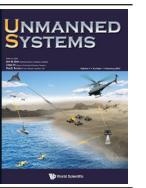
Unmanned Systems, Vol. 5, No. 3 (2017) 181–196

DOI: 10.1142/S2301385017400064

UNMANNED SYSTEMS
http://www.worldscientific.com/worldscinet/us



3D Mapping for Autonomous Quadrotor Aircraft

Sajad Saeedi[*,‡], Carl Thibault[*,§], Michael Trentini[†,¶], Howard Li[*,‖]

[*]*University of New Brunswick, Fredericton, Canada*

[†]*Defence Research and Development Canada-Suffield, Alberta, Canada*

Autonomous navigation in global positioning system (GPS)-denied environments is one of the challenging problems in robotics. For small flying robots, autonomous navigation is even more challenging. These robots have limitations such as fast dynamics and limited sensor payload. To develop an autonomous robot, many challenges including two-dimensional (2D) and three-dimensional (3D) perception, path planning, exploration, and obstacle avoidance should be addressed in real-time and with limited resources. In this paper, a complete solution for autonomous navigation of a quadrotor rotorcraft is presented. The proposed solution includes 2D and 3D mapping with several autonomous behaviors such as target localization and displaying maps on multiple remote tablets. Multiple tests were performed in simulated and indoor/outdoor environments to show the effectiveness of the proposed solution.

Keywords: Autonomous navigation; quadrotor; 3D SLAM.

US

1. Introduction

In robotics, perceiving the environment and representing the acquired knowledge have always been the challenging problems. 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.

Quadrotors, the most popular unmanned aerial vehicles (UAVs), have recently attracted a lot of attention in the robotics community. These robots have outstanding capabilities. They are less complex compared with conventional helicopters and are 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 with a quadrotor rotorcraft and

Received 29 August 2016; Revised 22 June 2017; Accepted 23 June 2017; Published 9 October 2017. This paper was accepted for publication in the Special issue on Aerial Robotics: Success Stories of Today and Hopes of Tomorrow. Guest Editors: Farrokh Janabi Sharifi, Youmin Zhang, Hossein Bonyan Khamseh and Howard Li.
Email Addresses: ‡*sajad.saeedi.g@unb.ca*, §*carl.t@unb.ca*, ¶*howard@unb.ca*, ‖*Mike.Trentini@drdc-rddc.gc.ca*

achieve autonomy with minimum human intervention. The paper explains the requirements of three-dimensional (3D) mapping for an autonomous quadrotor aircraft. The paper is an extension to the previous work [1] where the autonomy for the quadrotor was achieved using a custom-designed sensor suite including a laser ranger and an IMU. In the extended project, an RGB-D camera is added to the quadrotor and the quadrotor has been modified and redesigned to carry the new sensor. In the extended project, new features such as 3D mapping, target localization, and remote mapping have been introduced.

Autonomous navigation in GPS-denied environments has many applications such as surveillance, security, maintenance, and inspection. An autonomous robot can achieve the goals of these applications more efficiently by minimizing the overall cost and avoiding human error. With the motivation of building an autonomous robotic system for such applications, the goal is to enable the following capabilities on quadrotor:

- Localization and mapping with one scanning laser ranger and one IMU.
- Localization and mapping with one RGB-D camera and one IMU.
- Localization and mapping with a combination of these sensors.
- Exploring an unknown environment autonomously.
- Path planning between desired waypoints.
- Localizing targets.
- Following the high-level goals of a mission.
- Interacting with multiple users through tablets.

The rest of the work is organized as follows: Section 2 presents the 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 the experimental results in simulated and real-world environments. Finally, Sec. 5 summarizes the work.

1.1. *Contribution*

This paper builds upon the previous works of the authors [1, 2]. The preliminary results of the paper was accepted for presentation at IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) [3]. In addition to the previous contributions of the authors, such as exploration and two-dimensional (2D) mapping by the quadrotor, this paper's novel contributions are summarized as follows:

- The mapping is performed in 2D and 3D. In other words, the system can perform 2D mapping and localization with a scanning laser ranger and an IMU or to perform 3D mapping and localization with an RGB-D camera and an IMU. It is also possible to do 2D and 3D localization and mapping simultaneously.
- Target localization is performed based on the color of the targets which can help the quadrotor to perform various autonomous tasks. The recognized targets are marked in the 3D map, so that the remote users and operators can easily identify them and take proper actions.
- The robot can implement any sequence of the required behaviors such as mapping, localization, path planning, exploration, target detection, and interaction with users.
- Maps are displayed on multiple remote tablets, providing multiple operators with real-time information. The operators can modify the mission based on the acquired information.
- The proposed work is designed in such a way that it can be utilized on ground robots as well. The sensor suite and the software package are easily portable to various other robot platforms.

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 the background to its history, motivation, definition, advantages, and disadvantages.

Although quadrotors are less complex than helicopters, there are limitations that impose challenges on quadrotor autonomous navigation. These limitations include

- **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,

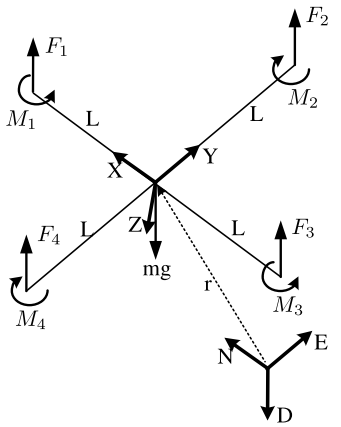


Fig. 1. The coordinate systems and the free body diagram of the quadrotor with acting forces. The frame shown by *X*–*Y*–*Z* is the body frame. The frame shown by *N*–*E*–*D* is the inertial frame, following the North–East–Down standard.

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.

Figure 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 ($i = 1, \ldots, 4$) are the aerodynamic forces and moments produced by the ith rotor, respectively. L is the moment arm, the distance from the center of the 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 [4].

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

Figure 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 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 [4, 5]. For more information on controlling a quadrotor, see [6, 7].

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, Thrun *et al.* [8] 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.* [9] from the University of Freiburg and Bachrach *et al.* [10] 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 from 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.

Figure 3 depicts these key problems and their relationships. *Mapping* is the process of modeling a robot's world with 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.

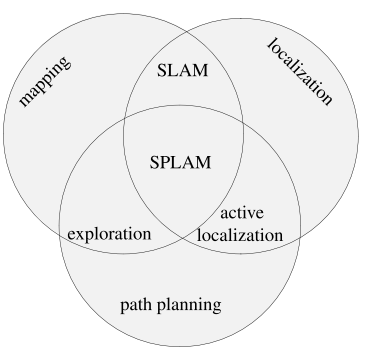


Fig. 3. Tasks that need to be accomplished towards map learning [11]. An autonomous robot needs mapping, localization, and path planning to complete its tasks efficiently.

There is a tight dependency between these components. In an unknown environment, mapping and localization cannot be decoupled. This is because to make a map, the robot needs to know its current position for which it needs to have a map. *Simultaneous localization and mapping* (*SLAM*) addresses this issue. 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 [11]. Complete information on different methods for components of map learning in relation to mapping and localization can be found in [12], and for path planning in [13].

SLAM and path planning for indoor quadrotors were first performed in 2009 by Grzonka *et al.* [9] from the University of Freiburg and Bachrach *et al.* [10] from MIT, independently. Later, other researchers continued the same trend to improve on the previous results. In 2010, Dryanovski *et al.* [14] at the City College of New York presented their approach for quadrotor SLAM. It was similar to the work proposed by Grzonka *et al.* [9]. In 2011, Kohlbrecher *et al.* [15] 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 ranger as a key perception sensor to perform SLAM. Other researchers at the University of Pennsylvania [16] and the Swiss Federal Institute of Technology in Zurich (ETH Zurich) [17] published similar results for SLAM or path planning with the quadrotor.

In the literature, there are various 3D mapping algorithms; however, most of them have limitations in real-world applications. The Kinect Fusion algorithm [18] is a fast and reliable 3D mapping algorithm, but it operates in relatively small environments and needs very fast processing units. It is based on the iterative closest point matching and the main contribution of the paper is representing the map of the environment using signed distance function (SDF) [19] instead of the traditional occupancy grid map.

The 3D dense tracking and mapping algorithm [20] is also based on the SDF. This algorithm is very sensitive to the loss of data, and in the reported experiments, the captured depth and color images are transmitted to the ground control station using a wired link. This is a major limitation for quadrotors.

In the fast visual odometry and mapping [21], a 3D mapping algorithm is used to develop real-time 3D maps. The mapping algorithm uses extracted features from consecutive images. It develops a model of the world using the extracted features. Once a new frame is received, it is matched with the model. Compared with other mapping algorithms which are based on matching images frame-by-frame, the model-based mapping accumulates significantly less error; however, the algorithm has drawbacks which limit its application. First, it requires a wired link for data transmission. Second, the change in the pose of the camera should be very small; otherwise, the mapping algorithm fails to produce consistent maps.

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 [22, 23]. 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 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 [22].

3. Proposed Perception and Autonomy Solution

This section presents the proposed method for autonomous quadrotor navigation.

3.1. *System overview*

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 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*

There are few works that offer a fully autonomous quadrotor system [24]. Most research that demonstrates outstanding results with quadrotors, such as the work by Nathan *et al.* [7] and Juggling Quadrotors by ETH [25], 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 senors require further processing which is the main challenge in real-world applications.

Some studies lack path planning or exploration as an important part of map learning [26–29]. For instance, in [14, 15], 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*

As shown in Fig. 5, our proposed method has two modules: *Mission Planner* and *Autonomy*, each with several subblocks. The *Mission Planner* module is responsible for the autonomous behaviors, and the *Autonomy* module implements the behaviors.

As shown in Fig. 5, sensor connections are shown by dashed lines, for optional sensors, and solid lines, for required sensors. Each block performs the following actions:

- 2D SLAM: using the IMU and the horizontal scanning laser rangefinder, a 2D SLAM is performed.
- 3D SLAM: using the RGB-D camera, a visual 3D SLAM is performed.
- KF Fusion: the results of these two SLAM block, the altimeter measurements, and the IMU measurements are fused by a Kalman filter. In outdoor environments, if GPS is available, it is also used in this block.
- Planner: in this block, the pose of the vehicle is used to find waypoints and plan paths for navigation.
- Controller: the calculated waypoints and position feedback are passed to this block to drive the quadrotor.
- Obstacle Avoidance: this block uses the laser ranger to avoid dynamic obstacles.
- Voxel mapping: using the generated accurate pose of the vehicle, a 3D volumetric map is built by the vertical scanning laser rangefinder.

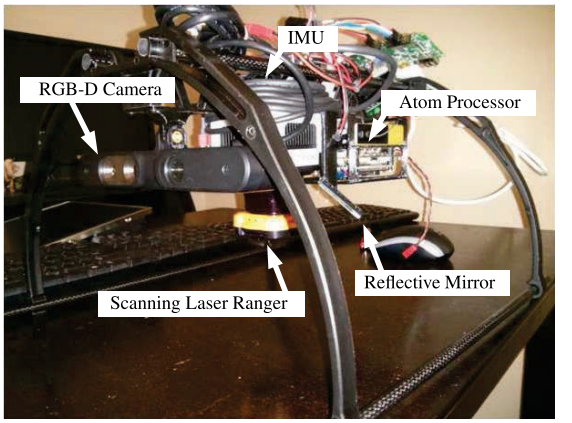


Fig. 4. The sensor suite of the quadrotor includes an IMU (not visible in the figure), an RGB-D camera, a laser ranger, and a reflective mirror. An atom processor is used to interface the sensors and communicate with the ground control station.

Figure 4 shows the sensors of the quadrotor. The sensor suite includes a CH-Robotics IMU, an AUSU Xtion Pro RGB-D camera, a Hokuyo UTM-30LX scanning laser rangefinder with a horizontal scan, and a reflective mirror mounted on the laser ranger to reflect a few beams to the ground and measure the altitude of the quadrotor. An atom processor is used to interface the sensors and communicate with the ground control station.

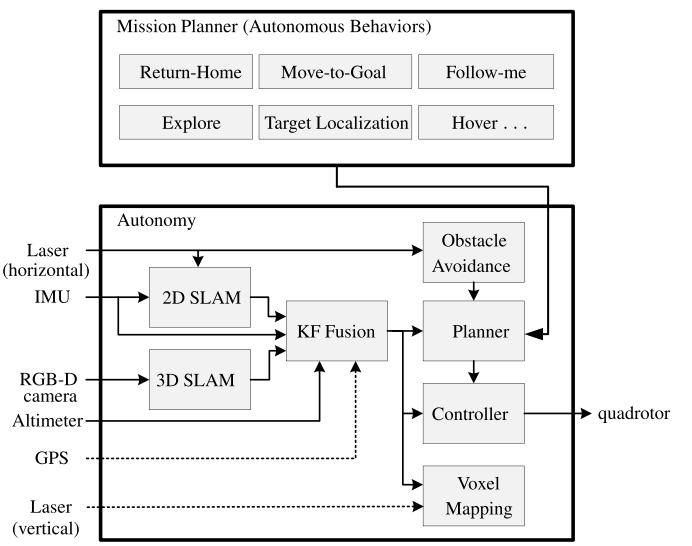


Fig. 5. The proposed method has two modules: *Mission Planner* and *Autonomy*, each with several subblocks. The *Mission Planner* module is responsible for the autonomous behaviors, and the *Autonomy* module implements the behaviors.

3.4.1. *Autonomous behaviors*

Any complicated mission can be represented as a set of behaviors [2]. A sample mission composed of the aforementioned behaviors is presented in Fig. 6. 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 6 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 behaviors 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 criterion is the distance of the robot to the goal. If the distance is less than a threshold, the behavior 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 behavior, and the exploration behavior continues until there is nothing left to explore.

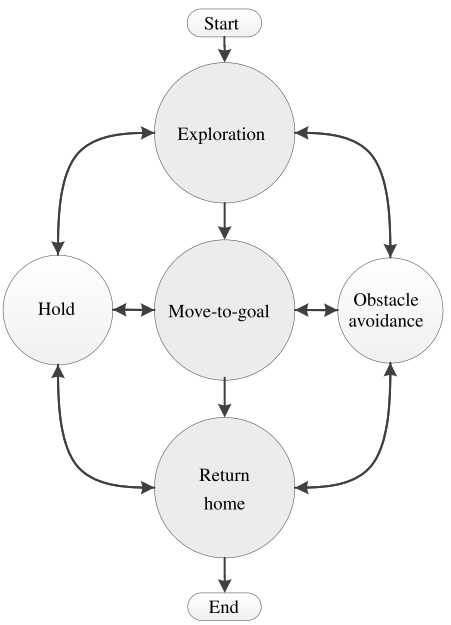


Fig. 6. A sample mission composed of basic navigation behaviors. First, the robot explores an unknown environment, then it moves to a given goal. Once it arrives at the goal, it returns home. While performing these behaviors, obstacle avoidance and hold (as an emergency stop) are running at a higher priority than others.

3.4.2. *2D SLAM*

Using the horizontal scanning laser ranger, a 2D occupancy grid map is generated and used for navigation. The grid map is used for path planning and exploration. The 2D SLAM algorithm has been explained in the authors' previous work [2].

3.4.3. *3D SLAM*

Features of the environment can be used to calculate the pose of the robot. 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 an Asus Xtion Pro RGB-D camera using a 3D SLAM algorithm.

To perform 3D SLAM with a flying robot such as a quadrotor, three major problems should be addressed. These problems are computational demand, communication issues, and fast dynamics. In the next few paragraphs, each of these issues is presented in detail.

Most 3D SLAM algorithms which are based on the RGB-D cameras rely on very fast processing units. Unfortunately, such units are rarely available on quadrotors. One solution to handle this problem is to transmit image and depth data to a ground control station, perform 3D mapping on the workstation, and send the results to the quadrotor. This effective solution creates a major limitation which is the need for reliable and high bandwidth communication channels. Compared with other sensors, such as the scanning laser ranger, the data rate of an RGB-D camera is very high. For instance, for QVGA quality, the size of each depth image is 320×240 pixels and the size of each color image is $3 \times 320 \times 240$ pixels. The camera operates at 30 Hz. Each color pixel occupies one byte, and each depth pixel occupies two bytes; therefore, the transmission rate of the camera is $(2 \times 320 \times 240 + 3 \times 320 \times 240) \times 30 = 11.52\,\text{MB/s}$, or approximately 11 MB/s. In contrast, a scanning laser ranger which operates at 40 Hz and has 1082 beams (each beams produces a float64 and needs four bytes) has the transmission rate of $1082 \times 40 \times 4 = 0.17\,\text{MB/s}$. This rate is approximately 68 times less than the data rate of the camera. In such high rate transmission, the rate of the package loss also increases. Additionally, some frames arrive out-of-sequence which either should be discarded or reprocessed with some previous frames.

Another major problem in 3D feature-based SLAM with quadrotors is its fast dynamics. If the quadrotor has a large rotation or translation, the amount of the overlap between

two consecutive frames of the onboard camera decreases. Therefore, matching features of the frames will produce invalid estimates.

Most 3D SLAM algorithms with RGB-D sensors use a wired link connected to a high speed processing unit to avoid communication issues and high computational demand. For instance, [18, 21] use a wired connection. [20] also develops 3D SLAM on a quadrotor while the camera is connected to a ground control station with a USB link. Additionally, these algorithms suffer from fast dynamics of the quadrotor.

To deal with these limitations, the information from the IMU and the laser ranger is used to produce an initial and accurate estimate of the 3D pose of the quadrotor. This initial estimate, which includes the 3D coordinates and 3D orientation of the robot, is refined by using the information from the RGB-D camera. There is no need to have a wired link. Also, the motion of the camera can be very aggressive, and sudden changes, like the changes we expect from a quadrotor, are dealt easily.

Figure 7 shows the required processing blocks of the proposed 3D SLAM algorithm. Inputs of 3D SLAM are color 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. The first three blocks, *Feature Extraction*, *Initial Pose Estimation*, and *3D Pose Estimation*, provide accurate 3D visual odometry. The last block, *Pose Graph Optimization*, provides global consistency of transformations and detects loop closures. In the visual odometry, first Shi–Tomasi features are extracted from the RGB image. Then, the results of 2D SLAM and roll and pitch angles are used to provide an initial estimate of the change-in-pose of the robot. This is an initial estimate and needs to be refined by the model-based pose estimation algorithm. Here, each block is explained in detail.

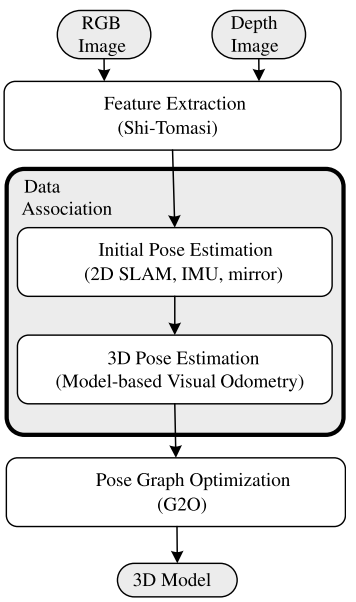


Fig. 7. Flowchart of 3D SLAM. First, Shi–Tomasi features are extracted. Then, an initial estimate is determined using 2D SLAM, IMU, and the reflective mirror. Then, the extracted features are matched with a model of the environment. Finally, a G2O optimization is performed over the poses.

Feature Extraction: To extract features, Shi–Tomasi, which is a corner detector algorithm, is used. Shi–Tomasi descriptor is rotation invariant and offers a good tradeoff between computational demand and robustness [30].

Initial Pose Estimation: When the motion of camera involves a large rotation or translation, any visual odometry algorithm will fail to provide accurate pose estimates. Having a large rotation or translation is very common in robots with fast dynamics, such as quadrotors. Therefore, it is very important to have an initial estimate of the pose of the robot.

For ground robots, direct odometry can be used as an approximate initial estimate, but such direct odometry is not available for quadrotors. As mentioned previously, the proposed 2D SLAM block provides x, y, and heading of the robot. Based on the extensive experiments, the 2D pose of the quadrotor is very accurate and reliable. Additionally, the roll and pitch angles, which are directly used from the EKF implementation of the IMU, have less than one degree error. And the z component can easily be calculated using the reflective mirror mounted on the laser. About 10 beams are reflected to the ground, and the results are averaged to produce an average distance. If the average distance of the beams is d, the z component of the 3D pose is calculated as follows:

$$z = d \cos\phi \cos\theta, \tag{1}$$

where ϕ and θ are roll and pitch angles. The 2D SLAM, IMU, and reflective mirror can provide a reliable initial estimate of the 3D pose of the robot. The initial estimate is refined in the next block.

3D Pose Estimation: 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.

In most visual odometry algorithms, features of two consecutive frames are extracted and matched. While this frame-based feature matching looks efficient, it operates locally which means the correlation of the features in the current frame with two or three frames earlier or very past frames is ignored. To address this issue, instead of frame-based feature matching, a model-based feature matching paradigm is used.

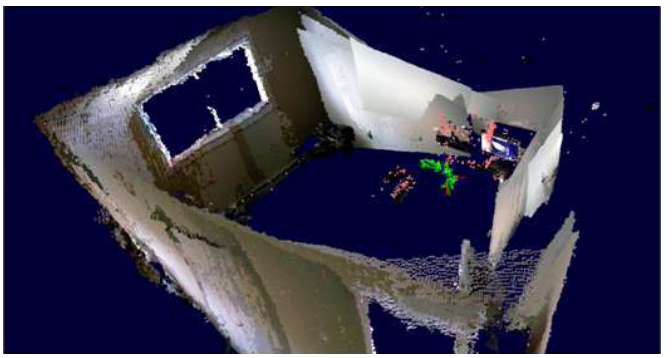


Fig. 8. A 3D map, developed using the model based mapping algorithm.

To perform model-based visual odometry, first a model of the world using the extracted features is developed. Once a new frame is received, it is matched with the model. The new features which are not available in the model are added to the model. This process is repeated with each incoming frame. The model-based mapping is a reliable and robust algorithm. Compared with other mapping algorithms which are based on matching images frame-by-frame, the model-based mapping accumulates significantly less error. Figure 8 shows a developed map using this algorithm.

Pose Graph Optimization: The global and consistent mapping of the SLAM algorithm is based on the G2O algorithm which is a bundle adjustment algorithm [31]. 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 [12]. This is performed by the SLAM back-end [32]. Usually the optimization by the back-end takes more time and memory than extracting the constraints; therefore, it runs at a lower frequency.

3.4.4. *Target localization*

A target localization algorithm is used to localize targets according to their colors [33]. The localized targets are marked in the developed map with an arrow. Two attached videos of two experiments demonstrate this capability. The algorithm can identify multiple targets of the same color or different colors, as defined in the configuration file. The targets are identified in the image frame, but since the depth information is also available, the 3D coordinates of the targets are easily calculated. Given that the pose of the camera is known, the global coordinates of the targets are calculated by transforming the coordinates from the camera frame to the global frame.

3.4.5. *Remote mapping*

It is often required to monitor the perception of the quadrotor on small, portable, and remote stations such as tablets. For instance, if a quadrotor is exploring a building where humans cannot enter, having access to camera view of the quadrotor and also 2D and 3D maps on multiple tablets will help the mission planner or quadrotor users to manage the mission efficiently.

To show the maps on multiple remote tablets, two different approaches were taken and tested. The first one was developing a web-based mapping application. In the web-based application, map processing is performed on a server, and the results are displayed as web content on a remote client computer. This approach requires webGL which works well on Windows and Android systems with fast CPUs; however, the application is not stable on Android tablets due to memory and processing limitations. For example, it works well on a Core2Duo laptop with Windows, but on a Galaxy Tab 3, in 50% of the tests, the browser fails.

The other approach taken to handle this problem is through web data transmission. Figure 9 shows the diagram of this approach. The 3D mapping application runs on the ground station. A jpg-server captures jpeg images from the map visualizer. The captured images are streamed to the tablets by the motion-jpeg server which is based on the motion-jpeg standard. Each tablet specifies the type of the image that should be streamed, and the motion-jpeg server on the workstation subscribes to the requested image and sends it to the tablet. Camera view can also be transmitted to the tablets.

3.4.6. *State estimation*

Pose estimations by 2D SLAM and 3D SLAM are fused with IMU measurements. The fused poses are then used by the controller and planner blocks. More details about the state estimation algorithm is presented in the authors' previous work [2].

3.4.7. *Planner*: *Exploration of unknown space*

The frontier algorithm is used to explore an unknown environment [34]. Algorithm 1 explains the frontier exploration.

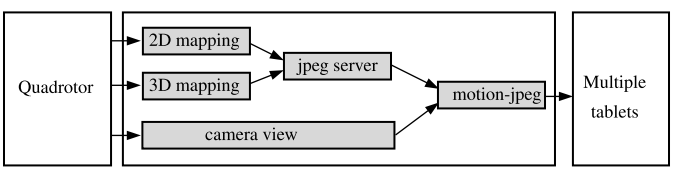


Fig. 9. Remote mapping on multiple tablets.

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 ith 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: $c_{\text{wp}} \leftarrow \text{argmax}_{c^i} \frac{n_c^i}{d_c^i}$.

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, $i = 1, \ldots, m$, 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 m, which eliminates the oscillatory behavior of the robot between the waypoints. Exploration terminates when there is no new waypoint, $c_{\text{wp}} = \emptyset$. This occurs, for instance, when there is no frontier cell left, or when the number of the cells in each cluster is negligible.

Once a waypoint is determined, the path planning algorithm designs a path to guide the robot.

3.4.8. *Planner*: *Path planning*

For path planning, the wavefront algorithm is used [13]. The algorithm has been modified to minimize any collision risks. The details are given in Algorithm 2. The free space is represented by a grid, marked with 0 as unvisited (line 5). The occupied and dilated cells are marked with 1 to avoid getting too close to the obstacles (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 (lines 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 (lines 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 (lines 14–19).

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: $q_{\text{goal}} \leftarrow$ 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: $q \leftarrow q_{\text{start}}$
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**

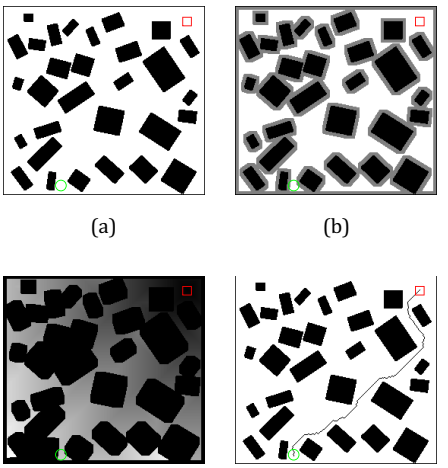


Fig. 10. An example of the wavefront algorithm. (a) A simulated environment with obstacles. Obstacles are shown in black while the free space is shown in white. A path should be designed from the start point, the green circle, to a goal point, the red square. (b) Obstacles are dilated to avoid getting 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 color-coded. (d) Using the generated wave, a path between the start and goal points is designed, shown by a black curve.

Figure 10 shows an example of the wavefront path planning algorithm. In Fig. 10(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. 10(b), obstacles are dilated to avoid getting too close to the obstacles. In Fig. 10(c), a wave is generated from the goal point to the start point. The wave is color-coded. Finally, Fig. 10(d) shows the path designed using the generated wave.

3.4.9. *Obstacle avoidance*

To avoid dynamic and unpredicted obstacles, the laser beams of the horizontal scanning laser ranger is used. First, the noisy beams are removed by a sliding window filter. Then, the beams are divided into several bins. Then, the average beam for each bin is calculated by averaging the beams in the bin. If a bin is likely to collide with an obstacle, then a different bin which has larger average beam is selected and the quadrotor is guided to that bin until there is no risk of collision. Once there is no risk, the previous path is restored and followed. Details of the algorithm have been explained in the authors' previous work [2].

3.4.10. *3D voxel mapping*

In this work, the Octomap algorithm is used to generate a voxel map [35]. Octomap is based on octree, where the space is repeatedly and dynamically divided into eight smaller cubes or voxels (see Fig. 11).

The occupancy probability of a voxel v given the sensor measurements from time 1 to t, $z_{1:t}$ is estimated by [35]

$$p(v|z_{1:t}) = \left(1 + \frac{1-p(v|z_t)}{p(v|z_t)} \frac{1-p(v|z_{1:t-1})}{p(v|z_{1:t-1})} \frac{1-p(v)}{p(v)}\right)^{-1}, \tag{2}$$

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.

3.4.11. *Controller*

The controller block is based on the cascaded controller developed by Nagaty *et al.* [4], which has inner loop and outer loop PID controllers. Once the PID parameters are tuned, the controller can deal with minor disturbances.

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

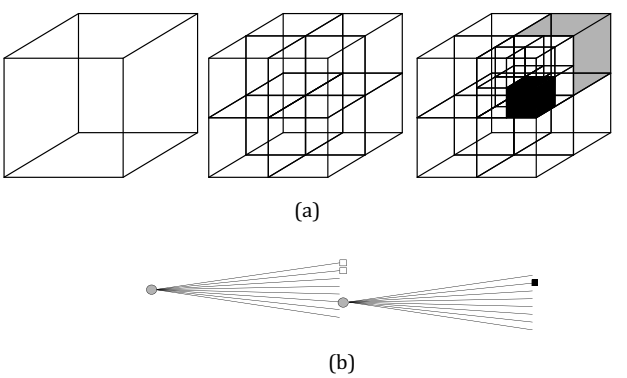


Fig. 11. An example of an octree structure. (a) Occupied and free octants depicted in black and gray. Other octants are unknown. (b) The corresponding tree representation [35].

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 [13]. 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 have demonstrated the performance. Quadrotors have fast dynamics; therefore, it is important to provide pose estimates as frequently as possible. The 2D SLAM algorithm operates at 40 Hz which is an acceptable rate to control a quadrotor.

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

4. Experiment

To test the proposed perception and navigation solution, the mission shown in Fig. 6 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. *Case1*: *Simulation in Gazebo*

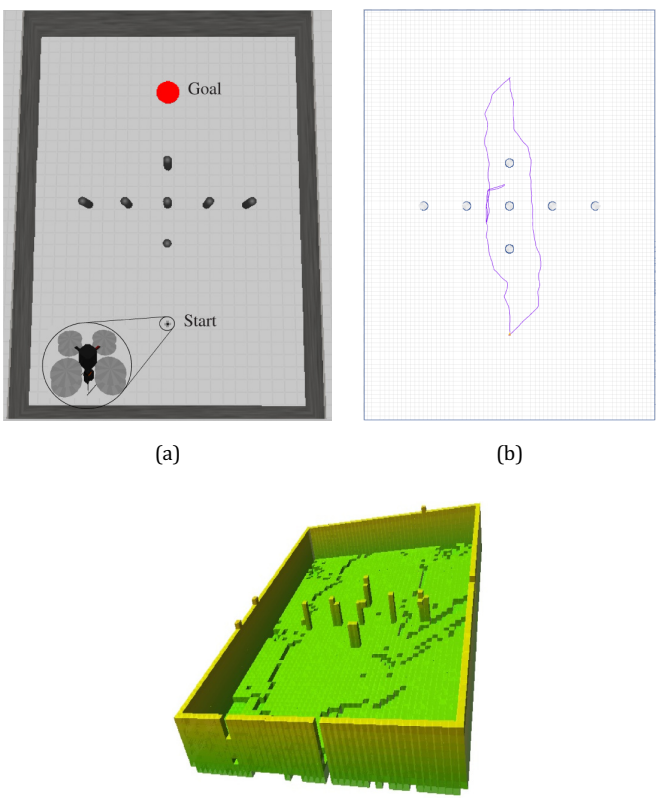
The first experiment was done in Gazebo [36], with a simulated X8 quadrotor [37], with an IMU, two perpendicular scanning laser rangers, and a SONAR sensor.

4.1.1. *Description*

Figure 12(a) shows the Gazebo world. Its size is 32×50 m. For this experiment, the mission shown in Fig. 6 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. 12(a)). The goal, which it attains after the exploration, is located 30 m away, marked by a red disc.

4.1.2. *Results*

The mission was done in 210 s. Figure 12(b) shows the occupancy grid map and Fig. 12(c) depicts the voxel map. The video of the experiment in available here [38].



(a) (b)

(c)

Fig. 12. Simulation in Gazebo. (a) Simulation world. (b) Occupancy grid map. (c) Voxel map.

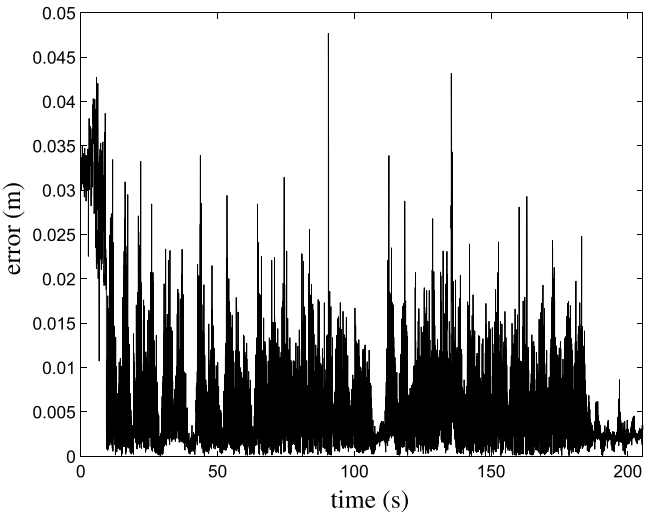


Fig. 13. Robot position error during mission.

Figure 13 shows the error of the position of the robot, the difference between the ground-truth and the estimated trajectory, during the mission. The average error is 0.0065 m.

4.2. *Case 2*: *Unmanned ground robot*

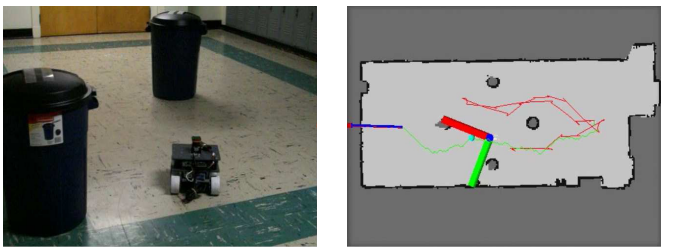
To make sure that the proposed solution works well in the real world, we started the experiments with a CoroBot built by CoroWare, Inc. The main objective for this test is to avoid crashing the flying robot due to unpredicted run-time problems. Furthermore, it is easier to test functionality of different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

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 m and four garbage bins were placed in the room as obstacles. Because of the floor is flat, there was no rolling or pitching involved. Figure 14(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*

Figure 14(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 [38]. All states of the mission are shown in the video.



(a) (b)

Fig. 14. 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.

4.3. *Case 3*: *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 transition from outdoors to indoors. Details of the experiment including flight time, the length of the path, and the error of achieving the goal points are not reported for the sake of brevity.

4.3.1. *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. Note 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.

4.3.2. *Results*

Figure 15 shows the simulated experiment. The figure represents the moment that the robot is returning to the start point. In Fig. 15(a), a snapshot of the simulated world is shown. The quadrotor is inside the building and is not visible. The green disc is the start point. Figure 15(b) shows the developed 3D map, built by the vertical laser range-finder. Figure 15(c) shows the camera view. Finally, Fig. 15(d) demonstrates the 2D map, developed by the

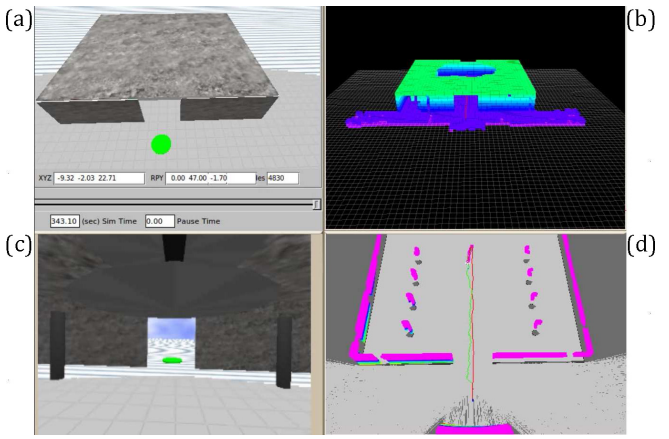


Fig. 15. Autonomous entry and exit in simulation. (a) Simulated world in Gazebo. (b) The developed 3D map. (c) Camera view. (d) The developed 2D map, path, and trajectory.

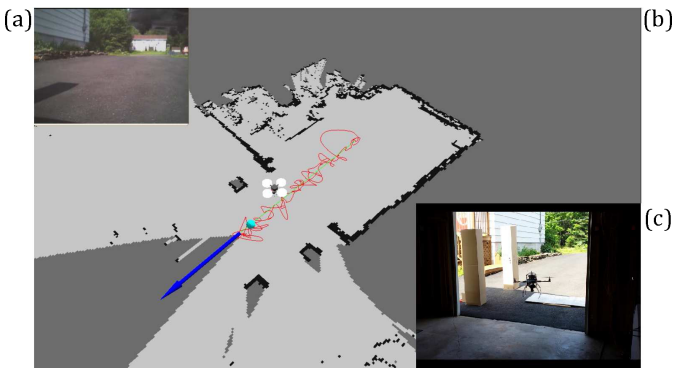


Fig. 16. Autonomous entry and exit in a real-world experiment. This figure shows the experiment at the moment that the quadrotor returns home. (a) Onboard camera view. (b) The developed 2D map, path, and trajectory. (c) Ground view of the quadrotor, exiting through the door.

horizontal laser rangefinder, designed path, and trajectory of the robot.

Figure 16 shows the same scenario in a real-world experiment. Figure 16(a), shows the test environment. Figure 16(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 [38]. There is no ground-truth trajectory to calculate the trajectory error; however, the error of returning to the exact start point after finishing the mission is 0.15 m.

4.4. *Case 4: Outdoor unstructured environment*

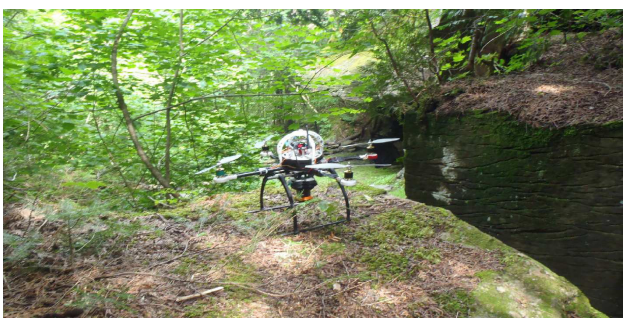
This outdoor experiment demonstrates the ability of the quadrotor in mapping and navigating in unstructured environments, such as woods and cliffs. The experiment is performed at UNB Woodlot in Fredericton, New Brunswick. This experiment demonstrates the usefulness of the quadrotor and the proposed solution 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.

4.4.1. *Results*

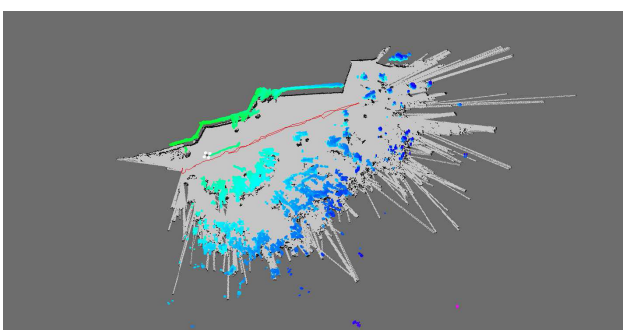
Figure 17(a) shows the experiment site. As the figure shows, the environment is not traversable by ground robots. Figure 17(b) shows the 2D developed map and trajectory of the robot. Trees and cliffs are mapped consistently by the sensor suite.

4.5. *Case 5: 3D mapping*

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 an ASUS RGBD camera, a Hokuyo UTM-30LX scanning laser ranger, a CH Robotics UM6 IMU,



(a)



(b)

Fig. 17. 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.

and a SensComps MINI-A PB Ultrasonic Transducer SONAR ranger. The scanning laser ranger is mounted upside down for more stability during flight. Figure 4 shows the quadrotor. An atom board is used to interface sensors and send data to a ground station. The test environment is approximately 60 m^2. The quadrotor explores the room, then it returns home.

4.5.1. *Results*

The mission took about 50 s. Figure 18(top) shows the 3D map of the environment. The green curve is the trajectory of the quadrotor. Figure 18(bottom) shows the 2D map of the same environment, developed at the same time that the 3D map was built. A video of the experiment is available in [38]. In the video, it has been demonstrated that a blue target has been identified and marked in the 3D map. The robot completed the mission by exploring, mapping, and navigating through obstacles successfully. Similarly, in this experiment, there is no ground-truth trajectory to measure the trajectory error, but the error of returning to the start point at the end of the mission is 0.12 m.

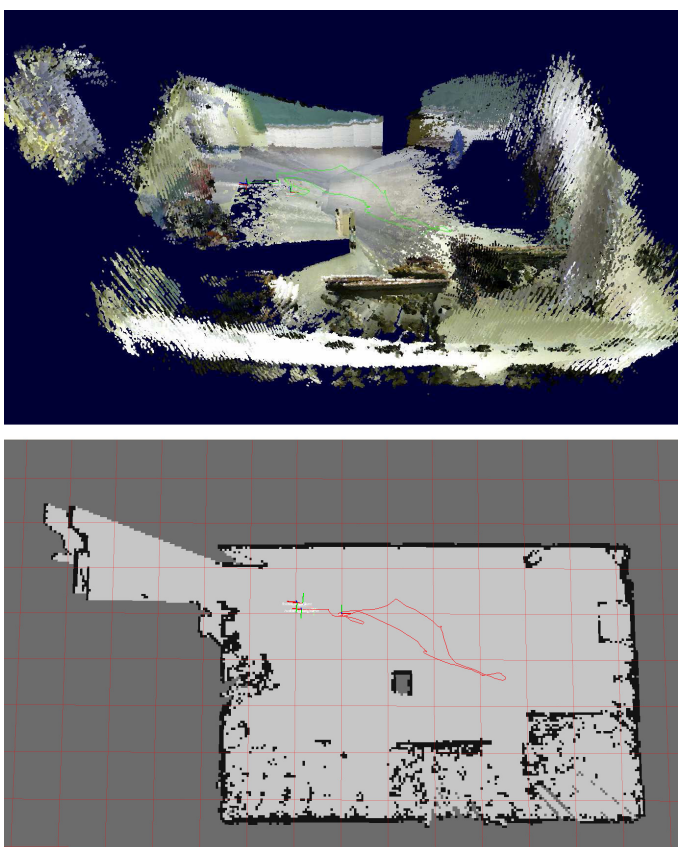


Fig. 18. (top) 3D map of an indoor test environment, and (bottom) 2D map of the same environment.

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.

During the last experiment, the Kinect camera was not used because of its heavy weight. However, with the 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 many experiments, the errors in attaining the target points, the goal and the home, are very small. For the goal point, it is 0.16 m and for the home it is 0.11 m. 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

This paper presented an autonomy solution for an unmanned rotorcraft, which includes mapping, localization, and path planning. To manage the mission, a behavior-based mission control was used. The proposed method was tested in simulated and real-world environments with commercial and custom designed quadrotors.

In the future, it would be desirable to extend the work to multiple-robots: cooperatively exploring, mapping, localizing, and performing the mission.

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] S. Saeedi, A. Nagaty, C. Thibault, M. Trentini and H. Li, Perception and navigation for autonomous rotorcraft, in *Int. Conf. Intelligent Unmanned Systems (ICIUS)*, Montreal, Canada (2014).

[2] S. Saeedi, A. Nagaty, C. Thibault, M. Trentini and H. Li, Perception and navigation for an autonomous quadrotor in GPS-denied environments, *Int. J. Robot. Autom.* **31**(6) (2016).

[3] S. Saeedi, A. Nagaty, C. Thibault, M. Trentini and H. Li, 3D mapping and navigation for autonomous rotorcraft, in *2016 IEEE 29th Canadian Conf. Electrical and Computer Engineering (CCECE)*, Vancouver, Canada (2016).

[4] A. Nagaty, S. Saeedi, C. Thibault, M. Seto and H. Li, Control and navigation framework for quadrotor helicopters, *J. Intell. Robot. Syst.* **70**(1–4) (2013) 1–12.

[5] S. Bouabdallah, A. Noth and R. Siegwart, PID vs LQ control techniques applied to an indoor micro quadrotor, in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, Sendai, Japan (2004), pp. 2451–2456.

[6] R. Mahony, V. Kumar and P. Corke, Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor, *IEEE Robot. Autom. Mag.* **19**(3) (2012) 20–32.

[7] N. Michael, D. Mellinger, Q. Lindsey and V. Kumar, The GRASP multiple micro-UAV testbed, *IEEE Robot. Autom. Mag.* **17**(3) (2010) 56–65.

[8] S. Thrun, M. Diel and D. Hahnel, Scan alignment and 3D surface modeling with a helicopter platform, *Field Serv. Robot.* **24** (2006) 287–297.

[9] S. Grzonka, G. Grisetti and W. Burgard, Towards a navigation system for autonomous indoor flying, in *Proc. IEEE/RSJ Int. Conf. Robotics and Automation (ICRA)*, Kobe, Japan (2009), pp. 2878–2883.

[10] A. Bachrach, R. He and N. Roy, Autonomous flight in unknown indoor environments, *Int. J. Micro Air Veh.* **1**(4) (2009) 217–228.

[11] C. Stachniss, *Robotic Mapping And Exploration* (Springer, 2009).

[12] S. Thrun, W. Burgard and D. Fox *Probabilistic Robotics* (The MIT press, Cambridge, MA, 2005).

[13] 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, Cambridge, MA, 2005).

[14] I. Dryanovski, W. Morris and J. Xiao, An open-source pose estimation system for micro-air vehicles, in *Proc. IEEE/RSJ Int. Conf. Robotics and Automation (ICRA)*, Shanghai, China (2011), pp. 4449–4454.

[15] S. Kohlbrecher, O. V. Stryk, J. Meyer and U. Klingauf, A flexible and scalable SLAM system with full 3D motion estimation, in *IEEE Int. Symp. Safety, Security and Rescue Robotics (SSRR)*, Kyoto, Japan (2011), pp. 155–160.

[16] 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, *J. Field Robot.* **29**(5) (2012) 832–841.

[17] S. Weiss, D. Scaramuzza and R. Siegwart, Monocular-slambased navigation for autonomous micro helicopters in gps-denied environments, *J. Field Robot.* **28**(6) (2011) 854–874.

[18] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges and A. Fitzgibbon, KinectFusion: Real-time dense surface mapping and tracking, in *2011 10th IEEE Int. Symp. Mixed and Augmented Reality (ISMAR)*, Basel, Switzerland (2011), pp. 127–136.

[19] B. Curless and M. Levoy, A volumetric method for building complex models from range images, in *Proc. 23rd Annual Conf. Computer Graphics and Interactive Techniques, SIGGRAPH '96*, New Orleans, LA, USA (1996), pp. 303–312.

[20] J. Sturm, E. Bylow, F. Kahl and D. Cremers, Dense tracking and mapping with a quadrocopter, in *Unmanned Aerial Vehicle in Geomatics (UAV-g)*, Rostock, Germany, September 2013.

[21] I. Dryanovski, R. Valenti and J. Xiao, Fast visual odometry and mapping from RGB-D data, in *Proc. IEEE/RSJ Int. Conf. Robotics and Automation (ICRA)*, Karlsruhe, Germany (2013), pp. 2305–2310.

[22] R. A. Brooks, Elephants don't play chess, *Robot. Auton. Syst.* **6** (1990) 3–15.

[23] M. J. Mataric, Behavior-based control: Examples from navigation, learning, and group behavior, *J. Exp. Theor. Artif. Intell.* **9** (1997) 323–336.

[24] T. Tomi, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa and D. Burschka, Toward a fully autonomous UAV: Research platform for indoor and outdoor urban search and rescue, *IEEE Robot. Autom. Mag.* **19**(3) (2012) 46–56.

[25] Available at: http://www.flyingmachinearena.org/videos/, retrieved on June 10, 2016.

[26] K. Schmid, T. Tomic, F. Ruess, H. Hirschmller and M. Suppa, Stereo vision based indoor/outdoor navigation for flying robots, *2013 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Tokyo, Japan (2013), pp. 3955–3962.

[27] S. Weiss, M. W. Achtelik, S. Lynen, M. C. Achtelik, L. Kneip, M. Chli and R. Siegwart, Monocular vision for long-term micro aerial vehicle state estimation: A compendium, *J. Field Robot.* **30**(5) (2013) 803–831.

[28] H. Lim, J. Park, D. Lee and H. J. Kim, Build your own quadrotor: Open-source projects on unmanned aerial vehicles, *IEEE Robot. Autom. Mag.* **19**(3) (2012) 33–45.

[29] S. A. Scherer and A. Zell, Efficient onbard RGBD-SLAM for autonomous MAVs, in *2013 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, Tokyo, Japan (2013), pp. 1062–1068.

[30] F. Fraundorfer and D. Scaramuzza, Visual odometry : Part II: Matching, robustness, optimization, and applications, *IEEE Robot. Autom. Mag.* **19**(2) (2012) 78–90.

[31] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige and W. Burgard, G2o: A general framework for graph optimization, in *Proc. IEEE/RSJ Int. Conf. Robotics and Automation (ICRA)*, Shanghai, China (2011), pp. 3607–3613.

[32] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese and C. Hertzberg, Hierarchical optimization on manifolds for online 2D and 3D mapping, in *Proc. IEEE/RSJ Int. Conf. Robotics and Automation (ICRA)*, Anchorage, AK, USA (2010), pp. 273–278.

[33] J. Bruce, T. Balch and M. Veloso, Fast and inexpensive color image segmentation for interactive robots, in *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)* (2000), pp. 2061–2066.

[34] B. Yamauchi, A. Schultz and W. Adams, Integrating exploration and localization for mobile robots, *Adapt. Behav.* **7**(2) (1999) 217–229.

[35] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss and W. Burgard, octoMap: A probabilistic, flexible, and compact 3D map representation for robotic system, in *Proc. IEEE/RSJ Int. Conf. Robotics and Automation (ICRA)*, Anchorage, AK, USA (2010).

[36] Available at: http://gazebosim.org/, retrieved on June 10, 2016.

[37] Available at: http://www.draganfly.com/, retrieved on June 10, 2016.

[38] Available at: http://www.ece.unb.ca/COBRA/quadrotor.htm, retrieved on July 3, 2016.