Optimized Snapshot-based Visual Homing for UAVs

Technical Report #CSSE17-05

Sheetz, Emily
Monmouth College
esheetz@monmouthcollege.edu

Brown, James F.
Southern Connecticut State U.
brownj78@southernct.edu

Chapman, Richard
Auburn University
chapmro@auburn.edu

Saad, Biaz
Auburn University
biazsaa@auburn.edu

July 13, 2017

Abstract

As unmanned aerial vehicles (UAVs) are more widely used, visual homing becomes an increasingly important area of research. Several researchers explore visual navigation or visual target finding on UAVs or ground robots with promising results in simulations and implementation. To explore new approaches for UAV visual homing, techniques for image processing and machine learning must be carefully evaluated, selected, and implemented in order to develop a system that is efficient and practical for real time applications. This work explores a unique visual homing approach for UAVs. Snapshots taken during the UAV's exploratory journey from home create a sparse representation of the traveled path. Feature extraction and brute force feature matching are used to estimate the homography between reference snapshots and camera images on the return journey. The homography is then used to navigate the UAV home. The same feature matching techniques are used in a visual approach to path optimization. This approach allows the UAV to follow near-optimal return paths based on similarities between saved snapshots. These techniques were tested in simulation and showed promising results for accurate and near-optimal visual navigation to home.

1 Introduction

Unmanned aerial vehicles (UAVs) have become an important area of research. UAVs are used in military as well as civilian applications, such as search and rescue, structure inspection, and environmental surveying [10]. However, the operational safety of UAVs continues to be a concern. Visual homing—the UAV's ability to return to the starting position after exploring an area—could continue improving safety and practicality by preventing crashes and aircraft loss.

Much of the research on visual homing has been implemented on ground robots. Although research on visual navigation, target finding, and homing has been done on UAVs, these methods could be improved by combining techniques used by different research teams. Visual homing in the absence of GPS signals is an important problem to solve as it would make UAV operation easier and safer.

1.1 Related Work

Visual homing for unmanned aerial vehicles (UAVs) has become an important topic of research due to the flaws of GPS navigation. A third party can use a jammer to intercept satellite communications to the UAV and send incorrect coordinates to the aircraft. By sending false coordinates from the jammer to the

UAV, a third party would be able to hijack the flight path of the aircraft [17]. Visual homing can add a layer of security to UAV navigation by providing an alternative to easily compromised GPS signals.

Visual homing also offers advantages over other navigation techniques. Cameras are less expensive and can replace bulkier sensors, which is important due to the small payload of UAVs. In addition, camera equipment consumes less power than laser or LIDAR, allowing the UAV to stay in the air longer. Computer vision can also multi-task, with the camera serving as the primary sensor for navigation as well as other missions [10]. Visual homing can provide a sparse representation of the environment, often through a number of snapshots taken by the UAV during its flight. These images can be used for navigation and are independent of maps, which may be imprecise. Path planning and navigation based on snapshots can be done with relatively low computation [18]. Furthermore, the Federal Aviation Administration (FAA) regulations assume that pilots rely on vision, specifically that pilots must "see and avoid" other aircraft [21]. Visual navigation of UAVs offers distinct advantages over other navigation techniques.

Problems similar to visual homing include visual navigation and visual target finding. In the event of GPS jamming or GPS failure, computer vision can be used to approximate the position of the ground robot or UAV, making navigation possible [6] [22]. In research conducted by ChengHao, et al., ground images, or snapshots, were stitched together to create a map of the environment. By combining on-board cameras and an inertial navigation system (INS), the position of the UAV could be approximated [7]. Similarly, in UAV swarms, communication between drones can allow for accurate position estimation and navigation when some of the drones do not have a working GPS [23]. These experiments rely on previous GPS information for navigation [19]. Although the techniques used to address the problem of visual navigation can be applied to visual homing, this project differs in that the research addresses visual homing in the complete absence of GPS information.

Visual target finding is another problem related to visual homing. In research done by Cumbo, et al., the homing strategies of bees inspired the techniques through which a ground robot learned to use landmarks in the environment to approach a particular target. Even when the target was not present, the robot could navigate to the area the target had been based on the landmarks. In some cases, when the target was moved, the robot used landmarks to approach the area and found the target from there. However, when the landmarks were removed, the ground robot could not find the target unless its starting position was relatively close to the target [9]. In this case, the learned target finding was specific to the environment. Once the environment changed by removing the landmarks, the ground robot often could not find its target. While visual homing can be thought of as visual target finding—with the home position being the target—this project's approach to visual homing will be more robust since it will not be environment specific.

There are a variety of approaches to visual homing. Many of these approaches are based on the behaviors of insects such as bees, since they are able to navigate home with poor resolution vision and limited computational brain power [10]. Visual homing includes long-range homing—where the target is not in view—and local homing—where the target is in view [11]. The task of image processing can be achieved through image-based or feature-based approaches. Using the whole image has the advantage of eliminating the need for expensive feature extraction and guarantees a solution that is not dependent on landmarks specific to a particular environment [2]. Feature-based visual homing involves the tasks of feature extraction and feature or image matching [22]. The visual homing approach used in this paper will address long-range homing using feature-based image processing.

Feature extraction is the **first** task of feature-based visual homing. This task can be approached with feature preselection—in which the UAV navigates by matching images from the camera to snapshots—or without feature preselection—in which flow-based matching methods, such as optic flow, are used to estimate how much features move [3].

The **second** task of feature-based visual homing is *image matching*. Because extracting features from an image can be time consuming and computationally expensive, researchers take unique approaches to minimize the total time and computation costs. By reducing an image down to a number of critical points—the criteria for "critical" will depend on the application—the cost function for image matching techniques can be computed only on the select points, rather than every pixel in the image, without losing information [26]. Similarly, knowing only the observed scale and the bearing of select key points which are visible from both the current position of the UAV and the home position, visual homing can be achieved by matching the key points [18].

Image matching depends on being able to match the images taken on the exploratory journey away from the home position with the images the UAV sees on its return journey. In order to compare these images taken from different angles, a homography can be used. The relationship between different images—the homography—can be evaluated by comparing the positions of four or more pairs of reference points in the images. The homography can then be used to compute the control vector, which tells the UAV in which direction to continue home. Research done by Lewis and Beard shows that a homography can be used to achieve effective visual homing in GPS denied environments [16]. Research by DeTone, Malisiewicz, and Rabinovich shows that given two images, deep convolutional neural networks can learn to compute the homography relating the two images. This method has the advantage of avoiding feature extraction and feature matching, as the neural network is able to directly estimate the homography from the two images [12]. Other research has used machine learning and neural networks for various tasks involved in visual navigation [2,3,6,9,22]. This paper proposes using a convolutional neural network to extract features from the reference snapshots and the camera images and compute the homography relating the two images, achieving both the feature extraction and the image matching tasks in the application of visual homing.

Using a series of snapshots taken during the exploratory journey away from the home position, UAVs can match current camera images to the snapshots to navigate back home. As explained in research by Denuelle and Srinivasan, using snapshots for visual homing turns the problem of long-range visual homing into a series of target finding problems through visiting local waypoints. By visiting each of the waypoints represented by the snapshots, the UAV will reach its home position. Snapshots also allow for a sparse representation of the environment [11]. The project presented here uses a similar sparse-snapshot approach and deconstructs the task of visual homing to the task of visiting a series of snapshot waypoints.

Achieving visual homing by visiting a series of snapshot waypoints is equivalent to the UAV retracing the steps it took to reach its goal position. While this technique will allow the UAV to reach its home position, the return journey could be inefficient. The snapshot-based visual navigation scheme proposed by Denuelle and Srinivasan achieved some path optimization, particularly with repeated journeys between the home and goal locations [11]. Unique approaches to visual homing could consider techniques for optimizing the return journey to home.

Research in path optimization typically focuses on finding optimal paths between given starting and ending positions using numerical methods. When searching a region for a particular target, the generated path must be optimal in terms of area coverage, search path length, and search time [1]. Using landmarks, Babel developed a numerical method for finding an optimal path between a given start and end point for a UAV with visual navigation capabilities. This method creates a network of possible paths, and the best path is selected from the generated network. The best path is considered optimal in terms of length, the sequence of landmarks visited, and the distance between landmarks for navigation updates [4].

Snapshot-based visual homing techniques serve as a method for preselecting a series of landmarks along the UAV's journey to home. The start and end points of the path are also predetermined by the exploratory journey. The approach to path optimization presented in this paper will differ from previous research by exploring a visual approach rather than a numerical approach. The advantage of a visual approach to path optimization is that only data gathered from the snapshots taken on the exploratory journey is used to create a near-optimal path. Furthermore, the logic that achieves feature extraction and feature matching for homography computation can be reused for path optimization.

1.2 Contributions

Based on previous research that addresses visual navigation or visual homing for ground robots, the aim of this project is to bring the navigation strategies developed on land into the air. With respect to the research that addresses visual homing in UAVs, the goal is to combine techniques to find a novel solution to visual homing.

This paper addresses long-range visual homing with no previous GPS information. Snapshots will be taken during the exploratory journey away from home with minimal overlap to create a sparse representation of the path. The homography between snapshots and images from the on-board camera will be computed using feature extraction and brute force feature matching. The homography will be used to direct the UAV to each of the reference snapshots. This approach achieves visual homing that is not environment specific.

The visual homing scheme will also have an option for optimization of the return path to home. After the UAV has finished its exploratory journey, the features of the snapshots will be compared. Snapshots with closely matched features will be considered to represent nearby locations in the path, and any snapshots in between will be eliminated from the return journey. This scheme will detect and avoid closed loops in the random exploratory journey, for near-optimal return paths to home. While at least one snapshot will be eliminated from the return journey, the waypoints that could not be optimized will each be visited. At some point, the UAV will have to retrace its steps to home. However, preprocessing the snapshots to nearly optimize the return journey will reduce the distance and time traveled to home.

The methods presented will be tested in a simulated environment based on the work of Lewis and Beard [16], which was generated in MATLAB and Simulink. The Robotics Operating System (ROS) was used to interface with the simulated UAV, with the ROS master being hosted in MATLAB. Python and the open source image processing library *OpenCV* were used for the image processing tasks of feature extraction and feature matching. Past "See And Avoid" research by Morgan and Jones demonstrates the effective coupling of similar technologies [21]. By combining the techniques used for visual homing in ground vehicles and in UAVs, this research will explore a novel approach to the problem of visual homing for UAVs.

1.3 Organization of the Paper

The remainder of the paper is organized as follows: Section 2 details the theory that motivates the approach. The methods used to achieve and test this visual homing framework are explained in Section 3. Section 4 discusses the results of the simulations. Potential future work is outlined in Section 5. Finally, Section 6 explores the conclusions of the research.

2 Theoretical Background

The theory and computation behind homographies and homography control law are central to the optimized approach to visual homing proposed by this research. This section of the paper draws from the book *Multiple View Geometry in Computer Vision* by Hartley and Zisserman about projective transformations [14] and the work done by Lewis and Beard about homography control law [16]. Though outside of the scope of this project, this section will also briefly discuss research from DeTone, Malisiewicz, and Rabinovich about convolutional neural networks and homography estimation. However, this project handles image processing for homography computation in Python using *OpenCV*'s ORB feature detection and brute force feature matching. The homography estimation was used to navigate the UAV home.

2.1 Projective Transformations

Computer vision forces certain assumptions to be made in order to model the three-dimensional world. Images project a 3D scene into two dimensions. This projective transformation does not preserve shapes, lengths, angles, distances, or ratios of distances, as these geometric features can be distorted based on how an image is taken. However, straightness is preserved by such projections. To work with this geometric property, these models, called projective spaces, assume that two lines always meet. To create a projective space $\mathbb{P}^n$, extend the Euclidean space $\mathbb{R}^n$ to include points at infinity. This ensures that parallel lines in the projection will meet at points in the projective space, namely a point at infinity.

To model the world and work with the 2D image representations of the world, computer vision assumes that the world is a 3D projective space and the image corresponds to a 2D projective space. A pixel $(x, y)^T$ in an image corresponds to the homogeneous coordinates $(x, y, 1)^T$ in the projective space $\mathbb{P}^2$. The coordinates of the points in $\mathbb{P}^n$ are vectors with $n+1$ elements, where the $n+1$ element is a constant, k. In fact, the point $(x, y, 1)^T$ in $\mathbb{P}^2$ defines an equivalence class:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \begin{bmatrix} kx \\ ky \\ k \end{bmatrix} \tag{1}$$

Any point that can be related to $(x, y, 1)^T$ through some constant k is considered equivalent to $(x, y, 1)^T$.

It is useful to relate points in the world—represented by projective space $\mathbb{P}^3$—to points in the image of the world—projective space $\mathbb{P}^2$. For example, consider a point in the three-dimensional world with homogeneous coordinates $(X, Y, Z, T)^T$ in $\mathbb{P}^3$. Suppose this point corresponds to a point in the two-dimensional image with homogeneous coordinates $(x, y, w)^T$ in $\mathbb{P}^2$. This means that the image point $(x, y, w)^T$ represents the homogeneous coordinates to the point $(X, Y, Z, T)^T$. These homogeneous points can be related by a linear transformation:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \mathbf{P}_{3\times 4} \begin{bmatrix} X \\ Y \\ Z \\ T \end{bmatrix} \tag{2}$$

where

$$\mathbf{P}_{3\times 4} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3}$$

The linear transformation in Equation 2 simply eliminates the final element of the point in projective space $\mathbb{P}^3$ to find the homogeneous coordinates in projective space $\mathbb{P}^2$.

A projective transformation represents the relationship between points that lie in the same plane. For example, a projective transformation could relate corresponding points between two images. Consider again the points $(X, Y, Z, T)^T$ in $\mathbb{P}^3$ and $(x, y, w)^T$ in $\mathbb{P}^2$—equivalent, homogeneous points—and suppose that each point is captured in one of two images. Because these corresponding points in the two images lie in the same plane, let $Z = 0$. The projective transformation between these two points can be expressed as:

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \mathbf{P}_{3\times 3} \begin{bmatrix} X \\ Y \\ T \end{bmatrix} \tag{4}$$

The matrix $\mathbf{P}$ in Equation 4 represents the projective transformation relating the two points.

The next sections will describe the applications of the theory of projective transformations more specifically to the task of visual homing using two images. More information on multiple view geometries and projective transformations can be found in *Multiple View Geometry in Computer Vision* by Hartley and Zisserman [14].

2.2 Homography

When a landscape is viewed by the same camera from two different angles, the resulting images can be related using a homography [16]. The homography $\mathbf{H}$ is a 3×3 matrix that can be found by relating at least four points—three of which must be noncollinear—from the current image p_c to the matching points in the reference image p_r. Once the homography is found, the two images can be related to one another with the equation

$$p_r = \mathbf{H} p_c \tag{5}$$

where

$$\mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{6}$$

Equation 5 projects the points of image p_c onto image p_r.

Rather than relating two whole images, the homography can also relate two particular points in the images. Let x and y be pixel coordinates of point p_1 in image p_c and x' and y' be the corresponding pixel coordinates of point p'_1 in image p_r. The homography relates these two particular points:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{7}$$

Note that Equation 7 shows similarity rather than equality. As explained in Section 2.1, all scalar multiples of the point $(x, y, 1)$ define an equivalence class. The relationship in Equation 7 holds for points in p_r of the form (kx', ky', k) for any k. The similarity represents that the vectors p'_1 and $\mathbf{H}p_1$ will have the same direction, but may differ in magnitude. Based on the discussion in the previous section, specifically Equation 4, the homography $\mathbf{H}$ defines a projective transformation [14].

The homography can be computed using at least four points in the current image p_c and their matching points in reference image p_r. In order to compute the homography accurately, the selected feature points must be matched between the two images carefully. Given the four matched feature points, we can solve the matrix equation for the vector $\vec{h}$:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 & -x'_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 & -y'_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 & -x'_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 & -y'_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 & -x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4 & -y'_4 \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \vec{0} \tag{8}$$

and rearrange the elements of $\vec{h}$ to obtain matrix $\mathbf{H}$, as in Equation 6. The homography $\mathbf{H}$ is also referred to as the Direct Linear Transformation (DLT) that relates corresponding points in two images [14].

2.3 Reprojection Error

Once the homography, $\mathbf{H}$, has been found, the accuracy of the projection of the current image p_c onto the reference image p_r can be checked. This is achieved by comparing measured feature points in reference image p_r to the projected points in image $\hat{p}_r$, where $\hat{p}_r$ is the result of the matrix multiplication in Equation 5. An individual pixel in $\hat{p}_r$ of the form $(\hat{p}_{r_x}, \hat{p}_{r_y})$ is obtained as in Equation 7. The error between points in reference image p_r and projected image $\hat{p}_r$ is known as the reprojection error e_r, and can be computed as follows:

$$e_r = \sqrt{(p_{r_x} - \hat{p}_{r_x})^2 + (p_{r_y} - \hat{p}_{r_y})^2} \tag{9}$$

Minimizing the reprojection error indicates that the computed homography accurately projects the current image onto the reference image, and therefore properly represents the relationship between the two images. The success of the proposed visual homing scheme relies on correct homography estimation, as the homography will be used to direct the UAV to its next waypoint, represented by a reference snapshot.

2.4 Homography Four-Point Parameterization

Equivalent parameterizations of the homography can relate images in the same way. In fact, the research team of DeTone, Malisiewicz, and Rabinovich found that their deep convolutional network had more success estimating one such parameterization—the four-point homography parameterization, $\mathbf{H}_{4point}$—than the matrix homography in Equation 6, $\mathbf{H}_{matrix}$ [12]. Similar to the matrix homography, the four-point homography parameterization requires four carefully matched points in the current image p_c and the reference image p_r. However, whereas the matrix homography represents the translational and rotational components between the two images, the four-point homography represents the relationship between the four pairs of matched feature points. Considering points (x_i, y_i) from image p_c and points (x'_i, y'_i) from image p_r, it is possible to compute the directional offsets for each component of the points:

$$\Delta u_i = u'_i - u_i \tag{10}$$

Once the offsets Δx_i and Δy_i have been calculated for each point, the four-point homography parameterization can be found:

$$\mathbf{H}_{4point} = \begin{bmatrix} \Delta x_1 & \Delta y_1 \\ \Delta x_2 & \Delta y_2 \\ \Delta x_3 & \Delta y_3 \\ \Delta x_4 & \Delta y_4 \end{bmatrix} \tag{11}$$

There exists a one-to-one correspondence between $\mathbf{H}_{4point}$ and $\mathbf{H}_{matrix}$ that can be found using the Direct Linear Transform (DLT) [14] or the `getPerspectiveTransform()` function in *OpenCV* [12]. While the four-point homography is not used in this project for the purpose of neural network homography estimation, the conversion from $\mathbf{H}_{4point}$ to $\mathbf{H}_{matrix}$ is still a necessary step in the simulations. The methods used for the purposes of this project are described in Section 3.3.

Future research in visual homing could use a convolutional neural network to estimate the four-point homography, $\mathbf{H}_{4point}$, which would then be converted to the equivalent reparameterization, $\mathbf{H}_{matrix}$. The homography would then be used to navigate home. However, the research presented in this paper does not rely on the four-point homography as this deep learning approach is outside of the scope of this project. These methods are further discussed in Section 3.1 to encourage future research in this area.

2.5 Navigation using Homography

Once the four-point homography $\mathbf{H}_{4point}$ has been estimated and converted to the matrix homography $\mathbf{H}_{matrix}$, the UAV will navigate towards the next waypoint using the relationship between the reference snapshot and the current camera view. By visiting each snapshot waypoint, the UAV will eventually navigate home.

Homography control law allows users to compute the vector in the direction that the UAV should travel [16]. The control law is based on the computed homography and the center of gravity of the selected feature points. The center of gravity of the feature points of current camera view is projected onto the reference image using the computed homography. The control vector will point from the reprojected center of gravity to the center of gravity of the feature points in the reference image. The control vector is a 3×1 vector computed by:

$$\vec{v} = \frac{\bar{p}_r^T \mathbf{H} \bar{p}_c}{\bar{p}_r^T \bar{p}_r} \bar{p}_r - \bar{p}_c \tag{12}$$

where

$$\bar{p}_c = \frac{1}{n} \sum_{i=1}^{n} p_{c_i} \tag{13}$$

$$\bar{p}_r = \frac{1}{n}\sum_{i=1}^{n} p_{r_i} \tag{14}$$

Equation 13 represents the center of gravity of the feature points in the current camera view and Equation 14 is the center of gravity of the reference feature points.

The control vector $\vec{v}$ in Equation 12 will direct the UAV to align itself with the reference image. As it reaches the reference image, it will visit the waypoint represented by that particular snapshot and switch to homing in on the next snapshot waypoint. This series of local homing tasks will allow the UAV to navigate home, even in long-range homing.

2.6 Visual Techniques for Path Optimization

The UAV's exploratory journey may double over on itself or form a closed loop. While simply retracing the exploratory journey will allow the UAV to navigate home, the path may be inefficient.

Previous research in visual navigation techniques has achieved some path optimization during repeated journeys between the home and goal positions [11]. However, path optimization was not the focus of the research. Research in path optimization focuses on numerical methods, which may be time consuming or require previous knowledge of wide regions of the environment [4]. Visual approaches to path optimization may offer advantages over other methods in terms of computation time and the amount of data required.

Feature extraction and feature matching for homography computation are achieved in *OpenCV* using ORB feature detection and brute force matching. For each matched point, the brute force feature matching computes an associated "distance," which is a metric that represents how well the features in the two images match. Lower distance values indicate strong matches, while higher values indicate poor matches. When matching features between two images that share several well defined features, the matches will be associated with low distance values, with 0.0 representing a perfect match. However, if given two images that share no features, the brute force matcher will attempt to match similar features. The incorrect matches will result in a high distance value.

The proposed visual approach to path optimization examines the distance associated with feature matches. Before the return journey begins, the snapshots taken during the exploratory journey will be preprocessed. Features will be extracted and matched between pairs of snapshots. If a pair of snapshots has well matched features—the distance metric of the features lies below a certain threshold—it can be assumed that the snapshots represent nearby positions in the path. Rather than retracing all of the steps between the two well matched snapshots, the path optimization algorithm will indicate that the UAV can travel directly from one to the other, eliminating the snapshots in between. This method guarantees that at least one snapshot will be avoided on the return journey when optimization is possible.

Previous research has combined path optimization with visual navigation by using landmarks [4]. The proposed approach to path optimization relies on landmarks—specifically the snapshots—that are close enough together to allow for regular navigation updates from the UAV.

The visual path optimization technique explored in this research will achieve near-optimal return paths to home by skipping over at least one snapshot waypoint. This scheme is not guaranteed to produce completely optimal paths, as the UAV may have to retrace its exploratory journey at some point. However, this approach is worth exploring because it requires no previous knowledge of the environment. The only data needed to achieve near-optimal return paths will be gathered by the UAV during its exploratory journey.

3 Methods

This section describes methods for UAV navigation, visual techniques for path optimization, and the simulation framework for testing. There will also be a brief discussion of methods that could be used to train a convolutional neural network to estimate the homography between two images, though this is not the focus of the approach to visual homing proposed in this project.

3.1 Convolutional Neural Network

As explained in the research of Lewis and Beard, homography computation involves feature extraction, brute force feature matching, and the computation in Equation 8 [16]. DeTone, Malisiewicz, and Rabinovich demonstrated that a neural network could compute the homography directly from two images [12].

While the concept of using a convolutional neural network for computing homography was explored, it was not implemented in the navigation methods presented in this paper. Hardware and time limitations prevented the effective implementation of a convolutional neural network. Future research would be more successful with the availability of a GPU and a substantial amount of training data. The research of DeTone, Malisiewicz, and Rabinovich indicated it took eight hours and 500,000 image pairs to adequately train a deep neural network to estimate the homography between two images on an NVIDIA Titan X GPU [12].

Though the actual neural network was not implemented, a system was created for generated the training data. A Python program was written to randomly crop out a 400×400 pixel image. All of these images were cropped from a Google Earth image which was 5000×5000 pixels. The randomly cropped section was then randomly distorted and warped. Features from the original cropped section and the distorted version were compared and the homography between the two was calculated using *OpenCV*, an open source computer vision library. In the Python program, the *OpenCV* `findHomography()` function was limited to matching the strongest four points in each set of images. These pairs of images, the sets of points, and the homography calculated between them would have been used to train the neural network to estimate the four-point homography, as discussed in Section 2.4.

3.2 Snapshot Preprocessing for Path Optimization

During the randomly generated exploratory journey, the UAV took and stored snapshots of its path. Before beginning the return journey to home, the system has the option of optimizing the return path based on the stored snapshots using the methods described in Section 2.6. To determine whether optimization was possible between two matches, the best eight feature matches were considered. If all eight features were associated with a distance metric below a certain threshold—the simulation framework used a distance value of 15—the algorithm would optimize the return path between these two snapshots.

The return journey was only optimized once. In cases where there were multiple opportunities for optimization, the system would optimize once, but otherwise retrace the exploratory journey. This means that the proposed visual techniques for path optimization produce near-optimal paths. The path optimization algorithm prioritizes opportunities for optimization closest to the goal position of the UAV—which is the current position at the time the algorithm is run. The algorithm also prioritizes the opportunity for optimization that eliminates the most snapshot waypoints. The algorithm stops as soon as it finds an optimization closest to the goal position of the exploratory journey. This means it may neglect to find longer opportunities for optimization farther away from the goal position.

Once the path was optimized, the UAV could begin its return journey and navigate home, using the methods described in the next section.

3.3 Navigation

The homography estimate is central to the methods through which the UAV navigates home. If implemented and integrated into the system, the convolutional neural network would estimate the four-point homography parameterization, $\mathbf{H}_{4point}$, which must be converted to the matrix homography, $\mathbf{H}_{matrix}$. The matrix homography is computed directly from the four pairs of feature points in the camera and reference images, as in Equation 8. This is also known as the Direct Linear Transform (DLT). As explained by DeTone, Malisiewicz, and Rabinovich, the points used must maintain the relationship described by the elements of the four-point homography [12]. To obtain the four point pairs from the four-point homography parameterization, suppose the feature points in the reference snapshot image p_r are oriented around an origin:

$$\begin{aligned}(x'_1, y'_1) &= (1, 1)\\(x'_2, y'_2) &= (-1, 1)\\(x'_3, y'_3) &= (-1, -1)\\(x'_4, y'_4) &= (1, -1)\end{aligned} \tag{15}$$

The selection of these points defines the coordinate system.

With the feature points in the reference view selected, the four-point homography can be used to obtain the feature points in the camera image p_c:

$$\begin{aligned}(x_1, y_1) &= (1 + \Delta x_1, 1 + \Delta y_1)\\(x_2, y_2) &= (-1 + \Delta x_2, 1 + \Delta y_2)\\(x_3, y_3) &= (-1 + \Delta x_3, -1 + \Delta y_3)\\(x_4, y_4) &= (1 + \Delta x_4, -1 + \Delta y_4)\end{aligned} \tag{16}$$

Using this method, pairs of feature points that preserve the relationship described in the four-point homography can be found. These feature points were used to compute the matrix homography using Equation 8.

The four-point homography parameterization and the described conversion to the matrix homography would be necessary if visual homing were to be tested with neural network homography estimation. Though this research does not use a neural network, the four-point parameterization was still used. As will be discussed in the next section, the simulation framework is coded in MATLAB and Simulink, with function calls to Python and *OpenCV* for the feature extraction and matching that could have been done by a neural network. Simulations run in Simulink do not run in real time, making the simulation run time quite slow. Calling Python functions from MATLAB and manipulating the Python outputs to be readable in MATLAB further slowed down the system. The number of outputs needed from the Python code was minimized in an attempt to reduce the simulation run time where possible. As a result, it was simpler to get the directional offsets of the matched feature points in each image, rearrange the outputs into the four-point homography parameterization, and then convert to the matrix homography.

Once the matrix homography was computed, the control vector was found using homography control law, as expressed in Equation 12. The resulting vector $\vec{v}$ points in the direction that the UAV should travel to approach the reference snapshot. The length of the vector is not needed for the purpose of navigation, so the control vector $\vec{v}$ was normalized.

3.4 Simulation Framework

The optimized visual homing approach was tested in simulations designed using MATLAB and Simulink. A random exploratory journey consisting of a series of straight line paths was generated. These paths were determined by the number of straight line segments comprising the path and the duration of the flight.

During the exploratory journey, snapshots were taken and stored for later navigation home. Snapshots were taken at fixed intervals of time. The fixed altitude of the UAV affected the dimensions of the snapshot. The altitude of the UAV was adjusted to get snapshots large enough to capture features that could be used for homography computation, but small enough to offer a sparse representation of the path. The snapshots were also taken so that they overlapped neighboring snapshots. This guaranteed that the position of the UAV would always correspond to the region captured in a snapshot.

The Robotics Operating System (ROS) was used to communicate messages about the UAV's direction of travel. These messages were expressed as unit direction vectors. The ROS master was simulated through the MATLAB Robotics System Toolbox.

Once the UAV reached the end of the predetermined exploratory journey, the UAV turned around by reversing the previous direction vector. This computation indicated for the UAV to immediately turn around and head in the direction from which it came. After the UAV turned around, it started using homography control law to navigate home—as explained in Section 2.5 and Section 3.3—and switched between snapshots when necessary.

Because snapshots taken during the exploratory journey were taken at fixed time intervals, the simulation framework assumed that snapshots would be reached at the same time intervals during the return journey. The time interval was altered when path optimization had occurred between two particular snapshots.

Simulink models do not run in real time. As a result, the simulations ran quite slowly. The system was further slowed by calling functions in Python—which was necessary to work with *OpenCV* image processing tools—from MATLAB. Despite the slow run time of the simulations, the computation that allowed for successful optimized visual homing occurred relatively quickly. For this reason, these approaches could be implemented on a physical aircraft for real-time optimized visual homing.

3.5 System Specifications

Parameters used to represent aspects of the system had to be tuned to serve the particular purposes of this project. Snapshots were taken every 30 seconds to provide sufficient overlap for navigation. On the return journey, the system homed in to a different snapshot waypoint every 30 seconds.

Path optimization occurred when the best eight matched features between snapshots were all associated with a "distance" value less than 15. When path optimization occurred between two snapshots, these snapshots were generally closer together. As a result, the time the UAV had to home in on the optimized snapshot was reduced to 15 seconds.

The camera view was updated every 5 seconds, meaning that the control vector had to be recomputed and updated. This update time ensured that the camera view and snapshot were distinct enough to allow for good homography computation and gave the UAV plenty of time to recompute the vector and adjust its trajectory if it got off course.

The total flight time and sample time were adjusted to control the duration and length of the path along with the number of samples during the exploratory journey. The simulated tests allowed for 3 minutes of flight on the exploratory journey away from home with a sample time of $\Delta t = 0.1$ seconds. The return journey was also allowed to run for 3 minutes, though the return simulation would terminate early when the UAV successfully reached home.

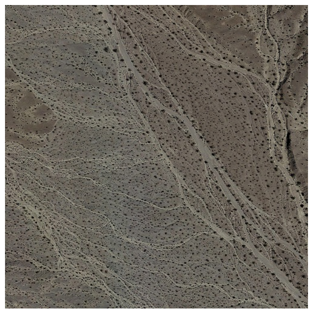
It was assumed that the camera had a 65° view angle in each direction, as in the simulations of Lewis and Beard [16]. The altitude of the UAV remained fixed at all points of the simulation, but was adjusted to obtain snapshots large enough to contain a reasonable number of features. An altitude of 93.3 pixels created snapshots that were 400×400 pixels, which proved to be a sufficient size for navigation purposes. The paths could be completely represented by a total of seven snapshots.

3.6 Experiments

To test the visual homing and path optimization techniques, random exploratory journeys were generated in three environments: a suburb, a desert, and a forest. Figure 1 shows the images of the environments. Three paths were tested in five locations in each environment, for a total of 15 experiments per environment and 45 total experiments.

Data from both the return journey and the optimized return journey were collected in order to test both the visual homing techniques and the combination of this approach with path optimization. For each experiment, the gathered data included: plots of the exploratory and return journeys, positions at which snapshots were taken, positions at which the camera and snapshot views were updated, and the history of the direction vectors and positions for the exploratory and return journeys. Position estimates were in pixel coordinates relative to the starting position of the journey, which was assumed to be the origin.

(a) Suburban environment.


(b) Desert environment.

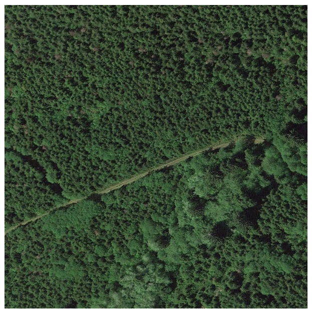

(c) Forest environment.

Figure 1: Environments tested.

4 Results

Data from the return and optimized return journeys was analyzed to determine the effectiveness of the techniques for visual homing and path optimization.

The results of the experiments in the suburban, desert, and forest environments are displayed in Tables 1, 2, and 3, respectively. The overall results of all of the experiments are described in Table 4. These tables show the percentage of tests that successfully reached the home position, the percentage of tests that were also successful when the return path was optimized, and the average reduction in travel distance and time when optimized.

Suburb Experiment Results	
Successful Homing	87%
Successful Optimized Homing	85%
Average Distance Reduced	48%
Average Time Saved	84 seconds

Table 1: Results of experiments in suburban environment.

Desert Experiment Results	
Successful Homing	100%
Successful Optimized Homing	67%
Average Distance Reduced	54%
Average Time Saved	95 seconds

Table 2: Results of experiments in desert environment.

Forest Experiment Results	
Successful Homing	93%
Successful Optimized Homing	50%
Average Distance Reduced	59%
Average Time Saved	104 seconds

Table 3: Results of experiments in forest environment.

Overall Experiment Results	
Successful Homing	93%
Successful Optimized Homing	67%
Average Distance Reduced	53%
Average Time Saved	93 seconds
Average Snapshots Optimized	2.7 snapshots

Table 4: Results of all experiments.

The results of the experiments show that the optimized visual homing performed poorly in the desert and forest environments. This behavior was to be expected because the visual homing and path optimization techniques depend on feature extraction and matching. Both the desert and forest environments contain uniform features across the map, making it more difficult to match features and detect opportunities for path optimization. Similarly, when the UAV would get lost in the suburban environment, it was typically when working with images of roads, driveways, or groups of trees, where there are few features to detect.

The path optimization techniques ensured that at least one snapshot waypoint would be eliminated from each return path. On average, two to three snapshots were eliminated in the near-optimal return path. In the optimal case, the goal position at the end of the exploratory journey was close enough to the home position that the path optimization would eliminate all snapshots besides the first—the snapshot representing the home location—and the last—the snapshot representing the goal location. An example of successful navigation in the suburb, desert, and forest can be found in Figures 2, 3, and 4, respectively. In all of these figures, the blue path represents the exploratory journey, the green circles represent the snapshots taken during the exploratory journey, and the red path represents the return journey.

While the proposed near-optimal visual homing framework failed in some tests, the results of the simulations are promising. Fine-tuning the parameters of the simulations could improve feature extraction and intelligent path optimization. The results of the experiments demonstrate the success of combining visual navigation, sparse-snapshot representation, homography control law, and visual path optimization techniques.

5 Future Work

There are several limitations of the techniques and simulations proposed in this project, which could serve as areas for potential future work related to optimized visual homing strategies.

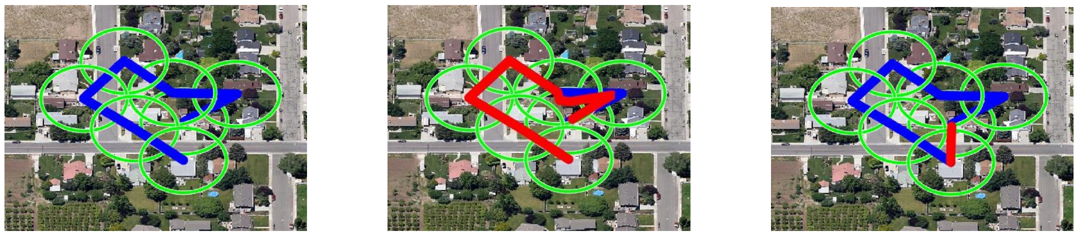
5.1 Neural Network Homography Estimation

Future work would include implementing the neural network strategy discussed previously in Section 3.1. Combining neural network homography estimation and homography control law for visual navigation merits research. It is anticipated that a GPU and a vast amount of training data will be required to successfully implement the neural network.

Using a convolutional neural network for feature extraction, feature matching, and homography estimation could result in a visual homing scheme that is more robust. The approach presented in this paper failed when working with images with uniform features. Poor feature matches caused inaccurate computation of the control vector, in which case the UAV would get lost and be unable to find home. A well trained convolutional neural network could estimate the homography even in more challenging environments.

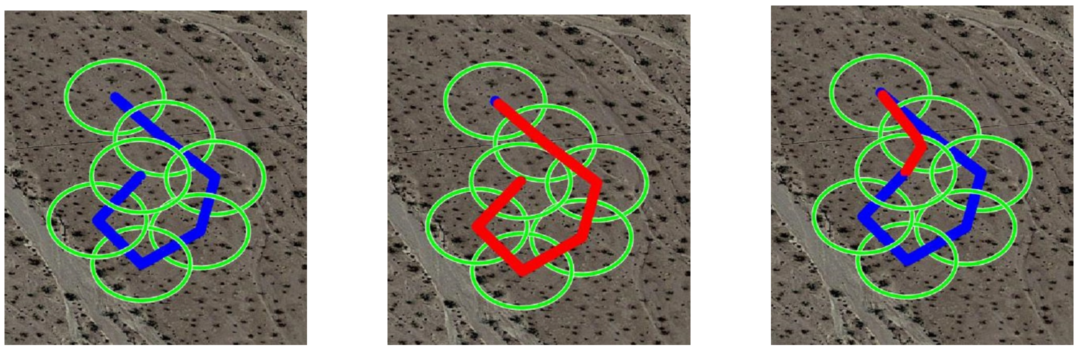
5.2 Snapshot Switching Logic

During navigation towards home, it is necessary to identify when a snapshot waypoint has been approximately reached, so the system can switch to the next reference snapshot. The simulation framework used



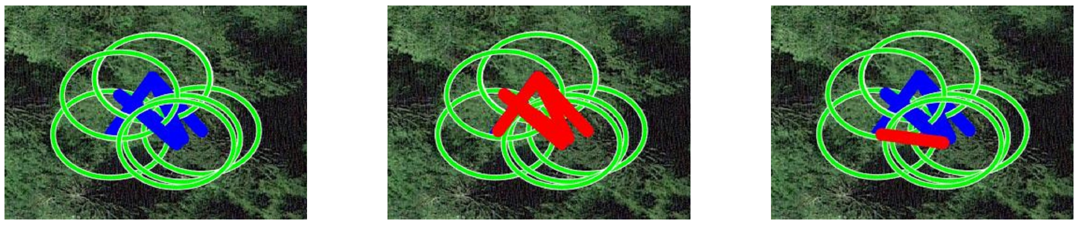
(a) Exploratory journey. (b) Return journey. (c) Optimized return journey.

Figure 2: Sample experiment run in suburban environment. In both the return journey and the optimized return journey, the UAV successfully navigated home. The optimized return journey went directly from the final snapshot—in this case, the snapshot closest to the bottom of the image—to home—near the center of the image.



(a) Exploratory journey. (b) Return journey. (c) Optimized return journey.

Figure 3: Sample experiment run in desert environment. In both the return journey and the optimized return journey, the UAV successfully navigated home. The near-optimal return journey moved from the final snapshot—the snapshot near the center of the image—to the second snapshot. From there, the UAV retraced its exploratory journey to the home position—the snapshot near the top of the image.



(a) Exploratory journey. (b) Return journey. (c) Optimized return journey.

Figure 4: Sample experiment run in forest environment. In both the return journey and the optimized return journey, the UAV successfully navigated home. The optimized return journey went directly from the goal position—near the bottom right of the image—to the home position—near the left of the image.

in this research switches snapshots after fixed amounts of time. Even if the UAV did not reach a snapshot waypoint in the allotted time, the system would switch to the next snapshot. The UAV would often be unable to recover its course and would not find home.

An alternative snapshot switching scheme could improve simulation results and give the UAV a chance to recover if it initially heads the wrong way. As explained by Lewis and Beard, the control vector in Equation 12 can be used as an estimate of the pixel offset between images [16]. As the UAV moves away from the area represented in the snapshot, the control vector pointing from the reprojected center of gravity, $\mathbf{H}\bar{p}_c$, to the center of gravity of features in p_r, $\bar{p}_r$, will increase in length. The length of the control vector, therefore, can be used as an estimate of the pixel offset between the two images. Once the pixel offset increases beyond some predefined threshold, the system can switch to the next snapshot. The threshold value will depend on the system.

The success of this approach depends on the snapshots overlapping. When considering how often a snapshot should be taken, previous research has considered the catchment area corresponding to each snapshot [11] or the number of features shared between snapshots [16].

Though using time as a snapshot switching technique showed promising results, using the pixel offset for snapshot switching could improve results and allow the UAV to recover if it is initially lost.

5.3 Path Optimization

The path optimization algorithm could be improved for more reliable visual homing results. Experiments that show flaws in the path optimization algorithm are displayed in Figures 5, 6, and 7. Switching between optimized snapshots sometimes caused the UAV to lose its course if it did not quite reach the snapshot before the system switched. Furthermore, the threshold “distance” value used to evaluate opportunities for optimization could be tuned for better optimization. More features could be examined and the number of well matched features could be used to better prioritize opportunities for optimization. Taking advantage of optimization between multiple snapshots could further improve the visual path optimization techniques.

5.4 Simulations and Implementation

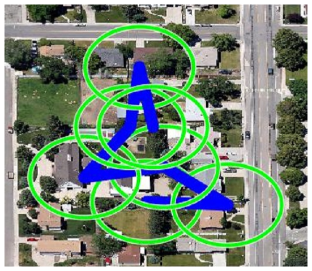
The simulation framework used in this project did not consider realistic flight dynamics or characteristics of a particular aircraft. More realistic simulations tested in a wider variety of environments could help refine the techniques for optimized visual homing. After further simulations, implementation on a physical aircraft could lead to more improvements in these techniques.

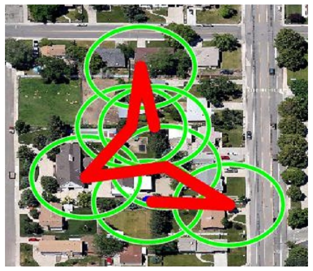
With further research and testing, the proposed techniques could prove to be an efficient and practical solution to optimized visual homing for UAVs.

6 Conclusion

The results presented in this paper demonstrate that it is possible to navigate through an environment without the use of a GPS. The simulations show that a UAV can potentially navigate along near-optimal paths using a sparse representation of the environment. The simulated UAV was able to navigate home almost all of the time before calculating an optimized path. However, the UAV did not experience a similar success rate when navigating back to home along an optimized path. In the suburban environment, the UAV was able to successfully navigate optimized paths most of the time. When the UAV failed to navigate home and got lost, it was often in areas that had very similar features, such as wide streets and open fields. Houses, cars, and sidewalks proved to be reliable features for navigation.

Throughout all environments, the UAV would get lost based on the features shared between snapshots. When the overlap between snapshots did not contain distinct features, *OpenCV* would be forced to find poor matches. Poorly matched features would cause inaccurate homography computation and throw the UAV off course. Once a UAV was off course, it could rarely recover and would not be able to navigate back to the home location. The desert and forest environments saw much less success in optimizing successfully


(a) Exploratory journey.


(b) Return journey.

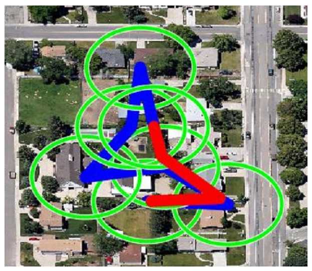
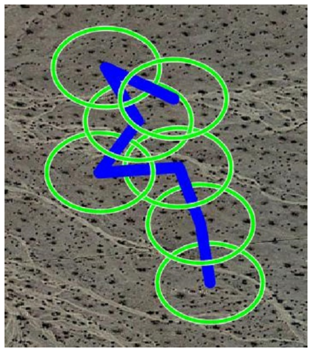
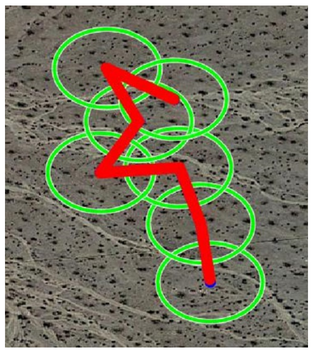

(c) Optimized return journey.

Figure 5: Sample experiment run in suburban environment. This experiment demonstrates the opportunity for path optimization in two separate sections of the path. The path optimization algorithm prioritizes the longest opportunity for optimization closest to the final snapshot—near the center of the image. However, after the path optimization, the UAV begins retracing its steps from the exploratory journey, when it could optimize a second time and head straight to home—near the bottom of the image.


(a) Exploratory journey.


(b) Return journey.

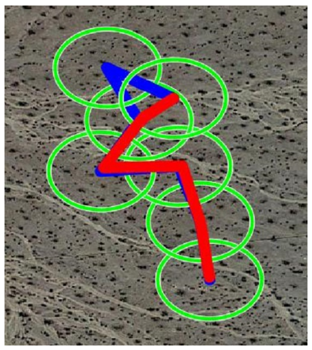
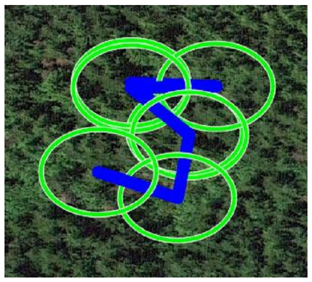
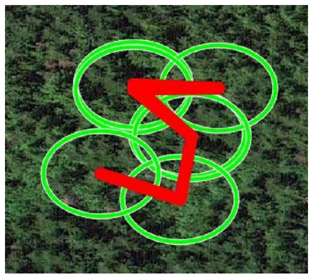

(c) Optimized return journey.

Figure 6: Sample experiment run in desert environment. This experiment demonstrates the opportunity for path optimization in two separate sections of the path. Rather than retracing the exploratory journey after cutting out one snapshot, the path optimization algorithm could be applied again for a more direct return path to home—near the bottom of the image.


(a) Exploratory journey.


(b) Return journey.

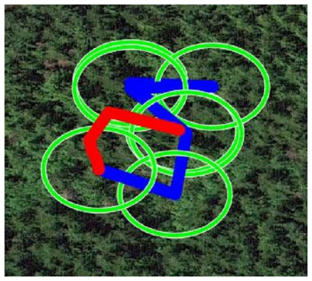

(c) Optimized return journey.

Figure 7: Sample experiment run in forest environment. This experiment is an example of when path optimization would cause the UAV to get lost. During the optimized return journey, the UAV initially got lost and went the wrong way when leaving the goal location—the snapshot near the bottom left of the image. However, the UAV eventually recognized it was close to the home snapshot—near the center of the image—and navigated home.

navigated paths; the UAV failed to navigate home along the optimized path one third to one half of the time. The failure of the UAV to follow the optimized path in the desert and forest can be attributed to the uniformity of the environment.

While the results of the navigation simulations are not exceptional, they do demonstrate that it is possible to optimize the homing path of UAV, even with a sparse snapshot representation of the journey. When optimized visual homing was achieved, it significantly reduced the the flight distance back to home. The average reduction in the flight path of the UAV was 59%. The reduction in distance almost halved the amount of flight time required to return to the home location. While the homing system developed in this report was not the most reliable, it provided real benefits when strong feature matches were available. With more research and time, these techniques will prove to be even more effective and reliable. The results of this simulation demonstrate the potential of GPS-less optimized visual navigation for UAVs.

References Cited

[1] Vitaly Ablavsky, Daniel Stouch, and Magnús Snorrason. Search path optimization for UAVs using stochastic sampling with abstract pattern descriptors. In *Proceedings of the AIAA Guidance Navigation and Control Conference*, 2003.

[2] Abdulrahman Altahhan. Robot visual homing using conjugate gradient temporal difference learning, radial basis features and a whole image measure. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–10. IEEE, 2010.

[3] P Arena, S De Fiore, L Fortuna, L Nicolosi, L Patané, and G Vagliasindi. Visual homing: experimental results on an autonomous robot. In *Circuit Theory and Design, 2007. ECCTD 2007. 18th European Conference on*, pages 304–307. IEEE, 2007.

[4] Luitpold Babel. Flight path planning for unmanned aerial vehicles with landmark-based visual navigation. *Robotics and Autonomous Systems*, 62(2):142–150, 2014.

[5] G Bianco, R Cassinis, A Rizzi, N Adami, and P Mosna. A bee-inspired robot visual homing method. In *Advanced Mobile Robots, 1997. Proceedings., Second EUROMICRO workshop on*, pages 141–146. IEEE, 1997.

[6] José RG Braga, Haroldo FC Velho, Gianpaolo Conte, Patrick Doherty, and Élcio H Shiguemori. An image matching system for autonomous UAV navigation based on neural network. In *Control, Automation, Robotics and Vision (ICARCV), 2016 14th International Conference on*, pages 1–6. IEEE, 2016.

[7] Zhang ChengHao, Chen JiaBin, Song ChunLei, and Xu JianHua. An UAV navigation aided with computer vision. In *Control and Decision Conference (2014 CCDC), The 26th Chinese*, pages 5297–5301. IEEE, 2014.

[8] Gianpaolo Conte and Patrick Doherty. An integrated UAV navigation system based on aerial image matching. In *Aerospace Conference, 2008 IEEE*, pages 1–10. IEEE, 2008.

[9] Kodi CA Cumbo, Samantha Heck, Ian Tanimoto, Travis DeVault, Robert Heckendorn, and Terence Soule. Bee-inspired landmark recognition in robotic navigation. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference Companion*, pages 1039–1042. ACM, 2016.

[10] Aymeric Denuelle and Mandyam V Srinivasan. Bio-inspired visual guidance: From insect homing to UAS navigation. In *Robotics and Biomimetics (ROBIO), 2015 IEEE International Conference on*, pages 326–332. IEEE, 2015.

[11] Aymeric Denuelle and Mandyam V Srinivasan. A sparse snapshot-based navigation strategy for UAS guidance in natural environments. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3455–3462. IEEE, 2016.

[12] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *arXiv preprint arXiv:1606.03798*, 2016.

[13] Verena V Hafner, Ferry Bachmann, Oswald Berthold, Michael Schulz, and Mathias Müller. An autonomous flying robot for testing bio-inspired navigation strategies. In *Robotics (ISR), 2010 41st International Symposium on and 2010 6th German Conference on Robotics (ROBOTIK)*, pages 1–7. VDE, 2010.

[14] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision.* Cambridge university press, 2003.

[15] Yu-Te Huang, Yao-Hua Ho, Hao-hua Chu, and Ling-Jyh Chen. Adaptive drone sensing with always return-to-home guaranteed. In *Proceedings of the 1st International Workshop on Experiences with the Design and Implementation of Smart Objects*, pages 7–12. ACM, 2015.

[16] Benjamin P Lewis and Randal W Beard. A framework for visual return-to-home capability in GPS-denied environments. In *Unmanned Aircraft Systems (ICUAS), 2016 International Conference on*, pages 633–642. IEEE, 2016.

[17] Chang Li and Xudong Wang. Jamming research of the UAV GPS/INS integrated navigation system based on trajectory cheating. In *Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), International Congress on*, pages 1113–1117. IEEE, 2016.

[18] Ming Liu, Cedric Pradalier, and Roland Siegwart. Visual homing from scale with an uncalibrated omnidirectional camera. *IEEE Transactions on Robotics*, 29(6):1353–1365, 2013.

[19] P Lukashevich, A Belotserkovsky, and A Nedzved. The new approach for reliable UAV navigation based on onboard camera image processing. In *Information and Digital Technologies (IDT), 2015 International Conference on*, pages 230–234. IEEE, 2015.

[20] Ezio Malis and Manuel Vargas. *Deeper understanding of the homography decomposition for vision-based control*. PhD thesis, INRIA, 2007.

[21] Andrew Morgan, Zach Jones, and Richard Chapman. Computer vision see and avoid simulation using OpenGL and OpenCV. 2016.

[22] Victor Sineglazov and Vitaliy Ischenko. Intelligent system for visual navigation. In *Methods and Systems of Navigation and Motion Control (MSNMC), 2016 4th International Conference on*, pages 7–11. IEEE, 2016.

[23] Amedeo Rodi Vetrella and Giancarmine Fasano. Cooperative UAV navigation under nominal GPS coverage and in GPS-challenging environments. In *Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 2016 IEEE 2nd International Forum on*, pages 1–5. IEEE, 2016.

[24] Kai Virtanen, Harri Ehtamo, Tuomas Raivio, and Raimo P Hamalainen. VIATO-visual interactive aircraft trajectory optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 29(3):409–421, 1999.

[25] Anastasiia Volkova and Peter W Gibbens. Extended robust feature-based visual navigation system for UAVs. In *Digital Image Computing: Techniques and Applications (DICTA), 2016 International Conference on*, pages 1–7. IEEE, 2016.

[26] Jane You, Edwige Pissaloux, and Harvey A Cohen. A hierarchical image matching scheme based on the dynamic detection of interesting points. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 4, pages 2467–2470. IEEE, 1995.