Autonomous Flight in Unstructured and Unknown Indoor Environments

Abraham Bachrach, Ruijie He, and Nicholas Roy*
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

This paper presents our solution for enabling a quadrotor helicopter, equipped with a laser rangefinder sensor, to autonomously explore and map unstructured and unknown indoor environments. While these capabilities are already commodities on ground vehicles, air vehicles seeking the same performance face unique challenges. In this paper, we describe the difficulties in achieving fully autonomous helicopter flight, highlighting the differences between ground and helicopter robots that make it difficult to use algorithms developed for ground robots. We then describe our solutions to the key problems, including a multi-level sensing and control hierarchy, a high-speed laser scan-matching algorithm, EKF data fusion, and a high-level SLAM implementation. Finally, we show experimental results that illustrate the helicopter's ability to navigate accurately and autonomously in unknown environments.

1 Introduction

Micro Aerial Vehicles (MAVs) are increasingly being used in military and civilian domains, including surveillance operations, weather observation, and disaster relief coordination. Enabled by GPS and MEMS inertial sensors, MAVs that can fly in outdoor environments without human intervention have been developed [1, 2, 3, 4].

Unfortunately, most indoor environments and many parts of the urban canyon remain without access to external positioning systems such as GPS. Autonomous MAVs today are thus limited in their ability to fly through these areas. Traditionally, unmanned vehicles operating in GPS-denied environments can rely on dead reckoning for localization, but these measurements drift over time. Alternatively, simultaneous localization and mapping (SLAM) algorithms build a map of the environment around the vehicle while simultaneously using it to estimate the vehicle's position. Although there have been significant advances in developing accurate, drift-free SLAM algorithms in large-scale environments, these algorithms have focused almost exclusively on ground or underwater vehicles. In contrast, attempts to achieve the same results with MAVs have not been as successful due to a combination of limited payloads for sensing and computation, coupled with the fast, unstable dynamics of the air vehicles.

*Email addresses: {abachrac, ruijie, nickroy}@mit.edu

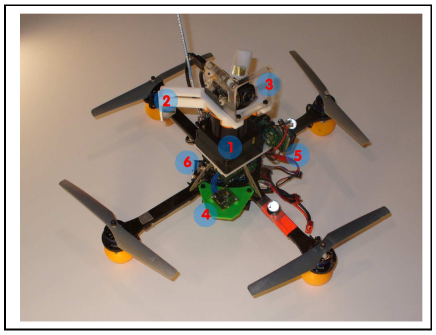


Figure 1: Our quadrotor helicopter. Sensing and computation components include a Hokuyo Laser Rangefinder (1), laser-deflecting mirrors for altitude (2), a monocular camera (3), an IMU (4), a Gumstix processor (5), and the helicopter's internal processor (6)

Figure 2: Autonomous flight in unstructured indoor environments

In this work, we present our quadrotor helicopter system, shown in Figure 1, that is capable of autonomous flight in unstructured indoor environments, such as the one shown in Figure 2. The system employs a multi-level sensor processing hierarchy designed to meet the requirements for controlling a helicopter. The key contributions of this paper are:

1. Development of a fully autonomous quadrotor that relies only on onboard sensors for stable control without requiring prior maps of the environment.
2. A high-speed laser scan-matching algorithm that allows successive laser scans to be compared in real-time to provide accurate velocity and relative position information.
3. A modified 2D SLAM algorithm that handles the 3D motion of the vehicle;

After discussing related work in Section 2, we begin in Section 3 by analyzing the key challenges MAVs face when attempting to perform SLAM. We then describe the algorithms employed in our system, highlighting the key enabling technologies that were developed to accomplish mapping with a MAV. Finally, we demonstrate our helicopter navigating autonomously in 3 different unstructured indoor environments.

2 Related Work

In recent years, autonomous flying robots has been an area of increasing research interest. Many capabilities have been developed for autonomous operations in outdoor environments, including high-speed flight through cluttered environments [2], helicopter acrobatics [3], autonomous landing, terrain mapping [4], coordinated tracking and planning of ground vehicles [1], etc. These systems typically take advantage of GPS measurements for state estimation, which are not available indoors.

While some authors [5, 6] have demonstrated indoor flight using GPS simulated from motion capture systems, we seek to develop flying robots that are able to operate autonomously while carrying all sensors used for localization, control and navigation onboard. Other authors [7, 8] use a small number of ultrasound sensors to perform altitude control and obstacle avoidance. Their helicopters are able to take-off, land and hover autonomously; however, they do not achieve goal-directed flight.

There have been numerous efforts to fly helicopters autonomously indoors using monocular camera sensors. [9] performed visual servoing over known Moire patterns to extract the full 6 *dof* state of the vehicle for control, while [10] detects lines in a hallway, and [11] tracked edges in office environments with known structure. While these authors have demonstrated autonomous flight in limited indoor environments, their approaches have been constrained to environments with specific features, and thus may not work as well for general navigation in GPS-denied environments. [12] extracted corner features that are fed into an EKF-based Vision-SLAM framework, building a low-resolution 3D map sufficient for localization and planning. However, an external motion capture system was used to simulate inertial sensor readings.

This paper builds on our previous work in [13], where we present a planning algorithm for a laser-equipped quadrotor helicopter that is able to navigate autonomously indoors with a given map. Here, we extend the work by developing a system that is able to navigate, localize, build maps and explore autonomously *without* a prior map.

Recently, [14, 15] designed helicopter configurations that were similar to the one presented in [13]. [14] scan-matched successive laser scans to hover their quadrotor helicopter, while [15] used particle filter methods to globally localize their helicopter with a precomputed map that was generated by a ground-based robot. However, none of these papers have presented experimental results demonstrating the ability to stabilize all 6 *dof* of the helicopter autonomously using the onboard sensors.

3 MAV-Specific Challenges

In the ground robotics domain, combining wheel odometry with sensors such as laser rangefinders, sonars, or cameras in a probabilistic SLAM framework has proven very successful [16]. Many algorithms exist that accurately localize ground robots in large-scale environments. Unfortunately, mounting equivalent sensors onto a helicopter and using existing SLAM algorithms does not result in the same success. The requirements and assumptions that can be made with flying robots are sufficiently different from those that can be made with ground robots that they must be managed differently.

3.1 Payload

MAVs have a maximum amount of vertical thrust that they can generate to remain airborne, which severely limits the amount of payload available for sensing and computation compared to similar sized ground vehicles. This weight limitation eliminates popular sensors such as SICK laser scanners, large-aperture cameras and high-fidelity IMUs. Instead, indoor air robots rely on lightweight Hokuyo laser scanners, micro cameras and/or lower-quality MEMS-based IMUs, all of which have limited ranges and fields-of-view and are noisier compared to their ground equivalents.

Unlike ground vehicles, air vehicles are unable to measure odometry directly; most SLAM algorithms need these measurements to initialize the estimates of the vehicle's motion between time steps. Although one can obtain relative position estimates by double-integrating acceleration measurements, lightweight MEMS-based IMUs are often subject to biases that can cause the accelerations to drift very quickly, as shown in Figure 3(a). We must therefore obtain relative position estimates measurements by using either visual odometry [17] or laser scan-matching [18, 19] algorithms.

Finally, despite the advances within the community, SLAM algorithms continue to be computationally demanding even for powerful desktop computers, and are therefore not implementable on today's small embedded computer systems that can be mounted onboard indoor MAVs. The computation can be offloaded to a powerful groundstation by transmitting the sensor data wirelessly; however, communication bandwidth then becomes a bottleneck that constrains sensor options. Camera data must be compressed with lossy algorithms before it can be transmitted over wifi links, which adds noise and delay to the measurements. This noise particularly affects feature detectors which look for high frequency information such as corners in an image. Additionally, while the delay can often be ignored for slow-moving, passively-stable ground robots, helicopters have fast and unstable dynamics, making control under large sensor delay conditions impossible.

3.2 Dynamics

The helicopter's fast dynamics result in a host of sensing, estimation, control and planning implications for the vehicle. Filtering techniques such as Kalman Filters are often used to obtain better estimates of the true vehicle state from noisy measurements. Smoothing the data generates a cleaner signal, but adds delay to the state estimates. While delays generally have insignificant effects on vehicles with slow dynamics, the effects are amplified by the MAV's fast dynamics. This problem is illustrated in Figure 3(b), where we compare the normal hover accuracy to the accuracy when the state estimates are delayed by $.2s$. While our vehicle is normally able

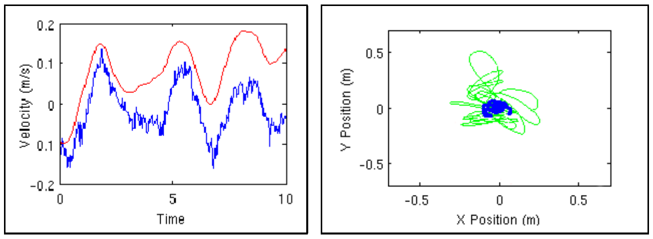


Figure 3: (a) Ground truth velocities (blue) vs. integrated acceleration (red). (b) Comparison of the hover accuracy using PD-control with no delay (blue), PD control with $.2s$ of delay (green).

to achieve an RMS error of $6cm$, with the delay, the error increases to $18cm$.

In addition, as will be discussed in Section 4, the quadrotor is well-modeled as a simple 2^{nd}-order dynamic system with no damping. The underdamped nature of the dynamics model implies that simple proportional control techniques are insufficient to stabilize the vehicle, since any delay in the system will result in unstable oscillations. This effect has been observed experimentally. We must therefore add damping to the system through the feedback controller, which emphasizes the importance of obtaining accurate and timely state estimates for both position and velocity. Traditionally, most SLAM algorithms for ground robots completely ignore the velocity states.

Unlike ground vehicles, a MAV cannot simply stop and perform more sensing when its state estimates contain large uncertainties. Instead, the vehicle will probably be unable to estimate its velocity accurately, and as a result, may pick up speed or oscillate, degrading the sensor measurements further. Therefore, planning algorithms for air vehicles must not only be biased towards paths with smooth motions, but must also explicitly reason about uncertainty in path planning, as demonstrated in [13].

3.3 3D effects

Finally, MAVs operate in a truly 3D environment since they can hover at different heights. The visible 2D slice of a 3D environment can change drastically with height and attitude, as obstacles suddenly appear or disappear. However, if we treat map changes resulting from changes in height and attitude as sensor errors, allowing the map to be updated to account for these changes, we will see that a 2D representation of the environment is surprisingly useful for MAV flight.

4 System Overview

We addressed the problem of autonomous indoor flight as primarily a software challenge, focusing on algorithms rather than exotic hardware. To that end, we used off-the-shelf hardware throughout the system. Our quadrotor helicopter, shown in Figure 1, is the AscTec Hummingbird from Ascending Technologies GmBH[1], and is able to carry roughly $250g$ of payload. We outfitted it with a Gumstix[2] microcomputer, which provides a Wi-Fi link between the vehicle and a ground control station, and a lightweight Hokuyo[3] laser rangefinder for localization. The laser rangefinder provides a $270°$ field-of-view at $40Hz$, up to an effective range of $30m$. We deflect some of the laser beams downwards to estimate height above the ground plane.

The AscTec Hummingbird helicopter is equipped with attitude stabilization, using an onboard IMU and processor to stabilize the helicopter's pitch and roll [20]. This tames the nastiest portions of the quadrotor's extremely fast, nonlinear, and unstable dynamics [6], allowing us to focus on stabilizing the remaining degrees of freedom in position and heading. The onboard controller takes 4 inputs, $\mathbf{u} = [u_\phi, u_\psi, u_t, u_\theta]$, which denote the desired pitch and roll angles, overall thrust and yaw velocities respectively. The onboard controller allows the helicopter's dynamics to be approximated with simple 2^{nd}-order linear equations:

$$\begin{aligned} \ddot{x}^b &= k_\phi u_\phi + b_\phi & \ddot{z} &= k_t u_t + b_t \\ \ddot{y}^b &= k_\psi u_\psi + b_\psi & \dot{\theta} &= k_\theta u_\theta + b_\theta \end{aligned} \tag{1}$$

where $\ddot{x}^b$ and $\ddot{y}^b$ are the resultant accelerations in body coordinates, while k_* and b_* are model parameters that are functions of the underlying physical system. We learn these parameters by flying the helicopter inside a Vicon[4] Motion capture system and fitting parameters to the data using a least-squares optimization method. We also experimented with a dynamics model that includes damping terms,

$$\ddot{s} = k_1 u + k_2 \dot{s} + b \tag{2}$$

However, when fitting this model to the data, we found that $k_2 \approx 0$, confirming pilot experience that the system is underdamped. Using the Matlab® linear quadratic regulator (LQR) toolbox, we then find feedback controller gains for the dynamics model in Equation 1. Despite the model's apparent simplicity, our controller achieves a stable hover with $6cm$ RMS error.

To compute the high-precision, low-delay state estimates needed for such hover performance, we designed the 3-level sensing and control hierarchy, shown in Figure 4, distinguishing processes based on the real-time requirements of their respective outputs. This system was designed as a combination of asynchronous modules, building upon the CARMEN[5] robot navigation toolkit's software architecture. We describe details of the individual modules next.

5 Enabling Technologies

5.1 High-Speed Laser Scan-Matching Algorithm

As discussed in Section 3.1, we cannot directly measure the MAV's odometry; instead, we align consecutive scans from the laser rangefinder to estimate the vehicle's motion. To do this, we developed a very fast and robust laser scan-matching algorithm that builds a high-resolution local map based on the past several scans, aligning incoming scans to this map at the $40Hz$ scan rate. This scan-matching algorithm

[1]Ascending Technologies GmBH. `http://www.asctec.de`
[2]Gumstix Verdex. `http://www.gumstix.com`
[3]Hokuyo UTM-30LX Laser. `http://www.hokuyo-aut.jp`
[4]Vicon Motion Capture Systems. `http://www.vicon.com`
[5]CARMEN. `http://carmen.sourceforge.net`

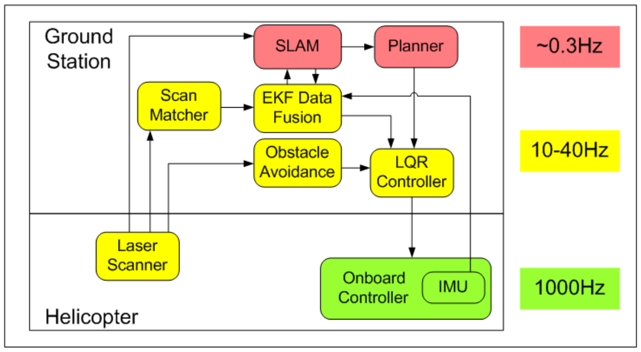


Figure 4: Schematic of our hierarchical sensing, control and planning system. At the base level, the onboard IMU and controller (green) create a tight feedback loop to stabilize the vehicle's pitch and roll. The yellow modules make up the real-time sensing and control loop that stabilize the vehicle's pose at the local level and avoids obstacles. Finally, the red modules provide the high-level mapping and planning functionalities.

is a modified version of the algorithm in [19], and is a key component for closing the loop and performing SLAM on an air vehicle. The algorithm first generates a local likelihood-map from past scans, before finding the optimal rigid body transform that maximizes the likelihood of the current scan. While the scan matching algorithm described below is based on the original implementation in [19], it required several modifications to enable it to be used on the MAV. Specifically, our contributions are:

1. Using a drawing primitive, described in Section 5.1.1, to generate the local map in real-time.
2. Using a different notion of a "local map," adding scans based on insufficient overlap rather than distance traveled.
3. Using image addition primitives to accumulate the pose likelihood map.
4. Adapting the search window based on the maximum expected acceleration of the vehicle.
5. Developing a different method for obtaining a covariance estimate, described in Section 5.1.3.

5.1.1 Local Map Generation

To find the best alignment for an incoming laser scan, one needs a method for scoring candidate poses based on how well they align to past scans. The first challenge in doing this is that laser scanners provide individual point measurements. Successive scans will generally not measure the same points in the environment, which means that attempting to correspond points directly can produce poor results. However, if we know the *shape* of the environment, we can easily determine whether a point measurement is consistent with that shape. We model the shape of the environment as a set of polyline contours. Contours are extracted from the laser readings by an algorithm that iteratively connects the endpoints of candidate contours until no more endpoints satisfy the joining constraints. The algorithm prioritizes joining nearby contours, which allows it to handle partially transparent surfaces such as the railings in the environment depicted by Figure 5(a). Candidate contour merges are scored and stored in a MinHeap data structure, which allows the best candidate to be extracted efficiently. The overall contour extraction algorithm processes a 350-point scan in 0.5ms on modern hardware.

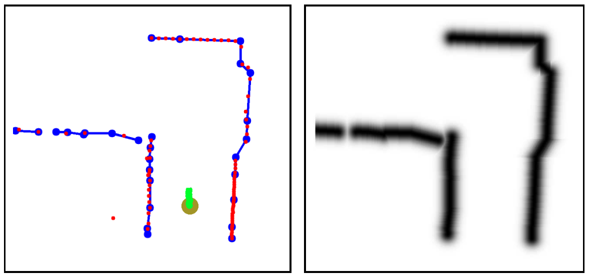


Figure 5: (a) Contours (blue lines) extracted from the raw laser measurements alongside the raw laser readings (red dots). (b) The resulting likelihood map generated from the contours. Darker indicates higher likelihood.

Once we have the set of contours, we can evaluate the likelihood of an alignment between scans. We assume that all range measurements are taken independently, and we compute the likelihood of an alignment as the product of likelihoods for each individual point in the scan. We use a noise model for the laser scanner that approximates the probability of a single lidar point (x, y) as proportional to the distance, d, to the nearest contour C, such that $P(x, y|C) \propto e^{(-d/\sigma)}$, where σ is a variance parameter that accounts for the sensor's noise characteristics. We then pre-compute a likelihood-map where each cell represents the approximate log-likelihood of a laser reading occurring at a given location.

Accurately estimating the *velocity* of the vehicle, where small rounding errors in the position get magnified significantly, requires a high resolution likelihood map that can be difficult to create in real-time. However, if one examines a likelihood map, such as the one shown in Figure 5(b), one quickly realizes that for any reasonable value of σ, the vast majority of cells will be zero. So, while conventional methods compute the value of every cell in the map, and therefore require at least $O(n^2)$ operations, where n is the number of cells along an edge, we developed a likelihood map generation algorithm that exploits the sparsity of the grid map, resulting in a computational complexity of $O(m)$ where $m \ll n^2$ is the number of nonzero cells.

We created a drawing primitive that explicitly draws the nonzero likelihoods by sliding a kernel along the pixels of the input line segment, applying a $\max$ operator between the current map value and the kernel's. Naively using a square kernel, with values set based on $P(x, y|C)$ above would result in cells being modified many times as the kernel slides along the line; however, one can avoid this problem by using a 1 pixel-wide horizontal or vertical kernel, depending on the slope of the line. For lines that are not perfectly horizontal or vertical, this kernel must be widened by $1/cos(s)$, where s is the slope of the line. Creating the likelihood map with this primitive simply requires drawing all the line segments in the extracted contours, which takes around $20ms$ even for extremely large $7.5mm$ resolution likelihood maps.

We create the map from a set of k previous scans, where new scans are added when an incoming scan has insufficient overlap with the current set scans. This creates a locally accurate sliding window where the map contains the recent history around the vehicle, and ensures that all motions within this map will be registered to it in a consistent manner. For example, if the vehicle is hovering in one place, the map will not change, and the position estimates will be drift-free.

5.1.2 Scan-to-Map Alignment

The second task is to find the best rigid body transform (x, y, θ) for each incoming scan with respect to the precomputed likelihood map. Many scan-matching algorithms use gradient descent techniques to optimize these values. However, since the 3D pose likelihood space is often very complicated, even for fairly simple environments, gradient descent is subject to local optima. We chose to use a very robust, if potentially computationally inefficient, exhaustive search over a grid of possible poses. In addition to being extremely robust, computing the likelihoods for the entire grid of poses allows us to easily determine the uncertainty of our match.

While this exhaustive search might initially seem hopelessly inefficient, if implemented carefully, it can be done very quickly. Much of the search time is incurred by transforming the laser scan to the desired pose. However, if we hold θ constant, for a given point, the set of likelihoods associated with each translation is the square window surrounding the point in the likelihood map. This means that the likelihood for all translations can be computed from a single transformation. These windows can be accumulated into the pose likelihood map for an entire scan using the optimized image addition functions available in the Intel Performance Primitives,[6] which provide a factor of 2 speedup.

The optimized exhaustive search implementation makes our method considerably faster than a naive implementation; however we must still ensure that the search area is not too large. Since we do not have wheel odometry with which to initialize the scan matching, we assume that the vehicle moves at a constant velocity between scans. The range of poses that must be searched over can then be selected based on the maximum expected acceleration of the vehicle, which means that at high scan rates, the search volume is manageable.

In our implementation, we use a grid spacing of $7.5mm$ in x, y, and $.15°$ in θ. At this resolution, it takes approximately $5ms$ to search over the approximately $15,000$ candidate poses in the search grid to find the best pose for an incoming scan. Scans that need to be added to the likelihood map are processed in a background thread, allowing pose estimation to continue without impeding the real-time processing path.

[6]Intel Performance Primitives. `http://www.intel.com`

5.1.3 Covariance Estimation

In addition to being very robust, computing the best alignment by exhaustive search has the advantage of making it easy to obtain a good estimate of the covariance by examining the shape of the pose likelihood map around the global optima. This estimate of the covariance is important when we integrate the relative position estimates with other sensors in the data fusion EKF, as described in Section 5.2. While the entire pose likelihood map has many local maxima, it is usually a fairly smooth bell shape in the immediate vicinity of the global optima. If the environment surrounding the vehicle has obstacles in all directions, such as in a corner, the alignment of scans will be highly constrained, resulting in a very peaked likelihood map. On the other hand, if the environment does not constrain the alignment, the map will be nearly flat at the top.

While one could directly fit a multi-variate Gaussian to the 3D pose likelihood map, for simplicity, we compute the covariance in rotation separately from translation. For translation we look at the 2D slice of the pose likelihood map at the optimal rotation. We then threshold this 2D map at the 95^{th} percentile, and fit an ellipse to the resulting binary image. The area and orientation of this ellipse is used as our estimate of the measurement covariance. For the rotation portion, we find the score of the best translation for each rotation, and recover the width of the resulting bell shaped 1D curve.

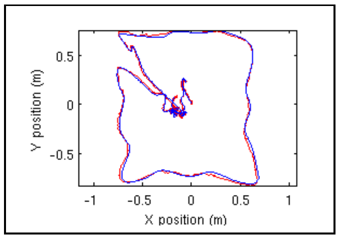
5.2 EKF Data Fusion

The scan matcher outputs the estimated vehicle position (x, y, θ), so to compute the full state estimate, including the velocities, we use an EKF to fuse the scan matcher estimates with the acceleration readings from the IMU. This has several advantages over directly using the position estimates from the scan matcher and their derivatives to control the vehicle. Although the IMU readings drift significantly and are therefore not useful over extended time periods, they are useful over short time periods, allowing us to improve our estimate of the vehicle's velocities. However, the wireless link and scan matcher processing adds a variable delay to the measurements, which can cause problems for controlling the vehicle. In our EKF formulation, we perform the measurement updates asynchronously, while the motion model prediction step is performed on a fixed clock. This allows us to cleanly interpolate the state estimates that are used in the feedback controller.

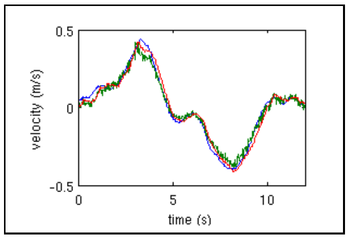
Our filter is a standard EKF, implemented using the open source KFilter library[7]. We use the filter to estimate the positions, velocities, and accelerations of the vehicle, along with the biases in the IMU, resulting in a large state vector. Finding good variance parameters by hand would be a very time-consuming and error-prone task. Instead, we learn the variance parameters using the method described in [21]. By flying the helicopter with the state estimation process running in a motion capture system, we obtain ground-truth values with which to compare our state estimates. This allows us to run stochastic gradient descent to find a good set of variance parameters. The parameter-learning algorithm results in EKF

[7]KFilter. `http://kalman.sourceforge.net`

state estimation that provides significantly improved velocity estimates compared to the variance parameters chosen by hand, as shown in Figure 6(b).



(a)



(b)

Figure 6: (a) Comparison between the position estimated by the onboard sensors (green) with ground truth measurements (blue). (b) Comparison of the ground truth velocity (blue) with the estimate from the EKF before (green) and after (red) optimization.

Figures 6(a) and 6(b) demonstrate the quality of our EKF state estimates. We compared the EKF state estimates with ground-truth state estimates recorded by the motion capture system, and found that the estimates originating from the laser range scans match the ground-truth values closely in both position and velocity. Throughout the 1min flight, the average distance between the two position estimates was less that $1.5cm$. The average velocity difference was $0.02m/s$, with a standard deviation of $0.025m/s$. The vehicle was not given any prior information of its environment (i.e., no map). However, since all the walls in the room were constantly within the laser's field-of-view in this experiment, the SLAM module was not needed to eliminate drift.

5.3 SLAM

We made use of the publicaly available implementation of the GMapping [22] algorithm that is available in the OpenSlam repository[8], which performs slam in 2D. Despite the fact that the helicopter operates in the full 3D environment, the algorithm works surprisingly well and serves as a proof of concept for implementing SLAM on a MAV.

GMapping is an efficient Rao-Blackwellized particle filter which learns grid maps from laser range data. We chose it due to its outstanding accuracy, real-time performance, and its ability to handle changes to the 2D map that occur due to changing height and attitude, as discussed in Section 3.3. While the algorithm worked reasonably well out of the box, we made modifications that improved its performance when used in 3D environments on a MAV. The motion model for the particles in the GMapping algorithm was based on a standard motion model for ground robots with wheel odometry. However, since we use estimates computed by the laser scan matching module, we modified GMapping's motion model to propagate the particles using the uncertainties computed by the scan-matching module.

In addition to the motion model, we modified the map representation so that the map gets updated rapidly in response to changes in height. The algorithm computes the probability that each grid cell is occupied or free based on the number of times a laser beam reflects off, or passes through, the cell. If a particular cell has been hit many times, the algorithm places a very high confidence that the cell is occupied. However, if the helicopter changes heights, and the cell becomes part of free space, this confidence is no longer warranted. Unfortunately the laser must pass through the cell at least as many times as it was hit before the algorithm will be convinced that the cell is actually now free, resulting in a very slow adaptation of the map. Hence, we modified the map representation to cap the maximum confidence for each grid cell, allowing it to change from occluded to free (and vice-versa) more rapidly.

With these modifications, we are able to create large scale maps of the environment such as those shown in Section 6. The algorithm usually takes 1 to 2 seconds to process incoming laser scans, allowing it to be run online, but is not suitable to be directly incorporated into the real-time control loop. Instead, the GMapping algorithm periodically sends position corrections to the data fusion EKF. Since the position corrections are delayed significantly from when the measurement upon which they were based was published, we must account for this delay when we incorporate the correction. This is done by retroactively modifying the appropriate position in the state history. All future state estimates are then recomputed from this corrected position, resulting in globally consistent state estimates. By incorporating the SLAM corrections after the fact, we allow the state estimates to be processed and published with low enough delay to control the MAV, while still incorporating the information from SLAM to ensure drift-free position estimates.

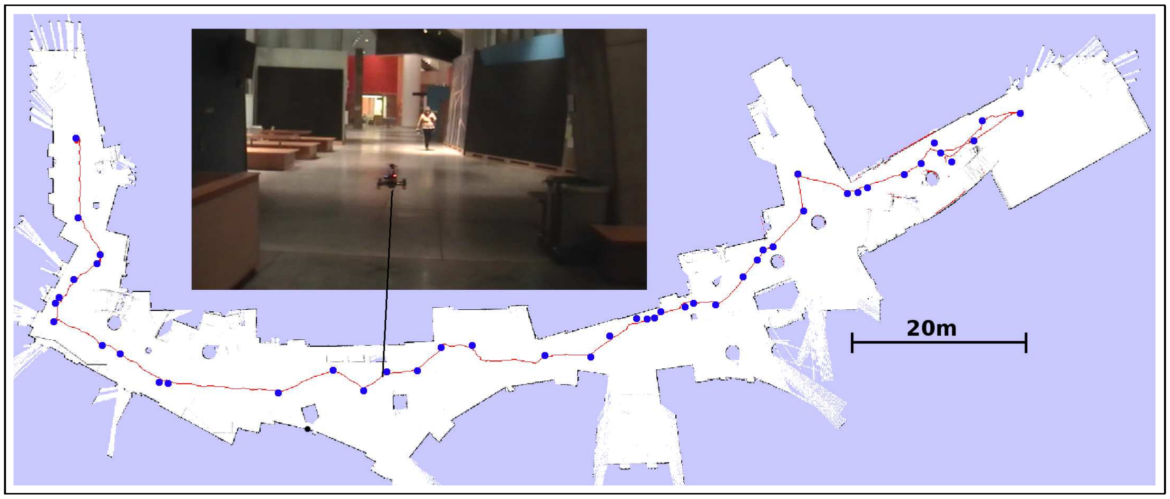
5.4 Planning and Exploration

Finally, to achieve full autonomy, we require a high-level planner that enables the helicopter to either explore or move towards a desired goal autonomously. While exploration has been well-researched in ground robotics, differences between air and ground vehicles, as discussed in Section 3.1, require us to make different considerations when deciding where to go next. In particular, the need to constantly provide control signals to the helicopter means that while we seek to explore the environment, we must ensure that the helicopter is able to remain well-localized and estimate its velocity. We use a modified definition of frontiers [23] to choose positions in free space where the helicopter should fly to next such that it explores previously unexplored regions in the environment. We extend this concept by seeking to find a frontier pose that maximizes both the amount of unexplored space that is expected to be explored and the ability of the helicopter to localize itself. The planner then uses the best frontier as its goal and computes a path using dynamic programming.

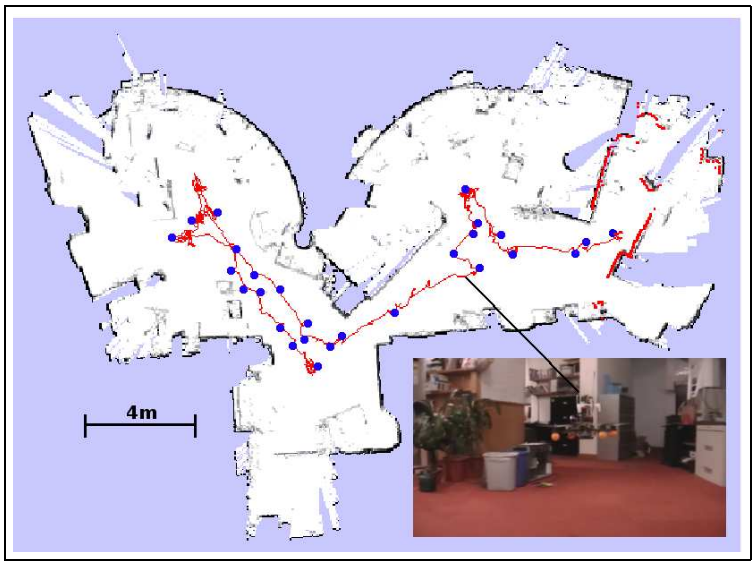
6 Experiments and Results

We integrated the suite of technologies described above to perform autonomous navigation and exploration in unstructured and unknown indoor environments. In this section, we present results demonstrating that the system is capable of fully autonomous operation in a variety of indoor environments. To get a full picture of our system in action, we suggest that the reader also view the videos taken of these experiments available at: `http://groups.csail.mit.edu/rrg/videos.html`.

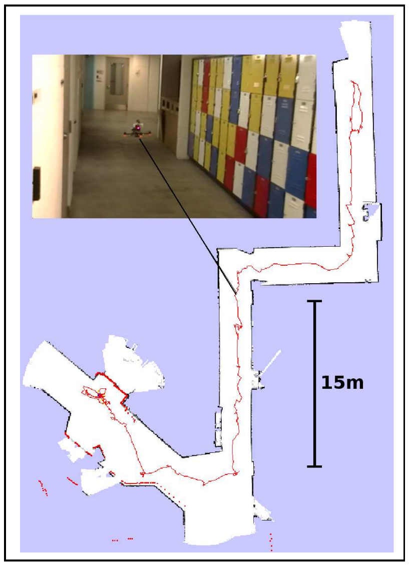
[8]OpenSlam. `http://openslam.org`



(a) Map of MIT Stata Center, 1st Floor.



(b) Map of MIT Stata Center, 3rd Floor.



(c) Map of MIT Stata Center, basement.

Figure 7: (a) Map of the first floor of MIT's Stata center constructed by the vehicle during autonomous flight. (b) Map of a cluttered lab space with significant 3D structure. (c) Map of constrained office hallway generated under completely autonomous exploration. Blue circles indicate goal waypoints clicked by human operator. Red line indicates path traveled based on the vehicle's estimates.

6.1 Autonomous navigation in open lobbies

We flew the vehicle across the first floor of MIT's Stata Center. The vehicle was not given a prior map of the environment, and flew autonomously using only sensors onboard the helicopter. In this experiment, the vehicle was guided by a human operator clicking high-level goals in the map that was being built in real-time, after which the planner planned the best path to the goal. The vehicle was able to localize itself and fly stably throughout the environment, and Figure 7(a) shows the final map generated by the SLAM algorithm at the end of the experiment. During the 8min flight until the battery was exhausted, the vehicle flew a distance of $208.6m$.

6.2 Autonomous navigation in cluttered environments

While unstructured, the lack of clutter along the walls in the lobby environment allowed the 2D map assumption to hold fairly well. We next tested our system by flying through a cluttered lab space (Figure 2, insert of Figure 7(b)), operating close to the ground. At this height, chairs, desks, robots, plants, and other objects in the area caused the 2D cross-sectional scan obtained by the laser rangefinder to vary dramatically with changes in height, pitch, and roll. The resultant SLAM map of the environment is shown in Figure 7(b). The grey features littered within the otherwise free space denote the objects that clutter the environment and are occasionally sensed by the laser rangefinder. Despite the cluttered environment, our vehicle was able to localize itself and maintain a stable flight for 6min over a distance of $44.6m$, a feat that would not have been possible with a static map assumption.

6.3 *Autonomous exploration in office hallways*

Finally, to demonstrate fully autonomous operation of the vehicle, we closed the loop with our exploration algorithm, as discussed in Section 5.4. The helicopter was tasked to explore the hallway environment shown in the insert of Figure 7(c). Once the helicopter took off and began exploring, we had no human control over the helicopter's actions as it autonomously explored the unknown environment. The helicopter continuously searched for and generated paths to areas of new information. Figure 7(c) shows the map built from 7min of autonomous flight, after traveling a distance of $75.8m$.

7 Conclusion

In this work, we have developed a quadrotor helicopter that is capable of fully autonomous exploration in unstructured and unknown indoor environments without a prior map, relying solely on sensors onboard the vehicle. By reasoning about the key differences between autonomous ground and air vehicles, we have created a suite of algorithms that accounts for the unique characteristics of air vehicles for estimation, control and planning. Having developed a helicopter platform that has many of the capabilities of autonomous ground robots, we believe that there is great potential for future extensions of such platforms to operate in fully 3-dimensional environments.

Acknowledgments

Abraham Bachrach was supported by the ARO MAST CTA, Ruijie He was supported by the Republic of Singapore Armed Forces, and Nicholas Roy was supported by NSF Division of Information and Intelligent Systems under grant # 0546467. The authors wish to thank the following people for their support in the project: Giorgio Grisetti, Cyrill Stachniss and Wolfram Burgard provided the GMapping software toolkit. Daniel Gurdan, Jan Stumpf and Markus Achtelik provided the quadrotor helicopter and the support of Ascending Technologies. Edwin Olson for providing us with a reference implementation of his scan-matcher. Samuel Prentice and Garrett Hemann assisted with the development of the hardware.

References

[1] A. Bachrach, A. Garamifard, D. Gurdan, R. He, S. Prentice, J. Stumpf, and N. Roy. Co-ordinated tracking and planning using air and ground vehicles. In *Proc. ISER*, 2008.

[2] S. Scherer, S. Singh, L. Chamberlain, and S. Saripalli. Flying Fast and Low Among Obstacles. In *Proc. ICRA*, pages 2023–2029, 2007.

[3] A. Coates, P. Abbeel, and A.Y. Ng. Learning for control from multiple demonstrations. In *Proc. ICML*, pages 144–151. ACM, 2008.

[4] T. Templeton, D.H. Shim, C. Geyer, and S.S. Sastry. Autonomous Vision-based Landing and Terrain Mapping Using an MPC-controlled Unmanned Rotorcraft. In *Proc. ICRA*, pages 1349–1356, 2007.

[5] J.P. How, B. Bethke, A. Frank, D. Dale, and J. Vian. Real-time indoor autonomous vehicle test environment. *Control Systems Magazine, IEEE*, 28(2):51–64, 2008.

[6] G.M. Hoffmann, H. Huang, S.L. Waslander, and C.J. Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proc. of GNC*, Hilton Head, SC, August 2007.

[7] J.F. Roberts, T. Stirling, J.C. Zufferey, and D. Floreano. Quadrotor Using Minimal Sensing For Autonomous Indoor Flight. In *Proc. EMAV*, 2007.

[8] S. Bouabdallah, P. Murrieri, and R. Siegwart. Towards autonomous indoor micro vtol. Autonomous Robots, Vol. 18, No. 2, March 2005.

[9] G.P. Tournier, M. Valenti, J.P. How, and E. Feron. Estimation and control of a quadrotor vehicle using monocular vision and moirè patterns. In *Proc. of AIAA GNC, Keystone, Colorado*, 2006.

[10] N.G. Johnson. Vision-assisted control of a hovering air vehicle in an indoor setting. Master's thesis, BYU, 2008.

[11] C. Kemp. *Visual Control of a Miniature Quad-Rotor Helicopter*. PhD thesis, Churchill College, University of Cambridge, 2006.

[12] S. Ahrens. Vision-based guidance and control of a hovering vehicle in unknown environments. Master's thesis, MIT, June 2008.

[13] R. He, S. Prentice, and N. Roy. Planning in information space for a quadrotor helicopter in a gps-denied environments. In *Proc. ICRA*, 2008.

[14] G. Angeletti, J.R. P. Valente, L. Iocchi, and D. Nardi. Autonomous indoor hovering with a quadrotor. In *Workshop Proc. SIMPAR*, pages 472–481, 2008.

[15] S. Grzonka, G. Grisetti, and W. Burgard. Towards a navigation system for autonomous indoor flying. In *Proc. ICRA*, 2009.

[16] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Artificial Intelligence*, 128:99–141, 2000.

[17] A. Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *Proc. IROS*, 2008.

[18] Lingemann K., Nchter A., Hertzberg J., and Surmann H. High-speed laser localization for mobile robots. *Robotics and Autonomous Systems*, 51(4):275–296, June 2005.

[19] Edwin Olson. *Robust and Efficient Robotic Mapping*. PhD thesis, MIT, Cambridge, MA, USA, June 2008.

[20] D. Gurdan, J. Stumpf, M. Achtelik, K.M. Doth, G. Hirzinger, and D. Rus. Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. In *Proc. ICRA*, 2007.

[21] P. Abbeel, A. Coates, M. Montemerlo, A.Y. Ng, and S. Thrun. Discriminative training of kalman filters. In *Proc. RSS*, 2005.

[22] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[23] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. CIRA*, 1997.