Faculty Publications

2016-05-15

Relative Navigation in GPS Degraded Environments

Timothy McLain

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

David Wheeler

Brigham Young University - Provo, dowheeler@gmail.com

Paul W. Nyholm

Department of Mechanical Engineering, Brigham Young University

Daniel P. Koch

Department of Mechanical Engineering, Brigham Young University

Gary J. Ellingson

Department of Mechanical Engineering, Brigham Young University, gellings13@gmail.com

Follow this and additional works at: <https://scholarsarchive.byu.edu/facpub>

next page for additional authors
Part of the [Mechanical Engineering Commons](#)

Original Publication Citation

Wheeler, D., Nyholm, P., Koch, D., Ellingson, G., McLain, T., Beard, R. Relative Navigation in GPS Degraded Environments, *Encyclopedia of Aerospace Engineering: UAS*, Wiley, May 2016, DOI: 10.1002/9780470686652.eae1154.

BYU ScholarsArchive Citation

McLain, Timothy; Wheeler, David; Nyholm, Paul W.; Koch, Daniel P.; Ellingson, Gary J.; Lewis, Benjamin J.; and Beard, Randall W., "Relative Navigation in GPS Degraded Environments" (2016). *Faculty Publications*. 1894.

<https://scholarsarchive.byu.edu/facpub/1894>

This Book Chapter is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Faculty Publications by an authorized administrator of BYU ScholarsArchive. For more information, please contact ellen_amatangelo@byu.edu.

Authors

Timothy McLain, David Wheeler, Paul W. Nyholm, Daniel P. Koch, Gary J. Ellingson, Benjamin J. Lewis, and Randall W. Beard

Relative Navigation in GPS-Degraded Environments

David O. Wheeler

Mechanical Engineering Department, Brigham Young University, Provo UT, USA

Paul W. Nyholm

Mechanical Engineering Department, Brigham Young University, Provo UT, USA

Daniel P. Koch

Mechanical Engineering Department, Brigham Young University, Provo UT, USA

Gary J. Ellingson

Mechanical Engineering Department, Brigham Young University, Provo UT, USA

Benjamin J. Lewis

Electrical Engineering Department, Brigham Young University, Provo UT, USA

Timothy W. McLain

Mechanical Engineering Department, Brigham Young University, Provo UT, USA

Randal W. Beard

Electrical Engineering Department, Brigham Young University, Provo UT, USA

1 INTRODUCTION

As processing, sensing, and battery technologies continue to develop there are increased opportunities for unmanned air vehicles (UAVs) to contribute to society. Emerging applications include fire surveillance, search and rescue, infrastructure and agriculture monitoring, and the delivery of medical supplies to remote locations [1, 2, 3, 4]. However, the majority of these applications require additional technology development and will likely be restricted to less populated environments. The integration of autonomous aircraft into mainstream life will depend in large part on the ability to safely and effectively operate in varied environments and with varied tasking. For example, current autonomous systems typically require external sensing or computation, such as a motion capture system, Global Positioning System (GPS) localization, a priori maps of the environment, or at least offboard sensor fusion and decision making. Other systems are accompanied with strong, limiting assumptions, such as a highly structured environment (vertical walls, flat floors, stationary scenes, etc). Small UAVs are also limited by size, weight, and power (SWaP) constraints. While autonomous flight is currently possible in specialized circumstances, the development of robust, real-time, onboard methods for autonomous control in cluttered, non-structured environments without external or a priori information remains an open field of research.

A noted obstacle in reaching the navigation robustness necessary for the integration of UAVs in the national airspace is the heavy reliance on GPS. In 2010 the United States Joint Chief of Staff, Norton Schwartz, stated “It seems critical to me that the Joint force should reduce its dependence on GPS-aided precision navigation and timing, allowing it to ultimately become less vulnerable, yet equally precise, and more resilient” [5]. GPS not only provides global position estimates to constrain the drift introduced by noisy rate sensors like MEMS-based inertial measurement units (IMUs) but also provides a way to estimate ground speed and orientation, allowing the vehicle to estimate wind effects. A robust navigation solution cannot depend on accurate GPS measurements due to the varied sources of uncertainty presented in Table 1. In particular, GPS is unreliable indoors, when shadowed by buildings or foliage, or in the presence of GPS jammers. Various sources report that GPS loss, even for a brief period, often results in catastrophic failure. As a result, GPS-denied navigation has become a strong emphasis of research over the last decade.

Table 1: Sources of GPS Uncertainty

Type	Description
Multi-path	Signal bounces before reaching receiver (false psuedo-range).
Number of satellites	Few visible satellites increase sensitivity to timing errors.
Dilution of precision	Visible satellites are poorly spaced.
Spoofing	Signal is locally recreated with false information.
Atmospheric delays	Signal is delayed due to ionsphere and troposphere influences.

Figure 1: The multirotor vehicle used for the example implementation in this paper. Details on the specific hardware are given in Table 2

One important approach to GPS-denied navigation is known as Simultaneous Localization and Mapping (SLAM). This approach involves estimating the vehicle’s state relative to a local frame by creating or adding to a map and localizing the vehicle within that map. This method allows for the direct use of relative measurements, such as visual odometry (VO), to estimate the vehicle’s state. This article will serve as a tutorial outlining the basic components of a SLAM-based relative navigation solution. Although the framework is applicable to other airframes and implementations, several of the relative navigation framework modules have been implemented on a multirotor aircraft and are presented as examples. Figure 1 and Table 2 describe the multirotor platform used in these implementations. Section 2 is an overview of the framework while Sections 3 and 4 explain in detail the components of the relative front end and global back end. Conclusions are presented in Section 5.

2 RELATIVE NAVIGATION FRAMEWORK

Just as the driver of an automobile is the safest as he or she focuses on the road ahead of them rather than on their map, GPS, or communication devices, the relative navigation approach (see Figure 2) uniquely decouples the relative, in-flight control from less critical global updates. Sections 2.1 and 2.2 outline the relative navigation framework, while Section 2.3 presents several scenarios that prove challenging for

Table 2: Hardware Details

Component	Description
Vehicle	Mikrokopter Hexacopter XL
Autopilot	Flight-Ctrl V2.1 ME
RGB-D Camera	ASUS Xtion Pro Live
IMU	MicroStrain® 3DM-GX3®-15
Ultrasonic Altimeter	LV-MaxSonar®-EZ3
Processor	Intel Core i7-2710QE (2.1GHz $\times$ 4)

Figure 2: Relative navigation framework. The flight-critical elements of local estimation, path planning, and control are decoupled from global measurements. Odometry estimates, loop closures, and GPS measurements are optimized to form a globally consistent map making high-level missions possible.

Figure 3: 2D illustration of estimated and true states relative to the current node frame. Roll and pitch (not shown) are inertially defined because the node frame z -axis is aligned with gravity. Position (p_x, p_y, p_z) and yaw (ψ) are defined relative to the current node frame, k . When a new node frame is declared, the estimated p_x, p_y, p_z , and ψ are saved as an edge in the global back end pose graph. Node frame $k + 1$ is defined as the current true state (with the roll and pitch removed). As a result, the filter zeros p_x, p_y, p_z , and ψ and their corresponding covariance values. The state error and uncertainty are now removed from the front end and delegated to the global back end.

non-relative frameworks.

2.1 Relative Front End Overview

The core of the front end is a state estimation scheme (Section 3.2) that fuses sensor data, generally from a high rate IMU and infrequent exteroceptive sensors, such as an altimeter or a camera for visual odometry (Section 3.1). The state is estimated relative to a local, gravity-aligned coordinate frame (within about one meter for a multirotor vehicle) known as a node frame. As the vehicle moves, new node frames are established and the transformation between node frames is stored as an edge (see Figure 3). A local path planner (Section 3.3) uses sensor data for obstacle avoidance and route planning and provides inputs to the aircraft’s control (Section 3.4).

2.2 Global Back End Overview

Decoupled from the flight-critical front end is the need for a globally accurate map. The back end, represented as a pose graph, is seeded with the node frames and edges created by the front end (Section 4.1). To eliminate accumulated drift error, place recognition algorithms efficiently compare the current

image with all previous images (Section 4.2). When the vehicle returns to a previously visited location, known as a loop closure, an additional constraint is introduced to the set of edges, thereby overdefining the map. Optionally, if GPS measurements are available, they can be added in the back end as constraints (Section 4.3). As desired, a robust optimization step minimizes constraint errors introduced by loop closures, GPS information, and odometry estimates (Section 4.4). With these refinements, the global map is then used by the high-level planner to fulfill global missions through a series of relative goals (Section 4.5). Subsequent loop closures are used to refine and improve the global map, making it sufficient for persistent, repeatable navigation. This global framework only interacts with the aircraft by influencing what low-level, relative goals are introduced, providing safer, more robust control in GPS-degraded environments.

2.3 Motivating Scenarios

While working with respect to a single, inertial reference frame makes intuitive sense, the following scenarios highlight advantages to be gained from a relative framework. In each example the UAV is assumed to be controlled with respect to its globally estimated position:

- A UAV loses GPS signal and receives IMU measurements only for several minutes. Upon reacquiring GPS, the state estimate jumps drastically and the plane is unable to recover.
- A multirotor vehicle moves from indoors to outdoors. Upon acquiring GPS signal for the first time its global state estimates, with respect to an arbitrary origin inside the building, will jump drastically.
- After a loop closure, a UAV's current estimated global state may jump significantly resulting in sudden, unintentional, and unpredictable vehicle motion.
- After flying for some time the size of the optimization problem delays any updates to the UAV's estimated global state. The vehicle's control suffers as a result.
- A vehicle receives an erroneous loop closure or GPS measurement. The estimated global state degrades without a method to later remove the effects of the outlying measurement.

3 RELATIVE FRONT END

The principal components of the relative front end, as introduced in Section 2.1, are explained in greater detail below:

3.1 Visual Odometry

Visual odometry (VO) is the process of computing the motion of a camera by comparing the captured images [6]. VO algorithms fall into two general categories: appearance-based and feature-based. Appearance-based methods use information from all pixels in the images to compute motion, while feature-based methods use visually distinct features in the environment, such as corners, that are tracked from one image to the next. VO can be implemented using either monocular or stereo cameras, and more recently has been implemented using RGB-D cameras (e.g. [7]). In addition to providing a standard color image, RGB-D cameras, like the Microsoft Kinect sensor, provide a depth image that encodes how far the object imaged at each pixel is from the camera. With monocular VO the change in orientation of the camera can be computed, but the translation can be computed only up to an unknown scale factor. The depth information provided by stereo and RGB-D cameras enables the calculation of this scale factor.

Some VO algorithms compute the camera's motion between consecutive images while others find the transformation to a chosen reference image, referred to as a key frame. While most VO algorithms could be used in the relative navigation framework, a key-frame approach fits best with the pose graph SLAM paradigm. A new key frame is chosen once the vehicle has moved far enough that there is insufficient overlap between features in the current and key frame images. The key frame approach results in low drift in the VO estimates, especially for a hovering multirotor vehicle.

Good general tutorials on visual odometry are found in [8, 9]. Additional approaches for VO using RGB-D cameras include those presented in [7, 10], and a comparison between various methods can be

found in [11]. Approaches utilizing laser scanners have also been widely explored in the literature (e.g. [12, 13]), and techniques using point clouds from LiDAR or other sensors have been explored as well (e.g. [14]).

3.2 Estimation

To maintain stable flight, a filter must fuse available sensor data to provide robust estimates for attitude and velocity. Furthermore, many high-level goals require position to be estimated reliably. Several methods have been developed to fuse a motion model with intermittent measurements for vehicle state estimation, including the complimentary filter and the particle filter. However, the most prevalent estimation approach for UAV platforms is the Kalman filter. While the following description outlines a Kalman filter approach, any estimation scheme could be used.

What most distinguishes the relative navigation approach from its conventional global counterparts is the decoupling of local state estimation from global states. Position and yaw states are relative to a local, gravity-aligned node frame. As illustrated in Figure 3, state estimates will drift from truth with the uncertainty estimated by the Kalman filter’s covariance matrix. After moving a small distance, the current true state (with pitch and roll removed) is declared to be the origin of a new relative coordinate frame. The estimated position (p_x, p_y, p_z) and yaw (ψ) are saved as an edge in the global back-end pose graph as the estimated transformation between node frames. These states are then replaced with states relative to the new node frame. Since the position and yaw states with respect to the new coordinate frame are now exactly zero, the associated covariance values are set to zero. This process can be thought of as augmenting and subsequently marginalizing the filter’s states. The accumulated error and its accompanied uncertainty is effectively removed from the front-end filter and delegated to the back-end pose graph.

A multiplicative extended Kalman filter (MEKF) implementation of this relative estimation framework is presented in [15] and [16]. The filter estimates relative position, attitude (with relative yaw), body-fixed velocity, and gyroscope and accelerometer biases. The attitude of the vehicle is represented by a quaternion. While quaternions require four scalars to define a three degree of freedom rotation, they are computationally more efficient than Euler angles and avoid the singularity known as gimbal lock. When using quaternions however, the estimated attitude error, found in the typical update step of an extended Kalman filter (EKF), cannot simply be added to the current state estimate. A common approach is to multiply the attitude quaternion by the estimated attitude error, maintaining the quaternion norm and earning the term multiplicative EKF [17]. The underlying dynamics of the filter make use of the typical kinematic relationships [18] coupled with an enhanced rotorcraft drag model presented in [19]. The velocity estimates are constrained by including, in conjunction with accelerometer measurements, drag terms that are proportional to the body-fixed forward and right velocities. In this implementation, gyroscope measurements are considered as inputs to the system (mechanization), while accelerometer measurements are used as updates. A feature-based VO for a forward-facing RGB-D camera, using FAST features [20] with BRIEF descriptors [21] was used (see [15] for details). Flight tests were performed using the hardware outlined in Table 2 and the MEKF described above. Figure 4 compares the MEKF’s estimated forward and yaw states with truth as measured by a motion capture system. The discontinuities occur as the vehicle transitions from one node frame to the next.

3.3 Low-level Path Generation and Following

The autonomous vehicle must be able to maneuver relative to the local environment without reliance on global state information. This is accomplished as local obstacles are determined from sensor information and goals with respect to the local node frame are received from the high-level path planner (Section 4.5).

For example, one approach when using a LiDAR or RGB-D sensor is to simplify the path generation problem onto the 2D horizontal plane at the elevation of the vehicle. Obstacles are identified from the most recent depth information and saved with respect to the current node frame. A 2D path is defined relative to the current node frame that progresses towards the goal while avoiding obstacles. Obstacles, goal locations, and the current path are transformed into each new node frame. A path following approach, such as the one described in [18], can be used to provide a desired state to the controller.

Figure 4: Performance plots for a MEKF implementation using the relative navigation framework. The estimated forward and yaw states (blue) are compared to truth (green). The discontinuities indicate a change in node frame.

Other approaches may be more appropriate given different vehicles and sensor information and may be extended beyond 2D. For example, if a fixed-wing aircraft is flying with a LiDAR then paths may be generated that account for the vehicle dynamics to avoid obstacles (e.g. [22]). This includes Dubins paths [18, 23] and vector fields for path following.

3.4 Control

Control approaches for both fixed-wing and multirotor vehicles have been widely explored in the literature and will not be reviewed in detail here. One of the key differences between a controller implementation for a relative framework to its global counterparts is that the desired states of the vehicle are expressed in the local node frame rather than in the global frame. However, because the actual state of the vehicle is also expressed in the local frame, the error between the actual and desired states will be the same as if both were expressed in the global frame. Therefore many controllers designed to work in a global framework will also work well within the relative framework.

The position controller used in the example implementation for this paper is adapted from the controller detailed in [24], while attitude stabilization and control is achieved using the standard PID loops implemented on the autopilot. The position estimation and control run at approximately 100Hz on the onboard computer, while the attitude estimation and control run at a higher rate on the autopilot. The outputs of the position controller are roll, pitch, yaw rate, and thrust set points that become the inputs to the attitude controller. This architecture is illustrated in Figure 5.

4 GLOBAL BACK END

The principal components of the global back end, as introduced in Section 2.2, are explained in greater detail below.

4.1 Pose Graph

To maintain a spatially, or globally consistent map, the back end stores information about the vehicle’s trajectory as a pose graph. A pose graph is a graphical representation of a vehicle’s path that encodes global vehicle pose estimates as graph vertices and the relative transformation between two poses as graph edges. A pose graph representation is effective because efficient graph optimization algorithms have been thoroughly developed and can be applied to refine pose estimates; additionally, the graph conveniently

Figure 5: Control system architecture for the example implementation. The position estimation and control run on the onboard computer, while attitude estimation and control run at a higher rate on the autopilot. In this figure $\mathbf{x}^c$ is the commanded position, $\mathbf{u}$ is the motor commands, and $(\phi^c, \theta^c, \dot{\psi}^c, T^c)^\top$ are the roll, pitch, yaw rate, and thrust commands sent from the onboard computer to the autopilot

serves as an abstract map that can be used for 3D and human-readable map construction as well as high-level path planning.

Figure 6 depicts how a pose graph is constructed from vehicle odometry. In a relative framework, a pose graph vertex represents the global estimate of a node frame’s translation and orientation. The graph edge connecting two consecutive poses encodes the relative transformation between two node frames and its respective covariance. The front-end estimator provides relative transformation edges to the back end. The transformations are compounded with all previous transformations to estimate the global position of the new node frame and a corresponding vertex is added to the graph.

A pose graph is also capable of encoding measurements other than odometry. Loop closure measurements introduce edges between two existing, non-consecutive nodes (Section 4.2). Similarly GPS measurements can be added as edges between the vertex where the measurement was received and the origin of the global coordinate system (Section 4.3). Since Lu and Milios’s seminal work in [25], pose-graph SLAM has become the predominate method of SLAM in the literature. An excellent tutorial on pose-graph SLAM is found in [26].

4.2 Place Recognition

A vehicle’s global position estimate will drift significantly over time in the absence of global updates. A common solution in the SLAM literature makes use of place recognition (PR). In place recognition the current image is compared with previous images to determine if the vehicle has returned to the same location. While a naive solution would require significant computation and memory, not generally available with the size, weight, and power (SWaP) constraints of a UAV, the computer vision community has developed efficient vocabulary-based PR algorithms. As with VO algorithms, distinct image features are represented by mathematical descriptors. Using a large, representative training dataset of images, the most prominent, distinct feature descriptors are saved offline and referred to as words in a vocabulary. During flight, any image can be succinctly represented by the set of nearest vocabulary words found in the image. Images can be quickly compared using word occurrences, similar to many search engine algorithms [27]. Further work has improved place recognition performance in the presence of aliasing, where high correlation is found on non-correlated images, like pictures of brick walls [28].

In a relative navigation approach, each key frame image (Section 3.1) is passed through the place recognition software and archived. After a match is found, the images are recalled and visual odometry methods provide the estimated transformation from one node frame to another. The transform is communicated to the pose graph as an edge between non-consecutive nodes, allowing for optimization and reduction of accumulated drift.

Figure 6: A loop closure edge (solid purple) over-constrains the back-end pose graph. After optimization, the estimated node frame states (orange) will more closely resemble truth (green). The tear drops represent graph vertices (node frames) while lines represent graph edges.

4.3 Intermittent GPS Integration

With optimization, odometry and loop closures can be integrated to maintain a globally-consistent, local map; the first node frame is considered to be the origin of a global coordinate system and the subsequent node frames are defined with respect to it. This type of map is useful for navigation but does not allow for high level path planning techniques such as navigating to desired GPS coordinates. To position the globally-consistent, local map on the earth’s coordinate frame, GPS measurements are incorporated into the back end.

A virtual zero is added to the graph to enable translation and rotation of the local map on the global plane [29]. The virtual zero can be thought of as the origin of the Earth’s global coordinate frame. A virtual constraint is added as an edge between the virtual zero vertex and any other vertex on the graph. The virtual constraint has infinite covariance, meaning that there is no certainty about the relative transformation encoded by the edge. Adding a virtual zero and virtual constraint to the graph allows the map to translate freely within the global coordinate system.

Figure 7 depicts the process of adding a GPS measurement to the pose graph. First, when a measurement is received, an odometry edge and vertex is added to the graph representing the location where the measurement was received. This edge comes from the front end’s current estimated state relative to the latest node frame. When the odometry has been established, an additional edge is created linking the virtual zero vertex and the newly created vertex. As with odometry and loop closures edges, the GPS edge encodes the relative transformation between the two vertices with its associated covariance. With GPS edges in place, the graph can be optimized to yield a globally consistent map that aligns with a known global coordinate system.

4.4 Map Optimization

Because of erroneous odometry measurements, graph optimization is critical to maintaining a globally consistent map. A key advantage of using a pose graph to represent a map is that it can easily be formulated as a least-squares optimization problem where poses and edges become free variables and constraints respectively. Least-squares optimization attempts to find the arrangement of poses that most likely results from a given set of odometry, loop closures, and GPS measurements. Many efficient graph optimization algorithms exist, some of which are popular open-source projects [30].

Figure 8 shows a pose-graph map before and after optimization. The data was gathered using the relative front end implementation described in Section 3.2 and the map was optimized with the popular

Figure 7: A virtual zero vertex is added to the pose graph and represents the origin of a global coordinate system. Edges between the virtual zero vertex and node frame vertices allow the map to be translated and oriented to its proper global location

Figure 8: Map before (left) and after (right) optimization. Odometry and loop closure edges are shown in gray and black respectively.

g2o library [31]. A multirotor aircraft was flown approximately 125 meters through a series of hallways, forming a loop. About 25 meters of overlap exist in the path. It is clear that prior to optimization, errors accumulated and the global pose estimate drifted from truth. Optimization mitigates these errors and returns a globally consistent map that matches the flight path.

Intuitively, pose graph optimization can be thought of a mass spring system settling in a state of least energy. In this analogy, graph vertices are masses and edges are springs. A spring's stiffness corresponds to its edge's covariance, becoming increasingly stiff as the covariance approaches zero. In a thought experiment, one can visualize that when a graph constructed of only odometry edges is optimized, the output is simply the original graph because there are no additional loop closure or GPS springs to deform the system. When loop closures are added to the graph, additional stiff springs are added between non-consecutive masses, resulting in a graph that more closely represents the true vehicle trajectory. Finally, in this example the virtual zero and virtual constraint can be thought of as an infinitely long string tying the mass-spring system to the origin of the global coordinate system, as shown in Figure 9. This establishes a reference to the global coordinate system and allows the mass system to translate about the origin. GPS edges are then added as springs between the virtual zero and corresponding masses, pushing the map into its appropriate absolute position.

One drawback of least-squares optimization is that it is inherently sensitive to outliers. This means that false positive loop closures and erroneous GPS measurements have catastrophic effects on graph optimization. To compensate for these effects, dynamic covariance scaling (DCS) [32], a variation of least-

Figure 9: Illustration of mass spring optimization example. A virtual zero is established as the origin of the global coordinate frame and is tied to the mass spring system. GPS edges are added as springs between the virtual zero and the poses where a measurement is received.

squares that is robust to outliers, can be used. DCS can be thought of as giving no certainty to edges that are deemed as outliers. By using DCS one is able to produce globally consistent maps even in the presence of erroneous loop closures and GPS measurements. Other methods exist for making graph optimization robust to outliers [33, 34, 35, 36]. A comparison of several of these methods is given in [37].

4.5 High-level Path Planning

The purpose of the high-level path planner is to transform global information, such as waypoints, into the current node frame. As the relative estimator transitions from one node frame to the next, the high-level path planner passes up-to-date relative goals to the low-level path planner. These relative goals are the only way that the global back end influences the front-end control, effectively isolating the UAV from the effects of jumps in the global state estimate due to optimization or global measurements. The high-level path planner can be used for autonomous waypoint following, exploration and mapping, target tracking, or landing.

5 CONCLUSION

The integration and widespread use of unmanned air systems for many applications depends heavily on the ability of these aircraft to operate in a safe and reliable manner, often in the presence of degraded GPS signals. A relative SLAM-based approach has been presented as a viable solution for robust navigation in GPS-degraded environments. By decoupling the real-time local estimation and control from the global position estimation, the vehicle is able perform essential tasks such as stabilization and obstacle avoidance without being dependent on consistent and accurate global estimates. A pose-graph based global back end allows techniques such as place recognition, loop closures, GPS integration, and map optimization to opportunistically improve global estimates. The separation between the relative front end and global back end also eliminates control issues that can arise when global estimates change as new information is incorporated. An example implementation has been demonstrated for a multirotor aircraft; however, the presented relative navigation framework is not implementation specific and could be adapted to other vehicles, sensors, and mission profiles.

REFERENCES

- [1] V. Ambrosia, S. Wegener, T. Zajkowski, D. Sullivan, S. Buechel, F. Enomoto, B. Lobitz, S. Johan, J. Brass, and E. Hinkley, “The Ikhana unmanned airborne system (UAS) western states fire imaging missions: from concept to reality (2006–2010),” *Geocarto International*, vol. 26, pp. 85–101, April 2011.

- [2] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey, "Supporting wilderness search and rescue using a camera-equipped mini UAV," *Journal of Field Robotics*, vol. 25, no. 1, pp. 89–110, 2008.
- [3] S. R. Herwitz, L. F. Johnson, J. C. Arvesen, R. Higgins, J. Leung, and S. Dunagan, "Precision agriculture as a commercial application for solar-powered unmanned aerial vehicles," *AIAA 1st Technical Conference and Workshop on Unmanned Aerospace Vehicles*, May 2002.
- [4] "Matternet - our vision." <http://www.matternet.us/our-vision>. Accessed: 2014-12-11.
- [5] N. Schwartz, "The united states as an aerospace nation: Challenges and opportunities," in IFPA Fletcher Conf. on National Security Strategy and Policy, Medford, MA, 2010.
- [6] D. Nister, O. Naroditsky, and J. Bergen, "Visual Odometry," in *Proceedings of the International Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 652–659, 2004.
- [7] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Fox, and N. Roy, "Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera," in *International Symposium on Robotics Research (ISSR)*, 2011.
- [8] D. Scaramuzza and F. Fraundorfer, "Visual Odometry: Part I: The First 30 Years and Fundamentals," *IEEE Robotics & Automation Magazine*, no. December, pp. 80–92, 2011.
- [9] F. Fraundorfer and D. Scaramuzza, "Visual Odometry: Part II: Matching, Robustness, Optimization, and Applications," *IEEE Robotics & Automation Magazine*, no. June, pp. 78–90, 2012.
- [10] J. Zhang, M. Kaess, and S. Singh, "Real-time depth enhanced monocular odometry," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 4973–4980, Sept. 2014.
- [11] Z. Fang and S. Scherer, "Experimental study of odometry estimation methods using RGB-D cameras," *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 680–687, Sept. 2014.
- [12] S. Grzonka, G. Grisetti, and W. Burgard, "Towards a Navigation System for Autonomous Indoor Flying," in *2009 IEEE International Conference on Robotics and Automation*, no. Section III, pp. 2878–2883, 2009.
- [13] A. G. Bachrach, S. Prentice, R. He, and N. Roy, "RANGE – Robust Autonomous Navigation in GPS-Denied Environments," *Journal of Field Robotics*, vol. 28, no. 5, pp. 644–666, 2011.
- [14] W. L. D. Lui, T. J. J. Tang, T. Drummond, and W. H. Li, "Robust egomotion estimation using ICP in inverse depth coordinates," in *2012 IEEE International Conference on Robotics and Automation*, pp. 1671–1678, Ieee, May 2012.
- [15] R. C. Leishman, T. W. McLain, and R. W. Beard, "Relative navigation approach for vision-based aerial GPS-denied navigation," *Journal of Intelligent and Robotic Systems*, vol. 74, no. 1–2, pp. 97–111, 2014.
- [16] R. C. Leishman and T. W. McLain, "A Multiplicative Extended Kalman Filter for Relative Rotorcraft Navigation," *IEEE Transactions on Robotics*, 2013.
- [17] E. Lefferts, F. Markley, and M. Shuster, "Kalman Filtering for Spacecraft Attitude Estimation," *Control Systems Magazine, IEEE*, vol. 5, no. 5, pp. 417–429, 1982.
- [18] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012.
- [19] R. C. Leishman, J. C. Macdonald, R. W. Beard, and T. W. McLain, "Quadrotors and Accelerometers: State Estimation with an Improved Dynamic Model," *Control Systems Magazine, IEEE*, vol. 34, no. 1, pp. 28–41, 2014.
- [20] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Computer Vision - ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3951 of *Lecture Notes in Computer Science*, pp. 430–443, Springer Berlin Heidelberg, 2006.
- [21] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Computer Vision - ECCV 2010* (K. Daniilidis, P. Maragos, and N. Paragios, eds.), vol. 6314 of *Lecture Notes in Computer Science*, pp. 778–792, Springer Berlin Heidelberg, 2010.
- [22] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty*," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 723–730, IEEE, 2011.
- [23] M. Owen, R. W. Beard, and T. W. McLain, "Implementing dubins airplane paths on fixed-wing uavs*," *Handbook of Unmanned Aerial Vehicles*, pp. 1677–1701.

- [24] J. Ferrin, R. C. Leishman, R. W. Beard, and T. W. McLain, "Differential flatness based control of a rotorcraft for aggressive maneuvers," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2688–2693, Ieee, Sept. 2011.
- [25] F. Lu and E. Milios, "Globally Consistent Range Scan Alignment for Environment Mapping," *Auton. Robots*, vol. 4, pp. 333–349, Oct. 1997.
- [26] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard, "A Tutorial on Graph-Based SLAM," *IEEE Intell. Transp. Syst. Mag.*, vol. 2, no. 4, pp. 31–43, 2010.
- [27] J. Sivic and A. Zisserman, "Video google: Efficient visual search of videos," vol. 4170, pp. 127–144, 2006.
- [28] 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, 2008.
- [29] "Global pose estimation with limited GPS and long range visual odometry," *2012 IEEE Int. Conf. Robot. Autom.*, pp. 627–633, May 2012.
- [30] C. Stachniss, U. Frese, and G. Grisetti, "OpenSLAM," 2014.
- [31] R. Kuřmmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "g2o: A General Framework for Graph Optimization," in *IEEE Int. Conf. Robot. Autom.*, pp. 3607–3613, May 2011.
- [32] "Robust map optimization using dynamic covariance scaling," *2013 IEEE Int. Conf. Robot. Autom.*, pp. 62–69, May 2013.
- [33] G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Robust pose-graph loop-closures with expectation-maximization," *2013 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 556–563, Nov. 2013.
- [34] Y. Latif, C. Cadena, and J. Neira, "Robust loop closing over time," *Robotics*, 2013.
- [35] E. Olson and P. Agarwal, "Inference on networks of mixtures for robust robot mapping," *Int. J. Robot. ...*, 2013.
- [36] N. Sunderhauf and P. Protzel, "Switchable constraints for robust pose graph SLAM," *2012 IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, pp. 1879–1884, Oct. 2012.
- [37] N. Sunderhauf and P. Protzel, "Switchable constraints vs. max-mixture models vs. RRR - A comparison of three approaches to robust pose graph SLAM," *2013 IEEE Int. Conf. Robot. Autom.*, pp. 5198–5203, May 2013.