Autonomous Navigation and Exploration of a Quadrotor Helicopter in GPS-denied Indoor Environments

Markus Achtelik[a], Abraham Bachrach[b], Ruijie He[b], Samuel Prentice[b] and Nicholas Roy[b]

[a] Technische Universität München, Germany
[b] Massachusetts Institute of Technology, Cambridge, MA, USA

ABSTRACT

This paper presents our solution for enabling a quadrotor helicopter to autonomously navigate, explore and locate objects of interest in unstructured and unknown indoor environments. We describe the design and operation of our quadrotor helicopter, before presenting the software architecture and individual algorithms necessary for executing the mission. Experimental results are presented demonstrating the quadrotor's ability to operate autonomously in indoor environments. Finally, we address some of the logistical and risk management issues pertaining to our entry in the competition.

1. INTRODUCTION

We describe our quadrotor helicopter solution for navigating, exploring and locating objects of interest autonomously in unstructured and unknown indoor environments. Our goal is to develop a complete autonomous air vehicle system that can take off, fly through a window, explore and map an unknown indoor environment, search for an object of interest, and transmit the video footage of the object location back to the ground station.

There are two technical challenges to this problem. The first challenge is to design the quadrotor helicopter and onboard-electronics that is able to autonomously stabilize its attitude. The second challenge was primarily a software challenge of developing the requisite algorithms that rely only on on-board exteroceptive sensors, such as a laser rangefinder and stereo cameras, in order to develop a complete system that is capable of autonomous navigation and exploration in GPS-denied environments.

1.1 Conceptual solution

Our system employs a three level sensing hierarchy, as shown in Figure 1. At the base level, the on-board IMU and processor create a very tight feedback loop to stabilize the helicopter's pitch and roll, operating at $1000Hz$. At the next level, the realtime odometry algorithm (laser or visual) estimates the vehicle's position relative to the local environment, while an Extended Kalman Filter (EKF) combines these estimates with the IMU outputs to provide accurate state estimates of the position and velocity at $50Hz$. These estimates enable the LQR controller to hover the helicopter stably in small, local environments. In addition, a simple obstacle avoidance routine ensures that the helicopter maintains a minimum distance from obstacles.

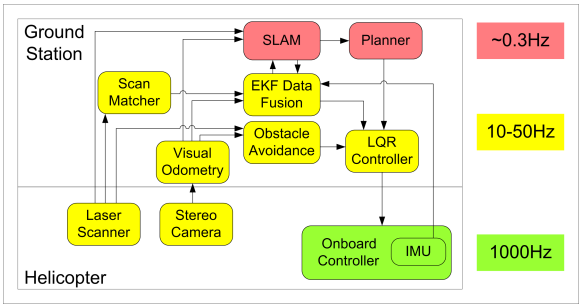


Figure 1. Schematic of our hierarchical sensing, control and planning system. At the base level, the on-board IMU and controller (green) creates a tight feedback loop to stabilize the helicopter's pitch and roll. The yellow modules make up the real-time sensing and control loop that stabilizes the helicopter's pose at the local level and avoids obstacles. Finally, the red modules provide the high-level mapping and planning functionalities.

Small errors made by the odometry algorithms will be propagated forward, resulting in inaccurate state estimates in large environments. To mitigate these errors, we introduce a third sensing level, where a Simultaneous Localization and Mapping (SLAM) algorithm uses both the EKF state estimates and incoming sensor measurements to create a global map, ensuring globally consistent estimates of the vehicle position and map to detect when the robot returns to a previously-visited point, and to correctly compute the resulting map from this closed loop in the trajectory. Since SLAM algorithms today are incapable of processing sensor data fast enough, this module is not part of the real-time feedback control loop at the second level. Instead, it provides delayed correction signals to the EKF, ensuring that our real-time state estimates remain globally consistent.

To achieve autonomous operation, a planner enables the vehicle to both navigate to desired locations and explore unknown regions of the world. A navigation module plans waypoint trajectories through the known grid map maintained by the SLAM process to reach desired goal locations. Also, to cope with unknown regions of the environment, our system relies on a frontier-based explorer to identify "frontier" regions of unexplored space that serve as future goal locations.

Finally, to autonomously complete the proposed challenge, we require a high-level mission planner to coordinate the execution of a series of mission tasks. The problem can be decomposed into several core tasks, including: autonomous takeoff; window entrance; exploration and map-building; target search; target data transmission; and safe, autonomous landing. The mission planner operates as a decision-making layer above of our perception, control, and planning system, arbitrating between these modules and determining when to transition between mission tasks.

1.2 Key Challenges for Autonomous MAVs

Combining wheel odometry with exteroceptive sensors in a probabilistic SLAM framework has proven very successful for ground robotics;[15] many algorithms for accurate localization of ground robots in large-scale environments exist. Unfortunately, performing the same tasks on a MAV requires more than mounting equivalent sensors onto helicopters and using the existing SLAM algorithms. Flying robots behave very different from ground robots, and therefore, the requirements and assumptions that can be made for flying robots must be explicitly reasoned about and managed. We have reported this analysis previously but we summarize it here.

Limited sensing payload: MAVs must generate sufficient vertical thrust to remain airborne, limiting the available payload. This weight limitation forces indoor air robots to rely on lightweight Hokuyo laser scanners, micro cameras and lower-quality MEMS-based IMUs, all of which are limited compared to their ground equivalents.

Indirect odometry: Unlike ground vehicles, air vehicles are unable to measure odometry directly, which most SLAM algorithms require to initialize their estimate of the vehicle's motion between time steps. Although odometry can be obtained by double-integrating accelerations, lightweight MEMs IMUs are often subject to time-varying biases that result in high drift rates.

Limited computation on-board: Despite advances within the research community, SLAM algorithms continue to be computationally demanding even for powerful desktop computers, and are therefore not implementable on today's small embedded computer systems that can be mounted on-board indoor MAVs. While the computation can be offloaded to a powerful ground-station by transmitting sensor data wirelessly, communication bandwidth then becomes a potential bottleneck, especially for camera data.

Fast dynamics: The helicopter's fast dynamics also result in a host of sensing, estimation, control and planning implications for the vehicle. Filtering techniques, such as the family of Kalman Filters, are often used to obtain better estimates of the true vehicle state from noisy measurements. Smoothing the data generates a cleaner signal but adds delay to the state estimates. While delays generally have insignificant effects on vehicles with slow dynamics, these effects are amplified by the MAV's fast dynamics, and cannot be ignored.

Need to estimate velocity: In addition, the helicopter's under-damped dynamics imply that proportional control techniques are insufficient to stabilize the vehicle; we must therefore estimate the vehicle's velocities, whereas most SLAM algorithms completely ignore the velocity. While the helicopter can accurately hover using PD-control, it oscillates unstably with only a P-control. This emphasizes the importance of obtaining accurate and timely state estimates of both position and velocity states.

Constant motion: Unlike ground vehicles, a MAV cannot simply stop and re-evaluate when its state estimates have large uncertainties. Instead, the vehicle is likely to oscillate, degrading the sensor measurements further. Therefore, planning algorithms for air vehicles must not only be biased towards paths with smooth motions, but must also explicitly reason about uncertainty in path planning, as demonstrated by He et al.[9]

1.3 Yearly milestones

In March 2008, the team participated in the 1st US-Asian Demonstration and Assessment of Micro-Aerial and Unmanned Ground Vehicle Technology in Agra, India. Competing in a hostage-rescue mission scenario, the team won the "Best Mission Performance Award", the "Best Rotary Wing Aircraft Award", and an AMRDEC-award.

In 2009, the team plans to accomplish the 5th mission of the International Aerial Robotics Competition.

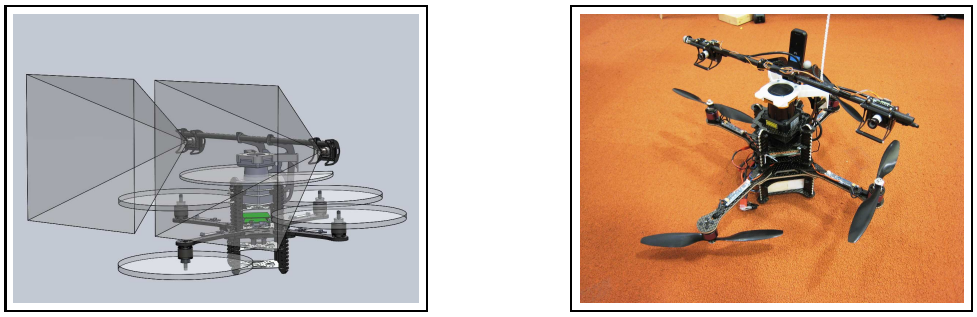
2. AIR VEHICLE

2.1 Hardware Design

The vehicle we use is a quadrotor helicopter with four smaller rotors instead of a main and tail rotor. Two pairs of rotors spin clockwise and counterclockwise respectively, such that the sum of their reaction torques is zero during hovering. Unlike normal helicopters, the rotors of the quadrotor applied in this work have fixed pitch angles. This minimalistic hardware design, (no rotor linkages etc) makes the quadrotor robust so that it can survive the inevitable crashes during experiments without getting seriously damaged.

In cooperation with Ascending Technologies, we designed a quadrotor helicopter capable of carrying a requisite sensor payload (described in Section 3) for the 10-minute mission time. The weight of the components was determined to be $\sim 400g$ plus a safety margin of $\sim 100g$ for cables and mounting material.

The top of the quadrotor was designed as an interlocking rack to be able to flexibly mount components for experiments. Sensor positions were aligned so that the fields of view do not intersect with parts of the quadrotor (shown in Figure 2), and the front rotor was placed below the arm to maintain a low center of gravity, while avoiding camera obstruction. The electronics and algorithms are shared with the successful Ascending Technologies Hummingbird quadrotor .[7] Compared to the Hummingbird, the new vehicle uses larger (10in) rotors as well as more powerful brushless motors to increase the payload.



(a) Original Image

Figure 2. (a): CAD drawing of the quadrotor design with simulated fields of view. (b): final assembled quadrotor with stereo camera system, laser range-finder, WiFi dongle and on-board computer

2.2 Attitude Stabilization

The six DOF of the quadrotor are controlled by four inputs (roll, pitch, yaw, thrust) by varying the lift forces and the balance of reaction torques through changing the rotating speed of the rotors. Roll and pitch can be changed by increasing the speed of one rotor and decreasing the speed of the other rotor in a pair of rotors. The robot will tilt towards the slower rotor which leads then to an acceleration in the y- and x-axis respectively. The momentum stays constant during this maneuver. The yaw-angle is controlled by varying the speed of each pair of rotors will keep the sum of lifting forces constant, while the balance of reaction torque of the rotors changes. This yields an angular acceleration around the z-axis. Acceleration in the direction of the z-axis (height) is simply achieved by changing the speed of all rotors collectively.

The quadrotor is an unstable system, and therefore needs a low level attitude and heading controller that is provided by the on-board AutoPilot. It estimates the absolute attitude (roll and pitch angles) by fusing the acceleration vector measured by the linear acceleration sensors and angular velocities measured by the gyroscopes. The yaw-angle is estimated relatively by integrating the angular speed measured by the yaw-gyro. The three axes of rotation (roll, pitch, yaw) are then stabilized by the AutoPilot with three independent PD controllers at a control loop frequency of 1 kHz.[7]

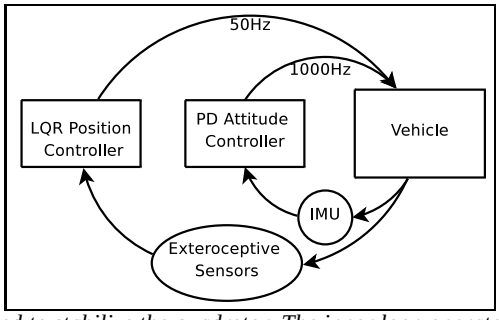


Figure 3. The two control loops employed to stabilize the quadrotor. The inner loop operates at 1 kHz to stabilize the attitude. The outer loop, operates at 50 Hz to stabilize the position.

While this low level controller is able to stabilize the attitude of the vehicle, it cannot perform position control because the IMU drift is too large to allow double-integration to be successful. Instead, we use two levels of feedback to control the vehicle and maintain a stable hover as shown in Figure 3. The inner loop of the on-board attitude controller stabilizes the high-frequency dynamics of the quadrotor at a rate of 1 kHz.

The slightly slower (although still underdamped and unstable) position control is performed using the state estimates generated by the modules described in Section 3. The on-board controller takes 4 inputs, $\vec{u} = [u_p, u_r, u_t, u_\theta]$, which denote the desired pitch and roll angles, overall thrust and yaw velocities. The on-board controller allows the helicopter's dynamics to be approximated with simpler linear equations:

$$\begin{aligned} \ddot{x}^b &= k_p u_p + b_p & \ddot{z} &= k_t u_t + b_t \\ \ddot{y}^b &= k_r u_r + b_r & \dot{\theta} &= k_\theta u_\theta + b_\theta \end{aligned} \tag{1}$$

where $\ddot{x}^b$ and $\ddot{y}^b$ are the resultant accelerations in body coordinates, while k_* and b_* are model parameters that are functions of the underlying physical system. We learn these parameters by flying inside a Vicon* motion-capture system and fitting parameters to the data using a least-squares optimization. Using Matlab®'s linear quadratic regulator (LQR) toolbox, we then find feedback controller gains for the dynamics model in Equation 1. Despite the model's apparent simplicity, our controller achieves a stable hover with $6cm$ RMS error.

2.3 Flight Termination System

To enable safe operation, the vehicle is able to switch from autonomous to manual control with the flip of a switch on the safety pilot's RC transmitter. The software running on the on-board processor constantly monitors the associated RC channel, and only allows input from the position controller if the switch is in the correct position. If both the RC and data-link connections are lost, the vehicle will attempt a controlled descent.

3. PAYLOAD

3.1 Sensors

Our quadrotor helicopter is outfitted with a laser rangefinder, stereo camera rig, color camera, and an on-board computer.

The lightweight Hokuyo† UTM laser rangefinder provides a $270°$ field-of-view at $40Hz$, up to an effective range of $30m$. We deflect some of the laser beams downwards to estimate height above the ground plane.

We built a custom stereo-camera rig providing mounts for two grayscale uEye cameras. These cameras have a resolution of 752×480px (WVGA), and are placed facing forward, with a baseline of 35 cm separation, and lenses with $65°$ FOV. By placing them as far apart from each other as possible, we increase the resolution available for stereo triangulation. An additional color uEye camera is mounted directly above the center of the vehicle.

We also mounted a Lippert CoreExpress 1.6Ghz Intel Atom board with a wifi link to the ground control station. This processor board provides the bandwidth and computational power required to transmit the stereo camera images to the ground station at 10Hz, as well as enabling our control and state estimation modules to be run on-board. Computation-heavy processes are run off-board at the ground station on a cluster of laptops.

*Vicon Motion Capture Systems. http://www.vicon.com/

†Hokuyo UTM-30LX Laser. https://www.hokuyo-aut.jp/

3.2 Laser Odometry

The Hokuyo laser rangefinder is used to estimate the vehicle's motion by aligning consecutive scans from the laser rangefinder. We developed a very fast and robust laser scan-matching algorithm that builds a high-resolution local map based on the past several scans, aligning incoming scans to this map at the $40Hz$ scan rate. This scan-matching algorithm is an improved version of the algorithm by Olson et al.,[13] which we have modified to allow for high resolution, yet realtime operation. The algorithm generates a local cost-map, from which the optimal rigid body transform that maximizes a given reward function can be found.

To find the best rigid body transform to align a new laser scan, we score candidate poses based on how well they align to past scans. Unfortunately, the laser scanners provide individual point measurements, and because successive scans will in general not measure the same points in the environment, attempting to correspond points directly can produce poor results. However, if we know the *shape* of the environment, we can easily determine whether a point measurement is consistent with that shape. We model our environment as a set of polyline contours, and these contours are extracted using an algorithm that iteratively connects the endpoints of candidate contours until no more endpoints satisfy the joining constraints. With the set of contours, we generate a cost-map that represents the approximate log-likelihood of a laser reading at any given location.

We create our cost-map from a set of k previous scans, where new scans are added when an incoming scan has insufficient overlap with the existing set of scans used to create the cost-map.

For each incoming scan, we compute the best rigid body transform (x, y, θ) relative to our current map. Many scan-matching algorithms use gradient descent techniques to optimize these values, but these methods are subject to local optima. Instead, we perform an exhaustive search over the grid of possible poses, which can be done fast enough by setting up the problem such that optimized image addition primitives can be used to process blocks of the grid simultaneously. In addition to improved robustness, generating the complete cost-surface allows us to easily quantify the uncertainty of our match, by looking at the shape of the cost surface around the maximum. As with the dynamics model, more details can be found in previous publications.

3.3 Visual Odometry

We also developed a visual odometry algorithm which makes use of the on-board stereo camera rig (note: some analysis, results and figures appeared previously in[2,3]). In general, a single camera is sufficient for estimating relative motion of the vehicle, by using the feature correspondences of consecutive image frames, but only up to an arbitrary scale factor.[12] If enough feature correspondences (at least 7-8) are available, the fundamental matrix describing the motion of the camera can be computed. Decomposing this matrix yields the relative rotation and translation motion of the camera, and while the rotation can be uniquely computed with respect to this scalar factor, we can only compute the translational direction of the motion, and not its magnitude.

To resolve this scale ambiguity, either scene knowledge is necessary, or two successive views must have distinct vantage points and a sufficiently large baseline between them. This makes estimation of motion in the direction of the camera's optical axis difficult. Given that prior knowledge of unknown environments is typically unavailable, and MAVs often move slowly and forward with the camera facing front, autonomous navigation for MAVs has proven to be very challenging . This motivates our choice for using a stereo camera to reconstruct the 3D-position of environmental features accurately. The stereo-rig not only enforces a baseline distance between the two cameras, but also allows us to reconstruct the feature positions in a single timestep, rather than using consecutive frames from a monocular camera. In this section, *left* and *right* denote the images taken from the left and right stereo cameras respectively, as seen from the helicopter's frame of reference.

Our approach for stereo visual odometry is outlined in Figure 4. Features are first detected in the *left* frame from the previous time-step (1). These features are then found in the previous *right* frame (2), enabling us to reconstruct their positions in 3D-space using triangulation. This "sparse-stereo" method not only avoids unnecessary computation of depth and depth error-handling in areas that lack features, but also avoids image rectification as long as the camera's optical axis are approximately parallel. Successfully reconstructed features are then tracked from the previous *left* to the current *left* frame (3), and a similar reconstruction step is performed for the current frames (4). This process results in two "clouds" of features that relate the previous and current views. With these correspondences, the quadrotor's relative motion in all 6 *dof* can be computed in a least-squares sense with a closed form solution (5).

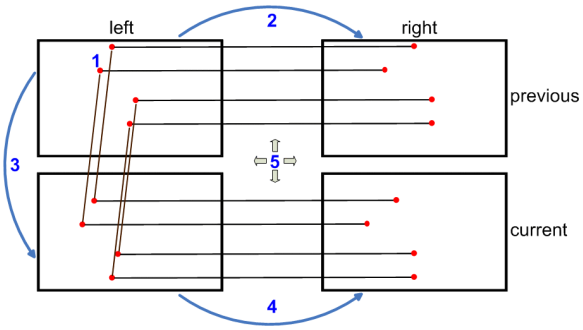


Figure 4. Basic scheme of the stereo visual odometry algorithm: 1. Feature detection; 2. Tracking from left *to* right *frame and depth reconstruction; 3. Tracking from previous to current frame; 4. Similar to 2.; 5. Frame to frame motion estimation.*

3.3.1 Feature detection

Although SIFT[11] and SURF[4] features are popular choices for visual feature detectors, computing them fast enough for our purposes on modern hardware remains computationally infeasible, given the control requirements of our quadrotors discussed in Section 1.2. Instead, we adopted the FAST[14] feature detector. In order to avoid detecting many distinct features that are in close geometric proximity to each other, we down-sample the image before running the feature-detector, and transform these feature locations to the full size image after detection. Unfortunately, many edge features are still detected by FAST, leading to inaccurate feature-tracking results. We therefore refine these corner estimates using the Harris corner-response[8] for each feature. A Harris parameter of 0.04 eliminates most of the edge features in the feature set. The use of the FAST detector beforehand allows us to compute the more computationally intensive image-gradients and corner-responses used by the Harris detector only for the areas detected by the FAST detector – a small subset of the entire image.

To track the features between the *left* and *right* frames, as well as from the previous to the current frames, we use the pyramidal implementation of the KLT optical flow tracker available in OpenCV.[5] This implementation allows us to track features robustly over large baselines and is robust to the presence of motion blur. Compared to template matching methods such as SAD, SSD and NCC, this algorithm finds feature correspondences with subpixel accuracy. For correspondences between the *left* and *right* frames, error-checking is done at this stage by evaluating the epipolar constraint $x_{right}^T F x_{left} = 0 \pm \epsilon$, where x denotes the feature location in the respective frame, F is the fundamental matrix pre-computed from the extrinsic calibration of the stereo rig, and ϵ is a pre-defined amount of acceptable noise.

3.3.2 Frame to frame motion estimation

Once we have the sets of corresponding image features, these are projected to a 3D-space by triangulation between the left and right cameras. Then, with these two sets of corresponding 3D-feature locations, we can estimate the relative motion between the previous and current time-steps using the closed form method proposed by Umeyama.[18] This method computes rotation and translation separately, finding an optimal solution in a least squares sense. Unfortunately, least square methods are sensitive to outliers, and we therefore use Umeyama's method to generate a hypothesis for the robust MSAC[16] estimator, a refinement of the popular RANSAC method. After finding a hypothesis with the maximum inlier set, the solution is recomputed using all inliers.

This method gives the transformation of two sets of points with respect to a fixed coordinate system. In our application, the set of points is fixed while the camera or MAV coordinate system is moving. The rotation ΔR and translation Δt of the helicopter to it's previous body frame are then given by $\Delta R = R^T$ and $\Delta t = -R^T t$, where R and t denote the rotation and translation which were derived from the method above. Using homogeneous transforms, the pose of the helicopter to an initial pose, T_0, is given by:

$$T_{current} = T_0 \cdot \Delta T_{t-n+1} \cdot \ldots \cdot \Delta T_{t-1} \cdot \Delta T_t = T_{previous} \cdot \Delta T_t \quad \text{with} \quad T = \begin{bmatrix} R & t \\ \underline{0} & 1 \end{bmatrix} \tag{2}$$

3.3.3 Nonlinear motion optimization

Similar to the laser scan-matching process, small measurement errors will accumulate over time and result in highly inaccurate position estimates over time, according to Equation 2. Additionally, since the motion between frames tends

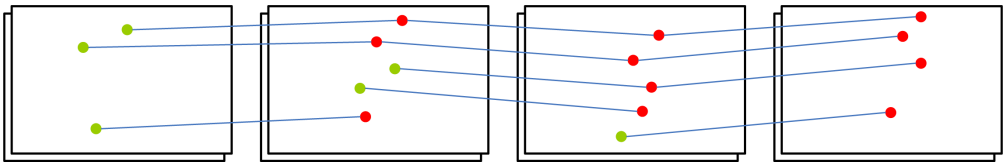


Figure 5. Bundle adjustment incorporates feature correspondences over a window of consecutive frames.

to be small, the velocity signal is highly susceptible to noise. However, because many of the visual features remain visible across more than two consecutive frames, we can estimate the vehicle motion across several frames to obtain more accurate estimates. This can be done using bundle adjustment[17] (shown in Figure 5), where the basic idea is to minimize the following cost function:

$$c(\mathbf{X}_i, R_j, t_j) = \sum_{i=0}^{m} \sum_{j=0}^{n} d(\mathbf{x}_{ij}, P_j \mathbf{X}_i)^2 \quad \text{with} \quad P_j = \begin{bmatrix} K_j R_j & K_j t_j \end{bmatrix} \tag{3}$$

where $d(\mathbf{x}_{ij}, P_j \mathbf{X}_i)$ is the re-projection error due to the projection of the 3D-feature, $\mathbf{X}_i$, onto the camera's image plane using the j-th view, so as to obtain the 2D-point, $\mathbf{x}_{ij}$. Here, m and n are the number of 3D-features and views respectively, while K_j is the intrinsic camera matrix, which is assumed to be constant.

We are therefore seeking to find the optimal arrangement of 3D-features and camera-motion parameters so as to minimize the sum of the squared re-projection errors. This problem can be solved using an iterative nonlinear least squares methods, such as the technique proposed by Levenberg-Marquardt. Here, normal equations with a Jacobian (J) from the re-projection function have to be solved. Although the cost function appears simple, the problem has a huge parameter space. We have a total of $3m + 6n$ parameters to optimize – three parameters for each 3D-feature and six parameters for each view. In addition, in each Levenberg-Marquardt step, at least m re-projections have to be computed per view. Computation in real-time therefore quickly becomes infeasible with such a large parameter space. Fortunately, J has a sparse structure, since each 3D-feature can be considered independent, and the projection of X_i into x_{ij} depends only on the j-th view. This sparse-bundle-adjustment problem can be solved using the generic package by Lourakis and Argyros,[10] and we used this package for our application.

Running bundle adjustment over all frames would quickly lead to computational intractability. Instead, we pursued a sliding-window approach, bundle-adjusting only a window of the latest n frames. By using the adjustment obtained from the old window as an initial estimate of the next bundle adjustment, we ensured that the problem is sufficiently-constrained while reducing the uncertainties due to noise. The presence of good initial estimates also reduces the number of optimization steps necessary. Performing bundle adjustment for 150 features using a window size of $n = 5$ took approximately 30-50ms.

Thus far, we have assumed that the feature correspondences necessary for bundle adjustment are known. These correspondences could be found by chaining the matches from frame-to-frame; unfortunately, our optical-flow-tracking approach does not compute the descriptors in the current frame when tracking the relative motion between the previous and current frames. Therefore, the Harris corner response for each feature is re-computed and sorted as described in Section 3.3.1. Given that the number of features diminishes over successive frames, new features are also added at every timestep, and when the new features are located close to old features, the old ones are preferred.

3.4 EKF Data Fusion

Having obtained relative position estimates of both the vehicle and environmental features from the individual sensors, these estimates can then be used in a sensor-independent manner for state estimation, control, map building and planning. We use an Extended Kalman Filter (EKF) to fuse the relative position estimates (x, y, z, θ) of the vehicle with the acceleration readings from the IMU. Using the open source KFilter library, we estimate the position, velocity, and acceleration of the vehicle, as well as biases in the IMU. We perform the measurement updates asynchronously, since the wireless communication link adds variable delays to the measurements, while the motion model prediction step is performed on a fixed clock. As per,[1] we learn the variance parameters by flying the helicopter in a Vicon motion-capture system, which provides ground-truth position and velocity values for comparing our state estimates against. We then run stochastic gradient descent to find a set of variance parameters that gives the best performance.

3.5 SLAM

With sufficiently accurate estimates of the vehicle's position and velocity to achieve a stable hover, we can use SLAM algorithms, originally developed for ground robots, on our MAV. With only slight modifications, we can close loops and create globally consistant maps. We chose the Gmapping[6] algorithm that is available in the OpenSlam repository. GMapping is an efficient Rao-Blackwellized particle filter which learns grid maps from laser range data. We chose it due to its outstanding accuracy, real-time performance, and its ability to handle the changing map that occurs due to changing height and attitude. Using this algorithm required 4 changes:

1. Modifying the motion model to use the variance estimates computed by the scan matcher
2. Making the map incorporate new information faster
3. Configuring the system to discard stale data when the map inference started to lag the arrival of new data
4. Finding a set of parameters that worked well with the shorter range Hokuyo laser scanners

3.6 Navigation and Exploration

To achieve autonomous operation, we require a planner that enables the vehicle to both navigate to desired locations autonomously, and explore unknown regions of the world. A navigation module was developed to facilitate motion-planning within the known grid map maintained by the SLAM process. The navigator generates a trajectory consisting of a sequence of waypoints in the environment to reach a goal location from the current vehicle position. This goal trajectory is computed using dynamic programming over a trajectory graph within the map, and balances finding short paths with those that promote safer execution.[9]

Since no prior map of the mission environment is available, an exploration module is required to determine where the quadrotor should fly to next based on the partially complete map maintained by the SLAM process. We use a frontier-based explorer,[19] where unknown map cells are grouped into "frontier" regions and serve as possible goal locations for the robot. These goals can be prioritized based on the expected information gain in each region, and the navigator subsequently used to generate trajectories to these frontier points.

3.7 Target Identification

To identify the target location within the environment, we developed a camera-based LED detector module that processes color video footage and identifies image regions with a high likelihood of containing the target blue LED. While navigating the environment, the LED detector is used to maintain an augmented map representation, noting regions of interest within the environment.

3.8 Threat Avoidance

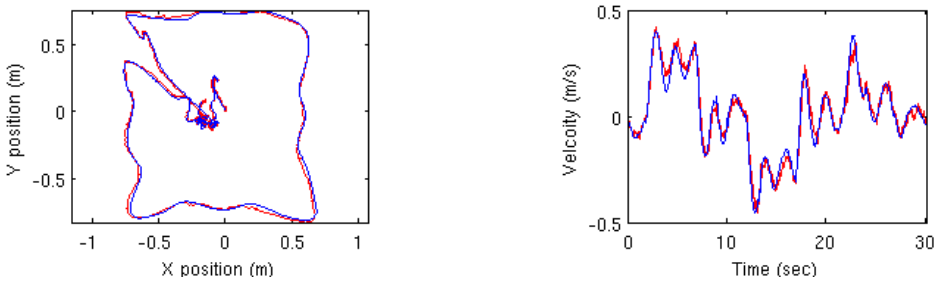
An obstacle avoidance routine ensures that the helicopter maintains a minimum distance from obstacles detected by the laser rangefinder or stereo cameras. This routine assumes control if an unmapped obstacle is encountered.

3.9 Experiments and Results

The technologies described above were evaluated in autonomous navigation tasks in unstructured and unknown indoor environments. During development and testing, experiments for the laser scanmatching and visual odometry algorithms were performed separately, and the corresponding sensors mounted on Ascending Technologies Hummingbird helicopters. Videos of our system in action are available at:
`http://groups.csail.mit.edu/rrg/videos.html`.

Figures 6(a) and 6(b) demonstrate the quality of our EKF state estimates using the laser range data. We compared the EKF state estimates with ground-truth state estimates recorded by the Vicon motion capture system, and found that the estimates originating from the laser range scans match the ground-truth values closely in both position and velocity. Throughout the 1 minute flight, the average distance between the two position estimates was less that 1.5cm. The average velocity difference was 0.02m/s, with a standard deviation of 0.025m/s. The vehicle was not given any prior information of its environment (i.e., no map).

Figure 7 demonstrates the quality of our EKF state estimates using the camera range data. We compared the EKF state estimates from the camera with ground-truth state estimates recorded by the Vicon motion capture system, and



(a) Position using the Laser (b) Velocity Accuracy using the Laser

Figure 6. (a) Comparison between the position estimated by the on-board laser sensor with ground truth measurements from an external camera array. (b) Comparison of the velocities during part of the trajectory.

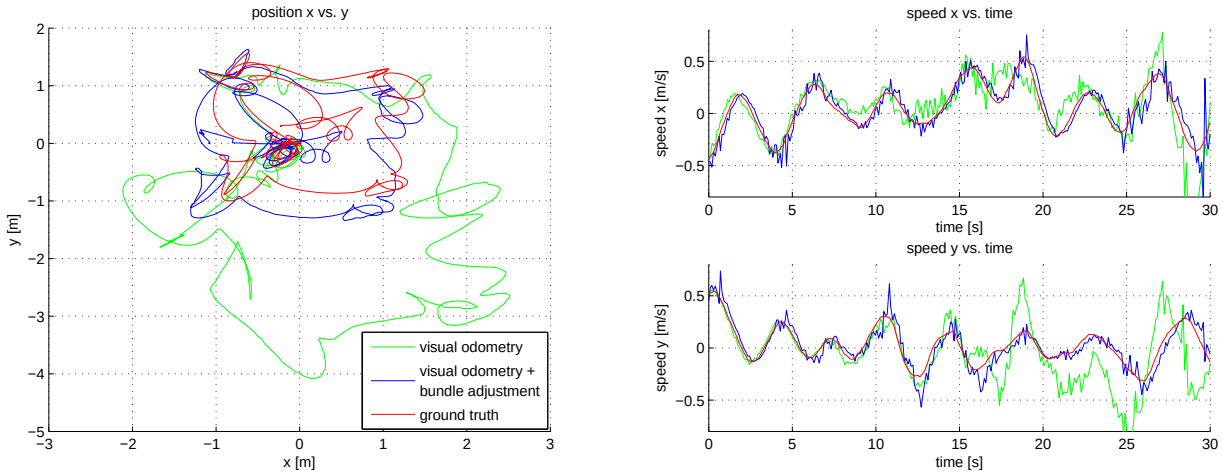


Figure 7. Position and speed over 1200 frames estimated by simple visual odometry from frame to frame (green) and by optimization with bundle adjustment (blue) compared to ground truth (red) obtained by the Vicon motion-capture system. The vehicle was flying with position control based on the estimates from bundle adjustment

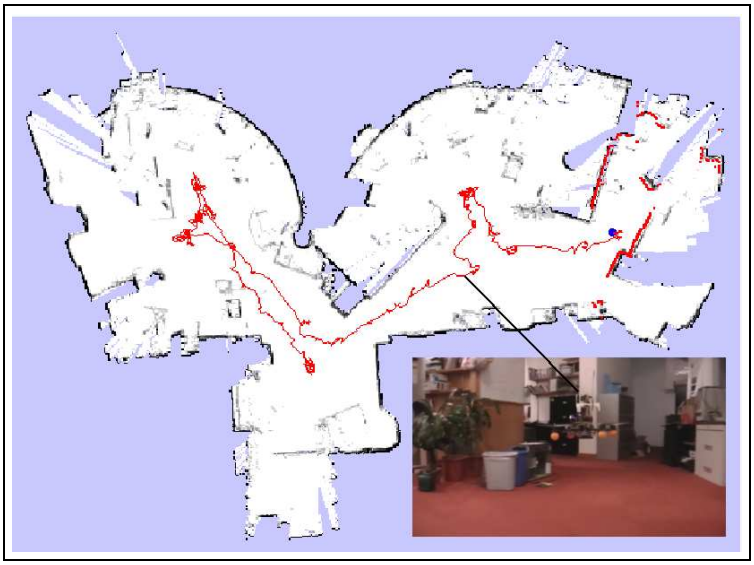
found that the estimates originating from the camera data also match the ground-truth values closely in both position and velocity. Additionally, the bundle adjustment substantially reduces the total error. As before, the vehicle was not given any prior information of its environment (i.e., no map).

Finally, we flew the laser-equipped helicopter 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 on-board the helicopter. The vehicle was guided by a human operator clicking high-level goals in the map that was being built in real-time. The vehicle was able to localize itself and fly stably throughout the environment, and Figure 8(a) shows the final map generated by the SLAM algorithm. While unstructured, the relatively vertical walls in the Stata Center environment allows the 2D map assumption of the laser rangefinder to hold fairly well.

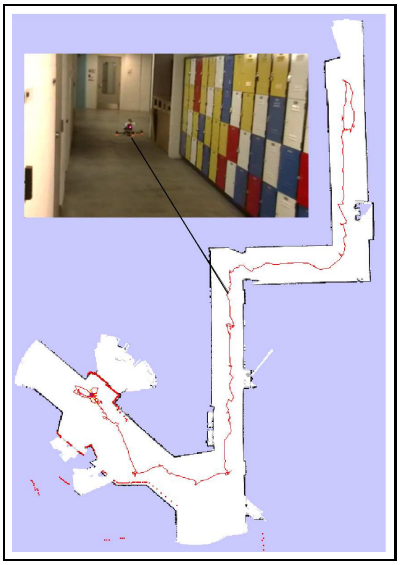
3.10 Autonomous navigation in cluttered environments

While unstructured, the relatively vertical walls in the Stata Center environment allows our 2D map assumption to hold fairly well. We next tested our system by flying through a cluttered lab space (insert of Figure 8(b)), operating close to the ground. At this height, chairs, desks, robots, plants, and other objects in the area make the 2D cross-sectional scan obtained by the laser rangefinder vary dramatically with changes in height, pitch, and roll. The resultant SLAM map of the environment is shown in Figure 8(b). The grey features littered within the otherwise free space denote the objects that clutter the environment and are occasionally sensed by the laser rangefinder. Despite the cluttered environment, our vehicle is able to localize itself and maintain a stable flight for 6 min over a distance of $44.6m$, a feat that would not have been possible with a static map assumption.

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



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



(c) Map of MIT Stata Center, basement.

Figure 8. (a) Map of the first floor of MIT's Stata center constructed by the vehicle during autonomous flight. (b) Map of a cluttered lab space with significant 3D structure. (c) Map of constrained office hallway generated under completely autonomous exploration.

3.11 Autonomous exploration in office hallways

Finally, to demonstrate fully autonomous operation of the vehicle, we closed the loop with the exploration algorithms. The helicopter was tasked to explore a hallway, shown in the insert of Figure 8(c). Once the helicopter took off and began exploring, we had no human control over the helicopter's actions as it autonomously explored the unknown environment. The helicopter continuously searched for new frontier goals and generated paths to these areas of new information. Figure 8(c) shows the map built from 7 min of autonomous flight, after traveling a distance of $75.8m$.

4. OPERATIONS

4.1 Flight Preparations

A process has been established to ensure safety and consistent operation when initializing the system and executing a mission. The process includes a hardware phase, ensuring the electro-mechanical integrity of the system, and a software phase that initializes all modules for mission deployment.

4.1.1 Flight Checklist

1. inspect vehicle propellers and linkages
2. test vehicle battery and plug into quadrotor
3. power on vehicle and ground station
4. test communication links and startup processes (clock synchronization, messaging daemons)
5. run the mission process manager, start all processes
6. ensure health of all processes
7. safety pilot: start the vehicle, enter active monitoring
8. ground station operator: in mission planner, start mission
9. flight termination upon mission completion, mission abort, or safety pilot override

4.2 Human-Robot Interface

A graphical user interface is used for starting and monitoring all processes, which provides a single interface for aggregating system operation and health monitoring of all software modules. A process file is loaded into the mission process manager, which provides an ordered list of each software module and the corresponding host machine it is run on. The process manager enables the operator to start and stop processes, and shows the health of the process (in easily identifiable color-coded states), the status of host communication links, and statistics on CPU utilization. An additional interface shows the mission status and aggregates the relevant output from all software modules. This interface enables the operator to start, pause, and re-start a mission, and provides display windows for camera modules and a current map output by the SLAM module that is annotated with information from the state estimator, navigator, and LED detector.

5. RISK REDUCTION

A number of measures have been taken to manage the safety risks of operating and testing the vehicle. The vehicle has been outfitted with several safety features, such as vibration-damped sensor mounts and blade guards that provide for the safety of the vehicle and humans in the environment in the event of a collision. In addition to mounted safety hardware, we use a simulation environment and a Vicon motion-capture system to provide a test harness for our software modules and algorithms during development. Further, we have implemented autonomous safety behaviors that identify known risks and react accordingly.

5.1 Vehicle Status and Safety

The vehicle status is constantly monitored both by software modules and human operators. Two human operators are required to monitor the vehicle status: one as a safety pilot with manual override capabilities; the other as the ground station operator that oversees the status of software modules and vehicle state estimation. Further, autonomous safety behaviors include an obstacle avoidance module and system vital monitors including power level and communication link health. In the event of communication loss or low battery voltage, the vehicle performs a controlled descent to the ground.

5.2 Modeling, Simulation and Testing

During software development, modules are tested in a 3D simulation environment and within a Vicon motion-capture system. The simulator provides a model of our quadrotor vehicle dynamics and position, laser, and camera sensor information. The motion-capture system is used when flying the physical vehicle to provide ground-truth position and velocity data for system identification and controller tuning.

After validation in the simulator, software modules are tested on the vehicle within the motion-capture system. A safety-harness controller, with motion-capture feedback, has been developed to automatically assume control of the vehicle if it leaves a prescribed flight region. This provides a safe and powerful intermediate testing environment for the vehicle before testing in real scenarios.

6. CONCLUSION

In this work, we have developed a quadrotor helicopter system that is capable of autonomous navigation in unknown and unstructured indoor environments using sensors on-board the vehicle. We have also developed a hierarchical suite of algorithms that accounts for the unique characteristics of air vehicles for estimation, control and planning, and can be applied with various exteroceptive sensors. In the near future, we hope to fully integrate the 3-dimensional information available in our sensor suite to perform autonomous MAV operations in rich indoor environments.

REFERENCES

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

[2] M. Achtelik. Vision-based pose estimation for autonomous micro aerial vehicles in gps-denied areas. Master's thesis, Technische Universität, München, Germany, 2009.

[3] Markus Achtelik, Abraham Bachrach, Ruijie He, Samuel Prentice, and Nicholas Roy. Stereo vision and laser odometry for autonomous helicopters in gps-denied indoor environments. In *Proceedings of the SPIE Conference on Unmanned Systems Technology XI*, Orlando, FL, 2009.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *In ECCV*, pages 404–417, 2006.

[5] Jean Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm, 2002.

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

[7] D. Gurdan, J. Stumpf, M. Achtelik, K.-M. Doth, G. Hirzinger, and D. Rus. Energy-efficient autonomous four-rotor flying robot controlled at 1 khz. *Proc. ICRA*, pages 361–366, April 2007.

[8] Chris Harris and Mike Stephens. A combined corner and edge detector. In *The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[9] R. He, S. Prentice, and N. Roy. Planning in information space for a quadrotor helicopter in a gps-denied environments. In *Proc. ICRA*, pages 1814–1820, Los Angeles, CA, 2008.

[10] M.I.A. Lourakis and A.A. Argyros. The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, Aug. 2004. Available from `http://www.ics.forth.gr/~lourakis/sba`.

[11] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004.

[12] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3D Vision: From Images to Geometric Models*. Springer Verlag, 2003.

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

[14] Edward Rosten and Tom Drummond. Machine learning for high speed corner detection. In *9th European Conference on Computer Vision*, May 2006.

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

[16] P. H. S. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding*, 78:2000, 2000.

[17] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 298–372, London, UK, 2000. Springer-Verlag.

[18] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(4):376–380, Apr 1991.

[19] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. CIRA*, pages 146–151. IEEE Computer Society Washington, DC, USA, 1997.