Visual Navigation for Micro Air Vehicles

Abraham Bachrach∗, Albert S. Huang∗, Daniel Maturana†, Peter Henry‡,
Michael Krainin†, Dieter Fox‡, and Nicholas Roy∗

∗Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA
Email: {abachrac,albert,nickroy}@csail.mit.edu
†Department of Computer Science, Pontificia Universidad Católica de Chile, Santiago, Chile. Email: dimatura@puc.cl
‡Department of Computer Science & Engineering, University of Washington, Seattle, WA
Email:{peter, mkrainin, fox}@cs.washington.edu

Abstract—Recent advances in state estimation, planning, mapping, and control algorithms combined with ever-increasing computational power available onboard micro air vehicles have enabled a number of researchers to demonstrate advanced autonomous capabilities in GPS-denied environments. Much of this work has focused around the use of laser range-finder sensors, however the use of a 2D sensor in a 3D environment requires a number of assumptions that limit the ability of the MAV to operate in unconstrained environments. In this talk, we describe a system for visual odometry and mapping using a stereo or RGB-D camera. By leveraging results from recent state-of-the-art algorithms and hardware, our system enables 3D flight and planning in cluttered environments using only onboard sensor data.

I. INTRODUCTION

Stable and precise control of an autonomous micro-air vehicle (MAV) demands fast and accurate estimates of the vehicle’s pose and velocity. Previously, we have developed algorithms for MAV flight in cluttered environments using planar LIDAR [2]. LIDAR sensors provide range measurements with unparalleled precision, but are unable to detect objects that do not intersect the sensing plane. Thus, they are most useful in environments characterized by vertical structures, and less so in more complex scenes. We have also developed a system for controlling the vehicle using stereo cameras [1], however that algorithm was too slow to run on the vehicle.

Estimating a vehicle’s 3D motion from sensor data typically consists of first estimating its relative motion at each time step from successive frames or scans. The 3D trajectory is then obtained by integrating the relative motion estimates. While often useful for local position control and stability, these methods suffer from long-term drift and are not suitable for building large-scale maps. To solve this problem, we incorporate our previous work on RGB-D Mapping [8], which detects loop closures and maintains a representation of consistent pose estimates for previous frames.

This paper presents our approach to providing an autonomous micro-air vehicle (MAV) with fast and reliable state estimates and a 3D map of its environment by using an onboard stereo or RGB-D camera and inertial measurement unit (IMU). Together, these allow the MAV to safely operate in cluttered, GPS-denied indoor environments. A more detailed coverage of the work presented here can be found in Huang et al. [12].

II. APPROACH

The problem we address is that of a quadrotor helicopter navigating in an unknown environment. The quadrotor must use the onboard sensors to estimate its own position (local estimation), build a dense 3D model of the environment (global simultaneous localization and mapping) and use this model to plan trajectories through the environment.

Our algorithms are implemented on the vehicle shown in Figure 1. The vehicle is a Pelican quadrotor manufactured by Ascending Technologies GmbH. The vehicle has a maximal dimension of 70 cm, and a payload of up to 1000 g. We have mounted a stripped down Microsoft Kinect sensor which is connected to the onboard flight computer. The flight computer, developed by the Pixhawk project at ETH Zurich [15], is a 1.86 GHz Core2Duo processor with 4 GB of RAM. The computer is powerful enough to allow all of the real-time estimation and control algorithms to run onboard the vehicle.

Following our previous work, we developed a system that decouples the real-time local state estimation from the global simultaneous localization and mapping (SLAM). The local state estimates are computed from visual odometry (section II-A), and to correct for drift in these local estimates the estimator periodically incorporates position corrections provided by the SLAM algorithm (section II-B). This architecture allows the SLAM algorithm to use much more processing time than would be possible if the state estimates from the SLAM algorithm were directly being used to control the vehicle.
A. Visual Odometry

The visual odometry algorithm that we have developed is based around a standard stereo visual odometry pipeline, with components adapted from existing algorithms. While most visual odometry algorithms follow a common architecture, a large number of variations and specific approaches exist, each with its own attributes. Our overall algorithm is most closely related to the approaches taken by Mei et al. [14] and Howard [11].

1) **Preprocessing**: A Gaussian pyramid is constructed to enable more robust feature detection at different scales. Each level of the pyramid corresponds to one octave in scale space. Features at the higher scales generally correspond to larger image structures in the scene, which generally makes them more repeatable and robust to motion blur.

2) **Feature Extraction**: Features are extracted at each level of the Gaussian pyramid using the FAST feature detector [16]. The threshold for the FAST detector is adaptively chosen to ensure a sufficient number of features are detected in each frame. Feature bucketing is employed to maintain a more uniform distribution of features. The depth corresponding to each feature is computed using either sparse stereo, or by querying the depth image provided by the RGB-D camera.

3) **Initial Rotation Estimation**: For small motions such as those encountered in successive image frames, the majority of a feature’s apparent motion in the image plane is caused by 3D rotation. Estimating this rotation allows us to constrain the search window when matching features between frames. We use the technique proposed by Mei et al. [14] to compute an initial rotation by directly minimizing the sum of squared pixel errors between downsampled versions of the current and previous frames.

4) **Feature Matching**: Each feature is assigned a descriptor consisting of the intensity values of the $9 \times 9$ pixel patch around the feature. Features are then matched across frames using a mutual-consistency check. The score of two features is the sum-of-absolute differences (SAD) of their feature descriptors [11]. A feature match is declared when two features have the lowest scoring SAD with each other, and they lie within the search window defined by the initial rotation estimation. Once an initial match is found, the feature location in the newest frame is refined to obtain a sub-pixel match. Refinement is computed by minimizing the sum-of-square errors of the descriptors, using ESM [3].

5) **Inlier Detection**: Although the constraints imposed by the initial rotation estimation substantially reduce the rate of incorrect matches, an additional step is necessary to further remove bad matches. We follow Howard’s approach of computing a graph of consistent feature matches, and then using a greedy algorithm to approximate the maximal clique in the graph [11, 9].

The graph is constructed to leverage the fact that rigid body motions are distance-preserving operations – the Euclidean distance between two features at one time should match their distance at another time. Thus, each feature match is a vertex in the graph, and an edge is formed between two feature matches if the 3D distance between the features does not change substantially. For a static scene, the set of inliers make up the maximal clique of consistent matches. The max-clique search is approximated by starting with an empty set of feature matches and iteratively adding the feature match with greatest degree that is consistent with all feature matches in the clique (Fig. 2). Overall, this algorithm has a runtime quadratic in the number of feature matches, but runs very quickly due to the speed of the consistency checking.

6) **Motion estimation**

The final motion estimate is computed from the feature matches in three steps. First, an initial motion is estimated using an absolute orientation method to find the rigid body motion minimizing the Euclidean distance between the inlier feature matches [10]. Second, the motion estimate is refined by minimizing feature reprojection error. This refinement step implicitly accounts for the fact that the depth uncertainty originates from the stereo matching in image space. Finally, feature matches exceeding a fixed reprojection error threshold are discarded from the inlier set and the motion estimate is refined once again.

To reduce short-scale drift, we additionally use a keyframe technique. Motion is estimated by comparing the newest frame against a reference frame. If the camera motion relative to the reference frame is successfully computed with a sufficient number of inlier features, then the reference frame is not changed. Otherwise, the newest frame replaces the reference frame after the estimation is finished. This simple heuristic serves to eliminate drift in situations where the camera viewpoint does not vary significantly, a technique especially useful when hovering.
The algorithms have been heavily optimized, using SIMD instructions to speed up computations where possible. On a 2.6 Ghz laptop computer, our algorithm requires roughly 15 ms per frame. The timing per stage is as follows. Preprocessing: 2.1 ms, feature extraction: 3.1 ms, initial rotation estimation: 1.0 ms, feature matching: 6.0 ms, inlier detection: 2.2 ms, and motion estimation required less than 0.1 ms. Runtimes on the MAV are slightly higher due to the slower clock speed, but are well within real-time.

B. Mapping

Visual odometry provides locally accurate pose estimates; however, global consistency is needed for metric map generation and navigation over long time-scales. We therefore integrate our visual odometry system with our previous work in RGBD-Mapping [8]. This section focuses on the key decisions required for real-time operation; we refer readers to our previous publication for details on the original algorithm that emphasizes mapping accuracy [8].

Unlike the local pose estimates needed for maintaining stable flight, map updates and global pose updates are not required at a high frequency and can therefore be processed on an offboard computer. The groundstation computer detects loop closures, computes global pose corrections, and constructs a 3D log-likelihood occupancy grid map. For coarse navigation, we found a 10 cm resolution to provide a useful balance between map size and precision. Depth data is downsampled to 128 x 96 prior to a voxel map update to increase the update speed, resulting in spacing between rays of approximately 5 cm at a range of 6 m. Incorporating a single frame to the voxel map currently takes approximately 1.5 ms.

We employ a keyframe approach to loop closure – new frames are matched against a small set of keyframes to detect loop closures, using a fast image matching procedure [8]. When a new keyframe is added, a RANSAC procedure over FAST keypoints [16] with Calonder randomized tree descriptors [5] is run. The correspondence is accepted if there are at least 10 inliers with small enough reprojection errors after RANSAC. The final refined relative pose between keyframes is obtained by solving a two-frame sparse bundle adjustment system [13], which minimizes overall reprojection error. The final pose graph is optimized using TORO [6].

C. State estimation and control

To control the quadrotor, we integrate the new visual odometry and RGBD Mapping algorithms into our system previously developed around 2D laser scan-matching and SLAM [2]. The motion estimates computed by the visual odometry are fused with measurements from the onboard IMU in an Extended Kalman Filter. The filter computes estimates of both the position and velocity, which are used by the PID position controller to stabilize the position of the vehicle.

We keep the SLAM process separate from the real-time control loop, instead having it provide corrections for the real-time position estimates. Since these position corrections are delayed significantly from when the measurement upon which they were based was taken, we incorporate the correction by retroactively modifying the appropriate position estimate in the state history. All future state estimates are then recomputed from this corrected position, resulting in globally consistent real-time state estimates.

III. Experiments

We have conducted a number of autonomous flight experiments, both in a motion capture studio, and in larger environments at a number of locations around the MIT campus, and the Intel Research office in Seattle. The vehicle flew autonomously with state estimates provided by the algorithms presented in this paper. The vehicle was commanded through the environment by a human operator selecting destination waypoints using a graphical interface.

Figure 3 shows an example where the MAV was commanded to hover at a target point, along with statistics about how well it achieved this goal. Ground truth was recorded by a motion capture system. In larger environments, the SLAM algorithm limits the global drift on its position estimates by detecting loop closures and correcting the trajectory estimates. The trajectory history can then be combined with the depth data to automatically generate maps that are useful both for a human operator’s situational awareness, and for autonomous path planning and decision making. Figure 5 shows a rendering of the MAV’s internal state estimates as it flew through the environment depicted in Figure 6, and a path planned using the occupancy map and a simple dynamic programming search strategy. A video demonstrating
IV. DISCUSSION AND FUTURE WORK

The system described in this paper enables autonomous MAV flight in many unknown indoor environments. However, there remain a great number more challenging situations that would severely tax our system’s abilities. Motion estimation algorithms based on matching visual features, such as ours and virtually all other visual odometry techniques, do not perform as well in regions with few visual features. In large open areas, the visible structure is often far beyond the maximum range of the stereo based sensors.

Handling these challenges will likely require integration of other sensors such as laser range-finders. As these sensors have different failure modes, they serve to complement each other’s capabilities. Additional sensing modalities can reduce, but not eliminate, state estimation failures. Further robustness can be gained by designing planning and control systems able to respond appropriately when the state estimates are extremely uncertain [4], or to plan in ways that minimize future uncertainty [7].

Looking further afield, the current system allows the vehicle to reason about the world in places that is has mapped, or that are within the sensing horizon of the range sensors. However, it will usually not have information in the direction of travel. Extending the planning horizon will require us to break the dependence on metric range sensing, and leverage the qualitative information about the environment available in monocular imagery. In doing so, we hope to continue to increase the capabilities of the vehicle to plan and explore in large scale unknown environments.

ACKNOWLEDGMENTS

This research was supported by the Office of Naval Research under MURI N00014-07-1-0749 and the Army Research Office under the MAST CTA.
REFERENCES


