UNIVERSIDAD DE LOS ANDES

Facultad de Ingeniería

A Vision Based Algorithm for the Guidance of a Glider

Dissertation presented to fulfill the requirements for the degree of
Master of Science in Computer Science

Author: John Jairo Martínez Benavides
Advisor: José Tiberio Hernández

Bogotá, D.C. June 2013
Acknowledgments

This research was financed by Industria Militar de Colombia (INDUMIL). The project was directed by professor Carlos Francisco Rodríguez to whom the authors would like to thank for his support in the development of the ideas behind this research.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acknowledgments</td>
<td>2</td>
</tr>
<tr>
<td>1 Abstract</td>
<td>5</td>
</tr>
<tr>
<td>2 Introduction</td>
<td>6</td>
</tr>
<tr>
<td>3 Problem Statement</td>
<td>9</td>
</tr>
<tr>
<td>3.1 Aim</td>
<td>9</td>
</tr>
<tr>
<td>3.2 Objectives</td>
<td>10</td>
</tr>
<tr>
<td>4 Solution</td>
<td>11</td>
</tr>
<tr>
<td>4.1 Feature Detection</td>
<td>12</td>
</tr>
<tr>
<td>4.2 Transformation Calculation</td>
<td>13</td>
</tr>
<tr>
<td>4.3 Point Estimation</td>
<td>14</td>
</tr>
<tr>
<td>4.4 Deformation Indicator</td>
<td>14</td>
</tr>
<tr>
<td>5 Simulation Design and Implementation</td>
<td>16</td>
</tr>
<tr>
<td>5.1 Glider’s Dynamic Model</td>
<td>16</td>
</tr>
<tr>
<td>5.1.1 Bang-Bang Control</td>
<td>19</td>
</tr>
<tr>
<td>5.2 Scene Rendering</td>
<td>19</td>
</tr>
<tr>
<td>5.3 Solution Implementation</td>
<td>20</td>
</tr>
<tr>
<td>5.4 Algorithm Evaluation</td>
<td>21</td>
</tr>
<tr>
<td>5.4.1 Transforming Earth Coordinates to Image Coordinates</td>
<td>21</td>
</tr>
<tr>
<td>5.4.2 Transforming Image Coordinates to Earth Coordinates</td>
<td>22</td>
</tr>
<tr>
<td>6 Algorithm Evaluation</td>
<td>24</td>
</tr>
<tr>
<td>6.1 Glider’s Flight Simulation</td>
<td>24</td>
</tr>
<tr>
<td>6.2 Intensity Based Tracking</td>
<td>26</td>
</tr>
</tbody>
</table>
Chapter 1

Abstract

In this thesis, a vision estimation algorithm to aid the guidance system of an unmanned aerial vehicle (UAV) is proposed. Specifically, we are considering a glider UAV trying to reach a target zone. Since the UAV is falling fast, the scale of the images captured by the on-board camera grows. The vehicle dynamics and the control algorithm cause fast changes in the region seen by the camera and in the way it projects to the image plane. An algorithm based on SURF points and projective transformations is proposed. The algorithm estimates the target coordinate during the glider’s flight trajectory. Several simulations were performed on different types of scenarios. The simulations were used to characterize the type of zones where the algorithm is useful in the guidance of a glider. The results showed that the algorithm provides accurate position estimation in various types of scenarios, allowing the control system to guide the glider to the target region.
Chapter 2

Introduction

A glider is an unmanned aerial vehicle (UAV) without propulsion. It is usually carried on an aircraft, launched with given initial conditions, and then moves down due to the gravity acceleration. Usually, the vehicle has some control surfaces that allow it to arrive to different target locations. It also has sensors to measure the current flying state. A guidance system takes the measurements and moves the control surfaces to guide the glider to the target zone.

UAVs have the ability to perform dangerous tasks avoiding putting human lives at risks [1]. Unmanned gliders can be used in civil applications like in zone recognition of dangerous areas and rescue tasks. In military applications, there are other uses like unmanned aircraft that deliver cargo munitions in battle zones and unpowered rockets or guided bombs [2].

To accomplish any of the applications gliders need to be able to reach a target location. To do so, they need a navigation system. Navigation can be roughly described as the process of determining a suitable path between a starting and a goal point for a robot [3]. There are many possible purposes or needs for navigation in human activities, mostly used to track or locate the position of land, sea, airborne or space vehicles [4]. There are various alternatives to create navigation systems for gliders. Inertial and GPS navigation are the two most common alternatives. In inertial navigation, attitude rates and acceleration are measured and integrated overtime in order to track the object position and orientation, usually using an IMU (Inertial Measurement Unit). The problem with this strategy is that integration over time generates additive error, which leads to non-negligible estimation errors at the end of the gliders flight trajectory. Another alternative and the most common is the use of Global Positioning Systems (GPS). The
limiting factor on the effectiveness of this navigation system is often the accuracy of the coordinates provided as target points \[5\], especially when there are emerging targets. As a response to the above mentioned problems, during last decades, there has been a growing interest in developing computer vision algorithms to help with the navigation system of gliders. The works presented in \[1\] and \[5\] focused their attention in finding the target geo-location. In \[5\], the authors present an approach where multiple photos of the target zone are taken in a noncollinear large subtended angle. The algorithm uses the GPS coordinates of photos along with the camera calibration information to generate a 3D coordinate estimation of the target. The main drawback of this approach is that it requires a reconnaissance flight to take multiple photos in a specific fashion, before being able to launch the glider. In \[1\], the authors propose an algorithm to find the north-east-down (NED) geo-location of the target based on the pixel coordinates of the video frame and the use of a range finder. The algorithm assumes that a human specifies the target to track at a ground control station in a pixel coordinate frame. The algorithm tracks the coordinate using a Horn-Schunk optical flow method over a fusion of images taken from regular and thermal cameras. This method is capable of providing a 3D estimation of the target coordinate in real time.

Other alternatives exist for the guidance systems of gliders. The use of dissimilarity measures \[6\] or optical flow techniques \[7, 8\] is the most common approach to track a target in an image and has been applied directly to guide UAVs. The authors in \[9\] propose a vision algorithm to estimate the movement of a UAV using a combination of optical flow and Kalman filter. Their approach is divided in two parts; the first one estimates the location of the UAV using path integration of the optical flow measurement, and the second uses a Kalman filter to continuously estimate and correct the position errors. The simulated conditions in their work don’t consider the full six degrees of freedom of the UAV. The results presented in their work showed that the accuracy of the estimation is reduced when changes in altitude are added to the simulation.

In \[10\], the authors use an approach based on projective transformations using SIFT feature points. A reference image is previously processed and feature points are stored. Then, when the vehicle is flying, the online images are processed to look for matches with the points in the reference image. If matches are found, a RANSAC algorithm is used to find a projective transformation relating the reference image and the online captured one.

The current vision algorithms have some issues for the guidance of a glider. Traditional vision algorithms are not capable of handling the changes in the image due to the cam-
era movement and change in attitude during the flight of the glider. Some algorithms are designed to work under low degrees of freedom in the movement of the vehicle, but the movement of a glider has full six degrees of freedom. Some other algorithms require having a reference image to compare during the whole flight, but since the change in scale in this application is huge, that alternative doesn’t work.

To solve the above mentioned issues, we propose a vision only estimation algorithm to track a target coordinate in real time during the flight trajectory of a glider. The proposed approach receives the target designation online. The human operator intervenes once to select a target pixel from an image captured by the vehicle, and then the vision algorithm allows the bang-bang control to take the glider to the target zone. The vision algorithm provides accurate position estimation in different type of scenarios and under the movement of the glider.
Chapter 3

Problem Statement

Gliders require a navigation system to provide inputs to the guidance system. Most gliders use inertial navigation, GPS navigation or a combination of both. These navigation systems are useful but since often the coordinates provided as targets are no accurate, the guidance system cannot take the glider to the precise desired location. To overcome this problem, a method where the user specifies the input target coordinate in an image capture by the onboard glider’s camera is required. Also, a method with these characteristics facilitates escaping maneuvers for the aircraft in charge of launching the glider. The method has to provide accurate inputs to the control system, and has to be capable of maintaining its accuracy despite the six degrees of freedom of the glider’s movement.

Specifically, a vision based algorithm is required. The algorithm has to track a target coordinate during an image sequence captured by the onboard camera of a glider. The target coordinate is specified by a human operator in the first image of the sequence. The glider moves down due to the gravity acceleration, and the scale of the images captured by the onboard camera grows quickly. The type of image sequences and the estimation expected for a typical glider’s flight can be seen in Figure 3.1.

3.1 Aim

To design a computer vision algorithm that provides accurate inputs to the guidance system of a glider.
3.2 Objectives

- A computer vision algorithm to guide a glider to a target zone will be proposed.

- A simulation environment that allows the quantitative error estimation evaluation of the vision algorithm will be designed.

- Several computer simulations to test the accuracy of the algorithm will be performed.
Chapter 4

Solution

In this research, a method to track a target coordinate during the flight trajectory of an unmanned glider was proposed. The target is defined just before the glider is launched. Since the range of zones where the glider can be sent to is very diverse, it’s difficult to segment out the landing zone, and also impractical to create a database of possible targets.

A method to overcome the above mentioned problems, using SURF [11] feature points is proposed. The method estimates the target coordinate during the glider’s flight and helps guiding the glider to the target zone. Projective transformations obtained with the RANSAC algorithm are used to find the target coordinate during the flight trajectory. A block diagram of the algorithm can be seen in Figure 4.1.

![Figure 4.1: Block diagram of the algorithm](image)

[11]: Reference number
4.1 Feature Detection

In order to estimate the location of the target coordinate in the image sequence, a SURF feature detector is used to find correspondences between the reference image $I_{ref}$ and the current image $I_c$. Shortly, SURF can be described in two phases. The first one uses a Hessian matrix to find interest points (blob-like points). The second one uses a Wavelet Haar transformation to create a 64 vector that describes the region around the interest point. The descriptor contain information about the gradients’ orientation in the region. SURF uses masks of different sizes to find feature points in different scales. The filtering process is done using integral images, making this algorithm computationally efficient. The type of points obtained after processing a test image can be seen in Figure 4.2. A detailed description of the algorithm can be found in [11].

![Figure 4.2: Feature points obtained by SURF algorithm in a test image](image)

After calculating the feature points and its descriptors in $I_{ref}$ and $I_c$, correspondences between them need to be found. This process is usually known as matching and is described below.

We define the set of SURF feature points found in $I_{ref}$ as $P$, and the set of SURF feature points found in $I_c$ as $Q$. The matching of the points is done by using the next
Chapter 4. Solution

equation (a sum of square differences, SSD):

\[ e_{u,v} = \sum_{i=1}^{64} (\alpha_i(p_u) - \alpha_i(q_v))^2 \]  

(4.1)

where \( \alpha_i(p_u) \) is the \( i \)th descriptor of \( p_u \). \( \alpha_i(q_v) \) is the \( i \)th descriptor of \( q_v \). The summation is repeated for every \( p_u \) that belongs to \( P \), and every \( q_v \) that belong to \( Q \). At the end of the process a matrix of errors \( e_{u,v} \) is obtained.

For every \( p_u \), a point \( q_v \) is found by using the minimum of row \( u \) in \( e_{u,v} \). These pairs of points form a new set \( Z \) with tuples \( \{p_l, q_l\} \). The tuples with an error bigger than a threshold value are eliminated. The set \( Z \) contains the matched points of the two images.

This process was implemented using the Matlab Computer Vision Toolbox.

4.2 Transformation Calculation

The RANSAC [12] algorithm is used to determine the transformation that relates \( I_c \) with \( I_{ref} \). RANSAC is an algorithm created to fit a model to experimental data. RANSAC is capable of interpreting/smoothing data containing a significant percentage of gross errors, and is suited for our application where the correspondences obtained through the feature point detector are error-prone.

RANSAC is used to calculate a projective transformation that relates the point correspondences found in \( I_c \) and \( I_{ref} \). The equation that represents the transformation is described by:

\[ \bar{x}_c = A \bar{x} = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} \bar{x}_c \\ \bar{x} \end{bmatrix} \]  

(4.2)

where \( \bar{x}_c \) represents the homogeneous coordinates of a point in \( I_c \), \( \bar{x} \) represents the homogeneous coordinates of a point in \( I_{ref} \) and \( A \) is the projective matrix that takes coordinates in \( I_{ref} \) and converts them in coordinates in \( I_c \).
4.3 Point Estimation

The point estimation step takes the estimated model $A$ and the known coordinates in the reference image $\bar{x} = [X_{\text{ref}}, Y_{\text{ref}}, 1]^T$, and calculates the output coordinate $\bar{x}_c = A\bar{x}$ in $I_c$.

4.4 Deformation Indicator

According to [13] the uncertainty of an estimated transformation depends on many factors, including the number of points used to compute it, the accuracy of the given point matches, as well as the configuration of the points in question. In our application, when trying to use the first image as reference image during the whole flight path, the estimated coordinate was unstable due to various reasons: the low number of correct matched points that are present as input to the RANSAC algorithm; the low number of feature points found in last images that match points in the first image (because the viewed zone corresponds to a small region of the first image); the content of the feature vectors change because there’s a significant modification in the point and angle of view. Thus, an indicator of change in the characteristics of the images was proposed to decide when the reference image is replaced by a new one.

![Figure 4.3](image)

Figure 4.3: Transformation of the rectangle used by the deformation indicator under a projective transformation $A$

To create such indicator, the points $p_1, p_2, p_3, p_4$ are defined in the reference image as:

$$
p_1 = (0.25W, 0.25H) \quad p_2 = (0.75W, 0.25H) \\
p_3 = (0.75W, 0.75H) \quad p_4 = (0.25W, 0.75H)
$$
where \( W, H \) are the width and the height of the image respectively. The points \( p'_1, p'_2, p'_3, p'_4 \) are calculated using the estimated transformation as \( p_n' = Ap_n \). The deformation measurement is defined as:

\[
def = |d'_1 \cdot d'_2| + |d'_2 \cdot d'_3| + |d'_3 \cdot d'_4| + |d'_4 \cdot d'_1|
\]

(4.4)

where \( d'_1, d'_2, d'_3, d'_4 \) are the direction vectors in the current image defined as:

\[
d'_1 = \frac{p'_2 - p'_1}{\|p'_2 - p'_1\|}
\]

\[
d'_2 = \frac{p'_3 - p'_2}{\|p'_3 - p'_2\|}
\]

\[
d'_3 = \frac{p'_4 - p'_3}{\|p'_4 - p'_3\|}
\]

\[
d'_4 = \frac{p'_1 - p'_4}{\|p'_1 - p'_4\|}
\]

(4.5)

The indicator increases when the angle between two direction vectors is different than 90°. Since the region considered to calculate the feature vectors in SURF algorithm don’t consider the changes in the viewing angle or point of view, the proposed indicator acts as a quantification of the degradation of the feature vector obtained by SURF. Also, the indicator changes with the modification of the distribution of the points in the image, which means that the uncertainty of the transformation estimation has been modified. In our algorithm, the reference image is changed when a threshold value is exceeded.

In [14], the authors use the number of matched points over the number of points found in the reference image, as an indicator of the uncertainty of the estimated transformation. But, the indicator presented here reflects better the characteristics of uncertainty described in [13] and analyzed at the beginning of this section.
Chapter 5

Simulation Design and Implementation

A simulation environment to test the behavior of the proposed solution was designed and implemented. The simulation environment needed to fulfill various requirements. First, the movement of the simulated camera has to be as close as possible to a camera mounted on a real glider. Second, the rendering of the images need to reflect the actual movement of the glider’s camera considering the three translation axes and the three attitude angles. Third, the simulation has to allow a quantitative evaluation of the estimation error.

To accomplish these requirements, the stages shown in Figure 5.1 were considered. A simulink dynamic model of the glider presented in [2] was implemented. The results of the simulation were used as input in a VRML model controlled through the Matlab 3D Animation Toolbox to export a video of the trajectory. The video was processed with the implementation of the proposed solution. The proposed solution was implemented using the Matlab Computer Vision Toolbox. And finally, the evaluation of the algorithm was implemented in a MATLAB script using the results of the proposed solution and the dynamic simulation.

5.1 Glider’s Dynamic Model

To create a realistic simulation of the images captured in a glider’s flight, a dynamic model of the system was implemented. The model presented in [2] was used. There,
the authors propose a dynamic model of an unmanned glider based on the description done in [15]. The equations describing the model are:

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{z} \\
\dot{u} \\
\dot{v} \\
\dot{w} \\
\dot{\phi} \\
\dot{\theta} \\
\dot{\psi} \\
\dot{P} \\
\dot{Q} \\
\dot{R}
\end{bmatrix} =
\begin{bmatrix}
-\psi s\theta u + (\psi s\phi s\theta - c\phi s\psi)v + (s\phi s\psi + c\phi c\psi s\theta)w \\
\phi s\psi u + (\phi c\psi + s\phi s\psi s\theta)v + (c\phi \psi s\theta - c\psi s\phi)w \\
-\phi s\theta + C_x(\alpha, \beta, \delta_r, \delta_e)qS/m + Rv - Qw \\
gs\phi c\theta + C_y(\alpha, \beta, \delta_r, \delta_e)qS/m + Pw - Ru \\
gc\phi c\theta + C_z(\alpha, \beta, \delta_r, \delta_e)qS/m + Qu - Pw \\
P + Qs\phi t\theta + Rc\phi t\theta \\
c\phi Q - s\phi R \\
Qs\phi/c\theta + Rc\phi/c\theta \\
QR(I_{yy} - I_{zz})/I_{xx} \\
(C_m(\alpha, \delta_e)qSc + PR(I_{zz} - I_{xx}))/I_{yy} \\
(C_n(\beta, \delta_r)qSb + PQ(I_{xx} - I_{yy}))/I_{zz}
\end{bmatrix}
\tag{5.1}
\]

Where \([x, y, z]^T\) is the position vector of the glider respect to an earth fixed coordinate system, \([u, v, w]^T\) are the body frame velocities, \([\phi, \theta, \psi]^T\) are the three Euler angles describing the glider orientation and \([P, Q, R]^T\) are the body frame angular velocities. The body frame velocities \([u, v, w]^T\) go in the direction of \([X_b, Y_b, Z_b]^T\) respectively. The control inputs are the angle of the elevator (\(\delta_e\)) and the angle of the rudder (\(\delta_r\)).

Figure 5.2 shows the coordinate systems involved in the simulation. The glider’s attitude is specified by first rotating about the \(Z_E\) axis (yaw angle \(\psi\)), then about the
rotated $Y_E$ axis (pitch angle $\theta$), and finally about the rotated $X_E$ axis (roll angle $\phi$).

The rotation matrix that expresses the vectors of the final frame in terms of components of the initial frame may be obtained by post multiplication of the rotation matrices in the following order: $R = R_{Z_E}(\psi)R_{Y_E}(\theta)R_{X_E}(\phi)$

Since the camera on-board of the glider is aero-stabilized, the camera’s attitude depends on the earth fixed coordinate velocities and not in the glider’s attitude. The following equations describe the camera’s orientation angles.

$$\theta_1 = \text{atan2} \left( \frac{V_{Y_E}}{V_{X_E}} \right) \tag{5.2}$$

$$\theta_2 = \text{atan2} \left( \frac{V_{Z_E}}{\sqrt{V_{X_E}^2 + V_{Y_E}^2}} \right) \tag{5.3}$$

where $[V_{X_E}, V_{Y_E}, V_{Z_E}]^T$ is the velocity vector of the glider respect to the earth fixed coordinate system. The camera’s attitude is specified by first rotating about the $Z_E$ axis (yaw angle $\theta_1$), then about the rotated $Y_E$ axis (pitch angle $-\theta_2$), and finally about
the rotated $X_E$ axis (roll angle $\phi$).

### 5.1.1 Bang-Bang Control

To guide the glider to a target coordinate $P_t = [X_t, Y_t, Z_t]^T$ a bang-bang control was used. When the camera is looking at the target point, the controller moves the elevator and rudder to attempt to align the projection of $P_t$ to the center of the image. Figure 5.3 shows the control signals generated by the controller depending on the location of the target coordinate.

![Control signals generated by the bang-bang algorithm depending on the image location of the target](image)

Figure 5.3: Control signals generated by the bang-bang algorithm depending on the image location of the target

A projective camera was simulated as described in [13] to project the target coordinate to the image plane. A description of the process involved in this part can be found in subsection 5.4.1.

### 5.2 Scene Rendering

The Simulink 3D Animation Toolbox was used to generate the images captured by the glider’s on-board camera during the flight trajectory. Simulink 3D Animation Toolbox uses VRML worlds to create simulations. Figure 5.4 shows the two coordinate systems involved in the simulation. The results of the dynamic model simulation in the earth coordinate system are used to render the images using the VRML coordinate system.
To simulate the camera movement in VRML equivalences between the position and attitude of the camera in the two coordinate systems must be found. An equivalent movement in VRML coordinates was defined. The position in VRML coordinates is \( \{X_V = Y_E, Y_V = -Z_E, Z_V = -X_E\} \). The rotation matrix used to align the camera coordinate system to the VRML Coordinate System is defined as:

\[
R = R_Y(-a_2)R_X(-a_1)R_Z(-a_3)
\]  

(5.4)

Where \( a_1 = \theta_2, a_2 = \theta_1, a_3 = \phi \).

These coordinate transformations allow to generate the videos (image sequences) of the trajectories flown by the glider. Different types of image textures were used to create a realistic simulation environment.

### 5.3 Solution Implementation

The proposed solution was implemented using the Matlab Computer Vision Toolbox. A complete description of the algorithm is found in chapter 4.
5.4 Algorithm Evaluation

To quantitatively evaluate the algorithm, measures of error in the earth fixed coordinate system and the image plane were calculated. To do this, the following process is necessary:

- Transform the user input in image coordinates to an earth fixed frame coordinate.
- Simulate the glider’s dynamic model as described in section 5.1 using as target coordinate the earth coordinate of the user input.
- Render the scene using the process described in section 5.2.
- Run the proposed solution implemented as described in chapter 4.
- Evaluate the error in the earth fixed frame and in image coordinates (pixels). To do this two steps are required:
  - Transform the image coordinate estimations done by the proposed solution into earth fixed coordinates. This allows to calculate the error in the earth fixed frame.
  - Project the target coordinate in the earth fixed frame to the image plane. This allows to calculate the error in the image plane (in pixels).

5.4.1 Transforming Earth Coordinates to Image Coordinates

A point in the earth fixed coordinate system can be projected to the image plane by using the Camera Projection Matrix $P$ as shown in the following equation:

$$x = KR[I|\bar{C}]X = PX$$

(5.5)

where $X$ is the earth coordinate to project. $x$ is the homogeneous image coordinate of the projected point. $\bar{C}$ is the earth coordinate of the camera center. $R$ is the rotation matrix that aligns the camera coordinate system with the earth fixed coordinate system. $K$ is the calibration matrix, and is defined in our application as:

$$K = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(5.6)
Where $f$ represents the focal length of the simulated camera. In [10] a discussion of how to handle the focal length in the projection of points is done. In this research, the focal length was defined as: $f = 1/(\tan(\text{fov}/2))$, where $\text{fov}$ is the field of view of the camera in radians. This implies that the pixel coordinates were normalized so that they go from $[-1; 1)$ along the longer image dimension and $[-a^{-1}; a^{-1})$ along the shorter axis, where $a \geq 1$ is the image aspect ratio. To this the image coordinates are transformed using the following equations:

$$
x'_s = (2x_s - W)/S
$$
$$
y'_s = (2y_s - H)/S
$$
$$
S = \max(W, H)
$$
(5.7)

And finally, the rotation matrix $R$ is defined as:

$$
R = \begin{bmatrix}
0 & -1 & 0 \\
0 & 0 & -1 \\
1 & 0 & 0
\end{bmatrix} R_X(-\phi)R_Y(\theta_2)R_Z(-\theta_1)
$$
(5.8)

where $R_A(\alpha)$ represents a rotation of $\alpha$ radians about the $A$ axis.

### 5.4.2 Transforming Image Coordinates to Earth Coordinates

As described in last subsection, the matrix $P$ can be used to project a earth coordinate $\mathbf{X}$ to an image coordinate $\mathbf{x}$. A new matrix $\bar{P}$ known as the Homogeneous Camera Projection Matrix is defined as:

$$
\bar{P} = \begin{bmatrix}
P & 0_{3x1} \\
0_{1x3} & 1
\end{bmatrix}
$$
(5.9)

This matrix does the same than $P$ matrix, the only difference is that the input coordinate is now $\bar{\mathbf{X}}$ the homogeneous earth coordinate.

To transform an image coordinate into a world coordinate the inverse of $P$ is required.

$$
p_w = \bar{P}^{-1} \begin{bmatrix}
\bar{\mathbf{x}} \\
1
\end{bmatrix}
$$
(5.10)
The homogeneous world coordinate $p_w$ (green dot in Figure 5.5) is not necessarily in $Z_E = 0$. But, $p_w$ is in a line that passes through the camera center and the real point in earth coordinates (blue cross in Figure 5.5). A line $q_w = \overline{C} + (\overline{C} - p_w)t$ is used to find the real point in earth coordinates. The parameter $t$ is found using $t = \overline{C}_z/((\overline{C}_z - p_w_z))$. $q_w$ is the earth coordinate.

This transformation process was used to transform the user input to earth coordinates and allow the simulation of the glider’s dynamic model. It was also used to transform the image coordinates obtained using the vision algorithm to earth coordinates and calculate the earth frame errors.
Chapter 6

Algorithm Evaluation

To evaluate the algorithm various tests were done. First a complete simulation of the glider’s flight is presented. Second, for the simulated trajectory, the usage of an intensity based tracking algorithm is studied, showing that the obtained results are not good enough to guide a glider to a target region. Third, the proposed vision estimation algorithm is evaluated in the same trajectory. The results show that the estimation done by the algorithm is accurate and that the algorithm is suited for the application. Fourth, an evaluation of the algorithm in different type of scenarios is performed. Fifth, an evaluation of the improvement generated by the use of the deformation indicator is presented. And finally, a comparison with the approach in [9] is done.

6.1 Glider’s Flight Simulation

In this section, the results of releasing the glider from \([X_{E_0} = 0\ m, Y_{E_0} = 0\ m, -Z_{E_0} = 1524\ m]\) are presented. We set \([u_0 = 102.89\ m/s, v_0 = 0\ m/s, w_0 = 0\ m/s], [\phi_0 = 0^\circ, \theta_0 = -10^\circ, \psi_0 = 0^\circ]\) and \([P_0 = 0^\circ/s, Q_0 = 0^\circ/s, R_0 = 0^\circ/s]\) as initial conditions. The target zone was located in \([X_t = 2200, Y_t = 490, Z_t = 0]\). The simulation showed that the glider landed in \([X_f = 2200.97\ m, Y_f = 488.84, Z_f = 0]\), which is just 1.52\ m away from the desired location.

Figure 6.1a shows the flown trajectory of the glider. Figure 6.1b shows the attitude of the glider. The yaw and pitch angles change with time allowing the vehicle moving toward the target. Since the camera is aero-stabilized, its orientation (Figure 6.1c) de-
depends on the earth fixed frame velocities of the glider, which makes the changes in the camera’s attitude smoother than the ones in the glider’s orientation. However, camera’s attitude changes are not smooth during the whole trajectory, this is due to the stronger moments generated by the controls signals when the vehicle is moving faster. Figure 6.1d shows the control signals generated using the bang-bang control and the estimation algorithm, the $\delta_e$ and $\delta_r$ angles change to align the projection of $P_t$ to the center of the image.
6.2 Intensity Based Tracking

Tracking a moving target is usually done by using intensity comparison techniques, like optical flow or image similarity measures. The most common algorithm in this line of work is the Lukas-Kanade optical flow algorithm [7]. Since the application of this research is also the tracking of a target, an implementation of the Lukas-Kanade algorithm presented in [8] was studied. The results of the simulation under the trajectory presented in section 6.1 are presented below.

Figure 6.2: Intensity Based Tracking Results

Figure 6.2a and Figure 6.2b show that in the seventh frame of the simulation (0.7s after the launching of the glider), the target position estimation slides to another image
zone that looks similar but it’s in a different location than the original. Accidentally, the estimation returns to the correct zone after 43 frames. But at the end of the trajectory, the estimation moves to other zones. Due to the instability of the estimation, this algorithm is not suited to guide a glider to a target point.

Figure 6.2a shows that the position estimation error of the target zone at the end of the trajectory is 208.62m, which is far from the desired target location. Figure 6.2c shows a comparison between the estimated coordinate and the real coordinate in image pixel coordinates. The estimated coordinate slides to a different location at the end of the trajectory. If this estimation were used to guide a glider, the control system would take it to a zone that is not the target.

Traditional image intensity comparison techniques have additional problems. If the target zone lies outside of the viewed zone, then an estimation is no longer possible. Most of the scenarios have textures that are almost even, which causes problems when comparing intensities. These problems are bigger if illumination changes occur. The use of feature detectors reduces the problem of almost even textures, because they use the regions’ gradients which are more distinctive than the intensity values.

### 6.3 Vision Estimation Results

In this section, the results of the proposed estimation algorithm over the trajectory described in section 6.1 are presented. The evaluation was done in various ways. First a qualitative evaluation of the estimation was performed. Videos were created showing the estimation during the whole flight. In this report, the first and last images of the video are shown. Second, the estimation error at the end of the trajectory in the earth fixed frame is calculated. Finally, the error in pixels of the estimated coordinate compared to the ideal location of the projection of $P_t$ is computed.

Figure 6.3 shows the qualitative results of the estimation algorithm. The qualitative result shows that after processing the images captured by the glider during the flight trajectory, the target zone remains the same.

Figure 6.4 shows the estimation errors in the earth fixed frame. The light blue lines show the frames where the reference image was changed, which means that the threshold of the deformation indicator was exceeded.
Figure 6.3: Qualitative Result of the Proposed Estimation Algorithm

(a) First image, the red cross indicates the user input coordinate

(b) Last image, the red cross indicates the estimated coordinate at the end of the trajectory

Figure 6.4: Estimation error in the two horizontal axes \([X_E, Y_E]\) in the earth fixed frame

Figure 6.5 shows the estimation errors in the image plane. The processed images are 800x600 pixels. Due to the smooth movements of the glider during the first seconds of the flight, the estimation error increases slowly as shown in Figure 6.4 and Figure 6.5. At the end of the trajectory, the moments generated by the changes in \(\delta_e\) and \(\delta_r\) are stronger, and that causes stronger changes in the glider’s and camera’s attitude. Something similar happens with the increase in scale. Since the viewing area reduces with time and the velocity increases, the scale increases faster at the end of the trajectory. Another factor that makes the estimation harder toward the end is the oscillation of the roll angle \(\phi\).

To analyze Figure 6.4 and Figure 6.5, it is important to note that the area represented by a pixel at the beginning of the trajectory is bigger than the area represented at
Chapter 6. Algorithm Evaluation

Figure 6.5: Estimation error in the image plane

The end. This means, that even when the error in the fixed coordinate system doesn’t increase with time, the error in pixels could increase. For this application, it’s more important to look at the error in the fixed coordinate system, because it represents the measurement error to which the control system is subjected to.

The figures above also show the behavior of the deformation indicator. Since at the beginning of the trajectory the changes in the image are smooth, the reference image is kept for longer periods of time than at the end of the trajectory. An evaluation of the indicator is presented in section 6.5.

Figure 6.6: Real vs Estimated Image Coordinates

Figure 6.6 shows the comparison between the ideal projection of \( P_t \) versus the es-
timated coordinate using our algorithm. The bang-bang controller attempts to keep the projection of \( P_t \) in the center of the image \( [X = 400, Y = 300] \). During most part of the trajectory, the vision algorithm is highly accurate, but toward the end of the trajectory small differences between the ideal and estimated coordinate appear. For the application in question, it’s more important to have a good estimation during the first part of the flight, because at the end, the changes in the control signals \( \delta_e \) and \( \delta_r \) may cause instability to the system.

The results of our approach are suited to guide a glider to the target zone, because there’s no instability in the estimation and the final estimation error is low.

### 6.4 Evaluation in different scenarios

To define the type of zones where a glider could be launched and guided with our algorithm, tests to evaluate the performance in different types of scenarios were done using 20 different high resolution images as textures. We characterize the scenarios as: forests, plains, urban and river side zones as shown in Figure 6.7. The analyzed scenarios are squares of side 10km.

![Image](image_url)

Figure 6.7: The four types of scenarios considered to evaluate the proposed algorithm

A total of 400 simulations were performed, corresponding to 20 simulations in each scenario. Every simulation of the same scenario had a different target zone. For each simulation the glider was launched from \( [X_{E_0} = 0m, Y_{E_0} = 0m, -Z_{E_0} = 1524m] \), and we set \([u_0 = 102.89m/s, v_0 = 0m/s, w_0 = 0m/s], [\phi_0 = 0^\circ, \theta_0 = -10^\circ, \psi_0 = 0^\circ] \) and \([P_0 = 0^\circ/s, Q_0 = 0^\circ/s, R_0 = 0^\circ/s] \) as initial conditions. The estimation error of the algorithm at the end of the trajectory was calculated and is presented below.
Figure 6.8: Box-plot of the position estimation error by type of scenario. For each type of scenario 100 simulations were performed.

Figure 6.8 shows the results of the 400 simulations in a box-plot. The red line in each box indicates the median of the estimation error. The points marked by a + sign are the errors considered as outliers according to the normal distribution of the errors in each scenario. The biggest error presented in all the simulations was 20 meters.

The algorithm behaves better in urban scenarios. Urban scenarios present a lot of blob-like regions, thus the number of feature points found by the SURF algorithm is high. Also, the image textures of urban zones are diverse which makes the descriptors distinctive.

In the case of river side zones and forests, the algorithm has a good behavior, but more outliers are presented. Forests present a lot of blob-like regions and have many feature points, but the textures of the points found by the SURF algorithm are very similar, making similar their descriptors. This makes the transformation harder to estimate and may increase the estimation error.

Plains contain fewer blob like regions and thus fewer feature points, but generally the textures of the feature points are distinctive and differ between them. So, despite that the number of features is smaller than in the other scenarios, these features are more useful to calculate the transformation relating two images. The estimation error outliers presented in these simulations were all due to the same texture, where the scene was a desert-like plain, and in this special case the feature descriptors were very similar causing bigger estimation errors.

The proposed algorithm is not suited to work under desert scenarios due to the low number of blob-like regions and to the similarity of the descriptors found in those
zones. Several simulations in this type of scenarios were performed, and we found that
the number of matched points is low and that the RANSAC algorithm has more outliers
than inliers. These two factors could be considered as an indicator of error to avoid
using a bad estimation in the guidance of a glider.

6.5 Deformation Indicator Evaluation

In order to evaluate the performance of the proposed deformation indicator as a method
to define when to change the reference image, a set of 160 simulations was performed.
80 simulations considered the indicator, and the other 80 changed the reference image
every iteration. Figure 6.9 shows a box-plot of the obtained results. The box plot
shows that the median estimation error obtained when using the deformation indicator
is smaller than the median error obtained when changing the reference image every
iteration. This means, that in most simulations the use of the indicator helps reducing
the estimation error at the end of the trajectory.

![Box-plot comparison](image)

Figure 6.9: Comparison of the estimation error between the vision algorithm using the
deformation indicator and the vision algorithm that changes the reference image every
iteration. For each case 80 simulations were performed.
6.6 Comparison to other vision algorithm for UAV position estimation

Our final test was to compare the proposed method with another published navigation system for UAVs. In the compared system [9], the authors use a combination of optical flow and a Kalman filter to track a target coordinate. They call it OFNS/OFAN system (Optical Flow Navigation System / Optical Flow Aided Navigation).

The authors performed various simulations to test the OFNS/OFAN system. In the first one they consider the UAV flying along two axes (x and y) without altitude variation. The results obtained in their simulation are shown in Figure 6.10. The maximum magnitude of the position error is around $5m$ in each axis.

![Position Error of the OFNS/OFAN](image)

Figure 6.10: Position errors of the compared algorithm in a 2D movement simulation along x and y axes. The red line represents the error along the x axis. The blue line represents the error along the y axis. Image taken from [9]

They also performed a 3D simulation. In this simulation the UAV moves in the three axes, but it doesn’t change the attitude of the vehicle, neither the attitude of the camera. The results of this simulation are shown in Figure 6.11. The OFNS/OFAN system can provide an estimation of the altitude of the vehicle, using two height reference images to measure height changes. The maximum error in the two horizontal axes is close to $25m$. And the maximum estimation error in the vertical axis is close to $50m$. 
The simulations performed to test our algorithm had more degrees of freedom in the movement of the vehicle. We not only considered the three translation axes, but also the three angles of the camera’s attitude. The performed simulations showed that in a typical trajectory in urban environments, the median error was 2.5m, which is an order of magnitude smaller than the error obtained in the compared work (25m). For the simulation of the compared work a urban like image was used as a texture. Also, an evaluation of our algorithm in other type of scenarios was performed, we characterized that the best behavior is in urban like zones. But, even in other type of zones, the results showed that our median estimation error was always smaller than 3m, and in the worst case was 20m. Our position estimation error results are comparable to the estimation errors in the simple simulation of the 2D translation of the OFNS/OFAN system, which shows that for the considered application our algorithm can provide a better input to a control system.
Chapter 7

Conclusion

This research presented a method to track a target coordinate for an unmanned glider. The tracked coordinate was used in a bang-bang control system to guide the glider. The proposed algorithm consists of a feature point detector followed by a RANSAC algorithm to estimate a projective transformation. A deformation indicator was added to the system to decide when the reference image is replaced. The simulation results showed that the algorithm is capable of providing accurate inputs to the guidance system.

One of the advantages of the proposed algorithm is that it only requires the target coordinate to be specified in one image. This extends the applications of gliders to situations where specifying the GPS target coordinate is difficult. It also facilitates escaping maneuvers for the aircraft in charge of releasing the glider.

The algorithm was evaluated under urban, plain, river side and forest scenarios. The results showed that the algorithm behaves correctly in all of them, but its best behavior is in the urban regions, because they contain many blob-like regions and their textures make the descriptors more distinctive.

An evaluation of the deformation indicator under urban zones was performed. The results were compared to changing the reference image every frame, showing that the indicator helps reducing the estimation error at the end of the flown trajectories.

The results of the proposed algorithm were compared to the OFNS/OFAN system presented in [9]. The simulations showed that the magnitude of the typical position estimation error in our algorithm are smaller than in their approach. Also, the quality of their estimation deteriorates when changes in altitude are introduced, and they don’t consider changes in the camera’s attitude. Our proposed algorithm has smaller
estimation error, and includes the changes in the camera’s attitude and altitude. Until now, the algorithm has been tested under high visibility situations. This leads us to conclude that this algorithm works when there is visibility from the launching to the target point. To avoid interference caused by weather conditions like rain or clouds, near-infrared or infrared images could be considered. This would extent the applicability of the approach.

As future work, we intend to focus on the problem of considering scenes that are not quasi-planar. Now, the algorithm can work in geographical zones that are almost even. To consider rugged regions we need to rethink the transformation between images. Other lines of work are considering applications like the generation of trajectories based on the vision algorithm for unmanned quadrotors or unmanned helicopters.
Appendix A

SURF

SURF (Speeded-Up Robust Features) is a fast and robust algorithm for local, similarity invariant representation and comparison. SURF selects interest points of an image from the salient features of its linear box-space, a series of images obtained by the convolution of the initial image with box filters at several scales. Then SURF builds local features based on the histograms of gradient-like local operators. The main interest of the SURF approach lies in its fast computation of operators in the box-space, enabling real-time applications such as tracking and object recognition [17].

The SURF algorithm is composed of two phases:

1. Interest point detection

2. Building the descriptor associated with each interest points

A.1 Integral Images

Much of the performance increase in SURF can be attributed to the use of an intermediate image representation known as the Integral Image [18]. The integral image is computed rapidly from an input image and is used to speed up the calculation of any upright rectangular area. Given an input image $I$ and a point $(x, y)$ the integral image $I_{\sum}$ is calculated by the sum of the values between the point and the origin. Formally this can be defined by the formula:

$$I_{\sum}(x, y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{j \leq y} I(x, y)$$  \hspace{1cm} (A.1)
Using the integral image, the task of calculating the area of an upright rectangular region is reduced four operations. If we consider a rectangle bounded by vertices A, B, C and D as in Figure A.1, the sum of pixel intensities is calculated by:

$$\sum = A + B - (C + B)$$  \hfill (A.2)

Since computation time is invariant to change in size this approach is particularly useful when large areas are required. SURF makes good use of this property to perform fast convolutions of varying size box filters at near constant time.

Figure A.1: Area computation using integral images

### A.2 Interest Point Detection

The SURF detector is based on the determinant of the Hessian matrix. The Hessian matrix, $H$, is the matrix of partial derivatives of the function $f$.

$$H(f(x, y)) = \begin{bmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{bmatrix}$$  \hfill (A.3)

The determinant of this matrix, known as the discriminant, is calculated by:

$$det(H) = \frac{\partial^2 f}{\partial x^2} \frac{\partial^2 f}{\partial y^2} - \left( \frac{\partial^2 f}{\partial x \partial y} \right)^2$$  \hfill (A.4)

The value of the discriminant is used to classify the maxima and minima of the function by the second order derivative test. If the determinant is negative then the eigenvalues have different signs and hence the point is not a local extremum; if it is positive then either both eigenvalues are positive or both are negative and in either case the point is
classified as an extremum. Translating this theory to work with images rather than a continuous function is a fairly trivial task. First we replace the function values \( f(x, y) \) by the image pixel intensities \( I(x, y) \). Next, a method to calculate the second order partial derivatives of the image. The second order scale normalized Gaussian is the chosen filter as it allows for analysis over scales as well as space. We can construct kernels for the Gaussian derivatives in \( x, y, xy \) direction such that we calculate the four entries of the Hessian matrix. Use of the Gaussian allows us to vary the amount of smoothing during the convolution stage so that the determinant is calculated at different scales. Furthermore, since the Gaussian is an isotropic function, convolution with the kernel allows for rotation invariance. We can now calculate the Hessian matrix, \( H \), as function of both space \( \mathbf{x} = (x, y) \) and scale \( \sigma \).

\[
H(\mathbf{x}, \sigma) = \begin{bmatrix}
L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\
L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma)
\end{bmatrix}
\]  
(A.5)

Where \( L_{xx}(\mathbf{x}, \sigma) \) refers to the convolution of the second order Gaussian derivative \( \frac{\partial^2 g(\sigma)}{\partial x^2} \) with the image at point \( \mathbf{x} = (x, y) \) and similarly for \( L_{yy} \) and \( L_{xy} \). These derivatives are known as Laplacian of Gaussians.

Since calculating \( H \) is too slow, an approximation to the Hessian matrix by using box filter representations of the respective kernels is proposed. Figure 2 illustrates the similarity between the discretized and cropped kernels and their box filter counterparts. Considerable performance increase is found when these filters are used in conjunction with the integral images.

Since the Hessian matrix is calculated using an approximation, the determinant of the

Figure A.2: Laplacian of Gaussian Approximation. Top Row: The discretized and cropped second order Gaussian derivatives in the x, y and xy-directions. We refer to these as \( L_{xx}, L_{yy}, L_{xy} \). Bottom Row: Weighted Box filter approximations in the x, y and xy directions. We refer to these as \( D_{xx}, D_{yy}, D_{xy} \). Figure taken from [19]
Hessian is corrected by using the following formula:

\[ \text{det}(H_{\text{approx}}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \]  \hspace{1cm} (A.6)

The determinant here is referred to as the blob response at location \( \mathbf{x} = (x, y, \sigma) \). The search for local maxima of this function over both space and scale yields the interest points for an image.

Figure A.3: Filter Pyramid. The traditional approach to constructing a scale-space (left). The image size is varied and the Gaussian Filter is repeatedly applied to smooth subsequent layers. The SURF approach (right) leaves the original image unchanged and varies only the filter size. Figure taken from [19]

To finally define interest points, the Hessian determinant is thresholded. Points below the predefined threshold value are removed. After thresholding, a non-maximal suppression is performed. To do this each pixel in the scale-space is compared to its 26 neighbors, comprised of the 8 points in the native scale and the 9 in each of the scales above and below. Figure A.4 illustrates the non-maximal suppression step.

Figure A.4: Non-Maximal Suppression. The pixel marked X is selected as a maxima if it’s greater than the surrounding pixels on its interval and intervals above and below. Figure taken from [19]
A.3 Interest Point Descriptor

The SURF descriptor describes how the pixel intensities are distributed within a scale dependent neighborhood of each interest point detected. Integral images used in conjunction with filters known as Haar wavelets are used in order to increase robustness and decrease computation time. Haar wavelets are simple filters which can be used to find gradients in the $x$ and $y$ directions.

Extraction of the descriptor can be divided into two distinct tasks. First each interest point is assigned a reproducible orientation before a scale dependent window is constructed in which a 64-dimensional vector is extracted.

To determine the orientation, Haar wavelet responses of size $4\sigma$ are calculated for a set of pixels within a radius of $6\sigma$ of the detected point, where $\sigma$ refers to the scale at which the point was detected. The specific set of pixels is determined by sampling those from within the circle using a step size of $\sigma$.

The responses are weighted with a Gaussian centered at the interest point. In keeping with the rest the Gaussian is dependent on the scale of the point and chosen to have standard deviation $2.5\sigma$. Once weighted the responses are represented as points in vector space, with the $x$-responses along the abscissa and the $y$-responses along the ordinate. The dominant orientation is selected by rotating a circle segment covering an angle of $\pi/3$ around the origin. At each position, the $x$ and $y$-responses within the segment are summed and used to form a new vector. The longest vector lends its orientation the interest point. This process is illustrated in Figure A.6.

The first step in extracting the SURF descriptor is to construct a square window around the interest point. This window contains the pixels which will form entries in the descriptor vector and is of size $20\sigma$, where $\sigma$ refers to the detected scale. Fur-
thermore the window is oriented along the found direction. The descriptor window is divided into $4 \times 4$ regular subregions. Within each of these subregions Haar wavelets of size $2\sigma$ are calculated for 25 regularly distributed sample points. If we refer to the $x$ and $y$ wavelet responses by $dx$ and $dy$ respectively then for these 25 sample points (i.e. each subregion) we collect:

$$v_{\text{subregion}} = \left[ \sum dx, \sum dy, \sum |dx|, \sum |dy| \right]$$  \hspace{1cm} (A.7)

The resulting SURF descriptor is invariant to rotation, scale, brightness and, after reduction to unit length, contrast.

Most of the information presented in this section has been taken from [19].
Figure A.7: Descriptor Components. The green square bounds one of the 16 subregions and blue circles represent the sample points at which we compute the wavelet responses. As illustrated the x and y responses are calculated relative to the dominant orientation. Figure taken from [19].
Appendix B

RANSAC

The RANSAC (RANdom SAmple Consensus) algorithm is a general parameter estimation approach designed to cope with a large proportion of outliers in the input data. RANSAC is a resampling technique that generates candidate solutions by using the minimum number observations (data points) required to estimate the underlying model parameters. As described in [12], unlike conventional sampling techniques that use as much of the data as possible to obtain an initial solution and then proceed to prune outliers, RANSAC uses the smallest set possible and proceeds to enlarge this set with consistent data points.

The basic algorithm is summarized as follows:

1. Select randomly the minimum number of points required to determine the model parameters.

2. Solve for the parameters of the model.

3. Determine how many points from the set of all points fit with a predefined tolerance $\epsilon$.

4. If the fraction of the number of inliers over the total number points in the set exceeds a predefined threshold $\tau$, re-estimate the model parameters using all the identified inliers and terminate.

5. Otherwise, repeat steps 1 through 4 (maximum of $N$ times).

The number of iterations, $N$, is chosen high enough to ensure a specific value of the probability $p$ (usually set to 0.99) that at least one of the sets of random samples does
not include an outlier. Let $u$ represent the probability that any selected data point is an inlier and $v = 1 - u$ the probability of observing an outlier. $N$ iterations of the minimum number of points denoted $m$ are required, where

$$1 - p = (1 - u^m)^N$$  \hspace{1cm} (B.1)$$

and thus

$$N = \frac{\log(1 - p)}{\log(1 - (1 - v)^m)}$$  \hspace{1cm} (B.2)$$
Bibliography


