map. Assume the transformation is represented by $\delta = (x, y, \psi)^T$. For the i^{th} beam of the scan, if the transformed end point of the scan is represented by $s_i = S_i(\delta)$, then the map value at the end point is shown by $M(S_i(\delta))$. The transformation should minimize the difference between the current map and the new transformed scan:

$$\delta^* = \underset{\delta}{\operatorname{argmin}} \sum_{i=1}^{n} (1 - M(S_i(\delta)))^2, \tag{1}$$

where n is the number of scan beams and δ^* is the desired outcome. This problem is formulated and solved in [29] using the gradient ascent approach. To do this, first order Taylor expansion is applied to equation (1) and the resulting Gauss-Newton equation is solved.

To speed up the mapping process and avoid local minima, a multi-resolution map representation is used [29]. The multi-resolution map representation can also be used for path planning and navigation purposes.

3.4.3 3D GraphSLAM

Features of the environment can be used to calculate the pose of the robot. To perform feature-based SLAM, at least one camera is needed. The result from the feature-based SLAM can act as an extra source of information for data fusion by Kalman filtering. In this work, feature-based SLAM is accomplished by a Kinect camera using the rgbd-SLAM algorithm.

An RGB-D camera is a sensing system that captures RGB images with depth information for each pixel. Generally, RGB-D cameras use either stereo technology [35] or infrared (IR) time-of-flight sensing [36]. Kinect [37] is an IR-based RGB-D camera developed by PrimeSense which has recently attracted the attention of researchers.

The rgbdSLAM algorithm, which uses an RGB-D camera, is a robust feature-based SLAM algorithm in which features of the environment are extracted and used for localization and mapping. Inputs of the rgbdSLAM are color (RGB) and depth images, captured simultaneously. These two images are processed to extract the pose of the camera and build up the map of the environment. Figure 6 shows the required processing blocks of the rgbdSLAM [38], [39]. The first three blocks, speeded up robust feature (SURF), random sample consensus (RANSAC), and generalized iterative closest point (GICP), provide accurate visual odometry. The last block, hierarchical optimization for pose graphs on manifolds (HOGMAN), provides global consistency of transformations and detects loop closures. In the visual odometry, first SURF features are extracted from the RGB image. Based on the correspondence of the depth values of the features, the transformation between images is extracted using RANSAC. This is an initial estimate and needs to be refined by the GICP algorithm. Here each block is explained in detail.

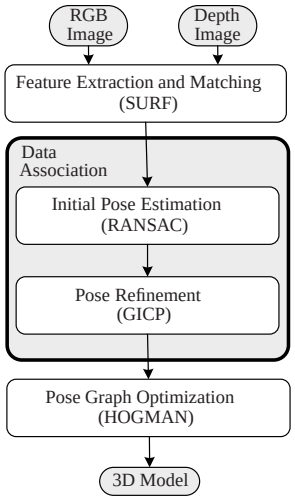


Figure 6: Flow chart of rgbdSLAM [39]. Data association in rgbdSLAM is composed of two components, RANSAC and GICP. RANSAC provides an initial estimate of the transformation matrix. The initial estimate is used in the batch data association of GICP to refine the results.

Feature Extraction and Matching. To extract features from images, the speeded up robust feature (SURF) method is used. This method is based on scale-invariant feature transform (SIFT), but it is several times faster and more robust against different image transformations. SURF relies on sums of approximated two dimensional Haar wavelet responses and integral images. The extracted features are matched against features from the previous image. Then using the depth image at locations of the corresponding features, a set of point-wise and 3D correspondences between frames is generated. This set is used for localization by further processing.

Initial Estimation. The 3D matching features are used to find the relative transformation between the frames using random sample consensus (RANSAC) method. RANSAC estimates a mathematical model from a set of observed data iteratively. The data is assumed to be noisy and contain inliers and outliers. This can produce false models in case the assumption does not hold. The extracted transformation through this method is not accurate and needs to be refined. The next block uses this estimate as an initial estimate to refine the results.

Estimation Refinement. Data association is the key element of localization and mapping. This becomes more important when SLAM relies on features only. Correct localization relies on finding correct correspondences between observations and the available map. Incorrect associations may cause pose estimates to rapidly diverge.

A significant improvement in robustness and accuracy of data association is achieved by using batch data association. In batch data association a set of observations is as-

signed at once, instead of individual assignment and association. By this approach, geometric relations of features are taken into account and association distinction will be based on their combined association likelihoods.

If the observed features are internally correlated (which is the case in most SLAM applications), then batch data association techniques can improve the results. As explained in the rgbdSLAM and shown in Fig. 6, GICP [40] is the technique used for batch data association. GICP is initialized with RANSAC which is required to provide accurate results. GICP takes two sets of features as inputs with an initial estimate of their relative transformation. Unlike most other techniques, GICP takes into account the internal geometry of the features by assigning a plane to each set and produces more accurate results. GICP is based on the locally planar structure of point clouds where plane-to-plane distance is used instead of point-to-point distance. In this method, maximum likelihood estimation is used to estimate the required transformation iteratively.

Pose Graph Optimization. The global and consistent mapping of the SLAM algorithm is based on the hierarchical optimization for pose graphs on manifolds (HOGMAN) [41]. HOGMAN is a 2D/3D optimization approach for graphSLAM. It considers the underlying space as a manifold, not a Euclidian space. The method is accurate and efficient, and it works well in online operations.

Generally, in graphSLAM, poses of the robot are represented as nodes in a graph. The edges connecting nodes are modeled with motion and observation constraints. These constraints are extracted and calculated by the SLAM front-end. Next, these constraints need to be optimized to calculate the spatial distribution of the nodes and their uncertainties [23]. This is performed by the SLAM back-end [41]. Usually the optimization by the back-end takes more time and memory than extracting the constraints; therefore, it runs at a lower frequency.

Calculating constraints by observations is a data association problem. The data association usually operates locally; thus, the search space is limited. The limit for the search space can be achieved by considering the uncertainty of the pose estimates.

In HOGMAN, the optimization is performed by Gauss-Newton with sparse Cholesky factorization. The state space is represented as a manifold and therefore avoids the parameterization singularities for estimating 3D rotations. This optimization is performed using a hierarchical method. The lower level of the hierarchy represents the original constraints, while the higher level corresponds to the abstract structural information.

The task for the back-end is to find the configuration of the nodes in such a way that the likelihood of the observations is maximized. Put mathematically, for two nodes, i and j, let x_i and x_j be two poses and z_{ij} and Ω_{ij} be their mean and information matrix of the edge that connects node i to node j. Furthermore, assume e_{ij} is the innovation of the observation (i.e, the difference of the expected observation and the sensed observation). GraphSLAM seeks to find a configuration of nodes, $x^\star$, that minimizes the negative log likelihood of all measurements:

$$x^\star = argmin_x \sum_{(i,j)\in \mathrm{C}} e_{ij}^T \Omega_{ij} e_{ij}, \tag{2}$$

where C is the set of pairs of indices that an observation constraint exists for them.

Linearizing (2) by Taylor expansion, the following relation is the result

$$\mathbf{H}\boldsymbol{\Delta}\mathbf{x}^{\star} = \mathbf{b}, \tag{3}$$

where the matrix $\mathbf{H}$ is a sparse information matrix of the system and $\mathbf{b}$ is the residual vector. This allows equation (3) to be solved efficiently by sparse Cholesky factorization. Once $\Delta x^{\star}$ is solved, it is added to the initial estimate to find the optimized poses (i.e., $x = \hat{x} + \Delta x^{\star}$).

In all solutions of graphSLAM, the main challenge is how to solve equations (2) and (3). HOGMAN uses iterative Gauss-Newton on a manifold to solve (3). In the iterative approach, each time the result is used as an initial estimate for the next iteration. The main difference distinguishing HOGMAN from other methods is that the HOGMAN linearizes on a manifold rather than in Euclidean space. Linearization in Euclidian space is not a valid assumption for SLAM, since orientation angles span a non-Euclidean space.

3.4.4 State Estimation

Results from SLAM are fused with the information from the IMU by a Kalman filter to provide state estimation. This estimate is used by the planner and controller blocks. Prior to using IMU measurements in the Kalman filtering, they are filtered by an AHRS system; therefore, IMU measurements are not present in the state vector of the system. The state of the system is defined as

$$\mathbf{x} = \begin{bmatrix} \mathbf{r^T} & \mathbf{v^T} \end{bmatrix}^{\mathbf{T}}, \tag{4}$$

where $\mathbf{r}$ is the position of the robot and $\mathbf{v}$ is the velocity of the robot.

$$\mathbf{r} = \begin{bmatrix} x & y & z \end{bmatrix}^T, \tag{5}$$

$$\mathbf{v} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T. \tag{6}$$

The Kalman filter operates in two steps: prediction and update. The state prediction is performed using the accelerometers of the IMU, $\mathbf{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^{\mathbf{T}}$, given the following system equations

$$\dot{\mathbf{r}} = \mathbf{v} \tag{7}$$

$$\dot{\mathbf{v}} = \mathbf{R}\mathbf{a} + \mathbf{g}, \tag{8}$$

where $\mathbf{g}$ is the constant gravity vector and $\mathbf{R}$ is the direction cosine matrix. Euler angles from the AHRS are used in the direction cosine matrix. When a measurement from the accelerometers is available, the prediction is performed. The covariance matrix accompanied with the state vector is a 6×6 matrix which is initially zeros. More details about the Kalman filtering can be found in [23].

In the update step, the state of the system is updated using the other sensors. The altimeter, visual and laser measurements are used to update the state. For instance, a measurement from the horizontal laser ranger is used to update x and y coordinates,

since the horizontal laser ranger provides only the 2D coordinates. An altimeter updates only the z coordinate. Although the proposed system does not rely on GPS signals, if GPS measurements are available, they can also be used to update the coordinates in the update step.

3.4.5 Planner: Exploration of Unknown Space

To explore an unknown environment, the frontier algorithm is used. This algorithm uses unknown cells located in the boundaries of the known cells as *frontiers* and moves towards them [42]. Algorithm 1 explains the frontier exploration. According to the algorithm, first frontier cells are extracted (line 2). Then the frontier cells are clustered based on connectivity-eight (line 3). Assuming that there are m clusters (line 4), for each cluster, first the center of the cluster is calculated (line 6); then the distance between the center of the cluster and the robot is calculated (line 7). (x_c^i, y_c^i) is the center of the i^{th} cluster, $i = 1..m$, and d_c^i is the distance of the cluster from the robot. The number of cells in each cluster is calculated in line 8, shown by n_c^i for the i^{th} cluster. Finally, the center of a cluster that has more frontier cells and is closer to the current position of the robot is chosen as the next waypoint. This means that a cluster is chosen which can minimize the ratio of $\frac{n_c^i}{d_c^i}$. Because of the perception range of the sensors, there is no need to wait for the robot to get to the target waypoint. Therefore, this process can continue and update the waypoints at fixed time intervals or distances. In this work, waypoints are updated every 4 meters, which eliminates the oscillatory behavior of the robot between the waypoints. Exploration terminates when there is no new waypoint, $c_{wp} = \emptyset$. This occurs, for instance, when there is no frontier cell left, or when the number of the cells in each cluster is negligible.

Algorithm 1 Frontier exploration.

Input: m: partially explored map,
(x_r, y_r): global coordinates of the robot
Output: c_{wp}: waypoint
1: $U, C \leftarrow \emptyset$
2: $U \leftarrow$ frontier cells.
3: $C \leftarrow$ clusters of U.
4: $m \leftarrow |C|$
5: **for** $i = 1 \rightarrow m$ **do**
6: $(x_c^i, y_c^i) \leftarrow$ centroid of the i^{th} cluster
7: $d_c^i = \sqrt{(x_r - x_c^i)^2 + (y_r - y_c^i)^2}$
8: $n_c^i \leftarrow$ number of cells of each cluster
9: **end for**
10: $c_{wp} \leftarrow \text{argmax}_{c^i} \frac{n_c^i}{d_c^i}$.

Once the next waypoint for the exploration is known, a path planning algorithm is used to guide the robot to the new waypoint.

3.4.6 Planner: Path Planning

To navigate between two given points, the wavefront algorithm is used. This is performed using the map generated while exploring the environment. The wavefront produces an optimal solution and no local minima are generated [26]. The details of the wavefront path planning are given in Algorithm 2. To avoid getting close to the obstacles, occupied cells are dilated such that the planner generates a path with a safe distance from obstacles (line 1). The free space is represented by a grid, marked with 0 as unvisited (line 5). The occupied and dilated cells are marked with 1 (line 6). The algorithm generates a wave from the goal cell. The goal cell is marked as a visited cell and given a value of 2 as the start point of the wavefront (line 7-8). The algorithm then iteratively finds unvisited (marked with 0) neighbors of the wave cells and assigns them values of one greater than the visited wave cell (line 10-12). The result will be a 'wavefront' radiating outwards from the goal cell as new cells are assigned increasing values. Once the wave reaches the start point, the planner travels back on the wave using gradient descent and generates the path (line 14-19).

Fig. 7 shows an example of the wavefront path planning algorithm. In Fig. 7-a, a simulated environment with obstacles is depicted. Obstacles are shown in black and the free space is shown in white. The start point is marked with a green circle and the goal point is marked with a red square. A path should be designed from the start point to the goal point. The dilated areas are shown in gray. In Fig. 7-b, obstacles are dilated to avoid getting too close to the obstacles. In Fig. 7-c, a wave is generated from the goal point to the start point. The wave is colour coded. Finally, Fig. 7-d shows the path designed using the generated wave.

3.4.7 Obstacle Avoidance

Obstacle avoidance is achieved using laser beams directly to avoid any unexpected collisions caused by disturbance or dynamic objects. Laser beams are smoothed by a sliding window to remove noisy measurements. Then beams are divided into b bins (for example, for the Hokuyo UTM-30LX laser ranger, 54 bins are used, each covering 5°). The range of a bin is defined as the average range of laser beams in that bin. A collision bin is one which identifies an obstacle within a given range. This bin has collision risks. A free bin has a range greater than a threshold. The free bin is the bin used to avoid the collision. If a bin is identified as a collision bin, then the current waypoint is changed such that the new waypoint has 180° offset from the collision bin.

Algorithm 3 summarizes the obstacle avoidance. Inputs to the algorithm are: the scan s, the current waypoint w_c, number of bins b which is used to divide the scan into b sectors, and angular span of the scan α. The output is a new waypoint w_n. If the robot is too close to an obstacle, a waypoint in the collision-free zone of the scan is produced. Otherwise, the waypoint is not changed. In line 1, the collision bin is initialized. In line 2, the free bin is initialized. w_n is calculated from this bin. In line 3, the scan is filtered by a low pass filter to remove noisy sparks. In line 4, the scan is divided into b bins, and each bin is checked to see if there is any risk of collision. This is performed by function $check\_collision(\cdot)$. If there is a risk of collision within a bin, then the identification number (ID) of the bin is returned for further processing.

Algorithm 2 Wavefront path planning

Input: m: map, q_{start}: start cell, q_{goal}: goal cell, r_{dilate}: dilation size
Output: p: path

1: dilate occupied cell for r_{dilate} cells
2: **if** q_{goal} or q_{start} are located within the dilated cells **then**
3: relocate them to the closest free cell.
4: **end if**
5: mark all free space with 0, as unvisited
6: mark all occupied, dilated and unknown cells with 1
7: $index \leftarrow 2$
8: $q_{goal} \leftarrow index$
9: **while** q_{start} not visited **do**
10: set all free cells neighboring a cell with value $index$ to $index + 1$
11: mark all cells with value $index + 1$ as visited
12: $index \leftarrow index + 1$
13: **end while**
14: $q \leftarrow q_{start}$
15: add q to p
16: **while** q_{goal} not in p **do**
17: $q \leftarrow$ neighbor of q with minimum value
18: add q to p
19: **end while**

Based on the ID of the collision bin, a free bin is selected. To do this, first, a candidate bin is chosen with 180° angular offset from the collision bin. If the candidate bin is also a collision bin, the closest collision-free bin to the candidate bin is selected as the free bin. This operation is performed by function $find\_free\_bin(\cdot)$ in line 6. In line 7, the next waypoint from the collision-free bin is calculated. The bearing of the new waypoint with respect to the current pose of the robot is equal to the orientation of the free bin. The range of the new waypoint with respect to the current position of the robot is a fixed number. In this research, it is set to be one meter. Once the robot is out of collision risk, it resumes its previous behavior.

3.4.8 3D Voxel Mapping

The 3D voxel mapping is based on a compact and flexible 3D mapping algorithm, known as Octomap [43]. Octomap uses octree mapping. An octree is a tree-like data structure where each node can have eight children. In an octree structure, a three dimensional space is subdivided into eight octants recursively. Similar to the 2D occupancy grid mapping, Octomap takes into account the probability of the occupancy of each voxel. This capability makes it a probabilistic map which can be used in dynamic environments. The occupancy probability of a voxel v given the sensor measurements from time 1 to t, $z_{1:t}$ is estimated by [43]

$$p(v|z_{1:t}) = \Big(1 + \frac{1 - p(v|z_t)}{p(v|z_t)} \frac{1 - p(v|z_{1:t-1})}{p(v|z_{1:t-1})} \frac{1 - p(v)}{p(v)}\Big)^{-1} \tag{9}$$

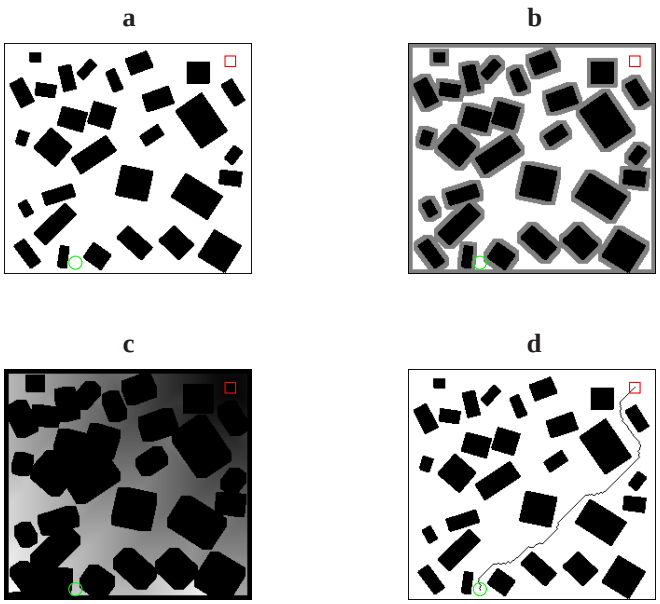


Figure 7: An example of the wavefront algorithm. **a)** A simulated environment with obstacles. Obstacles are shown in black while the free space is shown in white. A path should be designed from the start point, the green circle, to a goal point, the red square. **b)** Obstacles are dilated to avoid getting too close to the obstacles. The dilated areas are shown in gray. **c)** A wave is generated from the goal point to the start point. The wave is colour coded. **d)** Using the generated wave, a path between the start and goal points is designed, shown by a black curve.

where $p(v)$ is the initial state of the voxel. Octomap is a flexible mapping method which does not need to know the extent of the map in advance. In fact, the map expands dynamically as required. Since it models occupied, free, and unknown octants, it can be used in exploration and path planning applications. In this work, the vertical laser ranger is used to generate a 3D Octomap.

3.4.9 Controller

The controller block accepts waypoints generated by the planner block and adjusts rotor speeds. The controller block is not a part of this work. Nagaty et al. [3] developed a cascaded controller for a quadrotor which is used here. The controller is composed of inner loop and outer loop PID controllers. The inner loop controller stabilizes roll, pitch, yaw and altitude while the outer loop controller is responsible for position tracking or waypoint following. The controller is tuned properly to deal with

Algorithm 3 Obstacle avoidance

Input: s: laser scan, w_c: current waypoint, b: number of bins, α: angular coverage of a scan

Output: w_n: next waypoint

```
collision_bin_id ← ∅
free_bin_id ← ∅
s ← smooth(s)
collision_bin_id ← check_collision(s, b)
if collision_bin ≠ ∅ then
    free_bin_id ← find_free_bin(collision_bin_id, α, b)
    w_n ← waypoint(free_bin_id)
else
    w_n ← w_c
end if
return w_n
```

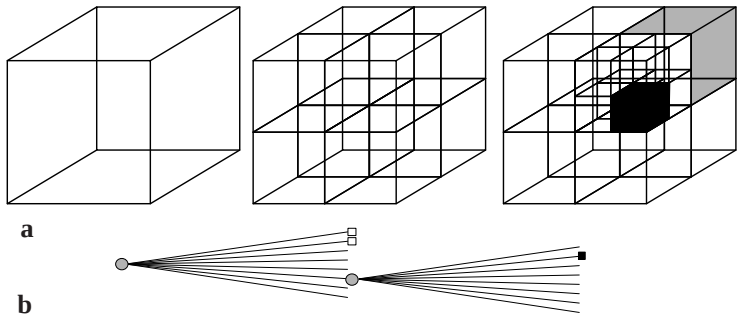


Figure 8: An example of an octree structure. **a)** Occupied and free octants depicted in black and gray. Other octants are unknown. **b)** The corresponding tree representation [43].

minor disturbances.

During flight tests, there is the risk of damaging the flying robot. It is very important to have the option of taking over the control of the robot manually at all positions and times.

3.5 Advantages and Disadvantages

In this section, the advantages and disadvantages of each component in the autonomy solution are reviewed separately.

Mission planning is composed of a set of basic behaviors which can be used to create a mission. The advantage of using basic behaviors is obvious: almost any type of mission is composed of a set of basic behaviors. The core of these behaviors is dependent on planning and following a path from one point to another point. This, in turn, relies on the knowledge of the position of the robot within the environment,

which is taken care of by SLAM. For instance, *follow me* and *go home* are two different behaviors but in fact they are very similar. The first one requires a path from the current position of the robot to the position of a person. The second one needs a path from the current position of the robot to the start point of the mission.

The wavefront path planning algorithm is a fast algorithm which avoids local minima and generates an optimal path [26]. Moreover, it is relatively easy to implement; however, there are two problems with this algorithm. First, it does not generate a smooth path. Second, it creates paths which are sometimes very close to obstacles. The first problem is solved during path following. The path following process selects waypoints from the path which automatically smoothes the path. For the second problem, obstacles are dilated to avoid the path getting too close to the obstacles. Also, this algorithm relies on the map of the environment; thus, if a goal point is given to the algorithm which is out of the scope of the map, then it fails to generate a path.

The 2D SLAM algorithm is a fast and robust method. Extensive experiments with a ground robot and the COBRA quadrotor has demonstrated its performance. Quadrotors have fast dynamics; therefore, it is important to provide pose estimates as frequently as possible. The 2D SLAM algorithm operates at $40Hz$ which is an acceptable rate to control a quadrotor.

The rgbdSLAM algorithm is a solution based on graphSLAM. Generally, graphSLAM solutions tend to be slow. While 2D SLAM operates at $40Hz$, rgbdSLAM performs at about $1Hz$. However, rgbdSLAM provides a 3D solution which recovers Cartesian coordinates and the orientation of the robot. Also, rgbdSLAM relies on features; therefore, if the environment lacks features, it may fail.

The Kinect camera is a part of the sensor suite which is used for rgbdSLAM. The main problem with the Kinect camera is its weight, about 400 grams, which is a relatively heavy load for a typical quadrotor. There is at least one laser ranger onboard which weighs about 370 grams. To fly a Kinect camera with the laser ranger, a quadrotor with larger payload-carrying capability is needed.

4 Experiment

To test the proposed perception and navigation solution, the mission shown in Fig. 5 was implemented. The experiments were implemented and tested in three different configurations, including:

- Simulation in Gazebo,
- Real-world experiment: unmanned ground vehicle, and
- Real-world experiment: quadrotor rotorcraft.

Each experiment is explained in detail. Additionally, two more experiments, demonstrating other behaviors, are also presented.

4.1 Case 1: Simulation in Gazebo

The proposed solution for perception and navigation is tested in a simulated Gazebo world. Gazebo is a three-dimensional and multi-robot simulator which simulates a team of robots, objects, and various sensors [44]. It also simulates gravity and interaction between objects. Gazebo has been integrated with ROS.

4.1.1 Simulated Flying Robot

The developed simulated quadrotor model is based on the X8 [45] rotorcraft, designed and built by Draganfly Innovations Inc. The X8 has four pairs of rotors, which provide more thrust than four rotors. The simulated robot is designed in Blender and then integrated with Gazebo. The robot is equipped with a simulated IMU, a SONAR, and two Hokuyo UTM-30LX scanning laser rangers, mounted horizontally and vertically.

4.1.2 Description

Fig. 9-a shows the Gazebo world. The size of the simulated world is 32×50 meters. Seven cylinders, each with the diameter of one meter, are placed in the world as obstacles. The distance between any two adjacent cylinders is five meters. The height of walls and obstacles is six meters. During flight, the robot flies at an altitude of three meters.

For this experiment, the mission shown in Fig 5 is executed. According to this mission, first, the robot explores the world, then it goes to a goal point, and finally, it returns to the start point. The robot is initially placed at the origin of the coordinate system (Fig. 9-a). The goal, which it attains after the exploration, is located 30 meters away, marked by a red disc.

4.1.3 Results

The complete mission took 210 seconds to perform. In Fig. 9-b, the map developed by the horizontal laser ranger is shown. Fig. 9-c shows the voxel map of the environment, developed by the vertical laser ranger. A video of the experiment including all states of the mission, i.e., exploration, move to goal, and return home, can be found in [46].

To evaluate the performance of the autonomous navigation, a few criteria are investigated. These criteria evaluate the accuracy of localization and the mission accomplishment.

4.1.4 Evaluating Pose Localization

Since the ground-truth position of the robot is available in the simulation, it is possible to compare the estimated position with the ground-truth position. Assume $\mathbf{r}$ and $\mathbf{r_{gt}}$ are positions of the robot through SLAM and ground-truth respectively. The same is assumed for orientations of the robot through $\mathbf{\Omega}$ and $\mathbf{\Omega_{gt}}$ variables. The following

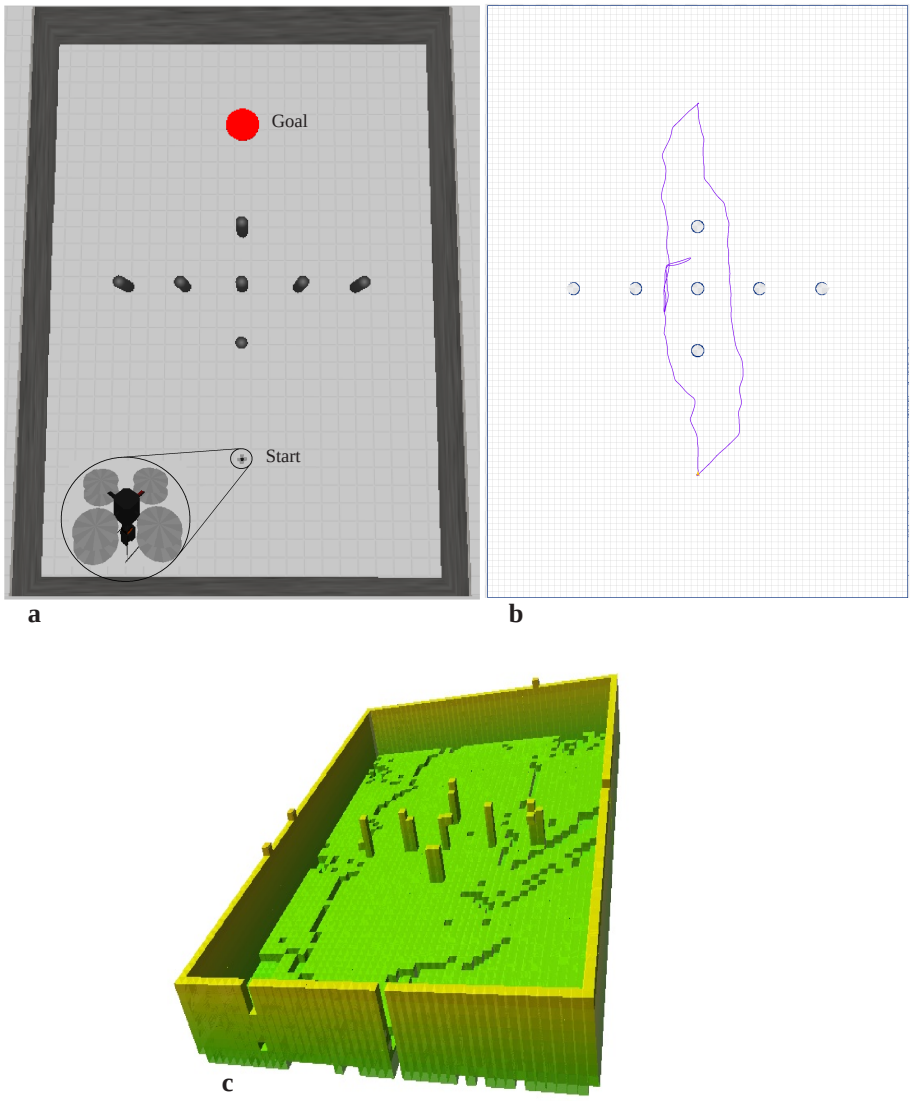


Figure 9: Simulation in Gazebo. **a)** Simulation environment. Size of the world is 32×50 metres. Magnified simulated X8 is shown inside a circle with two perpendicular laser rangers mounted beneath the rotorcraft. **b)** Trajectory and 2D map.**c)** 3D voxel map.

relations evaluate the error of the position and orientation

$$\mathbf{e_r} = ||\mathbf{r} - \mathbf{r_{gt}}||, \quad (10)$$

$$\mathbf{e_\Omega} = ||wrap(\mathbf{\Omega} - \mathbf{\Omega_{gt}})||, \quad (11)$$

where $wrap(\cdot)$ is a function which bounds the orientation so that the orientation is within the interval of $(-\pi, \pi]$. For two vectors of the same size, such as $\mathbf{p}$ and $\mathbf{q}$, the operator $||\cdot||$ shows the Euclidian distance defined as $||\mathbf{p}-\mathbf{q}|| = \sqrt{(\mathbf{p}-\mathbf{q})\cdot(\mathbf{p}-\mathbf{q})}$, where the $\cdot$ is the dot product.

4.1.5 Evaluating Mission Accomplishment

The implemented mission has two target points that the robot should attain. After the exploration phase is finished, the first target point is set as the goal point (the red disc). Once the robot has reached the goal point, the target point is switched the the second goal point that is defined to be the original starting point. The coordinates of target points are known, so by comparing the position of the target points to the ground-truth position of the robot, it is possible to calculate an error measure to evaluate targeting accuracy. If the 2D coordinates of the goal are (x_g, y_g) and the 2D ground-truth position of the robot, when it arrives at the goal, is (x_{gt}, y_{gt}), then the target achievement error is defined as

$$e_g = \sqrt{(x_g - x_{gt})^2 + (y_g - y_{gt})^2}. \quad (12)$$

Fig. 10 shows the error of the position of the robot, $\mathbf{e_r}$, during the mission. The average error is 0.0065 meters. Table 1 summarizes the performance indices. In this table, the first row represents the average position error over the course of the mission. The second row shows the average orientation error over the course of the mission. Rows 3 and 4 show the error of the position when the robot is at the goal and start points.

Table 1: Evaluating the experiment in Gazebo.

no	index	value
1	average position error, $\mathbf{e_r}$	0.0065 (m)
2	average orientation error, $\mathbf{e_\Omega}$	0.0016 (rad)
3	error of achieving goal point	0.09 (m)
4	error of achieving start point	0.02 (m)

4.2 Case 2: Unmanned Ground Robot

To make sure that the proposed solution works well in the real world, we started the experiments with a CoroBot built by CoroWare, Inc. The main objective for this test is to avoid crashing the flying robot due to unpredicted run-time problems. Furthermore, it is easier to test functionality of different components of the proposed solution on a

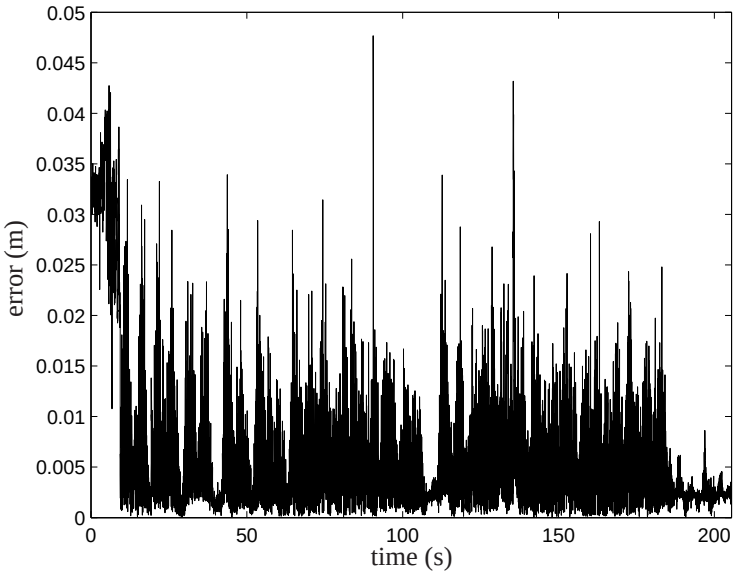


Figure 10: Robot position error, $\mathbf{e_r}$, during mission.

ground robot than on a flying robot. This experiment shows that the proposed method and sensor suite are suitable to achieve the desired goals.

4.2.1 Description

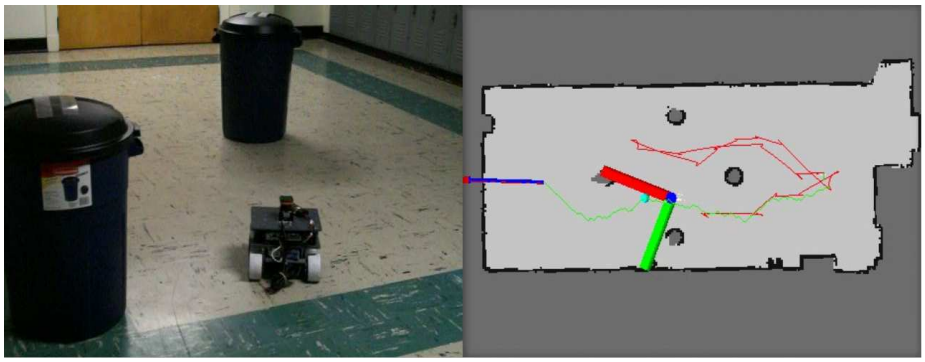
This test was performed in a room at the University of New Brunswick. The size of the room is approximately 5×12 meters and four garbage bins were placed in the room as obstacles. Because of the floor is flat, there was no rolling or pitching involved. Fig. 11-a shows the CoroBot in the test environment. The robot is equipped with one horizontal laser ranger, an IMU, and two encoders.

4.2.2 Results

Fig. 11-b shows the map of the environment with the trajectory of the robot (red curve) and the planned path (green curve). A video of the experiment can be found in [46]. All states of the mission are shown in the video.

4.3 Case 3: Quadrotor in an Indoor Environment

This real-world experiment was performed with the custom-built COBRA quadrotor. The COBRA quadrotor is a custom-built four-rotor rotorcraft designed and developed at the COBRA lab at the University of New Brunswick (UNB). It is equipped with



a **b**

Figure 11: Experiment with a CoroBot robot. **a)** The robot and the experimental environment. **b)** Trajectory and 2D map. The blue arrow shows the goal, the green curve shows the planned path and the red curve shows the trajectory of the robot. The perpendicular green and red bars show the body frame coordinates. The small cyan ball on the planned path shows the waypoint to be followed by the robot.

a Hokuyo UTM-30LX scanning laser ranger, a CH Robotics UM6 IMU, and a SensComps MINI-A PB Ultrasonic Transducer SONAR ranger. The scanning laser ranger is mounted upside down for more stability during flight. Fig. 12 shows the quadrotor. A Gumstix Overo board is used to interface sensors and send data to a ground station.

4.3.1 Description

The test environment is approximately $94.7m^2$ and two boxes are placed in the environment as obstacles. Fig. 13 shows the quadrotor in the test environment. According to the mission plan, shown in Fig. 5 and starting from the point shown by a black circle in Fig. 13-a, first, the quadrotor explores the environment and maps all unknown places such as behind boxes. Then it moves to a goal point, shown by a black square. Finally, it returns home, where it started the mission.

4.3.2 Results

The whole mission took about $320sec$, with an average speed of $0.25m/s$. Fig 13-b shows the robot at approximately 280 seconds into the test, where the robot is returning home. The green curve shows the path generated by the planner. The red curve shows the trajectory of the robot. The arrow shows the final destination of the robot which is the home. The cyan sphere shows a waypoint along the path, generated by the planner for the quadrotor. (The path, the trajectory and the waypoint are projected to the ground level for clarity of the figure.) A video of the experiment is available in [46]. The robot successfully completed the mission by exploring, mapping, and navigating through obstacles.

Fig. 14 shows the path of the robot for each stage of the mission. Starting from the

Figure 12: The COBRA quadrotor equipped with a scanning laser rangefinder, an IMU, and a SODAR.

point marked with the black circle, the blue curve shows the path of the *exploration*. The black curve shows the path of the *move to goal* behavior where the goal is depicted by a black square. The green curve shows the path of the *return home* behavior. The red curve shows the trajectory of the robot during all three stages.

Fig. 15-a shows the performance of the waypoint following. The error at each time shows the distance between the current position of the robot and the given waypoint at that time. To show the process, Fig. 15-b shows an enlarged section of the error enclosed by a red rectangle in Fig. 15-a. The vertical lines indicates the times when a new waypoint was generated by the planner and the quadrotor tried to follow the waypoint. The planner updates waypoints at the approximate rate of $2Hz$ and tries to keep the waypoints at a fixed distance of $0.9m$ from the current position of the robot, unless the robot is close to the final goal. Once the robot is closer than 0.5 meters to a given goal, the planner asks the robot to hover.

Table 2 summarizes the performance indices for this experiment. In this experiment, the ground-truth data is not available, so evaluation of the localization error is not possible. Rows 1 and 2 show the error of the position when the robot is in the goal and start points.

4.4 Case 4: Autonomous Entry and Exit

This is another simulated and real-world experiment performed with COBRA quadrotor. The purpose of this test is to demonstrate the capability of the quadrotor in tran-

a

b

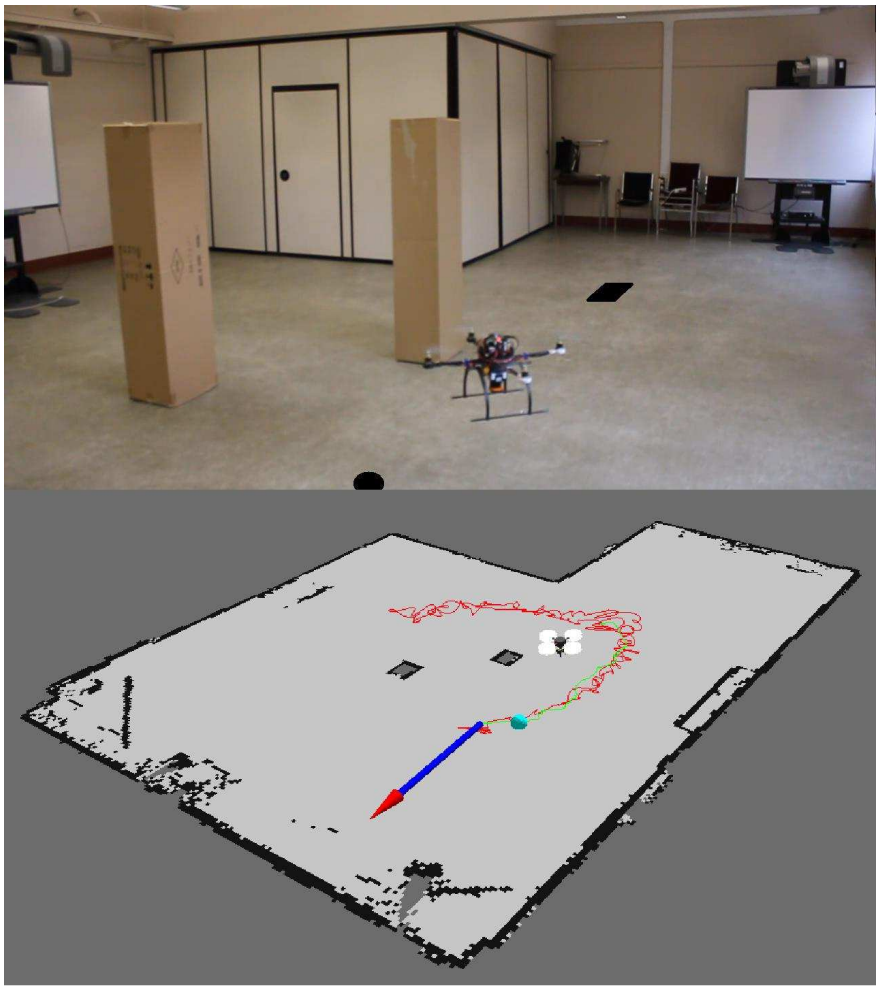


Figure 13: Experiment with COBRA quadrotor. This experiment, performs the three-stage mission depicted in Fig. 5: *exploration*, *move-to-goal*, and *return-home*. **a)** This figure shows the test environment. **b)** A snapshot of the developed map and trajectory of the robot.

Table 2: Evaluating the real-world experiment with COBRA quadrotor.

no	index	value
1	error of achieving goal point	0.16 (m)
2	error of achieving start point	0.11 (m)

sition from outdoors to indoors. Details of the experiment including flight time, the

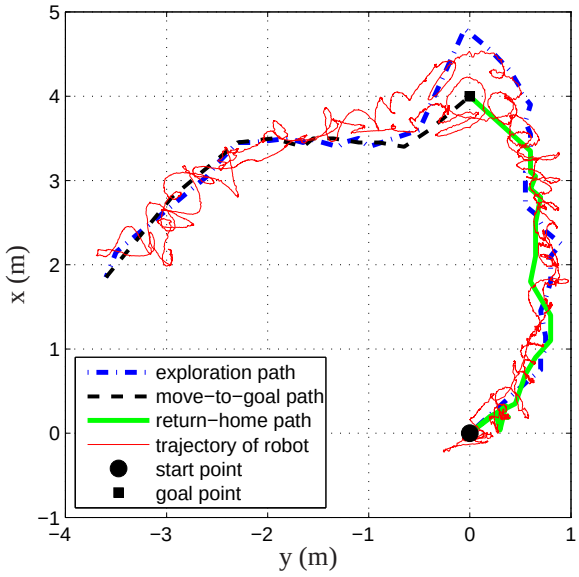


Figure 14: Paths and trajectory of the robot for each stage of the three-stage mission depicted in Fig. 5: *exploration*, *move-to-goal*, and *return-home*. The start point is shown by a circle. First the quadrotor explores the environment. The desired path for the exploration is shown in blue. Then it goes to the goal point marked by a square (*move-to-goal*). Once it reached the goal, it returned home.

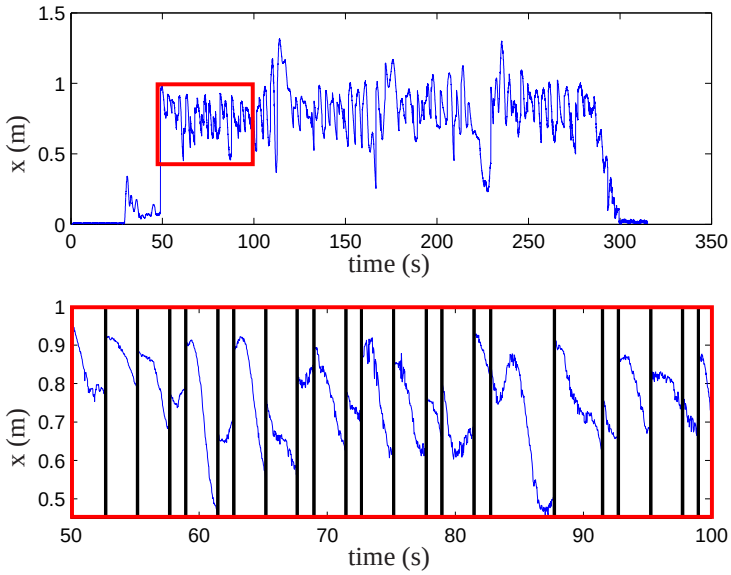


Figure 15: Performance of waypoint following. **a)** Waypoint following error for the whole trajectory shown in Fig. 14 **b)** Enlarged section of the waypoint following error enclosed in a red rectangle in Fig. 15-a. The vertical solid black lines indicate the times when a new waypoint is received. The quadrotor tries to follow the waypoint.

length of the path, and the error of achieving the goal points are not reported for the sake of brevity.

4.4.1 Description

For the simulated experiment, performed in Gazebo, a building was simulated which is unknown for the robot. It starts the mission outside of the building and the mission is to go inside, map the building, and return to the start point. Notice that in this simple scenario, exploration is not involved. The robot chooses a traversable goal point inside the building, flies to the point according to the planned path, and returns. The same scenario is implemented for the real-world experiment. These experiments demonstrate the ability of the robot to move from outdoors to indoors which is an important task in most autonomy solutions.

4.4.2 Results

Fig. 16 shows the simulated experiment. The figure represents the moment that the robot is returning to the start point. In Fig. 16-a, a snapshot of the simulated world is shown. The quadrotor is inside of the building and is not visible. The green disc is the start point. Fig. 16-b shows the developed 3D map, built by the vertical laser rangefinder. Fig. 16-c shows the camera view. Finally, Fig. 16-d demonstrates the 2D map, developed by the horizontal laser rangefinder, designed path, and trajectory of the robot.
Fig. 17 shows the same scenario in a real-world experiment. Fig. 17-a, shows the test environment. Fig. 17-b shows the developed map, trajectory of the robot, and designed path to return to the start point. A video of the experiment, including both simulated and real-world experiments, can be found in [46].

4.5 Case 5: Outdoor Unstructured Environment

This outdoor experiment demonstrates the ability of the quadrotor in mapping and navigating in unstructured environments, such as woods and cliffs. The experiment is performed at UNB Woodlot in Fredericton, New Brunswick. This experiment demonstrates the usefulness of the quadrotor and the proposed solution in different applications such as forest and vegetation control, terrain monitoring and inspection in untraversable and catastrophic environments, and saving human lives trapped in dangerous spots.

4.5.1 Results

Fig. 18-a shows the experiment site. As the figure shows, the environment is not traversable by ground robots. Fig. 18-b shows the 2D developed map and trajectory of the robot. Trees and cliffs are mapped consistently by the sensor suite.

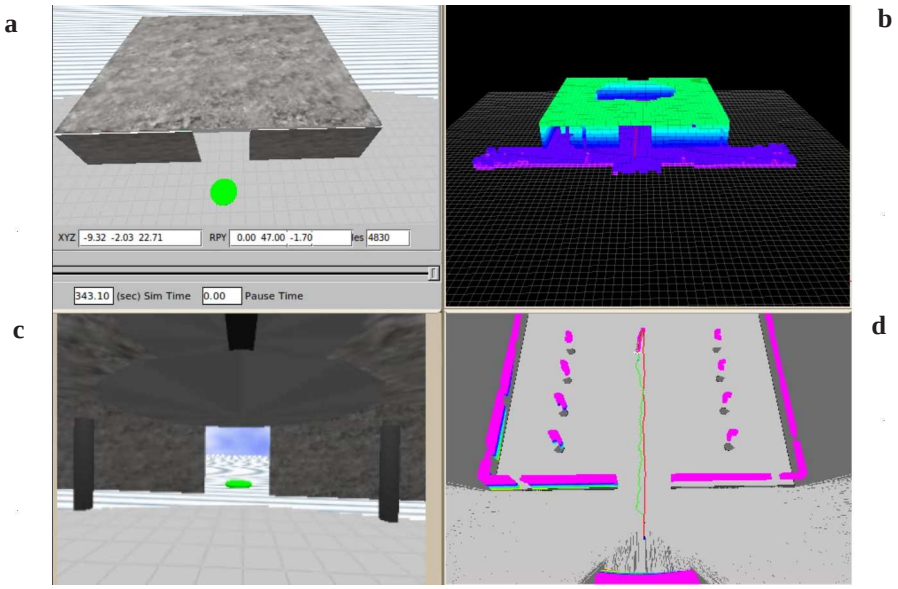


Figure 16: Autonomous entry and exit in simulation. **a)** Simulated world in Gazebo. **b)** The developed 3D map **c)** Camera view. **d)** The developed 2D map, path, and trajectory.

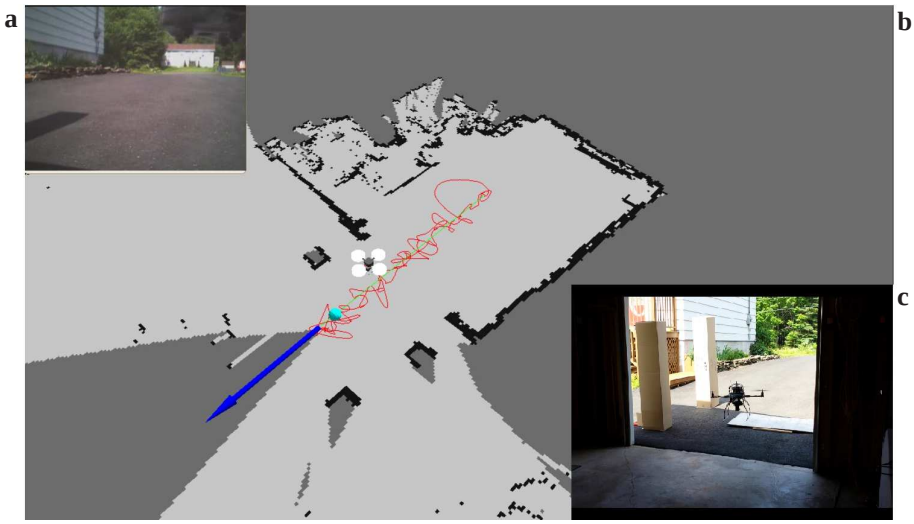


Figure 17: Autonomous entry and exit in a real-world experiment. This figure shows the experiment at the moment that the quadrotor returns home. **a)** Onboard camera view **b)** The developed 2D map, path, and trajectory. **c)** Ground view of the quadrotor, exiting through the door.

a

b

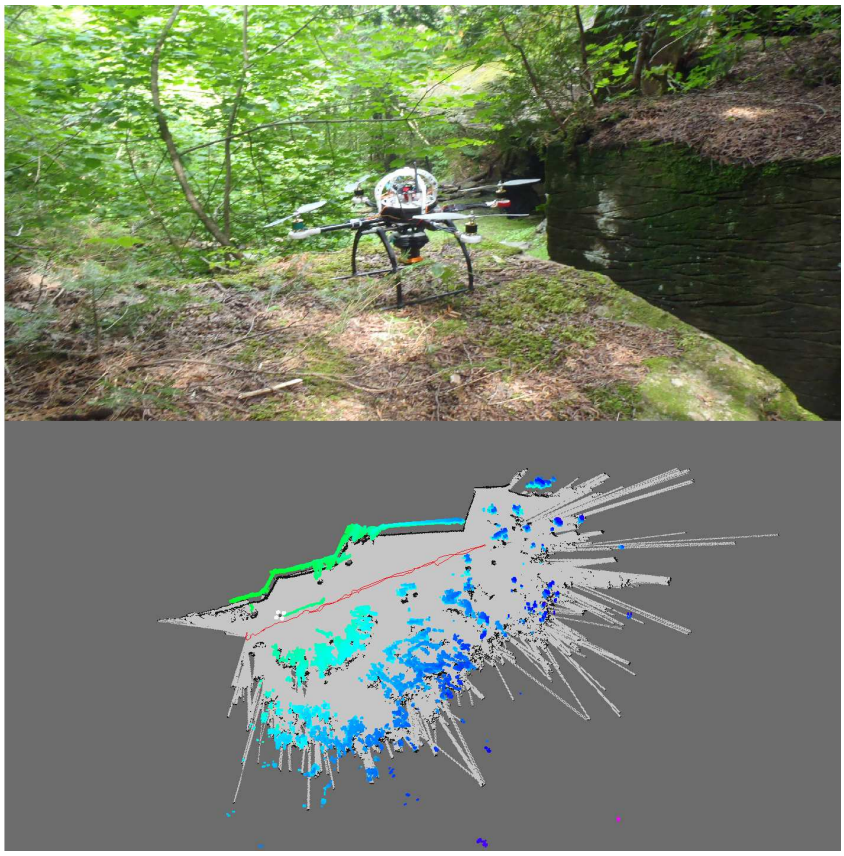


Figure 18: The experiment in an outdoor unstructured environment. **a)** The test environment. **b)** The developed map and trajectory of the robot. Laser measurements are color-coded.

4.6 Discussion

The sequence of experiments, from the simulated quadrotor to the ground robot and then to the COBRA quadrotor, demonstrated an efficient method to construct the autonomous navigation. This approach minimizes the amount of time and resources to develop components of the system. Moreover, it shows the portability of the sensor suite and the autonomy solution from the ground robot to the quadrotor.

In simulation, as seen in the videos, the motion of the quadrotor was smoother than in the last experiment, performed in the real world. This is because tuning the controller in a simulated environment is easier than in a real-world environment. Also, wind effects are ignored in the simulated model. This can not be ignored in real-world experiments, especially when the quadrotor is flying close to the ground surface where surface effects visibly affect the stability of the quadrotor.

This experiment was done with the onboard IMU and the laser ranger. The 2D SLAM algorithm was able to perform localization and mapping successfully. These two sensors are minimum sensors required to perform SLAM with the quadrotor. The Kinect camera as an extra source of information may help provide more accurate results. According to the third experiment, the errors in attaining the target points, the goal and the home, are very small. For the goal point, it is 0.16 meters and for the home it is 0.11 meters. This means that the quadrotor can perform specific actions while it is at a specific point. For instance, it can land on a target point such as a ground robot, or it can pick up a light object from a point and put it in a specific place.

5 Conclusion and Future Work

In this work, an autonomy solution for an unmanned rotorcraft was proposed and implemented. The proposed solution tackles a few key requirements for autonomous navigation in GPS-denied environments, including mapping, localization, exploration, and path planning. A behavior based mission control was proposed to plan, organize, and sequence different flight behaviors. The proposed solution is composed of a sensor suite and an autonomous navigation stack which enable the quadrotor to navigate and achieve the desired goals autonomously. The proposed autonomous navigation system was tested in Gazebo with a simulated flying rotorcraft and in real-world environments with an unmanned ground robot and a custom-designed quadrotor. The various simulated and real-world experiments demonstrate the effectiveness of the proposed method and sensor suite.

In the future, it would be desirable to extend the current work to multiple-robots: cooperatively exploring, mapping, localizing, and performing the mission. Additionally, adding a Kinect camera to the sensor suite and improving the ability of the quadrotor to carry heavier payloads will be investigated.

Acknowledgments

This research is supported by Defence Research and Development Canada (DRDC), Natural Sciences and Engineering Research Council of Canada (NSERC), and Canada Foundation for Innovation (CFI).

References

[1] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue," *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[2] I. Palunko, P. Cruz, and R. Fierro, "Agile load transportation : Safe and efficient load manipulation with aerial robots," *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 69–79, 2012.

[3] A. Nagaty, S. Saeedi, C. Thibault, M. Seto, and H. Li, "Control and navigation framework for quadrotor helicopters," *Journal of Intelligent and Robotic Systems*, vol. 70, no. 1-4, pp. 1–12, 2013.

[4] S. Bouabdallah, A. Noth, and R. Siegwart, "PID vs LQ control techniques applied to an indoor micro quadrotor," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004, pp. 2451–2456.

[5] X. Dong, B. M. Chen, G. Cai, H. Lin, and T. H. Lee, "Development of a comprehensive software system for implementing cooperative control of multiple unmanned aerial vehicles," *International Journal of Robotics and Automation*, vol. 26, no. 1, 2011.

[6] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[7] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro-UAV testbed," *IEEE Robotics and Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.

[8] A. Mokhtari, A. Benallegue, and Y. Orlov, "Exact linearization and sliding mode observer for a quadrotor unmanned aerial vehicle," *International Journal of Robotics and Automation*, vol. 21, no. 1, pp. 39–49, 2006.

[9] A. Jabbar and F. M. Malik, "Sampled-data backstepping control of a quadrotor unmanned aerial vehicle," *International Journal of Robotics and Automation*, vol. 4, no. 2, 2014.

[10] S. Thrun, M. Diel, and D. Hahnel, "Scan alignment and 3-D surface modeling with a helicopter platform," *Field and Service Robotics*, vol. 24, pp. 287–297, 2006.

[11] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a navigation system for autonomous indoor flying," in *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2009, pp. 2878–2883.

[12] A. Bachrach, R. He, and N. Roy, "Autonomous flight in unknown indoor environments," *International Journal of Micro Air Vehicles*, vol. 1, no. 4, pp. 217–228, 2009.

[13] H. D. Whyte and T. Bailey, "Simultaneous localization and mapping (SLAM): Part I the essential algorithms," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[14] S. Saeedi, L. Paull, M. Trentini, M. Seto, and H. Li, "Group mapping: A topological approach to map merging for multiple robots," *IEEE Robotics Automation Magazine*, vol. 21, no. 2, pp. 60–72, 2014.

[15] ——, "Efficient map merging using a probabilistic generalized Voronoi diagram," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4419–4424.

[16] ——, "Map merging for multiple robots using hough peak matching," *Robotics and Autonomous Systems*, vol. 62, no. 10, pp. 1408–1424, 2014.

[17] ——, "Map merging using Hough peak matching," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 4683–4688.

[18] S. Saeedi, L. Paull, M. Trentini, and H. Li, "A neural network-based multiple robot simultaneous localization and mapping," *IEEE Transactions on Neural Networks*, vol. 22, no. 12, pp. 2376–2387, 2011.

[19] ——, "A neural network-based multiple robot simultaneous localization and mapping," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 880–885.

[20] ——, "Occupancy grid map merging for multiple robot simultaneous localization and mapping," *International Journal of Robotics and Automation*, vol. 30, no. 2, pp. 149–157, 2015.

[21] ——, "Multiple robot simultaneous localization and mapping," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 853–858.

[22] C. Stachniss, *Robotic Mapping And Exploration*. Springer Tracts in Advanced Robotics, 2009, vol. 55.

[23] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: The MIT press, 2005.

[24] S. Saeedi, M. Trentini, M. Seto, and H. Li, "Multiple-robot simultaneous localization and mapping: A review," *Journal of Field Robotics*, 2015.

[25] L. Paull, S. Saeedi, M. Seto, and H. Li, "Auv navigation and localization: A review," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 131–149, 2014.

[26] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.

[27] L. Paull, S. Saeedi, M. Seto, and H. Li, "Sensor-driven online coverage planning for autonomous underwater vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1827–1838, 2013.

[28] I. Dryanovski, W. Morris, and J. Xiao, "An open-source pose estimation system for micro-air vehicles," in *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2011, pp. 4449–4454.

[29] S. Kohlbrecher, O. v. Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," in *IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, 2011, pp. 155–160.

[30] N. Michael, S. Shen, K. Mohta, Y. Mulgaonkar, V. Kumar, K. Nagatani, Y. Okada, S. Kiribayashi, K. Otake, K. Yoshida, K. Ohno, E. Takeuchi, and S. Tadokoro, "Collaborative mapping of an earthquake-damaged building via ground and aerial robots," *Journal of Field Robotics*, vol. 29, no. 5, pp. 832–841, 2012.

[31] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slambased navigation for autonomous micro helicopters in gps-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[32] R. A. Brooks, "Elephants don't play chess," *Robotics and Autonomous Systems*, vol. 6, pp. 3–15, 1990.

[33] M. J. Mataric, "Behavior-based control: Examples from navigation, learning, and group behavior," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 9, pp. 323–336, 1997.

[34] 2013, retrieved June 10, 2013, from http://www.flyingmachinearena.org/videos/.

[35] K. Konolige, "Projected texture stereos," in *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2010.

[36] 2013, retrieved June 10, 2013, http://www.mesa-imaging.ch/.

[37] 2013, retrieved June 10, 2013, http://www.primesense.com/.

[38] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments." in *Proceedings of the International Symposium on Experimental Robotics (ISER)*, 2010.

[39] N. Engelhard, F. Endres, J. Hess, J. Sturm, and W. Burgard, "Real-time 3D visual SLAM with a hand-held RGB-D camera," in *Proceedings of the RGB-D Workshop on 3D Perception in Robotics at the European Robotics Forum*, 2011.

[40] A. Segal, D. Haehnel, and S. Thrun, "Generalized ICP," in *Proceedings of of Robotics, Science and Systems (RSS)*, 2009.

[41] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, and C. Hertzberg, "Hierarchical optimization on manifolds for online 2D and 3D mapping," in *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2010.

[42] B. Yamauchi, A. Schultz, and W. Adams, “Integrating exploration and localization for mobile robots,” *Autonomous Robots*, 1999.

[43] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, “octoMap: A probabilistic, flexible, and compact 3D map representation for robotic system,” in *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2010.

[44] 2013, retrieved June 10, 2013, http://gazebosim.org/.

[45] 2013, retrieved June 10, 2013, http://www.draganfly.com/.

[46] 2013, retrieved July 3, 2013, http://www.ece.unb.ca/COBRA/quadrotor.htm.

Sajad Saeedi received his BEng in Electrical Engineering from K.N. Toosi University of Technology in 2001. He received his M.Sc. from Tarbiat Modares University in 2004, and his PhD from the University of New Brunswick in 2014. His research interests include simultaneous localization and mapping, multi-agent systems, intelligent systems, and nonlinear control.

Amr Nagaty received his B.Sc. in Mechatronics Engineering from the German University in Cairo in 2011. He also received his M.Sc. degree in Electrical Engineering from the University of New Brunswick in 2014. His research interests include: control theory, state estimation, vision and unmanned systems.

Carl Thibault received his B.Sc. degree in mechanical engineering from the University of New Brunswick in 2009. He received his M.Eng. degree in Electrical Engineering from the same university in 2014. His interests include unmanned vehicles systems, vertical take off and landing aircraft, controls and application of robotics.

Michael Trentini received his B.Sc. degree in mechanical engineering from Queens University in 1994, and the Ph.D. degree in mechanical engineering from the University of Calgary in 1999. In 2003, he joined the Autonomous Intelligent Systems Section at Defence Research and Development Canada, and in 2008 became the Group Head of the Complex Controls Group. His current research interests include control systems theory for helicopter and quad-rotor flight control systems and the control of unmanned systems that combine control, vision, map building and path planning to navigate complex unstructured environments.

Howard Li is an Associate Professor in the Department of Electrical and Computer Engineering, University of New Brunswick, Canada. He received his PhD from the University of Waterloo, Canada. He also worked for Atlantis Systems International, Defence Research and Development Canada, and Applied AI Systems Inc. to develop unmanned ground vehicles, unmanned aerial vehicles, autonomous underwater vehicles, and mobile robots for both domestic and military applications. He is a registered Professional Engineer in the Province of Ontario, Canada. His research interests include linear control, nonlinear control, intelligent control, distributed control, unmanned vehicles, mechatronics, robotics, multi-agent systems, artificial intelligence, motion planning, and simultaneous localisation and mapping.