Self-Contained Ranging Sensor Aided Autonomous Guidance, Navigation, and Control for Indoor Flight

Girish Chowdhary*, D. Michael Sobers, Jr.†, Chintasid Pravitra‡,
Claus Christmann §, Allen Wu‡, Hiroyuki Hashimoto¶,
Chester Ong §, Roshan Kalghatgi ‖ and Eric N. Johnson**
Georgia Institute of Technology, Atlanta, GA, 30332-0152, USA

This paper describes the design and flight test of a completely self-contained autonomous indoor Miniature Unmanned Aerial System (M-UAS). Guidance, navigation, and control algorithms are presented, enabling the M-UAS to autonomously explore cluttered indoor areas without relying on any off-board computation or external navigation aids such as GPS. The system uses a scanning laser rangefinder and a streamlined Simultaneous Localization and Mapping (SLAM) algorithm to provide a position and heading estimate, which is combined with other sensor data to form a six degree-of-freedom inertial navigation solution. This enables an accurate estimate of the vehicle attitude, relative position, and velocity. The state information, with a self-generated map, is used to implement a frontier-based exhaustive search of an indoor environment. Improvements to existing guidance algorithms balance exploration with the need to remain within sensor range of indoor structures such that the SLAM algorithm has sufficient information to form a reliable position estimate. A dilution of precision metric is developed to quantify the effect of environment geometry on the SLAM pose covariance, which is then used to update the 2-D position and heading in the navigation filter. Simulation and flight test results validate the presented algorithms.

I. Introduction

Autonomous indoor reconnaissance and surveillance can bring key capabilities in both civilian and military applications. Soldiers can use Miniature Unmanned Aerial Systems (M-UAS) to negotiate cluttered and confined areas without risking human life. This technology can also bring increased capabilities for disaster management and monitoring in confined urban spaces. M-UAS are ideal candidates for such missions as they can use three dimensional maneuvers to overcome obstacles that prevent the use of ground vehicles. Indoor area exploration with ground robots that combine vision or scanning range sensors with Inertial Measurement Units (IMUs) has been previously studied (see e.g. Refs. 1,2,3). However, significant technological challenges exist in order to ensure reliable operation of M-UAS in such environments. First, the M-UAS must be sufficiently small in order to successfully maneuver in cluttered indoor environments, consequently limiting the amount of computational and sensory power that can be carried onboard. This constraint on onboard computational power is further exacerbated by the fact that the control of miniature rotorcraft platforms requires fast and accurate angular rate and pose information (pose is defined here as the combination of position and attitude). Particularly, unlike in the case of ground robots, the onboard controller must always be active and the vehicle cannot operate slowly in a stop-and-go manner. Typical odometry methods such as wheel rotation are also useless for flying vehicles. Further, it is well known that

*Research Engineer II, School of Aerospace Engineering
†Major, USAF. Graduate of the School of Aerospace Engineering. The views expressed in this paper are those of the authors and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the U.S. Government.
‡Graduate of the School of Aerospace Engineering
§Graduate Research Assistant, School of Aerospace Engineering
¶Graduate Student, School of Aerospace Engineering
‖Graduate Student, School of Mechanical Engineering
**Lockheed Martin Associate Professor of Avionics Integration, School of Aerospace Engineering

integrating forward in time the acceleration and angular rate measurements from strapdown accelerometers and gyroscopes without correcting for initial misalignment errors and sensor biases is not feasible as the attitude and velocity estimates tend to drift rapidly (see e.g. Ref. 4). Traditionally for UAS, this information is obtained by fusing inertial measurements with GPS-based absolute position information (see e.g. Refs. 5, 6). As a result, most current algorithms for UAS guidance, navigation, and control rely heavily on GPS signals, and hence are not suitable for indoor navigation where GPS is normally not available. Finally, the M-UAS should be designed as an expendable unit for operation in potentially dangerous environments; hence low-cost, lightweight designs need to be explored. These restrictions pose significant technological challenges for the design of reliable M-UAS platforms capable of navigating in cluttered areas in a GPS-denied environment.

This paper describes Guidance, Navigation, and Control (GNC) algorithms and their implementation on an M-UAS capable of exploring cluttered indoor areas without relying on external navigational or computational aids such as GPS. In this paper, vehicles that are human-portable with maximum dimensions between 30 cm and 2 m are considered "miniature", where smaller vehicles are considered "micro" UAS.[7] As shown in Figure 1, the system consists of an air vehicle, a ground station, and a safety pilot interface for manual flight. The vehicle, referred to as the GTQ, is capable of completely self-contained autonomous indoor flight. A ground station computer is used only to monitor vehicle health and status, to view the vehicle-generated map, and to interact with the GTQ; it does not perform any GNC-related computation. An off-the-shelf quadrotor platform is equipped with a low-cost off-the-shelf avionics package, on which the presented algorithms are implemented. The navigation algorithm described here uses a nonlinear process model to fuse information from a scanning laser range sensor, an IMU, and a sonar altitude sensor to form an accurate estimate of the vehicle attitude, velocity, and position relative to indoor structures through an Extended Kalman Filter (EKF) framework while simultaneously mapping the environment. A streamlined Simultaneous Localization and Mapping (SLAM) routine is implemented to provide position updates to the navigation software. The SLAM position estimates rely on the presence of features in the environment, and since good position estimates are required for accurate navigation and mapping, a compact exploration strategy is developed to ensure the vehicle maintains a trajectory that keeps it within sensor range of indoor features. As described below in more detail, the control architecture uses a linear cascaded inner-outerloop structure with an integrator element.

An important aspect of incorporating any measurement into the navigation state estimate is the uncertainty associated with the measurement. A typical SLAM algorithm, such as CoreSLAM, provides a three-state pose measurement consisting of the 2-D position with respect to the environment and a heading estimate.[8] Similar routines can also usually provide the "fit quality" of the scan measurement compared to the stored map. However, this and other similar algorithms fail to identify the pose uncertainty due to the *structure* of the environment. For example, a vehicle in a long, straight hallway may have a good estimate of its position with respect to the walls on each side, but have very little certainty of its position along the length of the hallway. Similarly, a vehicle in the middle of a circular room might have a good position estimate, but a poor estimate of its heading using only the scan data unless additional feature-tracking methods are used. In these situations, a scan-matching algorithm may produce a fit that has very little error, while at the same time providing very little information in certain directions. This Dilution of Precision (DOP) problem results in a pose estimate that quickly becomes overconfident in directions where few features are observed. To solve this problem, a metric is developed and described below to identify the uncertainty in the relative 2-D position and heading (x, y, and ψ) based on the orientation and distance to line segments extracted from the scan data.

SLAM algorithms that combine IMU and vision sensors have been previously implemented for outdoor aerial vehicles (see e.g. Refs. 9, 10), however the position estimate obtained using these methods was not suitably accurate for indoor operation. Algorithms that use a laser scanner to implement SLAM have also been previously implemented for indoor air vehicles (see e.g. Refs. 11, 12, 13). In those implementations, information was transmitted to ground computers which performed the required SLAM computations and relevant GNC computations. The key difference here is that the developed GNC algorithms, along with SLAM, are specifically designed to be implemented entirely using the limited computational resources of onboard embedded computers, without requiring any off-board computation. A fully onboard approach was demonstrated by Shen et al in 2011, however that system uses a simple scan matching routine for pose estimation, and no details are provided on how pose estimation error is determined.[14] The GTQ, which first flew autonomously in August 2010, is unique in that it requires no external navigational or computational aids

to accomplish indoor exploration, and it uses SLAM-based pose estimation, with statistically appropriate covariance, as a position update to the navigation filter. The reliance on onboard computation makes the presented approach resilient to loss of data link, which can be unreliable in cluttered environments. Furthermore, it avoids information bottlenecks, thereby making the approach scalable to multiple vehicles by reducing the ground support equipment necessary for operation. The guidance algorithm described here and a preliminary description of the system has appeared in conference papers.[15,16,17] The new contributions in this paper include detailed descriptions of the guidance, navigation and control system, and a discussion of the DOP metric for characterizing the influence of environment topology on SLAM-based pose estimates. Finally, this paper also presents and discusses flight test results of the integrated GNC algorithms implemented on the GTQ.

A discussion of the vehicle platform and the avionics suite employed is presented in Section II. The details of the guidance, navigation, and control algorithms are presented in Sections III, IV, and V respectively. Simulation and flight test results are presented in Section VI, and the paper is concluded in Section VII.

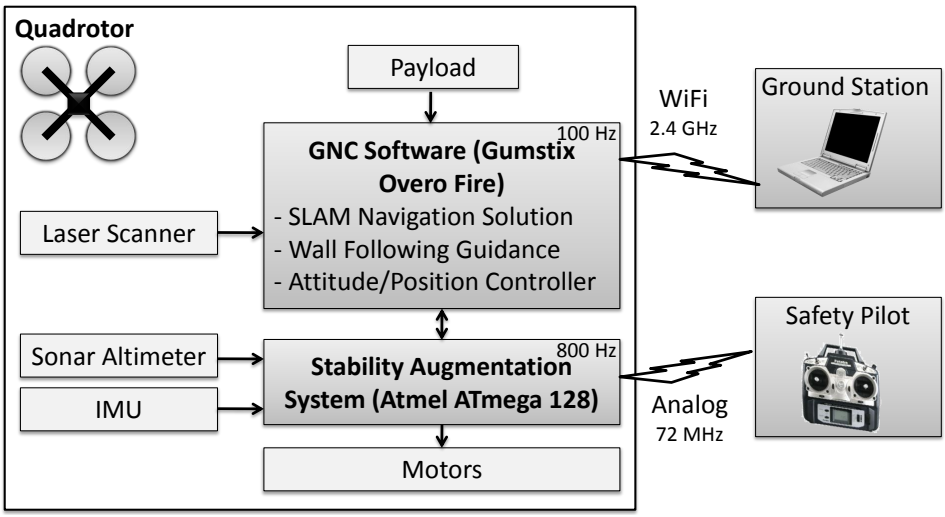
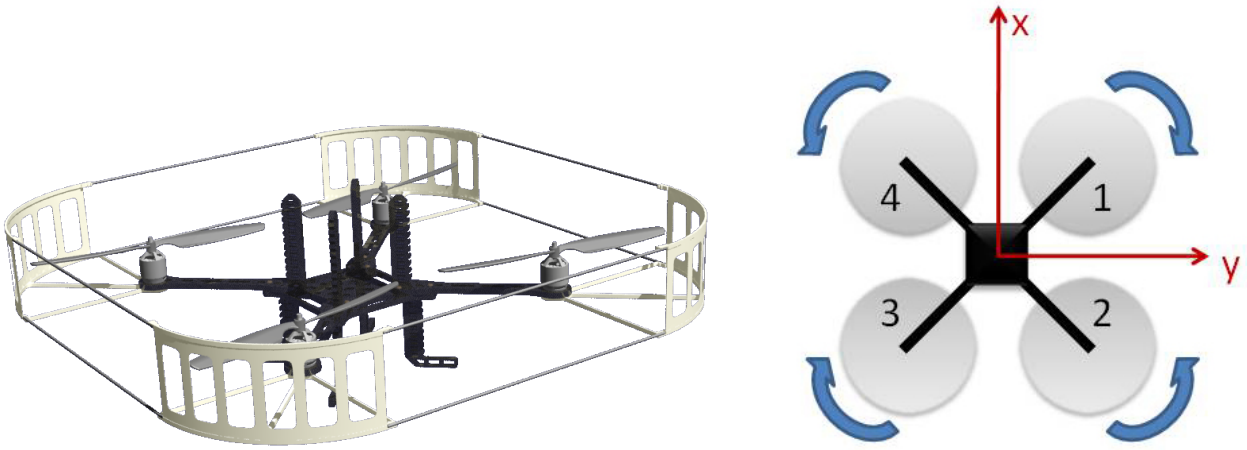


Figure 1. The GTQ system architecture is designed to fuse information onboard from a scanning laser, sonar altimeter, and an Inertial Measurement Unit (IMU). The IMU and the sonar altimeter sensors are read by the Atmel ATmega microcontroller, which also serves as a rate-damping stability augmentation system. The scanning laser rangefinder is directly read by the Gumstix Overo Fire onboard processor, which fuses information from all sensors to form a six degree-of-freedom navigation solution onboard. This information is used along with a frontier guidance exploration technique and a dynamic inversion controller to autonomously explore cluttered GPS denied indoor areas. The ground station and safety pilot are not required for autonomous flight.

II. Description of Vehicle and Avionics

The AscTec Pelican quadrotor, made by Ascending Technologies GmbH, was selected as the base airframe for the GTQ (see Figure 2(a)). The vehicle structure, motors, and rotors of the AscTec Pelican were used without modification. The vehicle generates lift using four fixed-pitch propellers driven by electric motors. Control is achieved by creating a relative thrust and torque difference between the propellers to effect pitching, rolling, and yawing motion. Quadrotors can either be flown in a diamond configuration (one motor in the front, right, back, and left corners) or a square configuration (two motors in the front, and two motors in the back). Although many quadrotors in the aerial robotics community fly with a diamond configuration (see e.g. Refs. 18, 19, 20), the square configuration (shown in Figure 2(b)) was selected for this research to allow the vehicle to go through smaller openings without changing orientation during flight. A detailed simulation model of the GTQ was developed (see Ref. 17) using rigid body dynamics, accounting for aerodynamic forces, and approximating motor dynamics with a second order system, in a manner similar to the implementation found in Refs. 21, 20, 18, 19.

The GTQ uses three primary measurement sensors for navigation, stability and control: a laser range



(a) AscTec Pelican Platform (b) Quadrotor Flying in Square Configuration

Figure 2. The GTQ uses the Ascending Technologies Pelican for the aerial platform.

finder, a sonar altimeter, and an IMU. The laser range finder used is the Hokuyo URG-04LX-UG01. It is capable of measuring distances up to 4 m over a 240-degree field of view, with a resolution of 1 mm and 0.36 degrees respectively. The sonar altimeter used is the MB1040 LV MaxSonar EZ4 ultrasonic range finder. It is capable of measuring distances up to 6.45 m away with resolution of 25.4 mm. The IMU is the ADIS-16365-BMLZ built by Analog Devices Inc. It consists of a three-axis digital gyroscope and a three-axis accelerometer that can measure forces up to ±18 g. These sensors are integrated with the Gumstix Overo Fire onboard computer, which is a small and cost-effective ARM Cortex-A8 OMAP3530 computer-on-module, by using the Atmel ATmega microcontroller as an interface (see Figure 1). The Gumstix computer is equipped with 256 MB Flash RAM, and it can communicate using UART, SPI, and I2C interfaces as well as 802.11g and Bluetooth wireless links. Two three-cell lithium polymer battery packs are used: one drives the motors to provide lift, and the other powers the onboard computer.

III. Guidance Algorithm

Indoor navigation employing SLAM is by its nature based on measurements of local features. As a result, any guidance system reliant on SLAM for navigation must use information found in the local environment. This section describes a compact guidance strategy for exploration that performs an exhaustive search along a path that stays close to walls in an effort to keep the scanning laser rangefinder within range of the features needed for the SLAM algorithm to work. The implementation of SLAM and path-planning algorithms on ground robots is a well-studied problem.[22,23,24] However, in extending these methods to M-UASs, new challenges are encountered. Hovering aircraft have more stringent power and weight constraints; vehicle dynamics are faster and often unstable; and motion spans six degrees of freedom. In fact, many well-known optimal guidance and path planning techniques are not currently feasible for onboard implementation on M-UASs because of limited computational power. As a result, in contrast to many published works on guidance and path planning, the following approach does not focus on optimal solutions for exploration. Instead, it emphasizes an efficient algorithm that can quickly determine the location of unexplored areas while keeping the vehicle within sensor range of geographic features.

SLAM with autonomous exploration using an M-UAS has been demonstrated by Achtelik et al.[12] Achtelik et al. use frontier-based exploration with goal-driven dynamic programming trajectory generation. However, such navigation and guidance algorithms are computationally expensive. Sobers et al. have previously developed a totally self-contained M-UAS architecture with a very compact SLAM algorithm and a simple wall following guidance strategy.[15] However, wall-following guidance alone does not place higher weight on unexplored areas. In some cases, the method may only track the outer walls of a building and may leave inner rooms completely unexplored. In other cases, unfavorable geometry may cause the vehicle to stay in the same room or avoid certain rooms altogether. The guidance algorithm described here is designed to improve upon simple wall-following logic by introducing an efficient global book-keeping feature, while

keeping the algorithm simple enough to run alongside the SLAM algorithm without over-taxing the onboard flight computer.

Frontier-based exploration can provide a way to "remember" what parts of the map have been previously visited. The method was first introduced by Yamauchi[25] as an effective way for a mobile robot to explore an unknown environment. Without frontier-based exploration, a robot may have to explore an unknown environment randomly with some form of obstacle avoidance logic. The principle of frontier-based exploration can be simply stated: *"try to get as much new information as possible by going to a boundary between explored and unexplored territory"*. Various forms of frontier-based exploration have been developed, most of which require some form of global map in order to find frontiers and plan trajectories.[25,26,27] A global map can be grid-based, feature-based, or polygonal-based. Unlike the SLAM problem, which stores a global map but does scan-matching only against local features, a map-based guidance system often requires nonlinear optimization of multiple degrees of freedom. As a result, guidance algorithms utilizing such a global map are typically not computationally practical onboard an M-UAS.

Rather than using a global map, Freda and Oriolo applied the guidance principle of frontier-based exploration to a data structure called a Sensor-Based Random Tree (SRT).[28,29] This guidance system uses an SRT method called SRT-Star to store frontiers and safe-regions, and to sequence new waypoints.[28] In this research, the SRT-Star method is improved by the addition of a wall-following algorithm to ensure that the vehicle keeps close to walls, thereby increasing the chance of being in a geometry that is favorable for the SLAM navigation routine.

Assuming near-hover dynamics and a semi-structured environment, first note that the three-dimensional guidance problem can be simplified by decoupling level flight guidance from altitude guidance. Here, the altitude guidance algorithm is simply a command to maintain a constant altitude above ground level. Independent from altitude, the level flight guidance algorithm commands horizontal velocity and heading. While the heading command (ψ_{cmd}) is generated solely from the location of frontiers, the horizontal velocity command ($\mathbf{v}_{cmd}$) is composed of a contribution from the wall-following guidance algorithm ($\mathbf{v}_{wf}$) and the frontier guidance algorithm ($\mathbf{v}_{fr}$). Thus, the total commanded velocity is expressed by the relationship $\mathbf{v}_{cmd} = \mathbf{v}_{wf} + \mathbf{v}_{fr}$. The commanded velocity and heading are determined by using data from the laser rangefinder to determine exploration waypoints, which are stored in an SRT that is updated upon receipt of new scan data at a rate of 10 Hz.

Algorithm 1 illustrates the sequence of commands that are executed at each update time step. The main algorithm requires the vehicle's current position ($\mathbf{x}$), the position of the commanded waypoint ($\mathbf{x}_{waypoint}$), laser scan data ($scan$), the sensor based random tree structure (SRT), and a threshold distance (d).

Algorithm 1 Compute Velocity Command ($\mathbf{v}_{cmd}$) and Heading Command (ψ_{cmd})

Require: $\mathbf{x}$, $\mathbf{x}_{waypoint}$, $scan$, SRT, and d
1: $\mathbf{v}_{wf} \Leftarrow getWallFollowingVelocity(scan)$
2: **if** $||\mathbf{x} - \mathbf{x}_{waypoint}|| < d$ **then**
3: $newFrontier \Leftarrow frontierSearch(scan)$
4: $SRT \Leftarrow updateSRT(SRT, newFrontier)$
5: **if** $frontierExist(SRT)$ **then**
6: $\mathbf{x}_{waypoint} = newWaypoint(SRT)$
7: **else**
8: $\mathbf{x}_{waypoint} = previousWaypoint(SRT)$
9: **end if**
10: **end if**
11: $\mathbf{v}_{fr} \Leftarrow getFrontierVelocity(\mathbf{x}, \mathbf{x}_{waypoint})$
12: $\psi_{cmd} \Leftarrow getHeading(\mathbf{x}, \mathbf{x}_{waypoint})$
13: $\mathbf{v}_{cmd} \Leftarrow \mathbf{v}_{wf} + \mathbf{v}_{fr}$

The algorithm begins by computing the wall-following velocity component ($\mathbf{v}_{wf}$) directly from the scan data. The purpose of the wall-following velocity is to create an obstacle avoidance potential field while keeping within sensor range of observable features in the environment. Let r_i denote the i^{th} range measurement shown in Figure 3. The incremental velocity command (v_{ri}) in the radial direction for each scan point is obtained

from:

$$v_{ri} = \begin{cases} K_{wf}(r_i - r_t) & \text{if } r_i \geq r_{safe}; \\ K_{safe}(r_i - r_t) & \text{if } r_i < r_{safe}; \end{cases} \tag{1}$$

where K_{wf} and K_{safe} are gains that are chosen to be used for nominal flight and obstacle avoidance (safe) flight as determined by a chosen safe radius r_{safe}. Furthermore, r_t is a user-specified parameter representing distance from walls that the vehicle tries to maintain.[15]

In this paper, two standard aerospace right-handed coordinate systems are defined: the inertial frame (denoted by the preceding superscript i) is a local frame that is aligned to the local north, east, and down directions; the body frame (denoted by the preceding superscript b) is aligned such that the x axis points forward and the z axis points downward.[4] The orthonormal rotation matrix operator $L_{b\to i}$ transports vectors from the body to the inertial frame.[10]

Let θ_i shown in Figure 3 denote the angle of the i^{th} range measurement with respect to body frame $(\mathbf{x}, \mathbf{y})$, and let n denote the number of in-range scans points. The wall-following velocity command in the body frame is calculated by projecting each v_{ri} into the body frame and summing over n as shown in (2).

$$^{b}v_{x_{wf}} = \sum_{i=1}^{n} v_{ri}\cos(\theta_i) \qquad ^{b}v_{y_{wf}} = \sum_{i=1}^{n} v_{ri}\sin(\theta_i) \tag{2}$$

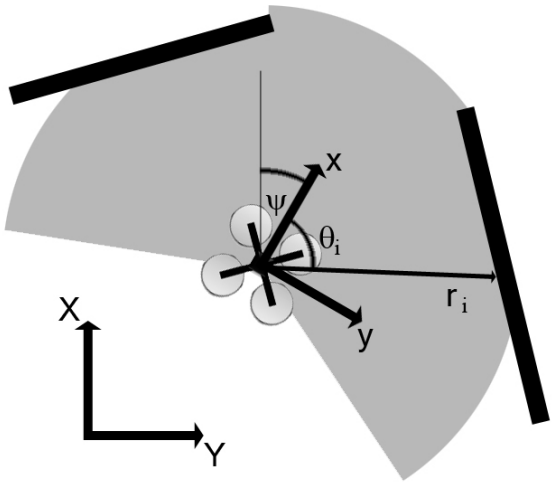


Figure 3. Reference frame description of vehicle and scan points

Let ψ be the heading angle referenced from an arbitrary inertial frame $(\mathbf{X}, \mathbf{Y})$. The velocity command in the body frame can be converted to the inertial frame using (3).

$$\mathbf{v}_{wf} = \begin{bmatrix} ^{i}v_{x_{wf}} \\ ^{i}v_{y_{wf}} \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) \\ \sin(\psi) & \cos(\psi) \end{bmatrix} \begin{bmatrix} ^{b}v_{x_{wf}} \\ ^{b}v_{y_{wf}} \end{bmatrix} \tag{3}$$

The overall effect is attraction to a wall if the vehicle is too far away, and repulsion from a wall if the vehicle is too close. In cases where multiple walls are in sensor range, the resultant velocity depends on wall geometry and range. For instance, the vehicle will be attracted towards a wall further away compared to a closer wall of the same length. Note that this implicitly promotes exploration.

The algorithm then checks the distance to the commanded waypoint ($\mathbf{x}_{waypoint}$). If the vehicle has not arrived at the commanded waypoint, the waypoint is not modified. If the vehicle has arrived at the commanded waypoint, a new waypoint is generated by the frontier planner (Algorithm 1, lines 3 through 8). The vehicle is considered to have arrived at the commanded waypoint when it is within distance d of the commanded waypoint. Finally, the commanded waypoint and vehicle's 2-D pose are used to generate the heading command (ψ_{cmd}) and frontier velocity component ($\mathbf{v}_{fr}$). Notice that although the commanded waypoint may not change at a particular time step, the frontier velocity and heading change at every time

step due to changes in the vehicle's position.

SRT-Star, as outlined in lines 2 through 10 of Algorithm 1, uses the 2-D laser scanner measurements for its frontier search.[28] Upon arriving at a waypoint, SRT-Star divides the laser scan data into sectors. Each sector can have up to three frontiers: a left-point, a mid-point, and a right-point. The mid-point of a sector is marked as a frontier when there is absolutely no scan return from all scan points in that sector. The left-point of a sector is marked as a frontier if there is a large difference between the smallest scan value of the sector and the smallest scan value of the left adjacent sector. The right-point of a sector is marked as a frontier if there is a large difference between the smallest scan value of the sector and the smallest scan value of the right adjacent sector. A sector containing at least one frontier is considered unexplored. Sectors and frontier points are illustrated in Figure 4. If at least one frontier exists, a new commanded waypoint is generated by randomly picking a point in the middle of an unexplored sector. If no frontiers exist, the vehicle backtracks to the previous waypoint. Further details of SRT-Star are discussed in Ref. 28.

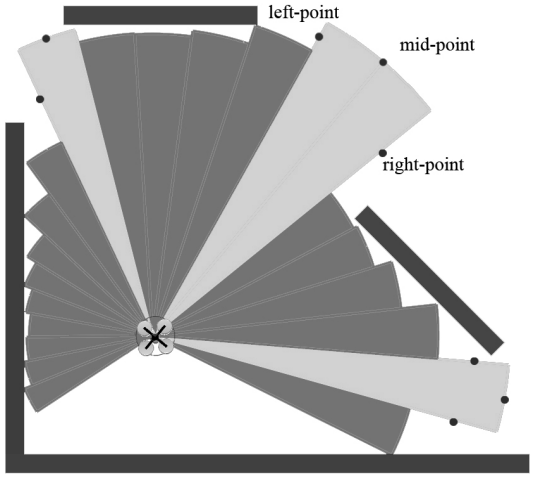


Figure 4. SRT-Star divides laser a scan into sectors. Here, the sectors are visualized by displaying the minimum range detected in each sector. The dots are frontiers marked by the algorithm. Sectors with frontiers, colored light gray, are considered unexplored. The frontiers are stored in the SRT, and a commanded waypoint is chosen randomly near the center of an unexplored sector.

One difference between SRT-Star as discussed by Freda et al.[28] and the method described here is that the original SRT-Star algorithm feeds the commanded waypoint directly to the controller. The method outlined in Algorithm 1 calculates a frontier velocity command and a heading command based on the commanded waypoint. It then blends the frontier velocity command with the wall-following velocity command, with the resultant velocity and heading commands passed to the controller.

While the wall-following velocity command is described in (1-3) above, the frontier velocity command is calculated using the simple proportional feedback law shown in (4). Here, K_{fr} is the frontier velocity gain specified by user, the commanded waypoint in the inertial frame is given by $x_{waypoint}$ and $y_{waypoint}$, and the vehicle's current position in the local inertial frame is given by x and y.

$$\mathbf{v}_{fr} = \begin{bmatrix} {}^{i}v_{x_{fr}} \\ {}^{i}v_{y_{fr}} \end{bmatrix} = \begin{bmatrix} K_{fr}(x_{waypoint} - x) \\ K_{fr}(y_{waypoint} - y) \end{bmatrix} \tag{4}$$

The heading controller is a simple proportional law that points the vehicle toward the commanded waypoint. Note that this helps complete the map by steering the scanner field of view toward the neighborhood of the frontier containing the waypoint. In addition, the wall-following velocity component will command attraction to any walls in the vicinity, which also implicitly brings the vehicle towards the commanded waypoint.

To improve flight safety, the total commanded velocity and the yaw rate are limited by the onboard software. Furthermore, a time-out is implemented between any two waypoints. This is particularly useful to counter drift of the inertial frame due to imperfections in the SLAM-based navigation solution, which

can result in waypoints being placed in areas that are inaccessible without violating safe wall-distance parameters. The following constant values were used for algorithms in this section to achieve the flight test results presented below in Section VI: $K_{wf} = 0.001\ s^{-1}$, $K_{safe} = 0.3\ s^{-1}$, $r_t = 1.22\ m$, $r_{safe} = 0.61\ m$, $K_{fr} = 0.1\ s^{-1}$.

IV. Navigation Algorithm

This section describes a navigation system that fuses IMU measurements, sonar altimeter measurements, and SLAM-based pose estimates using an EKF-based navigation algorithm to produce accurate vehicle state estimates in GPS denied environments. A novel method for determining the uncertainty of laser range measurements relative to the topology of the environment is also presented.

A. Dilution of Precision

A typical scanning laser rangefinder performs its measurements by panning a laser across the environment and sampling the return at fixed angular increments. The result is a series of 1-D range readings recorded at consecutive, known angles. Although the error associated with an individual laser range measurement is a function of many parameters, such as target color and surface finish, only the effect of range to target is readily known during real-time operation. As a result, it is common practice to estimate the range error as a function of the measured range, neglecting all other sources of error.[30,31] However, if the local topology is also considered, more information about the measurement uncertainty can be determined.

The topology of the local environment measured by the scanning laser affects the accuracy of the pose estimate created when matching scans against a reference scan or a map. In essence, the vehicle position estimate is most accurate when range is measured perpendicular to the local environment. As such, a long featureless wall or hallway only provides position information perpendicular to the walls, and no information parallel to the walls. Thus, any position estimate using scans in this environment will have a strong error correlation in the x and y directions. This correlation is not detected by scan matching routines–the shape of the environment must be considered. Likewise, a good heading estimate requires that range readings be different in different directions such that slight changes in heading can be detected. Hence, straight walls provide a good basis for measuring heading, while concave curved surfaces do not. Any pose estimation routine that uses an error metric based on scan match quality rather than instrument accuracy and environment topology ignores this phenomenon. In order to produce a pose estimate with statistically appropriate confidence, DOP must be considered.

The information contained in a laser scan can be calculated by combining the information provided by each range measurement to form the *information matrix*. An inverse form of the Kalman filter, called the *information filter*, utilizes the information matrix as defined in (5).[32] In this formulation, the *information* contained in each measurement is the inverse of the variance of the measurement, transformed by a measurement matrix, $\mathbf{H}$. When using SLAM pose estimation in-the-loop, the *measurement* provided to the EKF navigation algorithm is the pose estimate (x, y, ψ), not the individual laser range measurements. The measurement matrix thus describes the statistically proper way to incorporate the laser scanner range variance into the pose estimate covariance matrix.

The measurement matrix, given in (6), is created by considering the geometry shown in Figure 5, while $\mathbf{\Sigma}_r^2$ is a diagonal matrix containing the error in the individual range measurements (7). First, the laser scan is processed using the Polar Recursive Line Segmentation (PRLS) method described by Sobers.[15] The PRLS algorithm is an efficient way to calculate local normals, its primary purpose here, but it also provides a segmented-line model of the environment in normal form, where ρ_i is the perpendicular distance from the origin to each segment and ϕ_i is the angle to the normal direction. The PRLS algorithm uses the variable substitution shown in (8) to enable a linear regression via Kalman filter sequential processing of the laser scan points. Since the intent of this research is to demonstrate a very streamlined approach to SLAM, the navigation algorithm did not perform feature tracking on the extracted line segments, although this is a well-studied technique that could be incorporated to improve map accuracy if desired and the computational expense is warranted. Here, the emphasis is on accurately modeling the error associated with each range measurement to improve pose estimation.

$$\mathbf{I} \triangleq \mathbf{H}^T {\mathbf{\Sigma}_r^2}^{-1} \mathbf{H} = \mathbf{R}^{-1} \tag{5}$$

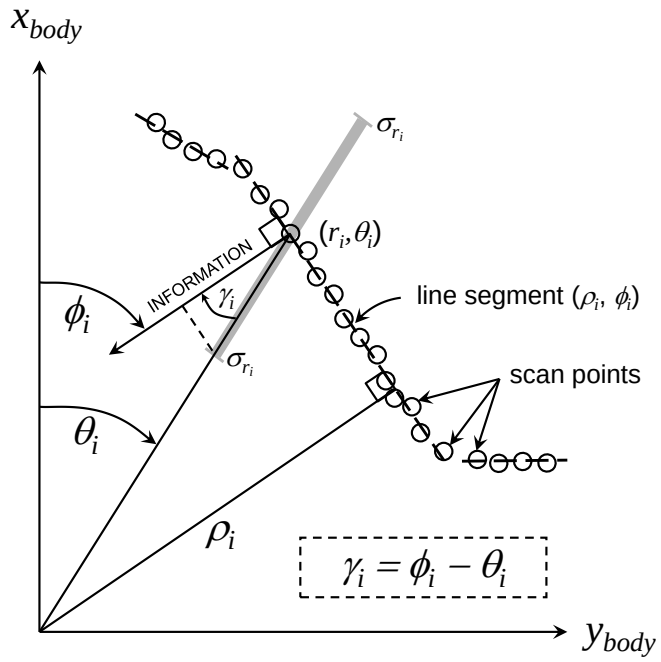


Figure 5. The information available from a single laser range measurement r_i at measurement angle θ_i is in the direction perpendicular to the local topology. Here, ϕ_i is the angle between the body x axis and the local normal for each measurement, and θ_i is the angle from the body x axis to the measurement point. Hence, γ_i is the angle (in the body frame) between the measurement direction and the local normal.

$$\mathbf{H} = \begin{bmatrix} \cos(\phi_1)\cos(\gamma_1) & \sin(\phi_1)\cos(\gamma_1) & r_1\sin(\gamma_1) \\ \vdots & \vdots & \vdots \\ \cos(\phi_n)\cos(\gamma_n) & \sin(\phi_n)\cos(\gamma_n) & r_n\sin(\gamma_n) \end{bmatrix} \tag{6}$$

$$\mathbf{\Sigma}_r^2 = diag(\sigma_{r_1}^2, \sigma_{r_1}^2, \sigma_{r_1}^2, \ldots, \sigma_{r_i}^2, \sigma_{r_i}^2, \sigma_{r_i}^2, \ldots, \sigma_{r_n}^2, \sigma_{r_n}^2, \sigma_{r_n}^2) \tag{7}$$

$$A_i \triangleq \frac{\cos\phi_i}{\rho_i} \quad , \qquad B_i \triangleq \frac{\sin\phi_i}{\rho_i} \tag{8}$$

To estimate DOP, the measurement matrix projects the range information ($1/\sigma_{r_i}^2$), which is in the range direction, through angle γ_i onto the vector normal to the surface at each scan point. This captures the effect that no information from the laser range measurement and scan matching routine is available parallel to a surface. Then, the $(x, y)_{body}$ components of the information projection are calculated using angle ϕ_i to get the information in each body direction. Thus, $\mathbf{H}$ is constructed, and the information matrix can be calculated by summing over the entire set of scan points as shown in (9), where the summation terms are defined in (10). Inverting the 3×3 information matrix produces the pose estimate covariance matrix, $\mathbf{R}_{pose}$, which is used to update the EKF full state estimate. Note from (10) that more position information is present in range measurements when the surface is close to perpendicular (γ_i small), while measurements have more heading information when the surface is close to parallel and farther away from the laser source (r_i large, $\gamma_i \approx \pm\frac{\pi}{2}$).

$$\mathbf{I} = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xy} & \Sigma_{x\psi} \\ \Sigma_{xy} & \Sigma_{yy} & \Sigma_{y\psi} \\ \Sigma_{x\psi} & \Sigma_{y\psi} & \Sigma_{\psi\psi} \end{bmatrix} \tag{9}$$

$$
\begin{aligned}
\Sigma_{xx} &\triangleq \sum_{i=1}^{n} \frac{\cos^2(\phi_i)\cos^2(\gamma_i)}{\sigma_{r_i}^2} & \Sigma_{xy} &\triangleq \sum_{i=1}^{n} \frac{\cos(\phi_i)\sin(\phi_i)\cos^2(\gamma_i)}{\sigma_{r_i}^2} \\
\Sigma_{yy} &\triangleq \sum_{i=1}^{n} \frac{\sin^2(\phi_i)\cos^2(\gamma_i)}{\sigma_{r_i}^2} & \Sigma_{x\psi} &\triangleq \sum_{i=1}^{n} \frac{r_i\cos(\phi_i)\cos(\gamma_i)\sin(\gamma_i)}{\sigma_{r_i}^2} \\
\Sigma_{\psi\psi} &\triangleq \sum_{i=1}^{n} \frac{r_i^2\sin^2(\gamma_i)}{\sigma_{r_i}^2} & \Sigma_{y\psi} &\triangleq \sum_{i=1}^{n} \frac{r_i\sin(\phi_i)\cos(\gamma_i)\sin(\gamma_i)}{\sigma_{r_i}^2}
\end{aligned} \tag{10}
$$

While the above form preserves at least some geometric insight into the problem, the speed of the algorithm can be significantly improved in implementation by avoiding excessive trigonometric evaluations where possible. Although the scan points are measured directly in $(r,\,\theta)$ body coordinates, maps stored in memory are generally in local inertial cartesian coordinates. Hence, transformation from polar to cartesian coordinates is required regardless of the pose estimation algorithm used. Once that transformation is accomplished, avoiding further trigonometric calculations where possible improves performance. In this algorithm, significant reductions in computational burden were achieved by leveraging the parameters A and B provided directly by the PRLS algorithm (8), the fact that $\rho_i^2 = (A_i^2 + B_i^2)$, and the trigonometric identities shown in (11). The resulting formulation shown in (12) requires no trigonometric evaluation other than the initial conversion from polar to cartesian coordinates.

Figures 6-8 show the error ellipses associated with the pose measurement covariance calculated by the algorithm presented here using scan data collected during experimental testing. In the figures, the error is normalized using the sensor standard deviation σ_r and scaled up as indicated in the plots to improve visualization of the shape and relative size of the measurement uncertainty. The 1-σ 2-D position uncertainty is shown by an ellipse, while the 1-σ heading uncertainty is shown by a circle. In comparing the different scenarios, the relative size of the circle indicates the relative uncertainty in the heading estimates, while the size and shape of the error ellipse indicates the relative position uncertainty in the position estimation.

$$
\begin{aligned}
\cos\theta_i &= \tfrac{x_i}{r_i} & \cos(\gamma_i) &= \cos(\phi_i - \theta_i) = \cos\phi_i\cos\theta_i + \sin\phi_i\sin\theta_i \\
\sin\theta_i &= \tfrac{y_i}{r_i} & \sin(\gamma_i) &= \sin(\phi_i - \theta_i) = \sin\phi_i\cos\theta_i - \cos\phi_i\sin\theta_i
\end{aligned} \tag{11}
$$

$$
\begin{aligned}
\Sigma_{xx} &\triangleq \sum_{i=1}^{n} \frac{A_i^2[A_i^2x_i^2 + 2A_iB_ix_iy_i + B_i^2y_i^2]}{(A_i^2+B_i^2)^2r_i^2\sigma_{r_i}^2} & \Sigma_{xy} &\triangleq \sum_{i=1}^{n} \frac{A_iB_i[A_i^2x_i^2 + 2A_iB_ix_iy_i + B_i^2y_i^2]}{(A_i^2+B_i^2)^2r_i^2\sigma_{r_i}^2} \\
\Sigma_{yy} &\triangleq \sum_{i=1}^{n} \frac{B_i^2[A_i^2x_i^2 + 2A_iB_ix_iy_i + B_i^2y_i^2]}{(A_i^2+B_i^2)^2r_i^2\sigma_{r_i}^2} & \Sigma_{x\psi} &\triangleq \sum_{i=1}^{n} \frac{A_i[A_iB_i(x_i^2 - y_i^2) - x_iy_i(A_i^2 - B_i^2)]}{(A_i^2+B_i^2)^2r_i^2\sigma_{r_i}^2} \\
\Sigma_{\psi\psi} &\triangleq \sum_{i=1}^{n} \frac{[B_i^2x_i^2 - 2A_iB_ix_iy_i + A_i^2y_i^2]}{(A_i^2+B_i^2)^2r_i^2\sigma_{r_i}^2} & \Sigma_{y\psi} &\triangleq \sum_{i=1}^{n} \frac{B_i[A_iB_i(x_i^2 - y_i^2) - x_iy_i(A_i^2 - B_i^2)]}{(A_i^2+B_i^2)^2r_i^2\sigma_{r_i}^2}
\end{aligned} \tag{12}
$$

B. CoreSLAM Implementation

A variety of SLAM algorithm implementations are available for free use at the web site OpenSLAM.org. The algorithm used for this research, called CoreSLAM, was chosen primarily because it is simple, easy to implement, and it uses integer math where possible to improve computational speed.[33,8] There are two main parts to a SLAM routine. The first task is to measure the distance to obstacles or landmarks in the environment and map them given the vehicle's position and orientation (i.e. mapping). The second task is to determine the best estimate of the vehicle's position and orientation based on the latest scan (or series of scans) given a stored map (i.e. localization). The mapping and localization tasks are performed together to

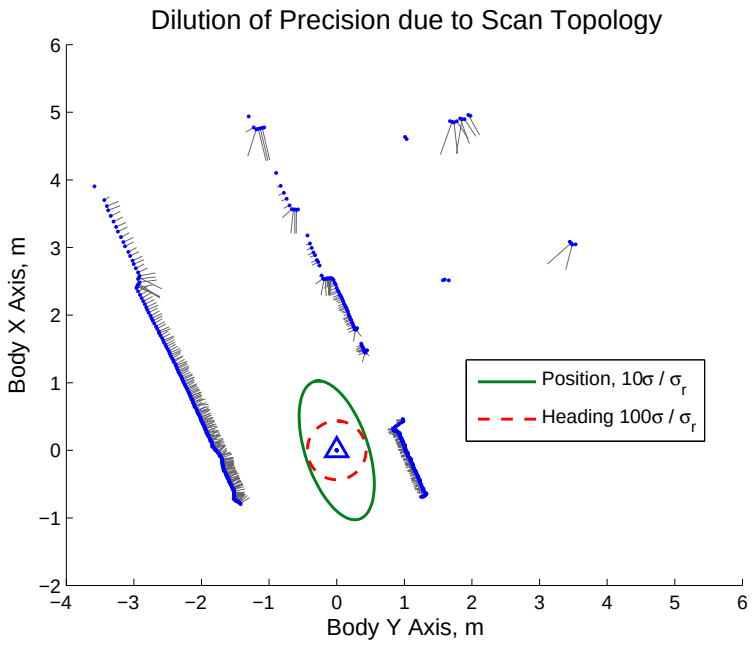


Figure 6. In a hallway, uncertainty is greater in the direction of the hallway. Note, the position and heading error information has been nondimensionalized and scaled to enable visualization on the same graph.

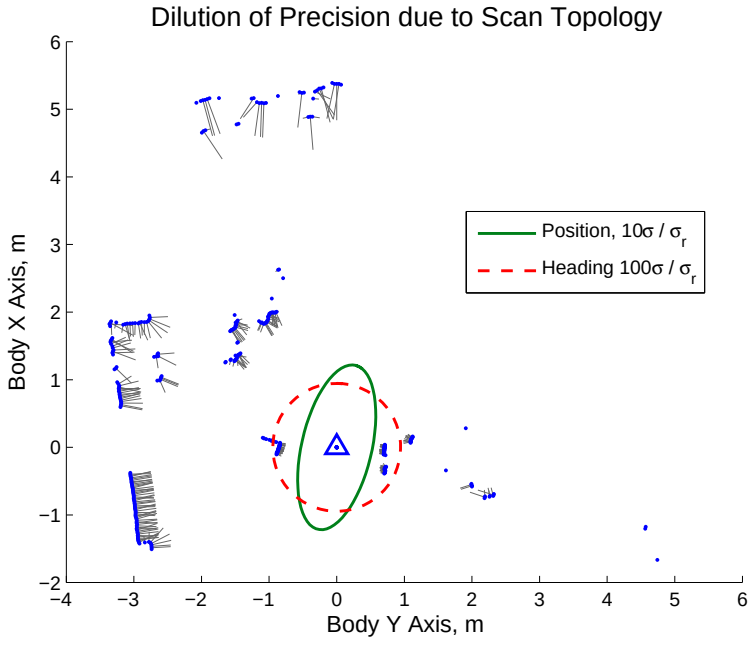


Figure 7. If many surface normals point toward the vehicle, the heading uncertainty is greater. Note, the position and heading error information has been nondimensionalized and scaled to enable visualization on the same graph.

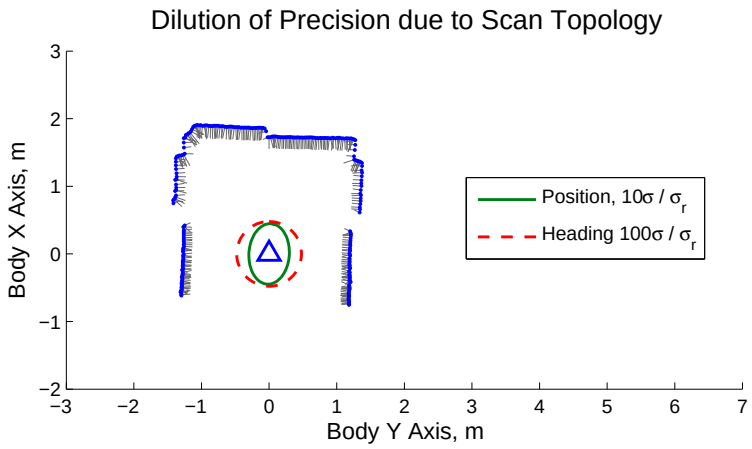


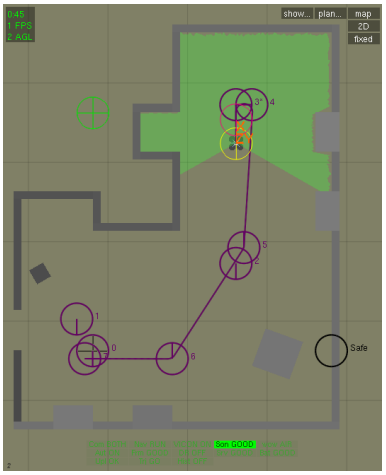
Figure 8. An environment with walls on several sides reduces uncertainty in the pose measurement. Note, the position and heading error information has been nondimensionalized and scaled to enable visualization on the same graph.

maintain the most current map and position estimate. As discussed in Ref. 15, the robotics and computer science communities place considerable emphasis on keeping track of vehicle motion and solving chains of pose constraints between different locations to make corrections to a global map. For basic indoor navigation of flying vehicles a tradeoff exists between the desired level of accuracy in a global map and the computational power available onboard. Therefore, in the presented results, localization and pose estimation with respect to the immediate environment has been prioritized over building and maintaining a highly accurate global map. Particularly, the implemented CoreSLAM mapping routine does not detect or correct errors in past observations by performing loop closure.

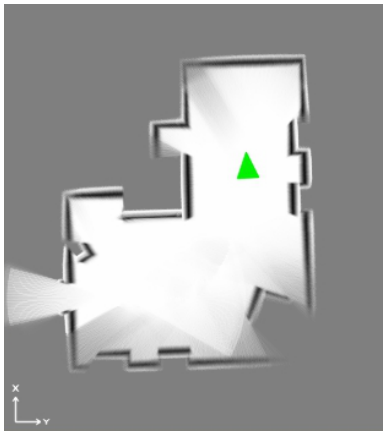
In CoreSLAM, the estimated vehicle pose is used to align each new scan with the generated map. CoreSLAM maintains a map that consists of a two-dimensional array containing integer values ranging from 0 to 65535. The map is initialized to a middle value of 32768, and values are adjusted as each new scan is processed to reflect the evolving map. In order to easily visualize the map, the values are scaled to the range 0-255 and displayed as a one-to-one scale 8-bit grayscale image. As observations are made, areas where obstacles are detected are darkened, and areas that are clear of obstacles increase in brightness. The map image thus represents an occupancy grid similar to that developed by Thrun et al, with color value representing the probability that a square is occupied.[34] As more areas are explored, the observed obstacles are shown by darkened areas on the map, while clear areas are displayed by lighter colored pixels. In order to reduce skew in the map caused by pitch and roll of the aircraft, the scans are projected into the 2-D plane of the SLAM map using the vehicle attitude estimates from the EKF navigation filter. In addition, occasional ceiling or floor measurements or other out-of-plane observations during flight have little temporal persistence and as such have low probabilities of occupation on the map. Hence these do not adversely affect the scan registration algorithm or resulting pose estimation.

Once the map is initialized, the algorithm incorporates new scan data into the map after each scan. The scanning laser returns scan data as a vector of range measurements corresponding to a counterclockwise increasing angle. CoreSLAM converts the data from polar to Cartesian coordinates before it is processed. For this CoreSLAM implementation, the pose estimation algorithm was improved by initializing it using the full EKF navigation solution rather than simply using the previous CoreSLAM pose estimate. The pose estimation routine uses a Monte Carlo search of nearby poses to find a good estimate for the new pose, based on user-defined parameters (see Ref. 8). During the Monte Carlo search, the current scan information is evaluated at the predicted new pose to see how closely it matches the current map. If the scan matching routine can find a better pose to match the current scan with the current map, the current pose is updated to the new pose. In Ref. 8, the measure of how well a particular scan matches the current map is described as the "distance" between the scan and the map. The Monte Carlo routine calculates the distance between

the scan and the map and returns the pose that minimizes this distance. Next, the scan data is incorporated into the map using the updated pose. Figure 9(a) shows a simulated building interior being explored by the GTQ with a scanning laser rangefinder. Figure 9(b) shows the map generated during a simulated flight.



(a) Mapping a simulated environment



(b) Map generated during simulation

Figure 9. The map maintained by the onboard mapping routine represents an occupancy grid, where the value of each pixel in the image represents the likelihood that a particular grid square is occupied. Here, lighter colors represent free space, while darker colors represent obstacles and medium gray areas are unexplored. Areas with higher contrast represent greater certainty due to longer observation periods during the flight. The green triangle represents the vehicle's estimated position and heading.

For this research, the CoreSLAM algorithm was modified and improved in two important ways.[15] First, the position covariance matrix of the navigation solution was used as an input to the map update function. In the original algorithm, a user-defined constant value was used to create the Gaussian uncertainty on obstacle locations. To prevent unrealistic confidence in the map, the actual sensor range uncertainty (which is a function of range detected) was added to the vehicle's position uncertainty. The map update algorithm was then modified to use this total uncertainty when assimilating scan data into the map. A second improvement made to the CoreSLAM algorithm was to use the vehicle's state estimate and covariance as inputs to the Monte Carlo search. Thus, the random search initial conditions and search scope were significantly improved over the original algorithm by providing statistically appropriate search parameters.

C. Extended Kalman Filter State Estimation

The presented navigation algorithm for vehicle state estimation is an extension of an EKF-based navigation architecture developed previously.[5,10] The major nontrivial modification here is that it is augmented to use the SLAM pose estimate and sonar altitude measurements, with updates at 10 Hz and 20 Hz respectively, in lieu of GPS position measurements and magnetometer heading measurements. The resulting nonlinear filtering problem that is solved here is to correct for the drift biases of the inertial sensors and estimate the position and attitude accurately by combining measurements. The EKF is a widely-used tool for obtaining suboptimal solutions to nonlinear filtering problems (see e.g. Refs. 4, 34). The architecture presented here is of the mixed continuous-discrete type. The navigation filter is used to estimate the following vehicle states: the position ${}^{i}\mathbf{x} = [x_1, x_2, x_3]^T$, velocity ${}^{i}\mathbf{v} = [v_1, v_2, v_3]^T$, and the attitude quaternion $\mathbf{q} = [q_1, q_2, q_3, q_4]^T$. It is beneficial to use a minimal representation of the attitude quaternion for describing the orientation of the aircraft.[35] This is achieved by defining an error quaternion as $\delta\mathbf{q} \approx [1, \mathbf{s}]^T$ such that $\delta\mathbf{q} \otimes \hat{\mathbf{q}} = \mathbf{q}$, where $\otimes$ is the quaternion product operator. The error quaternion is propagated during EKF updates, and is used to correct the attitude estimates after each measurement update. Note that in practice the error quaternion takes on very small values, therefore, the update equation for error quaternions can be approximated by $\dot{\hat{\mathbf{s}}} = 0$. The vector $\mathbf{s} \in \Re^3$ to be tracked is the minimal representation of the attitude error. The accelerometer measurements ${}^{b}\mathbf{a} = [a_x, a_y, a_z]^T$ and the gyroscope measurements ${}^{b}\boldsymbol{\omega} = [p, q, r]$ from the IMU (which updates at 100 Hz) are modeled to account for sensor biases as follows: ${}^{b}\mathbf{a} = \mathbf{a}_{raw} - \mathbf{b}_a$ and ${}^{b}\boldsymbol{\omega} = \boldsymbol{\omega}_{raw} - \mathbf{b}_g$. The navigation filter is designed to estimate the biases $\mathbf{b}_a \in \Re^3$ and $\mathbf{b}_g \in \Re^3$ of

the accelerometer and gyroscope respectively. This is important, as measurements from low-cost off-the-shelf IMUs are often subject to an unknown bias. In combination, the state vector to be estimated by the navigation filter is $\hat{\mathbf{x}}(t) \in \Re^{15}$ and is given as

$$\hat{\mathbf{x}}(t) = \left[\hat{\mathbf{s}}, {}^{i}\hat{\mathbf{x}}, {}^{i}\hat{\mathbf{v}}, \hat{\mathbf{b}_a}, \hat{\mathbf{b}_g}\right]^T, \tag{13}$$

where $\hat{}$ represents estimated values. The following nonlinear process model is used for predicting state estimates:

$$\dot{\hat{\mathbf{b}}}_a = 0, \tag{14}$$

$$\dot{\hat{\mathbf{b}}}_g = 0, \tag{15}$$

$$\dot{\hat{\mathbf{s}}} = 0, \tag{16}$$

$${}^{i}\dot{\hat{\mathbf{x}}} = \hat{\mathbf{v}}, \tag{17}$$

$${}^{i}\dot{\hat{\mathbf{v}}} = \hat{\mathbf{L}}_{b\to i}{}^{b}\mathbf{a}. \tag{18}$$

The following equation is used to predict the quaternion estimate using bias-corrected measurements from the gyroscopes:

$$\dot{\hat{\mathbf{q}}} = \frac{1}{2}\begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \tag{19}$$

The error covariance $\mathbf{P}$ is propagated using the following equation

$$\dot{\mathbf{P}} = \mathbf{AP} + \mathbf{PA}^T + \mathbf{Q}, \tag{20}$$

where $\mathbf{A}$ is the Jacobian of the process model with respect to the states, and $\mathbf{Q}$ is the process covariance matrix. Let the superscripts $-$ and $+$ denote predicted and corrected variables, respectively. The process model is propagated using (14)-(19) to yield $\hat{\mathbf{x}}^-$. The navigation EKF then corrects the predicted states and covariance using a measurement model $h(\hat{\mathbf{x}}_k^-)$ as follows:

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k\big(\mathbf{z}_k - h(\hat{\mathbf{x}}_k^-)\big) \tag{21}$$

$$\mathbf{P}_k^+ = \big(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k(\hat{\mathbf{x}}_k^-)\big)\mathbf{P}_k^- \tag{22}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\hat{\mathbf{x}}_k^-)\big(\mathbf{H}_k(\hat{\mathbf{x}}_k^-)\mathbf{P}_k^-\mathbf{H}_k^T(\hat{\mathbf{x}}_k^-) + \mathbf{R}_k\big)^{-1} \tag{23}$$

where $\mathbf{R}_k$ denotes the measurement covariance matrix and $\mathbf{H}_k$ denotes the Jacobian of the measurement model (the information matrix). The updates are performed sequentially. That is, whenever a sensor measurement is available, the state and error covariances are updated for that measurement.[10] This handles the real-world situation where sensor measurements arrive at different update rates. The sonar measurement is uncorrelated to other measurements, hence its measurement model is given by $z_{sonar} = x_3 + \omega_{sonar}$, where ω_{sonar} is white noise with experimentally determined variance. The covariance of the SLAM pose measurement, $\mathbf{R}_{pose}$, is calculated by inverting the information matrix as shown in (5). $\mathbf{R}_{pose}$ is in the body frame, and is transformed to the inertial frame using $\mathbf{L}_{b\to i}$. Note that the quaternion is normalized and the error quaternion $\hat{\mathbf{s}}$ is reset to 0 after every measurement update.

V. Control Algorithm

A. Stability Augmentation System

The quadrotor platform is inherently unstable, and without control inputs the platform would enter an uncontrolled drift in velocity and angular rates eventually colliding with the ground or nearby obstacles. Quadrotors are also known to be notoriously hard to control even for human pilots, particularly because

the relationship between thrust and stick deflection is nonlinear and the attitude is coupled heavily with velocity. Hence, it is desirable to incorporate rate damping on all the angular rate axes to aid the pilot in controlling the quadrotor. Let $\hat{p}$, $\hat{q}$, and $\hat{r}$ denote the gyroscope measurements of the quadrotor roll, pitch, and yaw rates, and δ_{p_p}, δ_{q_p}, and δ_{r_p} denote the pilot roll, pitch, and yaw stick deflections. Then, the actual rate commands are assigned using the following proportional control logic:

$$\delta_p = \delta_{p_p} - K_p\hat{p} \tag{24}$$

$$\delta_q = \delta_{q_p} - K_q\hat{q} \tag{25}$$

$$\delta_r = \delta_{r_p} - K_r\hat{r} \tag{26}$$

In the above equations, K_p, K_q, and K_r denote the linear gains chosen to provide appropriate rate damping.

B. Attitude and Position Controller

The complexity of a control system depends not only on the quantities being controlled, but also on the dynamics of the system itself. Unlike ground vehicles, unstable air vehicles are susceptible to oscillation and divergent flight when the control system is not properly tuned. Even for stable flying vehicles, coupling between lateral and longitudinal motion, as well as aerodynamic interaction with the environment, must be considered. The control architecture used here leverages a proven model reference control architecture developed for control of a VTOL UAS.[36,37,38] In this architecture, a position control loop generates a velocity command, a velocity control loop generates an attitude command, and an attitude control loop generates servo commands to stabilize the vehicle by controlling the angular rates. Kannan has shown that such a nested and cascaded control loop architecture with actuator saturation can indeed be used to control VTOL UAS.[37]

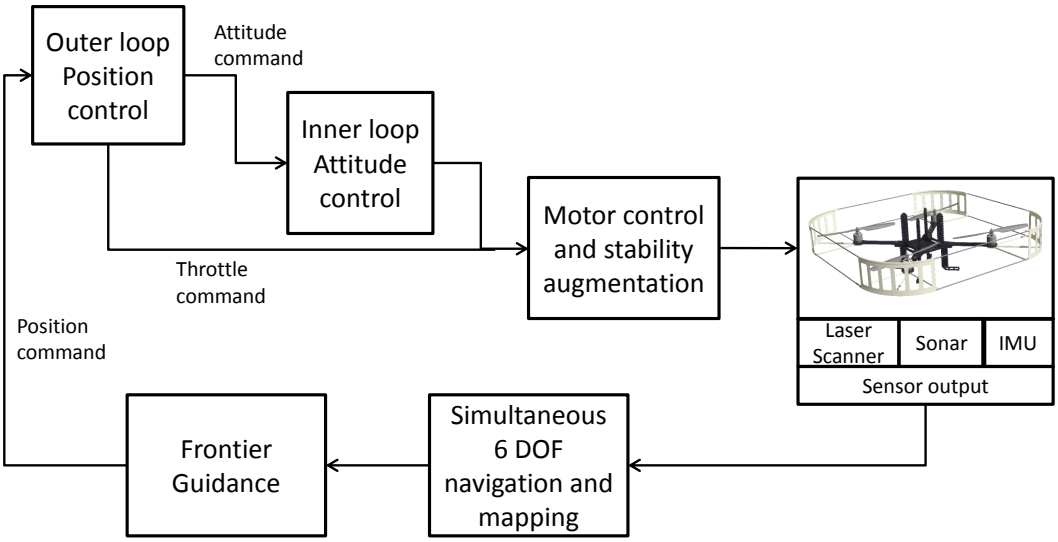


Figure 10. A representation of the GTQ control architecture. Sensor information is fused to simultaneously estimate the vehicle velocity, attitude, and position as well as for mapping the environment. This information is used by the frontier guidance algorithm to command position-based waypoints. A cascaded outer-loop position and inner-loop attitude controller is used to track the desired waypoints.

This system of nested control loops (see Figure 10) requires that the vehicle maintain an estimate of its position, velocity, attitude, and angular rate, with the addition of an external inertial position reference if position control is to be ultimately achieved. Following is a brief description of the implemented dynamic inversion based model reference control architecture (see references Refs. 39, 36, 37, 38 for further details).

Let the state of the system for controls purposes be represented by $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2]^T$, where $\mathbf{x}_1 = [u, v, w]^T$ denotes the outerloop state vector representing the body velocity in the x, y, and z axes respectively, and $\mathbf{x}_2 = [p, q, r]^T$ denotes the innerloop state vector representing the body angular rate around x, y, and z axes

respectively. Let $\boldsymbol{\delta} = [\delta_p, \delta_q, \delta_r]^T$ denote the roll, pitch, and yaw commands. Next follows a discussion of the innerloop design; the outerloop design follows similarly. Begin by assuming that the rotorcraft dynamics around a trim point can be represented by the following linear equation:

$$\dot{\mathbf{x}}_2 = \mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 + \mathbf{B}\boldsymbol{\delta} + \mathbf{B}\boldsymbol{\delta}_{trim} \quad , \tag{27}$$

where $\mathbf{A}_1 \in \Re^{3\times3}$ and $\mathbf{A}_2 \in \Re^{3\times3}$ denote the matrices containing the linearized estimates of the translational and attitude aerodynamic derivatives and $\mathbf{B} \in \Re^{3\times3}$ represents the matrix containing the control effectiveness derivatives. Note that with nonzero control effectiveness derivatives $\mathbf{B}$ is an invertible matrix. Let $\boldsymbol{\alpha}_{des}$ denote the desired angular acceleration. Then, the required control input $\boldsymbol{\delta}$ can be found by inverting an approximation of (27) by ignoring the unknown trim input:

$$\boldsymbol{\delta} = \mathbf{B}^{-1}(\boldsymbol{\alpha}_{des} - \mathbf{A}_1\mathbf{x}_1 - \mathbf{A}_2\mathbf{x}_2). \tag{28}$$

A linear reference model is selected to characterize the desired response of the system

$$\dot{\mathbf{x}}_{2_{rm}} = \mathbf{A}_{rm}\mathbf{x}_{2_{rm}} + \mathbf{B}_{rm}\mathbf{r}. \tag{29}$$

This results in a tracking error $\mathbf{e} = \mathbf{x}_{2_{rm}} - \mathbf{x}_2$. The desired angular acceleration is calculated as

$$\boldsymbol{\alpha}_{des} = \mathbf{u}_{rm} + \mathbf{u}_{pd} - \mathbf{u}_I, \tag{30}$$

where $\mathbf{u}_{rm}$ denotes a feedforward command set to be equal to $\dot{\mathbf{x}}_{2_{rm}}$, $\mathbf{u}_{pd} = \mathbf{K}\mathbf{e}$ denotes a feedback command with $\mathbf{K} \in \Re^{3\times3}$ a positive definite matrix containing the linear feedback gains, and $\mathbf{u}_I$ denotes the output of an integrator. Differentiating the tracking error with respect to time, we obtain

$$\begin{aligned}
\dot{\mathbf{e}} &= \dot{\mathbf{x}}_{2_{rm}} - \dot{\mathbf{x}}_2 && (31)\\
&= \mathbf{u}_{rm} - (\mathbf{A}_1\mathbf{x}_1 + \mathbf{A}_2\mathbf{x}_2 + \mathbf{B}(\mathbf{B}^{-1}(\boldsymbol{\alpha}_{des} - \mathbf{A}_1\mathbf{x}_1 - \mathbf{A}_2\mathbf{x}_2)) + \mathbf{B}\boldsymbol{\delta}_{trim}) && (32)\\
&= -\mathbf{u}_{pd} + \mathbf{u}_I - \mathbf{B}\boldsymbol{\delta}_{trim} && (33)\\
&= -\mathbf{K}\mathbf{e} + \mathbf{u}_I - \mathbf{B}\boldsymbol{\delta}_{trim}. && (34)
\end{aligned}$$

Therefore, if $\mathbf{u}_I - \mathbf{B}\boldsymbol{\delta}_{trim} = 0$, the tracking dynamics are exponentially stable. The integral control input $\mathbf{u}_I$ is updated using the following differential equation

$$\dot{\mathbf{u}}_I = -\mathbf{e}^T\mathbf{P}\mathbf{B}\mathbf{1}, \tag{35}$$

where $\mathbf{P}$ is the positive definite solution to the Lyapunov equation $0 = \mathbf{K}^T\mathbf{P} + \mathbf{P}\mathbf{K} + \mathbf{Q}$ for a nonnegative definite matrix $\mathbf{Q}$, and $\mathbf{1} = [1, 1, 1]^T$. The stability of the control law in (30) can be established using Lyapunov techniques.[36] Note that in the presented control architecture the dynamics of the quadrotor are assumed to be linear. For the purpose of the flight tests results reported in this paper, this approximation was found to be sufficient. This assumption however, can be removed by using an adaptive controller with a neural network adaptive element.[36,38]

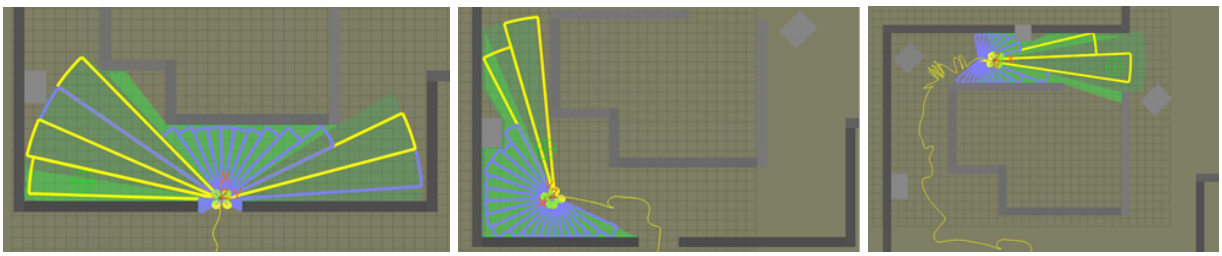
VI. Results

A. Simulation Results

The Georgia Tech UAV Simulation Tool (GUST) framework was utilized for developing a simulation of the vehicle, sensors, and environment.[40] The simulation is designed to reduce development time through risk-free testing of GNC routines. The simulation includes modeling of uncertainties such as gusts, modeling of indoor environments, simulation of complex vehicle dynamics, and elaborate emulation of all sensors and their noise characteristics. The key feature of this simulation environment is that onboard flight code can be directly tested in the simulation, allowing seamless integration of guidance and control laws refined through software-in-the-loop and hardware-in-the-loop testing.

Figure 11 shows a sequence of screen-captures from a simulated flight of the developed integrated guidance navigation and control strategy. The vehicle, its trajectory, commanded waypoint, laser scan, active SRT, and the building are shown in Figure 11. The results confirm that the vehicle is able to sense its surroundings,

form a feasible six degree of freedom navigation solution, and autonomously avoid obstacles while creating the SRT and exploring the building. In Figure 11(c), the quadrotor is able to pass through a small gap on the northwest corner without collision. The quadrotor then explores towards the east of the building, attempting to do an exhaustive search of the building.



(a) (b) (c)

Figure 11. Simulated of exploration of an unknown indoor environment.

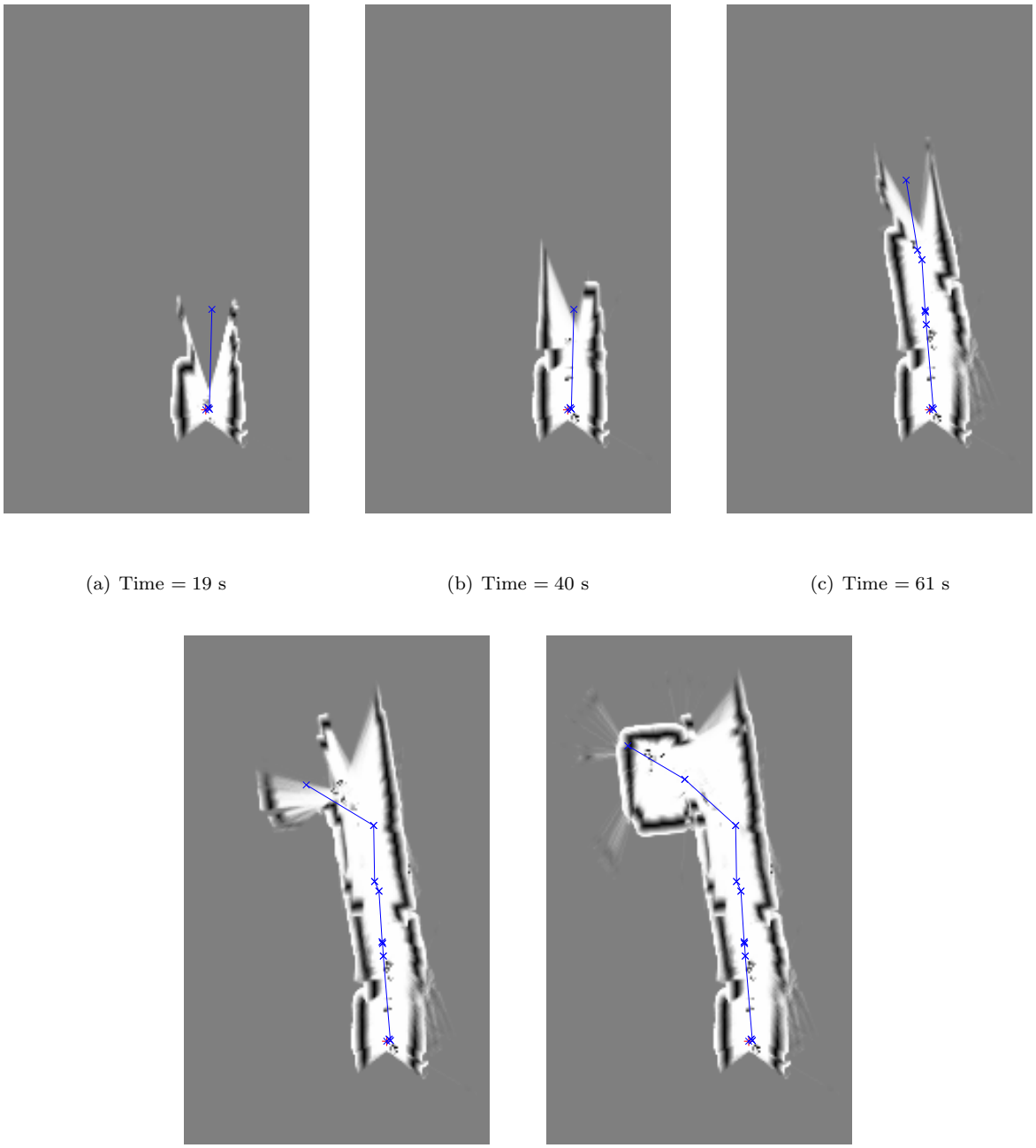
B. Flight Test Results

This section presents flight test results of the GTQ during fully autonomous exploration of an indoor cluttered environment without any external sensing aids (such as GPS). The onboard GNC algorithm does not assume any *a priori* knowledge of the indoor environment. Navigation is performed by solving the SLAM problem online using techniques presented in Section IV. Guidance is achieved by a frontier-guidance method coupled with wall-following guidance as described in Section III, and control is achieved using the control architecture described in Section V. The flight test begins with the aircraft hovering autonomously about 0.85 meters above the ground. The onboard guidance logic then commands waypoints that take the aircraft towards unexplored frontiers, and the onboard navigation routine provides the aircraft with pose estimates and simultaneously builds a map of the environment in real-time. The map information is fed back into the guidance logic so that new frontiers can be found and explored. Note that all computation, including SLAM, is performed onboard using measurements from the sensors described in Section II. The GNC algorithms were optimized for execution completely onboard the embedded computer (Gumstix Overo Fire) by trading map accuracy for reliable, rapid convergence of the pose estimate. This trade-off results in a slight skewing of the onboard generated map. However, the position accuracy was found acceptable for the reliable exploration of indoor areas.

Figure 13 shows the position tracking performance of the GTQ. The figure shows that the onboard control algorithm accurately tracks the commanded position. Note that the standard aerospace North-East-Down coordinate system is used, this results in the z direction being negative up.

VII. Conclusion

This paper presented the details of a quadrotor miniature unmanned aerial system, the GTQ, designed for autonomous exploration of indoor areas. The GTQ uses an off-the-shelf platform equipped with off-the-shelf avionics and sensor packages, with custom flight software. Information from a scanning laser range sensor, inertial measurement unit, and an altitude measurement sonar are fused to form an elaborate navigation solution using Simultaneous Localization and Mapping methods. Two important features of this navigation architecture are that it does not rely on any external navigational aid, such as GPS, and that all SLAM and GNC computations are performed entirely onboard. A frontier-based exhaustive search is used for exploring unknown indoor environments using the laser scan data by placing command waypoints on the map generated by the SLAM routine. Since observable features in the environment are necessary for an accurate SLAM solution, the guidance algorithm is coupled with the navigation algorithm to ensure the vehicle approaches the frontier-based waypoints through a trajectory that is close to walls and other indoor structures, while maintaining a safe operating distance. A cascaded inner-outer loop control architecture is utilized, which relates stick commands to attitude commands and attitude commands to velocity commands. The control system also uses a linear stability augmentation system that uses rate feedback to dampen the vehicle angular rate response.



(a) Time = 19 s (b) Time = 40 s (c) Time = 61 s

(d) Time = 92 s (e) Time = 124 s

Figure 12. These snapshots of the onboard map were generated in a flight test, showing that the GTQ can autonomously explore an unknown cluttered indoor environment without any external sensing or computational aids. Stars mark the waypoints commanded by the guidance strategy which ensures the GTQ explores unexplored areas of the indoor environment while staying close to navigable features. The pictured area is approximately 15 by 24 meters. Note that all computation, including solving the SLAM problem, is performed onboard.

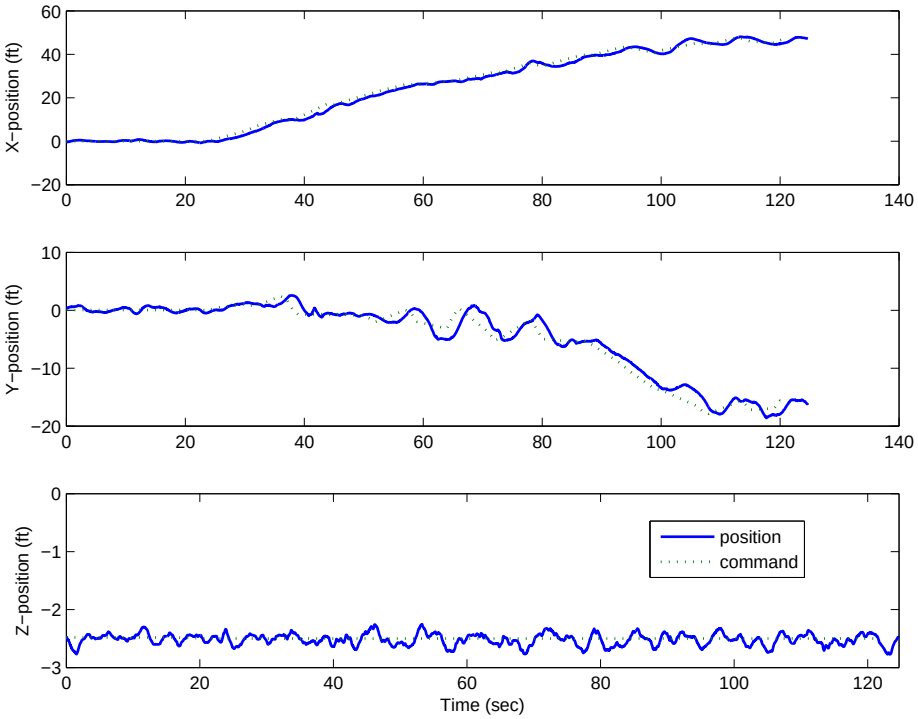


Figure 13. Position tracking performance of the GTQ while autonomously exploring an unknown cluttered indoor environment without any external sensing or computational aids. The dotted line represents the commanded position as extrapolated from the waypoints generated by the guidance algorithm of Section III.

An elaborate simulation model of the vehicle has been developed and the guidance, navigation, and control algorithms have been validated in simulation and several times in flight. Flight test results of the GTQ exploring unknown indoor environments without relying on any external sensing or computational aids were presented. These results establish the feasibility of the proposed approach to develop a completely self-contained miniature unmanned aerial system capable of autonomously exploring indoor areas.

Acknowledgments

The authors wish to thank the members and friends of the Georgia Tech Aerial Robotics team, including (in alphabetic order) Dmitry Bershadsky, Timothy Dyer, Eohan George, Jeong Hur, Dr. Suresh Kannan, Daniel Magree, Maximilian Mühlegg, Nimrod Rooz, Dr. Erwan Salaün, Sushaku Yamaura, and others. The authors also wish to thank Lockheed Martin Aeronautics Company; Raytheon; Dragonfly Pictures, Inc.; and Analog Devices, Inc. for their support.

References

[1]Dissanayake, M., Newman, P., Clark, S., Durrant-Whyte, H., and Csorba, M., "A solution to the simultaneous localization and map building (SLAM) problem," *Robotics and Automation, IEEE Transactions on*, Vol. 17, No. 3, jun 2001, pp. 229 –241.

[2]Zhou, W., Miro, J., and Dissanayake, G., "Information-Efficient 3-D Visual SLAM for Unstructured Domains," *Robotics, IEEE Transactions on*, Vol. 24, No. 5, oct. 2008, pp. 1078 –1087.

[3]Guivant, J. and Nebot, E., "Solving computational and memory requirements of feature-based simultaneous localization and mapping algorithms," *Robotics and Automation, IEEE Transactions on*, Vol. 19, No. 4, aug. 2003, pp. 749 – 755.

[4]Kayton, M. and Fried, W. R., *Avionics Navigation Systems*, John Wiley and Sons, 1997.

[5]Christophersen, H. B., Pickell, W. R., Neidoefer, J. C., Koller, A. A., Kannan, S. K., and Johnson, E. N., "A compact Guidance, Navigation, and Control System for Unmanned Aerial Vehicles," *Journal of Aerospace Computing, Information, and Communication*, Vol. 3, No. 5, May 2006, pp. 187–213, AIAA paper number 2006 1542-9423, doi 10.2514/1.18998.

[6]Wendel, J., Maier, A., Metzger, J., and Trommer, G. F., "Comparison of Extended and Sigma-Point Kalman Filters for Tightly Coupled GPS/INS Integration," *AIAA Guidance Navigation and Control Conference*, San Francisco, CA, 2005, AIAA

paper number AIAA-2005-6055.

[7]Miller, P. M., "Mini, Micro, and Swarming Unmanned Aerial Vehicles: A Baseline Study," *the Federal Research Division, Library of Congress*, 2006.

[8]Steux, B. and El Hamzaoui, O., "CoreSLAM: a SLAM Algorithm in less than 200 lines of C code," Tech. rep., Mines ParisTech, Center for Robotics, Paris, France, 2009.

[9]Kim, J. and Sukkarieh, S., "Autonomous airborne navigation in unknown terrain environments," *Aerospace and Electronic Systems, IEEE Transactions on*, Vol. 40, No. 3, july 2004, pp. 1031 – 1045.

[10]Wu, A. and Johnson, E., "Methods for Localization and Mapping Using Vision and Inertial Sensors," *Proceedings of AIAA GNC*, 2008, AIAA-2008-7441.

[11]Achtelik, M., Bachrach, A., He, R., Prentice, S., and Roy, N., "Stereo Vision and Laser Odometry for Autonomous Helicopters in GPS-Denied Indoor Environments," *Proceedings of SPIE*, Vol. 7332, 2009.

[12]M. Achtelik, N. Roy, et al, "Autonomous Navigation and Exploration of a Quadrotor Helicopter in GPS-denied Indoor Environments," *Proc. of the 1st Symposium on Indoor Flight, International Aerial Robotics Competition*, 2009.

[13]Grzonka, S., G. G. and Burgard, W., "Towards a navigation system for autonomous indoor flying," *Robotics and Automation, 2009. Proceedings IEEE International Conference on*, pp. 2878–2883.

[14]Shen, S., Michael, N., and Kumar, V., "Autonomous multi-floor indoor navigation with a computationally constrained micro aerial vehicle," *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, may 2011, pp. 2968 –2969.

[15]Sobers, D. M., Yamaura, S., and Johnson, E. N., "Laser-Aided Inertial Navigation for Self-Contained Autonomous Indoor Flight," *AIAA Guidance Navigation and Control Conference*, Toronto, Ontario, Canada, August 2010, AIAA-2010-8211.

[16]Pravitra, C., Chowdhary, G., and Johnson, E. N., "A Compact Exploration Strategy for Indoor Flight Vehicle," *IEEE Conference on Decision and Control*, Orlando, FL, December 2011, submitted.

[17]Chowdhary, G., Sobers, D. M., Pravitra, C., Christmann, C., Wu, A., Hashimoto, H., Ong, C., Kalghatgi, R., and Johnson, E. N., "Integrated Guidance Navigation and Control for a Fully Autonomous Indoor UAS," *Guidance Navigation and Control Conference*, AIAA, Portland, OR, August 2011, AIAA-2011-6720.

[18]Portlock, J. N. and Cubero, S. N., "Dynamics and Control of a VTOL Quad-Thrust Aerial Robot," *Mechatronics and Machine Vision in Practice*, edited by J. Billingsley and R. Bradbeer, Springer Berlin Heidelberg, 2008, pp. 27–40, 10.1007 978-3-540-74027-83.

[19]Guo, W. and Horn, J., "Modeling and simulation for the development of a quad-rotor UAV capable of indoor flight," *Modeling and Simulation Technologies Conference and Exhibit*, 2006.

[20]Bouabdallah, S., Noth, A., and R., S., "PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor," *Proc. of The IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[21]Johnson, E. and Turbe, M., "Modeling, Control, and Flight Testing of a Small Ducted Fan Aircraft," *Journal of Guidance Control and Dynamics*, Vol. 29, No. 4, July/August 2006, pp. 769–779.

[22]Wang, C.-C., Thorpe, C., and Thrun, S., "Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas," *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, Vol. 1, sept. 2003, pp. 842 – 849 vol.1.

[23]Newman, P., Cole, D., and Ho, K., "Outdoor SLAM using visual appearance and laser ranging," *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, may 2006, pp. 1180 –1187.

[24]Surmann, H., Nchter, A., and Hertzberg, J., "An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments," *Robotics and Autonomous Systems*, Vol. 45, No. 34, 2003, pp. 181 – 198.

[25]Yamauchi, B., "A Frontier-Based Approach for Autonomous Exploration," *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA97)*, Monterey, CA, July 1997, pp. 146–151.

[26]Gonzalez-Banos, H. and Latombe, J.-C., "Navigation Strategies for Exploring Indoor Environments," *Int. J. Robotics Research*, Vol. 21, No. 10, 2002, pp. 829–848.

[27]Makarenko, A. A., Williams, S. B., Bourgault, F., and Durrant-Whyte, H. F., "An Experiment in Integrated Exploration," *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS02)*, 2002.

[28]Freda, L. and Oriolo, G., "Frontier-Based Probabilistic Strategies for Sensor-Based Exploration," *Proc. IEEE International Conference on Robotics and Automation (ICRA05)*, Barcelona, Spain, April 2005.

[29]Freda, L., Loiudice, F., and Oriolo, G., "A Randomized Method for Integrated Exploration," *Proc. IEEE International Conference on Intelligent Robots and Systems (IROS06)*, Beijing, China, Oct. 2006.

[30]Arras, K. O. and Siegwart, R. Y., "Feature extraction and scene interpretation for map-based navigation and map building," Vol. 3210, SPIE, 1998, pp. 42–53.

[31]Núñez, P., Vázquez-Martín, R., del Toro, J., Bandera, A., and Sandoval, F., "Natural landmark extraction for mobile robot navigation based on an adaptive curvature estimation," *Robotics and Autonomous Systems*, Vol. 56, No. 3, 2008, pp. 247 – 264.

[32]Thrun, S., Liu, Y., Ng, A. Y., Ghahramani, Z., and Durrant-Whyte, H., "Simultaneous Localization and Mapping with Sparse Extended Information Filters," *The International Journal of Robotics Research*, Vol. 23, No. 7-8, August 2004, pp. 693–716, doi 10.1177/0278364904045479.

[33]Steux, B. and El Hamzaoui, O., "CoreSLAM on OpenSLAM.org," June 2010.

[34]Thrun, S., Burgard, W., and Fox, D., *Probablistic Robotics*, MIT Press, Cambridge, MA, 2005, pp. 281–308.

[35]Lefferts, E., Markley, F., and Shuster, M., "Kalman Filtering for Spacecraft Attitude Estimation," *Journal of Guidance, Control, and Dynamics*, Vol. 5, No. 5, sept-oct 1982, pp. 417–429.

[36]Johnson, E. and Kannan, S., "Adaptive Trajectory Control for Autonomous Helicopters," *Journal of Guidance Control and Dynamics*, Vol. 28, No. 3, May 2005, pp. 524–538.

[37]Kannan, S. K., *Adaptive Control of Systems in Cascade with Saturation*, Ph.D. thesis, Georgia Institute of Technology, Atlanta Ga, 2005.

[38]Chowdhary, G. and Johnson, E., "Theory and Flight Test Validation of a Concurrent Learning Adaptive Controller," *Journal of Guidance Control and Dynamics*, Vol. 34, No. 2, March 2011, pp. 592–607, 2011-0731-5090,DOI 10.2514/1.46866.

[39]Johnson, E. N., *Limited Authority Adaptive Flight Control*, Ph.D. thesis, Georgia Institute of Technology, Atlanta Ga, 2000.

[40]Johnson, E. N. and Schrage, D. P., "System Integration and Operation of a Research Unmanned Aerial Vehicle," *AIAA Journal of Aerospace Computing, Information and Communication*, Vol. 1, No. 1, Jan 2004, pp. 5–18.