2016 International Conference on
Unmanned Aircraft Systems (ICUAS)
June 7-10, 2016. Arlington, VA USA

A Framework for Visual Return-to-Home Capability in GPS-denied Environments

Benjamin P. Lewis[1] and Randal W. Beard[2]

Abstract— **This paper addresses the problem of navigating an Unmanned Aerial Vehicle in an environment where GPS quality is degraded or lost. A method is described to return the aircraft from the GPS-degraded region to its launch point by means of visual navigation techniques. We record a string of overlapping images, or keyframes, on the outbound flight. On the return journey, a control law based on the homography between images is used to bring the aircraft back over each keyframe. A complementary filter is used to smooth the path defined by the homography control. An implementation and results in simulation are described and further work with a physical aircraft is proposed.**

I. Introduction

In order to navigate effectively, Unmanned Aerial Vehicles (UAVs) rely on an accurate estimate of their location at all times. Location data is commonly provided by the GPS system, which is an easy and fairly reliable source. This has led to widespread use of GPS in the unmanned, commercial, and military aviation industries. However, there are circumstances which can cause a GPS signal to degrade or fail, such as extreme terrestrial or solar weather, multipath signal effects, signal occlusion, or malicious attacks [1].

The reliance on GPS in current state-of-the-art aerial technology is a potential weakness. In the event of a degraded or lost GPS signal, the aircraft state estimate can quickly develop large uncertainties, rendering navigation ineffective. Often, this means that an expensive aircraft is unrecoverable. At best, an aircraft that is equipped with a streaming camera and appropriate sensors can be manually returned by the intervention of a remote pilot.

There is not an obvious and universal solution for this problem. Much work has been done in recent years to create viable GPS-free solutions to estimate vehicle position [2], [3], [4], [5], [6], [7]. These solutions often incorporate a small onboard computer with sensors such as cameras, lidar, radar, and laser altimeters to help localize the aircraft and maintain its position. Accurate information from these sensors delays divergence of the calculated vehicle state until a GPS signal can be reacquired.

In this paper, we describe a framework which allows an aircraft to return to its launch point if it loses GPS positioning information. This function is of interest as autonomous aircraft become commonplace and begin to operate heavily within cities, where multipath effects and signal occlusion can degrade GPS. We demonstrate, in simulation, a complete return-to-home system. This functionality could also be used to return the UAV to non-GPS-denied airspace, such as when exiting an urban canyon, and allow the aircraft to continue the mission via an alternate route with a GPS signal.

(a) Outbound trip, capturing images (b) Loss of GPS; begin return journey (c) Return journey following image trail

Fig. 1: The return-to-home system use case. The green area is GPS-degraded airspace. When GPS is lost, the aircraft follows a series of images back to its launch point.

In brief, the system functions as follows: The aircraft uses an onboard downward-facing camera to capture imagery of the ground on its outbound flight, which it uses to identify and save a sequence of overlapping images of its flight path. Upon loss of GPS signal, the aircraft may hover (in the case of a multirotor vehicle) or loiter as it waits to recover GPS signal. When it determines that it does not have sufficient fuel to complete its mission, it uses computer vision techniques to follow the sequence of images back to its launch point. (See Fig. 1.)

This return-to-home framework has several advantages over the current aircraft recovery techniques. The system can be made fully autonomous, eliminating the need of a remote pilot to take over and return the aircraft. The return path is virtually guaranteed to be free of obstacles, since it was traversed minutes earlier. There is no reliance on a global estimate of state from GPS systems. All navigation guidance is determined entirely by onboard equipment, making the system resistant to outside interference.

This paper is structured as follows: Section II provides a review of work that has been performed in GPS-degraded UAV navigation, visual navigation, and other related topics. Section III discusses the theory of homography-based control, which is the basis of this work. Section IV describes the system in detail. Section V discusses simulation results and assumptions used in the simulation. Section VI addresses planned future work, including flight tests.

*This work was supported by the NSF Center for Unmanned Aircraft Systems

[1]Benjamin Lewis is a graduate student in Electrical Engineering, Brigham Young University, Provo, UT 84604, USA benjamin.lewis.1000@gmail.com

[2]Randal Beard is a professor in the Department of Electrical Engineering, Brigham Young University, Provo, UT 84604, USA beard@byu.edu

978-1-4673-9333-1/16/$31.00 ©2016 IEEE

II. Related Work

There has been much research in non-GPS based navigation schemes for UAVs and ground robotics in recent years. These include algorithms for fusing vision data with inertial sensors [2], matching air and ground image data [3], terrain [4] and horizon matching [5] navigation, SLAM-based approaches [6], and computation of position and orientation (pose) based on visual data [7]. These approaches help make drones and robots more reliable and operational in a wider variety of circumstances.

Work has been done in estimating aircraft and ground robot location using widely available satellite imagery and vision processing. With this technique, a large aerial image of the area, supplemented with exact global position information, is loaded into an onboard computer. As the robotic system moves, the point of view from its camera is matched to the ground image to determine its global position. In [3], imagery captured from a ground-based robot with a panoramic camera was warped to provide a birds-eye view of the ground. The warped image was then matched with data from Google Earth to accurately localize the robot.

Accurate inertial measurement unit (IMU) data has a positive effect on the accuracy of vehicle state estimation, which potentially can reduce the reliance on GPS. One method of increasing this accuracy is supplementing IMU data with homography-based visual state estimation [2]. In another study, a series of overlapping images was used to reconstruct the position and orientation of the aircraft. The homography between images was decomposed and fed into a filter to estimate the position in simulation [8]. Indoor flight navigation using LIDAR and vision processing has also been studied recently. The additional sensors provide additional information for pose estimation [9]. An impressive monocular pose estimation algorithm for UAVs was shown in [10].

Our approach to the return-to-home problem is a modified application of eye-in-hand visual servoing. The visual servoing problem involves using vision processing to move a camera mounted on a robotic arm into a desired position. This is accomplished by visually estimating the pose of the arm and determining the movements needed to position the arm correctly. The control law can be defined either directly from image comparison or by computing the arm position in real space using the homography. These two approaches are respectively called image-based and position-based visual servoing. Of the two approaches, image-based servoing is generally more robust to errors in robot or camera calibration [11]. In [12], a hybrid control strategy that switches between position- and image-based visual servoing control was proposed. The control was demonstrated on an eye-in-hand robot to great effect.

Most position-based visual servoing requires an accurate geometric model of the objects in the target image. The authors in [13] defined a visual servoing control system that is robust to inaccuracies in this geometric model. They estimate the camera displacement at each iteration of the control law, then use adaptive gains to compensate for model inaccuracies.

A. Contributions in this Paper

Eye-in-hand visual servoing has been performed many times in labs using a single target image. We extend this concept to UAV navigation by constructing a string of images that serve as visual servoing targets. We present a method for determining the string of keyframe images and, on the return journey, a method to switch between image targets to navigate the aircraft to its launch point.

In many visual servoing applications, such as [12] and [13], the problem of matching feature points is solved by simplifying the reference image into a set of multicolored, distinct dots on a planar surface. Feature point matching becomes unambiguous, allowing more attention to be devoted to other parts of the problem. Given the unstructured terrain over which UAVs must fly, however, computing and matching feature points in real time becomes a nontrivial challenge. In our approach, techniques such as comparing multiple possible matches for each point, detecting large numbers of features to minimize the effect of outliers, and using RANSAC while finding the homography are combined to obtain a high-quality homography estimate.

The return-to-home system in this paper represents an advance in autonomous aircraft recovery and can reduce the need for remote intervention in UAVs.

III. Theoretical Background

The computer vision and control theory concepts of the homography, homography decomposition, reprojection error, and the homography-based control law [14] are central to our approach.

A. Homography

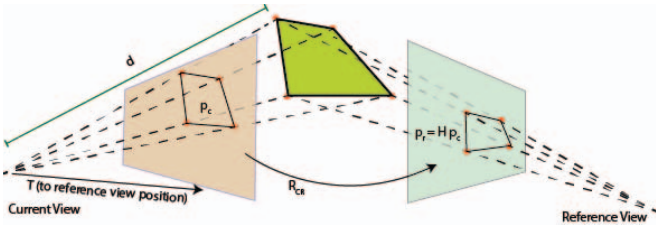


Fig. 2: A homography relates a set of physical points viewed by a camera from two different angles. The homography between the current image (left) and reference image (right) projects points in the current image onto the corresponding reference point as $p_r = \mathbf{H}_{proj} p_c$.

When a planar scene is viewed from two different locations by the same camera, the images from the camera are related mathematically by a 3x3 matrix called a homography. The homography is obtained by finding and matching a set of feature points that appear in both images. At least four sets of points are needed to compute the homography, three of which must be non-collinear. The homography matrix is computed by solving the matrix equation

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' & -x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' & -y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' & -x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' & -y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -y_3x_3' & -x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y_3' & -y_3y_3' & -y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -y_4x_4' & -x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y_4' & -y_4y_4' & -y_4' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = 0 \quad (1)$$

for all values of h_{ij}, where x_n and y_n are the x and y coordinates of a given image point n and x_n' and y_n' are the mathing points in the other image. The homography matrix is formed by reshaping the h vector:

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (2)$$

RANSAC (Random Sample Consensus) or LMEDS (Least Median of Squares) methods can be used to find the best homography that fits the greatest number of image points.

A homography found using image feature points is called a projective homography. These feature points correspond to physical points in the scene viewed by the camera, which in turn are related by a Euclidean homography. If the camera calibration matrix $\mathbf{K}$ is known, the Euclidean and projective homographies are related by:

$$\mathbf{H}_{euc} = \mathbf{K}^{-1}\mathbf{H}_{proj}\mathbf{K}. \quad (3)$$

Let C be the current location of the camera and R be the location of the camera when a reference image was taken, as shown in Fig. 2. The corresponding image planes are $\mathcal{C}$ and $\mathcal{R}$. Points in each image plane can be described as homogeneous points of the form $\{x_i, y_i, 1\}$, where x_i and y_i are the pixel locations of the point in the image plane i. The projective homography $\mathbf{H}_{proj}$ from $\mathcal{C}$ to $\mathcal{R}$, can be used to project a point in $\mathcal{C}$ to a point in $\mathcal{R}$ as follows:

$$p_r = \mathbf{H}_{proj}\, p_c \quad (4)$$

The projective homography can be found using OpenCV's `findHomography` function or similar methods [15]. In order to compute an accurate homography, care must be taken to find and match feature points accurately between the two images. Inaccuracies in the homography computation can be caused by bad feature matches, a lack of features, unevenly spaced feature points, outliers, and feature point distortion.

The projective homography maps the points used in its computation from one image to their corresponding points in the other image. An accurate homography does so with no little to no reprojection error.

B. Reprojection Error

The reprojection error is a metric of the accuracy of the homography computation. It is the geometric measurement of the pixel distance between a point projected by the homography into the corresponding image and the matched feature point in that image. For a projected point $\hat{p}_r = \mathbf{H}_{proj}p_c$ and measured point p_r, both normalized to homogeneous points of the form $\{p_{r_x}, p_{r_y}, 1\}$, the reprojection error e_r is:

$$e_r = \sqrt{(p_{r_x} - \hat{p}_{r_x})^2 + (p_{r_y} - \hat{p}_{r_y})^2} \quad (5)$$

.

Ideally, the reprojection error is zero and all points map exactly from one image to the other. In practice, noise, incorrectly matched points, and poorly distributed feature points will cause the reprojection error to be non-zero. An average reprojection error for the homography can be computed by averaging the reprojection errors of all pairs of matched points.

C. Homography Decomposition

For visual control, the Euclidean homography must be obtained from the projective homography. As described earlier, the Euclidean and projective homographies are fundamentally linked through the camera calibration matrix.

The Euclidean homography describes the physical relationship between the two camera positions C and R. The Euclidean homography from C to R, $\mathbf{H}_{proj}$ can be rewritten as

$$\mathbf{H}_{euc} = (\mathbf{R} + \frac{\mathbf{t}\mathbf{n}^T}{d}) \quad (6)$$

where $\mathbf{R}$ is the rotation matrix from frame C to frame R, $\mathbf{n}$ is the normal vector of the plane, $\mathbf{t}$ is the translation vector from C to R, and d is the distance from camera C to the planar surface in the direction of $\mathbf{n}$.

The distance d is the altitude of the aircraft for a downward-facing camera. We can extract $\mathbf{R}$, $\mathbf{n}$, and $\mathbf{t}$ from the Euclidean homography using analytical methods, such as the one described in [16]. This method gives 8 possible decomposition solutions. By making a few physical assumptions, such as both cameras viewing the plane from the same side and all reference points being in front of the camera, it is possible to reduce the size of the solution set to two.

D. Homography Control Law

A homography control law for image-based visual servoing is described in [14]. The control law is based on the homography and the center of gravity of the matched points. The center of gravity of feature points in the current image is reprojected into the reference image using the homography. A vector between the reprojected center of gravity and reference center of gravity is constructed by:

$$\mathbf{e}_v = \frac{\bar{p}_r^T \mathbf{H} \bar{p}_c}{\bar{p}_r^T \bar{p}_r} \bar{p}_r - \bar{p}_c \quad (7)$$

where $\bar{p}_r = \frac{1}{n}\sum_{i=1}^{n} p_{r_i}$ and $\bar{p}_c = \frac{1}{n}\sum_{i=1}^{n} p_{c_i}$. The center of gravity of the points in the reference and current frames,

$\bar{p}_r$ and $\bar{p}_c$ respectively, are homogeneous points represented as 3x1 vectors. $\mathbf{e}_v$ is the 3x1 control vector, which is shown by the paper's authors to be isomorphic to the camera pose. $\mathbf{e}_v$ can thus be used to control the platform to align with the reference image. Since computing a heading does not require the vector to have any given length, we normalize $\mathbf{e}_v$ in our approach and use it as a vector pointing in the correct travel direction. The vector $\mathbf{e}_v$ is weighted appropriately to provide control to the platform. The control function drives the platform to align with the reference image, and local stability is guaranteed [14].

IV. Methodology

In this section, we describe the general configuration of the system, which can be implemented easily in simulation or on an aircraft with an onboard computer. The control, navigation, and keyframe determination functions, which are the main ideas in this work, are also described in greater detail in this section.

A. System Configuration and Operation

The main parts of the return-to-home system are the camera input, homography computation, keyframe determination, and control and navigation functions. A flow diagram is shown in Fig. 3. System components are implemented in C++ and use the Robot Operating System (ROS) [17], a widely-used robotics framework, for inter-process communication. ROS uses a publisher-subscriber model with the effect that no individual module is dependent on any other model for anything except formatted data. By using ROS, the same software can be used with few modifications for simulation and on a physical system.

Many of the computer vision functions are implemented in C++ using OpenCV, a popular open-source computer vision library. The homography decomposition and control laws are written in C++.

Functionally, the main navigational information source for the system is the onboard, downward-facing camera. A gimbal may be used on a physical aircraft to keep the gimbal pointing down. After takeoff and reaching the target altitude, a signal causes the system to save the current image from the camera as the first keyframe. Each image after the keyframe then has its feature points computed and matched to the keyframe. The homography is computed between the keyframe and the current image. If it is detected that the homography quality is poor, then the current image is saved as the new keyframe. When the system enters return-to-home mode, this trail of keyframes becomes the keyframes to which the current image is compared, and a similar keyframe-switching method is used to determine when to start navigating relative to each previous keyframe (see section IV-C). When the system is inbound, the homography control data is fed to a complementary filter which provides navigational data (see section IV-D). This system is shown as a block diagram in Fig. 3.

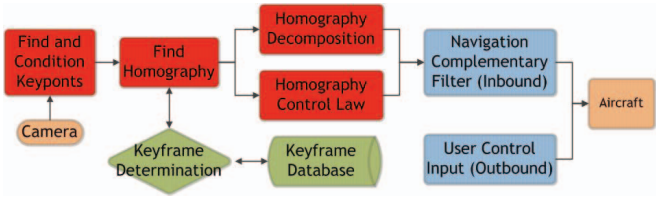


Fig. 3: A block-level diagram of the system.

B. Homography Computation

The homography calculation algorithm is designed to provide the most accurate homography possible in real time. To this end, feature point algorithms and homography computation algorithms were chosen for speed and number of feature points.

Feature points in each image are calculated using the ORB feature detection algorithm. ORB features were selected because a large number of feature points are produced in a short amount of time, the features are rotation-invariant, and the features are accurately matched by feature matchers available in OpenCV.

Feature points from the current and keyframe images are then matched using a brute force matcher. Two possible matches are found for each feature point, and a distance metric computed for each pair. If the ratio of the distances is too close to one, the feature points are considered to be too similar and the match is discarded. This helps prevent false matches, which would negatively impact the quality of the homography.

The matching feature points are then used to calculate a homography using OpenCV's findHomography. This method uses RANSAC to compute a homography for a set of points such that there is a 99% certainty that the homography is accurate.

C. Keyframe Determination and Switching

In this context, a keyframe is one of a series of overlapping images that create a visual trail leading from the launch point of the aircraft to the point where it turns around.

As the aircraft navigates along its route, it continually compares its current camera view to a keyframe and calculates the homography. However, after a time, the homography with the current keyframe starts to become ill-conditioned. We also want to ensure that some feature points in each keyframe are easily visible when the aircraft is using them for navigation on the inbound journey.

The method of creating a sequence of overlapping images solely from video feed is an open problem. In order for a keyframe to be useful in this context, it must satisfy two requirements: a sufficient number of feature points (at least four, preferably many more) must be shared between keyframes such that some feature points are visible on the return journey when switching keyframes, and it must be possible to compute a homography with a low average reprojection error at all times. When these conditions are successfully met on the outbound flight, it becomes much easier to smoothly navigate the inbound flight.

We use the constantly computed homography to satisfy both of these requirements on the outbound flight. First, the quality of each computed homography is assessed by computing its average reprojection error. When this error rises above a tunable threshold (20.0 pixels was chosen in simulation for a 640x640 pixel image), it is determined that the homography is ill-conditioned for navigation. In practical terms, this means that too many erroneous feature matches were not filtered out by RANSAC and that there is danger of losing the signal in the noise.

In many cases, the homography quality may be adequate but the area overlapped by the keyframe and current image may be shrinking. The overlap between two images is computationally expensive to detect directly, making this infeasible for a real time guidance system. However, we can use the homography to calculate an approximate pixel distance between the two images. The translation vector of the homography control law (7) computes a vector between the center of mass of the matched feature points in the two images. The scale of the vector is in pixels; thus, its length gives an estimate of the pixel offset between the two images. In our framework, we use this distance as another signal to switch to a new keyframe. We switch to a new keyframe if the pixel distance between images exceeds another tunable threshold. The higher the threshold, the further apart that keyframes will be; however there will be fewer feature points visible between one keyframe and the next. This threshold has a large effect on the performance of the system. In our simulation with a 640x640 pixel window, a threshold of 90 pixels led to occasional aircraft loss in well-featured environments, while a threshold of 40 pixels increased performance dramatically.

On the inbound journey, a similar switching technique is used. As the pixel offset between the two images falls below a threshold (30 pixels), it is determined that the aircraft has effectively reached the target keyframe and the next keyframe can be loaded for the next step of navigation.

The two keyframe switch signals are independent of each other but are both derived from the homography. When either signal occurs, the current image is saved to a database and used as the new keyframe. This approach is advantageous because it uses information that has already been calculated within the system (i.e. the homography and feature point locations), which adds very little additional computational overhead. In addition, by using the homography for both keyframe determination on the outbound journey and navigation on the inbound journey, we can ensure that the functional quality of the two functions is similar.

D. Inbound Navigation

At some point, the aircraft enters GPS-denied airspace and it is determined that the aircraft must turn around and return to its launch point. For the purposes of this paper, the turn-around decision is reached when a software signal is sent to the system.

Upon receiving the turn-around signal, the aircraft navigates toward its last keyframe. The navigational direction is derived using the homography control law and homography decompositions as control inputs to a complementary filter. The inputs and filter state are defined in the body-centered frame such that the complementary filter will always indicate a desired direction of travel relative to the aircraft (as opposed to a globally referenced heading.)

As a control input, the homography control law produces one vector that is isomorphic to the desired direction of travel. The decomposition produces two translation vectors, one of which is incorrect. In an ideal case, the normal component of the homography decomposition should remain almost constant for any well-conditioned homography computed with a given reference image. This makes it easy to tell which set of translation and rotation vectors is the correct decomposition. However, with the level of noise and imprecision in an unstructured visual environment, the decomposed normal vector varies widely between images, rendering this technique less effective.

The correct decomposed translation vector is determined by comparing each of the translations produced by the decomposition to the state of the complementary filter. The heading of each translation vector and of the filter state is computed, followed by the difference in heading between each translation vector and the filter state. The translation vector with a smaller angular difference in heading from the filter state is assumed to be the correct decomposition. The correct decomposition vector and the homography control law vector are then added into the complementary filter.

The control inputs are noisy in the short term, depending on the accuracy of the homography, but are more reliable on longer time scales. The complementary filter is used to smooth out these inputs and provide a smoothly time-varying control vector pointing in the direction of the keyframe. At each time step, the state of the filter and the inputs are normalized, such that the control vector points in the desired direction of travel and can be scaled to the desired speed of the aircraft.

The complementary filter is of the form

$$y = (1-\alpha)y + \alpha x \tag{8}$$

where y is the state of the filter, α is a weight parameter, and x is the new input. This function updates the heading of the filter with a normalized input weighted by α.

In general, the two input vectors are treated as independent inputs. As a result, the complementary filter runs twice for every homography computed. We use one function to compute α independent of the input type. The value of α in the complementary filter is a function of the reprojection error of the homography and the angle between the state of the complementary filter and the input vector. As the reprojection error increases, the homography becomes less well-conditioned; similarly, a vector pointing in a drastically different direction from the state of the filter is likely to be noise instead of useful data, since the direction to the reference frame will not change drastically over one sample period.

For each time step, α is

$$\alpha = 10^{\frac{-1|deg|}{90}} 10^{\frac{-|e_r|+1}{0.3}} \tag{9}$$

where deg is the angle in degrees between the filter state vector and the input vector and e_r is the reprojection error. The constants in the equation can be changed to allow quicker or slower response to changes in heading. In practice, we found that these constants gave enough freedom for legitimate heading changes, such as course corrections, without letting the state drift too much, which would induce jerky movement.

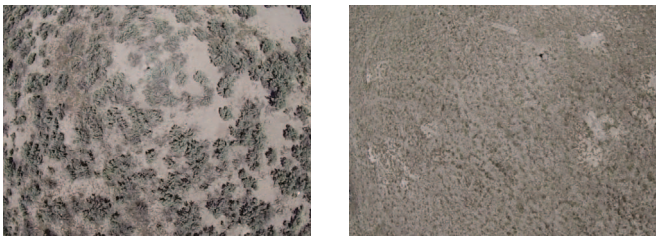
The state of the complementary filter cannot be continuous along the entire return journey. When a new keyframe is loaded from memory and set as the new destination, it is not generally the case that this keyframe lies in the same direction as the aircraft has been travelling. As such, the filter state must be allowed to change quickly to get a rough estimate of the new desired heading. Using a normalized state and input vector, we reset the state of the filter to (0,0) and update the filter for five iterations by cumulatively averaging the inputs.

E. System requirements

When implementing this system on a physical aircraft, software must be written that can fly the aircraft in the direction of the complementary filter. The open-source PX4 autopilot firmware [18] has functionality that allows an aircraft to be controlled by an onboard computer using ROS. Control algorithms can be written to interface with the autopilot. For a GPS-free solutions, an IMU-based attitude and heading reference system should be implemented to provide control feedback.

Furthermore, the aircraft should have an altitude control loop, since features that are visible in a lower-altitude image may become too small to detect at higher altitudes. To this end, the current altitude is saved when each keyframe is committed to the database. On the return trip, the aircraft returns to that altitude using PID control so that all features appear in the image at a similar scale.

For best results with the homography, that the aircraft should fly at an altitude such that the ground appears planar. This reduces motion blur and allows an accurate homography computation. Preliminary flight results have shown adequate feature tracking performance at an altitude of 2 meters over a flat surface. The aircraft also should operate in a fairly static yet feature-rich environment so that most image features do not move in the time between the outbound and inbound journey. Areas with high vehicular or pedestrian traffic, as well as non-static environments such as the ocean and featureless environments such as that shown in the second panel of Fig. 4. Nighttime conditions also preclude the effective operation of a traditional camera. However, the system still performs well in a wide range of environments, as shown in the next section.



(a) An environment with sufficient features

(b) An environment with insufficient features

Fig. 4: A comparison of environments in terms of feature availability. Imagery was taken in a preliminary test flight to help tune the system for both simulated and in-flight performance.

F. Advantages

The return-to-home system is useful in GPS-degraded environments because of its ability to autonomously return an aircraft to its launch point. It reduces the possibility of an aircraft being lost and reduces the need to have a remote pilot available to recover the aircraft. On the technical side, the ability to navigate off of keyframes limits the amount of data that must be stored on the aircraft. Further, determining when to save a new keyframe based on the homography increases the success rate of the inbound journey by ensuring that the keyframes are effective for homography navigation.

V. Results

In order to evaluate the effectiveness of the system, it was tested extensively in simulation. The simulator has propulsion and drag forces implemented. We are confident that the simulations are a good step in evaluating the system under certain conditions. The simulator is written in MATLAB and Simulink, and the Mathworks Robotics System Toolbox is used to connect the simulation to the ROS components of the system.

A. Simulation Setup

We created a simulation that mimics a real-world test case. We list here the assumptions and parameters that can be reasonably expected in an outdoor test flight. To test the performance of the system, we simulate the camera and aircraft nodes shown in Fig 3, while using the other elements of the system in the same configuration as they would be on a flight platform.

The simulation recreates a large visual environment in which the aircraft can fly, such as those shown in Fig. 7. It is worth noting that computer vision features are dependent on corners, edges, and gradients rather than on human-recognizable landmarks. In flight tests that were performed to evaluate sites for future flight tests, we found that natural environments such as those shown in subfigure 4a are sufficiently textured to provide feature points for homography computation. For comparison, a environment lacking sufficient visual information is shown in subfigure 4b.

The visual environments over which the aircraft flies are 5000x5000 pixel satellite images from Google Earth (see Fig. 7.) This image is subdivided into 100 equal parts on each side of the image. The simulated aircraft flies over this image plane. The camera is assumed to be gimballed straight down. The camera field of view is then constructed as a function of the x, y, and z location of the aircraft over the image, resulting in an area bounded by a quadrilateral on the ground plane image. This is the field of view of the camera. For reference, at an altitude of 10 units above the vision plane and a 65° camera angle of view in each direction, this field of view translates into a square approximately 12 units on a side. An example field of view projection is shown in Fig. 5 with the field of view geometry emphasized.

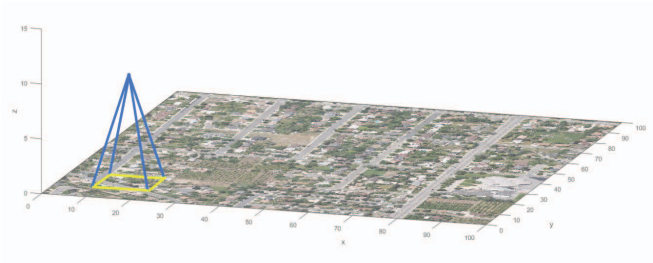


Fig. 5: A field of view projected from the aircraft position onto the visual environment.

The coordinates of the field of view corners can be used to transform that portion of the image environment into a 640x640 pixel image, which functions as the monocular camera input. When using the image transform MATLAB function, a non-square image ends up significantly distorted when viewed at different heading angles, which in turn distorts feature points and causes mismatch errors. For this reason, a square image was chosen in simulation; in general, this problem is non-existent with a physical camera.

The simulated camera view is published as a ROS topic. The simulated imagery data is processed at 7Hz on an Intel i7-2600 processor. However, preliminary follow-up tests with a hardware camera and real imagery show that the system can perform in real time on 30 FPS video with approximately 1500 unmatched feature points in each image. This means that the computational bottleneck in the simulation is computing and publishing the simulated image. Saving the keyframe images is assumed to be a trivial computational expense.

A camera intrinsic matrix must also be constructed for the simulated camera so that the homography decomposition can function. Using a focal length of 1700 pixels, assuming no image skew, and assuming that the principal axis of the camera runs through the center of the image, we obtain

$$K = \begin{bmatrix} 1700 & 0 & 320 \\ 0 & 1700 & 320 \\ 0 & 0 & 1 \end{bmatrix}. \tag{10}$$

The limitation of using a static image for simulation is that we cannot accurately simulate the effect of small scenery changes, such as a car moving, that occur after reaching a point in the outbound journey. However, RANSAC filtering of the keypoints is able to provide robustness to small changes in the environment when a large number of other features are static.

The aircraft is simulated in MATLAB and Simulink with force and drag dynamics in the x and y directions of the form

$$\ddot{x} = \frac{\mathrm{F}_x - sgn(\dot{x}) * \mathrm{C_D}\dot{x}^2}{m} \tag{11}$$

where $\ddot{x}$ is the acceleration in the x direction, $\dot{x}$ is the velocity in x, $\mathrm{C_D}$ is the coefficient of drag, sgn is the sign of the argument with $sgn(0) = 1$, and m is the mass of the aircraft. Acceleration in the y direction is of the same form. We set $\mathrm{C_D}$ to 0.7 and the mass as 3 kg. Forces applied to the system were normalized and scaled to a magnitude of 10N. This model was chosen because it roughly mimics the response of a multirotor aircraft to roll and pitch commands and can, when coupled with a suitable path-planning algorithm, follow a plausible fixed-wing trajectory. To aid the aircraft at the turn around point, velocity is zeroed out to prevent momentum from carrying it away from the keyframe before it can start its return journey.

In order to make the simulation applicable to both multirotor and fixed-wing aircraft, a path planning algorithm was implemented that enforces an effective maximum course deflection per second, which mimics a turn radius of a fixed-wing aircraft. In addition, course corrections are applied when approaching the boundaries of the image (outside of 10 to 90 units in the x or y direction) to turn the aircraft away from corners and edges and keep it over the visual field. When not on the edge of the visual area, the aircraft is allowed to change course by a random value bounded by ± 35 degrees per second of simulation time, which results in a path such as that shown in Fig. 6. The path planner is used to show that no bias for a particular flight path occurs in the simulations. By simulating a path with a minimum turning radius, we ensure that fixed-wing aircraft are able to use this system. A simpler application for multirotor vehicles only is to allow the aircraft to stop briefly on each keyframe to determine the correct heading of the next key frame; thus for multirotor aircraft, sharp course changes on the outbound journey would be allowed.

B. Simulations

The simulation was tested in the environments shown in Fig. 7. In each environment, 25 simulations were run with 150 simulation seconds of outbound flight time and at an altitude of 10 units above the field. By using the random path planner described above, we eliminated bias toward any given path. No attempt was made to detect loop closures; each time that the simulation passed over a location, it was saved off as a new keyframe.

The three simulation environments are chosen to show that the system works over a large variety of terrain and can handle seemingly repetitive environments such as forest as long as there are sufficient visual features.

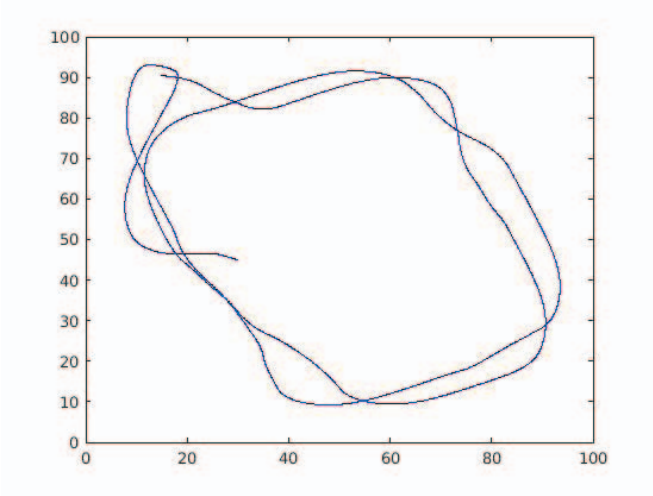


Fig. 6: A typical outbound path of the simulation path planner.

The average number of keyframes saved varied between environments. In the suburban environment, where the features are varied and there are few opportunities for mismatch, there were on average 80 keyframes saved off. The forest environment had approximately 100 keyframes per simulation run, while the field had approximately 130 keyframes in each successful run. A sample outbound and return journey for each environment is shown in Fig. 8.

In the field and forest environment, there were approximately 130 keyframes saved in each simulation run. The higher number of keyframes in these environments is due to the homography quality deteriorating more rapidly. In the forest, there are a large number of similar features, causing a large number of feature mismatches early on, causing rapid keyframe switching. In the field environment, fewer features lead to a poorly conditioned homography, again leading to faster keyframe switching.

In consequence of the low feature density in the field environment, there were more failed runs. Five runs failed in the lower right corner or the middle left side of the field environment while making a turn. As we can see in Fig. 7c, these areas are feature-poor due to fairly uniform texture, repeated patterns, and few lighting differences. However, the system was still able to navigate effectively much of the time despite poor features, due to momentum carrying the aircraft until it was able to identify features from its target keyframe.

In the city environment, two of the 25 simulations failed to return to home due to excessively sharp turns. In the forest environment, one simulation failed to turn around properly and two failed due to sharp turns. The inability to overcome sharp turns is an effect of the dynamics of the system, which carry the field of view out of sight of the target keyframe before it can correct its course; with a more accurate dynamics model, this problem can be avoided.

From these simulations, we can see that the return-to-home system performs very well in well-featured environments and successfully returns the aircraft to its launch point when the outbound path is appropriately chosen.

(a) A suburban environment. This was the primary environment used for development due to abundant features.

(b) A forest scene

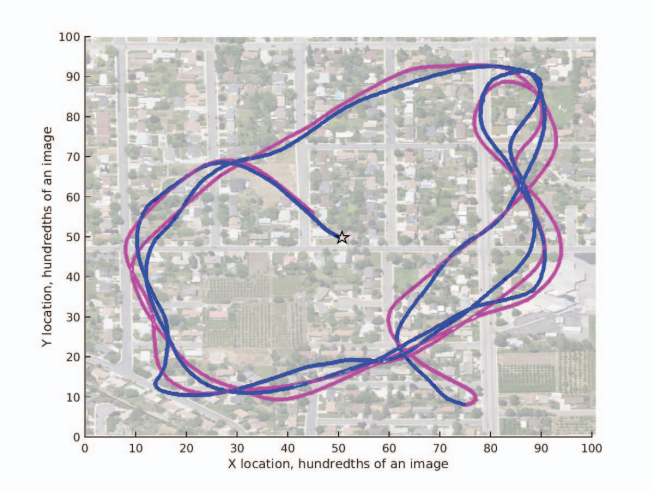
(c) An agricultural environment

Fig. 7: A few images used as the simulation environment. The original image sizes are 5000x5000 pixels. Satellite imagery from Google Earth, 2016.

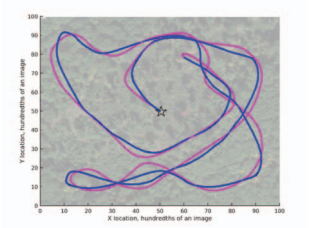
VI. Future Work

At the time of writing, the authors have begun flight testing implementations of this framework, including guidance controls as explained in IV-E. A X8 octocopter serves as the initial platform for proof-of-concept work. An onboard Gigabyte Brix i7 computer performs vision processing and guidance. The camera is a USB 3.0 IDS uEye camera with a global shutter to minimize motion blur. The computer will interface via Mavlink with a Pixhawk autopilot running the PX4 flight stack.

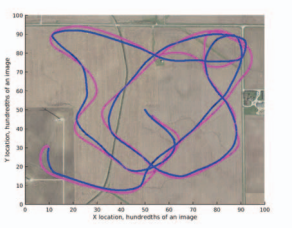
Preliminary data collection flight tests show that it is possible to extract return-to-home heading information from real imagery using the system described in this work. Work is ongoing to make the system more robust and to implement onboard attitude and heading control systems for a completely GPS-free solution.



(a) City out-and-back journey.



(b) Forest out-and-back journey.



(c) Field out-and-back journey.

Fig. 8: A sample of results of the return-to-home simulation. The blue path is the outbound path and the magenta path is the inbound path. The paths have been emphasized and superimposed on the visual terrain used.

On the algorithm side, work can be performed in planning paths that avoid feature-poor areas in order to increase system robustness in low-feature environments.

VII. Conclusion

In this paper, we have presented a system that grants return-to-home capability to a camera-equipped aircraft. This system is potentially of use in the growing field of autonomous aviation.

The simulation results presented in this paper are encouraging. The next step is to implement the system on a multirotor aircraft and perform tests in real environments. We anticipate that some of the specific implementation details, such as parameters in the equations, will need to change to accommodate the situation, but we are confident that the fundamental approach will be unchanged. After the concept has been proved with a multirotor aircraft, we plan to transition the system to use on a fixed-wing aircraft. This will involve changing the system to accommodate the dynamics of a fixed-wing platform.

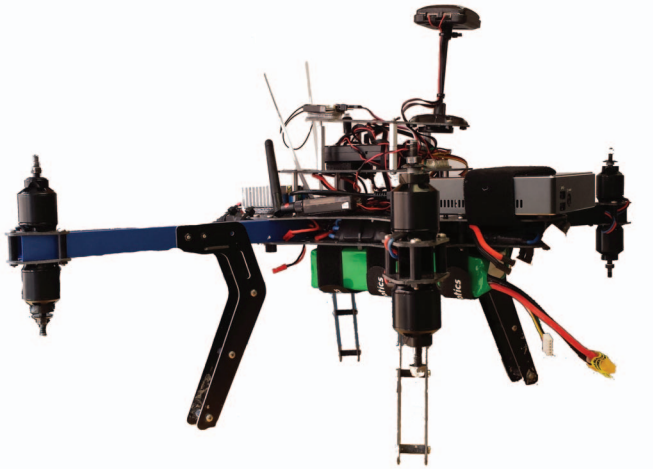


Fig. 9: The test aircraft, shown without rotors.

References

[1] John A. Volpe National Transportation Systems Center, "Vulnerability assessment of the transportation infrastructure relying on the global positioning system: Final report," Office of the Assistant Secretary for Transportation Policy, U.S. Department of Transportation, Tech. Rep., August 2001.

[2] A. Eudes, P. Morin, R. E. Mahony, and T. Hamel, "Visuo-inertial fusion for homography-based filtering and estimation." in *IROS*. IEEE, 2013, pp. 5186–5192. [Online]. Available: http://dblp.uni-trier.de/db/conf/iros/iros2013.htmlEudesMMH13

[3] A. Viswanathan, B. R. Pires, and D. Huber, "Vision based robot localization by ground to satellite matching in GPS-denied situations," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*, 2014, pp. 192–198. [Online]. Available: http://dx.doi.org/10.1109/IROS.2014.6942560

[4] D. Lee, Y. Kim, and H. Bang, "Vision-based terrain referenced navigation for unmanned aerial vehicles using homography relationship," *Journal of Intelligent & Robotic Systems*, vol. 69, no. 1-4, pp. 489–497, 2013.

[5] S. Dumble and P. Gibbens, "Efficient terrain-aided visual horizon based attitude estimation and localization," *Journal of Intelligent & Robotic Systems*, vol. 78, no. 2, pp. 205–221, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10846-014-0043-8

[6] S. Weiss, D. Scaramuzza, and R. Siegwart, "Monocular-slam–based navigation for autonomous micro helicopters in GPS-denied environments," *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, 2011.

[7] M. Kaiser, N. Gans, and W. Dixon, "Vision-based estimation for guidance, navigation, and control of an aerial vehicle," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 46, no. 3, pp. 1064–1077, 2010.

[8] K. Kaiser, N. Gans, and W. Dixon, "Position and orientation of an aerial vehicle through chained, vision-based pose reconstruction," 2006.

[9] R. Molero Fernandez, S. Scherer, L. J. Chamberlain, and S. Singh, "Navigation and control for micro aerial vehicles in GPS-denied environments," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-10-08, June 2011.

[10] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: fast semi-direct monocular visual odometry," in *2014 IEEE International Conference on Robotics and Automation, ICRA 2014, Hong Kong, China, May 31 - June 7, 2014*, 2014, pp. 15–22. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2014.6906584

[11] B. Espiau, "Effect of camera calibration errors on visual servoing in robotics," in *The 3rd International Symposium on Experimental Robotics III*. Springer-Verlag, 1993, pp. 182–192.

[12] N. Gans and S. Hutchinson, "Stable visual servoing through hybrid switched-system control," *IEEE Trans. on Robotics*, vol. 23, no. 3, pp. 530–540, June 2007.

[13] E. Malis, F. Chaumette, and S. Boudet, "2-1/2-D visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 2, pp. 238–250, April 1999.

[14] S. Benhimane and E. Malis, "Homography-based 2D visual tracking and servoing," *The International Journal of Robotics Research*, vol. 26, no. 7, pp. 661–676, 2007.

[15] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.

[16] E. Malis and M. Vargas, "Deeper understanding of the homography decomposition for vision-based control," Research Report RR-6303, 2007. [Online]. Available: https://hal.inria.fr/inria-00174036

[17] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[18] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, may 2015.