Estimation, Planning and Mapping for Autonomous Flight Using an RGB-D Camera in GPS-denied Environments

Abraham Bachrach*, Samuel Prentice*, Ruijie He, Peter Henry, Albert S. Huang, Michael Krainin, Daniel Maturana, Dieter Fox and Nicholas Roy

Abstract RGB-D cameras provide both color images and per-pixel depth estimates. The richness of this data and the recent development of low-cost sensors have combined to present an attractive opportunity for mobile robotics research. In this paper, we describe a system for visual odometry and mapping using an RGB-D camera, and its application to autonomous flight. By leveraging results from recent state-of-the-art algorithms and hardware, our system enables 3D flight in cluttered environments using only onboard sensor data. All computation and sensing required for local position control are performed onboard the vehicle, reducing the dependence on unreliable wireless links. However, even with accurate 3D sensing and position estimation, some parts of the environment have more perceptual structure than others, leading to state estimates that vary in accuracy across the environment. If the vehicle plans a path without regard to how well it can localize itself along that path, it runs the risk of becoming lost or worse. We show how the Belief Roadmap (BRM) algorithm (Prentice and Roy, 2009), a belief space extension of the Probabilistic Roadmap algorithm, can be used to plan vehicle trajectories that incorporate the sensing model of the RGB-D camera. We evaluate the effectiveness of our system for controlling a quadrotor micro air vehicle, demonstrate its use for constructing detailed 3D maps of an indoor environment, and discuss its limitations.

Abraham Bachrach and Samuel Prentice contributed equally to this work.

Abraham Bachrach, Samuel Prentice, Ruijie He, Albert Huang and Nicholas Roy
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139.
e-mail: abachrac, ruijie, albert, prentice, nickroy@mit.edu

Peter Henry, Michael Krainin and Dieter Fox
University of Washington, Department of Computer Science & Engineering, Seattle, WA.
e-mail: peter, mkrainin, fox@cs.washington.edu.

Daniel Maturana
The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. e-mail: dimatura@cmu.edu

1 Introduction

Unmanned air vehicles (UAVs) rely on accurate knowledge of their position for decision-making and control. As a result, considerable investment has been made towards improving the availability of global positioning infrastructure, including utilizing satellite-based GPS systems and developing algorithms to use existing RF signals such as WiFi. However, most indoor environments and many parts of the urban canyon remain without access to external positioning systems, limiting the ability of current autonomous UAVs to fly through these areas.

Localization using sonar ranging (Leonard and Durrant-Whyte, 1991), laser ranging (Thrun et al., 2000) or camera sensing (Se et al., 2002) has been used extremely successfully on a number of ground robots and is now essentially a commodity technology. Previously, we have developed algorithms for MAV flight in cluttered environments using laser range finders (Bachrach et al., 2009a) and stereo cameras (Achtelik et al., 2009). Laser range finders that are currently available in form factors appropriate for use on a MAV are very high precision, but only provide range measurements along a plane around the sensor. Since these sensors can only detect objects that intersect the sensing plane, they are most useful in environments characterized by vertical structures, and less so in more complex scenes.

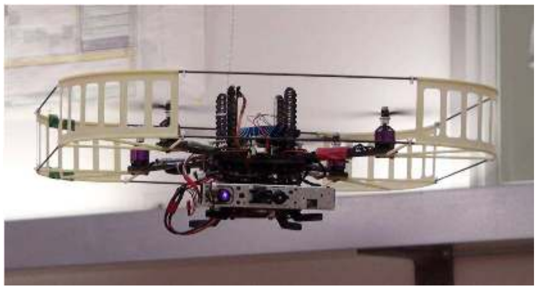


Fig. 1 Our quadrotor micro air vehicle (MAV). The RGB-D camera is mounted at the base of the vehicle, tilted slightly down.

Structured light RGB-D cameras are based upon stereo techniques, and thus share many properties with stereo cameras. The primary differences lie in the range and spatial density of depth data. Since RGB-D cameras illuminate a scene with a structured light pattern, they can estimate depth in areas with poor visual texture but are range-limited by their projectors. This paper presents our approach to providing an autonomous micro air vehicle with fast and reliable state estimates and a 3D map of its environment by using an on-board RGB-D camera and inertial measurement unit (IMU). Together, these allow the MAV to safely operate in cluttered, GPS-denied indoor environments. The control of a micro air vehicle requires accurate estimation of not only the position of the vehicle but also the velocity – estimates that our algorithms are able to provide. Estimating a vehicle's 3D motion from sensor data typically consists of estimating its relative motion at each time step by aligning successive sensor measurements such as laser scans or RGB-D frames, a process most often known as “visual odometry” when comparing camera or RGB-D images.

Given knowledge of the relative motion of the vehicle from sensor frame to sensor frame, the 3D trajectory of the vehicle in the environment can be estimated by integrating the relative motion estimates over time. Given knowledge of the vehicle position in the environment, the locations of obstacles in each sensor frame can also be used to construct a global map. While often useful for local position control and stability, visual odometry methods suffer from long-term drift and are not suitable for building large-scale maps. To solve this problem, we also demonstrate how our previous work on RGB-D Mapping (Henry et al., 2010) can be incorporated to detect loop closures, correct for accumulated drift and maintain a representation of consistent pose estimates over the history of previous frames.

However, the fast dynamics of UAVs have stringent requirements in terms of state estimation accuracy. The RGB-D sensor has a limited range and field of view, which strongly affects its reliability for state estimation on UAVs. When position information is temporarily not available from the onboard sensors, the state estimate will diverge from the true state much faster than on a ground vehicle, giving the UAV greater sensitivity to sensor limitations as it moves through the environment. Our approach to addressing this sensitivity is based on the Belief Roadmap (BRM) algorithm (Prentice and Roy, 2007; He et al., 2008a; Prentice and Roy, 2009). The BRM is a generalization of the Probabilistic Roadmap (PRM) algorithm (Kavraki et al., 1996), performing searches in the belief space of the vehicle efficiently by using the symplectic form of the Kalman Filter (KF) (Abou-Kandil, 2003) to find the minimum expected cost path for the vehicle.

In this paper, we provide two primary contributions. Firstly, we provide a systematic experimental analysis of how the best practices in visual odometry using an RGB-D camera enable the control of a micro air vehicle. Secondly, we give an extension of the BRM planning algorithm for a quadrotor helicopter (Figure 1). We describe our overall system, justify the design decisions made, provide a ground-truth evaluation, and discuss its capabilities and limitations. We conclude the paper with a demonstration of the quadrotor helicopter using the BRM algorithm to navigate autonomously in indoor environments.

This paper extends preliminary results given by Huang et al. (2011) and by He et al. (2008a), demonstrating the RGB-D mapping algorithm and the BRM algorithm. We give additional algorithmic details regarding estimation and mapping, provide the extension of the BRM to other sensor modalities such as the RGB-D camera, and give a more thorough experimental analysis in real-world environments.

2 Vehicle Position Estimation

The problem we address is that of quadrotor helicopter navigation. The quadrotor must use the onboard RGB-D sensor 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 $70cm$, and a payload of up to $1000g$. We have mounted a stripped down Microsoft Kinect sensor and connected it to the onboard flight computer. The flight computer, developed by the Pixhawk project at ETH Zurich (Meier et al., 2011), 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 2.1), and to correct for drift in these local estimates the estimator periodically incorporates position corrections provided by the SLAM algorithm (section 2.2). 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 used to control the vehicle.

If the UAV does not have access to perfect state knowledge, such as from external sources (e.g., motion capture, GPS, etc.), it can localize itself by first using sensors to measure environmental features and then by registering the measurements against a pre-existing map. To control the quadrotor, we integrated the new visual odometry and RGB-D Mapping algorithms into our system previously developed around 2D laser scan-matching and SLAM (Bachrach et al., 2009a). The motion estimates computed by the visual odometry are fused with measurements from the onboard IMU in an Extended Kalman Filter, described in Appendix A. 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 must account for this delay when 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. By incorporating the SLAM corrections after the fact, we allow the real-time state estimates to be processed with low enough delay to control the MAV, while still incorporating the information from SLAM to ensure drift free position estimation.

2.1 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.

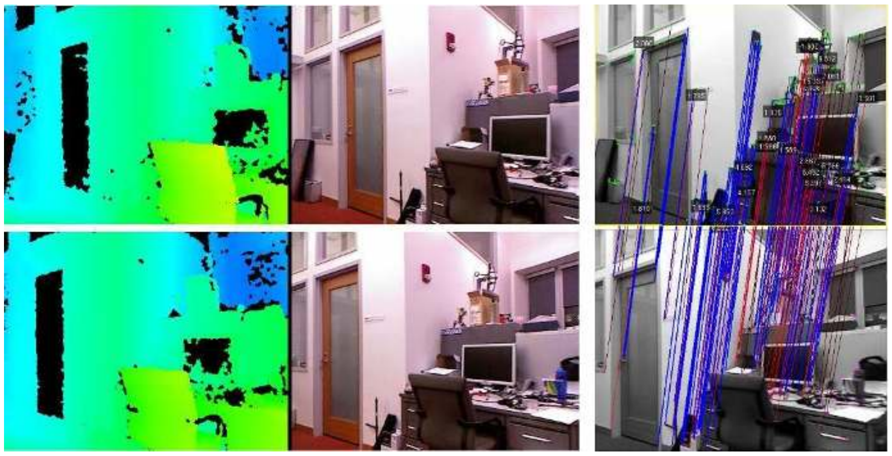


Fig. 2 The input RGB-D data to the visual odometry algorithm alongside the detected feature matches. The top row images are from time t, the bottom row images are from time $t + 1$. The left column is the depth image, and the middle column is the corresponding RGB image. The right hand column shows the pixels that are matched between frames, where inlier feature matches are drawn in blue and outliers are drawn in red.

While most visual odometry algorithms follow a common architecture, a large number of variations and specific approaches exist, each with its own attributes. A contribution of this paper is to specify the steps of our visual odometry algorithm and compare the alternatives for each step. In this section we specify these steps and we provide the experimental comparison of each step in the visual odometry pipeline. Our overall algorithm is most closely related to the approaches taken by Mei et al. (2009) and Howard (2008), and consists of the following sequence of operations:

1. **Image Preprocessing:** An RGB-D image is first acquired from the RGB-D camera (Fig. 2). The RGB component of the image is converted to grayscale and smoothed with a Gaussian kernel of $\sigma = 0.85$, and 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 (Rosten and Drummond, 2006). The threshold for the FAST detector is adaptively chosen using a simple proportional controller to ensure a sufficient number of features are detected in each frame. The depth corresponding to each feature is also extracted from the depth image. Features that do not have an associated depth are discarded. To maintain a more uniform distribution of features, each pyramid level is discretized into $80{\times}80$ pixel buckets, and the 25 features in each bucket with the strongest FAST corner score are retained.

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. (2009) to compute an initial rotation by directly minimizing the sum of squared pixel errors between downsampled versions of the current and previous frames.

   One could also use an IMU or a dynamics model of the vehicle to compute this initial motion estimate, however the increased generality of the image based estimate is preferable, while providing sufficient performance. An alternative approach would be to use a coarse-to-fine motion estimation that iteratively estimates motion from each level of the Gaussian pyramid, as proposed by Johnson et al. (2008).
4. **Feature Matching:** Each feature is assigned an 80-byte descriptor consisting of the brightness values of the 9×9 pixel patch around the feature, normalized to zero mean and omitting the bottom right pixel. The omission of one pixel results in a descriptor length more suitable for vectorized instructions. Features are then matched across frames by comparing their feature descriptor values using a mutual-consistency check (Nistér et al., 2004). The match score between two features is the sum-of-absolute differences (SAD) of their feature descriptors (Howard, 2008), which can be quickly computed using SIMD instructions such as Intel SSE2. 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 to solve the iterative nonlinear least squares problem (Benhimane and Malis, 2004). We also use SIMD instructions to speed up this process.
5. **Inlier Detection:** Although the constraints imposed by the initial rotation estimation substantially reduce the rate of incorrect feature matches between frames, an additional step is necessary to further prune away 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 (Howard, 2008; Hirschmuller et al., 2002).

   The graph is constructed according to 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 pair of matched features across frames is a vertex in the graph, and an edge is formed between two such pairs of matched feature if the 3D distance between the features does not change substantially from the prior frame to the subsequent frame. 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 matched feature pairs and iteratively adding matched feature pairs with greatest degree that is consistent with all matched feature pairs in the clique (Fig. 2). Overall,

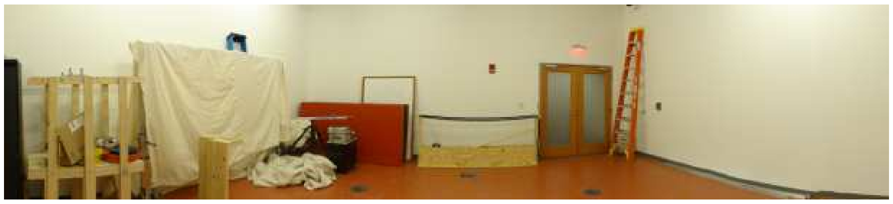


Fig. 3 Panorama photograph of the motion capture room used to conduct our ground-truth experiments. Visual feature density varies substantially throughout this room.

this algorithm has a runtime quadratic in the number of matched feature pairs, but runs very quickly due to the speed of the consistency checking. In our experimental analysis, we compare this approach to RANSAC-based methods (Nistér et al., 2004; Konolige et al., 2007).

6. **Motion estimation:** The final motion estimate is computed from the matched features in three steps. First, Horn's absolute orientation method provides an initial estimate by minimizing the Euclidean distances between the inlier feature matches (Horn, 1987). Second, the motion estimate is refined by minimizing feature reprojection error using a nonlinear least-squares solver (Benhimane and Malis, 2004). This refinement step implicitly accounts for the increase in depth uncertainty with range due to the fact that the depth estimates are computed by 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. If motion estimation against the reference frame fails, then the motion estimation is tried again with the second most recent frame. This simple heuristic serves to eliminate drift in situations where the camera viewpoint does not vary significantly, a technique especially useful when hovering.

2.1.1 Visual Odometry Performance

There are a variety of visual odometry methods, and the existing literature is often unclear about the advantages and limitations of each. We present results comparing a number of these approaches and analyze their performance. As is true in many domains, the tradeoffs can often be characterized as increased accuracy at the expense of additional computational requirements. In some cases, the additional cost is greatly offset by the improved accuracy.

We conducted a number of experiments using a motion capture system that provides 120 Hz ground truth measurements of the MAV's position and attitude. The motion capture environment can be characterized as a single room approximately $11m \times 7m \times 4m$ in size, lit by overhead fluorescent lights and with a wide variation of visual clutter – one wall is blank and featureless, and the others have a varying number of objects and visual features (see Fig. 3). While this is not a large volume, it is representative of many confined, indoor spaces, and provides the opportunity to directly compare against ground truth.

We recorded a dataset of the MAV flying various patterns through the motion capture environment, designed to challenge vision-based approaches to the point of failure, and includes motion blur and feature-poor images, as would commonly be encountered indoors and under moderate lighting conditions. Substantial movement in X, Y, Z, and yaw were all recorded, with small deviations in roll and pitch. We numerically differentiated the motion capture measurements to obtain the vehicle's ground truth 3D velocities, and compared them to the velocities and trajectories estimated by the visual odometry and mapping algorithms. Table 1 shows the performance of our integrated approach, and its behavior when adjusting different aspects of the algorithm. Each experiment varied a single aspect from our approach. We present the mean velocity error magnitude, the overall computation time per RGB-D frame, and the *gross failure rate*. We define a gross failure to be any instance where the visual odometry algorithm was either unable to produce a motion estimate (e.g., due to insufficient feature matches) or where the error in the estimated 3D velocities exceeded a fixed threshold of 1 m/s. Timing results were computed on a 2.67 GHz laptop computer.

Visual Odometry Variations

In developing our overall approach to visual odometry, we assessed different variants of the following steps of the process:

Inlier detection RANSAC based methods (Nistér et al., 2004) are more commonly used than the greedy max-clique approach. We tested against two RANSAC schemes, traditional RANSAC and Preemptive RANSAC (Nistér, 2005). The latter attempts to speed up RANSAC by avoiding excessive scoring of wrong motion hypotheses. In our experiments, when allocated a comparable amount of computation time (by using 500 hypotheses), greedy max-clique outperformed both.

Initial rotation estimation A good initial rotation estimate can help constrain the feature matching process and reduce the number of incorrect feature matches. Disabling the rotation estimate results in slightly faster runtime, but more frequent estimation failures.

Gaussian pyramid levels Detecting and matching features on different levels of a Gaussian pyramid provides provides resilience against motion blur and helps track larger features.

Reprojection error We compared unidirectional motion refinement, which minimizes the reprojection error of newly detected features onto the reference frame,

	Velocity error in m/s	% gross failures	total time in ms
Our approach	0.387 ± 0.004	3.39	14.7
Inlier detection			
RANSAC	0.412 ± 0.005	6.05	15.3
Preemptive RANSAC	0.414 ± 0.005	5.91	14.9
Greedy max-clique – *our approach*	0.387 ± 0.004	3.39	14.7
Initial rotation estimate			
None	0.388 ± 0.004	4.22	13.6
Gaussian pyramid levels			
1	0.387 ± 0.004	5.17	17.0
2	0.385 ± 0.004	3.52	15.1
3 – *our approach*	0.387 ± 0.004	3.39	14.7
4	0.387 ± 0.004	3.50	14.5
Reprojection error minimization			
Bidir. Gauss-Newton	0.387 ± 0.004	3.24	14.7
Bidir. ESM – *our approach*	0.387 ± 0.004	3.39	14.7
Unidir. Gauss-Newton	0.391 ± 0.004	3.45	14.6
Unidir. ESM	0.391 ± 0.004	3.47	14.6
Absolute orientation only	0.467 ± 0.005	10.97	14.4
Feature window size			
3	0.391 ± 0.004	5.96	12.8
5	0.388 ± 0.004	4.24	13.7
7	0.388 ± 0.004	3.72	14.2
9 – *our approach*	0.387 ± 0.004	3.39	14.7
11	0.388 ± 0.004	3.42	15.7
Subpixel feature refinement			
No refinement	0.404 ± 0.004	5.13	13.1
Adaptive FAST threshold			
Fixed threshold (10)	0.385 ± 0.004	3.12	15.3
Feature grid/bucketing			
No grid	0.398 ± 0.004	4.02	24.6

Table 1 Comparison of various approaches on a challenging dataset. Error computed using a high resolution motion capture system for ground truth.

with bidirectional refinement, which additionally minimizes the reprojection error of reference features projected onto the new frame. We additionally compared a standard Gauss-Newton optimization technique with ESM. Bidirectional refinement does provide slightly more accuracy without substantially greater cost, and we found no significant difference between Gauss-Newton and ESM.

Feature window size As expected, larger feature windows result in more successful motion estimation at the cost of additional computation time. Interestingly, a very small window size of 3×3 yielded reasonable performance, a behavior we attribute to the constraints provided by the initial rotation estimate.

Subpixel refinement, adaptive thresholding, and feature bucketing We found the accuracy improvements afforded by subpixel feature refinement outweighed its additional computational cost. While the lighting in the motion capture experiments did not substantially change, adaptive thresholding still yielded a lower failure rate. We would expect the accuracy difference to be greater when flying through more varied lighting conditions. Finally, without feature bucketing, the feature detector often detected clusters of closely spaced features, which in turn confused the matching process and resulted in both slower speeds and decreased accuracy.

Taking the best performing version of each of the above variations, our algorithm had a mean velocity error of 0.387 m/s and a 3.39% gross failure rate, and is unlikely to have been capable of autonomously flying the MAV through the entire recorded trajectory. In contrast, in environments with richer visual features, we have observed mean velocity errors of 0.08 m/s, with no gross failures, significantly lower than the values reported in table 1. Many of the gross failures are due to the blank wall on one side of the room — no state estimation process based on visual features can overcome this problem. To specifically address this problem and to ensure the safety of the vehicle, we will turn to planning algorithms presented in section 3.

2.2 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 (Henry et al., 2010). 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 (Henry et al., 2010).

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 MAV transmits RGB-D data to an offboard laptop, which detects loop closures, computes global pose corrections, and constructs a 3D log-likelihood occupancy grid map. For coarse navigation, we found that a grid map with $10cm$ resolution provided a useful balance between map size and precision. Depth data is downsampled to 128×96 prior to a voxel map update to increase the update speed, resulting in spacing between depth pixels of approximately $5cm$ at a range of $6m$. Incorporating a single frame into the voxel map currently takes approximately $1.5ms$.

As before, we adopt a keyframe approach to loop closure – new RGB-D frames are matched against a small set of keyframes to detect loop closures, using a fast image matching procedure (Henry et al., 2010). New keyframes are added when the accumulated motion since the previous keyframe exceeds either 10 degrees in rotation or 25 centimeters in translation. When a new keyframe is constructed, a

RANSAC procedure over FAST keypoints (Rosten and Drummond, 2006) compares the new keyframe to keyframes occurring more than 4 seconds prior. As loop closure requires matching non-sequential frames, we obtain putative keypoint matches using Calonder randomized tree descriptors (Calonder et al., 2008). A new keypoint is considered as a possible match to an earlier frame if the L_2 distance to the most similar descriptor in the earlier frame has a ratio less than 0.6 with the next most similar descriptor. RANSAC inlier matches establish a relative pose between the frames, which is accepted if there are at least 10 inliers. Matches with a reprojection error below a fixed threshold are considered inliers. The final refined relative pose between keyframes is obtained by solving a two-frame sparse bundle adjustment (SBA) system, which minimizes overall reprojection error.

To keep the loop closure detection near constant time as the map grows, we limit the keyframes against which the new keyframe is checked. First, we only use keyframes whose pose differs from the new frame (according to the existing estimates) by at most 90 degrees in rotation and 5 meters in translation. We also use Nistér's vocabulary tree approach (Nistér and Stewenius, 2006), which uses a quantized "bag of visual words" model to rapidly determine the 15 most likely loop closure candidates. Keyframes that pass these tests are matched against new frames, and matching is terminated after the first successful loop closure. On each successful loop closure, a new constraint is added to a pose graph, which is then optimized using TORO (Grisetti et al., 2007a). Pose graph optimization is typically fast, converging in roughly 30 ms. Corrected pose estimates are then transmitted back to the vehicle, along with any updated voxel maps.

Greater global map consistency can be achieved using a sparse bundle adjustment technique that optimizes over all matched features across all frames (Konolige, 2010). However, this is a much slower approach and not yet suitable for real-time operation.

3 Trajectory Planning

The visual odometry and SLAM processes in the previous sections described how to estimate the position of the vehicle and the environment around it, but did not describe how the vehicle should move to explore the environment around it. We assume that the vehicle is holonomic and that we have full control authority, allowing us treat the trajectory planning problem as a kinematic motion planning problem. Our UAV uses an onboard IMU and processor to auto-stabilize the helicopter's pitch and roll axes (Gurdan et al., 2007). As a result, the configuration space is $\mathcal{C} = \mathcal{R}^3 \times \mathcal{S}^1$: 3 dimensions for the UAV's position, and one for the UAV's yaw angle[1]. Exploring an unknown environment is often modelled as a problem of

[1] $\mathcal{C}$ denotes the configuration space (Lozano-Perez., 1983), the space of all vehicle poses. $\mathcal{C}_{free} \in \mathcal{C}$ is the set of all collision-free poses (based on a known map $\mathcal{M}$ of obstacles and the dimensions of the UAV) and $\mathcal{C}_{obst} \in \mathcal{C}$ is the set of poses resulting in collision with obstacles, so that $\mathcal{C} \equiv \mathcal{C}_{free} \cup \mathcal{C}_{obst}$.

coverage, where the objective is to visit all reachable states or "frontiers" that lie on the boundary of known free space (Yamauchi et al., 1998; Kollar and Roy, 2008). Therefore, given the current vehicle state $x_0 \in \mathcal{C}_{free}$ and the partial map of the environment, the planning problem is therefore to find a sequence of actions to move the vehicle from state $\mathbf{x}_0$ to a frontier state $\mathbf{x}_g \in \mathcal{C}_{free}$ without collisions.

The Probabilistic Roadmap (PRM) is a well-known algorithm (Kavraki et al., 1996) for planning in problems of more than two or three dimensions, in which a discrete graph is used to approximate the connectivity of $\mathcal{C}_{free}$. The PRM builds the graph by sampling a set of states randomly from $\mathcal{C}$ (adding the start state $\mathbf{x}_0$ and goal state $\mathbf{x}_g$), and then evaluating each sampled state for membership in $\mathcal{C}_{free}$. Samples that lie within $\mathcal{C}_{free}$ constitute the nodes of the PRM graph and edges are placed between nodes where a straight line path between nodes also lies entirely within $\mathcal{C}_{free}$. Given the PRM graph, a feasible, collision-free path can be found using a standard graph search algorithm from the start node to the goal node.

However, the PRM and its variants are not yet well-suited to the problem of a GPS-denied UAV, in that executing a plan requires a controller that can follow each straight-line edge joining two successive graph nodes in the planned path. If the UAV executing the plan does not have a good estimate of its state, it may not be able to determine when it has arrived at a graph node and is to start following a new edge. Even more seriously, UAV stability typically depends on accurate estimates of higher order variables such as velocity. Without environmental feedback, velocity estimates from an inertial measurement unit (IMU) can quickly drift, causing catastrophic control failures. We therefore need the motion planner to generate plans that ensure accurate state estimation along the planned path. By planning in the *belief space* (or space of distributions), the planner can distinguish between future state estimates where the covariance will be small (i.e., the vehicle has high confidence in its mean state estimate) and future state estimates where the covariance will be large (i.e., the mean state estimate is uncertain). To address the problem of planning in belief space, we use the Belief Roadmap (BRM) algorithm, first presented by Prentice and Roy (2007), and summarize the algorithm in the following section.

3.1 Belief Space Planning

When planning in belief space, a naive approach would be to treat the belief space as a high-dimensional configuration space with some dimensions given by the covariance of the belief, and then directly apply the probabilistic roadmap. Assuming the beliefs are provided by a variant of the Kalman filter, this approach would require sampling beliefs directly from the space of Gaussian distributions (μ, Σ) over the state, adding the initial belief b_0 to the set of graph nodes, placing edges between pairs of beliefs (b_i, b_j) for which a controller exists that can take the vehicle from belief b_i to b_j, and then carrying out graph search as before to find a path that leads to a belief with maximum probability at the goal. However, the direct application of the unmodified PRM to belief space has some obvious failures, which can be addressed by the following modifications to the PRM algorithm.

Firstly, in a Gaussian belief space, every belief has some (small) probability that the robot is at the goal state, hence a different objective function is required. In order to incorporate the full Gaussian distribution in our planning process, we continue to search for a shortest path trajectory, but add the additional constraint that the uncertainty of the belief must be small throughout the path, that is, the trace of the covariance of the helicopter's belief $tr(\Sigma) < \epsilon$ where ϵ is some defined safety parameter and Σ is the covariance of the UAV's state estimate[2].

To plan efficiently, the BRM uses the fact that each Gaussian belief b_t is a combination of some μ and some Σ, where the reachability of μ and Σ can be calculated separately. Under mild assumptions of unbiased motion and sensor models, the reachability of any μ is a function of the vehicle kinematics and the environmental structure, just as in the PRM. For any μ that is reachable from the μ_0 of the initial distribution, the corresponding reachable covariance can be predicted by propagating the initial covariance Σ_0 along the path using an iterative application of the motion and sensing models (see equations equations (7) and (8) in Appendix A). Therefore, to construct a graph of the reachable belief space, the planner first samples a set of mean poses $\{\mu_i\}$ from $\mathcal{C}_{free}$ using the standard pose sampling of the PRM algorithm, and places an edge e_{ij} between pairs (μ_i, μ_j) if the straight line between poses is collision-free. Forward search can then be used to search for a path through the graph, but each step of the search computes the posterior covariance at each node in addition to the cost-to-go.

Covariance propagation requires multiple EKF updates along each edge e_{ij}, and while this operation is a constant multiplier of the asymptotic search complexity, it can still dominate the overall search time. These EKF updates are not a one-time cost; the search process will find multiple paths to node i, each with a covariance must then be propagated outwards from i along edge e_{ij} to reach node j, incurring the computational cost of propagating along the edge (a series of EKF updates) for each covariance.

To reduce this computational cost, the BRM uses the property that the covariance of a Kalman filter-based state estimator can be factored as $\Sigma = BC^{-1}$, which allows the combined process and measurement update for an EKF gives B_t and C_t as a linear function of B_{t-1} and C_{t-1}. The linear forms of the process and measurement update do not depend on the specific factorization, so we can use a trivial initial factorization as $B_0 = \Sigma_0, C_0 = I$, such that

$$\Psi_t \triangleq \begin{bmatrix} B \\ C \end{bmatrix}_t = \begin{bmatrix} 0 & I \\ I & M \end{bmatrix}_t \begin{bmatrix} 0 & G^{-T} \\ G & RG^{-T} \end{bmatrix}_t \begin{bmatrix} B \\ C \end{bmatrix}_{t-1}, \tag{1}$$

where Ψ_t is defined to be the stacked block matrix $\left[\begin{smallmatrix} B \\ C \end{smallmatrix}\right]_t$ consisting of the covariance factors and $\zeta_t \triangleq \left[\begin{smallmatrix} W & X \\ Y & Z \end{smallmatrix}\right]_t$ is defined to be the one-step linear operator on the covari-

[2] Note that, depending on the problem statement, covariance terms such as velocity and orientation may or may not be included in the overall objective. A variety of alternatives to this objective function are discussed in the original BRM paper (Prentice and Roy, 2009)

Algorithm 1 The Belief Roadmap (BRM) algorithm.

```
Require: Start belief $(\mu_0, \Sigma_0)$, goal $\mu_{goal}$ and map $\mathcal{C}$
 1: Sample poses $\{\mathbf{x}_i\}$ from $\mathcal{C}_{free}$ to build belief graph node set $\{n_i\}$ such that $n_i = \{\mu = \mathbf{x}_i, \Sigma = \emptyset\}$
 2: Create edge set $\{e_{ij}\}$ between nodes $(n_i, n_j)$ if the straight-line path between $(n_i[\mu], n_j[\mu])$ is collision-free
 3: Build one-step transfer functions $\{\zeta_{ij}\} \quad \forall \quad e_{ij} \in \{e_{ij}\}$
 4: Augment each node $n_i$ with best path $p=\emptyset$ to $n_i$, such that $n_i=\{\mu, \Sigma, p\}$
 5: Create search queue with initial position and covariance $Q \leftarrow n_0 = \{\mu_0, \Sigma_0, \emptyset\}$
 6: while $Q$ is not empty do
 7:   Pop $n \leftarrow Q$
 8:   if $n = n_{goal}$ then
 9:     Continue
10:   end if
11:   for all $n'$ such that $\exists e_{n,n'}$ and not $n' \ni n[p]$ do
12:     Compute one-step update $\Psi' = \zeta_{n,n'} \cdot \Psi$, where $\Psi = \begin{bmatrix} n[\Sigma] \\ I \end{bmatrix}$
13:     $\Sigma' \leftarrow \Psi'_{11} \cdot {\Psi'_{21}}^{-1}$
14:     if $tr(\Sigma') < tr(n'[\Sigma])$ then
15:       $n' \leftarrow \{n'[\mu], \Sigma', n[p] \cup \{n'\}\}$
16:       Push $n' \rightarrow Q$
17:     end if
18:   end for
19: end while
20: return $n_{goal}[p]$
```

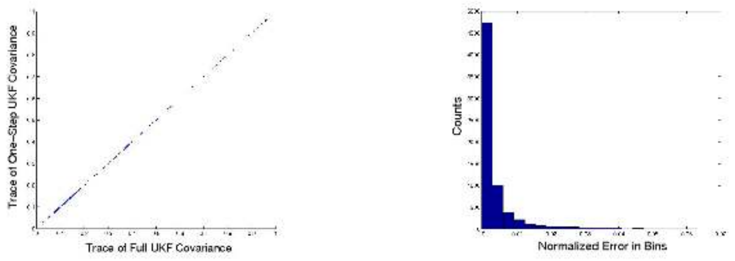
ance factors, equivalent to the process model and the measurement model, and we recover the posterior covariance from the posterior factors as $\Sigma_t = B_t C_t^{-1}$.

The EKF approximation assumes that the measurement function is locally linear, which is exactly the approximation that the Jacobian is locally constant. As a result, whenever the EKF assumptions hold, then we can assume that M_t is constant and known *a priori*. By multiplying Ψ_{t-1} by a series of matrices $\zeta_{t:T}$, we can compute the posterior covariance Σ_T from $T - t$ matrix multiplications and a single matrix inversion on C_T. This allows us to determine ζ_t for any point along a trajectory and the linearity of the update allows us to combine multiple ζ_t matrices into a single, one-step update for the covariance along the entire length of a trajectory.

Table 1 describes the complete Belief Roadmap algorithm, and Step 2 of the algorithm contains a pre-processing phase where each edge is labeled with the transfer function ζ_{ij} that allows each covariance to be propagated in a single step. By pre-computing the transfer function for each edge, the search complexity for belief space planning becomes comparable to configuration space planning.

3.1.1 Belief Space Planning using the Unscented Kalman Filter

The critical step of the BRM algorithm is the construction of the transfer function, which depends on terms R_t and M_t, the projections of the process and measurement noise terms into the state space. R_t and M_t represent the information lost due to



(a) Comparison of covariance predictions

(b) Distribution of errors using constant prior approximation

Fig. 4 (a) Comparison of trace of covariance from full UKF filtering and trace of covariance from one-step transfer function using UKF M_t matrix. (b) Distribution of ratio of error induced by computing the M_t matrix for the one-step transfer function using a constant prior.

motion, and the information gained due to measurements, respectively (again, see equations equations (7) and (8) in Appendix A). When using the Extended Kalman filter to perform state estimation, these terms are trivial to compute. However, the EKF is not always a feasible form of Bayesian filtering, especially when linearizing the control or measurement functions results in a poor approximation. One recent alternate to the EKF is the Unscented Kalman filter (UKF) (Julier et al., 1995), which uses a set of $2n + 1$ deterministic samples, known as "sigma points" from an assumed Gaussian density both to represent the probability density of a space of dimensionality n and to directly measure the relevant motion and measurement covariances. Appendix B provides a formal description of the UKF, and how to recover the information gain matrix M_t.

One concern is that change in information modelled by M_t is constant in the Kalman and extended Kalman filter models (assuming locally constant Jacobians), but for the UKF depends on the specific prior $\overline{\Sigma}_t$. Different choices of $\overline{\Sigma}_t$ for equation (26) may result in different one-step transfer functions. Nevertheless, the approximation error can be seen experimentally to be small. Figure 4(a) compares the covariances computed using the full UKF update with the covariances computed using the one-step transfer function for a range of motions and randomized initial conditions. The induced error is low; the traces of the posterior covariances computed with the one-step transfer function using the M_t matrix calculated in equation (26) closely match the posterior covariances from the full UKF model. Figure 4(b) shows a distribution of the ratio of the approximation errors to the magnitudes of the information gain, where 7000 trials were performed using 100 different priors and a range of initial conditions and trajectories were used to calculate the M_t matrix. The error induced in the one-step transfer function for using a constant M_t is less than 2% with a significance of $p = 0.955$, indicating low sensitivity to the choice of prior over a range of operating conditions.

3.2 Sampling in Belief Space

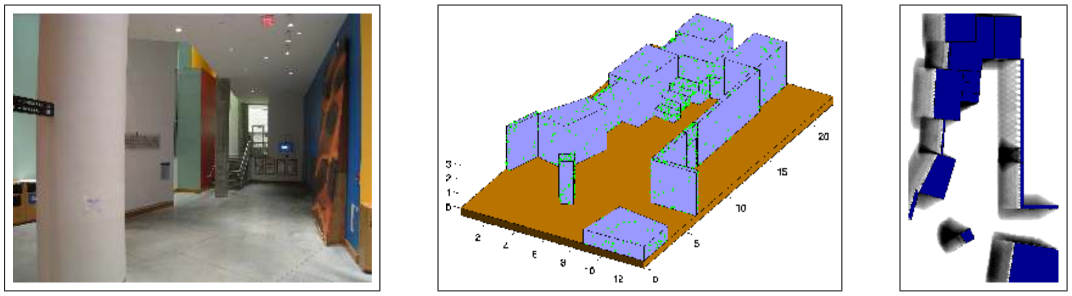
The original Belief Roadmap formulation presented by Prentice and Roy (2007, 2009) assumed some base sampling strategy for generating the graph through belief space. As the number of samples and the density of the graph grows, the BRM planning process will find increasingly low-covariance paths and is probabilistically complete. However, as the density of the graph grows, the cost of searching the graph will also grow; searching the graph will have time complexity $\mathcal{O}(b^d)$ for b edges per node and path of length d edges. We can reduce this complexity by minimizing the size of the graph using a sampling strategy that generates nodes that reflect the useful part of the belief space.

The optimal sampling strategy would generate samples that lie only on the optimal path to the goal; this would of course require knowing the optimal path beforehand. However, some samples are more likely to be useful than others: vehicle poses that generate measurements with high information value are much more likely to lie on the optimal path than vehicle poses that generate measurements with little information.

3.2.1 Sensor Uncertainty Sampling

If poses are initially sampled from $\mathcal{C}$ uniformly, but are accepted or rejected with probability proportional to the expected information gain from sensing at each point, the graph will still converge to one that maintains the connectivity of the free space. But, the distribution of the graph nodes will be biased towards locations that generate sensor measurements which maximize the localization accuracy of the vehicle. We call this sampling strategy *Sensor Uncertainty* (SU) sampling, after the “Sensor Uncertainty Field” defined by Takeda and Latombe (1992). The SU field is a mapping from location x to expected information gain, $x \to \mathcal{I}(x)$, where information gain is measured as the difference in entropy of the prior and posterior distributions, which in the Gaussian case is proportional to the lengths of the eigenvectors of the covariance. However, examining the information filter form of the measurement update in equation (11), we can see that the posterior covariance results from adding a fixed amount of information M_t; the covariance therefore increases in size by an amount proportional to M_t. We can efficiently approximate the SU field using the size of M_t such as $tr(M_t)$ (that is, the average of each eigenvector of M_t, Fedorov, 1972), rather than explicitly computing the posterior covariance and the resulting information gain. Finally, building the complete SU field is computationally expensive in practice; by sampling from this field in building the BRM graph, we gain the benefits of focusing the search on the states that lead to high information gain without the cost of explicitly building the SU field.

Figure 5(a) shows the ground floor of the MIT Stata Center with a 3D-view of this environment in figure 5(b). The environment has dimensions of $13m \times 23m$. The helicopter is equipped with a simulated RGB-D camera that is able to sense features, represented by the green crosses on the walls. We initially assume an unre-



(a) Photo of environment (b) 3D-environment (c) SU Field

Fig. 5 (a) MIT Stata Center, ground floor. (b) 3D-model of the unstructured, GPS-denied environment. The green dots are the known position of visual features to be used for localization. Each wall has a different density of visual features, though the features are assumed to be uniformly distributed on each wall. (c) Sensor Uncertainty Field for a fixed height and RGB-D camera orientation. The darker cells indicate locations expected to produce greater information gain.

alistically poor sensor model to highlight the variations of the different algorithms, specifically that the camera has a $2m$ range and a $30°$ field-of-view in the pitch and the yaw directions. A feature that falls within the helicopter's field-of-view will then generate a noisy measurement of the feature's distance and bearing relative to the helicopter's pose. The measurement noise is assumed to be Gaussian with $1m$ variance for the range measurement and $1rad$ in angular variance, and is independent of the distance between the feature and the helicopter[3]. By rotating the helicopter's yaw orientation, the planner can direct the camera in different directions, thereby enabling the helicopter to localize itself in the environment using the features in its field-of-view and a given map. This sensor model is unrealistic in terms of the maximum range and the constant noise model, but serves to illustrate how our planning approach achieves varying accuracy at the goal. In the subsequent sections, we show results for a more accurate sensor model in terms of reaching the goal.

To create the corresponding Sensor Uncertainty Field shown in figure 5(c), the trace of the information gain matrix, $tr(M)$, is evaluated at each location (x, y) in $\mathcal{C}_{free}$ for a fixed height, yaw, pitch and roll angle. Here, the camera is assumed to be pointing eastwards at a fixed height. Note that the full SU field is the same dimensionality as $\mathcal{C}$ and would require computing the information gain for every coordinate in $\mathcal{C}_{free}$. (The 2D slice of the sensor uncertainty field shown in figure 5(c) is given only to illustrate the concept.) We can, however, evaluate the information gain of a *specific* position efficiently, allowing us to draw samples randomly from $\mathcal{C}_{free}$ and accept them based on their information gains $tr(M)$. The intensities of the cells in the map in figure 5(c) correspond to the information gain, where darker map cells indicate locations that are expected to produce greater information gain.

[3] We also assume perfect data association, which for the purposes of experimental assessment allows us to control the noise of the sensor by varying terms in the measurement matrix. This is clearly an unrealistic assumption but experimentally did not affect the results.

For instance, the region in the center of the map has high information gain because of the high concentration of features along the walls in that region. The information gain increases with distance to each wall because the number of features in the field of view increases more than the growing covariance, until the distance to the wall is greater than the maximum range of the sensor. Locations where the associated sensor measurement is expected to detect more than one obstacle in the map also tend to have higher information gain compared to those that just encounter one obstacle. Remember that we do not need to simulate actual measurements; in order to compute $tr(M)$ we only need the measurement Jacobians.

Note that values of $tr(M)$ do not form a proper distribution, so we cannot accept or reject samples trivially according to $tr(M)$. Additionally, the range of values of $tr(M)$ will vary across environments, depending on how easy the given world is to localize in. Therefore, for a specific environment, and for k samples $\mathbf{x}_1, \ldots, \mathbf{x}_k$ with corresponding information gains $tr(M_1), \ldots, tr(M_k)$, we estimate a normal distribution over the information such that $tr(M_k) \sim N(\mu_k, \sigma_k)$, where (μ_k, σ_k) are the sample mean and covariances of $tr(M_{1:k})$. We then perform rejection sampling according to this distribution. For a new sample $\mathbf{x}_k$, we draw a rejection threshold P_k according to the latest sampled normal distribution, and we retain the sample if $tr(M_k) > P_k$, otherwise reject it. This provides us with an online method for estimating the distribution of information in the environment, and allows us to bias our accepted samples towards areas with greater expected information gain relative to the rest of the environment.

3.2.2 Unscented Kalman Filter Sampling

When using an state estimator that does not directly compute M_t, recall that we can recover M_t from the prior distribution $p(\mathbf{x})$ and the posterior distribution $p(\mathbf{x}|z)$, for example from the Unscented Kalman filter as in section 3.1. However, to recover M_t from the UKF prior and posterior requires us to invert the S_t matrix with complexity $\mathcal{O}(|Z|^3)$, where $|Z|$ is the number of measurements. In EKF-SU sampling, we were able to avoid this complexity because the M_t matrix could be computed directly from the measurement Jacobians. Given the number of measurements and the large number of samples that must be evaluated for information gain, inverting S_t may be computationally expensive. We therefore sample according to the traditional information gain $\mathcal{I}$,

$$\mathcal{I}(\mathbf{x}) = H(p(\mathbf{x})) - H(p(\mathbf{x}|z)), \tag{2}$$

where entropy is $H(p(\mathbf{x})) = -\int p(\mathbf{x}) \log p(\mathbf{x})$. Given that we have assumed that the belief of the helicopter's position is representable as a Gaussian distribution, $H(p(\mathbf{x}))$ is computationally cheaper to compute than M_t. In addition, since our analysis (Figure 4b) suggested that the measure of information gain was statistically insensitive to the choice of prior, we use a constant prior $p(\mathbf{x}) = \Sigma_0$ to evaluate sensor uncertainty, such that $H(p(\mathbf{x})) = P_0$. Furthermore, applying Bayes' rule, where $p(\mathbf{x}|z) = p(z|\mathbf{x}) \cdot p(\mathbf{x})$, we get

Algorithm 2 UKF Sensor Uncertainty Sampling Algorithm

```
Require: Map $\mathcal{C}$, Number of samples $N$, Constant prior $P_0$
 1: while size of graph < N do
 2:   Sample a pose, $\mathbf{x}_k$, from $\mathcal{C}$, with equal probability
 3:   if $\mathbf{x}_k \in \mathcal{C}_{free}$ then
 4:     Simulate expected sensor measurement, $z$, at $\mathbf{x}_k$
 5:     Generate sigma points, $\chi_i$, about $\mathbf{x}_k$ according to constant prior $P_0$, creat-
        ing prior distribution $p(\mathbf{x}_k)$
 6:     Calculate information gain $\mathcal{I}(\mathbf{x}_k) = P_0 - H(p(z|\mathbf{x}_k))$
 7:     Normalize $\mathcal{I}(\mathbf{x}_l)$ such that $\mathcal{I}(\mathbf{x}_k) \in [0,1]$
 8:     Update mean of $\mathcal{I}$, $\mu_k = \frac{1}{k}\sum_{m=1}^{k}\mathcal{I}(\mathbf{x}_m)$
 9:     Update cov of $\mathcal{I}$, $\sigma_k = \frac{1}{k-1}\sum_{m=1}^{k}(\mathcal{I}(\mathbf{x}_m) - \mu_k)^2$
10:     Sample threshold $P_k$ from normal distribution $\mathcal{N}(\mu_k, \sigma_k)$
11:     if $\mathcal{I}(\mathbf{x}_k) > P_k$ then
12:       Add $\mathbf{x}_k$ to graph with probability $\mathcal{I}(\mathbf{x}_k)$
13:     end if
14:   end if
15: end while
16: return graph
```

$$\mathcal{I}(\mathbf{x}) = P_0 - H(p(z|\mathbf{x})), \tag{3}$$

where $z = \operatorname{argmax}_z p(z|\mathbf{x})$. $p(z|\mathbf{x})$ is calculated according to the UKF algorithm by simulating the sensor measurement at the sample's location and finding the probability of the observing the sensor measurement at each of the sigma points. In general, the lower the probability of observation at the neighboring sigma points, the smaller the entropy of the posterior distribution, and therefore the greater the information gain. We normalize the information gain $\mathcal{I}(\mathbf{x})$ so that it lies in the range [0,1] by dividing by P_0. Similar to our approach for EKF-SU sampling, we then estimate a normal distribution over the information gain such that $\mathcal{I}(\mathbf{x}) \sim N(\mu_k, \sigma_k)$, where (μ_k, σ_k) are the sample mean and covariances of $\mathcal{I}(\mathbf{x})$. Finally, we choose a rejection threshold P_k according to this normal distribution, and accept the sample if $\mathcal{I}(\mathbf{x}) > P_k$. Algorithm 2 summarizes the UKF-SU sampling strategy.

Table 2 shows the computational benefit of rejection sampling according to the information gain $\mathcal{I}(x)$, rather than a measure on M_t. We evaluated the time taken to generate samples for a range of measurements using the two different rejection sampling functions. Regardless of the number of measurements, we saw an order of magnitude in time savings when calculating information gain, which can be significant for large graphs.

Figure 6(a) shows the samples drawn according to the sensor uncertainty. Observe that the sampling density is highest in the same region as the dark region in Figure 5(c) (center of map), and is lowest when far from any environmental struc-

	Number of measurements		
	90	300	500
$tr(M_t)$	0.731	2.84	7.23
$\mathcal{I}(\mathbf{x})$	0.0743	0.187	0.289

Table 2 Average time (in ms) to compute $tr(M_t)$ and $\mathcal{I}(x)$

ture, which consequently provides little or no localization information. For comparison, 6(b) shows the samples drawn according to a uniform sampling strategy.

In practice, the differences in sampling strategies can result in different paths and correspondingly different uncertainties. Figure 6(a) shows that the paths created by Sensor Uncertainty sampling tend to stay in regions with high information gain, since the samples were probabilistically chosen based on the amount of information gain each was expected to provide. The uniform sampling strategy also attempts to find a low-uncertainty path but the lack of samples in the regions with high information gain results in a path with higher uncertainty. It is worth noting that in the figures, the SU sampler appears to put samples close to obstacles, and a sampler that simply samples close to obstacle boundaries may do well. We will see in the next section that in fact the SU sampling strategy leads to better performance than sampling strategies that use the obstacle boundaries. In fact, sampling only near obstacles both leads to poor performance for camera models — being too close to obstacles can lead to reduced information content due to the reduced number of features in the field of view.

3.2.3 Alternate Sampling Strategies

In order to evaluate the effectiveness of the SU sampling strategy, we compared it with other sampling strategies that have gained popularity in the literature. Although these algorithms have been proposed to improve the performance of the PRM algorithm, they can also be used to test the performance of the SU strategy in the BRM context. In this section, we first describe three alternative sampling strategies (Uniform, Gaussian, Bridge), before reporting the results of the BRM path-planning when using each of these strategies.

Uniform Sampling: Uniform sampling is the most basic sampling strategy used by the majority of the sampling-based techniques. This algorithm does not use any known information about the world, but merely samples $\mathcal{C}$ uniformly, and adds samples to the graph that are in $\mathcal{C}_{free}$. By employing a simple collision-check function, the uniform sampling strategy is a very efficient means of obtaining the general connectivity of a given map. Figure 6(b) shows an example of the samples generated using this sampling method.

Gaussian Sampling: A significant limitation of the uniform sampling strategy is that it often fails to represent important regions in the $\mathcal{C}_{free}$, for instance, difficult regions such as narrow corridors and areas around obstacles may not be sampled

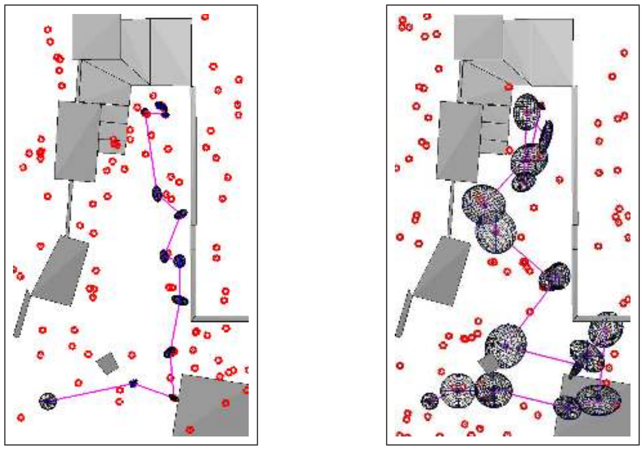
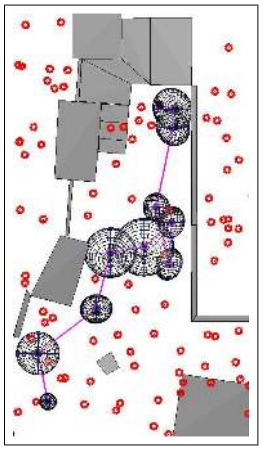

(a) SU Sampling (b) Uniform Sampling

Fig. 6 (a) Distribution of 100 samples (shown in red) drawn using Sensor Uncertainty sampling. (b) Distribution of 100 uniformly drawn samples. In both figures, the dark circles are the $1-\sigma$ ellipses of the covariance. Smaller circles are higher-certainty positions. Note that this is a bird's-eye view and the helicopter can fly over some obstacles. Also note that each sample is a point in $\mathcal{R}^3 \times \mathcal{S}^1 \times \mathcal{S}^1$; the SU samples have a high bias towards sensor orientations towards the environmental features. In both figures, the paths are found using the BRM, but because the uniform sampling strategy has many more samples with orientations that do not point towards the environmental features, the overall uncertainty is much higher.

unless a large number of samples are used, incurring a large computation cost. Boor et al. (1999) present the Gaussian sampling strategy that attempts to give better coverage of the difficult parts of the free configuration space, especially those areas that are close to obstacles. Gaussian sampling biases samples towards obstacles in the environment, which, in the context of the BRM, would seem to be a reasonable approximation for areas with higher information gain. The algorithm first uniformly samples the $\mathcal{C}$ space to obtain a sample, $\mathbf{x}_k^1$, regardless of whether it is in $\mathcal{C}_{free}$ or $\mathcal{C}_{obs}$. A distance value, d, and direction θ are then chosen according to normal distributions, and a second sample, $\mathbf{x}_k^2$, is generated at a location d away from $\mathbf{x}_k^1$ in the direction of θ. The two samples are then tested to determine if they belong to the subspaces $\mathcal{C}_{free}$ or $\mathcal{C}_{obs}$; if the samples are in different subspaces of $\mathcal{C}$, the sample that is in $\mathcal{C}_{free}$ is then added to the graph. For the purposes of our evaluation, using the general intuition that the samples should be within viewing range of the obstacles and features in the environments, we set σ, the standard deviation of the distribution on d, to be the maximum range of the sensor used for localization. Figure 7(a) shows an example set of samples generated by the Gaussian sampling strategy.

Bridge Sampling: A third algorithm addresses a specific problem encountered by many sampling strategies of not being able to identify narrow passages through a given environment. Being able to find narrow passages through $\mathcal{C}_{free}$ is often critical to finding good paths in many motion planning problems. However, narrow


(a) 100 Samples generated from Gaussian Sampling

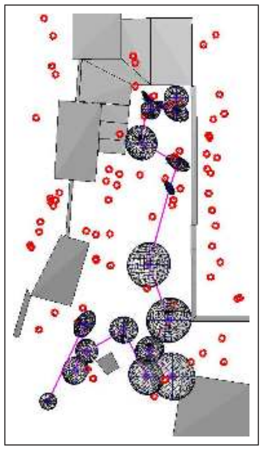

(b) 100 Samples generated from Bridge Sampling

Fig. 7 (a) Distribution of 100 samples drawn using Gaussian sampling. (b) Distribution of 100 samples drawn using bridge sampling.

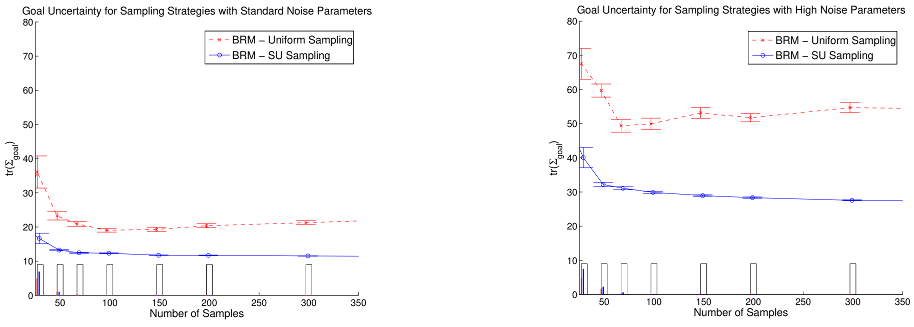
passages are also the hardest for a randomized sampler to find and add to the graph, requiring strategies that are biased towards finding paths in narrow passages.

To address this problem, the bridge test for sampling narrow passages was developed by Hsu et al. (2003). The key idea is to only add a sample to the graph when it is found to be between two obstacles. Two samples, $\mathbf{x}_k^1$ and $\mathbf{x}_k^2$, are first sampled from the map environment, with $\mathbf{x}_k^2$ being drawn from a normal distribution with mean $\mathbf{x}_k^1$ and a given variance σ. If both samples are found to be in $\mathcal{C}_{obs}$, the midpoint between the two samples is then generated and tested for collisions. This midpoint is added to the graph if it is in $\mathcal{C}_{free}$. For reasons similar to the Gaussian sampling strategy, σ was set at twice the maximum sensor range. Figure 7(b) shows an example set of samples generated using the bridge strategy.

3.3 *Comparison of Sampling Strategies: Simulated Camera-Equipped Helicopter*

We tested the effectiveness of the SU sampling strategy against the alternative sampling algorithms above by running experiments on the system described in section 3.2.1, a helicopter navigating in a simulated environment of the MIT Stata Center ground floor, as shown in 5(b).

We first observe that the SU strategy is particularly useful when there is variability in terms of the information available to the sensors throughout the environment. As discussed previously, our initial simulated experiments are performed with an unrealistically poor RGB-D camera model where the sensor's capability is particularly limited, such that finding paths that maximize information gain throughout the path then becomes even more critical. Figure 8 compares the performance of the SU and uniform sampling strategies under different noise and sensor limitation conditions. When the control and measurement noise is doubled and the maximum sensor range is reduced (Figure 8(b)), the resultant uncertainty for both sampling strategies increases. However, the emphasis on finding samples with high information gain under the SU sampling strategy reduces the effect of the noisier conditions, resulting in a greater absolute difference in uncertainty between both sampling strategies.



(a) Low Noise (b) High Noise

Fig. 8 Performance of SU and uniform sampling strategies under different noise conditions. High noise scenario has double the control and measurement noise relative to the low noise model, as well as a 25% reduction in maximum sensor range. The bar plots under each graph show the percentage of feasible paths that each algorithm was able to find.

Next, to compare amongst the different sampling strategies and illustrate the performance of the BRM strategy, we randomly selected 5 start and goal positions in the map where the straight-line distance between both points was at least of a minimum length of $8.53m$ and an average length of $13.66m$. For each start-goal combination, we sampled the environment using each of the 4 sampling strategies and a range of sample set sizes. After creating a graph of nodes from these samples,

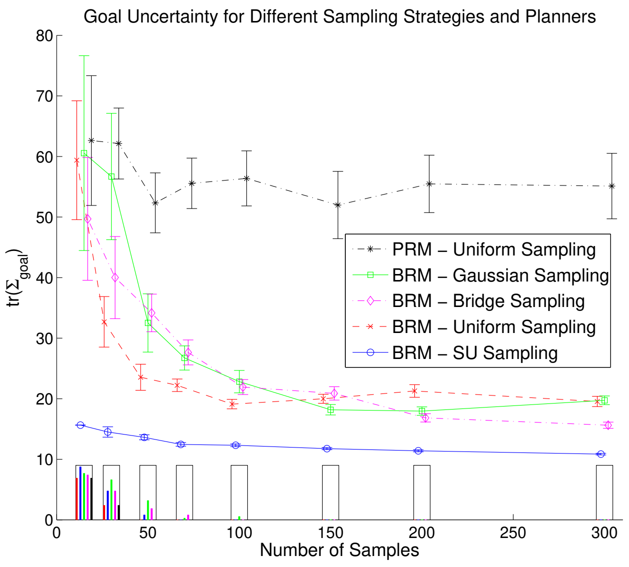


Fig. 9 Comparison amongst the different sampling strategies and planning methods. All trials presented in this graph used the same start and goal to perform the search. Each data point represents 30 trials, and the error bars represent the standard error of the computed matrix trace. The bar graphs along the bottom of the figure show, for each sampling-planning strategy and number of samples, the percentage of the 30 trials that failed to find a path that satisfied the constraints on the covariance (our true objective function). The line graph also plots the trace of the helicopter's expected covariance at the goal when we use different sampling strategies, sample sizes, and planning methods. Lower covariances at the goal also typically correspond to more accurate performance, and is often used as an alternate objective function for motion planning in belief space.

the BRM and PRM planning strategies were executed and the performance of the resulting plans compared. For a given start-goal combination, sampling strategy and sample set size, the experiment was repeated 30 times.

Figure 9 shows the advantage of planning with the BRM, and sampling using the Sensor Uncertainty sampling strategy. This figure reports the performance of each of the sampling strategies and planning methods, using a fixed start and goal for all trials, over a range of sample set sizes, where performance is measured by the percentage of trials that failed to find a feasible path (bar graphs), as well as the average trace of the helicopter's expected covariance at the goal after executing the planned path (line graphs).

Table 3 shows a comparison of the sampling strategies across various initial start and goal positions using 100 samples. An infeasible path was defined as one where the covariance of the state estimate was greater than a threshold. The results not only suggest that the BRM substantially outperforms the PRM in producing paths with lower uncertainty, but also that the SU sampling strategy allows us to achieve better paths with fewer samples. Regardless of the initial conditions, the SU sampling strategy consistently outperformed the other sampling strategies, in terms of both the percentage of paths found and the expected uncertainty at the goal. These results emphasize that SU sampling is also not equivalent to simply sampling close to obstacle boundaries.

		Path 1	Path 2	Path 3	Path 4	Path 5
Uniform	% success	100	96.6	100	100	100
	Final cov	17.87	22.60	2.22	19.11	1.48
SU	% success	**100**	**96.6**	**100**	**100**	**100**
	Final cov	**12.38**	**11.36**	**1.99**	**12.39**	**1.39**
Gaussian	% success	96.6	96.6	100	93.1	89.7
	Final cov	23.89	17.89	17.2	22.16	1.41
Bridge	% success	100	3.5	17.2	100	13.8
	Final cov	21.58	13.48	2.33	21.32	1.36

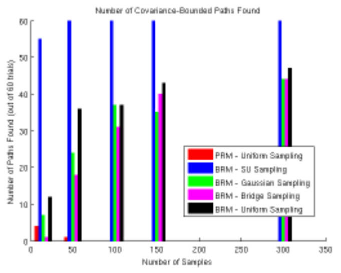
Table 3 Performance of different sampling strategies across different paths, using 100 samples.

Table 4 shows a comparison of the performance and time costs of different combinations of sampling and planning strategies. The conventional PRM method is unsurprisingly the fastest algorithm, but suffers from poor localization performance. The BRM suffers from additional time complexity when building the graph; in particular, the BRM with SU sampling incurs the largest time penalty in building the graph because of the need to calculate the information gain matrix of every potential sample. However, the graph construction is a one-time operation and can be amortized across multiple queries.

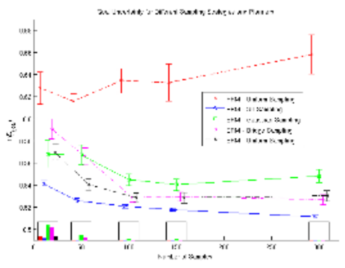
	Trace Cov. at Goal	Graph Build Time (s)	Path Search Time (s)
PRM, Uniform Sampling	56.38	0.79	0.15
BRM, Uniform Sampling	19.11	110.75	0.38
BRM, SU Sampling	**12.31**	323.12	0.55
BRM, Gaussian Sampling	22.82	88.14	0.21
BRM, Bridge Sampling	21.92	178.58	0.30

Table 4 Performance and time costs of different planners.

Lastly, we replaced the sensor model with a more realistic RGB-D model, and a more accurate map of a real environment (shown in figure 14(a)). We modeled the RGB-D sensor model as a Microsoft Kinect with a $4m$ max range and $57°$ field of view in the yaw and $43°$ in pitch directions and a Gaussian noise model that is a function of depth, such that $\sigma = 1.5 \times 10^{-5} \times d$ (Khoshelham, 2011). Note that for position estimation, we saw experimentally that the noise model of the individual features had little effect — the dominant effect was the number of available features. Figure 10 shows the performance of the different algorithms using the RGB-D camera model. Figure 10(a) is the relevant measure for our application, the ability to find paths that satisfy the constraint on the covariance and we see that even with very few samples, the BRM is able to find a feasible path. For comparison to figure 9, we also provide the covariance at the the goal. We see that the BRM algorithms are consistently able to find lower covariance trajectories (the absolute values of the covariances between figures 9 and 10(b) are not comparable because the sensor model and state space are different, and so different overall uncertainties are feasible.)



(a) Paths Found



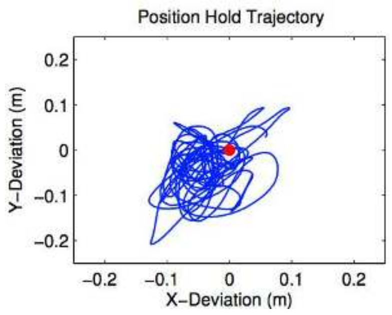
(b) Goal Uncertainty

Fig. 10 Performance of the Kinect camera model in the environment shown in figure 14. (a) The number of 60 trials that found a feasible path, as a function of the number of samples. The BRM using the SU sampler found a feasible path 100% of the time except when constrained to using 10 samples. (b) For comparison to figure 9, we also provide a comparison of the trace of the helicopter's expected covariance at the goal (line graph) and the percentage of feasible paths that each algorithm was able to find (bar graph along the bottom).

4 Indoor Navigation Results

In addition to evaluating the visual odometry algorithms reported in section 2.1, we conducted a number of autonomous flight experiments in the motion capture system and in larger environments. In these experiments, the vehicle flew autonomously with state estimates provided by the algorithms presented in this paper.

Figure 11 shows an example where the MAV was commanded to hover at a target point using the RGB-D camera, along with statistics about how well it achieved this goal. The ground truth trajectory and performance measures were recorded with the motion capture system.



Metric	
Duration	90 s
Mean speed	0.10 m/s
Mean pos. deviation	6.2 cm
Max pos. deviation	19 cm

Fig. 11 A plot showing the ground truth trajectory of the vehicle during position hold. The red dot near the center is the origin around which the vehicle was hovering. The vehicle was controlled using visual odometry, and its position measured with a motion capture system.

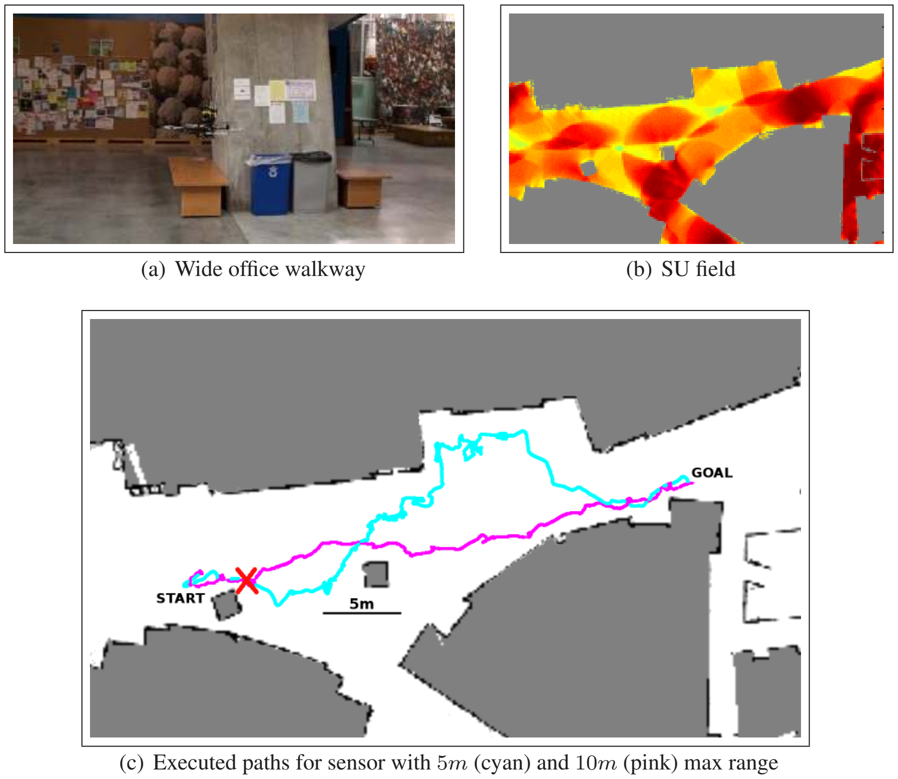
4.1 Laser-based Validation of Belief Space Navigation

We performed a number of experiments onboard an actual quadrotor helicopter to demonstrate the properties of our navigation in belief space. The vehicle was commanded through the environment by a human operator selecting destination waypoints using a graphical interface. The purpose of these experiments were to characterize the ability of the MAV to maintain a desired position and to follow a planned trajectory. We initially validated our results by building on our previous work (Bachrach et al., 2009b) that used a Hokuyo UTM-30LX laser rangefinder for navigation and localization. The UTM-30LX is a planar laser rangefinder that provides a 240° field-of-view at 40 Hz, up to an effective range of $30m$. The laser is mounted in the X-Y plane of the helicopter and we modified the laser to optically redirect 20° of its field-of-view to provide a small set of range measurements in the (downward) z direction. In a single scan, the vehicle is therefore able to estimate its position, yaw orientation and altitude with respect to environmental features. We have shown previously that the measurement of the ground plane is relatively noisy, although sufficient for altitude control.

We performed navigation experiments on two world environments, on the first floor of MIT's Stata center, which is a wide indoor walkway environment (Figure 12(a)), and on the first floor of MIT's Walker Memorial building, an old gymnasium open space that is often used for banquets and functions (Figure 13(a)). For these two environments, we focused on demonstrating the BRM algorithm's ability to generate paths that will allow the helicopter to remain well-localized. We did not compare the BRM's performance to the PRM algorithm to avoid potential loss of control and crashes resulting from inaccurate state estimation. Instead, we artificially limited the range of the laser rangefinder for both planning and state estimation; we demonstrate the effect of different sensor range limits on the planned paths and the need for incorporating sensor characteristics when planning, before moving to mapping using the RGB-D camera.

For each of these environments, we first generated a 2D map of the environment using SLAM technology that has been reported previously (Bachrach et al., 2009b). While it may appear that localization using a 2D map is difficult when the helicopter pitches and rolls, we also reported previously that 2D localization is relatively robust to changes in pitch and roll (Bachrach et al., 2009b). Figures 12(b) and 13(b) show the 2D map of both environments, as well as the SU field indicating areas of sensor uncertainty, computed according to equation (3). However, note that the SU field is never actually constructed but SU samples are generated via rejection sampling.

For each environment, two different paths were generated, each corresponding to a different maximum range for the laser rangefinder. Different maximum ranges affect the helicopter's ability to localize itself, thus affecting the paths generated by the BRM algorithm. Figures 12(c) and 13(c) show the helicopter's trajectories based on the paths generated by the BRM algorithm for the different sensor configurations. For the experiments along the office walkway, the cyan path denotes the trajectory when the sensor range was set to $5m$, while the pink path denotes the trajectory for the $10m$ configuration. For the open indoor environment, the cyan path represents



(a) Wide office walkway

(b) SU field

(c) Executed paths for sensor with $5m$ (cyan) and $10m$ (pink) max range

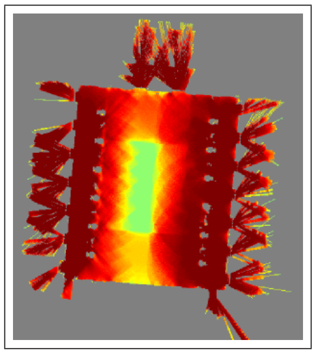
Fig. 12 Helicopter experiments in an office environment (a) View of environment (b) SU field of the environment. The lighter regions indicate areas of higher sensor uncertainty. Grey regions indicate obstacles in the environment (c) BRM paths executed when the laser range was set to $5m$ (cyan) and $10m$ (pink). The helicopter was able to successfully navigate both planned paths, traveling $44.05m$ and $36.28m$ respectively. The red cross denotes where the state estimation would have failed if the 10m path were attempted using the 5m sensor.

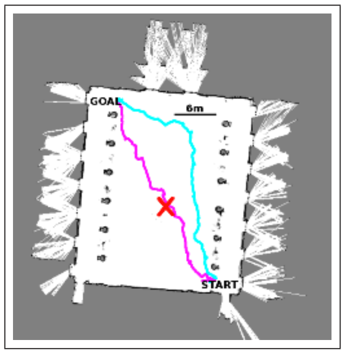
the $8m$ configuration, while the pink path represents the trajectory when the sensor range was $30m$. Due to the absence of a motion capture system, all paths were recorded based on the helicopter's state estimate from its localization module, and the helicopter's ability to reach the goal location was verified using the human eye.

In all of these scenarios, the helicopter successfully executed the paths generated by the BRM algorithm, and the actual and desired paths matched closely regardless of the range limits. In addition, the path generated for the laser with a shorter maximum range was always longer than that of the laser with the longer maximum range. In general, when the sensor is of higher quality, more of the environment is well-localizable, and hence the planned path more closely approximates the shortest path trajectory. In contrast, a low-quality sensor requires more careful planning to ensure that the vehicle remains well-localized, often resulting in a longer path.

We examined how the helicopter would perform if the BRM had assumed a better sensor than actually available, which allowed us to assess the effect of the sensor model on the planning process. To avoid potential crashes, we did not perform this

(a) Open, indoor environment


(b) SU field



(c) Executed paths for sensor with $8m$ (cyan) and $30m$ (pink) max range

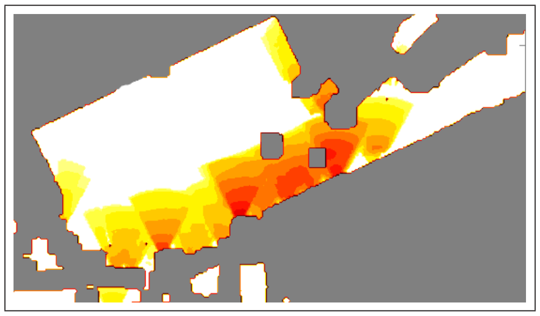
Fig. 13 Helicopter experiments in a large open indoor environment (a) View of environment (b) SU field of the environment. The lighter regions indicate areas of higher sensor uncertainty. Grey regions indicate obstacles in the environment (c) BRM paths executed when the laser range was set to $8m$ (cyan) and $30m$ (pink). The helicopter was able to successfully navigate both planned paths, traveling $36.58m$ and $32.21m$ respectively. The red cross denotes where the state estimation would have failed if the 10m path were attempted using the 5m sensor.

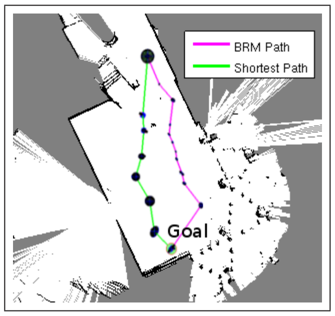
analysis on actual hardware, but instead modified the raw laser data from the earlier experiments. Specifically, we post-processed the raw laser data from the experiments shown in figures 12 and 13, truncating the laser data to have a shorter maximum range than was actually available or was known to the BRM during planning. We then re-estimated the vehicle's state along the trajectory using the modified laser data, and evaluated whether the helicopter was able to remain well-localized. In both cases, the vehicle followed a trajectory that did not contain enough information for the vehicle to stay well localized, since the truncation to a shorter maximum range removed a considerable amount of information from the sensor signal. Additionally, in both cases, the state estimate became sufficiently uncertain that the vehicle control would likely have become unstable. The crosses on both Figures 12(c) and 13(c) indicate the point at which the helicopter was no longer able to localize itself, determined when $tr(\Sigma)$ was greater than 1. Given the helicopter's strict requirements for localizability, where it is essential to be well-localized at every step, the crosses effectively mark the points where the helicopter would have crashed given the planned path and the modified sensor characteristics. It is therefore critical that sensor limitations are accurately incorporated when planning under uncertainty.

4.2 Belief Space Navigation using the RGB-D Camera

We also demonstrated the use of the BRM algorithm for navigation on the helicopter. Figure 14(a) shows an example environment of an open space, where the center of the environment is out of range of the RGB-D camera. Additionally, the left side of the environment (in the picture) is essentially featureless. In figure 14(b), we see that the sensor uncertainty field reflects the absence of information along this wall.

(a) Open, indoor environment


(b) SU field



(c) Paths for RGB-D sensor with $4m$ (pink) and $30m$ (green) max range

Fig. 14 Helicopter experiments in a large open indoor environment (a) View of environment (b) SU field of the environment (slice at $0°$ yaw). The lighter regions indicate areas of higher sensor uncertainty. Grey regions indicate obstacles in the environment (c) BRM paths using the RGB-D model (max range $4m$, pink) and laser (max range $30m$, green).

Figure 14(c) shows the paths generated by the shortest path planner (green) and the BRM planner using the RGB-D sensor model (pink), with the corresponding covariances of the state estimator drawn on top of each trajectory. As expected, we see that the covariances of the state estimate grow along the shortest path, but stay tightly bounded along the BRM trajectory.

4.3 Mapping using the RGB-D Camera

Finally, we experimentally validated our mapping and motion planning algorithms using the RGB-D camera. We have flown in a number of locations around the MIT campus, and at the Intel Research office in Seattle. Two such experiments are shown in figure 15. As the MAV covered greater distances, the RGB-D mapping algorithm limited the global drift on its position estimates by detecting loop closures and correcting the trajectory estimates. The trajectory history was then be combined with the RGB-D sensor data to automatically generate maps that are useful both for a human operator's situational awareness, and for autonomous path planning and decision making. While ground truth position estimates are not available, the quality of the state estimates computed by our system is evident in the rendered point cloud.

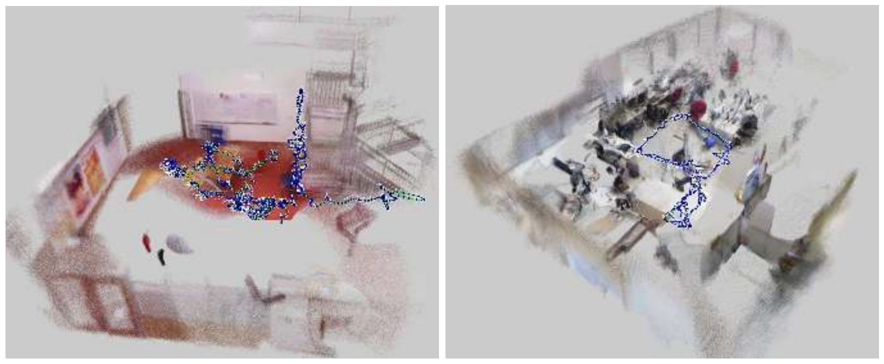

(a) (b)

Fig. 15 Trajectories flown by the MAV in two navigation experiments.

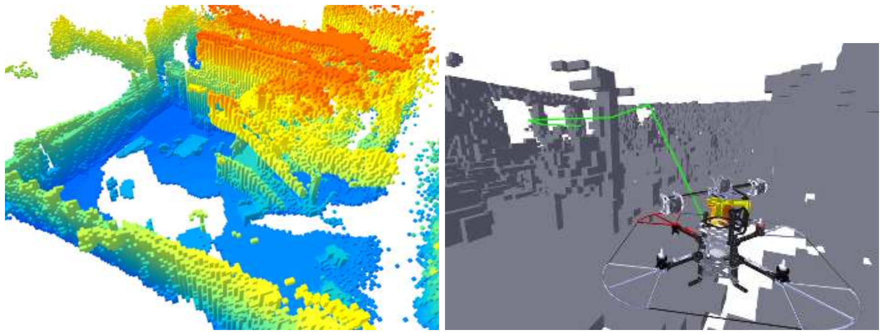

(a) (b)

Fig. 16 Voxel maps for the environments in Fig. 15. (a) Dense maximum-likelihood occupancy voxel map of the environment depicted in Fig. 15a, false-colored by height. Unknown/unobserved cells are also tracked, but not depicted here. (b) A voxel map of the environment in Fig. 15b allows the vehicle to plan a collision-free 3D trajectory (green).

Figure 16a shows an occupancy voxel map populated using the dense depth data provided by the RGB-D sensor. These occupancy maps can be used for autonomous path planning and navigation in highly cluttered environments, enabling flight through tight passageways and in close proximity to obstacles. Figure 16b shows a rendering of the MAV's internal state estimate as it flew through the environment depicted in Figure 15b. While these renderings are not necessary for obstacle avoidance, they would serve to provide a human operator with greater situational awareness of the MAV's surrounding environment.

5 Related Work

Visual odometry Visual odometry refers to the process of estimating a vehicle's 3D motion from visual imagery alone, and dates back to Moravec's work on the Stanford cart (Moravec, 1980). The basic algorithm used by Moravec and others since then is to identify features of interest in each camera frame, estimate depth to each feature (typically using stereo), match features across time frames, and then estimate the rigid body transformation that best aligns the features over time. Since then, a great deal of progress has been made in all aspects of visual odometry. Common feature detectors in modern real-time algorithms include Harris corners (Harris and Stephens, 1988) and FAST features (Rosten and Drummond, 2006), which are relatively quick to compute and resilient against small viewpoint changes. Methods for robustly matching features across frames include RANSAC-based methods (Nistér et al., 2004; Johnson et al., 2008; Konolige et al., 2007) and graph-based consistency algorithms (Howard, 2008). In the motion estimation process, techniques have ranged from directly minimizing Euclidean distance between matched features (Horn, 1987), to minimizing pixel reprojection error instead of 3D distance (Nistér et al., 2004). When computation constraints permit, bundle adjustment has been shown to help reduce integrated drift (Konolige et al., 2007).

Visual odometry estimates local motion and generally has unbounded global drift. To bound estimation error, it can be integrated with simultaneous localization and mapping (SLAM) algorithms, which employ loop closing techniques to detect when a vehicle revisits a previous location. Most recent visual SLAM methods rely on fast image matching techniques (Snavely et al., 2006; Newman et al., 2009) for loop closure. As loops are detected, a common approach is to construct a pose graph representing the spatial relationships between positions of the robot during its trajectory and environmental features, creating constraints that link previous poses. Optimization of this pose graph results in a globally aligned set of frames (Grisetti et al., 2007b; Olson et al., 2006; Kaess et al., 2008). For increased visual consistency, Sparse Bundle Adjustment (SBA, Triggs et al., 2000) can be used to simultaneously optimize the poses and the locations of observed features.

MAVs and Visual Navigation The primary focus in the visual odometry communities has been on ground vehicles, however, there has been significant amount of research on using visual state estimation for the control of MAVs. For larger outdoor helicopters, several researchers have demonstrated various levels of autonomy using vision based state estimates (Kelly and Sukhatme, 2007; Buskey et al., 2004). While many of the challenges for such vehicles are similar to smaller indoor MAVs, the payload and flight environments are quite different. For smaller MAVs operating in indoor environments, a number of researchers have used monocular camera sensors to control MAVs (Steder et al., 2008; Ahrens et al., 2009; Blösch et al., 2010; Celik et al., 2008). Ko et al. (2007) use the iMote2 technology and the UKF for state estimation in aerial vehicles, and Valenti et al. (2006) were the first to demonstrate reliable navigation and position estimation on quadrotor helicopters. However, these algorithms require specific assumptions about the environment (such as known patterns) to obtain the unknown scale factor inherent in using a monocular camera. Previous work in our group used a stereo camera to stabilize a MAV in unknown indoor environments (Achtelik et al., 2009), but the computation had to be performed offboard, and no higher level mapping or SLAM was performed.

Finally, there has been considerable work in using laser range finders for MAV navigation and control (He et al., 2008b; Bachrach et al., 2009a; Grzonka et al., 2009; Shen et al., 2011) with the limitations discussed earlier in this paper. Laser range finding on-board helicopters is also not a novel technology (Thrun et al., 2003; Mejias et al., 2006), and more recently, a number of quadrotor configurations have been developed (Angeletti et al., 2008; Grzonka et al., 2009) that are similar to the design we first proposed by He et al. (2008a).

Visual Mapping Our objective is not only alignment and registration, but also building 3D models with both shape and appearance information. In the vision and graphics communities, a large body of work exists on alignment and registration of images for 3D modeling and dense scene reconstruction (e.g., Pollefeys et al., 2008). However, our focus is on primarily on scene modeling for robot perception and planning, and secondarily for human situational awareness (e.g., for a human supervisor commanding the MAV). Strobl et al. (2009) combine a ToF camera with a stereo camera to build 3D object models in real-time. Kim et al. (2009) used a set of time-of-flight cameras in a fixed calibrated configuration and with no temporal alignment of sensor streams. Se and Jasiobedzki (2008) use a stereo camera combined with SIFT features to create 3D models of environments, but make no provision for loop closure or global consistency. Newcombe and Davison (2010) develop an impressive system that does real-time dense 3D reconstruction with a monocular camera, although their system is still limited to small feature-rich scenes.

There has also been a large amount of work on dense reconstruction from videos (e.g., Pollefeys et al., 2008) and photos (e.g., Debevec et al., 1996; Furukawa and Ponce, 2009), mostly on objects or outdoor scenes. One interesting line of work (Furukawa et al., 2009) attacks the arguably harder problem of indoor reconstruction, using a Manhattan-world assumption to fit simple geometric models for visualization purposes. Such approaches are computationally demanding and not very robust in feature-sparse environments.

Motion Planning under Uncertainty Modern approaches to planning with incomplete state information are typically based on the partially observable Markov decision process (POMDP) model or as a graph search through belief space (Bonet and Geffner, 2000). While the POMDP provides a general framework for belief space planning, the complexity of the solution grows exponentially in the length of the policy and the number of potential observations. Approximation algorithms exist to mitigate the problem of scalability (Pineau et al., 2003; Smith and Simmons, 2004), but these techniques still face computational issues in addressing large problems. Other papers have incorporated sensor characteristics for planning (Taïx et al., 2008), though the algorithm assumes that a non-collision path already exists, and focuses on determining the best landmarks to associate to each part of the path. den Berg et al. (2010) propose using a distribution over state estimates with a conventional RRT to generate motion plans, although this approach is not complete and can fail to find feasible plans. Bry and Roy (2011) proposed the Rapidly-exploring Random Belief Tree to track a distribution over state estimates along with the conventional Kalman filter covariance using an incremental sampling technique to refine trajectories, and is strongly related to the BRM algorithm.

The extended Kalman filter and unscented Kalman filter have been used extensively, especially for state estimation. The symplectic form (and related Hamiltonian form) of the covariance update has been reported before, most recently by Mourikis and Roumeliotis (2006). Planning algorithms have also incorporated these filters to generate paths that are robust to sensor uncertainty (Gonzalez and Stentz, 2007; Brooks et al., 2006). However, without the efficient covariance update presented in this paper, the deterministic search performed by these planning algorithms is computationally expensive.

6 Conclusion

This paper presented an experimental analysis of our approach to enabling autonomous flight using an RGB-D sensor. Our system combines visual odometry techniques from the existing literature with our previous work on autonomous flight and mapping, and is able to conduct all sensing and computation required for local position control onboard the vehicle. Using the RGB-D sensor, our system is able to plan complex 3D paths in cluttered environments while retaining a high degree of situational awareness. Additionally, we showed how the Belief Roadmap algorithm Prentice and Roy (2007, 2009) can be used to plan trajectories that incorporate a predictive model of sensing, allowing the planner to minimize the positional error of the helicopter at the goal using efficient graph search. The original BRM algorithm assumed an Extended Kalman filter model for position estimation, and we showed how this algorithm can be extended to use the Unscented Kalman filter and provided a new sampling strategy for UKF position estimation. We concluded with an experimental validation of our overall system for both laser- and RGB-D based navigation and mapping.

7 Acknowledgements

This research was supported by the Office of Naval Research under MURI N00014-07-1-0749, Science of Autonomy program N00014-09-1-0641 and the Army Research Office under the MAST CTA. Peter Henry and Dieter Fox were supported by ONR MURI grant number N00014-09-1-1052, and by the NSF under contract number IIS-0812671, as well as collaborative participation in the Robotics Consortium sponsored by the U.S Army Research Laboratory under Agreement W911NF-10-2-0016. Ruijie He was supported by the Republic of Singapore Armed Forces. Nicholas Roy was supported by the National Science Foundation Division of Information and Intelligent Systems under grant # 0546467.

Appendix

A. The Extended Kalman Filter

For reference, we provide a description of the extended Kalman filter equations. Bayesian filtering is one of the most robust methods of localization (Leonard and Durrant-Whyte, 1991), in which a probability distribution $p(\mathbf{x}_t|u_{1:t}, z_{1:t})$ is inferred over the (unknown) vehicle state $\mathbf{x}_t$ at time t following a series of noisy actions $u_{1:t}$ and measurements $z_{1:t}$. With some standard assumptions about the actions and observations, the posterior distribution (or belief) can be expressed as

$$p(\mathbf{x}_t|u_{1:t}, z_{1:t}) = \frac{1}{Z} p(z_t|\mathbf{x}_t) \int_S p(\mathbf{x}_t|u_t, \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}) d\mathbf{x}_{t-1}, \tag{4}$$

where Z is a normalizer. Equation (4), known as the Bayes' filter, provides an efficient, recursive way to update the state distribution.

The Kalman filter is a form of Bayes filtering that assumes that all probability distributions are Gaussian such that $p(\mathbf{x}_t) = N(\mu_t, \Sigma_t)$ with mean μ_t and covariance Σ_t, and that the transition and observation Gaussians are linearly parameterized by the state and control. The Extended Kalman filter (EKF) allows the same inference algorithm to operate with non-linear transition and observation functions by linearizing these functions around the current mean estimate. More formally, the next state $\mathbf{x}_t$ and observation z_t are given by the following functions,

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}, u_t, w_t), \qquad w_t \sim N(0, W_t), \tag{5}$$

$$\text{and} \qquad z_t = h(\mathbf{x}_t, q_t), \qquad q_t \sim N(0, Q_t), \tag{6}$$

where u_t is a control action, and w_t and q_t are random, unobservable noise variables. The EKF computes the state distribution at time t in two steps: a process update based only on the control input u_t leading to an estimate $p(\overline{\mathbf{x}}_t) = N(\overline{\mu}_t, \overline{\Sigma}_t)$, and a measurement update to complete the estimate of $p(\mathbf{x}_t) = N(\mu_t, \Sigma_t)$. The process step follows as:

$$\overline{\mu}_t = g(\mu_{t-1}, u_t), \qquad \overline{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + V_t W_t V_t^T, \tag{7}$$

where G_t is the Jacobian of g with respect to x and V_t is the Jacobian of g with respect to w. For convenience, we denote $R_t \triangleq V_t W_t V_t^T$. Similarly, the measurement step follows as:

$$\mu_t = \overline{\mu_t} + K_t(H_t \overline{\mu}_t - z_t), \qquad \Sigma_t = (I - K_t H_t)\overline{\Sigma}_t, \tag{8}$$

where H_t is the Jacobian of h with respect to $\mathbf{x}$ and K_t is known as the Kalman gain, given by

$$K_t = \overline{\Sigma}_t H_t^T \left(H_t \overline{\Sigma}_t H_t^T + Q_t \right)^{-1}. \tag{9}$$

An alternate form of the EKF represents the distribution $p(\mathbf{x}_t | u_{1:t}, z_{1:t})$ by an information vector ξ_t and information matrix $\Omega_t \triangleq \Sigma_t^{-1}$. The information form may be more efficient to compute in domains where the information matrix is sparse (Thrun et al., 2004). The information matrix update can be written as

$$\overline{\Omega}_t = \overline{\Sigma}_t^{-1} = \left(G_t \Sigma_{t-1} G_t^T + R_t \right)^{-1} \tag{10}$$

$$\Omega_t = \overline{\Omega}_t + H_t^T Q_t^{-1} H_t. \tag{11}$$

For convenience, we denote $M_t \triangleq H_t^T Q_t^{-1} H_t$ such that $\Omega_t = \overline{\Omega}_t + M_t$.

B. The Unscented Kalman Filter

The critical step of the BRM algorithm is the construction of the transfer function, which depends on terms R_t and M_t, the projections of the process and measurement noise terms into the state space. R_t and M_t also represent the information lost due to motion, and the information gained due to measurements, respectively. When using the Extended Kalman filter to perform state estimation, these terms are trivial to compute. However, the EKF is not always a feasible form of Bayesian filtering, especially when linearizing the control or measurement functions results in a poor approximation.

One recent alternate to the EKF is the Unscented Kalman filter (UKF) (Julier et al., 1995), which uses a set of $2n + 1$ deterministic samples, known as "sigma points" from an assumed Gaussian density both to represent the probability density of a space of dimensionality n and to directly measure the relevant motion and measurement covariances. These samples are generated according to:

$$\mathcal{X}_t^0 = \mu_{t-1}, \tag{12}$$

$$\mathcal{X}_t^i = \mu_{t-1} + \left(\sqrt{(n+\lambda)\Sigma_t} \right)^i, \quad i = 1, \ldots, n \tag{13}$$

$$\mathcal{X}_t^i = \mu_{t-1} - \left(\sqrt{(n+\lambda)\Sigma_t} \right)^i, \quad i = n+1, \ldots, 2n \tag{14}$$

where $\left(\sqrt{(n+\lambda)\Sigma_t}\right)^i$ is the ith column of the root of the matrix. Each sigma point $\mathcal{X}^i$ has an associated weight w_m^i used when computing the mean, and w_c^i is the weight used when computing the covariance, such that $\sum_{i=0}^{2n} w_c^i = 1$, $\sum_{i=0}^{2n} w_m^i = 1$. The weights and the λ parameters are chosen to match the mean and variance of the assumed Gaussian distribution; the mechanism for choosing these parameters can be found in Julier et al. (1995). The samples are propagated according to the non-linear process model such that

$$\overline{\mathcal{X}}_t^i = g(\mathcal{X}_t^i, u, 0), \tag{15}$$

generating the process mean and covariance

$$\overline{\mu}_t = \sum_{i=0}^{2n} w_m^i \overline{\mathcal{X}}_t^i \tag{16}$$

$$\overline{\Sigma}_t = \sum_{i=0}^{2n} w_c^i (\overline{\mathcal{X}}_t^i - \overline{\mu}_t)(\overline{\mathcal{X}}_t^i - \overline{\mu}_t) + R_t. \tag{17}$$

The measurement function uses the process mean and covariance to create sigma points in the measurement space, which are then used to generate the posterior mean and covariance (μ_t, Σ_t), such that

$$\overline{\mathcal{Z}}_t^i = h(\overline{\mathcal{X}}_t^i, 0) \qquad \overline{\mu}_t^z = \sum_{i=0}^{2n} w_m^i \overline{\mathcal{Z}}_t^i \tag{18}$$

$$S_t = \left(\sum_{i=0}^{2n} w_m^i (\overline{\mathcal{Z}}_t^i - \overline{\mu}_t^z)(\overline{\mathcal{Z}}_t^i - \overline{\mu}_t^z)^T\right) + Q_t \tag{19}$$

$$K_t = \left(\sum_{i=0}^{2n} w_c^i (\overline{\mathcal{X}}_t^i - \overline{\mu}_t)(\overline{\mathcal{Z}}_t^i - \overline{\mu}_t^z)^T\right) S_t^{-1} \tag{20}$$

$$\mu_t = \overline{\mu}_t + K_t(z_t - \overline{\mu}_t^z) \tag{21}$$

$$\Sigma_t = \overline{\Sigma}_t - K_t S_t K_t^T. \tag{22}$$

Note that $R_t \triangleq V_t W_t V_t^T$ and Q_t are the same process and measurement noise terms from the EKF definition given in equations (5-9). The advantage of the UKF is that the process and measurement functions are not projected into the state space by a linearization; instead, the Unscented Transform computes the moments of the process and measurement distributions in the state space itself. As a result, the UKF eliminates the need for linearization and captures the distribution accurately up to the second order, rather than the first order fidelity of the EKF.

Although the UKF provides a mechanism for efficiently tracking the posterior distribution as a Gaussian while avoiding linearization of the measurement model, the UKF no longer calculates the M_t matrix, which is a critical piece of the individual transfer functions ζ_t. Nevertheless, we can still recover M_t from the UKF update directly by working in the information form and noticing that M_t is the information gain due to measurement z_t. We can combine equation (11) and equation (22),

$$\Omega_t = \overline{\Omega}_t + M_t \tag{23}$$

$$\Rightarrow M_t = \Omega_t - \overline{\Omega}_t \tag{24}$$

$$= \Sigma_t^{-1} - \overline{\Sigma}_t^{-1} \tag{25}$$

$$= (\overline{\Sigma}_t - K_t S_t K_t^T)^{-1} - \overline{\Sigma}_t^{-1}. \tag{26}$$

In order to calculate the M_t matrix for a series of points along a trajectory, we therefore generate a prior covariance and compute the posterior covariance as in equation (22). The UKF is still a projection of the measurement noise into the state space, but is a more accurate projection than an explicit linearization of the measurement model. Additionally, the UKF covariance update does not depend on the actual measurement received, exactly as in the EKF.