Stereo Vision and Laser Odometry for Autonomous Helicopters 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 unstructured and unknown indoor environments. We compare two sensor suites, specifically a laser rangefinder and a stereo camera. Laser and camera sensors are both well-suited for recovering the helicopter's relative motion and velocity. Because they use different cues from the environment, each sensor has its own set of advantages and limitations that are complimentary to the other sensor. Our eventual goal is to integrate both sensors on-board a single helicopter platform, leading to the development of an autonomous helicopter system that is robust to generic indoor environmental conditions. In this paper, we present results in this direction, describing the key components for autonomous navigation using either of the two sensors separately.

Keywords: Autonomous navigation, visual odometry, laser odometry, unmanned aerial vehicles

1. INTRODUCTION

Micro Aerial Vehicles (MAVs) are small, light-weight and relatively quiet flying machines that can accomplish many military and civilian tasks, including surveillance operations, weather observation, and disaster relief coordination. In recent years, there has been increased interest within the research community in developing MAVs that can operate autonomously in indoor environments,[1–3] thereby enabling an even wider range of robotic tasks to be accomplished. Indoor environments and many parts of the urban canyon do not have access to external positioning systems such as GPS. Therefore, robotic agents operating within such environments must rely on exteroceptive sensors such as laser rangefinders, sonars, or cameras to estimate their position. Such sensors have successfully been employed on-board unmanned ground and underwater vehicles using simultaneous localization and mapping (SLAM) algorithms. These algorithms enable the vehicles to build maps of the environment while simultaneously using the maps to estimate their position. Unfortunately, attempts to achieve the same results with MAVs have not been as successful, due to their fast, unstable dynamics, as well as their limited payloads for sensing and computation.

In addition, although exteroceptive sensors such as laser rangefinders and camera sensors are popular options for enabling MAVs to operate indoors, each sensor has unique characteristics that lead to effectiveness only in certain environments. Laser rangefinders are effective when the environment has unique physical structure or shape. Unfortunately, laser rangefinders with limited range can fail around homogeneous building structures such as long corridors. In addition, since most existing sensors only generate 2D slices of their local environments, laser-equipped MAVs are unable perceive the extent of obstacles outside the sensing plane, limiting their ability to reason about flying over obstacles instead of around them. In contrast, camera sensors are particularly useful for providing rich 3D information; however, they require the environment to contain unique visual features based on variable appearance. In addition, camera sensors have limited angular field-of-views and are computationally intensive to work with.

Different exteroceptive sensors are therefore better suited for autonomous MAV operations under different environmental conditions. However, since the laser scanner and cameras rely on different environmental features, they will have complimentary failure modes. As a result, integrating both sensors onto a single MAV platform, we will enable autonomous navigation in a wide range of generic, unstructured indoor environments, such as those shown in Figure 1(b).

Markus Achtelik is with the department of Electrical Engineering and Information Technology, Technische Universität München, Germany. E-mail: markus@achtelik.net; Abraham Bachrach, Ruijie He, Samuel Prentice and Nicholas Roy are members of the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139.E-mail: {abachrac|ruijie|prentice|nickroy}@mit.edu.

(a) Laser-equipped Quadrotor helicopter (b) Autonomous flight (c) Camera-equipped Quadrotor

Figure 1. Our laser-equipped quadrotor helicopter is shown in Fig 1(a). Sensing and computation components include a Hokuyo Laser Rangefinder (1), laser-deflecting mirrors for altitude (2), a monocular camera (3), an IMU (4), a Gumstix processor (5), and the helicopter's internal processor (6). Fig 1(b) demonstrates autonomous flight in unstructured indoor environments. Our quadrotor helicopter equipped with a stereo camera rig is shown in Fig 1(c).

This paper presents our initial results towards developing an integrated helicopter system capable of indoor flight using both laser-rangefinder and camera sensors on-board. We present two separate sensing solutions, a laser-rangefinder and a stereo camera system. We show that each is independently capable of autonomous navigation in unstructured indoor environments. Figure 1(a) shows our laser-equipped quadrotor helicopter, and our stereo-camera quadrotor is shown in Figure 1(c). The baseline helicopter platforms were built by Ascending Technologies.[4] The system is enabled by a multi-level sensor processing system architecture that is designed not only to meet the requirements for autonomous control and navigation of an indoor helicopter, but also to be independent of the sensing modality used. Indeed, our experimental results on both helicopter platforms demonstrate our ability to reuse the system architecture with either, or both, sensors on-board.

The key contributions of this paper are:

1. To summarize previously reported results on fully autonomous quadrotors that only rely on on-board sensors for stable control, without requiring prior maps of the environment.
2. A stereo camera visual odometry algorithm that allows stereo camera images to be processed in real-time, providing accurate velocity and relative position information.
3. A multi-level sensor processing hierarchy and system architecture that is independent of the on-board exteroceptive sensor used for obtaining position estimates.

After discussing some of the key challenges for performing autonomous navigation and SLAM on MAVs, we provide an overview of the systems that we have developed, as well as the overarching system architecture that enables easy interchanging of the exteroceptive sensors. We then describe the high-speed laser scan-matching and visual odometry algorithms for laser and stereo camera sensors respectively, before highlighting the remaining algorithms that are applicable to both sensors. Finally, we present results of both platforms navigating autonomously in different unstructured indoor environments.

2. KEY CHALLENGES FOR AUTONOMOUS MAVS

Combining wheel odometry with exteroceptive sensors in a probabilistic SLAM framework has proven very successful for ground robotics;[5] 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 have limited ranges, fields-of-view and are noisier 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 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.[6]

3. SYSTEM OVERVIEW

We addressed the problem of autonomous indoor flight as primarily a software challenge. To that end, we used off-the-shelf hardware throughout the system. Our quadrotor helicopter platforms were designed by Ascending Technologies GmBH, and are able to carry roughly $250g$ of payload.

For the laser-equipped quadrotor (Figure 1(a)), we outfitted it with a Gumstix microcomputer that provides a wifi link between the vehicle and a ground control station, and a lightweight Hokuyo laser rangefinder for localization. The laser rangefinder provides a $270°$ field-of-view at $40Hz$, up to an effective range of $30m$. We deflect some of the laser beams downwards to estimate height above the ground plane.

Similarly, we outfitted another quadrotor helicopter (Figure 1(c)) with a stereo camera rig, using cameras from uEye. 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. Because of the additional bandwidth and computational power required to process the stereo camera images, a Lippert CoreExpress 1.6Ghz Intel Atom board was used to transmit the stereo camera images to the ground control station at 10Hz, as well as to act as the control link with the quadrotor helicopter. Computation was done off-board on a 2.4Ghz Core2Duo laptop.

On its own, the AscTec Hummingbird helicopter is equipped with attitude stabilization, using an on-board IMU and processor to stabilize the helicopter's pitch and roll.[4] This dynamic model has been reported previously but is summarized here. 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:

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

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

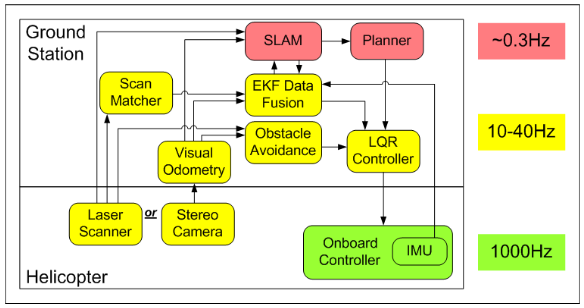


Figure 2. Schematic of our hierarchical sensing, control and planning system. At the base level, the onboard IMU and controller (green) creates a tight feedback loop to stabilize the helicopter's pitch and roll. The yellow modules make up the real-time sensing and control loop that stabilizes the helicopter's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

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.

Our system employs a three level sensing hierarchy, as shown in Figure 2. At the base level, the onboard 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, high frequency state estimates. 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.

Small errors made by the odometry algorithms will get propagated forward, resulting in inaccurate state estimates in large environments. To mitigate this, we introduce a third sensing level, where a SLAM algorithm uses both the EKF state estimates and incoming sensor measurements to create a global map, ensuring globally consistent state estimates by performing loop closures. Since SLAM algorithms today are incapable of processing the incoming 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.

4. SENSOR-DEPENDENT STATE-ESTIMATION ALGORITHMS

As discussed in Section 2, we cannot directly measure the MAV's odometry; instead, we need to estimate the helicopter's relative motion from sensor measurements. This section describes the algorithms that are specific to each of the two exteroceptive sensors used on our helicopter platforms. Both algorithms output estimates of the helicopter's relative position from the latest sensor measurement, as well as the relative positions of the obstacle features in the helicopter's local environment.

4.1 Laser: High-Speed Laser Scan-Matching Algorithm

For the quadrotor helicopter equipped with the Hokuyo laser rangefinder, we estimate the vehicle's motion by aligning consecutive scans from the laser rangefinder. We developed a very fast 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 a version of the algorithm by Olson et al.,[7] 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 maintain a cost-map that represents the approximate log-likelihood of a laser

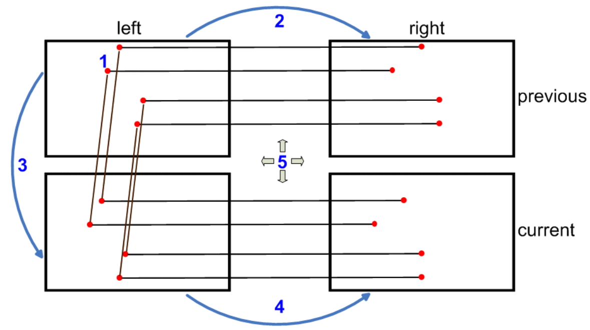


Figure 3. Basic scheme of the stereo visual odometry algorithm: 1. Feature detection; 2. Tracking from *left* to *right* frame and depth reconstruction; 3. Tracking from previous to current frame; 4. Similar to 2.; 5. Frame to frame motion estimation.

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.

4.2 Stereo: Visual Odometry Algorithm

For our quadrotor helicopter equipped with a stereo-camera rig, we developed a visual odometry algorithm that functioned as the stereo-camera equivalent of the laser scan-matching algorithm. 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.[8] 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, mainly because motion in the direction of the camera's optical axis cannot be computed accurately. Given that scene knowledge for 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 (refer to Section 7). 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 3. 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).

4.2.1 Feature detection

Although SIFT[9] and SURF[10] features are popular choices for visual feature detectors, computing them fast for our purposes on modern hardware remains computationally infeasible, given the control requirements of our quadrotors discussed in Section 2. Instead, we adopted the FAST[11] 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[12] 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.

Unfortunately, our current feature set might still contain features that are located in close proximity with other features; this incurs unnecessary computational cost and makes feature-tracking error-prone, since a large number of features results in a greater prevalence of local maxima. We therefore prune our feature set by computing the distance between all feature pairs, eliminating the feature with the smaller score if the distance is less than a specified threshold. While this process theoretically has a complexity of $O(n^2)$, the computation in practice is much faster because the features are already presorted by the FAST feature detector. These steps need 3-4 ms for about 500 feature candidates and leave around 150 valid features. After pruning out all the undesired features, the remaining feature locations are refined to subpixel accuracy based on image gradients; this process requires an additional 6 ms.

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.[13] 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.

4.2.2 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.[14] 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[15] 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:

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

4.2.3 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 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[16] (shown in Figure 4), where the basic idea is to minimize the following cost function:

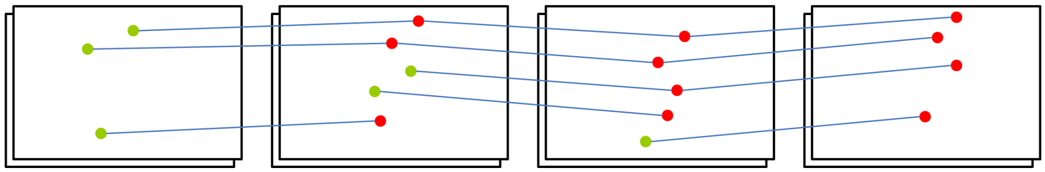


Figure 4. Bundle adjustment incorporates feature correspondences over a window of consecutive frames.

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

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,[17] and we used this package for our application.

The standard bundle adjustment approach is susceptible to outliers in the data, and because it is an iterative technique, good initial estimates are needed for the algorithm to converge quickly. We avoided both problems by using the robust frame-to-frame MSAC motion estimates, as described in Section 4.2.2, as well as their inlier sets of 3D features.

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.

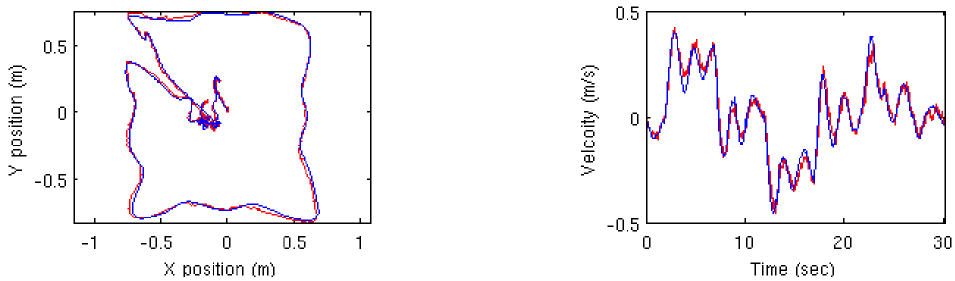
Thus far, we have assumed that the feature correspondences necessary for bundle adjustment are known. These correspondences could be found by chaining the matches from frame-to-frame; unfortunately, our optical-flow-tracking approach does not compute the descriptors in the current frame when tracking 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 4.2.1. 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.

4.3 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 both the IMU and those predicted by our control inputs. Using the open source KFilter library, we estimate the position, velocity, and acceleration of the vehicle, as well as biases in both the IMU and control inputs, in a 22-state vector. 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,[18] we learn the variance parameters by flying the helicopter in a Vicon Motion capture system, which provides ground-truth 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.

5. EXPERIMENTS AND RESULTS

We integrated the suite of technologies that were described above to perform autonomous navigation for both helicopters in unstructured and unknown indoor environments. Videos of our system in action are available at: `http://groups.csail.mit.edu/rrg/videos.html`.



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

Figure 5. (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.

We have reported analysis of the laser position estimates previously, but we reproduce the analysis here for comparison. Figures 5(a) and 5(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).

In contrast, Figure 6 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 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).

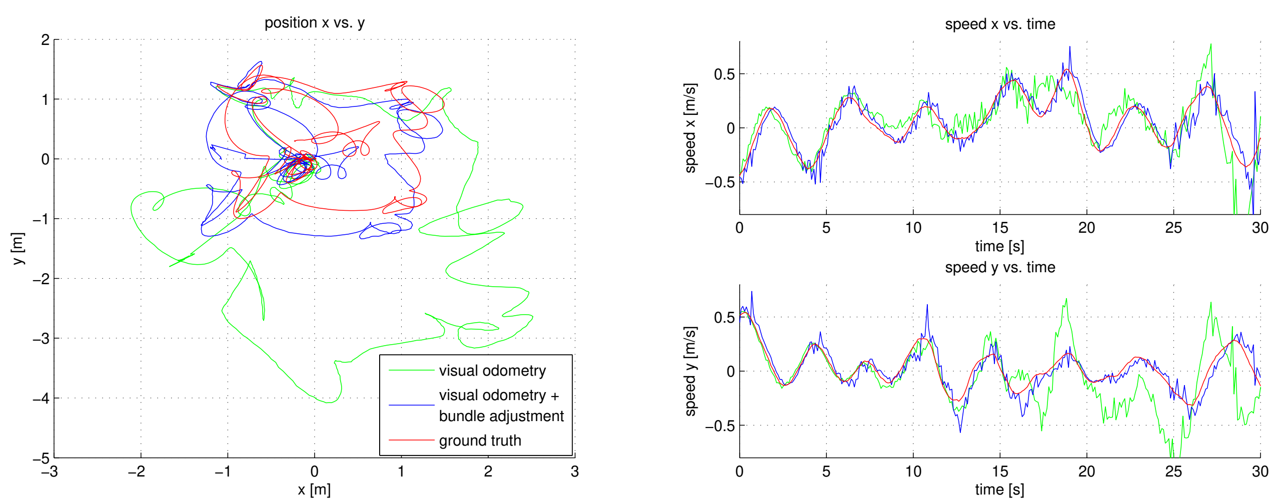


Figure 6. Position and speed over 1200 frames estimated by simple visual odometry from frame to frame (green) and by optimization with bundle adjustment (blue) compared to ground truth (red) obtained by a VICON system. The vehicle was flying with position control based on the estimates from bundle adjustment

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 7 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.

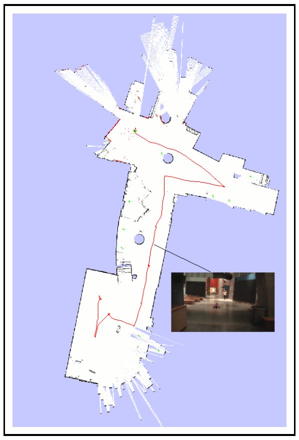


Figure 7. Map of the first floor of MIT's Stata Center constructed by the vehicle during autonomous flight.

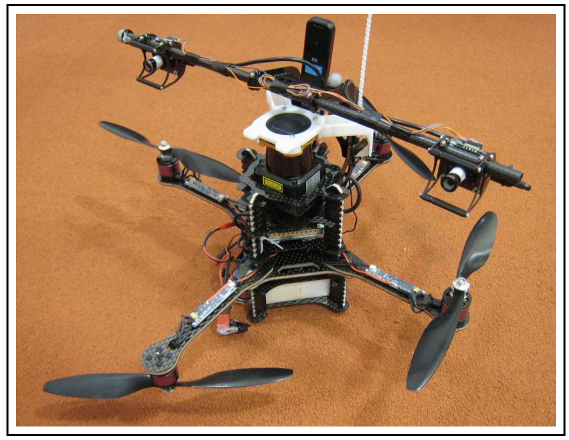


Figure 8. Quadrotor helicopter with Integrated Laser-Stereo Camera

6. EXTENSIONS AND FUTURE WORK

In this paper, we have demonstrated the ability of both a laser-equipped and stereo-camera-equipped quadrotor helicopter to operate autonomously in a variety of unstructured indoor environments. Nevertheless, as discussed in Section 1, both sensors have inherent limitations that constrain the range of autonomous operations that can be accomplished with a single sensor. Together with Ascending Technologies, we have recently developed a quadrotor helicopter that has a significantly higher payload (Figure 8), thus enabling us to mount both sensors on the same platform. We believe that integrating both laser and camera sensors on a single quadrotor helicopter will allow us to leverage on their respective strengths, enabling the helicopter to operate robustly indoors and build much richer maps of the environment. Specifically, the multi-level system hierarchy that we have developed allows us to interchange the exteroceptive sensor that is used to obtain the relative position estimates of the vehicle and environmental features, suggesting that we should be able to integrate both sensors relatively easily using the same system architecture.

7. RELATED WORK

Some previous work[19,20] shows results of indoor flight by relying on simulated GPS from motion capture systems. Others[21,22] use a small number of ultrasound sensors to perform altitude control and obstacle avoidance, developing helicopters are able to takeoff, land and hover autonomously; however, they are unable to achieve goal-directed flight. Tournier et al.[2] performed visual servoing over known Moire patterns to extract the full 6 *dof* state of the vehicle for control, Ahrens[23] extracted corner features that are fed into an EKF based Vision-SLAM framework, building a low-resolution 3D map sufficient for localization and planning. Recently, Angeletti et al.[24] and Grzonka et al.[25] designed helicopter configurations that were similar to the one presented by He at al.[6]

8. CONCLUSION

In this work, we have developed two separate quadrotor helicopters systems that are capable of autonomous navigation in unknown and unstructured indoor environments, rely only on sensors on-board the vehicle, and avoid relying on a prior map. We have also developed a hierarchical suite of algorithms that accounts for the unique characteristics of air vehicles for estimation, control and planning, and that can be used for both quadrotor helicopters with distinct exteroceptive sensors. In the near future, we hope to integrate both sensors onto a single helicopter platform, enabling us to perform autonomous MAV operations in fully 3-dimensional, generic indoor environments.

REFERENCES

[1] Celik, K., Chung, S., and Somani, A., "Mvcslam:mono-vision corner slam for autonomous micro-helicopters in gps denied environments," in [*Proc. of the AIAA GNC*], (2008).

[2] Tournier, G., Valenti, M., How, J., and Feron, E., "Estimation and control of a quadrotor vehicle using monocular vision and moirè patterns," in [*Proc. of AIAA GNC, Keystone, Colorado*], (2006).

[3] Langelaan, J., "State estimation for autonomous flight in cluttered environments," *Journal of Guidance, Control and Dynamics* **30**(5) (2007).

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

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

[6] He, R., Prentice, S., and Roy, N., "Planning in information space for a quadrotor helicopter in a gps-denied environments," in [*Proc. ICRA*], 1814–1820 (2008).

[7] Olson, E., *Robust and Efficient Robotic Mapping*, PhD thesis, MIT, Cambridge, MA, USA (June 2008).

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

[9] Lowe, D. G., "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision* **60**, 91–110 (2004).

[10] Bay, H., Tuytelaars, T., and Gool, L. V., "Surf: Speeded up robust features," in [*In ECCV*], 404–417 (2006).

[11] Rosten, E. and Drummond, T., "Machine learning for high speed corner detection," in [*9th European Conference on Computer Vision*], (May 2006).

[12] Harris, C. and Stephens, M., "A combined corner and edge detector," in [*The Fourth Alvey Vision Conference*], 147–151 (1988).

[13] Bouguet, J. Y., "Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm," (2002).

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

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

[16] Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W., "Bundle adjustment - a modern synthesis," in [*ICCV '99: Proceedings of the International Workshop on Vision Algorithms*], 298–372, Springer-Verlag, London, UK (2000).

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

[18] Abbeel, P., Coates, A., Montemerlo, M., Ng, A., and Thrun, S., "Discriminative training of kalman filters," in [*Proc. RSS*], (2005).

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

[20] Hoffmann, G., Huang, H., Waslander, S., and Tomlin, C., "Quadrotor helicopter flight dynamics and control: Theory and experiment," in [*Proc. of GNC*], (August 2007).

[21] Roberts, J., Stirling, T., Zufferey, J., and Floreano, D., "Quadrotor Using Minimal Sensing For Autonomous Indoor Flight," in [*Proc. EMAV*], (2007).

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

[23] Ahrens, S., *Vision-Based Guidance and Control of a Hovering Vehicle in Unknown Environments*, Master's thesis, MIT (June 2008).

[24] Angeletti, G., Valente, J. P., Iocchi, L., and Nardi, D., "Autonomous indoor hovering with a quadrotor," in [*Workshop Proc. SIMPAR*], 472–481 (2008).

[25] Grzonka, S., Grisetti, G., and Burgard, W., "Autonomous indoor navigation using a small-size quadrotor," in [*Workshop Proc. SIMPAR*], 455–463 (2008).