Neural Network Homography Estimation in Snapshot-based Visual Homing for UAVs

Technical Report #CSSE17-05

Sheetz, Emily
Monmouth College
esheetz@monmouthcollege.edu

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

June 20, 2017

Abstract

As unmanned aerial vehicles (UAVs) become more widely used, visual homing is an increasingly important area of research. Several publications 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 considered, selected, and implemented in order to develop a system that is practical for real time applications. The research presented in this paper takes a unique approach to the problem of visual homing for UAVs. A convolutional neural network was trained to estimate the homography between reference snapshots of the exploratory journey and camera images on the return journey by extracting necessary features. The homography was then used to navigate the UAV home. This approach was tested in simulation, showed promising results for accurate visual navigation to home, and could be a practical solution to achieve visual homing in real-time on-board UAVs.

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 [8]. However, the operational safety of UAVs continues to be a concern. Visual homing—the ability of the UAV 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 coordinates 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 based 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 the flight path of the aircraft [15]. 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 because of 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 multitask, with the camera serving as the primary sensor for navigation as well as other missions [8]. Visual homing offers 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 [16]. Furthermore, the Federal Aviation Administration (FAA) regulations assume that pilots rely on vision, specifically that pilots must "see and avoid" other aircraft [18]. Visual navigation of UAVs can offer distinct advantages over other navigation techniques.

Problems similar to visual homing include visual navigation or 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 [4] [19]. In research conducted by ChengHao, et al., ground images, or snapshots, were stitched together to create a map of the environment. By combining onboard cameras and an inertial navigation system (INS), the position of the UAV could be approximated [5]. 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 [20]. These and similar experiments rely on previous GPS information for navigation [17]. Though 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 [7]. 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 biological creatures such as bees or other insects, which are able to navigate home with poor resolution vision and limited computational brain power [8]. Visual homing includes long-range homing—where the target is not in view—and local homing—where the target is in view [9]. 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 [1]. Feature-based visual homing involves the tasks of feature extraction and feature or image matching [19]. 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 [2].

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 [22]. 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 [16].

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 [14]. 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, as the neural network is able to directly estimate the homography from the two images [10]. This paper's approach uses 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.

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 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 or visiting local waypoints. By visiting each of the waypoints represented by the snapshots, the UAV will reach its home position [9]. The project presented here uses a similar approach by deconstructing the task of visual homing to the task of visiting a series of waypoints represented by snapshots of the environment.

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. A convolutional neural network extracts features from snapshots of the environment taken on the exporatory journey and images from the onboard camera to compute the homography. The homography matches the images by representing the relationship between them. This machine learning approach achieves visual homing that is not environment specific. Previous research used neural networks for various tasks involved in visual navigation [1] [2] [4] [7] [19] and specifically for homography estimation [10].

The methods presented will be tested in a simulated environment based on the work of Lewis and Beard [14], 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 training data acquisition for the neural network. Past "See And Avoid" research by Morgan and Jones demonstrates the effective coupling of similar technologies [18]. 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. Finally, the conclusions of the research and future work will be explored in Section 5.

2 Theoretical Background

The theory and computation behind homographies and homography control law are central to our machine learning approach to visual homing. This section of the paper draws from the book *Multiple View Geometry in Computer Vision* by Hartley and Zisserman about projective transformations [12]; work done by Lewis and Beard about homography control theory [14]; and research from DeTone, Malisiewicz, and Rabinovich about convolutional neural networks and homography estimation [10]. For this project, the convolutional neural network was trained to extract features and compute the homography between camera images and reference snapshots. This 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 distored 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 homogenous 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-dimensonal world with homogenous coordinates $(X, Y, Z, T)^T$ in $\mathbb{P}^3$. Suppose this point corresponds to a point in the two-dimensional image with homogenous coordinates $(x, y, w)^T$ in $\mathbb{P}^2$. This means that the image point $(x, y, w)^T$ represents the homogenous coordinates to the point $(X, Y, Z, T)^T$. These homogenous 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 homogenous 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, homogenous 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 [12].

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 [14]. 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 [12].

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 [12].

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 our 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, $\mathbf{H}_{matrix}$ [10]. 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. 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 point:

$$\Delta x_i = x'_i - x_i \tag{10}$$

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) [12] or the `getPerspectiveTransform()` function in *OpenCV* [10]. The methods used for the purpose of this project are described in Section 3.3.

The convolutional neural network estimates the four-point homography, $\mathbf{H}_{4point}$, which is converted to the equivalent reparameterization, $\mathbf{H}_{matrix}$. The homography is then used to navigate the UAV towards home.

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 [14]. 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

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

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

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

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.

3 Methods

This section describes methods used to obtain training and testing data, methods through which the convolutional neural network learned to estimate the homography between two images, methods for UAV navigation, and the simulation framework for testing.

3.1 Data Acquisition for Learning

To train the convolutional neural network for homography estimation, traning and testing data sets consisting of images and homography estimates must be acquired.

The images used to simulate an environment for the UAV were Google Earth images used by Lewis and Beard [14]. These 5000×5000 pixel images are satellite photos of a forest, a desert, and a suburb. Sections of these images were used to train the neural network.

A Python program was written to randomly crop out a 300×300 pixel image. 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 were used to train the neural network.

3.2 Convolutional Neural Network

3.3 Navigation

The homography estimate from the convolutional neural network is central to the methods through which the UAV navigates home. The convolutional neural network estimates the four-point parameterization of the homography, $\mathbf{H}_{4point}$, 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 [10]. To obtain the four point pairs from the four-point homography parameterization, suppose the feature points in the camera image p_c are oriented around an origin:

$$\begin{aligned}(x_1, y_1) &= (1, 1)\\ (x_2, y_2) &= (-1, 1)\\ (x_3, y_3) &= (-1, -1)\\ (x_4, x_4) &= (1, -1)\end{aligned} \tag{15}$$

The selection of these points defines the coordinate system.

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

$$\begin{aligned}(x'_1, y'_1) &= (1 + \Delta x_1, 1 + \Delta y_1)\\ (x'_2, y'_2) &= (-1 + \Delta x_2, 1 + \Delta y_2)\\ (x'_3, y'_3) &= (-1 + \Delta x_3, -1 + \Delta y_3)\\ (x'_4, y'_4) &= (1 + \Delta x_4, -1 + \Delta y_4)\end{aligned} \tag{16}$$

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.

Once the matrix homography is 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 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. As explained by Lewis and Beard, the control vector in Equation 12 can be used as an estimate of the pixel offset between images [14].

Throughout the inbound flight towards home, the UAV compares its camera view to the reference snapshot by estimating the homography between these two images. 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 [9] or the number of features shared between snapshots [14]. This project instead considers time between snapshots. The first snapshot of each exploratory journey was of the home location, taken when time $t = 0$ seconds. For the purposes of this project, taking snapshots every 20 seconds generally provided enough overlap between snapshots.

3.5 Simulation Framework

The 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, the duration of the flight, and bounding parameters to prevent too drastic of turns during the flight. The methods used for generating these random paths affected how much time could pass between snapshots to ensure overlap.

The Robotics Operating System (ROS) was used to communicate messages about the position of the UAV based on the random exploratory journey. The ROS master was simulated through the MATLAB Robotics System Toolbox.

Once the UAV reached the end of the predetermined exploratory journey, an initial normalized direction vector was computed using the current estimated position of the UAV and its previous position. Position estimates were in pixel coordinates relative to the starting position of the journey, which was assumed to be the origin. 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 estimations from the pretrained convolutional neural network to navigate home—as explained in Section 3.3—and switched between snapshots when necessary—as described in Section 3.4.

Parameters used to represent aspects of the system had to be tuned to serve the particular purposes of this project. As stated in the previous section, taking snapshots every 20 seconds generally provided sufficient overlap for the snapshot switching logic. 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 $\Delta t = 0.2$ seconds. It was assumed that the camera had a 65° view angle in each direction, as in the simulations of Lewis and Beard [14]. 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. The code used to obtain training and testing samples for the neural network properly identified and matched features for 300×300 pixel images.

3.6 Experiments

4 Results

5 Conclusion

References Cited

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

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

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Andrew Morgan, Zach Jones, and Richard Chapman. Computer vision see and avoid simulation using opengl and opencv. 2016.

[19] 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.

[20] 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.

[21] 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.

[22] 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.