Perception and Navigation for Autonomous Rotorcraft

Sajad Saeedi[†], Amr Nagaty[†], Carl Thibault[†], Michael Trentini[⋆], and Howard Li[†]

***Abstract*—To enable a quadrotor to navigate autonomously, a set of interdependent requirements must be addressed, including control, state estimation, map learning, and mission planning. 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 paper proposes an integrated solution for all these requirements. The 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. Multiple tests, performed in simulated and real-world environments, show the effectiveness of the proposed solution.**

I. Introduction

Most robotic applications require reaching a desired location without human intervention, while taking into account all types of uncertainties. Generally, this is referred to as navigation. In an environment, unknown or partially known to a robot, navigation is more complicated. An autonomous robot needs to address two critical problems to survive and navigate within its surroundings: mapping the environment and finding its relative location within the map. Simultaneous localization and mapping (SLAM) is a process which aims to localize an autonomous mobile robot in a previously unexplored environment while constructing a consistent and incremental map of its environment. Correlation and dependency of localization and mapping on each other raise the complexity of the problem.

Most works on SLAM have focused on ground robots, where in most cases the motion of the robot is limited to the 2D plane. This becomes more complicated when SLAM is performed by a flying robot, such as a quadrotor rotorcraft. The added complexity derives from the inherent limitations of such robots. Limited payload, 3D motion, and lack of direct odometry are just a few examples of such limitations. Moreover, the quadrotor SLAM can be even more complicated if the robot is supposed to perform autonomous navigation on top of SLAM. In this paper, these challenges are investigated and an integrated solution for perception and navigation by an autonomous quadrotor is proposed.

A. Background of Research

This section presents background to rotorcraft perception. Different issues such as quadrotor, mapping, localization, and path planning are briefly reviewed.

1) Qudrotor: A quadrotor, also called quad rotorcraft, is a rotorcraft which 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. With the advances in electromechanical systems and sensing technologies, recently quadrotors have become popular in unmanned systems research. Their small size and maneuverability makes it possible to fly indoors and outdoors. Compared to other rotorcraft, they 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].

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

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

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

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, it may decide that the robot will explore an unknown environment, then it will return to the start point 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 level 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.

a) 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]. For more information on controlling a quadrotor, see [5] and [6].

[†]COBRA Group at the University of New Brunswick, Fredericton, Canada, http://www.ece.unb.ca/COBRA/, {sajad.saeedi.g, amr.nagaty, carl.t, howard}@unb.ca

[⋆]Defence Research and Development Canada-Suffield, Alberta, Canada, Mike.Trentini@drdc-rddc.gc.ca

b) 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 describes 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.

c) Map Learning: Deploying an autonomous robot in a real world scenario requires learning maps. Learning maps is different than just 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. 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 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 stack of mapping, localization, and path planning is often referred to as simultaneous planning, localization, and mapping (SPLAM); integrated autonomy solutions; or autonomy packages [7]. Complete information on different methods for components of map learning in relation to mapping and localization can be found in [8], and for path planning in [9].

B. Literature review

There has been an extensive amount of research on 2D perception and navigation specifically for ground robots [8]. In this section, the literature in relation to the quadrotor navigation is briefly reviewed.

1) Localization and Mapping: Thrun et al. [10], pioneered laser mapping by a low altitude flying helicopter. In their work a GPS, an IMU, a scanning laser rangefinder and a compass were used to perform laser mapping. The scan alignment is performed in a probabilistic framework. Unlike most other works in which scans are aligned in the $x-y$ plane, in their work scan alignment was done in the $x-z$ plane and in another step yaw and pitch angles were recovered.

In the Mikrokopter project [11], Grzonka et al. developed Monte-Carlo localization and graphSLAM for an indoor flying quadrotor. Their sensing platform includes a scanning laser rangefinder and an IMU. Generally, the IMU generates accurate roll and pitch angles, so in their work scans were projected onto a 2D plane using roll and pitch angles. Then by scan matching, x, y and yaw angles were estimated. Altitude estimates were provided by reflecting a few laser beams to the ground and then filtering with a Kalman filter. Their work is open-source. The work later was extended in [12] to be fully autonomous.

Bachrach et al. [13] developed a navigation system for a small quadrotor using a scanning laser rangefinder, a custom built stereo-camera rig, a color camera, and an IMU. Laser odometry is performed based on the work by Olson et al. [14] and visual odometry is done using FAST feature detection. Using an extended Kalman filter, odometry information is fused with the IMU. The filtered pose is used to provide localization and mapping using Gmapping [15]. The frontier exploration algorithm [16] is used to explore the unknown environment. The path to achieve waypoints is designed by dynamic programming in the information space [17].

Dryanovski et al. [18] developed a pose estimation system for a quadrotor micro air vehicle. In their method, three sensors are used: an IMU, a scanning laser rangefinder and a pressure altimeter. Given roll and pitch angles from the IMU, scans from the laser ranger are projected onto a 2D plane and a fast scan matching is used to perform 2D laser odometry [19]. A few laser beams are projected by a mirror to the ground to measure the altitude. The altitude measurement from the laser and the pressure altimeter are filtered by a Kalman filter. Gmapping [15] is also used to produce a 2D occupancy grid map. This work was developed under robot operating system (ROS) and is open-source.

In the Hector package [20], a fast solution for full 3D pose estimation was proposed. In this work, a scanning laser rangefinder and an IMU are the main sensors. The Hector package is flexible and was developed such that it can use GPS, compass, altimeter, and other sensors. The package was developed under ROS and is open-source. Their method has been tested in many different scenarios including unmanned ground vehicle (UGV) in rough terrains, unmanned surface vehicle (USV) in littoral waters, and a handheld embedded mapping system in indoor environments.

The autonomous urban search and rescue (USAR) platform [1] uses an Ascending Technologies Pelican quadrotor, equipped with a scanning laser rangefinder, an IMU and stereo cameras. The platform uses a canonical scan matcher by Censi [19] for the laser odometry and a correlation-based algorithm is used for the visual odometry. The results are fused by a Kalman filter. Three layers are defined for autonomous navigation: perception, which performs odometry and data fusion; cognition, which is responsible for path planning and controlling the mission; and action, which takes care of the controller. One experiment in a combined indoor and outdoor environment is presented to evaluate the proposed algorithm.

2) Path Planning: An important part of any autonomy package for a quadrotor is the motion or path planning. In most applications, the quadrotor is flown manually and the path is generated by a pilot. There exist a few papers developing a path planning algorithm for autonomous quadrotors. Generally path planning algorithms are categorized into two main groups: reactive and deliberative. Reactive methods such as bug algorithms, wavefront, and potential fields are fast and easy to implement while deliberate methods such as cell decomposition, A* search, and genetic algorithms are complicated and computationally expensive. The preference of the path planning method depends on the available onboard processing power and the level of the autonomy. Due to the ability of quadrotors to hover in one location and move in any direction with ease, path planning algorithms are generally designed in the x-y plane.

Bachrach et al. [13] have applied dynamic programming in the information space [17] to generated path. In [21], A* search on visibility graph and fast marching potential field were implemented. Grzonka [12] applied D* lite [22] which is a variant of the A* algorithm which utilizes previous results to improve an invalid plan. Vitus et al. [23] developed tunnel-mixed integer linear programming (MILP) path planning in obstacle-rich environments to increase autonomy of the Stanford testbed of autonomous rotorcraft for multi-agent control (STARMAC). In tunnel-MILP, first a path without any consideration of vehicle dynamics is generated. Then, a tunnel through which the robot travels is built as a sequence of convex polytopes. Finally, MILP is utilized to generate a dynamic trajectory which keeps the path of the robot inside the tunnel.

Whether the flight is autonomous or not, controlling the rotorcraft is an important task to achieve a successful flight. This task is out of the scope of this work and readers can find more information in [6], [24] and [3].

3) Contributions and Outline: The contribution of this work is an integrated autonomy solution. The proposed solution includes all necessary elements to have an autonomous quadrotor. 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 which need autonomy in GPS-denied environments.

The rest of the paper is organized as follows: Section II presents the autonomy solution for autonomous rotorcraft, Section III presents some experimental results in simulation and real-world environments, and Section IV makes some general conclusions and discusses future work.

II. Proposed Method

An overview of the proposed autonomy system is shown in Fig. 1. 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.

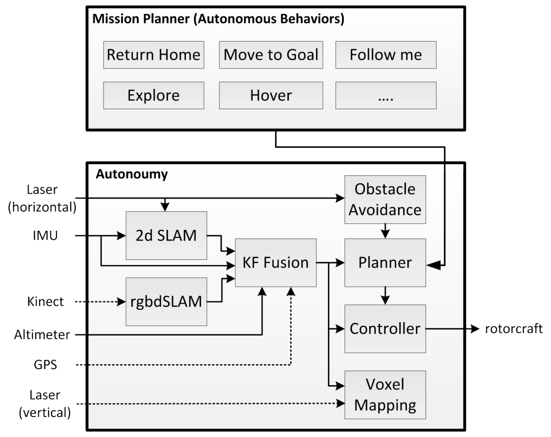

Fig. 1. Proposed autonomous navigation in GPS-denied environments is composed of two main modules: *Mission Planner* and *Autonomy*. *Mission Planner* takes care of organizing and sequencing the autonomous behaviors. The *Autonomy* block accepts these behaviors and takes proper actions. In the *Autonomy* block, an IMU, a horizontally mounted laser ranger, and an altimeter are minimum required sensors. These sensors are connected by solid lines to the related blocks. A vertically mounted laser ranger, a Kinect camera, and a GPS are optional sensors, connected by dotted lines to the related blocks. In the *2D SLAM* and *rgbdSLAM* blocks, view-based SLAM and feature-based SLAM are performed respectively. In the *KF Fusion* block, the results from the SLAM blocks are fused with the IMU, GPS, and the altimeter measurements to generate a pose estimate. The estimated pose is used by the *Planner* block to generate a waypoint. The waypoint is used by the *Controller* to fly the rotorcraft. The *Voxel Mapping* block, makes a 3D map using the estimated pose and the vertical laser ranger. The horizontal laser ranger is also used to avoid obstacles.

A. 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*, *return-home*, *move-to-goal* and *return-to-me* are based on path planning between two given points and following the path. *Obstacle avoidance* behavior 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 mentioned behaviors is presented in Fig. 2. 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 (*return-home* behavior).

Figure. 2 depicts the state machine of the navigation. Two behaviors, obstacle avoidance and hold (as an emergency stop), are running at a higher priority than others.

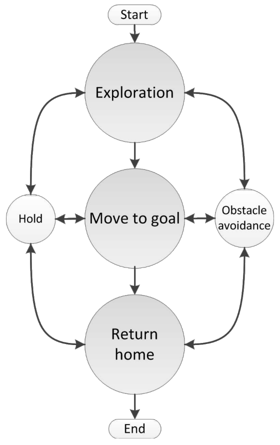

Fig. 2. A sample mission composed of basic navigation behaviors. First, the robot explores an unknown environment, then it moves to a given goal. Once it arrives at the goal, it returns home. While performing these behaviors, obstacle avoidance and hold (as an emergency stop) are running at a higher priority than others.

B. 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 [20] 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 $M(S_i(\delta))$, where $M(\cdot)$ represents the occupancy values from the occupancy grid map. Obviously, s_i should be an occupied cell, since it is the end point of a laser beam. Therefore, the map value at s_i should be compared with an occupied cell. This results in minimizing $(1 - M(S_i(\delta)))^2$, where 1 represents a fully occupied cell. By applying this to all beams, the transformation should minimize the difference between the current map and the new transformed scan as follows:

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

where n is the number of scan beams and δ^* is the desired outcome, which is the required transformation to fit the scan into the map. This problem is formulated and solved in [20] using the gradient ascent approach. To do this, first order Taylor expansion is applied to (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 [20]. The multi-resolution map representation can also be used for path planning and navigation purposes.

C. rgbdSLAM

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 rgbdSLAM algorithm.

rgbdSLAM, 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 [25].

D. State Estimation

Results from SLAM are fused with the information form 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

$$\mathbf{x} = \begin{bmatrix} \mathbf{r^T} & \mathbf{v^T} \end{bmatrix}^{\mathbf{T}}, \qquad (2)$$

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

$$\mathbf{r} = \begin{bmatrix} x & y & z \end{bmatrix}^T, \qquad (3)$$

$$\mathbf{v} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T. \qquad (4)$$

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

$$\dot{\mathbf{r}} = \mathbf{v} \qquad (5)$$

$$\dot{\mathbf{v}} = \mathbf{Ra} + \mathbf{g}, \qquad (6)$$

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. The GPS, altimeter, visual and laser updates are used for updating the state prediction.

E. 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 [16]. Algorithm 1 explains the frontier exploration. In line 1, set U which includes all frontier cells and set C which holds clusters of frontier cells are initialized to be empty. According to the algorithm, first frontier cells are extracted (line 2). Then the frontier cells are clustered based on the connectivity-8 neighborhood (line 3), where the connectivity-8 neighborhood means that all eight cells around a cell which touch either a corner or an edge of the cell, are considered as neighbors of the cell. In line 4, m gets the cardinality of set C (indicated by $|C|$). 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. This has the advantage that the robot does not show oscillatory behaviors between waypoints.

Algorithm 1 Frontier exploration.

```
Require: $m$: partially explored map,
    $(x_r, y_r)$: global coordinates of the robot
Ensure: $c_{wp}$: 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.

F. 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 [9]. 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 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 (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).

Algorithm 2 Wavefront path planning

```
Require: $m$: map, $q_{start}$: start cell, $q_{goal}$: goal cell, $r_{dilate}$:
    dilation size
Ensure: $p$: 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
```

G. Obstacle Avoidance

Obstacle avoidance is achieved using laser beams directly to avoid any unexpected collisions caused by disturbance or dynamic objects. Laser beams are filtered 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.

H. Voxel Mapping

The 3D voxel mapping is based on a compact and flexible 3D mapping, known as Octomap [26]. 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.

I. Controller

The controller block accepts waypoints generated by the planner block and adjusts rotor speeds. 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.

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, anytime, anywhere.

III. Experimental Results

To test the proposed perception and navigation solution, the mission shown in Fig. 2 based on autonomous behaviors 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. Videos of these experiments and a few additional experiments, including autonomous entry and exit, can be found in [27].

A. Demonstration in Simulation

The proposed solution for perception and navigation is tested in a simulated Gazebo world [28]. The simulated robot is designed in Blender and then integrated with Gazebo. The robot is equipped with a simulated IMU, a SODAR, and two Hokuyo UTM-30LX scanning laser rangers, mounted horizontally and vertically. Fig. 3-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 2 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. 3-a). The goal, which it attains after the exploration, is located 30 meters away, marked by a red disc.

To perform the complete mission, it took 210 seconds. In Fig. 3-b, the map developed by the horizontal laser ranger is shown. Fig. 3-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 [27].

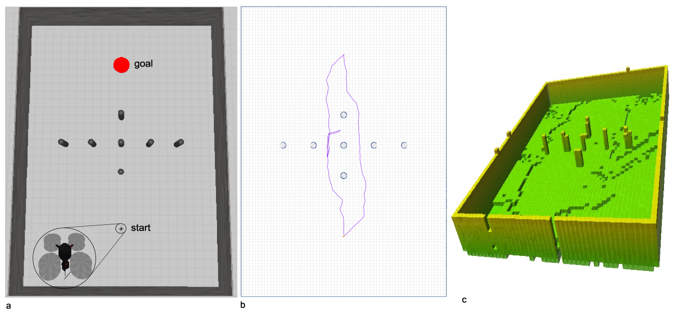

Fig. 3. Simulation in Gazebo. **a)** Simulation environment. Size of the world is 32×50 meters. Magnified simulated X8 is shown inside a circle with two perpendicular laser rangers mounted beneath the rotorcraft. **b)** Trajectory and 2D map.**c)** 3D voxel map.

Table I summarizes four performance indices for the mission. 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 I
Evaluating the experiment in Gazebo.

no	index	value
1	average position error	0.0065 (m)
2	average orientation error	0.0016 (rad)
3	error of achieving goal point	0.09 (m)
4	error of achieving start point	0.02 (m)

B. Demonstration on an Unmanned Ground Robot (UGV)

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 runtime 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.

This test was performed in a room with the approximate size 5×12 meters. Fig. 4-a shows the CoroBot in the test environment. The robot is equipped with one horizontal laser ranger, an IMU, and two encoders.

Fig. 4-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 [27]. All states of the mission are shown in the video.

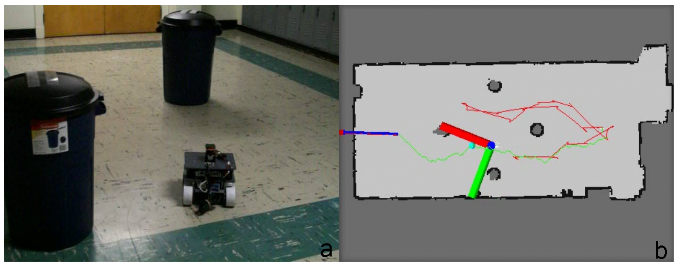

Fig. 4. Experiment with a CoroBot robot. **a)** The robot and the experimental environment. **b)** Trajectory and 2D map. The blue arrow shows the goal, the green curve shows the planned path and the red curve shows the trajectory of the robot. The perpendicular green and red bars show the body frame coordinates. The small cyan ball on the planned path shows the waypoint to be followed by the robot.

C. Demonstration on a Quadrotor Rotorcraft

The last real-world experiment was performed with the custom-built COBRA quadrotor. Fig. 5 shows the quadrotor in the test environment.

The test environment is approximately $94.7m^2$ and two boxes are placed in the environment as obstacles. According to the mission plan, shown in Fig. 2 and starting from the point shown by a black circle in Fig. 5-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.

The whole mission took about $320sec$, with an average speed of $0.25m/s$. Fig 5-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 [27]. The robot successfully completed the mission by exploring, mapping, and navigating through obstacles.

Fig. 6 shows the path of the robot for each stage of the mission. Starting from the 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. 7-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. 7-b shows an enlarged section of the error enclosed by a red rectangle in Fig. 7-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.

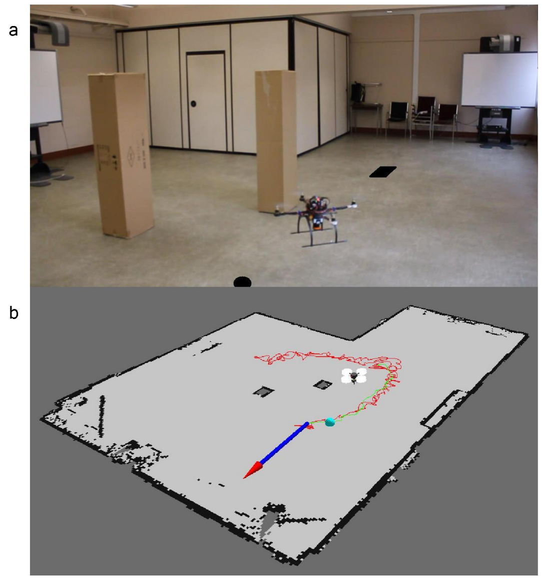

Fig. 5. Experiment with COBRA quadrotor. This experiment, performs the three-stage mission depicted in Fig. 2: *exploration*, *move-to-goal*, and *return-home*. **a)** This figure shows the test environment. The robot does not know the environment in advance. The black circle marks the start point. The robot should explore the world, and then move to a goal (shown by a black square) and then return to the start point. **b)** A snapshot of the developed map and trajectory of the robot. The robot is returning home, shown by an arrow. A path has been generated and the robot is following it. The cyan sphere indicates a waypoint along the path to be followed. The robot successfully mapped the environment and navigated through obstacles and completed the mis

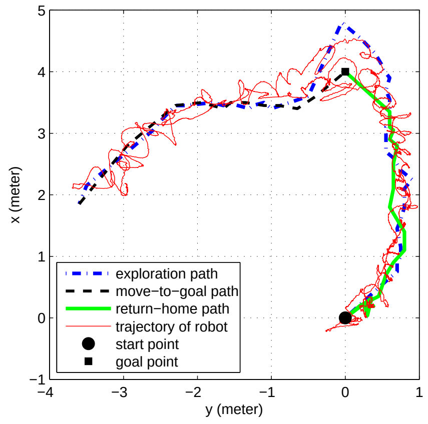

Fig. 6. Paths and trajectory of the robot for each stage of the three-stage mission depicted in Fig. 2: *exploration*, *move-to-goal*, and *return-home*. The start point is shown by a circle. First the quadrotor explores the environment. The desired path for the exploration is shown in blue. Then it goes to the goal point marked by a square (*move-to-goal*). Once it reached the goal, it returned home.

Table II 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.

IV. 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:

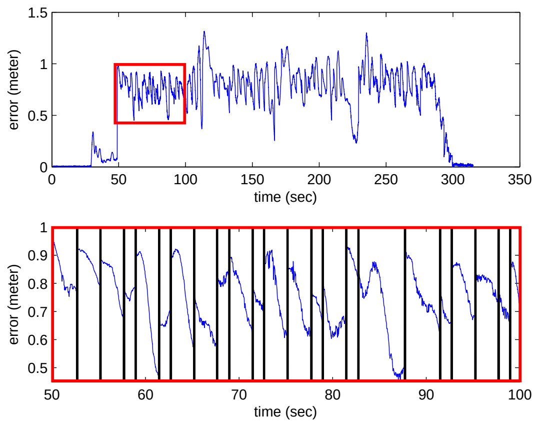

Fig. 7. Performance of waypoint following. **a)** Waypoint following error for the whole trajectory shown in Fig. 6 **b)** Enlarged section of the waypoint following error enclosed in a red rectangle in Fig. 7-a. The vertical solid black lines indicate the times when a new waypoint is received. The quadrotor tries to follow the waypoint.

TABLE II
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)

mapping, localization, and path planning. A behavior based mission control was proposed to plan, organize, and sequence different flight behaviors. 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.

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

ACKNOWLEDGEMENT

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

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," *Robotics Automation Magazine, IEEE*, vol. 19, no. 3, pp. 46–56, September 2012.

[2] I. Palunko, P. Cruz, and R. Fierro, "Agile load transportation : Safe and efficient load manipulation with aerial robots," *Robotics Automation Magazine, IEEE*, vol. 19, no. 3, pp. 69–79, September 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, "Pidd vs lq control techniques applied to an indoor micro quadrotor," in *Intelligent Robots and Systems (IROS), Proceedings of the IEEE/RSJ International Conference on*, 2004, pp. 2451–2456.

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

[6] 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.

[7] C. Stachniss, *Robotic Mapping And Exploration*. Springer Tracts in Advanced Robotics.

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

[9] 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, Jun. 2005.

[10] M. D. S, Thrun 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 *In IEEE International Conference on Robotics and Automation(ICRA)*, 2009, pp. 2878–2883.

[12] ——, "A fully autonomous indoor quadrotor," *IEEE Transaction on robotics*, vol. 28, no. 1, pp. 90–100, February 2012.

[13] 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, December 2009.

[14] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *in Proc. of the IEEE Int. Conference on Robotics and Automation (ICRA)*, 2006, pp. 2262–2269.

[15] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, pp. 34–46, 2007.

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

[17] R. He, S. Prentice, and N. Roy, "Planning in information space for a quadrotor helicopter in a gps-denied environment," in *In IEEE International Conference on Robotics and Automation(ICRA)*, 2008, pp. 1814–1820.

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

[19] A. Censi, "An icp variant using a point-to-line metric," in *Proceedings of the IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2008.

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

[21] G. M. Hoffmann, S. L. Wasl, and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," in *In Proc. AIAA Guidance, Navigation, and Control Conf*, 2008.

[22] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transaction on robotics*, vol. 21, no. 3, pp. 354–363, June 2005.

[23] M. Vitus, V. Pradeep, G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Tunnel-MILP: Path planning with sequential convex polytopes," in *2008 AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, USA, August 2008.

[24] S. Bouabdallah, "design and control of quadrotors with application to autonomous flying," Ph.D. dissertation, Ecole Polytechnique Federale De Lausanne (EPFL), Lausanne, Switzerland, 2007.

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

[26] K. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "octomap: A probabilistic, flexible, and compact 3d map representation for robotic system," in *In IEEE International Conference on Robotics and Automation(ICRA)*, 2010.

[27] S. Saeedi, A. Nagaty, C. Thibault, M. Trentini, and H. Li, "Perception and navigation of autonomous rotorcraft," accessed 19 July 2013. [Online]. Available: http://www.ece.unb.ca/COBRA/quadrotor.htm

[28] N. Koenig, J. Hsu, M. Dolha, and A. Howard, "Gazebo," accessed 19 July 2013. [Online]. Available: http://gazebosim.org/