RANGE - Robust Autonomous Navigation in GPS-denied Environments

Abraham Bachrach
Massachusetts Institute of Technology
Cambridge, MA 02139
abachrac@mit.edu

Samuel Prentice
Massachusetts Institute of Technology
Cambridge, MA 02139
prentice@mit.edu

Ruijie He
Massachusetts Institute of Technology
Cambridge, MA 02139
ruijie@mit.edu

Nicholas Roy
Massachusetts Institute of Technology
Cambridge, MA 02139
nickroy@mit.edu

Published in the Journal of Field Robotics (JFR 2011).

Abstract

This paper addresses the problem of autonomous navigation of a Micro Air Vehicle (MAV) in GPS-denied environments. We present experimental validation and analysis for our system that enables a quadrotor helicopter, equipped with a laser range-finder sensor, to autonomously explore and map unstructured and unknown environments. The key challenge for enabling GPS-denied flight of a MAV is that the system must be able to estimate its position and velocity by sensing unknown environmental structure with sufficient accuracy and low enough latency to stably control the vehicle. Our solution overcomes this challenge in the face of MAV payload limitations imposed upon sensing, computational, and communication resources. In this paper, we first analyze the requirements to achieve fully autonomous quadrotor helicopter flight in GPS-denied areas, highlighting the differences between ground and air robots that make it difficult to use algorithms developed for ground robots. We report on experiments that validate our solutions to key challenges, namely a multi-level sensing and control hierarchy which incorporates a high-speed laser scan-matching algorithm, data fusion filter, high-level SLAM, and a goal-directed exploration module. These experiments illustrate the quadrotor helicopter's ability to accurately and autonomously navigate in a number of large-scale unknown environments, both indoors and in the urban canyon. The system was further validated in the field by our winning entry in the 2009 International Aerial Robotics Competition (IARC), which required the quadrotor to autonomously enter a hazardous unknown environment through a window, explore the indoor structure without GPS, and search for a visual target.

1 Introduction

Many researchers have proposed the use of Micro Air Vehicles (MAVs) as a promising alternative to ground robot platforms for rescue tasks and a host of other applications. MAVs are already being used in several military and civilian domains, including surveillance operations, weather observation, disaster relief coordination, and civil engineering inspections. Enabled by the combination of GPS and MEMS inertial sensors, researchers have been able to develop MAVs that display an impressive array of capabilities in outdoor environments (Scherer et al., 2008; Templeton et al.,

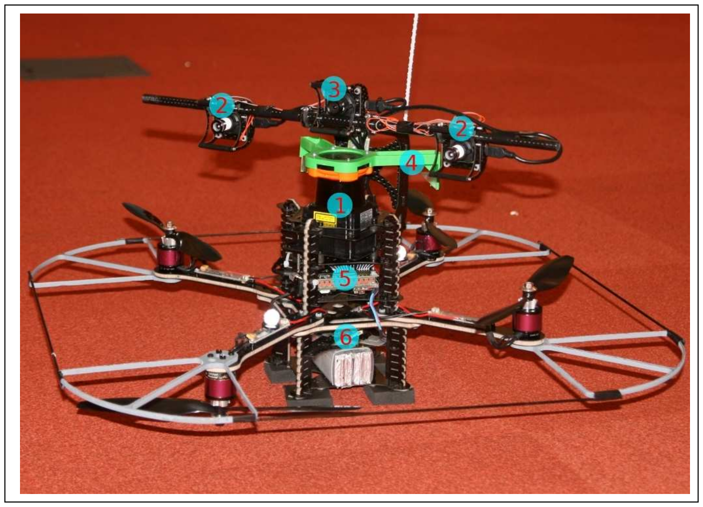


Figure 1: Our quadrotor helicopter. Sensing and computation components include a Hokuyo laser range-finder (1), stereo cameras (2), monocular color camera (3), laser-deflecting mirrors for altitude (4), 1.6GHz Intel Atom-based flight computer (5), and the Ascending Technologies internal processor and IMU(6). The laser scanner and IMU are used for localization. The camera sensors were used for tasks specific to the IARC competition.

2007; He et al., 2010). However, most indoor environments and many parts of the urban canyon remain without access to external positioning systems such as GPS. As a result, previous autonomous MAVs have been limited in their ability to operate in these areas.

In this paper, we describe the estimation, navigation and control system for a MAV operating in GPS-denied environments. We describe experimental assessments of first using onboard sensors to estimate the vehicle's position and secondly using the same sensor data to build a map of the environment around the vehicle, a process generally called simultaneous localization and mapping (SLAM). Although there have been significant advances in developing accurate SLAM algorithms in large-scale environments, these algorithms have focused almost exclusively on ground or underwater vehicles. There have been previous results for performing SLAM on MAVs, (Grzonka et al., 2009; Langelaan and Rock, 2005), however, due to a combination of limited payloads for sensing and computation, coupled with the fast dynamics of air vehicles, the algorithms have generally been tested in simulation, on ground robots, or with sensor data first collected from a manually piloted MAV. Our quadrotor helicopter system, shown in Figure 1, is capable of autonomous flight in unstructured and unknown GPS-denied environments. Developing this capability required careful engineering of a complex system that leverages existing algorithms to balance the trade-offs imposed by GPS-denied flight. We report flight results in a number of domains including indoor flight, outdoor flight through the MIT campus, and results from the International Aerial Robotics Competition 2009.

1.1 Key Challenges

In the ground robotics domain, many algorithms exist for accurate localization in large-scale environments; however, these algorithms are usually deployed on slow moving robots, which cannot handle even moderately rough terrain. MAVs face a number of unique challenges that make developing algorithms for them far more difficult than their indoor ground robot counterparts.

- **Limited Payload** Limited payload reduces the computational power available onboard, and eliminates popular sensors such as SICK laser scanners, large-aperture cameras and high-fidelity IMUs. Our hardware selection discussed in Section 2.1 addresses these tradeoffs.
- **Indirect Position Estimates** While MAVs will generally have an IMU, double-integrating acceleration measurements from lightweight MEMS IMUs results in prohibitively large position errors. We address this

challenge by estimating our position using the laser scan-matcher presented in Section 3.1.

- **Fast Dynamics** MAVs have fast and unstable dynamics which result in a host of sensing, estimation, control and planning implications for the vehicle. Furthermore, MAVs such as our quadrotor helicopter are well-modeled as undamped when operating in the hover regime. The data fusion filter presented in Section 4 combined with the position controller in Section 5 address these challenges.
- **Constant Motion** Unlike ground vehicles, a MAV cannot simply stop and perform more sensing or computation when its state estimates have large uncertainties. Instead, the vehicle is likely to be unable to estimate its position and velocity accurately, and as a result, it may pick up speed or oscillate, degrading the sensor measurements further. These concerns motivate our exploration algorithm in Section 7.

There are further challenges that we do not fully address in this work such as building and planning in 3D representations of the environment. Instead, we treat the large changes in the visible 2D cross section of a 3D environment as sensor noise, requiring the algorithms to be sufficiently robust to handle these changes.

1.2 Related Work

In recent years, the development of autonomous flying robots has been an area of increasing research interest. This research has produced a number of systems with a wide range of capabilities when operating in outdoor environments. For example vehicles have been developed that can perform high-speed flight through cluttered environments (Scherer et al., 2008), or even acrobatics (Coates et al., 2008). Other researchers have developed systems capable of autonomous landing and terrain mapping (Templeton et al., 2007), as well as a host of high level capabilities such as coordinated tracking and planning of ground vehicles (He et al., 2010), or multi-vehicle coordination (Furukawa et al., 2006; Tisdale et al., 2008; Casbeer et al., 2005). While these are all challenging research areas in their own right, and pieces of the previous work (such as the modeling and control techniques) carry over to the development of vehicles operating without GPS, these previous systems rely on external systems such as GPS, or external cameras (Matsuoka et al., 2007) for localization. Similarly, a number of researchers (How et al., 2008; Hoffmann et al., 2007) have flown indoors using position information from motion capture systems, or external cameras (M. Achtelik and Buss, 2009; Altug et al., 2002). In discussing further related work, we focus on flying robots that are able to operate autonomously while carrying all sensors used for localization, control and navigation onboard.

Outdoor Visual Control While outdoor vehicles can usually rely on GPS, there are many situation where relying on GPS would be unsafe, since the GPS signal can be lost due to multi-path, satellites being occluded by buildings and foliage, or even intentional jamming. In response to these concerns, a number of researchers have developed systems that rely on vision for control of the vehicle. Early work in this area by Saripalli et al. (2003) and Buskey et al. (2004) used a stereo camera to enable position hold capabilities. Other researchers have developed capabilities such as visual servoing relative to a designated target (Mejias et al., 2006), landing on a moving target (Saripalli and Sukhatme, 2007), and even navigation through urban canyons (Hrabar and Sukhatme, 2009). While the systems developed by these researchers share many of the challenges faced by indoor or urban MAVs, they operate on vehicles that are orders of magnitude larger, with much greater sensing and computation payloads. In addition, the outdoor environments tend to be much less cluttered, which gives greater leeway for errors in the state estimation and control.

Indoor Obstacle Avoidance Using platforms that are of a similar scale to the ones targeted in this paper, several researchers (Roberts et al., 2007; Bouabdallah et al., 2005; Matsue et al., 2005) use a small number of ultrasound or infrared sensors to perform altitude control and basic obstacle avoidance in indoor environments. While their MAVs are able to hover autonomously, they do not achieve any sort of autonomous goal-directed flight that would enable the systems to be controlled at a high level such that they could be built upon for more advanced autonomous applications.

Known Structure Instead of using low-resolution sonar and infrared sensors, several authors have attempted to fly MAVs autonomously indoors using monocular camera sensors. To enable tractable vision processing, this work has typically made strong (and brittle) assumptions about the environment. For example, Tournier et al. (2006) performed visual servoing over known Moire patterns. Kemp (2006) fit lines in the camera images to the edges of a 3D model of an office environment with known structure. Reducing the prior knowledge slightly, Johnson (2008) detected lines in a hallway, and used the assumption of a straight hallway to infer the vehicle pose. Similarly, Celik et al. (2008) developed the MVCSLAM system, which tracks corner features along the floor of a hallway. It is unclear how their work could be extended to other less structured environments. Their applicability is therefore constrained to environments with specific features, and does not allow for general navigation in GPS-denied environments.

Using a 2D laser scanner instead of a camera, prior work in our group (He et al., 2008) presented a planning algorithm for a quadrotor helicopter that is able to navigate autonomously within an indoor environment for which there is a known map. Recently, Angeletti et al. (2008) and Grzonka et al. (2009) designed quadrotor helicopters that were similar to the one presented by He et al. (2008). Angeletti et al. matched incoming laser scans to a known map to hover a quadrotor helicopter, while Grzonka et al. used particle filter methods to localize a MAV in a map built by a ground robot. However, none of these papers presented experimental results demonstrating the ability to stabilize all 6 degrees of freedom of the MAV using the onboard sensors, and all made use of prior maps, an assumption that is relaxed in this work.

Indoor SLAM There is a very large amount of prior work on performing SLAM on ground vehicles (Grisetti et al., 2007; Leonard and Durrant-Whyte, 1991; Lu and Milios, 1997; Thrun and Montemerlo, 2006). These algorithms enable the vehicles to localize themselves and build maps in large scale environments, however they are too slow to directly provide the real time state estimates required for controlling a MAV.

Ahrens et al. (2009) used monocular vision SLAM to stabilize the position of a quadrotor helicopter. Extracted corner features were fed into an extended Kalman filter based vision-SLAM framework, building a low-resolution 3D map sufficient for localization and planning. An external motion capture system was used to simulate inertial sensor readings, instead of using an onboard IMU. As such, their system was constrained to the motion capture volume where they had access to the high quality simulated IMU. Adopting a slightly different approach, Steder et al. (2007) mounted a downward-pointing camera on a blimp to create visual maps of the environment floor. While interesting algorithmically, this work does not tackle any of the challenges due to the fast dynamics of other MAVs. Similar work by Blosch et al. (2010) extended monocular vision SLAM to a quadrotor helicopter using lower quality acceleration estimates from a more realistic MAV-scale IMU, but this work also uses a downward-pointing camera and makes strong assumptions about the environment.

The system presented in this paper was also discussed in Bachrach et al. (2009a,b). Here we present a more detailed analysis and evaluation of the algorithms, as well as significantly expanded experimental results in the field, including new results in large-scale indoor environments and outdoor flight in the urban canyon. We also present the results from our team's winning entry in the 2009 AUVSI International Aerial Robotics Competition.

2 System Overview

To compute the high-precision, low-delay state estimates required for indoor and urban flight, we designed a 3-level sensing and control hierarchy, grouped by color in Figure 2, distinguishing processes based on the real-time requirements of their respective outputs. This hierarchical architecture is one of the key features of our system that enables flight in GPS-denied environments. By dividing the problem into the local and global parts that can be solved separately, we allow the local computation to run in real-time with the limited computational resources available on the MAV.

The first two layers run in real-time onboard the vehicle, and are responsible for stabilizing the vehicle and performing low-level obstacle avoidance. At the base level of the hierarchy (represented in green in Figure 2), the onboard IMU and processor developed by Ascending Technologies GmbH[1] creates a very tight feedback loop to stabilize the MAV's pitch and roll, operating at 1000Hz. All of our processes control the vehicle by interacting with this control loop. At

[1]Ascending Technologies GmbH. `http://www.asctec.de`

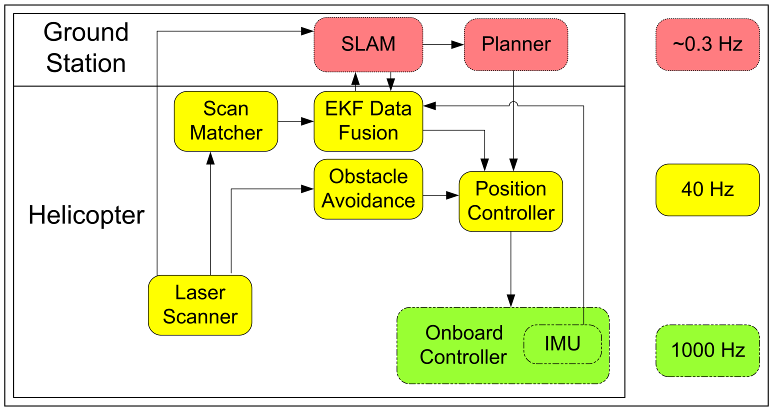


Figure 2: Schematic of our hierarchical sensing, control and planning system with the layers distinguished by color. At the base level, the onboard IMU and controller (green) create a tight feedback loop to stabilize the MAV's pitch and roll. The yellow modules make up the real-time sensing and control loop that stabilizes the MAV's pose at the local level and avoids obstacles. Finally, the red modules provide the high-level mapping and planning functionalities.

the next level (represented in yellow in Figure 2), a fast, high-resolution laser scan-matching algorithm, described in Section 3.1, estimates the vehicle's relative motion, while an Extended Kalman Filter-based data fusion module (Section 4) fuses the estimates with the IMU measurements to provide accurate, high frequency estimates of the vehicle state including velocity. These estimates enable the position controller to hover the MAV in small, room sized environments (Section 5). A simple obstacle avoidance module ensures that the MAV maintains a minimum distance from observed obstacles.

The third layer (represented in red in Figure 2) runs on a ground station computer and contains the modules responsible for creating a consistent global map of the world, as well as planning and executing high level actions. A SLAM algorithm (Section 6) uses the data fusion filter's state estimates and incoming laser scans to create a global map, ensuring globally consistent state estimates. Since the SLAM algorithm takes 1-2 seconds to incorporate incoming scans, it is not part of the real-time feedback control loops at the lower levels. Instead, it provides delayed correction signals to the data fusion filter, ensuring that our real-time state estimates remain globally consistent. Finally, a planning and exploration module (Section 7) enables the vehicle to plan paths within the map generated by the SLAM module, and guide the vehicle towards unexplored regions.

We implemented the hierarchical software system shown in Figure 2 as a set of independent processes which, communicate using the Lightweight Communications and Marshalling (LCM) library[2] (Huang et al., 2009). An 802.11n WiFi module provides a wireless link to the ground-station. The high bandwidth link allows real-time processing to be performed either on or offboard the vehicle; however, moving computation onboard reduces the delay, and makes the vehicle less susceptible to failures due to loss of the wireless connection. As a result, we run all of the real-time state estimation and control modules onboard the vehicle, while the more computationally intensive non-real-time processes such as the SLAM and planning modules are run at the ground-station.

2.1 Hardware Platform

Our system is built using consumer off-the-shelf (COTS) hardware throughout. We use the Pelican quadrotor helicopter designed by Ascending Technologies, which provided an extremely robust, stable, and safe platform for our experiments. Our system uses the Ascending Technologies attitude controller to stabilize the pitch and roll of the vehicle, as well to provide filtered measurements from its IMU at 100Hz. Since the attitude filter does not have access to the velocity estimates of the vehicle, it is unable to effectively estimate the biases in the accelerometers. As a result, we estimate these biases in our data fusion filter described in Section 4.

[2]LCM - Lightweight Communication and Marshalling. `http://code.google.com/p/lcm/`

The Pelican is able to carry roughly 750g of payload, which allows a relatively large sensing and computation payload for a vehicle of its size (maximal dimension of 70cm). The vehicle has a 15 minute endurance with a 6000mAh lithium polymer battery. We outfitted the vehicle with a lightweight Hokuyo[3] UTM-30LX laser range-finder. The laser range-finder provides a $270°$ field-of-view at 40Hz, up to an effective range of 30m indoors. We deflect some of the laser beams downwards using a right angle mirror to estimate the vehicle's height, while the remaining beams are used for localization. In bright sunlight, the range of the laser scanner is considerably reduced, down to an effective range of around 15m, however the experiments in Section 8.3 were performed at night.

In addition to the laser scanner and IMU, the vehicle is capable of carrying a monocular color camera and a set of grayscale stereo cameras. The cameras were used for mission-specific tasks during the International Aerial Robotics Competition (Section 9), however they are not currently used in the core navigation system described in this paper.

The onboard computer is based around a 1.6GHz Intel Atom processor with 1GB of RAM. The computer is powerful enough to allow all of the real-time processing to be performed onboard the vehicle.

3 Relative Motion Estimation

One of the major challenges that we identified in operating a MAV is estimating the position and velocity of the vehicle from on-board sensor data, and we address this challenge using the laser range-finder. While the quadrotor helicopter moves in SE(3), with the vehicle's pitch and roll controlled by the Ascending Technologies attitude controller, the motion can be well approximated as motion in SE(2) × R $(x, y, \text{yaw} \times z,)$. This allows us to decouple the motion estimation into two separate problems: estimation of the motion in SE(2) using laser scan-matching (Lu and Milios, 1997), and estimation of the height of the vehicle using the algorithm described in Section 3.2. To obtain velocity estimates for the vehicle, we need to differentiate the computed motion estimates. As a result, the algorithms must provide both high-resolution matching and fast real-time performance.

3.1 Robust High-Speed Probabilistic Scan-Matching

The laser scan-matching algorithm must solve the following problem: given two overlapping laser range scans $S_t, S_{t-1} \in \Re^{2\times n}$ each consisting of n distinct 2D points $\{x_i\} \in \Re^2$, find the optimal rigid body transform Δ that aligns the current laser scan with the previous scan such that applying the transform Δ to S_{t-1}, denoted $\Delta \otimes S_{t-1}$, results in a scan that matches S_t.

Rather than explicitly match pairs of scans S_{t-1} and S_t, we match each scan to an existing map M containing the data from a previous set of scans $S_{t-\tau:t-1}$. We use probabilistic scan-matching algorithms due to their robustness to large discontinuities in the range measurements experienced by the vehicle as it changes height and attitude in complex 3D environments. Changes in range measurements due to pitching and rolling can be partly addressed by projecting the laser rays using the attitude estimate from the IMU. However, common scan-matching algorithms such as iterative closest point (ICP; Zhang, 1994; Censi, 2008) are still likely to fail in such situations as they attempt to explicitly find correspondences for *all* points in the scans even though a number of points may not correspond to the currently visible cross section of the environment. The poor performance of an ICP based scan matching method[4] on data from a MAV is shown in Table 1.

In map-based probabilistic scan matching, a grid map M is created from previous scans, and incoming scans are matched against that map. Each cell in the map stores the likelihood of a laser return being measured at that point. Assuming that each of the point measurements in a laser scan are independent, the likelihood for an entire scan can be computed as

$$P(S_t|M) = \prod_{i=1}^{n} P(x_i|M), \tag{1}$$

[3]Hokuyo UTM-30LX Laser. `http://www.hokuyo-aut.jp`
[4]The C(anonical) Scan Matcher `http://andreacensi.github.com/csm/`

where $P(x_i|M)$ represents the probability of measuring point point $x_i \in S_t$ at that location in the map. The map then allows us to search over candidate rigid body transforms Δ (a 3D search space in (x, y, yaw)) to find the Δ^* that maximizes the likelihood of the measured laser scan,

$$\Delta^* = \underset{\Delta}{\text{argmax}} P(\Delta \otimes S_t|M), \tag{2}$$

where $\Delta \otimes S_t$ is the set of laser points S_t transformed by the rigid body transform Δ.

The two major design components of a map-based probabilistic scan-matching algorithm are the generation of a map from scans $S_{t-\tau:t-1}$ that allows us to compute the likelihood of a point x_i, and an alignment search procedure that allows us to compute Δ^*. Our algorithm uses Olson (2008)'s map representation, however after experimentally characterizing the performance of the search approaches, we chose to employ the coordinate ascent search strategy used by Haehnel (2004)[5]. When performing scan-matching at high scan rates, we found that the initial estimate from a constant velocity motion model is good enough for hill climbing to find the global optimum. This is in contrast to situations where scan-matching is used for loop closure detection, as demonstrated by the experiments performed by Olson (2009).

3.1.1 Likelihood Map Generation

A common approach to map generation from laser scans is to store all points x_i from previous scans, and compute the likelihood of each point in the new scan from the distance to the nearest point in the previous scans. However, many indoor and urban environments are made up of planar surfaces with a 2D cross section that is a set of piecewise linear line segments. The points in successive scans will generally not correspond to identical points in the environment due to the motion of the laser scanner, but will often measure points on the same surface. As a result, attempting to correspond points explicitly can produce poor results.

We therefore match the points of each new scan to contours extracted from previous scans to provide more robust matches. We build a map of likelihood contours by initializing the map with individual points, and then iteratively joining contours until no more contours satisfy a set of joining constraints. The algorithm prioritizes joining nearby contours, which allows it to handle partially transparent surfaces such as the railings in the environment depicted in Figure 3(a). The contour extraction algorithm can be implemented efficiently by storing candidate contour merges in a priority queue, sorted by the distance between the endpoints of the candidate contours. The overall contour extraction algorithm takes 0.5ms to process a 350 point scan on the onboard computer.

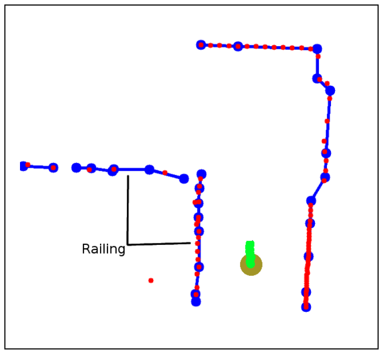
Laser range-finders provide noisy measurements of the range and bearing to obstacles in the environment. While each of these degrees of freedom has an independent noise term, we assume a radially symmetric sensor model for simplicity. Our noise model approximates the probability of a single lidar point x_i as proportional to the distance $d(x_i, C)$ to the nearest contour C, such that

$$P(x_i|C) \propto e^{(-d(x_i,C)/\sigma)}, \tag{3}$$

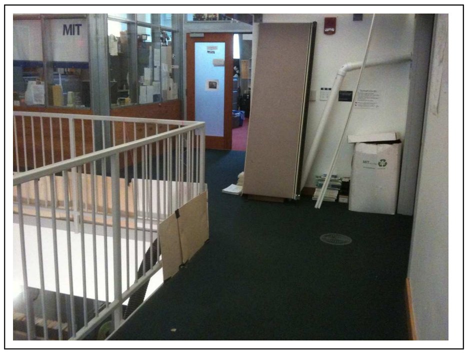
where σ is a parameter that accounts for the length scale of the sensor noise. To enable efficient evaluation of scan likelihoods, we precompute a grid-map representation where each cell stores the approximate log-likelihood of a laser reading being returned from a given location.

Map Resolution and Velocity Estimation For most ground robotics applications, a map resolution of 10cm or more is often sufficient. However, to compute the *velocity* of the vehicle, we must differentiate the position estimates from the scan-matcher, which means that rounding errors due to the discretization of the state space are amplified. As a result, we require a much higher resolution map. The resolution of the map has a direct effect on the accuracy of the scan-matcher velocity estimates as shown by Figure 4(b). With a map resolution of 10cm, the scan-matcher estimates have an RMS error of almost 0.5m/s, which is a very significant amount of noise for a vehicle that is attempting to hover. While the velocity estimate computed with a 10cm map resolution (red line in Figure 4(a)) could be improved by low-pass filtering, this would induce significant delay.

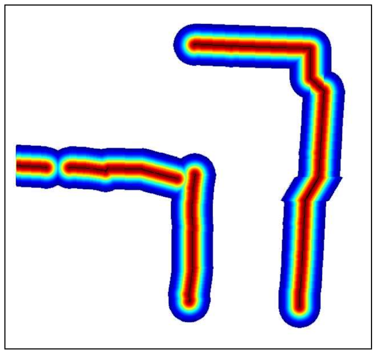
[5]Vasco Scan Matcher in CARMEN robotics toolkit. `http://carmen.sourceforge.net`



(a)



(b)



(c)

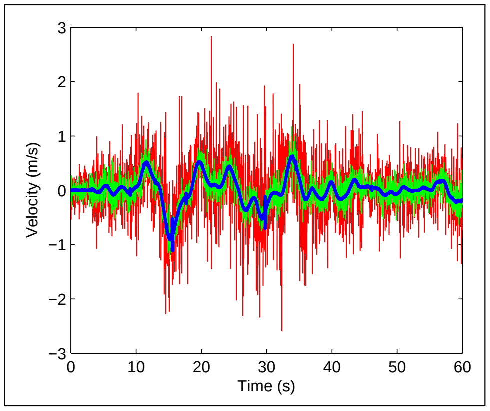
Figure 3: (a) Contours (blue lines) extracted from the raw laser measurements alongside the raw laser readings (red dots). Notice how the contour extraction algorithm handles the partially transparent railing on the left. A picture of the environment in which the laser scan was taken is shown in (b). The resulting likelihood map generated from the contours is shown in (c). Brighter colors (red) indicates higher likelihood.

Algorithm 1 Compute The Likelihood Map

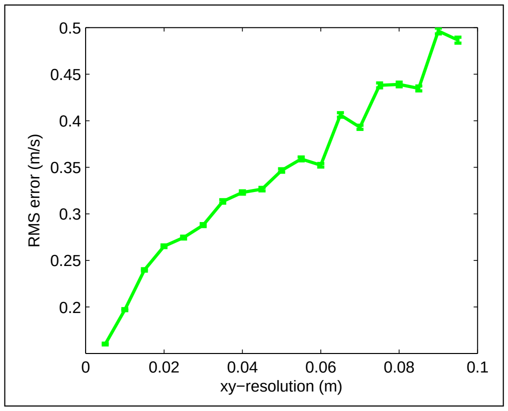
```
Require: set of piecewise linear contours $C$
Require: set of kernels $K$ indexed by slope
  create grid map $M$ allocated with enough space for contours
  for each line segment $s \in C$ do
    select kernel $k \in K$ based on slope of $s$
    for each pixel $p$ along segment $s$ do
      for each pixel offset $o$ in kernel $k$ do
        $M(p_i + o_i, p_j + o_j) = \max(M(p_i + o_i, p_j + o_j), k(o_i, o_j))$
      end for
    end for
  end for
  return $M$
```

Fast Likelihood Map Computation Generating such a high-resolution map is computationally intensive. However, if one examines a likelihood map such as the one shown in Figure 3(c), one quickly realizes that for any reasonable value of σ, each x_i will have zero probability for the vast majority of map cells. In order to create the high-resolution likelihood maps in real-time, we developed a drawing primitive that explicitly "draws" the non-zero likelihoods around each line segment. The drawing primitive operates by sliding a 1-pixel wide vertical or horizontal kernel along the cells spanned by a line segment (Bresenham, 1965), applying a $\max$ operator between the current map value and the kernel's value. The appropriate kernel is chosen based on the slope of the line, with the kernel values set according to Equation 3. The algorithm for rendering the likelihood map is shown in Algorithm 1. As a final optimization, the inner loop of the algorithm can be performed using optimized matrix libraries. The drawing primitive enables us to render the likelihood maps for normal sized environments in 20ms on the onboard computer. For comparison, the map generation routines due to Olson (2009) takes 200ms at the same resolution.

Sliding Window Local Maps Scan-matching algorithms generally operate on consecutive pairs of scans, without maintaining any history. However, any small errors in the matching will be retained and integrated into the current position estimate, resulting in drift. Instead of matching each pair of incoming scans, we create a sliding window local map constructed from a non-consecutive but overlapping set of previous scans that extend beyond the immediate field of view of the sensor. Matching against this map provides more accurate position estimates due to the fact that fewer scans get added into the representation, and the position estimates will be locally consistent, and drift-free as long as the vehicle navigates within the map. The reduced drift is demonstrated in Table 1, where the scan-to-scan matching results in significantly larger position error.



(a)



(b)

Figure 4: Demonstration of the effect of map resolution on the velocity estimates computed by the scan-matcher. (a) The ground truth velocity over time (blue), with the velocities estimated by the scan-matcher with a map resolution of 1cm (green) and 10cm (red), which have an RMS error of 0.2m/s and 0.5m/s respectively. (b) A plot of the RMS error in velocity as a function of the map resolution. Note, that a map resolution of 10cm or more is common for ground robots. Ground truth was given by an external motion capture system.

Method	Time Per Scan	RMS Vel Error	Position Error
Map-Based Coordinate Ascent	12.5 ms	0.197 m/s	1.05 m, 0.41°
Map-Based Multires Grid Search	16.0 ms	0.279 m/s	2.33 m, 0.46°
Map-Based scan-to-scan	32.5 ms	0.197 m/s	5.64 m, 3.35°
Iterative Closest Point	15.8 ms	0.198 m/s	10.3 m, 12.3°

Table 1: Comparison of the coordinate ascent and multi-resolution grid search methods, as well as the effect of the sliding window local map. All three methods are tested with a map resolution of 1cm. The exhaustive search method used a step size of $0.5°$ in yaw. The position error represents the accumulated error after closing the first loop of the Killian Court dataset in Figure 9, a 300m long trajectory. The robustness of our methods result in greatly reduced position error compared to the ICP implementation from Censi (2008).

Additionally, when the MAV changes height, new contours will appear and be added into the map. By including all contours induced by planes in the environment but only aligning against the best subset, the vehicle can remain well localized even in situations where oscillations in the attitude of the vehicle cause the incoming scan to jump back and forth between multiple surfaces.

We add contours to the local map when the fraction of points in an incoming scan that are given a high likelihood in the current map drops below a threshold. The threshold is set high enough that incoming scans are still able to be matched accurately, but low enough that scans are not added too often. We experimentally determined that a threshold of 75% gave a good trade-off, adding a scan to the map approximately every 2 seconds during normal operation.

3.1.2 Scan-to-Map Alignment

Given a likelihood map constructed from contours as described above, we now need a procedure to search for the most likely alignment of a new scan to the map. We experimented with both co-ordinate ascent and grid search methods to find the aligning rigid body transform Δ^* with respect to the precomputed map. While grid search might initially seem hopelessly inefficient for large search windows, it can be performed very quickly by structuring the computation appropriately (Olson, 2009).

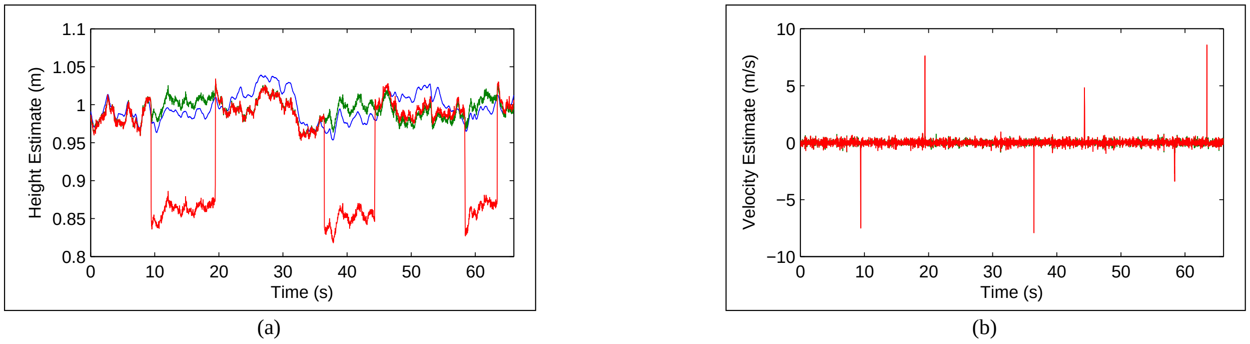


Figure 5: (a) A plot showing a comparison of the height estimates from our algorithm (green) alongside the raw measurements (red), and the ground truth (blue). (b) A plot showing the velocity estimates from differentiating the raw measurements. The large discontinuities in the red lines occurred when the vehicle flew over the edge of a 15cm tall object. Our algorithm produces position estimates with an RMS error of 2cm, while the velocity estimates have an RMS error of 0.2m/s.

We compared the two approaches (grid search and co-ordinate ascent) for performing the alignment search using logged data from the vehicle in two environments. In the first experiment, we flew the vehicle inside of a motion capture environment which gave us access to ground truth data alongside the laser measurements. The ground truth data allowed us to compare the velocity estimates produced by differentiating the position estimates. In the second experiment, we flew the vehicle around a large 300m long loop, which allowed us to measure the total accumulated error in position by using the scan matcher to register the final scan to first measurement at the starting point. We then computed the motion estimates with the scan-matcher using the two alignment search methods described above. As can be seen in Table 1, while both search methods are efficient enough to run in real-time (40Hz) on the 1.6Ghz Atom processor onboard the vehicle, the coordinate ascent search outperforms the grid-search both in terms of speed, and accuracy. This indicates that the additional robustness of grid search is not needed at the high scan rates used in our system. The improved accuracy is due to the sub-cell resolution enabled by coordinate ascent.

In our implementation, we use a map resolution of 1cm. At this resolution, it takes approximately 12.5ms to search for the aligning pose using coordinate ascent. Rebuilding the map when a scan needs to be added is an aperiodic job that would otherwise disrupt the real-time state estimation. As a result, we regenerate the likelihood map in a background computational thread, allowing state estimation to continue unimpeded.

3.2 Laser Based Height Estimation

We use the onboard IMU of the vehicle to estimate the roll and pitch of the vehicle, and the scan-matching uses measurements from a planar laser range-finder to estimate the (x, y, yaw) degrees of freedom. To estimate altitude, the remaining degree of freedom of the vehicle, we redirect a small portion of the field of view of the laser downward using a right-angle mirror in order to estimate the height of the vehicle. Only 20 beams are redirected toward the floor, so there is insufficient data to disambiguate the motion of the vehicle from changes in the height of the floor. If we assume that the vehicle is flying over a flat floor, we can directly use the average range (rejecting outliers) measured by the redirected beams of the laser scanner, r_t, as an estimate of the absolute height of the vehicle at time t. However, if the vehicle flies over an object such as a table, these height estimates will be incorrect. To make matters worse, flying over the table will appear as a large step discontinuity in the height estimate, as shown by the red line in Figure 5(a), which can result in aggressive corrections from the position controller.

However, if we look at the velocity v_t computed from the difference between consecutive measurements we see that there are very large outliers that occur when the vehicle flies over an object. As a result, we can use the maximum expected acceleration $a_{\max}$ of the vehicle to filter out these outliers, allowing us to obtain accurate estimates of the z-velocity with an RMS error of 0.2m/s.

Directly integrating the estimated velocity $\hat{v}_t$ would have a potentially unbounded drift. To eliminate this drift, we assume that the vehicle will predominantly be flying over a flat floor, but sometimes will fly over furniture or other objects. With this assumption, the range measurements from the downward pointed laser beams accurately measure the height of the vehicle, subject to short lived local disturbances.

We combine the range-to-floor and velocity estimates using an approach inspired by complementary filtering (Zimmerman and Sulzer, 1991). The range measurements, r_t, are treated as the globally accurate "low frequency" information, while the filtered velocity estimate, $\hat{v}_t$, is treated as the locally accurate "high frequency" information. However, since the amount of time that a vehicle will spend above an object is unknown, we cannot use a frequency criteria to combine the two signals. We instead use the distance the vehicle has traveled since it measured a discontinuity in the surface underneath it.

Algorithm 2 Laser Height Estimation

```
Require: $r_t, r_{t-1}$      //current and previous redirect range measurements
Require: $\hat{h}_{t-1}$      //previous height estimate
Require: $v_{t-1}$      //previous velocity estimate
Require: $d$      //linear distance to previous discarded measurement
  $v_t = (r_t - r_{t-1})/dt$
  if $|v_t - v_{t-1}| > a_{\max}$ then
    $h' = \hat{h}_{t-1}$
    $d = 0$
  else
    $h' = \hat{h}_{t-1} + (r_t - r_{t-1})$
  end if
  $\hat{h}_t = h' + \alpha\,\mathrm{sgn}(r_t - h')/(1 + exp(\sigma_d(c - d)))$
  return $\hat{h}_t, d, v_t$
```

The height estimation process is shown in Algorithm 2. The parameters σ_d and c control the width and center of the logistic function used to smoothly apply corrections. The parameters are chosen such that we expect the vehicle to no longer be above an object when the larger corrections are being applied ($\sim$ 2m). The current height estimate can be arbitrarily far away from the measured range to the floor, so we use the sign of the the error in the correction, ignoring the magnitude. The correction scaling term α is set such that the maximum correction per time step is small enough (1cm/s) to induce smooth motions of the vehicle.

4 Data Fusion Filter

To control the vehicle, we require accurate estimates of the MAV position and velocity. We compute these estimates by fusing the estimated change in position computed by the scan-matching algorithm described in the previous section with the acceleration measurements from the IMU. While the IMU readings drift significantly, they are useful over short time periods and allow us to improve our estimate of the vehicle velocities.

The data fusion filter provides us with a way to trade off the reliance on each sensor and fuse the different sources of information which arrive asynchronously, and at different rates. We estimate the state of the MAV at time step t, denoted by $\mathbf{x}_t$, given the set of IMU measurements $\{\mathbf{z}_1^I, \mathbf{z}_2^I \ldots \mathbf{z}_m^I\}$ and measurements computed from the laser data $\{\mathbf{z}_1^L, \mathbf{z}_2^L \ldots \mathbf{z}_n^L\}$ as:

$$P(\mathbf{x}_t|\mathbf{x}_1 \ldots \mathbf{x}_{t-1}, \quad \mathbf{z}_1^I \ldots \mathbf{z}_m^I, \quad \mathbf{z}_1^L \ldots \mathbf{z}_n^L) \tag{4}$$

We adopt the standard practice for fusing GPS and an IMU measurements on outdoor UAVs (Jun et al., 1999), which employs an Extended Kalman Filter (EKF) to estimate the vehicle state from noisy GPS and IMU measurements. However, instead of absolute position estimates provided by GPS, the scan-matcher provides relative position estimates Δ_t, which can be viewed as noisy measurements of the transformation between the states at two points in time $\mathbf{x}_t$

and $\mathbf{x}_{t-k}$. These relative measurements make the exact solution of the inference problem significantly more difficult. While this model could be solved using smoothing techniques such as those proposed by Ranganathan et al. (2007), for ease of implementation we simplified the model by incorporating the relative scan-matcher measurements conditioned on $\mathbf{x}_{t-k}$, that is the scan-matcher estimates are modeled as absolute measurements of the vehicle's position given the most likely originating state estimate.

$$\mathbf{z}_t^L = \Delta_t | \mathbf{x}_{t-k} \tag{5}$$

We then use a standard EKF formulation to solve the inference problem and provide estimates of the vehicle state. Our filter was implemented using the open source KFilter library[6]. The filter to estimates the positions, velocities, and accelerations of the vehicle, along with biases in the IMU. The state is represented as:

$$\mathbf{x} = [x, y, z, \dot{x}^b, \dot{y}^b, \dot{z}^b, \ddot{x}^b, \ddot{y}^b, \ddot{z}^b, \phi, \theta, \psi, b_{\ddot{x}}, b_{\ddot{y}}, b_{\ddot{z}}, b_{\phi}, b_{\theta}] \tag{6}$$

The position (x, y, z) is represented in the global coordinate frame, with the origin placed where the vehicle is initialized. The orientation (ϕ, θ, ψ) or (roll, pitch, yaw) is represented using Euler angles. The velocities and accelerations are represented in the body frame, where the origin is located at the center of body with the x-axis pointing towards the front of the vehicle, y to the left, and z up. We estimate the biases b_* in the accelerometers, roll and pitch of the IMU. We rely on the Ascending Technologies IMU to provide the attitude estimate, so we do not estimate the angular velocities or the biases in the gyros. In addition, since we use the scan-matcher to estimate the heading of the vehicle, we do not estimate a bias for yaw.

4.1 Filter Process Model

We estimate the heading of the vehicle separately from the pitch and roll. This in turn makes the z axis independent of the x and y axes. As a result, the nonlinear state update equations for x and y are:

$$\begin{aligned} x_t &= x_{t-1} + dt(\dot{x}^b_{t-1} \cos(\psi_{t-1}) - \dot{y}^b_{t-1} \sin(\psi_{t-1})) + \omega_x, & \omega_x &\sim N(0, \sigma_x) \\ y_t &= y_{t-1} + dt(\dot{x}^b_{t-1} \sin(\psi_{t-1}) + \dot{y}^b_{t-1} \cos(\psi_{t-1})) + \omega_y, & \omega_y &\sim N(0, \sigma_y) \end{aligned} \tag{7}$$

where dt is the filter update period, and $\omega_{x,y}$ are zero mean Gaussian noise terms. We have flown the vehicle at speeds up to 3m/s without issue, but at higher speeds these simplifications to the dynamics model may lead to state prediction errors.

We use discrete integration to update the z axis and velocity states:

$$\mathbf{v}_t = \mathbf{v}_{t-1} + dt\dot{\mathbf{v}}_{t-1} + \omega_{\mathbf{v}}, \qquad \omega_{\mathbf{v}} \sim N(0, \sigma_{\mathbf{v}}) \tag{8}$$

where $\mathbf{v} = [z, \dot{x}^b, \dot{y}^b, \dot{z}^b]$ is a sub-vector of the vehicle state $\mathbf{x}$.

We model the linear accelerations in x and y as proportional to the attitude and velocity, using the simple dynamics model from Tournier et al. (2006):

$$\begin{aligned} \ddot{x}^b &= k_\theta \theta - k_{\dot{x}} \dot{x}^b + \omega_{\ddot{x}}, & \omega_{\ddot{x}} &\sim N(0, \sigma_{\ddot{x}}) \\ \ddot{y}^b &= k_\phi \phi - k_{\dot{y}} \dot{y}^b + \omega_{\ddot{y}}, & \omega_{\ddot{y}} &\sim N(0, \sigma_{\ddot{y}}) \end{aligned} \tag{9}$$

The parameters k_θ and k_ϕ are a function of the mass, inertia, and thrust required for the vehicle to hover. While the damping parameters $k_{\dot{x}}$ and $k_{\dot{y}}$ are small enough that they could be ignored when the vehicle is in the hover regime, they are included here to prevent the model from allowing the estimated velocity to grow without bound in the absence of position corrections. The model parameters are learned by a linear least-squares system identification process from flight and control data collected offline in a motion capture studio.

Finally, the remaining states are modeled using a random walk motion model:

$$\mathbf{a}_t = \mathbf{a}_{t-1} + \omega_{\mathbf{a}}, \qquad \omega_{\mathbf{a}} \sim N(0, \sigma_{\mathbf{a}}) \tag{10}$$

where $\mathbf{a} = [\ddot{z}^b, \phi, \theta, \psi, b^{\ddot{x}}, b^{\ddot{y}}, b^{\ddot{z}}, b^{\phi}, b^{\psi}]$.

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

4.2 Filter Measurement Model

The measurements from each type of sensor arrive asynchronously and at different rates, which means that we must create a separate measurement model for each sensor. We receive IMU measurements from the vehicle's filter (used by the attitude controller) at 100Hz, while the scan-matcher estimates arrive at 40Hz. Before each measurement is integrated, we propagate the filter state forward using the process model described above for dt seconds, where dt is the time since the last measurement arrived.

IMU Measurements We subtract the gravity vector from the IMU measurements using the attitude estimate before integrating $\mathbf{z}^I$. This allows us to model the IMU as if it measures the accelerations and attitude plus the associated bias terms, corrupted by zero mean Gaussian noise,

$$\mathbf{z}_t^I = \begin{bmatrix} \ddot{x}_t^b + b_t^{\ddot{x}} \\ \ddot{y}_t^b + b_t^{\ddot{y}} \\ \ddot{z}_t^b + b_t^{\ddot{y}} \\ \phi_t + b_t^{\phi} \\ \theta_t + b_t^{\theta} \end{bmatrix} + v_{IMU}, \qquad v_{IMU} \sim N(0, \sigma_I). \tag{11}$$

Laser Measurements We condition the scan-matcher estimate on the prior filter state x_{t-k} (as described in Equation 5) to convert the relative motion estimate Δ computed by the scan-matcher in Equation 2 into a measurement of the absolute vehicle position $\mathbf{z}_t^L$. The relative transformation Δ_t is applied to x_{t-k}, obtained from the logged state history. This allows us to model the laser measurement as a measurement of the position and heading of the MAV corrupted by Gaussian noise,

$$\mathbf{z}_t^L = x_{t-k} \oplus \Delta_t + v_S, \qquad v_S \sim N(0, \sigma_L). \tag{12}$$

We use diagonal process and measurement covariance matrices to limit the number of parameters that needed to be tuned. We learn the variance parameters using a method similar to the one described by Abbeel et al. (2005).

4.3 Out of Order Measurements

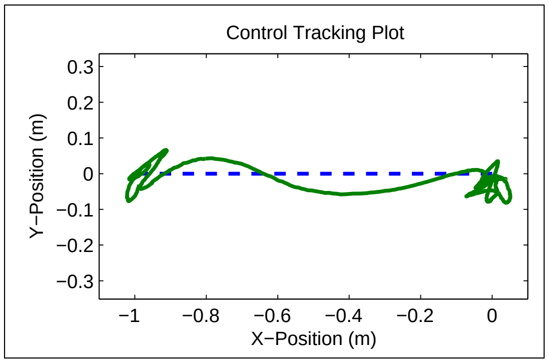
We process IMU measurements as soon as they arrive so that the controller has state estimates computed from all available information. Due to the computation time needed to process incoming laser readings to obtain the relative motion estimates, the scan-matcher measurements generally arrive out of order relative to the IMU measurements. One option for handling out of order measurements is to add measurements to a buffer, and wait until all delayed measurements have arrived before adding them to the filter. This approach imposes a delay on the state estimates that would be problematic for controlling the MAV. Instead we save the history of IMU measurements and associated filter states between scan-matcher measurements such that we can roll back the filter state and replay the measurements in the correct order. Our filter state is small enough that the relatively short computational delays typically experienced by repeated rollbacks are negligible.

5 Position Control

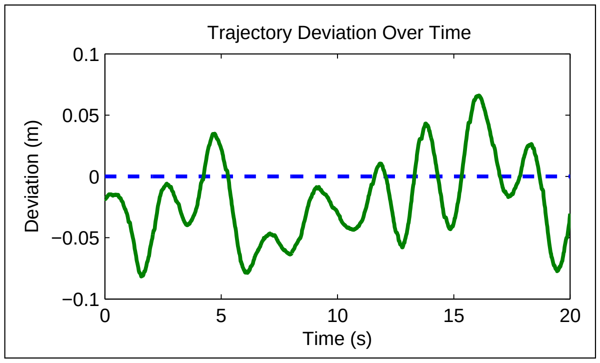
The Ascending Technologies quadrotor helicopters are equipped with a manufacturer-provided attitude stabilization module. This module uses an onboard IMU and processor to control the MAV's pitch and roll (Gurdan et al., 2007). The attitude stabilization allows us to focus on stabilizing the remaining degrees of freedom in position and heading. While the attitude controller simplifies the problem substantially, the remaining degrees of freedom in position are highly dynamic, requiring careful controller design to stabilize the vehicle.

The onboard controller takes 4 inputs:

$$\mathbf{u} = [u_\phi, u_\theta, u_t, u_{\dot{\psi}}] \tag{13}$$



(a)



(b)

Figure 6: Demonstration of the trajectory following performance with commanded position (blue dashed) and actual position (green). The vehicle was commanded to move along the X-axis. (a) The position of the vehicle alongside the desired trajectory and (b) the deviation from the desired trajectory over time. The maximum deviation was 8cm. The vehicle was flying autonomously with the state estimates generated by our system.

where u_ϕ and u_θ denote the desired set points for the onboard PD control loops in pitch (ϕ) and roll (θ) respectively. Unlike these two control inputs, u_ψ sets the desired rotational velocity in yaw (ψ) rather than specifying an absolute attitude. Finally u_t is mapped to the desired baseline rotation rate for all four motors in the motor speed controller, which gives control of the net thrust.

The decoupled control inputs provided by the Ascending Technologies autopilot enable us to control the vehicle's position using 4 independent PID control loops. To find the initial control gains, we perform offline linear least squares system identification on the simple dynamics model from Tournier et al. (2006), and use the Matlab linear quadratic regulator (LQR) toolbox. Subsequent tuning was performed manually with the vehicle flying autonomously.

The final controller was able to hover the vehicle with an RMS error of 6cm, and could accurately follow straight line trajectories with under 8cm deviation from the desired trajectory (RMS error of 7cm). An example of the vehicle following such a trajectory, along with the tracking error is shown in Figure 6.

6 Simultaneous Localization And Mapping

With the MAV state locally estimated by the scan-matcher and data fusion algorithms, we can leverage more computationally intensive SLAM algorithms originally developed for other platforms to refine the position estimates and build maps of the environment. The SLAM process could be integrated directly inside of the data fusion filter described in the previous section, however, this would add a significant computational delay to the estimation process since the filter state would expand to include the positions of environmental features. Instead we keep the SLAM process separate from the realtime control loop, having it provide periodic corrections to the real-time position estimates. This allows the SLAM algorithm to take much more time to integrate information than would be possible if it was part of the local estimation process.

The SLAM module sends position corrections to the data fusion filter to correct any drift in the position estimates. Since these position corrections are delayed significantly from when the measurement upon which they were based was taken, we must account for this delay when we incorporate the correction. Similar to how we compensated for the delay in computing posterior states from the scan-matcher, we retroactively modify the appropriate position estimate in the state history. All future local state estimates are then computed to be relative to the corrected position, resulting in globally consistent real-time state estimates. Under normal operation, the SLAM module sends out updates roughly every 2 seconds with the estimates delayed by ~ 200ms, so only the past 1 second of history must be kept. By incorporating the SLAM corrections after the fact, we allow the real-time state estimates to be processed with low enough delay to control the MAV, while still incorporating the information from SLAM to ensure drift free position estimation.

While there has been a tremendous amount of research on SLAM algorithms, the vast majority of the algorithms have focused on building 2D maps (Leonard and Durrant-Whyte, 1991; Lu and Milios, 1997; Thrun and Montemerlo, 2006; Grisetti et al., 2007). More recently, several groups have begun to achieve success with using either monocular (Davison et al., 2007; Klein and Murray, 2008) or stereo cameras (Mei et al., 2009; Paz et al., 2008) for performing 3D SLAM. Unfortunately, none of the publicly available implementations of 3D SLAM scale to large enough environments, so we instead we make use of more mature 2D SLAM implementations.

We have experimented and successfully flown with two separate SLAM algorithms. Initial experiments were performed with the GMapping algorithm (Grisetti et al., 2007) available in the OpenSlam repository[7], while more recent experiments use the pose graph optimization SLAM implementation described below. The two modules can be interchanged without requiring modifications to the rest of the system, which indicates that it should be easy to replace them with a 3D SLAM solution when one becomes available.

6.1 Pose Graph Optimization SLAM

Pose graph-based techniques are attractive for use on MAVs due to their robustness and computational efficiency. In our implementation, we construct the pose graph using the scan-matcher presented in Section 3.1 for both incremental odometry (after data fusion with IMU) and loop closure detection. We optimize the graph using Kaess' Incremental Smoothing and Mapping (iSAM) library [8] (Kaess et al., 2008) which is fast enough to operate online without difficulty. This approach was used in Kim et al. (2010).

The SLAM graph G is constructed from two types of edges: odometry edges, and loop-closing edges. The odometry edges connect successive poses of the vehicle through time, as estimated by the data fusion filter. For computational efficiency we add a new node to the graph whenever the vehicle has moved a sufficient distance (1m or $15°$). Each node G_i contains the associated laser scan G_i^s and the current best estimate of the pose G_i^p. Each node that gets added to the graph is checked against the rest of the nodes in the graph for a possible loop closure.

We use a relatively simple approach to loop closure that leverages the capabilities of the scan-matcher described in Section 3.1. The algorithm checks the current node against the closest node in the graph (excluding the nodes immediately preceding it), using the quality of the match, and a rigidity check to reject false positives. The loop closure detection algorithm is described in Algorithm 3. Whenever a loop closure edge is added to the pose graph we optimize the trajectory using iSAM, and send the optimized estimate of the current pose to the data fusion filter.

While the loop closure technique is limited by the scan-matcher search region (currently $10\text{m}\times 10\text{m}$), it has proven effective in our testing. The state estimates are accurate enough that even after going around the large loops such as the ones described in Sections 8.2 and 8.3 the estimate is within this search region. Checking a node for loop closure takes roughly 2sec, which is sufficiently fast for online operation since the SLAM algorithm is not part of the real-time control loop.

7 Planning and Exploration

To achieve full autonomy we require a high-level planner that chooses motion and sensing actions to guide the search of an unknown environment. We follow the well-known frontier-based exploration approach (Yamauchi, 1997), in which the thresholds of unknown portions of a map are identified as *frontier regions* deserving further investigation.

While the autonomous exploration problem is not new (Whaite and Ferrie, 1997; Stachniss et al., 2005; Rocha et al.,

[7]OpenSlam. `http://openslam.org`
[8]iSAM library. `http://people.csail.mit.edu/kaess/isam/`

Algorithm 3 Loop Closure Detection

```
Require: SLAM graph $G$          //Node to check for loop closure is $G_t$
 1: get nearest node $G_c \in G_{0:t-30}$
 2: if $||G_t^p - G_c^p|| > 10\text{m}$ then
 3:     return $\emptyset$
 4: end if
 5: Create a likelihood map $M$ from scans $G_{c-5}^s : G_{c+5}^s$ using algorithm 1
 6: $\Delta_t = \text{ScanMatch}(G_t^s, M)$
 7: if Number of points with high likelihood $< .85\%$ then
 8:     return $\emptyset$
 9: end if
10: //Perform a "rigidity" check:
11: $\Delta_{t-1} = \text{ScanMatch}(G_{t-1}^s, M)$
12: if $\|\|G_t^p \ominus G_{t-1}^p\| \ominus \|\Delta_t \ominus \Delta_{t-1}\|\| > .01$ then
13:     return $\emptyset$
14: end if
15: return $G_c$
```

2005), an additional consideration in planning for the quadrotor helicopter is the need to constantly maintain accurate position and velocity estimates to ensure safe flight. The planner must choose trajectories that provide sufficient sensor information to keep the vehicle well-localized. A sampling-based approach such as the Belief Roadmap (Prentice and Roy, 2009) could be used to generate safe trajectories; however, with no prior map and frequent SLAM updates, it becomes infeasible to maintain a belief roadmap in real-time without substantial deliberation costs. We propose using an abridged metric for localizability based on the Sensor Uncertainty Field (SUF) (He et al., 2008) to quickly evaluate goal locations.

7.1 Uncertainty Based Frontier Exploration

We use a modified definition of frontiers to choose possible poses in free space where the vehicle should fly to next, such that it both observes previously unexplored regions and remains well-localized. Yamauchi (1997) grouped free cells that are adjacent to unknown cells in the grid map into frontier regions as possible goals for the robot. In each of these frontier regions, we sample potential poses (x, y, ψ) and calculate their weight according to two metrics.

The first metric captures the amount of unexplored space that the vehicle will observe by simulating laser sensor data from the sampled pose and computing the number of unexplored grid cells in the laser's field-of-view. After dividing by the maximum number of grid cells covered by a laser range scan, we compute a normalized weight $\mathcal{I}_{\mathcal{UR}}(\mathbf{x}) \in [0, 1]$ for the amount of unexplored information that the vehicle is expected to gather from pose $\mathbf{x} = (x, y, \psi)$.

We use a second "Sensor Uncertainty" metric to measure localizability because the first metric favors poses facing into unexplored regions which could provide arbitrarily poor sensor information (e.g. wide open space generating maximum laser ranges). A Sensor Uncertainty Field, first coined by Takeda and Latombe (1992), maps locations $\mathbf{x}$ in the state space to expected information gain, $\mathbf{x} \rightarrow \mathcal{I}_{\mathcal{SU}}(\mathbf{x})$, by calculating the difference in entropy of the prior and posterior distribution $\mathcal{I}_{\mathcal{SU}}(\mathbf{x}) = H(p(\mathbf{x})) - H(p(\mathbf{x}|\mathbf{z}))$ where entropy is $H(p(\mathbf{x})) = -\int p(\mathbf{x}) \log p(\mathbf{x}) \mathbf{dx}$.

He et al. (2008) showed that the measure of information gain for laser data is typically insensitive to the choice of prior. We therefore use a constant prior $p(\mathbf{x}) = \Sigma_0$ such that $H(p(\mathbf{x})) = C$, as well as Bayes' rule to compute $p(\mathbf{x}|\mathbf{z}) = p(\mathbf{z}|\mathbf{x}) \cdot p(\mathbf{x})$, such that $\mathcal{I}_{\mathcal{SU}}(\mathbf{x}) = C - H(p(\mathbf{z}|\mathbf{x}))\Sigma_0$.

The entropy of $p(\mathbf{z}|\mathbf{x})$ is computed by deterministically extracting a set of sigma points (Julier et al., 1995), or samples along the main axes of the current covariance estimate, and observing how they are transformed by the measurement function. We simulate the laser measurement from a frontier pose and compute the probability of obtaining this observation at each sigma point. The lower the probability of the observation at the neighboring sigma points, the smaller the entropy of the posterior distribution, and therefore the greater the information gain. Poses with high

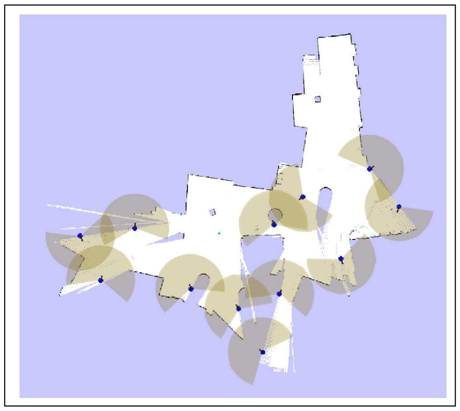


Figure 7: The blue points indicate frontiers that allow the quadrotor helicopter to explore and self-localize simultaneously. The laser's field-of-view at those frontiers is drawn in brown. Notice that at the edges of free space, the chosen frontiers position the vehicle such that the expected laser scan spans both unexplored regions for exploration and unique obstacles for localization.

information gain correspond to localizable regions in the state space. $\mathcal{I}_{\mathcal{SU}}(\mathbf{x}) \in [0, 1]$ is normalized by the prior entropy $H(p(\mathbf{x}))$.

In each frontier region, we accept as a goal point the sample that maximizes the weighted sum of the two information metrics, such that $\mathcal{I}(\mathbf{x}) = \mathcal{I}_{\mathcal{UR}}(\mathbf{x}) + \mathcal{I}_{\mathcal{SU}}(\mathbf{x})$. Figure 7 shows the frontier points generated accordingly, where points are chosen such that the expected laser scan will both uncover unexplored regions and observe known obstacles, enabling the MAV to simultaneously explore and localize itself.

7.2 Exploration Algorithm

The exploration module was designed as a general template that can be customized to address various tasks that require balancing time efficiency, map coverage, localizability, or other mission objectives such as target search (see Section 9.2). Our general exploration algorithm captures these trade-offs in the following cost function:

$$J(f_i) = \sum_{t=0}^{T} C_{SUF} + \alpha_d C_d - R_l, \tag{14}$$

where f_i is a candidate frontier, $C_{SUF} = \{0, \infty\}$ is the cost of uncertainty in localization determined by the SUF threshold, C_d is the shortest-path distance cost of motion, and R_l is the reward for viewing unexplored cells in the map with the laser (proportional to reduction in entropy over maps). The values of C_d, R_l are normalized to $[0, 1]$. α_d is the weight placed on traveling distance; a large value of α_d favors frontier waypoints that are close to the robot.

Note that C_{SUF} enables a quick metric for evaluating localizability; in practice, we have found that in typical indoor and office-like environments, a binary localizability term is sufficient to choose paths that avoid poses (such as frontward down a long corridor) that compromise the robot's ability to localize. Also, a conventional trajectory planner based on dynamic programming is used to determine which frontier goals are reachable, and then to plan and navigate a series of waypoints to the selected goal. Our general exploration algorithm is shown in Algorithm 4.

8 Flight Tests

The system presented in this paper has been tested extensively and has demonstrated stable control of the vehicle in many real world situations. Flight tests were performed in a number of different environments around the MIT campus,

Algorithm 4 Exploration algorithm.

```
1:  initialize, takeoff, enter environment
2:  while mission not complete do
3:      obtain partial map, robot position
4:      generate frontier set $\mathcal{F}$
5:      for all frontiers $f_i \in \mathcal{F}$ do
6:          compute $J(f_i) = \sum_{t=0}^{T} C_{SUF} + \alpha_d C_d - R_l$
7:      end for
8:      $goal = \min_{f_i} J(f_i)$
9:      while not reached $goal$ do
10:         execute trajectory to goal
11:     end while
12: end while
13: land
```

as well as in a competition setting at the 2009 AUVSI International Aerial Robotics Competition. The statistics of these different environments are shown in Table 2. In total, we have logged roughly 100 hours of autonomous flight time. In all situations, the vehicle did not have a prior map, or other information about the environment. The aerial imagery shown in the figures was manually aligned after the test flights for visualization purposes.

To ensure safety during autonomy tests, two people monitor the state of the system. One person manages the ground station, monitoring the real-time data and providing operator input if necessary. The second person acts as a safety pilot who is able to take over control of the vehicle via an RC transmitter in case any problems arise. The vehicle processes are managed from a mobile ground-station which consists of a laptop placed on a mobile cart. During flights, the laptop is powered by batteries which allows it to be moved to keep the vehicle within sight and communications range if necessary.

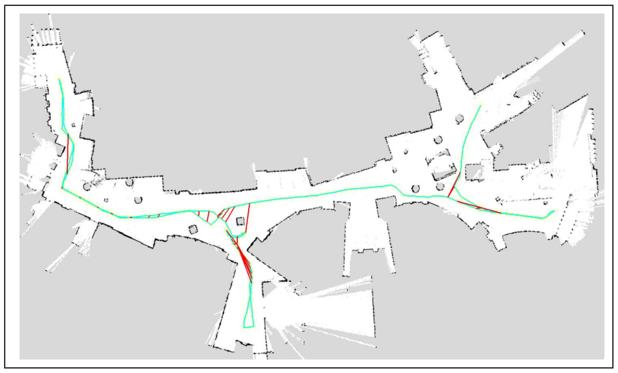
The vehicle is able to take off and land autonomously, so flight tests begin with the vehicle being placed on the ground at the designated start location. Then, once the ground control operator verifies that the state estimation and control processes are running properly, the safety pilot turns on the vehicle, and switches it into autonomous mode. The vehicle can either be instructed to explore the environment, or fly to high level waypoints provided by the user. The experiments presented in this section are conducted in the latter mode, where the operator is manually designating goal waypoints. In Section 9 we present experiments which employed the fully autonomous exploration capabilities of the system. Videos that demonstrate the capabilities of the system are avaiable at `http://groups.csail.mit.edu/rrg/jfr2011-mav`.

Environment	Stata Center	Killian Court	Urban Canyon
Flight Area	130m×90m	240m×130m	270m×160m
Distance Traveled	285 m	745 m	710 m
Flight Time	15 min	15 min	30 min
Nominal Cruise Velocity	0.6 m/s	1.0 m/s	0.5 m/s
Longest Loop	n/a	300 m	400 m
Max Height	1.9 m	1.5 m	3 m

Table 2: Statistics for the environments navigated in the flight test experiments. Note that we swapped out the battery during the urban canyon experiment to extend the flight time.

8.1 Autonomous Navigation in Unstructured Environments

To test the large-scale navigation capabilities of our system in an unstructured indoor environment, we flew the vehicle around 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 MAV. The vehicle was guided by a human operator choosing goals in the map that was being built in real-time, after which the planner planned the best path to the goal. Figure 8 shows


(a)

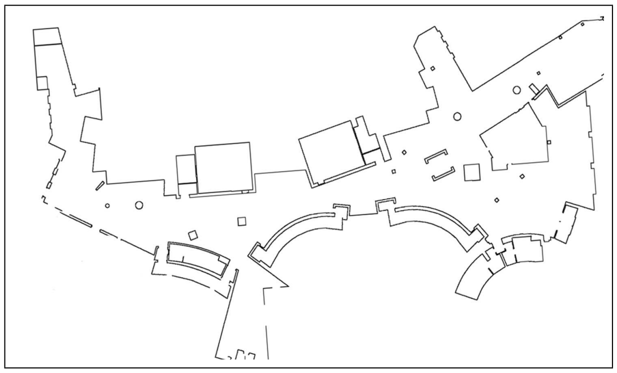
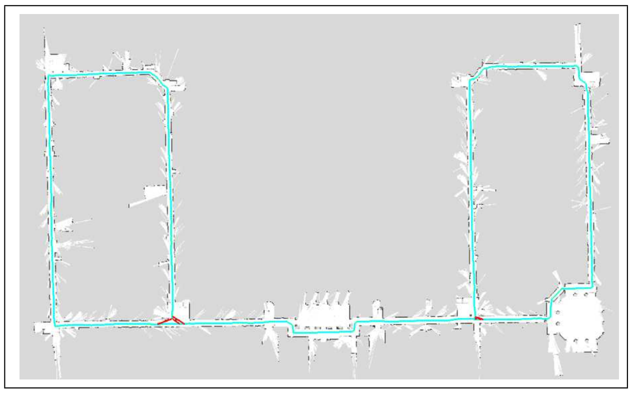

(b)

Figure 8: (a) The occupancy map of the first floor of MIT's Stata center constructed by the vehicle during autonomous flight. The trajectory of the vehicle is shown in cyan, and detected loop closures are drawn in red. (b) The architectural floor plan of the building for comparison purposes.


(a)

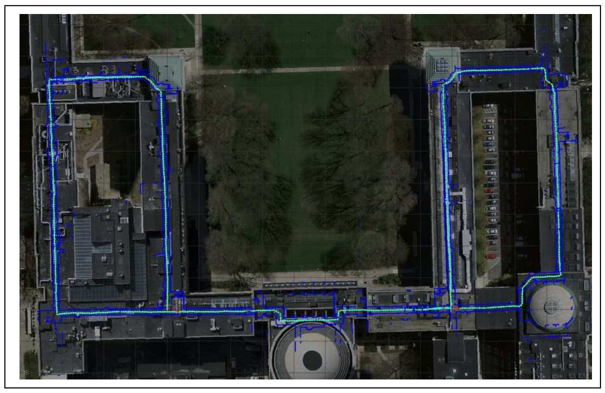

(b)

Figure 9: (a) The occupancy map of MIT's Killian Court area computed by the SLAM algorithm. The trajectory of the vehicle is shown in cyan, and detected loop closures are drawn in red. (b) The laser measurements overlaid upon aerial imagery. While the vehicle was flying inside the building, the detected structure aligns with the structure visible in the aerial imagery, providing a qualitative notion of accuracy.

the final map generated by the SLAM algorithm at the end of the experiment as well as the path taken by the vehicle. As can be seen from the map, the Stata Center has a free-form floor plan which would prevent the use of specific environmental assumptions such as straight hallways, as was done in Celik et al. (2008) and Johnson (2008). Despite these challenges, our system allowed the vehicle to fly until the battery was exhausted. During the 15 minutes of flight, the vehicle flew a distance of 285m.

8.2 Autonomous Navigation in Environments With Large Loops

While the first floor of the Stata Center provided a large-scale environment in which to test our vehicle, it did not provide opportunities for testing the SLAM algorithms in an environment with large loops. To test in such an environment, we flew the vehicle around MIT's Killian court area. The very large loops in this dataset pose a significant challenge to SLAM algorithms, and motivated our use of the pose graph SLAM approach described in Section 6. The SLAM module was able to successfully detect the loop closure occurrences, and optimize the trajectory. The computed map compares favorably with the structure visible in aerial imagery, as shown in Figure 9. During the 15 minute flight, the vehicle traveled a distance of 745m. Interestingly, the recessed door frames in the otherwise featureless hallways provided enough environmental structure for the scan-matcher to allow the system to maintain control of the vehicle throughout the flight.

8.3 Autonomous Navigation in the Urban Canyon

In addition to the indoor experiments, we have conducted a number of flight tests outdoors in GPS-denied urban canyon environments. Using our state estimation system in an outdoor environment presented a number of challenges not generally seen in indoor environments such as wind, large open spaces, and organic structures such as trees. The system proved robust and was able to handle many of the challenges presented by the urban canyon environment. While the vehicle was able to fly the chosen route across campus, other areas caused problems for the state estimation algorithms. For example, areas which contained mostly irregular non-vertical surfaces such as bushes and tree foliage (as opposed to the trunks), broke the assumptions made by the scan-matcher. Similarly, the limited range of the laser scanner is problematic in large open areas where the laser would not observe sufficient structure for the scan-matcher to estimate the position of the vehicle. The operator guiding the exploration during the flight test gave the vehicle waypoints that avoided such problem areas.

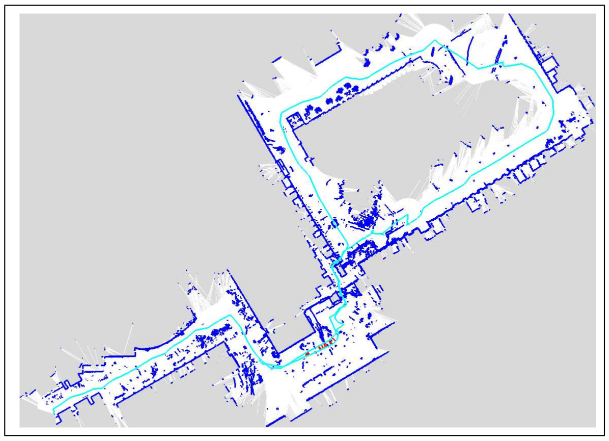
The vehicle carried a GPS receiver during the flight, however, due to the surrounding buildings and generally poor RF landscape on the MIT campus, the receiver never acquired a lock on enough satellites to estimate the position of the vehicle. While we were unable to use the GPS receiver to provide ground truth comparison, we aligned the laser point cloud on top of geo-registered aerial imagery which provides a qualitative measure of the accuracy of our state estimates.

Figure 10 shows the map generated by the vehicle as it was guided on a path across the MIT campus, as well as the laser measurements overlaid on aerial imagery. Perspective renderings of the 3D point cloud generated using the SLAM corrected state estimates are shown in Figure 11. One particularly interesting section of the trajectory is shown in Figure 11(a) when the vehicle passed through a building to cross between two open courtyards. The vehicle flew through a set of open doors, down the hallway, and then out on the opposite side of the building. The ability to transition between indoor and outdoor flight is likely to be very important for many of the potential applications envisioned for MAVs.

Figure 12 shows the path taken by the vehicle during a 45 second long portion of the experiment that we used to evaluate the performance of the position controller in the outdoor setting. The vehicle flew at 1m/s down a road between two buildings that was lined with trees and bushes. The height of the vehicle was commanded to be 2m so that the plane of the laser range-finder was above the parked cars alongside the road. Despite the clutter from the foliage, we can see that the position of the vehicle was accurately estimated by the straightness of the lines visible in the laser point cloud. Our scan matcher is robust enough that it was able to ignore the non-matching scans from the trees, while locking onto the vertical walls of the buildings.

While the state estimates provided by our system were sufficiently accurate to enable robust position control of the vehicle, the velocity estimates were noisier than in normal operation in indoor environments with vertical walls. The noisier state estimates resulted in larger tracking errors than those seen in Section 5. At the beginning of the trajectory (first 20 seconds), the laser scans were aligned quite well, resulting in crisp lines in the point cloud. During this time, the vehicle flew with an average deviation from the trajectory of 0.08m/s. Toward the end of the section, the state estimates were noisier as can seen by the slightly blurry lines in the right half of Figure 12(a). This caused the average error of the position controller to increased to 0.12m/s. In addition to the increased RMS error, the quadrotor helicopter experienced a maximum deviation from the desired trajectory of 0.27m. This maximum error was significantly higher than in the indoor settings and indicates the vehicle must plan more conservative trajectories outdoors. Fortunately, the scale of outdoor environments typically do not require flight in as tightly constrained spaces as indoors.

During the experiment, the vehicle was commanded to fly at various altitudes between 1.5m and 3m. However, since the ground was not level, the height estimation algorithm described in Section 3.2 was unable to estimate the absolute altitude of the vehicle. Instead of the absolute altitude estimate that the algorithm normally computes, it computed a smoothed estimate of the height relative to the local ground surface. While the ground relative height estimates were


(a)

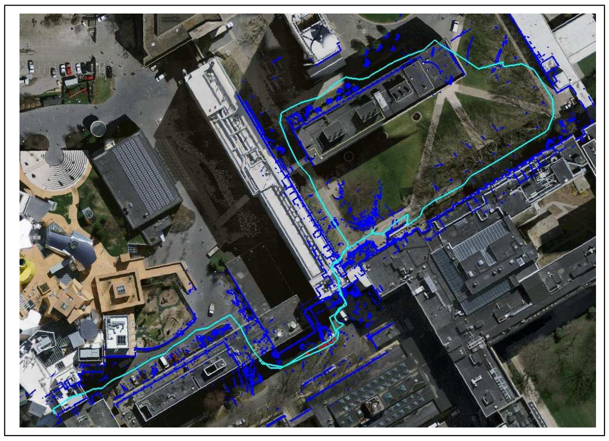
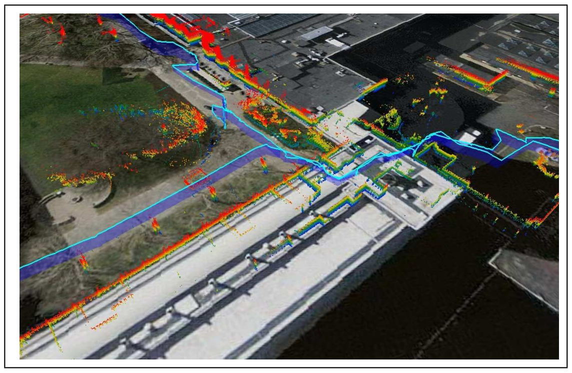
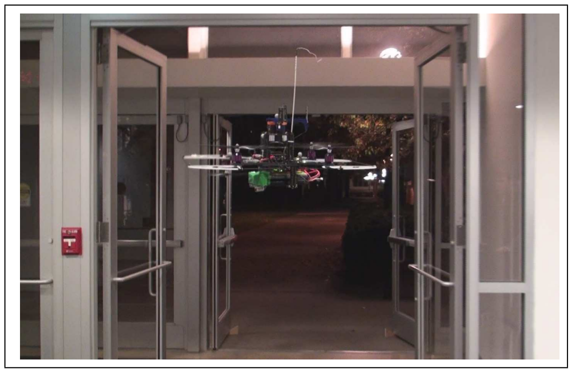
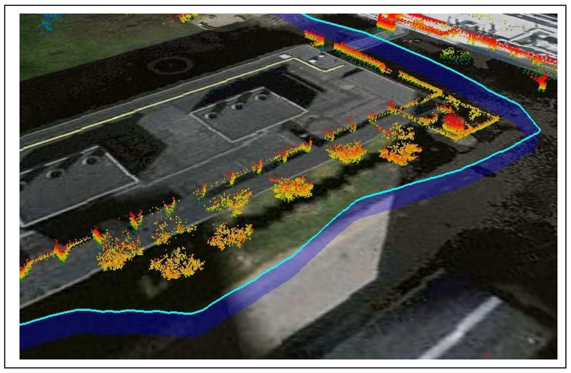

(b)

Figure 10: (a) The occupancy map created while flying across the MIT campus. (b) The laser measurements (blue) overlaid upon aerial imagery of the campus to provide a qualitative measure of the estimation accuracy while flying between the buildings. The trajectory flown by the vehicle is shown in cyan, with loop closures marked in red. Starting from the lower left corner, the vehicle traversed the path drawn in cyan, going around the large loop counter-clockwise.


(a)


(b)


(c)

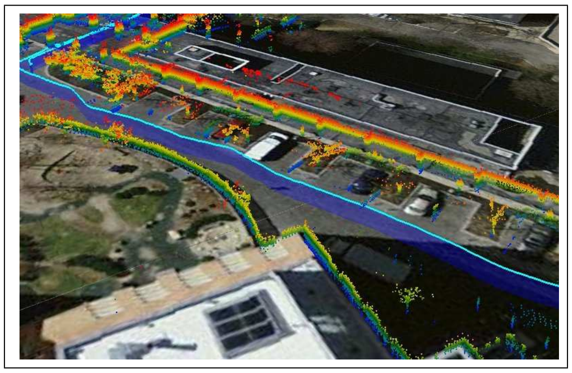
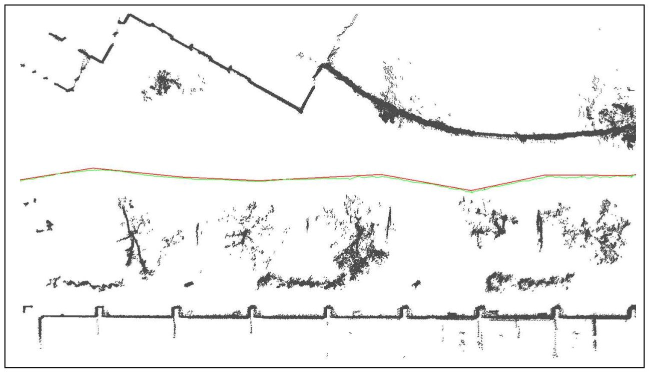
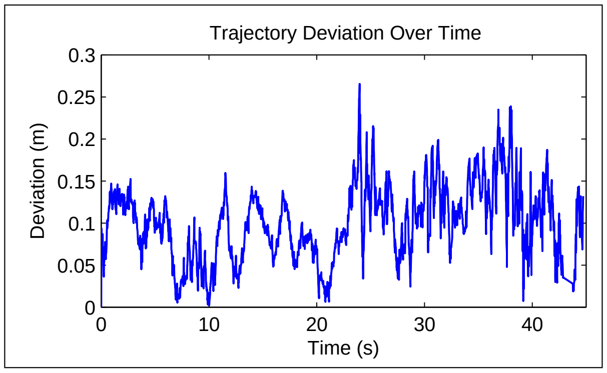

(d)

Figure 11: (a) A perspective view of the area where the vehicle crossed through a building. (b) A picture of the vehicle flying through the doors as it exited the building. (c) An area with trees that demonstrates the 3D structure visible in the laser point cloud. (d) A perspective view of the section used in Figure 12. In all renderings, the path taken by the vehicle is drawn in cyan, with a blue ribbon to indicate height. While the vehicle was predominantly flying at a constant height, the natural rocking of the vehicle causes the laser to sweep over the 3D environment above and below the vehicle. The laser point-cloud is false colored by height. Note that the point cloud is overlaid upon aerial imagery that depicts the roof of the buildings rather than the ground level where the vehicle was flying.


(a)



(b)

Figure 12: (a) The laser point cloud for the portion of the trajectory used to evaluate the performance of the position controller. The laser returns from trees and bushes create clutter, however the correctly aligned building structures are clearly visible. The overlapping red and green lines along the center are the commanded, and actual paths as the vehicle traversed the area from left to right. (b) The deviation from the desired trajectory while traversing this area. The quality of the position estimation degrades in the second half, leading to larger deviations from the trajectory.

not as good for mapping and environment modeling, they were sufficient for control purposes.

During the 30 minutes of flight, the vehicle flew more than 700m at speeds of up to 1.5m/s. Due to the limited flight time provided by a single charge, we stopped to change batteries during the flight.

9 The International Aerial Robotics Competition

The system presented in this paper was used in our winning entry as Team MIT-Ascending Technologies in the 2009 International Aerial Robotics competition (IARC), hosted by the Association of Unmanned Vehicle Systems International (AUVSI). The majority of the experiments described in Section 8 relied on a human operator for high-level guidance. In contrast, a primary constraint of the IARC mission was that no human intervention of any kind was allowed. As a result, we developed an additional set of modules that performed tasks specific to the competition. These modules were added to the top of the system hierarchy, and therefore did not affect the state-estimation or control.

The IARC mission consisted of a disaster recovery scenario in which the MAV must enter a nuclear power plant through an open window and search for a control panel that contained critical information for diagnosing the fault. Our vehicle completed the entire mission; this was the first time in the 19 years of IARC that a team won during the first year of a mission. The competition used artificial walls set up to simulate the interior of a power plant. An overhead view of the $30\text{m}\times15\text{m}$ competition arena is shown in Figure 14(a). The specific mission entailed the following tasks: (1) takeoff roughly 3m from the opening of the arena; (2) identify and fly through a $1\text{m}\times1\text{m}$ window into the arena; (3) explore the unknown environment and search for the control panel; and, (4) autonomously identify the correct gauge (designated by a steady blue LED) and transmit imagery to the judges.

The mission had to be performed completely autonomously in under 10 minutes and prior to each mission run, the arena layout was unknown. Once the operator told the vehicle to start, no human input was allowed until the mission was complete. Each team was given four attempts to complete the mission. For the system to reason about the mission task sequence, we developed a high-level mission planner module that was placed at the top of the system hierarchy (shown in Figure 2). This module commanded the execution of high-level tasks including window entry, exploration, boundary coverage, target inspection and final target designation.

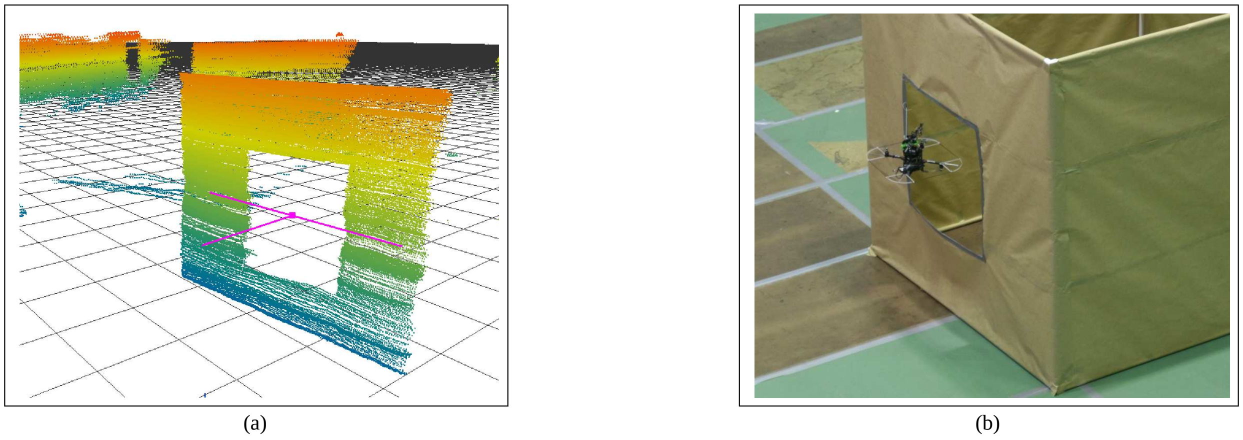


Figure 13: (a) A 3D rendering of the point cloud used for window detection, false colored by height. The laser measurements are projected using the state estimates computed by the filter in Section 4. The detected position of the window is designated by the pink lines. (b) A Photo of the MAV autonomously flying through the $1\text{m}\times 1\text{m}$ window into the arena.

9.1 Window Entry

The mission required flying through a window that was only 25cm wider than the width of the vehicle. Assuming the system could perfectly identify the center of the window, the vehicle could only deviate a maximum of 12cm from the straight-line entry trajectory and errors in the position controller or window identification could result in a catastrophic crash. Fortunately, this task was within the normal error bounds of our position controller described in Section 5.

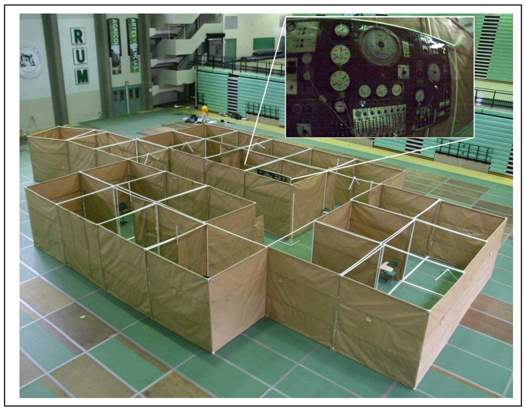
To detect the window, the vehicle performed a “vertical sweep”, accumulating 2D laser scans to generate a 3D point cloud of the window. With the position estimates generated by our state estimator, the system was able to register the set of laser scans into the dense 3D point cloud shown in Figure 13(a). The window was identified by searching the point cloud for a $1\text{m}\times 1\text{m}$ hole in an otherwise connected flat surface (designated by the pink lines in Figure 13(a)). Once detected, the mission planner executed a trajectory through the window center, as shown in in Figure 13(b).

The window detection and entry performed without error more than ten times during the IARC competition, providing a clear demonstration of the accuracy and precision of the state estimation. Furthermore, while we do not have 3D ground truth for the arena, we can visually inspect the 3D walls in the point cloud to see that there is not significant distortion.

9.2 Exploration and Search

From the mission constraints, we identified the need for a planner that could balance time efficiency, geometrical map exploration, visual boundary coverage, and localizability in the environment. The exploration module described in Algorithm 4 was extended to use information gathered by the camera sensor to ensure visual boundary coverage and enable investigation of potential visual targets. To facilitate this, we generate an additional set of candidate goal waypoints, referred to as *camera frontiers*, that correspond to locations in the SLAM map that contain views of the perimeter not yet viewed by the visual target detector.

A target detector processes each camera image to search for the LED-designated gauge, shown in the inset of Figure 14(a). We trained visual classifiers for two different purposes: (1) to safely discard viewed regions where the target is not detected; and, (2) to make a confident final detection at close range. The first target classifier was tuned for low false negative rates at a range of 4m so that we could confidently label observed regions of the environment as not containing the target. In the case that a target is detected, the second classifier, tuned for low false positive rates, is


(a)

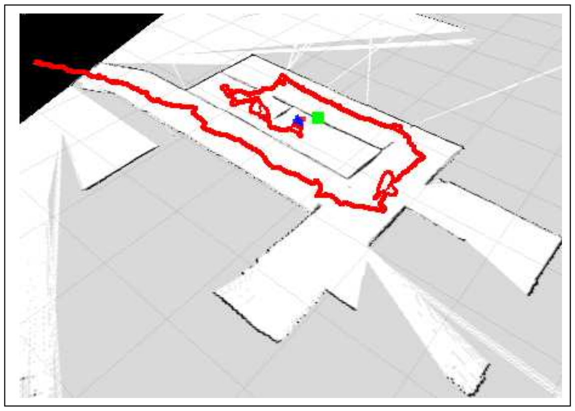

(b)

Figure 14: (a) An overhead view of the competition arena used for the 2009 AUVSI International Aerial Robotics Competition, 5^{th} mission. The inset shows a photo of the IARC control panel. This is one of the frames from the video stream sent to the judges, which successfully completed the mission. (b) The map built by the vehicle as it explored the IARC arena searching for the control panel. Black areas are obstacles (or no-fly zones), white is free space, and gray is unexplored. Starting from the left side, the red line shows the path taken by the vehicle. The blue and red cross is the final position of the vehicle. The green square in front of the vehicle is the location of the autonomously designated control panel.

used to provide a confident final classification once the vehicle has maneuvered to view the candidate gauge at close range (< 1m).

We augment the grid map by marking boundary regions that have been classified by visual target detector as negative. The end points of camera-viewed boundary regions are obtained by scanning the coverage map for grid cells neighbored by viewed and unviewed regions. Camera frontiers are constructed for each of these boundary points by ray-tracing obstacle-free viewpoints of the region.

Given a set of laser and camera frontiers $\mathcal{F}$, the exploration objective function in Equation 14 can be modified to capture mission trade-offs as follows:

$$J(f_i) = \sum_{t=0}^{T} C_{SUF} + \alpha_d C_d - \alpha_e R_l - (1 - \alpha_e) R_c, \tag{15}$$

where $f_i \in \mathcal{F}$ is a candidate frontier, $C_{SUF} = \{0, \infty\}$ is the cost of uncertainty in localization determined by the SUF threshold, C_d is the shortest-path distance cost, R_l is the reward for viewing unexplored cells in the map with the laser, R_c is the reward for viewing camera boundary cells at a given distance, α_d is the weight placed on travel distance, and α_e captures the trade-off between laser exploration and camera coverage. α_e can be varied to generate behaviors biased towards map building ($\alpha_e = 1$) or boundary coverage ($\alpha_e = 0$).

When a potential target is detected, the mission planner postpones exploration to investigate the target at close range. The vehicle plans a trajectory to a close range view of the target, visually servos to center the target in the camera field-of-view, and then makes a reliable final classification.

9.3 IARC Flight Performance

Each team was given four attempts to complete the IARC mission. In the first three attempts during the IARC, the vehicle explored a large portion of the environment, but was unable to reach the room with the control panel. While exploring, there were several doors that were exactly 1m wide; such narrow passages other than the entrance window were unanticipated and the "safety regions" set for obstacle avoidance and path planning labeled these doorways as impassable. After adjusting parameters, the vehicle found the control room on its fourth attempt and transmitted the picture of the target control panel shown in the inset of Figure 14(a) to the judges, which completed the mission. The path followed by the vehicle during the successful attempt is shown on top of the constructed map in Figure 14(b). The mission was completed quickly in 4.5 minutes without having to explore the entire environment.

10 Discussion

In this work we developed a system that enables completely autonomous exploration and mapping in GPS-denied environments. To enable this capability we developed and integrated a set of modules that address the key challenges laid out in Section 1.1. In designing the system, we found the following insights to be critical to our success:

- **Decoupled System Architecture:** Rather than using a single integrated state estimator, we realized that it was critical to develop a local estimator for controlling the position of the vehicle, with special emphasis on estimating and controlling the velocity. The multi-level system hierarchy presented in Section 2 allows for this decomposition and was a critical development that allowed our system to work. Additionally, once the position of the vehicle is accurately controlled, the task of designing and implementing higher level algorithms for a MAV is greatly simplified.

- **Fast and Robust Laser Scan-Matching:** The fast dynamics of MAVs reduce the amount of computation time allowed for the real-time state estimation algorithms. In addition, the assumptions required for using a 2D sensor in a 3D world are violated much more frequently than on a ground vehicle. Our work placed special emphasis on the development of a very fast scan matching algorithm that maintained the accuracy and robustness required for use on a MAV. The laser scan-matcher in Section 3.1 was the key enabling technology that allows our vehicle to fly in GPS-denied environments.

While the presented system has demonstrated very good performance in a number of real world settings, further work will be required to develop the ability to operate in completely unconstrained environments. We have identified two classes of environments in which our system was unable to operate:

- **Complex 3D Environments:** Some areas of the Stata Center at MIT have walls that are slanted. If the slanted walls subtend a large enough portion of the field of view of the laser scanner, the scan-matcher will be unable to disregard as outliers the false matches between scans taken at different heights. Similar failures occur in outdoor environments when the laser scanner predominately observes the leaves and branches of trees (as opposed to vertical tree trunks).

- **Featureless Environments:** Featureless hallways with no doors, windows, shelves, etc. provide no structure for the scan-matcher to distinguish. If the hallway is longer than the maximum range of the sensor (30m), such that the end of the hallway is not observed, then matching consecutive laser scans is ill-posed along the axis of the hallway and the scan-matcher is unable to estimate vehicle motion. Similar situations occur where the surrounding environmental structure is beyond the maximum range of the sensor, such as large open areas.

The failure modes described above are fundamental limitations of using a 2D laser scanner for estimating the motion of the vehicle. Developing a system that is able to operate in such environments will require the integration of other sensing modalities, such as camera sensors. While this would likely greatly reduce the situations in which the system would fail, it would not completely eliminate the failure modes. Fully handling these types of challenging scenarios will likely require relaxing the need to maintain a purely metric representations of the environment and the state of the vehicle (Kuipers et al., 2000; Sibley et al., 2010).

11 Conclusions & Future Work

This paper presented our solution for enabling robust autonomous navigation in GPS-denied environments. Our system leverages a multi-level sensing and control hierarchy which is able to successfully close the loop around the fast and unstable MAV dynamics using only sensors onboard the vehicle. We have developed a very accurate and robust high-speed laser scan-matching algorithm that allows us to compute the relative position estimates with enough accuracy and low enough delay for control purposes. By combining these measurements with the data from the onboard IMU in the data fusion filter, we are able to obtain accurate estimates of the vehicle states. These state estimates are accurate enough to enable the vehicle to hover and fly in constrained indoor and urban environments.

The system has been thoroughly tested in a number of large-scale environments, both indoors and outdoors in the urban canyon. In each of these environments the system was able to navigate autonomously, guided either by a human operator clicking high level goals in the map being built online, or by an exploration algorithm.

The autonomous capabilities of the system were employed by our team's winning entry into the 2009 AUVSI International Robotics Competition. The system presented a very robust and stable platform on top of which we were able to build the necessary additional components that carried out the mission. The state estimation and control architecture made it such that when we developed these components, we did not have to be concerned with stabilizing the vehicle or any of the other real-time concerns that traditionally make developing algorithms for MAVs particularly challenging. The mission was successfully completed with no human intervention once the vehicle was told to start.

In the future, we hope to expand the capabilities of the system by developing algorithms for perceiving, planning, and operating in arbitrary 3D environments. The 2D SLAM and exploration algorithms that we have developed provide a proof of concept for achieving high level autonomy on a MAV, however more work will be required to unlock the full potential of MAVs to operate in 3D. The incorporation of visual information from camera sensors will be an area of particular interest.

In addition to augmenting the 3D perception capabilities of the quadrotor helicopter, we hope to adapt the system for use on other UAVs. Moving to another hovering MAV should be fairly simple, only requiring the adaptation of the position controller for the new platform. The state estimation algorithms such as the scan-matcher are agnostic to the vehicle platform as long as the 2D laser sensor is kept near level. On the other hand, developing autonomous capabilities for faster moving vehicles such as a (quadrotor) helicopter flying aggressively, or a fixed-wing vehicle will involve significantly more effort. More work will be required to enable high quality state estimation in the face of increased motion blur and reduced reaction time. While these tasks will certainly be quite challenging, we believe that the system presented in this paper will provide a solid foundation for these future research directions.

12 Acknowledgements

This work was supported by the ARO MAST CTA and ONR under MURI N00014-07-1-0749 and Science of Autonomy program N00014-09-1-0641. We are grateful for their support. The authors wish to thank the following people for their support in the project: Daniel Gurdan, Jan Stumpf, Markus and Michael Achtelik provided the quadrotor helicopter and the support of Ascending Technologies; Michael Kaess provided the iSAM optimization engine; Edwin Olson provided a reference implementation of his scan-matcher; Garrett Hemann assisted with the development and maintenance of the hardware; and, Adam Bry was a safety pilot during testing.

References

Abbeel, P., Coates, A., Montemerlo, M., Ng, A., and Thrun, S. (2005). Discriminative training of Kalman filters. In *Proc. of Robotics: Science and Systems.*

Ahrens, S., Levine, D., Andrews, G., and How, J. (2009). Vision-based guidance and control of a hovering vehicle in unknown, GPS-denied environments. In *IEEE Int. Conf. Robotics and Automation*, pages 2643–2648.

Altug, E., Ostrowski, J. P., and Mahony, R. (2002). Control of a quadrotor helicopter using visual feedback. In *IEEE Int. Conf. Robotics and Automation*, volume 1, pages 72–77.

Angeletti, G., Valente, J. P., Iocchi, L., and Nardi, D. (2008). Autonomous indoor hovering with a quadrotor. In *Workshop Proc. of SIMPAR*, pages 472–481, Venice, Italy.

Bachrach, A., He, R., and Roy, N. (2009a). Autonomous flight in unknown indoor environments. *Int. Journal of Micro Air Vehicles*, 1(4):217–228.

Bachrach, A., He, R., and Roy, N. (2009b). Autonomous flight in unstructured and unknown indoor environments. In *Proc. European Micro Air Vehicle Conference*.

Blosch, M., Weiss, S., Scaramuzza, D., and Siegwart, R. (2010). Vision based mav navigation in unknown and unstructured environments. In *IEEE Int. Conf. Robotics and Automation*, pages 21–28. IEEE.

Bouabdallah, S., Murrieri, P., and Siegwart, R. (2005). Towards autonomous indoor micro VTOL. *Autonomous Robots*, 18(2):171–183.

Bresenham, J. (1965). Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30.

Buskey, G., Roberts, J., Corke, P., and Wyeth, G. (2004). Helicopter automation a using a low-cost sensing system. *Computing Control Engineering Journal*, 15(2):8 – 9.

Casbeer, D., Beard, R., McLain, T., Li, S., and Mehra, R. (2005). Forest fire monitoring with multiple small UAVs. In *Proc. of the American Control Conference, ACC*, pages 3530–3535 vol. 5.

Celik, K., Chung, S. J., and Somani, A. (2008). Mono-vision corner SLAM for indoor navigation. In *IEEE Int. Conf. on Electro/Information Technology*, pages 343–348.

Censi, A. (2008). An ICP variant using a point-to-line metric. In *IEEE Int. Conf. Robotics and Automation*, pages 19–25, Pasadena, CA.

Coates, A., Abbeel, P., and Ng, A. Y. (2008). Learning for control from multiple demonstrations. In *Proc. of the Int. Conf. on Machine Learning, ICML*, pages 144–151, New York, NY, USA. ACM.

Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). MonoSLAM: Real-time single camera SLAM. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(6):1052–1067.

Furukawa, T., Bourgault, F., Lavis, B., and Durrant-Whyte, H. (2006). Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets. In *IEEE Int. Conf. Robotics and Automation*, pages 2521–2526.

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

Grzonka, S., Grisetti, G., and Burgard, W. (2009). Towards a navigation system for autonomous indoor flying. In *IEEE Int. Conf. Robotics and Automation*, pages 2878–2883.

Gurdan, D., Stumpf, J., Achtelik, M., Doth, K.-M., Hirzinger, G., and Rus, D. (2007). Energy-efficient autonomous four-rotor flying robot controlled at 1 kHz. *IEEE Int. Conf. Robotics and Automation*, pages 361–366.

Haehnel, D. (2004). *Mapping with Mobile Robots*. PhD thesis, University of Freiburg.

He, R., Bachrach, A., Achtelik, M., Geramifard, A., Gurdan, D., Prentice, S., Stumpf, J., and Roy, N. (2010). On the design and use of a micro air vehicle to track and avoid adversaries. *The Int. Journal of Robotics Research*, 29(5):529–546.

He, R., Prentice, S., and Roy, N. (2008). Planning in information space for a quadrotor helicopter in a GPS-denied environments. In *IEEE Int. Conf. Robotics and Automation*, pages 1814–1820, Los Angeles, CA.

Hoffmann, G., Huang, H., Waslander, S., and Tomlin, C. (2007). Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proc. of the AIAA Guidance Navigation and Control Conference, GNC*, pages 1–20.

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

Hrabar, S. and Sukhatme, G. (2009). Vision-based navigation through urban canyons. *Journal of Field Robotics*, 26(5):431–452.

Huang, A. S., Olson, E., and Moore, D. (2009). Lightweight communications and marshalling for low-latency interprocess communication. Technical Report 2009-041, Massachusetts Institute of Technology.

Johnson, N. (2008). Vision-assisted control of a hovering air vehicle in an indoor setting. Master's thesis, Brigham Young University.

Julier, S., Uhlmann, J., and Durrant-Whyte, H. (1995). A new approach for filtering nonlinear systems. In *Proc. of the American Control Conference, ACC*, volume 3, pages 1628–1632.

Jun, M., Roumeliotis, S. I., and Sukhatme, G. S. (1999). State estimation of an autonomous helicopter using Kalman filtering. In *IEEE Int. Conf. on Intelligent Robots and Systems*, volume 3, pages 1346–1353.

Kaess, M., Ranganathan, A., and Dellaert, F. (2008). iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378.

Kemp, C. (2006). *Visual Control of a Miniature Quad-Rotor Helicopter*. PhD thesis, University of Cambridge.

Kim, B., Kaess, M., Fletcher, L., Leonard, J., Bachrach, A., Roy, N., and Teller, S. (2010). Multiple relative pose graphs for robust cooperative mapping. In *IEEE Int. Conf. Robotics and Automation*, pages 3185–3192.

Klein, G. and Murray, D. (2008). Improving the agility of keyframe-based SLAM. In *Proc. of the European Conf. on Computer Vision, ECCV*, pages 802–815, Berlin, Heidelberg. Springer-Verlag.

Kuipers, B., Browning, R., Gribble, B., Hewett, M., and Remolina, E. (2000). The spatial semantic hierarchy. *Artificial Intelligence*, 119:191–233.

Langelaan, J. and Rock, S. (2005). Towards Autonomous UAV Flight in Forests. *Proc. of the AIAA Guidance Navigation and Control Conference, GNC*, pages 1–13.

Leonard, J. and Durrant-Whyte, H. (1991). Simultaneous map building and localization for an autonomous mobile robot. In *IEEE Int. Conf. on Intelligent Robots and Systems*, volume 3, pages 1442 –1447.

Lu, F. and Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349.

M. Achtelik, T. Zhang, K. K. and Buss, M. (2009). Visual tracking and control of a quadcopter using a stereo camera system and inertial sensors. In *Proc. of the IEEE Int. Conf. on Mechatronics and Automation, ICMA*, pages 2863–2869.

Matsue, A., Hirosue, W., Tokutake, H., Sunada, S., and Ohkura, A. (2005). Navigation of small and lightweight helicopter. *Japan Society of Aeronautical Space Sciences Transactions*, 48:177–179.

Matsuoka, M., Chen, A., Singh, S. P. N., Coates, A., Ng, A. Y., and Thrun, S. (2007). Autonomous helicopter tracking and localization using a self-surveying camera array. *Int. Journal of Robotics Research*, 26(2):205–215.

Mei, C., Sibley, G., Cummins, M., Newman, P., and Reid, I. (2009). A constant time efficient stereo slam system. In *Proc. of the British Machine Vision Conference (BMVC)*, London.

Mejias, L. O., Saripalli, S., Cervera, P., and Sukhatme, G. S. (2006). Visual servoing for tracking features in urban areas using an autonomous helicopter. In *IEEE Int. Conf. Robotics and Automation*, pages 2503–2508.

Olson, E. (2008). *Robust and Efficient Robotic Mapping*. PhD thesis, Massachusetts Institute of Technology.

Olson, E. (2009). Real-time correlative scan matching. In *IEEE Int. Conf. Robotics and Automation*, pages 4387–4393.

Paz, L. M., Pinies, P., Tardos, J. D., and Neira, J. (2008). Large-scale 6-DOF slam with stereo-in-hand. *IEEE Transactions on Robotics*, 24(5):946–957.

Prentice, S. and Roy, N. (2009). The belief roadmap: Efficient planning in belief space by factoring the covariance. *The Int. Journal of Robotics Research*, 28(11-12):1448–1465.

Ranganathan, A., Kaess, M., and Dellaert, F. (2007). Fast 3D pose estimation with out-of-sequence measurements. In *IEEE Int. Conf. on Intelligent Robots and Systems*, pages 2486–2493, San Diego, CA.

Roberts, J., Stirling, T., Zufferey, J., and Floreano, D. (2007). Quadrotor using minimal sensing for autonomous indoor flight. In *Proc. European Micro Air Vehicle Conference*.

Rocha, R., Dias, J., and Carvalho, A. (2005). Entropy Gradient-based Exploration with Cooperative Robots in 3-D Mapping Missions. In *Proc. of ICRA Workshop on Cooperative Robotics*, volume 22.

Saripalli, S., Roberts, J. M., Corke, P. I., Buskey, G., and Sukhatme, G. S. (2003). A tale of two helicopters. In *IEEE Int. Conf. on Intelligent Robots and Systems*, volume 1, pages 805–810 vol.1.

Saripalli, S. and Sukhatme, G. S. (2007). Landing a helicopter on a moving target. In *IEEE Int. Conf. Robotics and Automation*, pages 2030–2035.

Scherer, S., Singh, S., Chamberlain, L., and Elgersma, M. (2008). Flying fast and low among obstacles: Methodology and experiments. *Int. Journal of Robotics Research*, 27(5):549–574.

Sibley, G., Mei, C., Reid, I., and Newman, P. (2010). Planes, trains and automobiles – autonomy for the modern robot. In *IEEE Int. Conf. Robotics and Automation*, pages 285 –292.

Stachniss, C., Grisetti, G., and Burgard, W. (2005). Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of Robotics: Science and Systems (RSS)*, pages 65–72.

Steder, B., Grisetti, G., Grzonka, S., Stachniss, C., Rottmann, A., and Burgard, W. (2007). Learning maps in 3D using attitude and noisy vision sensors. In *IEEE Int. Conf. on Intelligent Robots and Systems*, pages 644–649.

Takeda, H. and Latombe, J. (1992). Sensory uncertainty field for mobile robot navigation. *IEEE Int. Conf. Robotics and Automation*, pages 1002–1017.

Templeton, T., Shim, D., Geyer, C., and Sastry, S. (2007). Autonomous vision-based landing and terrain mapping using an MPC-controlled unmanned rotorcraft. In *IEEE Int. Conf. Robotics and Automation*, pages 1349–1356.

Thrun, S. and Montemerlo, M. (2006). The graph SLAM algorithm with applications to large-scale mapping of urban structures. *Int. Journal of Robotics Research*, 25(5-6):403–429.

Tisdale, J., Ryan, A., Kim, Z., Tornqvist, D., and Hedrick, J. (2008). A multiple UAV system for vision-based search and localization. In *Proc. of the American Control Conference, ACC*, pages 1985–1990.

Tournier, G., Valenti, M., How, J., and Feron, E. (2006). Estimation and control of a quadrotor vehicle using monocular vision and Moirè patterns. In *Proc. of the AIAA Guidance Navigation and Control Conference, GNC*, pages 21–24.

Whaite, P. and Ferrie, F. (1997). Autonomous exploration: Driven by uncertainty. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 19(3):193–205.

Yamauchi, B. (1997). A frontier-based approach for autonomous exploration. In *Proc. of the IEEE Int. Symposium on Computational Intelligence, Robotics and Automation, CIRA*, pages 146–151.

Zhang, Z. (1994). Iterative point matching for registration of free-form curves and surfaces. *Int. Journal of Computer Vision*, 13(2):119–152.

Zimmerman, M. and Sulzer, W. (1991). High bandwidth orientation measurement and control based on complementary filtering. In *Proc. of the Symposium on Robotics and Control*, pages 525–530, Vienna.