2014-4

Relative Navigation Approach for Vision-based Aerial GPS-denied Navigation

Timothy McLain
Mechanical Engineering Department, Brigham Young University, mclain@byu.edu

Randal W. Beard
Department of Electrical and Computer Engineering, Brigham Young University, beard@ee.byu.edu

Robert C. Leishman
US AFRL Sensors Directorate, Wright-Patterson AFB

Original Publication Citation

Leishman, R., McLain, T., and Beard, R. Relative Navigation Approach for Vision-based Aerial GPS-denied Navigation, Journal of Intelligent and Robotic Systems, vol. 74, no. 1-2, pp. 97-111, April 2014.

Relative Navigation Approach for Vision-Based Aerial GPS-Denied Navigation

Robert C. Leishman, Timothy W. McLain, Randal W. Beard

Abstract— **GPS-denied aerial flight is a challenging research problem and requires knowledge of complex elements from several distinct disciplines. Additionally, aerial vehicles can present challenging constraints such as stringent payload limits and fast vehicle dynamics. In this paper we propose a new architecture to simplify some of the challenges that constrain GPS-denied aerial flight. At the core, the approach combines visual graph-SLAM with a multiplicative extended Kalman filter. More importantly, for the front end we depart from the common practice of estimating global states and instead keep the position and yaw states of the MEKF *relative* to the current node in the map. This relative navigation approach provides simple application of sensor measurement updates, intuitive definition of map edges and covariances, and the flexibility of using a globally consistent map when desired. We verify the approach with hardware flight-test results.**

I. Introduction

Finding solutions to enable GPS-denied aerial flight in a priori unknown environments is an important research focus. Solutions are critical in applications that require inspecting and surveying areas that are difficult to access and that present a great amount of risk, such as search and rescue operations in damaged buildings after disasters, in underground areas or in urban canyons. The problem is challenging as the robot must discover its own location using only onboard sensors and computational resources. This task requires knowledge of complex elements from sevaral distinct disciplines. Additionally, the aerial vehicles that are used in this type of research also present difficult constraints like limited payload capacities and fast vehicle dynamics; constraints that are complicated further by using onboard generated state estimates in feedback control. Unlike ground robots, these vehicles cannot afford to pause in one place until complex algorithms converge and estimates are sufficiently stable to continue. Only a few researchers have been able to achieve successful flying implementations for autonomous goal-directed flight.

Planar laser scanner-based implementations such as those discussed in [1], [2] require strict assumptions regarding the nature of the environment. Six-degree-of-freedom (6DoF) motion estimation using vision is desirable due to a camera sensor's low cost, low power requirements and light weight. Furthermore, machine vision approaches are more flexible in that they require fewer assumptions about the environment.

Some of the earliest examples of vision-based estimation for quadrotor vehicles are [3], [4], [5]. Among the first to use vision-based estimates in the control loop was [6]. A few others utilize vision-based estimates in the control loop but must use other aids, such as off-board processing [7], [8], simulated vision using motion capture data [9] or artificial markers [7], [10], [11] to enable their approaches.

Huang et al. [12] combine work from [1] and [13] to enable a quadrotor that uses an RGB-D sensor for visual odometry (VO) and mapping. They present results for 3D maps in small environments with estimates in the control loop. However their approach requires feedback into the estimation from loop closure and global optimization algorithms due to the use of globally referenced states. They are unable to complete these two tasks onboard.

Weiss et al. [14] describe a system where parrallel tracking and mapping (PTAM) [15] is merged with an optical-flow algorithm for a down-pointed camera in an EKF framework. The optical-flow algorithm is necessary to maintain stability of the vehicle when the global navigation fails and needs to be reinitialized. They provide results demonstrating the accuracy of the optical-flow algorithm compared to truth and results for

R. Leishman is a Research Engineer, US AFRL Sensors Directorate, Wright-Patterson AFB, OH 45431 rleish@gmail.com

T. McLain is a Professor and Department Chair, Dept. of Mechanical Engineering, Brigham Young University, Provo, UT 84604 mclain@byu.edu

R. Beard is a Professor, Dept. of Electrical Engineering, Brigham Young University, Provo, UT 84604 beard@byu.edu

an autonomous hover. However, as the camera points downward, they are unable to do motion planning with obstacle avoidance.

Tomic et al. [16] introduce a quadrotor which utilizes navigation based on either stereo VO or laser scan matching, a combination which provides robustness. They report autonomous flight results moving from indoor to outdoor environments. The authors discuss the difficulties in dealing with relative measurements from the VO and jumps that occur in global position with the recognition of landmarks. The approach does not maintain a metric map but it does keep a topological one containing known landmarks in the environment. The system utilizes constraints set by the IMAV competition[1] for map initialization and landmark recognition which excludes it from use in general unknown environments.

Fraundorfer et al. [17] present a quadrotor capable of autonomous flight and exploration using stereo VO from forward-looking cameras and optical flow from a downward looking camera. Most of the computation is completed onboard. Graph-based global optimization and loop closure are required for global states and these algorithims are computed offboard. The authors present results for exploration and mapping in unknown environments and also localization within a known map. They emphasize that the optical flow of the downward-pointing camera is essential for the system to function.

In this paper we propose a new architecture to simplify some of the challenges that constrain GPS-denied aerial flight. In the proposed approach, visual graph-SLAM is combined with a multiplicative extended Kalman filter (MEKF), using only a front-facing RGB-D camera, IMU, and sonar altimeter, as shown in Figure 1, to produce filter outputs. The unique aspect about the proposed approach is that the position and yaw estimates of the MEKF are kept *relative* to the current node in the map, rather than to the global reference frame. Requiring global states incurs difficulties like the need for additional states to incorporate relative position measurements [14], [16], waiting periods for global consistency [6], inclusion of place recognition and map optimization algorithms in the time-critical path [1], [12], [17] and additional logic to accommodate large jumps in pose when loop closures are applied [16].

[1]www.imav2011.org

Fig. 1. The Mikrokopter hexacopter that is used to carry out the experiments. The only sensors utilized are an altimeter (not visible), IMU, and front-facing RGB-D camera.

We demonstrate in this paper that by maintaining relative information in the state estimates, vision-based measurements can be utilized directly, feedback to the filter from computationally expensive loop closure or SLAM algorithms is not required, and processes that are not essential for real-time estimation and control can be completed in the background. Additionally, the basis for the approach has been shown to scale well to large environments [18] and the images from the RGB-D camera represent a rich source of information for path planning and other high-level tasks.

The remainder of the article is outlined as follows. We explain the approach and advantages to relative navigation in Section II. The main result of the paper is in Section III, where the software architecture described in detail. Information is provided on each of the important algorithms that make up this architecture. Next, the results from several different flights of a hardware prototype are detailed in Section IV. These results demonstrate the potential of the proposed approach to provide a vehicle the abilityto navigate in GPS-denied environments. Finally in Section V, we summarize the paper.

II. Relative Navigation Approach

Relative navigation refers to navigation with respect to a local reference frame. We propose that the local frame change as the vehicle moves through the environment, establishing a topological representation of the world using a pose graph [19]. The changes in the local frame occur based on the needs of the VO algorithm. The algorithm is *keyframe* based. Instead of comparing consecutive images, each current image is compared to a reference image, called a keyframe, to obtain the 6DoF change in pose. New keyframes

are declared when the vehicle has moved further than a predetermined distance threshold from the previous keyframe and the overlap between images becomes too small for reliable matching. The local coordinate frames with respect to which the vehicle navigates are derived from the keyframes.

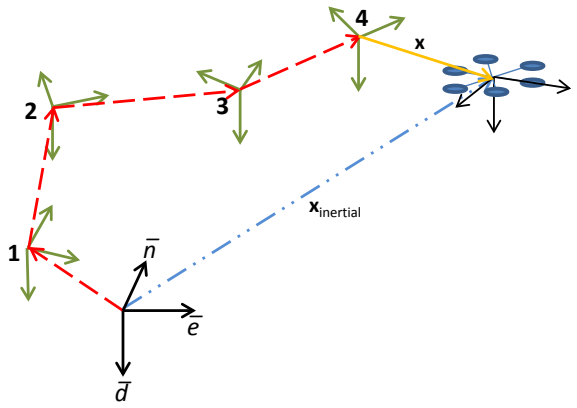

Fig. 2. Relative navigation using nodes and edges. As the vehicle flies through the environment, nodes are created using the VO keyframes and the edges are defined between them using the relative states of the MEKF. The vehicle state is relative to node four in this illustration.

The map in Figure 2 illustrates the relative topological approach. The VO algorithm initializes a keyframe at node 1 and an edge is added between the global frame and the node frame once this information is known. The filter estimates the position and yaw states of the vehicle with respect to the local coordinate frame at node 1 as the vehicle travels. When the VO algorithm requires a new keyframe to maintain good performance, a new keyframe and node are declared at pose 2. An edge is added to the map using the relative states and covariance in the MEKF. The navigation then continues with respect to node 2 by marginalizing out the old relative states and augmenting the state vector with new ones. This process continues as the vehicle moves through the environment, with new keyframes and nodes being declared as necessary and the MEKF changing the relative states each time a new keyframe is declared. As current images are compared to a keyframe, the position estimates will not drift when the vehicle is in hover. A vector chain of edges connects the hexacopter to the global reference frame. Global position and yaw for the vehicle can be estimated by first expressing all of the constraints in the same coordinate frame and then summing all of the edges and the current state.

This relative navigation approach has several key advantages: straightforward use of sensor information for state updates, easy creation of map edges using the filter state and covariance, and flexible use of global information.

Exteroceptive sensors provide relative information. In particular, the VO provides the change in 6DoF pose between the current and keyframe images. By expressing the VO result in the node coordinate frame, the position and attitude are updated directly in the filter. This simplification eliminates needing additional states in the filter or requiring VO measurements to update the velocity states.

Defining edges between consecutive nodes is a simple matter of saving the relative portions of the state and covariance just before a new node is created. The covariance can be used to compute a confidence measure of the current global position. For example, a path planning algorithm might use the combined covariances of the edges to indicate when estimates have drifted sufficiently to warrant a planned loop closure.

The proposed relative approach offers more flexibility than a globally-based method. The system can fly reliably both with and without loop closure constraints that constrain drift and with and without global optimization. This is possible as the local navigation and control take place regardless of global changes within the map. Without loop closure it is clear that the map will drift and not remain globally consistent. However, the relative relationships between nodes maintain locally consistent topological and metric relationships between saved locations. Therefore, the map could be traversed, even back to the start location, by using the relative relationships. This is also true when loop closure constraints are available and global optimization is not; we could then pursue a consistent but purely relative topological approach similar to that of [20]. Finally, by enabling both loop closure and global optimization we would be able to mimic the typical SLAM approach that provides globally consistent metric information of the environment.

III. Software Architecture

Figure 3 show the architecture of the proposed relative navigation system. The system is divided into the front end and back end. The whole system is intended to be run onboard a hexacopter, like the one shown in Figure 1.

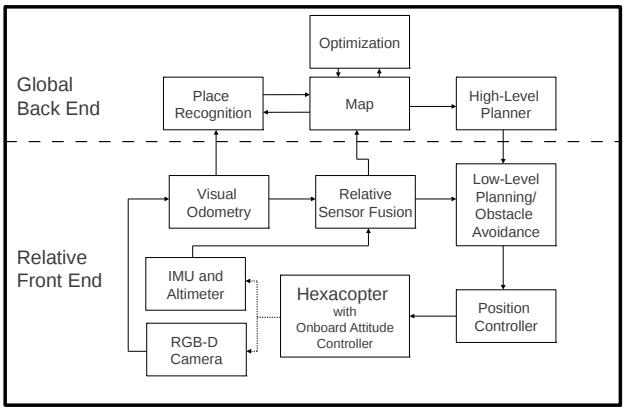

Fig. 3. Software architecture for the proposed system. The front end provides relative navigation based on keyframes from the VO algorithm. The back end provides a globally-consistent navigation solution. Notice that the only flow of information from the back end to the front end is provided by the high-level planner. Most proposed architectures require feedback from the costly optimization and place recognition algorithms to the estimation and control. ROS provides the functionality represented by the arrows between components.

The front-end subsystem provides the critical processes to keep the hexacopter flying, including the VO, sensor fusion, control, and obstacle avoidance. Consequently, this system is given priority over the back end. All of the components in the front end are based in the relative coordinate frame as explained above.

The back-end subsystem maintains globally consistent information, including a global map and high-level, global objectives when desired. Notice how the only flow of information from the back end to the front end is from the high-level planner. This is in stark contrast to other proposed architectures that have been developed, which require feedback from the computationally expensive recognition and optimization components. The relationship, illustrated in Figure 3, between the front and back ends is what allows the flexibility of the proposed approach.

The separation between the front and back ends provides an added level of robustness to any changes in the pose graph. For example, when loops are closed and global optimization is employed, it is possible for large jumps in the global location to occur. These large jumps can cause problems with the real-time control of an air vehicle that employs globally-referenced states. As our vehicle navigates with respect to a local node, global optimization can continually make changes without causing harm to the real-time estimation and control. Another advantage is the potential to utilize other types of constraints in the map between nodes, also without effecting the real-time essential processes. Possibilities include any measurement constraints which aid in the understanding of the global or relative vehicle location, such as intermittent GPS measurements or semantic information [16].

The system is implemented using the Robot Operating System (ROS) [21]. In fact, messages within the ROS framework make up all the arrows in Figure 3 and each block is written as a ROS node/package. Below we briefly describe each block that makes up the proposed relative navigation approach.

A. Visual Odometry

As we discussed above, VO is the process of comparing two images to find the relative change in pose between them. We utilize keyframes in these comparisons, rather than consecutive images, to reduce the amount of drift. Good tutorials on implementing VO are found in [22], [23].

We utilize a VO algorithm we developed that utilizes the 3D information from an RGB-D camera [24]. We provide a quick summary of the algorithm below.

1) 3D VO: Sets of color and depth images are sent to the algorithm. The first image pair is designated as the keyframe image pair and all following image pairs are compared to this set until a new keyframe image is assigned. A new keyframe is selected when the camera has moved 0.25 meters or 10 degrees in yaw from the location where the previous keyframe image was taken.

For each image FAST features [25] and BRIEF descriptors [26] are extracted and the feature positions are corrected using the distortion information of the camera. The 3D point location $\mathbf{p} = (X\ Y\ Z)^\top$ for the 2D image feature $\bar{\mathbf{p}} = (x\ y)^\top$ is found by looking up the depth Z in the depth image and using the projection equations

$$X = \frac{(x - c_x)Z}{f_x} \tag{1}$$

$$Y = \frac{(y - c_y)Z}{f_y}, \tag{2}$$

where c_x, c_y, f_x, and f_y are the intrinsic camera calibration parameters for the image center and focal points.

Next, correspondence between the current image features and the keyframe features are estimated using forward and backward constrained brute-force searches

in a mutual consistency check [22]. The corresponding features are passed into RANSAC [27], which is then employed to find a pose motion estimate while eliminating outliers. We use a three point singular value decomposition (SVD) algorithm based on [28] as the motion model in RANSAC. The solution estimate provides the 6DoF rotation and translation between the keyframe and the current coordinate frames and is of the form

$$\mathbf{p}^c = \mathbf{R}^c_{key}\mathbf{p}^{key} + \mathbf{T}^c. \quad (3)$$

Where $\mathbf{p}^c$ and $\mathbf{p}^{key}$ are the current and keyframe 3D feature position vectors, $\mathbf{R}^c_{key}$ rotates points expressed in the keyframe coordinate frame into the current image coordinate frame, and T^c is the origin of the keyframe coordinate frame expressed in the current image coordinate frame.

Inliers are found by re-projecting the 3D keyframe features onto the current image plane using the sample solution and the intrinsic calibration parameters. To be considered a valid inlier, the pixel error distance between the locations of the current image feature and the reprojected keyframe feature must be smaller than a threshold. Checking the error on the image provides improved results over solutions evaluated by error in 3D positions. The solution estimate with the highest inlier count is returned by the RANSAC algorithm.

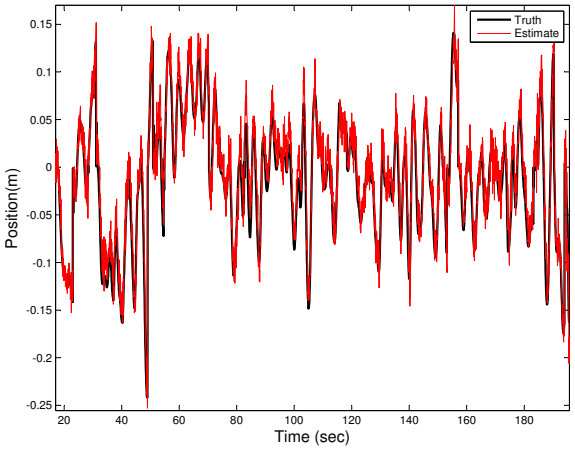

Fig. 4. Relative camera $\vec{z}$ (out of plane) position comparison between truth and estimates. The discontinuities in the plots are due to new nodes being created, causing the truth and the estimates to jump to the new relative position. We express the global truth from the motion capture system in the relative node coordinate frame for easy comparison. Results for the camera $\vec{x}$ and $\vec{y}$ positions are similar.

2) 3D VO Results: Figure 4 presents the camera $\vec{z}$ axis portion of the relative transformations between the current and keyframe images. The discontinuities are due to changing keyframes. Figure 5 presents the results for the rotation about the camera $\vec{y}$ axis. This angle corresponds to the vehicle yaw angle because of the change in axes. The algorithm actually outputs the change in rotation as a unit quaternion, which we have converted to Euler angles in this figure for ease of comparison.

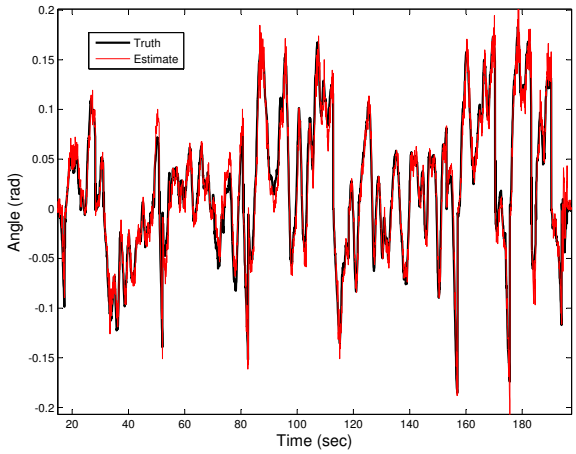

Fig. 5. Comparison between truth and estimates for the rotation about the camera $\vec{y}$ axis, which is equivalent to the vehicle yaw angle because of the change in axes. Again, the discontinuities in the plots are due to new nodes being created, causing the truth and the estimates to jump to the new relative position. Results for the camera roll and yaw positions are similar.

Table I provides the RMS error in the relative transformations over the flight. These values are representative of results that we routinely achieve in the motion capture environment.

TABLE I
RMS ERROR OF THE VISUAL ODOMETRY ESTIMATES.

RMS Error in Motion Estimates	
Transformation	**RMS Error**
Camera $\vec{x}$ position	0.033 (m)
Camera $\vec{y}$ position	0.041 (m)
Camera $\vec{z}$ position	0.041 (m)
Rotation about camera $\vec{x}$	0.020 (rad)
Rotation about camera $\vec{y}$	0.017 (rad)
Rotation about camera $\vec{z}$	0.013 (rad)

B. MEKF Sensor Fusion

The sensor fusion is provided by a multaplicative extended Kalman filter (MEKF) that has been designed

specifically to function with the relative navigation approach [29]. The MEKF is an indirect EKF, which means that the error in the state $\Delta\mathbf{x}$ and the covariance of the error are maintained in the filter rather than the best estimate $\hat{\mathbf{x}}$ and error covariance.

The true states $\mathbf{x}$ of the rotorcraft are defined as

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}^{n\top} & \mathbf{q}_n^{b\top} & \mathbf{v}^{b\top} & \beta^\top & \alpha^\top & \mathbf{q}_c^{b\top} & \mathbf{p}^{b\top} \end{bmatrix}^\top. \quad (4)$$

The position vector $\mathbf{p}^n$, relative to the current node, is the displacement of the body in the front f_j, right r_j, and down d_j directions with respect to node j. The quaternion $\mathbf{q}_n^b$ expresses the attitude of the body-fixed frame with respect to the node frame. The component of the quaternion for yaw is relative to the current node. $\mathbf{v}^b$ is the body frame velocity vector. The gyroscope bias vector is β. Accelerometer biases in the body x and y directions are represented by α. The last two parameters in (4) represent the transformation from the body-fixed coordinate frame to the camera coordinate frame and can be optionally included in the state. Once refinements to the transformation are obtained, these estimates can be saved as constants and then removed.

The inputs to the model are the gyroscope measurements and the z accelerometer

$$\mathbf{u} = \begin{bmatrix} p_{gyro} & q_{gyro} & r_{gyro} & z_{accel} \end{bmatrix}^\top. \quad (5)$$

When just the subscript $_{(i:j)}$ is used below, only elements i through j of $\mathbf{u}$ are considered.

The nonlinear equations of motion for the states (4) are

$$\dot{\mathbf{p}}^n = \mathcal{R}^\top(\mathbf{q}_n^b)\mathbf{v}^b, \quad (6)$$

$$\dot{\mathbf{q}}_n^b = \frac{1}{2}\Omega\left(\mathbf{u}_{(1:3)} - \beta - \eta_\omega\right)\mathbf{q}_n^b, \quad (7)$$

$$\dot{\mathbf{v}}^b = \mathbf{v}^b \times \left(\mathbf{u}_{(1:3)} - \beta - \eta_\omega\right) + \mathcal{R}(\mathbf{q}_n^b)\mathbf{g} - \frac{1}{m}\mathbf{M}\mathbf{v}^b + \mathbf{u}_{(4)}\vec{d}_j, \quad (8)$$

$$\dot{\beta} = \eta_\beta \quad (9)$$

$$\dot{\alpha} = \eta_\alpha \quad (10)$$

$$\dot{\mathbf{q}}_c^b = \eta_{cq} \quad (11)$$

$$\dot{\mathbf{p}}^b = \eta_{cp}. \quad (12)$$

A rotation matrix $\mathcal{R}(\mathbf{q}_a^b)$ from a quaternion $\mathbf{q}_a^b$ rotates the vector $\mathbf{v}$, expressed in the frame a, into frame b. The operator

$$\Omega(\omega) = \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix}$$

assumes that the order of a quaternion it multiplies is of the form $\begin{bmatrix} q_x & q_y & q_z & q_w \end{bmatrix}^\top$. The noise η_ω is the zero-mean Gaussian noise in the measured gyroscopes from the inputs $\mathbf{u}$. The constant matrix $\mathbf{M}$ is

$$\mathbf{M} = \begin{bmatrix} \mu & 0 & 0 \\ 0 & \mu & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

and the constants $\mathbf{g}$ and μ are the gravity and drag coefficient respectively. An improved model of the hexacopter dynamics, contained in (8), which accounts for the rotor drag with coefficient μ, provides the ability to fully utilize the information contained in the accelerometer measurements [30]. As a consequence, estimation accuracy improves and the requirements for VO or any other exteroceptive measurement updates are reduced [31].

1) Error Dynamics: The error dynamics are used to propagate the error covariance matrix $\mathbf{P}$ and are derived from the nonlinear dynamics (6) through (12). The length of the error state is reduced by one for each quaternion when compared to the length of the true state. The 20-element relative-error state is

$$\Delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{p}^{n\top} & \delta\theta_n^{b\top} & \delta\mathbf{v}^{b\top} & \delta\beta^\top & \delta\alpha^\top & \delta\theta_c^{b\top} & \delta\mathbf{p}^{b\top} \end{bmatrix}^\top. \quad (13)$$

The error dynamics can be linearized and result in the following linear model

$$\dot{\Delta\mathbf{x}} = \mathbf{A}\Delta\mathbf{x} + \mathbf{B}\Delta\mathbf{u}, \quad (14)$$

where $\mathbf{A}$ is the Jacobian of the error dynamics with respect to the error state $\Delta\mathbf{x}$ and $\mathbf{B}$ is the Jacobian of the error dynamics with respect to the input $\Delta\mathbf{u}$.

2) Prediction: We do not maintain a true indirect filter as we require the estimated states for control feedback. Instead, we keep track of the estimated state $\hat{\mathbf{x}}$ and the covariance $\hat{\mathbf{P}}$ of the error state $\Delta\hat{\mathbf{x}}$; a detailed derivation is found in [29]. During the prediction step, the estimated covariance is propagated forward by numerically integrating the equation

$$\dot{\hat{\mathbf{P}}} = \mathbf{A}\hat{\mathbf{P}} + \hat{\mathbf{P}}\mathbf{A}^\top + \gamma\left(\mathbf{B}\mathbf{G}\mathbf{B}^\top + \mathbf{Q}\right), \quad (15)$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are defined in (14), γ is a tuning parameter, $\mathbf{G}$ is a diagonal matrix of the measured covariance on the inputs (5) and $\mathbf{Q}$ is the process noise covariance which represents modeling error and disturbances. The estimated states are propagated forward by numerically integrating the nonlinear equations of motion, (6) through (12).

3) Measurement Updates: The filter is updated using altimeter, accelerometer, vision-based position, and motion-estimation orientation measurements, from the 3D VO. In this implementation we treat the position and orientation motion estimation measurements updates separately. This is possible because we account for the contribution of the rotational uncertainty in the position covariance when the covariances of the measurements are generated.

4) Delayed View-Matching Updates: An additional challenge with the motion estimation measurement updates is that they are delayed. The stochastic delay is due to the requisite image-processing time. Thus, computing the measurement updates requires a few additional steps. The state and covariance must first be restored to the time the image was taken; then the measurement updates are applied; finally, the state and covariance must be repropagated back to the current time by re-applying the prediction and measurement updates at their respective timesteps. The state, covariance, IMU, and altimeter information are saved at each timestep to accommodate this requirement.

5) Augment and Marginalize the Relative State: When a new node is created by the view-matching algorithm, the relative portions of the state and covariance must change. The states that change are the positions in $\hat{\mathbf{p}}^n$ and the yaw contained in $\hat{\mathbf{q}}_n^b$. The positions are simply replaced with zeros. The quaternion in the state must maintain the same pitch and roll but the yaw must be zeroed out. We use the relationship between quaternions and Euler angles [32] to adjust the state and covariance appropriately.

C. Relative Planning/Obstacle Avoidance

The low-level planner provides paths for the hexacopter to follow through the environment in the relative frame. The plans are recomputed frequently enough for the vehicle to avoid static and slow-moving obstacles, like a person walking at a casual pace. Point cloud data from the RGB-D sensor is used to create a cost map [33] of the 3D environment that is then projected onto the node f_j-r_j plane, as shown in Figure 6. The cost map is expressed in the relative node frame explained above. Given a goal location in the relative coordinate system, a path through the environment is computed using Dijkstra's algorithm [34]. The goal location is the only information received by the front end from the back-end subsystem, as shown in Figure 3. The path is expressed in the relative coordinate system.

Fig. 6. Visualization of the cost map and path planner. Point cloud information was provided to the relative path planner as the camera was moved around a room. The red objects are detected obstacles, the yellow areas are inflated costs around the obstacles, and the green line is the computed path, based on a goal location chosen by a user.

D. Position Control

We have modified the position controller detailed in [35] to provide control based on waypoints in the relative node coordinate frame and to include integral control for more robustness. The control algorithm utilizes a change of variables on the inputs of the model to eliminate nonlinearities and a linear quadratic regulator (LQR) provides the feedback control. The approach to follow waypoints is based on the procedure outlined in [36], adapted for rotorcraft. The waypoints are expressed in the current node frame when sent by the planning algorithm.

One challenge for the control is the change of coordinate systems when a new node is created. The sensor fusion algorithm changes the coordinate system of the relative states as soon as the keyframe image is

received. The path planner, however, does not instantly generate a new path for the new reference frame. Consequently the control algorithm must apply the relative transformation provided by the map edge to the old path. Each waypoint $\mathbf{wp}[i]$ in the path is transformed using

$$\mathbf{wp}_{new}[i] = \mathcal{R}\left(\mathbf{q}_j^{j+1}\right)\left(\mathbf{wp}[i] - \mathbf{p}^{j+1}\right), \qquad (16)$$

where $\mathbf{p}^{j+1}$ is the translation to the $(j+1)$-th node from the j-th node, expressed in the j-th node frame.

E. Map

The map used in this work is a collection of nodes and edges in a relative topological pose graph, illustrated in Figure 2. The map is flexible as it can be globally referenced through optimization but it is originally based on the relative transformations provided by the motion estimation. New nodes are created with each new keyframe. Edges are added between temporally and spatially consecutive keyframes.

A node is described, ultimately, by a keyframe RGB and depth image pair. Attached to the keyframe pair are the estimates of relative and global position and orientation, yet the image encodes the true instantaneous location of the vehicle. A relative local coordinate frame is defined as part of the node, based on the position and heading of the vehicle when the keyframe is taken, to enable navigation relative to the node.

Edges in the graph represent the estimated relative transformations between nodes. We currently only consider edges from the odometry but we are working to include other constraints, such as those from visual recognition loop closures and intermittent GPS measurements. The odometry edges are created using the MEKF, based on the measurements from the robust motion estimation algorithm. When a new node is received by the estimator, the old relative portions of the state and covariance are marginalized out and saved as the edge between the old and new nodes.

F. Place Recognition

Place recognition provides the capability to recognize when the current keyframe is already part of the map. Once the algorithm recognizes a match, a loop-closure constraint can be added to the map using one of the motion-estimation algorithms. Loop-closure constraints are essential to providing a topological consistent map as they constrain the drift in the map caused by odometry errors.

Place recognition is completed by comparing images to one another to find close matches [37], [38]. Each keyframe image in the map is assigned visual words, from a previously calculated visual vocabulary, based on the feature information in the image. Then the map is searched using the words to find images that contain the same information. Once several images are suggested by the algorithm as having a high probability of being the same location, a geometric consistency check is made to eliminate any false positive matches. The 6DOF loop-closure constraint is created by comparing the matching images using a motion estimation algorithm. This algorithm is an item of current work and it not yet fully implemented in the system.

G. Back-end Optimization

The role of nonlinear optimization is to iteratively refine the edges in the map to produce a globally consistent map when it is desired. Because of the flexibility of the relative navigation approach, this can be completed either offline after a flight, or in real time as a background process. In most navigation approaches, the sensor fusion relies on the revised global estimates, causing the computationally heavy optimization to be a part of the time-critical path that enables flight.

In [39], a new optimization approach is introduced, which focuses on the relative transformations between nodes rather than only on the global pose estimates, as is typically done. As a result, the algorithm provides improved estimates of the global poses and relative transformations in less computational time than other the state-of-the-art algorithms [40].

H. High-level Planner

The role of the high-level planner is to provide capabilities such as exploration, target following, or other higher-level tasks for the hexacopter system. The algorithm is provided an estimate of the map and the location of the hexacopter, as well as the current relative coordinate system in use by the front-end subsystem. Directions are then provided to the low-level planner in the form of goal locations in the current relative coordinate system. This setup allows the front-end subsystem flexibility. It does not need global information and it is allowed to create its own paths so that obstacles can be avoided.

IV. Experimental Setup and Results

We utilize the Mikrokopter hexacopter vehicle shown in Figure 1, and the hardware specified in Table II

for the experimental results. The computer is running Ubuntu 12.04 Linux and all the applications are implemented in C++ and connected using ROS. Truth data from a motion-capture system is only used to initialize the global position of the vehicle and in the comparisons made in the figures below. The relative MEKF runs at 100 Hz, the update rate of the IMU. Measurement updates for the altimeter and the visual odometry algorithm are applied at 40 Hz and 15 Hz, respectively. All of the processing is performed onboard. During the experiments presented below, the CPU usage averaged at about 40%. From these measurements, we believe that there is sufficient room for the place recognition and optimization algorithms to also run on the onboard computer.

TABLE II
HARDWARE DETAILS

Component	Description
Vehicle	Mikrokopter Hexacopter XL
Autopilot	Flight-Ctrl V2.1 ME
Sonar Altimeter	LV-MaxSonar®-EZ3
RGB-D Camera	ASUS Xtion Pro Live
IMU	MicroStrain® 3DM-GX3®-15
Motion Capture	Motion Analysis
Processor	Intel Core i7-2710QE

We present results for an autonomous hover which demonstrate the performance of the estimator and control algorithms. The estimates are compared to truth and the control maintains the vehicle in a hover about a fixed global location. We also show the true and estimated 3D positions of the vehicle while following a path in three different circumstances: the first is a short path in the motion capture room, the second is down a long, straight hallway, and the third is a path down three hallways.

A. Hover Results

Table III provides the standard deviations of the hover error during a typical flight. The vehicle was commanded to hover 1 m above the take-off location. Twelve nodes were created during this flight, but only three occurred during the hover, the others were created during takeoff and landing. Notice that even though the vehicle is navigating using relative states, it can stabilize quite well around a global location.

TABLE III
THE STANDARD DEVIATIONS IN THE GLOBAL DIRECTIONS OF THE HOVER ERROR.

Standard Deviation of Hover Error	
Direction	**Standard Deviation (m)**
global north (n)	0.083
global east (e)	0.071
global down (d)	0.02

B. Path Results

The next flight was an autonomous, goal-directed flight in a small room equipped with a motion capture system. The vehicle is performing all of the tasks of the front-end sub-system described in Figure 3, with all of the computation being completed onboard. Figures 7 through 9 demonstrate the performance of some of the state estimates of the filter compared to truth during a flight with the state estimates in the control loop.

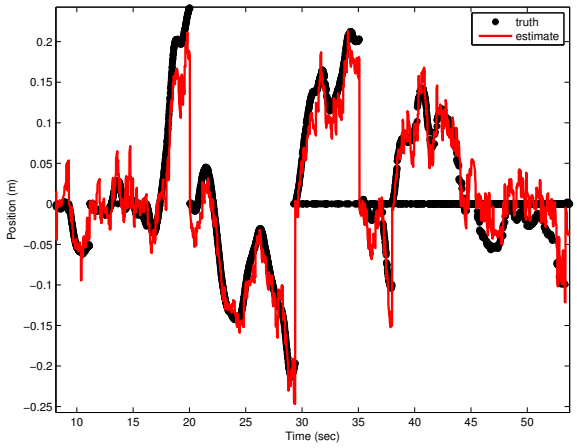

Fig. 7. Relative right position r truth and estimates comparison. This data is from the estimates-in-the-loop autonomous, goal-directed flight. The discontinuities in the plots are due to new nodes being created, causing the truth and the estimates to "jump" to the new relative position. We express the global truth from the motion capture in the relative node coordinate frame for the comparison of these results. Results for the relative front and down positions are similar.

In Figure 7 we see the results for the relative right position r, with respect to the current node. There were many new nodes created during this autonomous flight. We note that all of the state estimates transition between these coordinate-frame changes without difficulty. The body-fixed frame side velocity v results are depicted in Figure 8. The estimates track the truth, even though the magnitude of the speed is small. The y component

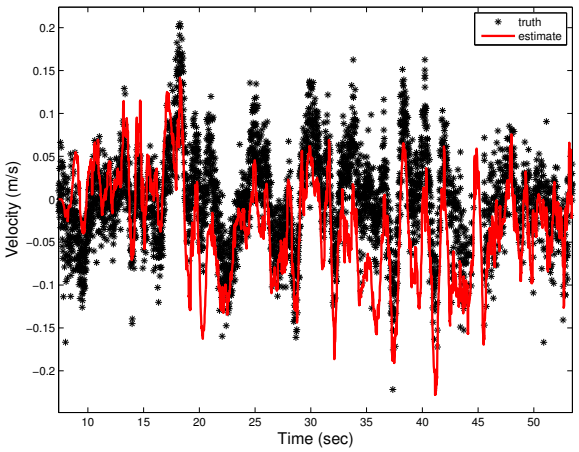

Fig. 8. Body-fixed frame side velocity v truth and estimate comparison. Notice that there are no discontinuities, as the body-frame velocity is not relative. Results for the front and down body-fixed velocities are similar.

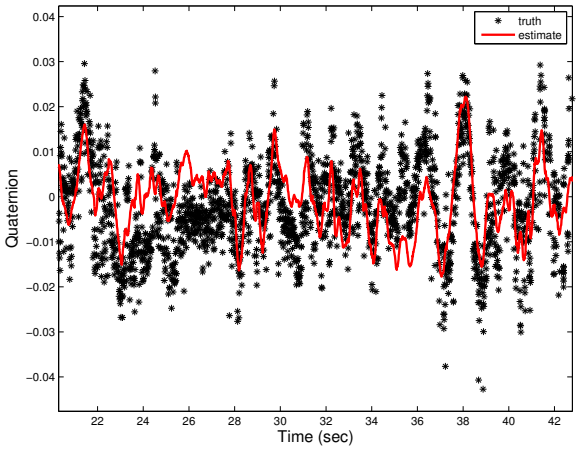

Fig. 9. The y component of the quaternion $\mathbf{q}_n^b$, which is approximately the pitch angle of the hexacopter for this flight, comparison for a portion of the flight. There are not any discontinuities, as this part of the quaternion is not relative.

of the quaternion $\mathbf{q}_n^b$ is shown in Figure 9. The y quaternion roughly corresponds to the pitch angle of the hexacopter for this flight.

Figures 10 and 11 show the global, dead-reckoning results for the autonomous, goal-directed flight. The goal locations that were commanded are shown in the figures. Recall that the state estimates, control, path planning, and goal locations are all originally relative to the current node in the graph. We have converted them into global estimates for display and comparison. The estimated global node locations are shown as green points along the estimated path. There is drift in the global locations as we are only conducting relative

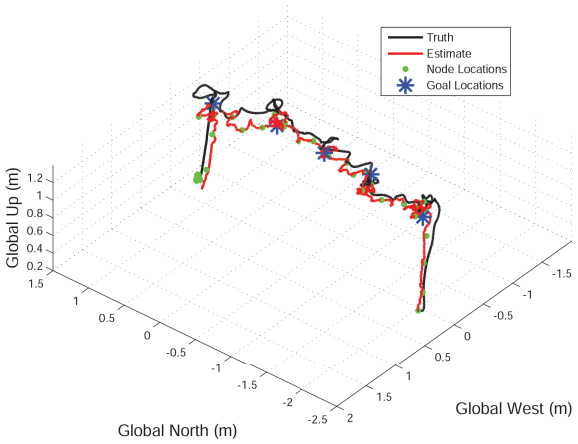

Fig. 10. The 3D path of a flight within the motion capture environment. We show the true path, the global estimate computed by summing the relative edges and the current state at each timestep, the node locations estimates, and the global positions of the relative goal points. Notice that even though the estimates drift globally, the vehicle arrives at each of the goal locations. This is possible since all the front end functionality is based on the relative system.

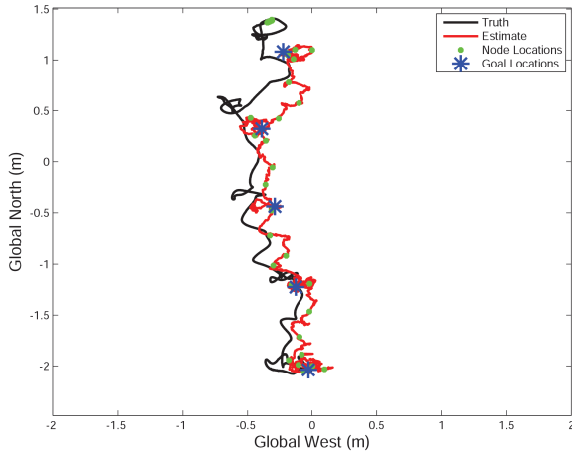

Fig. 11. The top view of the 3D path of a flight within the motion capture environment. Notice that even though the estimates drift globally, the vehicle arrives at each of the goal locations. This is possible since all the front end functionality is based on the relative system.

flights.

C. Long Hallway Results

In this section we describe results of flying the hexacopter down the main hallway of the third floor of the Crabtree building on the BYU campus,[2] see Figure 12. A long, large hallway is a challenging

[2]A video of this flight can be seen on the MAGICC Lab YouTube channel: `http://youtu.be/PMYwPihUKIM`

Fig. 12. The main hallway in the Crabtree building on BYU campus.

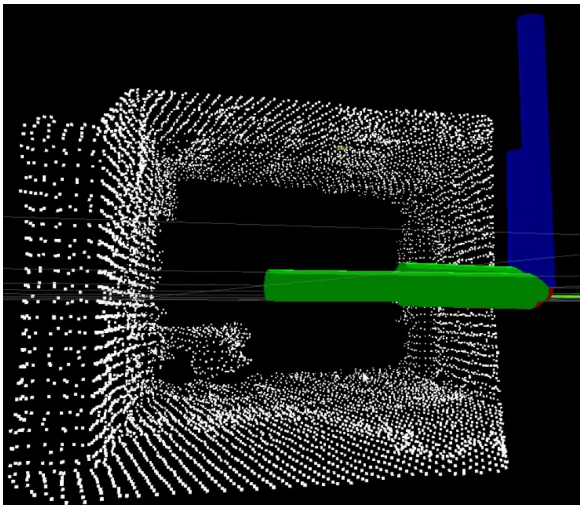

Fig. 13. The 3D point cloud produced by the RGB-D camera while the hexacopter was moving down the hallway. The valid information for use in the VO is only on the periphery of the image.

environment for the VO, as 3D information is only available on the periphery of the sensor: on the floor, walls and ceiling. Figure 13 shows a typical point cloud obtained while the hexacopter was moving down the hallway.

Figure 14 shows a top view of the global estimates of the path of the vehicle. The original reference frame was closely aligned with the hallway, so the vehicle drifted between 1 to 2 m laterally. This drift is not concerning since the vehicle navigates relative to the keyframes.

D. Flight Down Three Hallways

The final demonstration was to test how well the system could maneuver down hallways that require turns. Results for a flight on the third floor of the

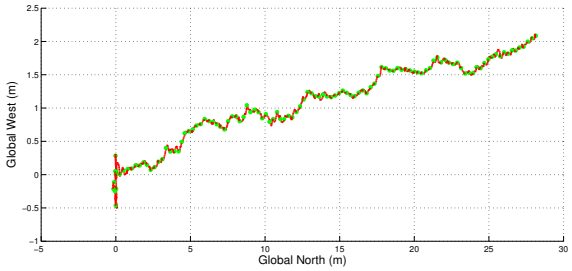

Fig. 14. The top view of the global position estimates. Note the difference in scale between the axes. These are dead-reckoning results, as optimization and loop closure are not enabled. Lateral drift is between one and two meters. The green dots denote the keyframe images created along the way.

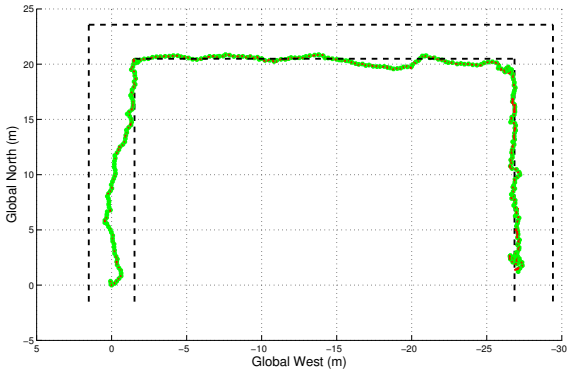

Fig. 15. The top view of the global position estimates for the flight down three hallways in BYU's Wilkinson Student Center. These are dead-reckoning results, as optimization and loop closure are not enabled. From this figure, we estimate the drift in north and west to be between one and two meters in each direction. The green dots denote the keyframe images created along the way.

Wilkinson Student Center on BYU campus are shown in Figure 15. In black, we show the basic dimensions of the hallways. We set goal locations near the center of the hallways. Notice that there is very little drift in the global yaw, otherwise, the path would not be as square. Recall that again, these are dead-reckoning results.

These larger path results demonstrate further evidence for the potential functionality of the proposed approach. The vehicle is able to autonomously fly through larger environments and transition through many different local coordinate frames, while simultaneously maintaining good dead-reckoning global estimates. However, the back-end algorithms mentioned above must be in place to demonstrate the system in a more practical environment.

V. Summary

A relative, vision-based framework, like the approach described here, is an important step in furthering

the capabilities of indoor aerial navigation. Alternative approaches that require globally-referenced states often suffer deficiencies from the need for additional state elements to incorporate relative measurements, waiting periods to process global consistency, inclusion of place recognition and map optimization algorithms in the time-critical path, or schemes to accommodate large jumps in pose when loop closures are applied.

Utilizing a relative approach allows more flexibility as the critical, real-time processes of localization and control do not depend on computationally-demanding optimization and loop-closure processes. Relative exteroceptive measurement updates are supported natively in the proposed MEKF and front-facing keyframes provide a rich source of information for path planning. The graph map also provides potential support for a variety of constraints, such as intermittent GPS and semantic information.

ACKNOWLEDGMENTS

This work was supported through the DoD SMART Scholarship program.

REFERENCES

[1] A. Bachrach, R. He, and N. Roy, "Autonomous Flight in Unstructured and Unknown Indoor Environments," in *Proc. of the EMAV Conference*. European Micro Air Vechicle, Sept. 2009, pp. 2–9.

[2] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained MAV," in *IEEE Intl. Conf. on Robotics and Automation*, May 2011, pp. 20–25.

[3] E. Altug, J. P. Ostrowski, and C. J. Taylor, "Control of a Quadrotor Helicopter Using Dual Camera Visual Feedback," *The International Journal of Robotics Research*, vol. 24, no. 5, pp. 329–341, May 2005.

[4] G. P. Tournier, M. Valenti, J. P. How, and E. Feron, "Estimation and control of a quadrotor vehicle using monocular vision and moire patterns," in *AIAA Guidance, Navigation and Control Conference*, no. August, 2006, pp. 21–24.

[5] S. Ahrens, D. Levine, G. Andrews, and J. P. How, "Vision-based guidance and control of a hovering vehicle in unknown, GPS-denied environments," in *IEEE Intl. Conf. on Robotics and Automation*, May 2009, pp. 2643–2648.

[6] M. Blosch, S. Weiss, D. Scaramuzza, and R. Siegwart, "Vision based MAV navigation in unknown and unstructured environments," in *Proc. IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 21–28.

[7] F. Bourgeois, L. Kneip, S. Weiss, and R. Siegwart, "Delay and Dropout Tolerant State Estimation for MAVs," in *Proc. Intl. Symposium on Experimental Robotics*, Berlin, 2010, pp. 1–14.

[8] L. R. García Carrillo, A. E. Dzul López, R. Lozano, and C. Pégard, "Combining Stereo Vision and Inertial Navigation System for a Quad-Rotor UAV," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 1–15, Aug. 2011.

[9] R. Leishman, J. Macdonald, R. W. Beard, and T. McLain, "Relative navigation and control of a hexacopter," in *IEEE Intl. Conf. on Robotics and Automation*, St. Paul, MN, USA, May 2012, pp. 4937–4942.

[10] L. Meier, P. Tanskanen, F. Fraundorfer, and M. Pollefeys, "PIXHAWK: A system for autonomous flight using onboard computer vision," in *EEE Int. Conf. on Robotics and Automation*, May 2011, pp. 2992–2997.

[11] D. Bohdanov and H. Liu, "Vision-based Quadrotor Micro-UAV Position and Yaw Estimation and Control," in *Proc. AIAA Conf. on Guidance, Navigation, and Control*, 2012.

[12] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera," in *Int. Symposium on Robotics Research*, Flagstaff, Arizona, USA, 2011.

[13] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments," in *Int. Symposium on Experimental Robotics*, 2010.

[14] S. Weiss, M. Achtelik, S. Lynen, M. Chli, and R. Siegwert, "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments," in *IEEE Intl. Conf. Robotics and Automation*, 2012.

[15] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *IEEE and ACM Int. Symposium on Mixed and Augmented Reality*, Washington, DC, USA, 2007, pp. 1–10.

[16] T. Tomic, K. Schmid, P. Lutz, M. Kassecker, E. Mair, I. Grixa, F. Ruess, M. Suppa, and D. Burshka, "Toward a Fully Autonomous UAV: Research Platform for Indoor and Outdoor Urban Search and Rescue," *Robotics and Automation Magazine*, no. September, 2012.

[17] F. Fraundorfer, L. Heng, D. Honegger, G. H. Lee, P. Tanskanen, and M. Pollefeys, "Vision-Based Autonomous Mapping and Exploration Using a Quadrotor MAV," in *IEEE Intl. Conf. on Intelligent Robots and Systems*, 2012.

[18] K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua, "View-based Maps," in *Proc. of Robotics: Science and Systems*, vol. 29, no. 8, Seattle, USA, June 2009.

[19] B. Kuipers and Y. Byun, "A robust, qualitative method for robot spatial learning," *Proc. of the AAAI*, 1988.

[20] G. Sibley, C. Mei, I. Reid, and P. Newman, "Planes, trains and automobiles: autonomy for the modern robot," in *IEEE Int. Conf. on Robotics and Automation*, May 2010, pp. 285–292.

[21] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS : an open-source Robot Operating System," in *IEEE Intl. Conf. on Robotics and Automation Workshop on Open Source Robotics*, Kobe, Japan, 2009.

[22] B. D. Scaramuzza and F. Fraundorfer, "Visual Odometry Part I: The First 30 Years and Fundamentals," *Robotics and Automation Magazine*, no. December, pp. 80—-92, 2011.

[23] F. Fraundorfer and D. Scaramuzza, "Visual odometry Part 2: Matching, Robustness, Optimization, and Applications," *Robotics and Automation Magazine*, no. June, pp. 78—-90, 2012.

[24] R. C. Leishman, D. Koch, and T. W. McLain, "Robust Motion Estimation Using an RBG-D Camera," in *AIAA Infotech @ Aerospace Conference*, Boston, MA, USA, 2013.

[25] T. D. Edward Rosten, "Machine learning for high-speed cor-

ner detection," in *European Conference on Computer Vision*, 2006, pp. 430—-443.

[26] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *Computer Vision ECCV 2010*, ser. Lecture Notes in Computer Science, K. Daniilidis, P. Maragos, and N. Paragios, Eds. Springer Berlin Heidelberg, 2010, vol. 6314, pp. 778–792.

[27] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[28] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-D point sets," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 9, no. 5, pp. 698–700, May 1987.

[29] R. C. Leishman, "A Vision-based Relative Navigation Approach for Autonomous Multirotor Aircraft," Ph.D. dissertation, Brigham Young University, 2013.

[30] R. C. Leishman, J. Macdonald, R. W. Beard, and T. W. McLain, "Quadrotors & Accelerometers State Estimation with an Improved Dynamic Model," *Control Systems Magazine, to Appear*, 2013.

[31] J. Macdonald, R. C. Leishman, R. W. Beard, and T. W. McLain, "Analysis of an Improved IMU-Based Observer for Multirotor Helicopters," *Journal of Intelligent & Robotic Systems, to Appear*, 2013.

[32] J. Kuipers, *Quaternions and Rotation Sequences*. Princeton University Press, 1999.

[33] E. Marder-Eppstein, "costmap_2d," 2013. [Online]. Available: http://www.ros.org/wiki/costmap_2d

[34] K. Konolige and E. Marder-Eppstein, "navfn," 2013. [Online]. Available: www.ros.org/wiki/navfn

[35] J. Ferrin, R. Leishman, R. Beard, and T. McLain, "Differential Flatness Based Control of a Rotorcraft For Aggressive Maneuvers," in *IEEE Int. Conf. Intelligent Robots and Systems*, 2011.

[36] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft*. Princeton University Press, 2012.

[37] M. Cummins and P. Newman, "FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, June 2008.

[38] ——, "Highly scalable appearance-only SLAM - FAB-MAP 2.0," in *Proc. of Robotics: Science and Systems*, Seattle, USA, June 2009, pp. 1–8.

[39] J. Macdonald, "Efficient Estimation for Autonomous Multi-Rotor Helicopters Operating in Unknown, Indoor Environments," Ph.D. dissertation, Brigham Young University, Nov. 2012.

[40] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A General Framework for Graph Optimization," in *IEEE Int. Conf. on Robotics and Automation*, Shanghai, May 2011, pp. 3607–3613.