NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

# RUNWAY DETECTION FROM MAP, VIDEO AND AIRCRAFT NAVIGATIONAL DATA

by

Jose R. Espinosa Gloria

March 2016

Thesis Advisor: Roberto Cristi
Co-Advisor: Oleg Yakimenko

| | | |
|---|---|---|
| **REPORT DOCUMENTATION PAGE** | | *Form Approved OMB No. 0704–0188* |

| | | |
|---|---|---|
| **1. AGENCY USE ONLY** ***(Leave blank)*** | **2. REPORT DATE** March 2016 | **3. REPORT TYPE AND DATES COVERED** Master's thesis |
| **4. TITLE AND SUBTITLE** RUNWAY DETECTION FROM MAP, VIDEO AND AIRCRAFT NAVIGATIONAL DATA | | **5. FUNDING NUMBERS** |
| **6. AUTHOR(S)** Jose R. Espinosa Gloria | | |
| **7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Naval Postgraduate School Monterey, CA 93943-5000 | | **8. PERFORMING ORGANIZATION REPORT NUMBER** |
| **9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)** N/A | | **10. SPONSORING / MONITORING AGENCY REPORT NUMBER** |

**11. SUPPLEMENTARY NOTES** The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB Protocol number ____N/A____.

| | |
|---|---|
| **12a. DISTRIBUTION / AVAILABILITY STATEMENT** Approved for public release;distribution is unlimited | **12b. DISTRIBUTION CODE** |

**13. ABSTRACT (maximum 200 words)**

As part of the reinforcement of operations performed by the Mexican Navy, unmanned aerial vehicles (UAV) have been equipped with daylight and infrared cameras. Processing the video information obtained from these devices opens the door to a number of strategic opportunities. By recognizing patterns in visual sources, we address one problem in particular: how to achieve corresponding runways using a map and an individual frame of video.

An approach to runway detection using two tools is presented in this thesis. The first tool is a geographical information program, which is used to set the runway map we want to detect. The second tool is a video frame of the same runway, recorded in a camera mounted on a UAV. The needed equations and related algorithms are developed and tested on a simulation that reconstructs the three-dimensional view of the aircraft camera employing the map and navigational data. Next, the algorithms are implemented using actual video frames. Finally, mismatches in the runway detection due to sensor noise and to an assumption made on the aircraft roll orientation angle are corrected using image-processing techniques, such as the Hough transform for linear features.

| | | | |
|---|---|---|---|
| **14. SUBJECT TERMS** runway, map, aircraft, video, detection, rotation matrix, Hough transform. | | | **15. NUMBER OF PAGES** 87 |
| | | | **16. PRICE CODE** |
| **17. SECURITY CLASSIFICATION OF REPORT** Unclassified | **18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified | **19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified | **20. LIMITATION OF ABSTRACT** UU |

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

# RUNWAY DETECTION FROM MAP, VIDEO AND AIRCRAFT NAVIGATIONAL DATA

Jose R. Espinosa Gloria
Lieutenant Commander, Mexican Navy
B.S., Mexican Naval Academy, 2002

Submitted in partial fulfillment of the
requirements for the degree of

**MASTER OF SCIENCE IN ELECTRICAL ENGINEERING**

from the

**NAVAL POSTGRADUATE SCHOOL**
**March 2016**

Approved by: Roberto Cristi
Thesis Advisor

Oleg Yakimenko
Co-Advisor

R. Clark Robertson
Chair, Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

# ABSTRACT

As part of the reinforcement of operations performed by the Mexican Navy, unmanned aerial vehicles (UAV) have been equipped with daylight and infrared cameras. Processing the video information obtained from these devices opens the door to a number of strategic opportunities. By recognizing patterns in visual sources, we address one problem in particular: how to achieve corresponding runways using a map and an individual frame of video.

An approach to runway detection using two tools is presented in this thesis. The first tool is a geographical information program, which is used to set the runway map we want to detect. The second tool is a video frame of the same runway, recorded in a camera mounted on a UAV. The needed equations and related algorithms are developed and tested on a simulation that reconstructs the three-dimensional view of the aircraft camera employing the map and navigational data. Next, the algorithms are implemented using actual video frames. Finally, mismatches in the runway detection due to sensor noise and to an assumption made on the aircraft roll orientation angle are corrected using image-processing techniques, such as the Hough transform for linear features.

# TABLE OF CONTENTS

## LIST OF FIGURES

# LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

## LIST OF ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| 2D | Two-dimensional |
| 3D | Three-dimensional |
| HFOV | Horizontal field of view |
| ISR | Intelligence, surveillance and reconnaissance |
| LTP | Local tangent plane |
| UAV | Unmanned aerial vehicle |

THIS PAGE INTENTIONALLY LEFT BLANK

# ACKNOWLEDGMENTS

First and foremost, I would like to thank my wife, Alma, for her support, love and patience during my time at Naval Postgraduate School, especially during the last nine months when I literally worked all day. In addition, I would like to thank my daughters, María Fernanda and María José, for motivating me to be my best. You bring meaning to my life. I love you so much.

My deepest gratitude goes to my advisor, Roberto Cristi, for providing his time despite his tight schedule and for explaining to me both easy and difficult concepts with a lot of patience. I could not have completed this thesis without his help.

I also would like to thank my co-advisor, Oleg Yakimenko, for the explanations about the TASE200 camera and the MATLAB algorithm that you provided for collecting the real data of the video frames. I wish I had more time here to learn more from you.

A special thanks also goes to Commander Mariano Lizarraga of the Mexican Navy for believing in me and giving me this great educational opportunity, which surely will change my life. His professionalism has been an example of excellence for me during my time here at NPS.

Finally, I want to thank to the professors who taught classes and helped me to succeed at NPS, especially James Calusdian and Xiaoping Yun.

THIS PAGE INTENTIONALLY LEFT BLANK

# I. INTRODUCTION

Unmanned aerial vehicles (UAVs) have become vastly more popular during the last few years. In the military, it is no surprise why. These vehicles and their payload capabilities allow the military to accomplish missions more effectively. UAVs with cameras attached gather zone-specific tactical information that operators can use to make decisions. They support intelligence, surveillance and reconnaissance (ISR) operations, providing invaluable data for ground forces. Furthermore, during natural disasters, cameras mounted on UAVs provide an edge to first-responder teams by delivering broad aerial views so teams can navigate directly to survivors instead of wasting valuable time.

Today, in most applications, the videos recorded by these cameras are usually not processed with image processing techniques. For example, an image of a runway captured in this way allows for a variety of questions to be addressed such as, how large is the runway? What is the runway number? How many airplanes are there? Are people or buildings surrounding the runway?

An approach for processing an image of an airport to obtain information to detect a runway by its corresponding map is presented in this thesis.

## A. PREVIOUS APPROACHES

There are many studies in the area of runway detection and recognition. Some of the most relevant of these studies are described in this section.

The approach developed in [1] proposes the use of a methodology based on the textural characteristics of runways since these properties "are the most descriptive element of an airport." This method uses the AdaBoost algorithm [2], which selects runway-useful features, such as image intensity and gradient or Zernike Moments, and extracts them on a set of 57 satellite images to achieve the runway detection. The results of using this methodology on satellite images to detect runways are illustrated in Figure 1.

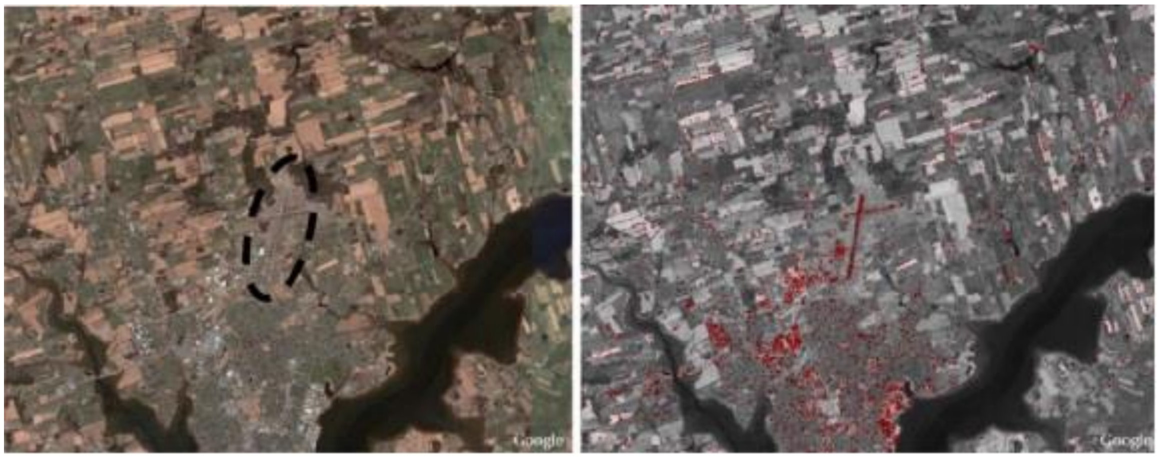

Figure 1. Original Image (Left) with the Runway Enclosed in Dashed Lines. Detected Runway Blocks are Shown in Red (Right)

Source: [1] Ö. Aytekin, U. Zöngür, and U. Halici, “Texture-based airport runway detection,” *IEEE Geosci.Remote Sensing Lett*., vol. 10, no. 3, pp. 471–475, May 2013.

Another approach from [3] points out how the location of a runway is estimated. It uses video frames from a forward-looking camera mounted on an UAV and compares the frames against a set of previously recorded images with different altitudes and positions along the glide path of a runway of a known location. This approximation uses the geometrical properties of the video frame—namely runway offset, runway angle and runway distance—rather than feature recognition. The video frames used as a reference for the comparison are shown in Figure 2.

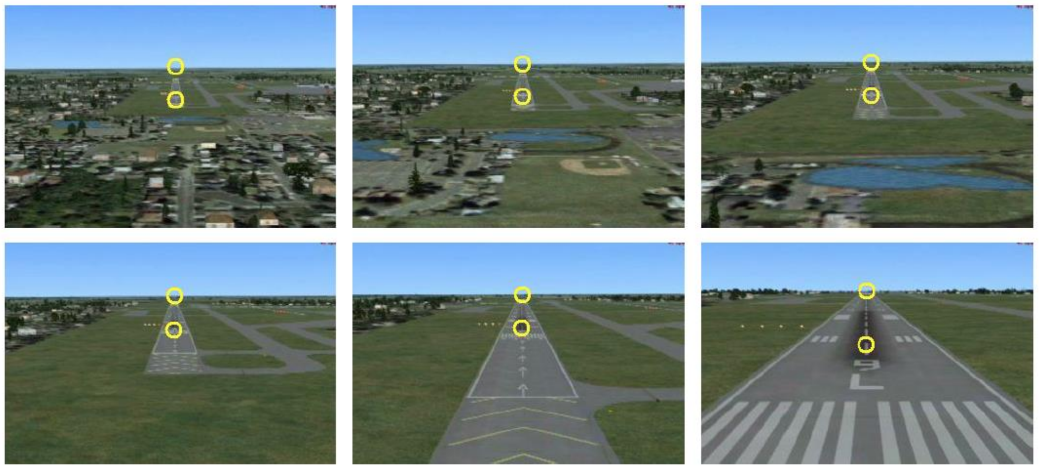

Figure 2. Reference Frames Used for Comparison

Source: [3] A. Miller, M. Shah, and D. Harper, “Landing a UAV on a runway using image registration,” in *Proceedings International Conference on Robotics and Automation*, Pasadena, CA., 2008, pp. 182–187.

A third approach that utilizes video data [4] presents a system based on the runway area and some visual representative points. This approach also introduces the use of the Hough transform and the least-square fit technique to detect the runway edges. The results of this methodology are shown in Figure 3.

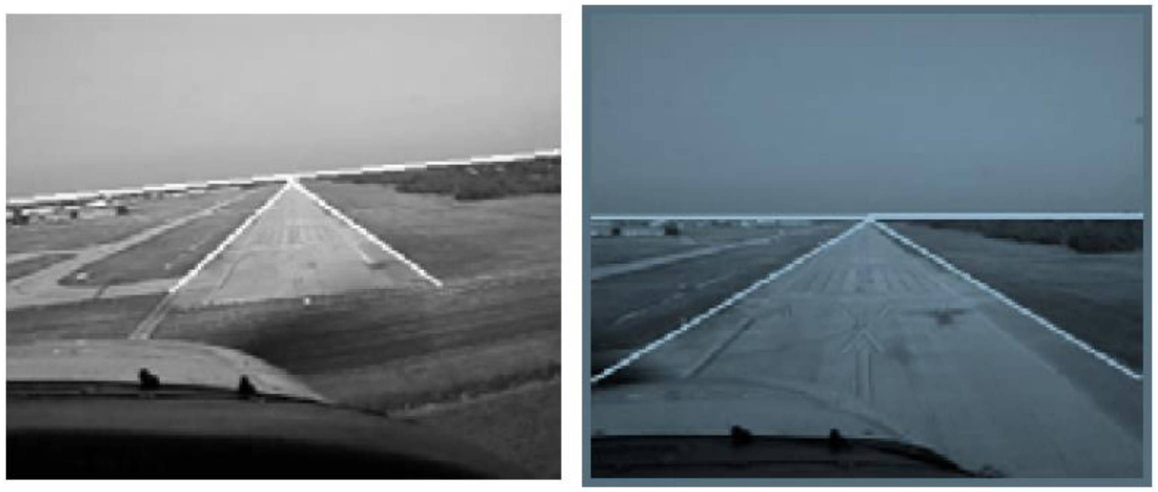

Figure 3. Detection Results for Runway and Horizon, Highlighted with a White Line

Source: [4] J. Shang, and Z. Shi, “Vision-based runway recognition for UAV autonomous landing,” *Int. Journal of Computer Sci. and Network Security*, vol. 7, no.3, pp. 112–117, Mar. 2007.

## B. THESIS OBJECTIVE

The intent of this thesis is to present a method that allows runway detection using two tools. The first tool is a map and geographical information program that is available for use on any computer. This software is used to set the runway to be detected. The second tool is a video frame of the same runway recorded in a camera mounted on a UAV. To start, we identify and set four points on the map, forming the two edges of the runway, and consider other parameters such as the position and attitude of the UAV (roll, pitch and yaw). Then we apply geometrical transformations to project these two edges on the map into the video frame in such a way that allows us to match the map runway with the runway in the frame of video.

As previously mentioned, there are many approaches to solve the runway detection problem. The method presented in this research differs from others in the sense that a map is used as a reference in addition to the video frame. Google Earth was used to retrieve the maps.

This methodology is meant to capitalize on new technologies, including UAVs, which the Mexican Navy has integrated into its operations.

## C. THESIS STRUCTURE

The remainder of this thesis is organized as follows:

In Chapter II, the problem of runway detection and matching is defined, and the mathematical process behind the matching of a line that lies on a plane with the same line in the camera view is described. Concepts such as the pinhole camera model, reference frames (map, aircraft and camera), rotation matrices, roll, pitch, yaw, tilt and pan are used to derive the equations needed to implement the runway detection and matching.

In Chapter III, how the equations from previous chapter are implemented in an algorithm that conducts a simulation method is described. This virtual scenario presents the camera view of a runway and compares it against the runway map.

In Chapter IV, how runway matching and detection are achieved using an image from a TASE200 camera rather than the simulated camera view is demonstrated. Small corrections are applied due to sensor noise, and the Hough transform methodology is utilized to correct this mismatch by detecting the external edges of the runway that are closer to the lines that were first detected.

Finally, the results, conclusions and recommendations for future work are summarized in Chapter V.

THIS PAGE INTENTIONALLY LEFT BLANK

# II. ANALYTICAL FRAMEWORK

## A. OVERVIEW

Important concepts used as a basis to develop the algorithms in this thesis are introduced in this chapter. First, we discuss the pinhole camera model, which is used to describe the geometrical relationship between a point in three-dimensional (3D) space and the two-dimensional (2D) coordinate system. We also address the concept of a rotation matrix since we must perform several transformations between coordinate systems. Once the projection from 3D to 2D has been introduced, particular attention is given to the concept of mapping relevant features such as lines since we are interested in detecting runway boundaries. The methodology derived in this thesis allows points between a map and the projected 2D coordinate system to be matched.

## B. CAMERA PINHOLE MODEL

A camera model mathematically maps a 3D scene in space on to a 2D image. In this research, we use the pinhole model shown in Figure 4 to describe the geometrical relationship between the *x* and *y* coordinates of a point in the 2D plane and its real coordinates in the 3D world.

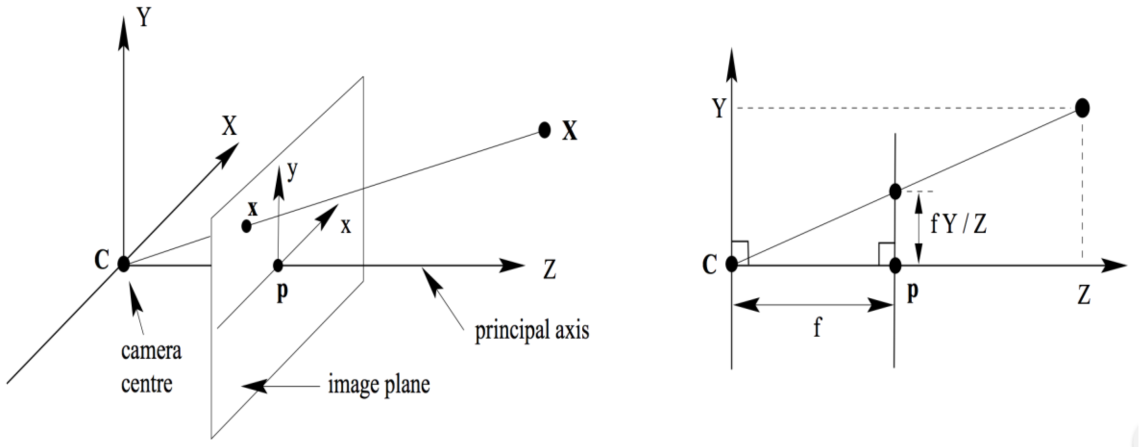

Figure 4. Pinhole Camera Model to Project 3D Coordinates onto 2D Plane (Left) and Triangular Similarities (Right)

Source [5]: R. Hartley and A. Zisserman. (2004). *Multiple View Geometry in Computer Vision*. [Online]. Available: https://itun.es/us/_pvzW.l

The relation between the *x, y* 2D coordinates in the camera plane and the *X, Y, Z* coordinates of the 3D object is computed from simple triangular similarities

$$\begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{1}$$

where *f* is the focal length of the camera, *X, Y, Z* are the three coordinates in 3D space (real world), and $x$, $y$ are the two coordinate in 2D plane (image).

In general, the two reference frames (world and camera) are different. Consequently, it is necessary to transform the world frame coordinates into the camera frame coordinates. The relation is obtained by a sequence of translations and rotations, which are defined by the specific location and orientation of the camera in world coordinates.

In some cases, such as the descriptions of certain reference frames like the world, camera, aircrafts, ships or vehicles, the frames are defined by standard conventions, which also define the terminology used to identify rotations and translations.

In the following section, we define these conventions, which are central to our research.

## C. ROTATION MATRICES

One important aspect to consider when projecting an object from the 3D real world into the 2D plane is that we must to execute several transformations between coordinate systems. These conversions are performed by rotation matrices [6], which are a way to describe the orientation of coordinate system *B* with respect to another coordinate system *A*. Both coordinate systems are shown in Figure 5.

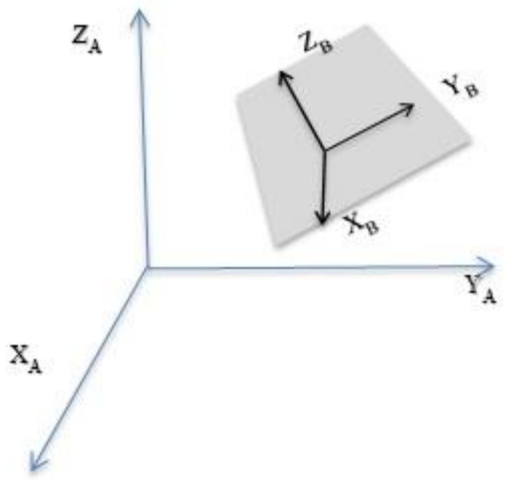

Figure 5. Coordinate System *A* and Object Coordinate System *B*.

There are three basic rotation matrices, depending about which axis (*x*, *y* or *z*) performs the rotation. These matrices are

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) \\ 0 & -\sin(\theta_x) & \cos(\theta_x) \end{bmatrix} \tag{2}$$

$$R_y = \begin{bmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) \\ 0 & 1 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix} \tag{3}$$

$$R_z = \begin{bmatrix} \cos(\theta_z) & \sin(\theta_z) & 0 \\ -\sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

## D. FRAMES

When doing the projection of a point (or a set of points) from the map to the image, it is important to consider the convention used for each of the frames. There are five frames: map, world, aircraft, camera, and image.

### 1. Map Frame

Once the location of the runway is determined, a section of the map is cropped to get an *N* by *M* image that represents the map. For this image representation, a convention was established, namely rows (down), columns (right) and elevation (up). The Salinas Municipal Airport Map image representation is shown in Figure 6.

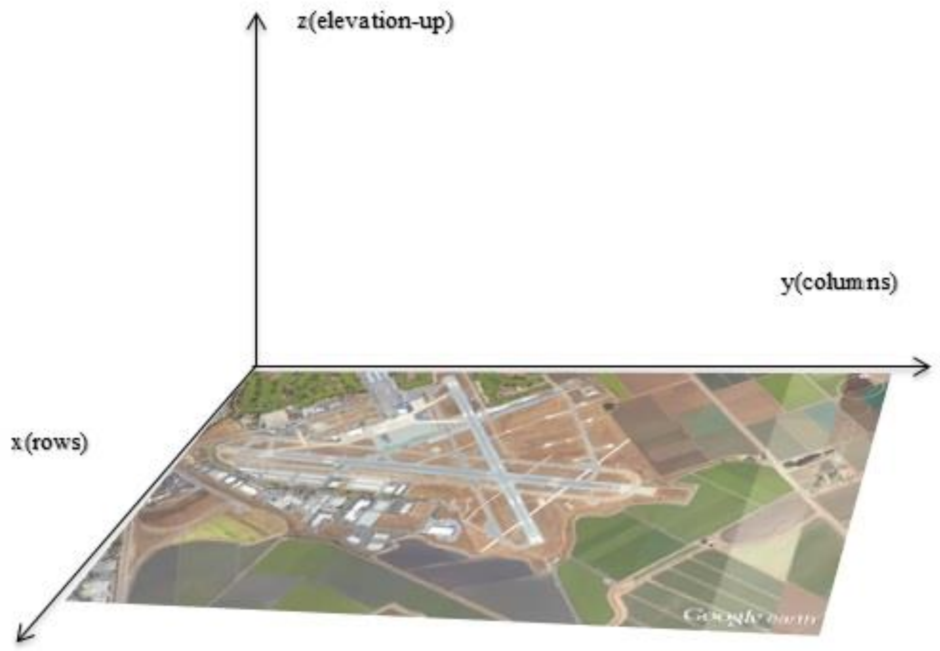

Figure 6. N Rows by M Columns Image of Salinas Municipal Airport Map

Adapted from [7]: "Salinas Municipal Airport." 2016. [Online]. Available: https://www.google.com/maps/@36.6612306,-121.6066075,2805m/data=!3m1!1e3. Accessed Jan. 22, 2016.

### 2. World Frame

Consider an object situated in the real world. Its location is given by the coordinates *x*, *y* and *z*, which are shown in Figure 7. This reference system is defined as the *world frame*, also known as the local tangent plane (LTP). The convention used for these three coordinate points is as follows: north for *x*, east for *y* and down for altitude (*z*). The origin can be anywhere of significance on the specific problem.

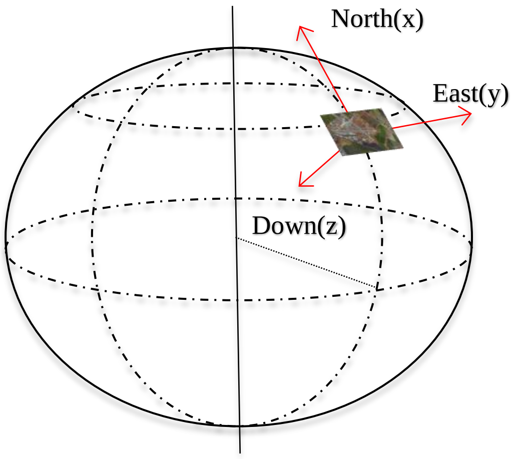

Figure 7. Arbitrary Object with Coordinates *x*, *y* and *z* in World Frame

Map adapted from [7]: “Salinas Municipal Airport.” 2016. [Online]. Available: https://www.google.com/maps/@36.6612306,-121.6066075,2805m/data=!3m1!1e3. Accessed Jan. 22, 2016.

### 3. Aircraft Frame

The orientation of the coordinate system of an aircraft with respect to another is described by the angles formed about its *x*, *y* and *z* axes. These three angles are known as roll, pitch and yaw, respectively. The sign convention for these angles was established as positive when they perform a clockwise rotation, and negative otherwise, considering that the axes’ direction with respect to the aircraft is roll forward, pitch right and yaw down, as depicted in Figure 8.

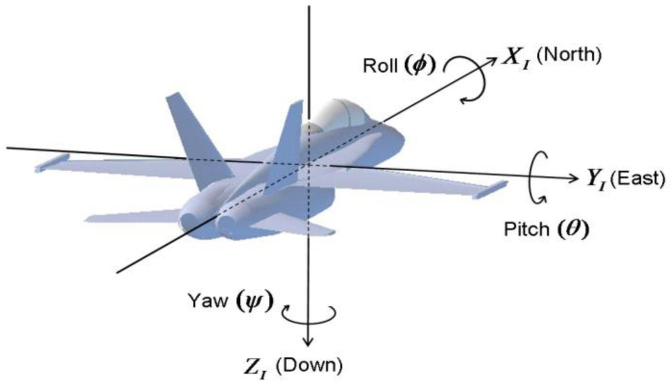

Figure 8. Roll, Pitch and Yaw Positive Rotation Angles in the Aircraft Frame

Source [8]: *Understanding Euler Angles*. [Online]. Available: http://www.chrobotics.com/library/understanding-euler-angles. Accessed Jan. 22, 2016.

### 4. Camera Frame

The orientation of the coordinate system of a camera with respect to another is described by the angles formed about its *x*, *y* and *z* axes. These three angles are roll, tilt and pan, respectively. They represent the Euler angles, and their rotations have to be executed in a specific order. This is called the *camera frame*. The sign convention for these angles was established as positive for a clockwise rotation and negative otherwise, considering that the axes' direction with respect to the camera is roll forward, tilt right and pan down, as illustrated in Figure 9.

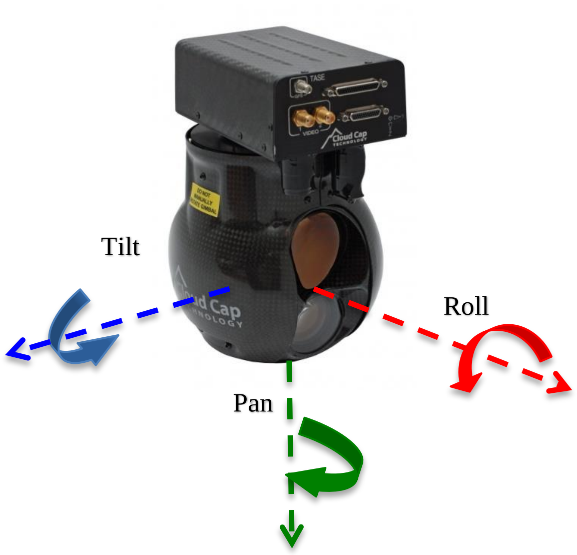

Figure 9. Roll, Tilt and Pan Positive Rotation Angles in Camera Frame

Adapted from [9]: *Cloud Cap Technology TASE 200 camera*. [Online]. Available: http://www.cloudcaptech.com/products/detail/tase-200. Accessed Jan. 23, 2016.

## E. PROJECTION OF A LINE

Some features are invariant when projected from a 3D world frame to a 2D image frame, based on the pinhole model. Particular interest in this research is the fact that lines in the 3D world are projected into lines on an image plane. The goal of this section is to establish these when the lines are borders of runways.

Let us consider a line on the plane with equation $y = ax + b$ and with the $z$ axis pointing toward the sheet, as shown in Figure 10. This is the case of a linear feature on a map.

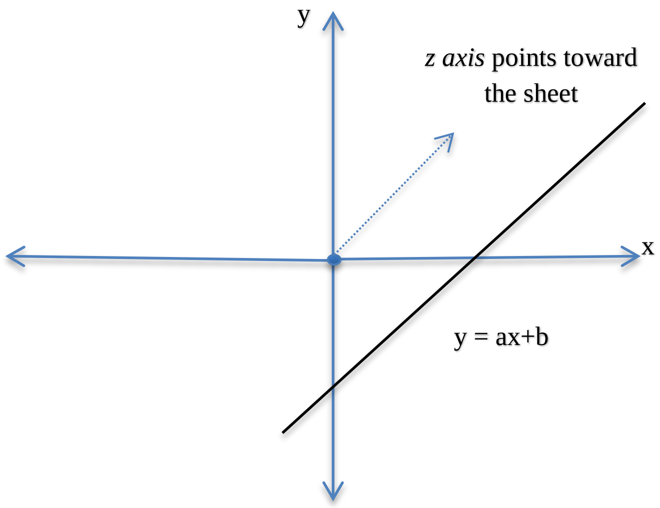

Figure 10. Line on a Plane and Its Basic Equation

Now, take a camera mounted on an airplane with an altitude *h* and tilt it by an angle $\delta$ around the *x*-axis. A side view of the *y*-*z* plane and a point *P on* the ground are depicted in Figure 11. The three axes of the camera frame are defined as $x_c$, $y_c$ and $z_c$, with the $x_c$ axis pointing out of the sheet. In particular, the $z_c$ axis is orthogonal to the camera plane. On the other hand, the three axes of the airplane frame are defined as $x_p$, $y_p$ and $z_p$, with the positive *x* axis pointing also out of the paper.

The coordinates of the point *P* on the ground are

$$P_g = \begin{bmatrix} x_g & y_g \end{bmatrix}^T \tag{5}$$

and in the airplane frame, assuming the plane is flying at zero pitch in the north direction, they are

$$P_p = \begin{bmatrix} x_p & y_p & h \end{bmatrix}^T \tag{6}$$

where *h* is the height of the airplane, namely the positive $z_p$ *axis.*

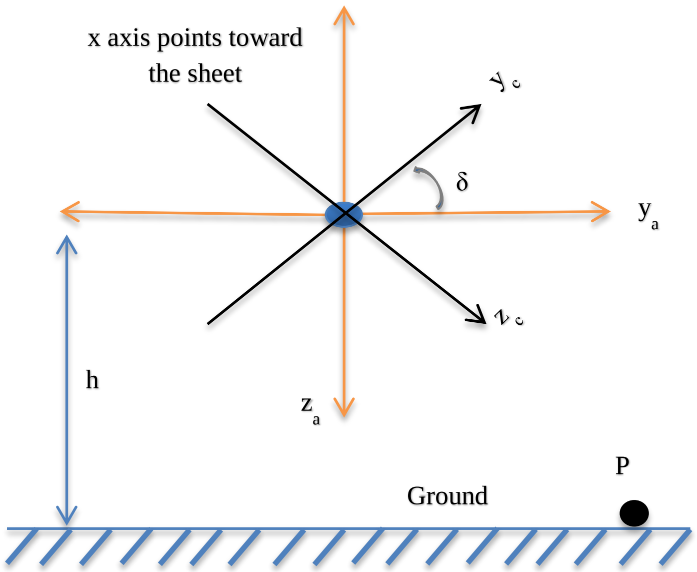

Figure 11. Side View of the *y-z Plane* (the Airplane and Camera Frame in Orange and Black, Respectively)

We have set the coordinates of a point on the ground from the perspective of the airplane frame. The final objective is to know these coordinates in the camera frame. Recalling the rotation matrices mentioned previously, we need to perform a rotation about the *x-axis*; therefore, the rotation matrix we need is

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\delta) & \sin(\delta) \\ 0 & -\sin(\delta) & \cos(\delta) \end{bmatrix} \tag{7}$$

which represents the projection of the camera frame on the airplane frame, where $\delta$ is the angle between both frames.

By multiplying the position of the point *P* in the airplane frame by the rotation matrix (7), we get $P_c$, which is the position of the point with respect to the camera frame:

$$P_c = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\delta) & \sin(\delta) \\ 0 & -\sin(\delta) & \cos(\delta) \end{bmatrix} \begin{bmatrix} x \\ y \\ h \end{bmatrix} \tag{8}$$

and

$$P_c = \begin{bmatrix} x \\ \cos(\delta)\,\mathrm{y} + \sin(\delta) h \\ -\sin(\delta)\,\mathrm{y} + \cos(\delta)\,\mathrm{h} \end{bmatrix}. \tag{9}$$

Recall that the projection of a vector into another is related to the cosine or sine of the angle $\delta$ formed between them.

Let us consider two planes, *y*-z and $y_c$-$z_c$, as depicted in Figure 12.

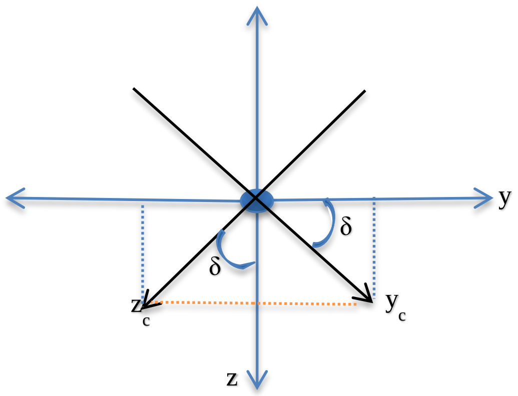

Figure 12. Planes *y*-*z* and $y_{c}$- $z_c$ with Angle $\delta$ Formed between Them

The relationship between vectors $y_c$, $z_c$ and vectors *y*, *z* is given by

$$y_c = \cos(\delta) y + \sin(\delta) z \tag{10}$$

$$z_c = -\sin(\delta) y + \cos(\delta) z \tag{11}$$

where from (10) and (11), we observe that

$$\begin{bmatrix} y_c & z_c \end{bmatrix} = \begin{bmatrix} y & z \end{bmatrix} \begin{bmatrix} \cos(\delta) & -\sin(\delta) \\ \sin(\delta) & \cos(\delta) \end{bmatrix} \tag{12}$$

and solving for $\begin{bmatrix} y & z \end{bmatrix}$ yields

$$\begin{bmatrix} y & z \end{bmatrix} = \begin{bmatrix} y_c & z_c \end{bmatrix} \begin{bmatrix} \cos(\delta) & \sin(\delta) \\ -\sin(\delta) & \cos(\delta) \end{bmatrix} . \tag{13}$$

If we have two random points *p₁* and *p₂* in the *y*-*z* plane, such that

$$P = \begin{bmatrix} y & z \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \tag{14}$$

their location, considering (13), with respect to the $y_c$-$z_c$ plane is given by

$$P_c = \begin{bmatrix} y_c & z_c \end{bmatrix} \begin{bmatrix} \cos(\delta) & \sin(\delta) \\ -\sin(\delta) & \cos(\delta) \end{bmatrix} \begin{bmatrix} p1 \\ p2 \end{bmatrix} . \tag{15}$$

Now, consider a line on the ground with equation $y = ax + b$. All the points *P(x)* on the line have the form

$$P(\mathrm{x}) = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x \\ ax+b \end{bmatrix} . \tag{16}$$

Substituting *y* by $ax+b$ in (9) gives

$$P_c(x) = \begin{bmatrix} x \\ \cos(\delta)(ax+b)+\sin(\delta)h \\ -\sin(\delta)(ax+b)+\cos(\delta)h \end{bmatrix} \tag{17}$$

with

$$\begin{aligned} x &= x \\ y &= \cos(\delta)(ax+b)+\sin(\delta)h \\ z &= -\sin(\delta)(ax+b)+\cos(\delta)h \end{aligned} \tag{18}$$

which is the position *x*, *y* and *z* (height) of all the points *P(x)* on the camera frame.

Substituting *x*, *y* and *z* from (17) into the pinhole camera model (1) yields the position of any point in the image frame as

$$\begin{bmatrix} M_1 \\ M_2 \end{bmatrix} = \frac{f}{-\sin(\delta)(ax+b)+\cos(\delta)h}\begin{bmatrix} x \\ \cos(\delta)(ax+b)+\sin(\delta)h \end{bmatrix}. \tag{19}$$

Simplifying this equation results in

$$\begin{bmatrix} M_1 \\ M_2 \end{bmatrix} = \begin{bmatrix} \dfrac{fx}{-\sin(\delta)(ax+b)+\cos(\delta)h} \\ \dfrac{f\cos(\delta)(ax+b)+f\sin(\delta)h}{-\sin(\delta)(ax+b)+\cos(\delta)h} \end{bmatrix}. \tag{20}$$

Discretizing $M_1$ and $M_2$ by the pixel size ($\Delta$) of the image yields

$$M_1 = m_1\Delta \tag{21}$$

$$M_2 = m_2\Delta\,. \tag{22}$$

Substituting (21) and (22) into (20), we get

$$\begin{bmatrix} m_1 \\ m_2 \end{bmatrix} \Delta = \begin{bmatrix} \dfrac{fx}{-\sin(\delta)(ax+b)+\cos(\delta)h} \\ \dfrac{f\cos(\delta)(ax+b)+f\sin(\delta)h}{-\sin(\delta)(ax+b)+\cos(\delta)h} \end{bmatrix}. \tag{23}$$

As a result,

$$\begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = \begin{bmatrix} \dfrac{fx}{\Delta(-\sin(\delta)(ax+b)+\cos(\delta)h)} \\ \dfrac{f\cos(\delta)(ax+b)+f\sin(\delta)h}{\Delta(-\sin(\delta)(ax+b)+\cos(\delta)h)} \end{bmatrix} \tag{24}$$

and, finally, we get

$$\begin{bmatrix} m_1 \\ m_2 \end{bmatrix} = \begin{bmatrix} \dfrac{\overline{f}x}{-\sin(\delta)(ax+b)+\cos(\delta)h} \\ \dfrac{\overline{f}\cos(\delta)(ax+b)+\overline{f}\sin(\delta)h}{-\sin(\delta)(ax+b)+\cos(\delta)h} \end{bmatrix} \tag{25}$$

with

$$\overline{f} = \frac{f}{\Delta}. \tag{26}$$

Now it is necessary to find a set of equations that relates the line $y = ax + b$ in the map frame with the line $y = \alpha x + \beta$ in the image frame.

This relationship can be derived from (25), where we equate $m_1 = g(x)$ and $m_2 = \alpha g(x) + \beta$. The result is

$$g(x) = \frac{\overline{f}x}{-\sin(\delta)(ax+b) + \cos(\delta)h} \tag{27}$$

and

$$\alpha g(x) + \beta = \frac{\overline{f}\cos(\delta)(ax+b) + \overline{f}\sin(\delta)h}{-\sin(\delta)(ax+b) + \cos(\delta)h}\ . \tag{28}$$

Substituting (27) into (28), we get

$$\frac{\overline{f}\cos(\delta)(ax+b) + \overline{f}\sin(\delta)h}{-\sin(\delta)(ax+b) + \cos(\delta)h} = \alpha\frac{\overline{f}x}{-\sin(\delta)(ax+b) + \cos(\delta)h} + \beta\ . \tag{29}$$

Now (29) can be written in the form

$$\overline{f}\cos(\delta)(ax+b) + \overline{f}\sin(\delta)h = \alpha\overline{f}x + \beta(-\sin(\delta)ax - \sin(\delta)b + \cos(\delta)h)\ . \tag{30}$$

Grouping terms in (30) yields

$$\overline{f}\cos(\delta)ax + (\overline{f}\cos(\delta)b + \overline{f}\sin(\delta)h) = x(\alpha\overline{f} - \beta\sin(\delta)a) - \beta\sin(\delta)b + \beta\cos(\delta)h)\ . \tag{31}$$

Equating terms from both sides of (31), we get

$$\overline{f}\cos(\delta)a = \alpha\overline{f} - \beta\sin(\delta)a \tag{32}$$

$$\bar{f}\cos(\delta)b + \bar{f}\sin(\delta)h = -\beta\sin(\delta)b + \beta\cos(\delta)h, \tag{33}$$

and solving (33) for $\beta$ yields the solution

$$\beta = \frac{\bar{f}(\sin(\delta)h + \cos(\delta)b)}{\cos(\delta)h - \sin(\delta)b}. \tag{34}$$

Solving (32) for $\alpha$ and substituting the value of $\beta$, we get

$$\alpha = \frac{ha}{\cos(\delta)h - \sin(\delta)b}. \tag{35}$$

From (34) and (35), we solve for $a$ and $b$ with the result

$$a = \frac{\bar{f}}{\cos(\delta)\bar{f} + \sin(\delta)\beta}\alpha \tag{36}$$

and

$$b = \frac{\cos(\delta)\beta - \sin(\delta)\bar{f}}{\cos(\delta)\bar{f} + \sin(\delta)\beta}h. \tag{37}$$

After all the derivations, (34), (35), (36) and (37) describe the relationship that exists between a line with equation $y = ax + b$ in the map frame and the line with equation $y = \alpha x + \beta$ in the image frame.

## F. HOUGH TRANSFORM

In order to be able to detect the boundary lines of the runway and compensate for the mismatch due to sensor noise, the Hough transform is applied. The Hough transform algorithm [10] is a method used to detect linear features using the parametric representation of a line as $r = x_o \cos(\theta) + y_o \sin(\theta)$. This method and its practical applications are described in the following paragraphs.

### 1. Mathematical Foundation

The equation that describes a line in the 2D *x-y* plane is $y = mx + b$ where *m* is the slope and *b* the intercept. For every parameter *m* and *b*, there is a line lying on this plane. From the two parameters cited above, set the *m-b* plane. Each line on this plane corresponds to a point *p* in the *x-y* plane. If we choose a point $\begin{bmatrix} x_o & y_o \end{bmatrix}$ in the *x-y* plane, then an infinite number of lines go through this point with *n* equations $y_o = m_n x_o + b_n$ defining them. Each one of these *n* lines is represented in the *m-b* plane by *n* points. In other words, a line in the *x-y* plane is a point in the *m-b* plane. Likewise, infinite lines (or one point) in the *x-y* plane is a line in the *m-b* plane.

All the lines in the *m-b* plane that intersect each other represent common lines in the *x-y* plane. Points and lines that correspond with each other have the same color. This relation is shown in Figure 13. The region with more lines intersected (one point) in the *m-b* plane represents a line that is common to a set of points, meaning they have same slope and intercept. Essentially, the Hough transform finds the correspondence between lines in the *x-y* plane and points in the *m-b* plane.

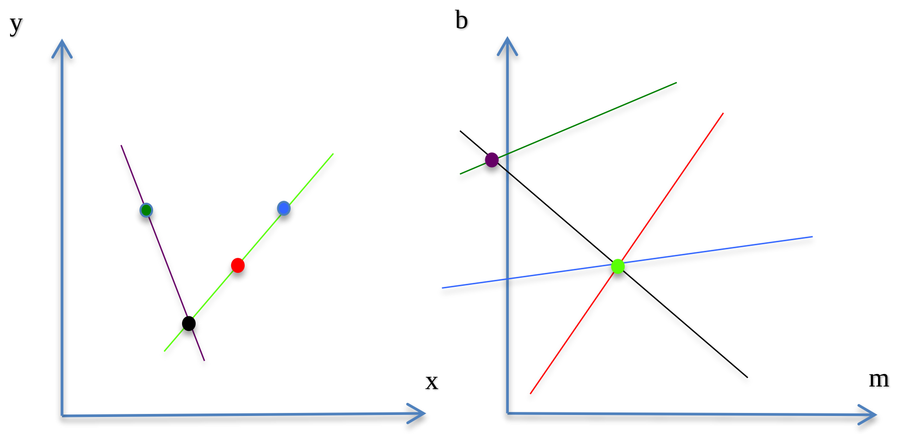

Figure 13. Correspondence between Points and Lines in Planes *x-y* and *m-b*

## 2. Practical Computation of Hough Transform

In practice for the Hough transform computation we represent a line using another way that involves two parameters, namely distance *r* and angle theta *θ*. This representation is illustrated in Figure 14.

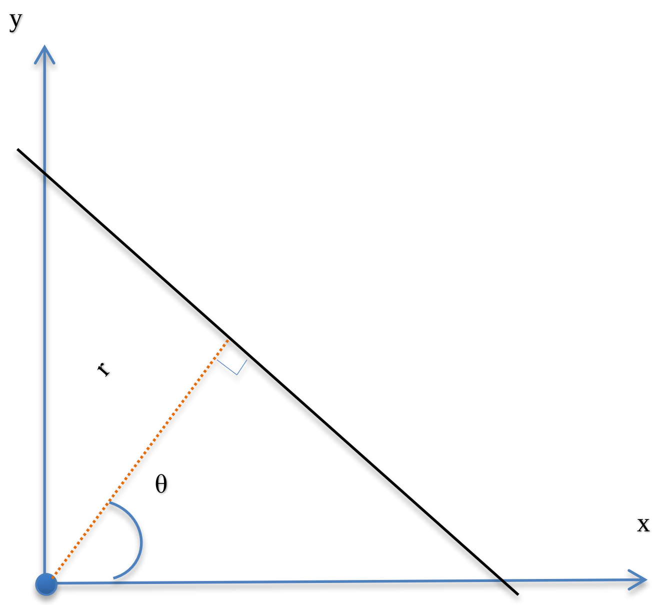

Figure 14. Line Described by Parameters Angle Theta *θ* and Distance *r* to the Origin

The line equation is written as

$$r = x_o \cos(\theta) + y_o \sin(\theta) \tag{38}$$

where *θ* is the angle between the positive *x* axis and the perpendicular line formed from the actual line to the origin, *r* is the distance from the origin to the actual line, measured over the perpendicular line.

As mentioned before, when using the regular line equation $y = mx + b$, we represent the *x-y* plane in the *m-b* plane. Similarly, using (38), we characterize the *x-y* plane in the *θ-r* plane. Thus, a line in the *x-y* plane is a point in the *θ-r* plane, and a point in the *x-y* plane is a sinusoid in the *θ-r* plane. All the sinusoids in the *θ-r* plane that intersect each other represent common lines in the *x-y* plane. This relation is shown in Figures 15 and 16. The region with more sinusoids intersected (one point) in the *θ-r* plane represents a line that is common to a set of points, meaning all the points lie on the same line.

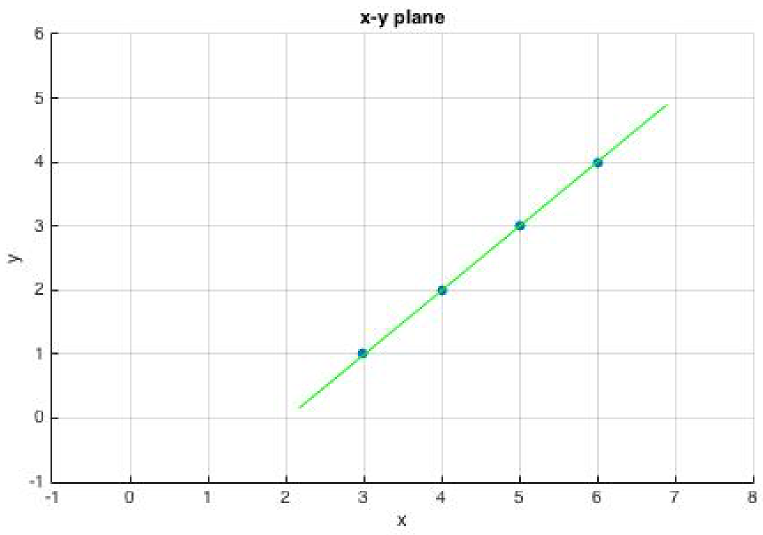

Figure 15. Four Points Form a Line in *x-y* Plane

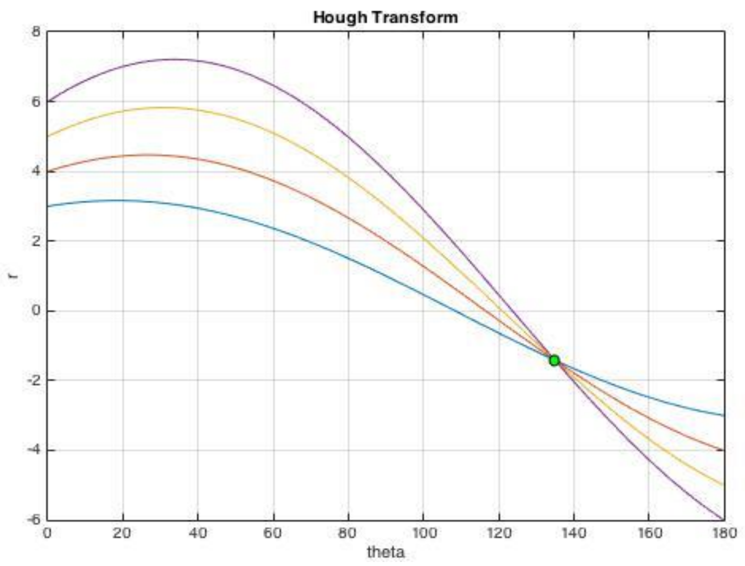

Figure 16. Four Sinusoids Intersect Each Other in One Point in *θ -r* Plane

THIS PAGE INTENTIONALLY LEFT BLANK

# III. MAP AND SIMULATED IMAGE MATCHING

## A. OVERVIEW

A simulation to validate and test the analytical framework introduced in Chapter II is presented in this chapter.

The simulator we developed manipulates real data in Google Maps taken from aerial photographs to construct an estimate of the camera view on an airplane under different altitude conditions as well as aircraft and camera orientations. The simulation was needed to validate the mathematical and geometrical techniques to map line features between different frames of reference in the pinhole camera model.

The virtual scenario corresponds to a simulated frame of video that shows a runway at an airport. The goal is to match the exterior lines of the runway displayed in the map with those shown in the virtual frame of video.

## B. TRANSFORMATIONS: MAP TO IMAGE AND IMAGE TO MAP

To get a virtual image from a camera's perspective that emulates a view of a runway on a map, it is necessary to perform a series of rotations, starting with the map frame up to the image frame. The goal is to reconstruct the 3D view of the aircraft camera from the map and simulated navigational data (i.e., the camera and aircraft orientation angles, position of the aircraft with respect to a reference, and so on).

In the following subsections, each of the rotations is discussed. It is important to keep in mind the convention used for each one of the frames since this property affects the components that are part of the rotation matrices.

In addition, the representation of any rotation from frame *a* to frame *b* is defined by $R_a^b$ and vice versa.

### 1. Rotation from Map Frame to World Frame

In Chapter II, we introduced the convention used for the map and world frames where the *y*-axis has the same direction for both of the frames. On the other hand, axes *x* (rows and north) and *z* (up and down) have opposite direction, as shown in Figure 17.

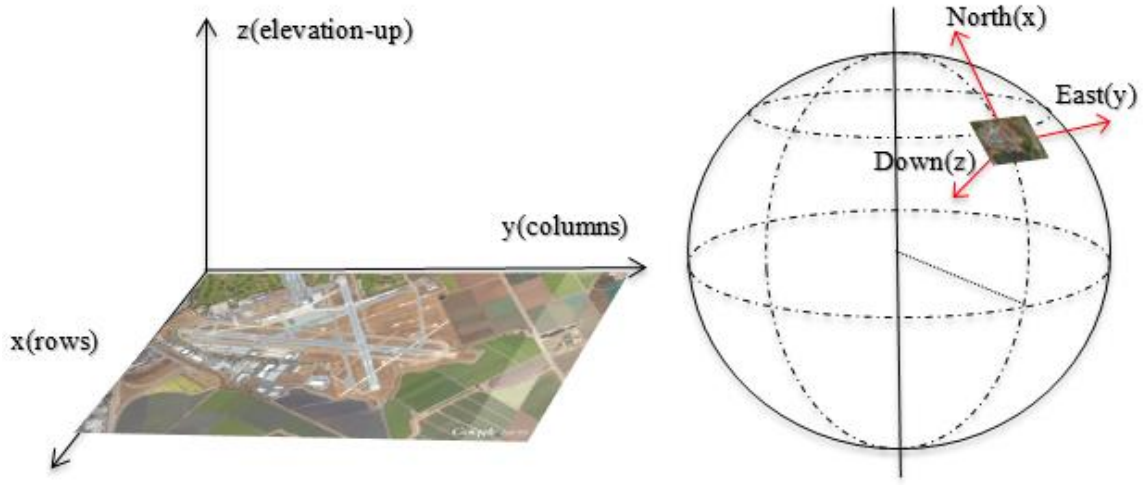

Figure 17. Map Frame (Left) and World Frame (Right)

Maps adapted from [7]: "Salinas Municipal Airport." 2016. [Online]. Available: https://www.google.com/maps/@36.6612306,-121.6066075,2805m/data=!3m1!1e3. Accessed Jan. 22, 2016.

To move from the map frame to the world frame, we multiply the map frame vectors by the following rotation matrix

$$R_m^w = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}. \tag{39}$$

By doing this, we can describe the position of a point in the world frame.

### 2. Rotation from World Frame to Aircraft Frame

Now that we have established the desired point in the world frame, let us move to the next step, which is to find the coordinates of the point in the aircraft frame.

To achieve this, we need to compare world and aircraft frames and to perform the needed rotation matrix according to the direction and convention of each axis. As illustrated in Figure 18, the directions of all the axes are the same for both frames.

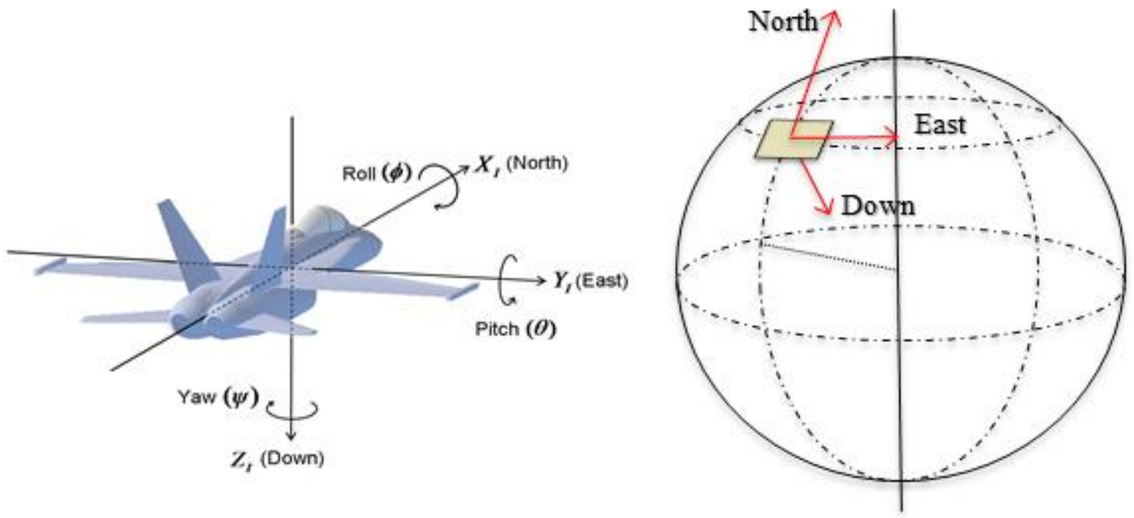

Figure 18. Aircraft Frame (Left) and World Frame (Right)

Aircraft from [8]: *Understanding Euler Angles*. [Online]. Available: http://www.chrobotics.com/library/understanding-euler-angles. Accessed Jan. 22, 2016.

To move from the world frame to the aircraft frame, we need to perform three rotations about the roll $\phi$, pitch $\theta$ and yaw $\psi$ orientation angles of the aircraft as follows:

$$R_w^a = R_{roll} R_{pitch} R_{yaw} . \tag{40}$$

The matrices used in (40) for rotations around each orientation angle are, respectively,

$$R_{roll} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix}, \tag{41}$$

$$R_{pitch} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \tag{42}$$

and

$$R_{yaw} = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{43}$$

### 3. Rotation from Aircraft Frame to Camera Frame

Since the camera is attached to the aircraft, it is expected that the three axes of the camera frame have the same direction of the respective aircraft frame axes. This is shown in Figure 19.

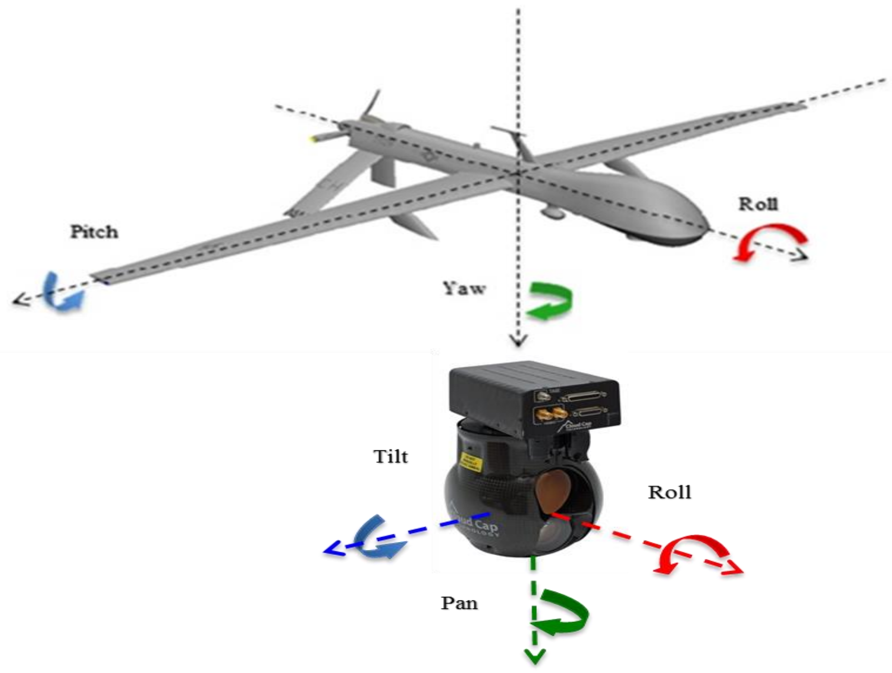

Figure 19. Aircraft Frame (Up) and Camera Frame (Down)

Camera adapted from [9]: *Cloud Cap Technology TASE 200 camera*. [Online]. Available: http://www.cloudcaptech.com/products/detail/tase-200. Accessed Jan. 23, 2016.

To go from the aircraft frame to the camera frame, we need to execute three rotations about the roll $\phi$, tilt $\theta$ and pan $\psi$ orientation angles of the camera.

The matrices used for rotations around each orientation angle are the same as in Equations (24), (25) and (26), with pan, tilt and roll replacing yaw, pitch and roll, respectively.

The rotation from the aircraft frame to the camera frame is given by

$$R_a^c = R_{roll} R_{tilt} R_{pan} . \tag{44}$$

### 4. Rotation from Camera Frame to Image Frame

At this point, we have gone from the map frame to the camera frame. This means that we know the position of a point in the map with respect to the camera frame. The last of the rotation matrices takes the point from the camera frame to the image frame. To understand the relation between these two frames, let us first compare the direction of their respective axes.

The convention used for the camera and image frames is such that the *y*-axis has the same direction for both of the frames. The *x*-axes (roll and rows) and *z*-axes (pan and up) have opposite directions, as shown in Figure 20.

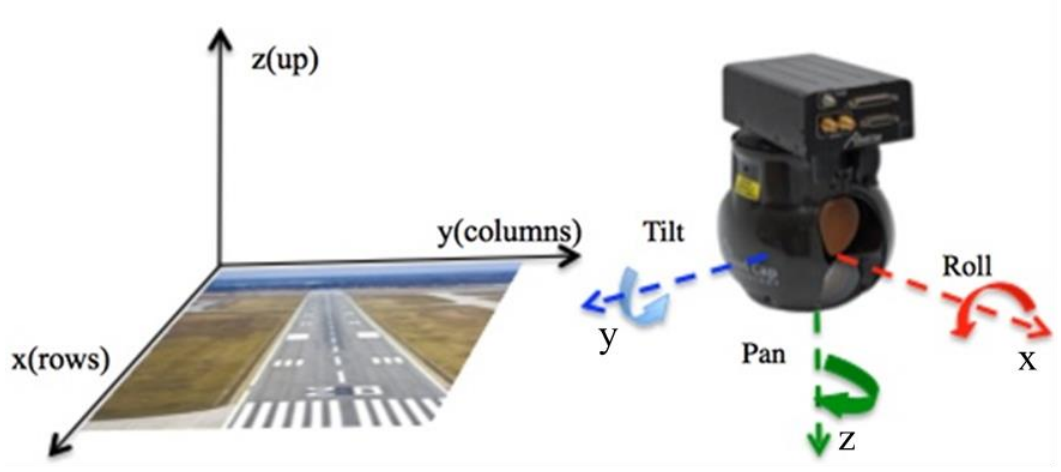

Figure 20. Image Frame (Left) and Camera Frame (Right)

Camera adapted from [9]: *Cloud Cap Technology TASE 200 camera*. [Online]. Available: http://www.cloudcaptech.com/products/detail/tase-200. Accessed Jan. 23, 2016.

The transformation from the camera frame to the image frame is given by

$$R_c^i = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \tag{45}$$

and the equation that yields the transformation from map to image is defined by

$$R_m^i = R_c^i R_a^c R_w^a R_m^w . \tag{46}$$

By transposing all the rotation matrices in equation (46), we can perform the transformation from image to map as

$$R_i^m = R_i^c R_c^a R_a^w R_w^m . \tag{47}$$

## C. COMPUTER SIMULATION

The mathematical operations related to the rotations explained previously were performed in two functions called map2image and image2map within a main MATLAB program (see Appendix). These functions map a number of points in the image plane to the corresponding points on the map plane. Since we are relating points on planar surfaces (image and map), the relation is one-to-one without ambiguity. In this operation, we clearly assume that the aircraft is flying at a sufficient high altitude so that the heights of the buildings are negligible. The points on the two planes (image and map) are represented as columns of two matrices, defined as

XYi: 2 by N points positions in image frame.

XYm: 2 by N points positions in map frame.

The main camera parameter is defined as the field of view (FOV). Dimensionless aircraft and camera orientations together with aircraft position and altitude are given as

thRPYaw: roll pitch yaw angles of aircraft

thRTPca: roll-tilt-pan angles of camera

posNEAw: position of the aircraft (north-east-altitude) in the world frame

Scale: size of pixel in the map, in meters/pixel.

The first part of the algorithm presents a virtual view of the aircraft camera using the map and the navigational data. The maps in Figures 21 and 22 of the Municipal Salinas and Regional Monterey Airport regions, respectively, were used to simulate the virtual view of the camera. The aircraft position is represented by the red dot.

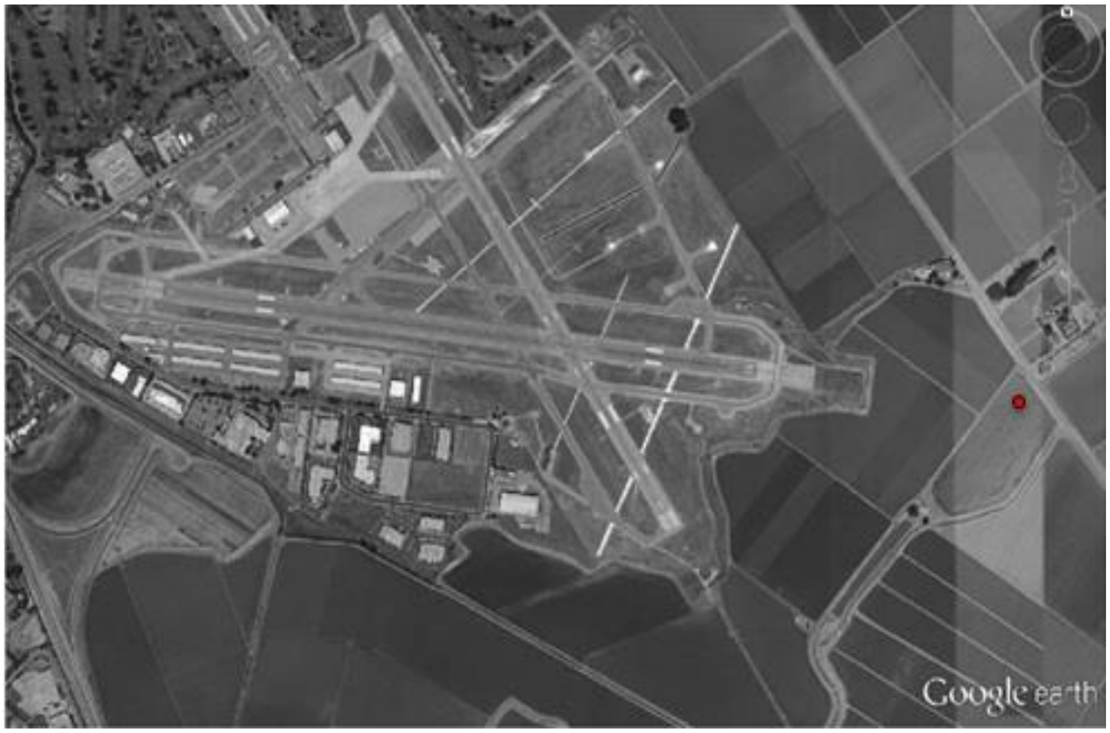

Figure 21. Salinas Municipal Airport and Aircraft Position (Red Dot)

Adapted from [7]: “Salinas Municipal Airport.” 2016. [Online]. Available: https://www.google.com/maps/@36.6612306,-121.6066075,2805m/data=!3m1!1e3. Accessed Jan. 22, 2016.

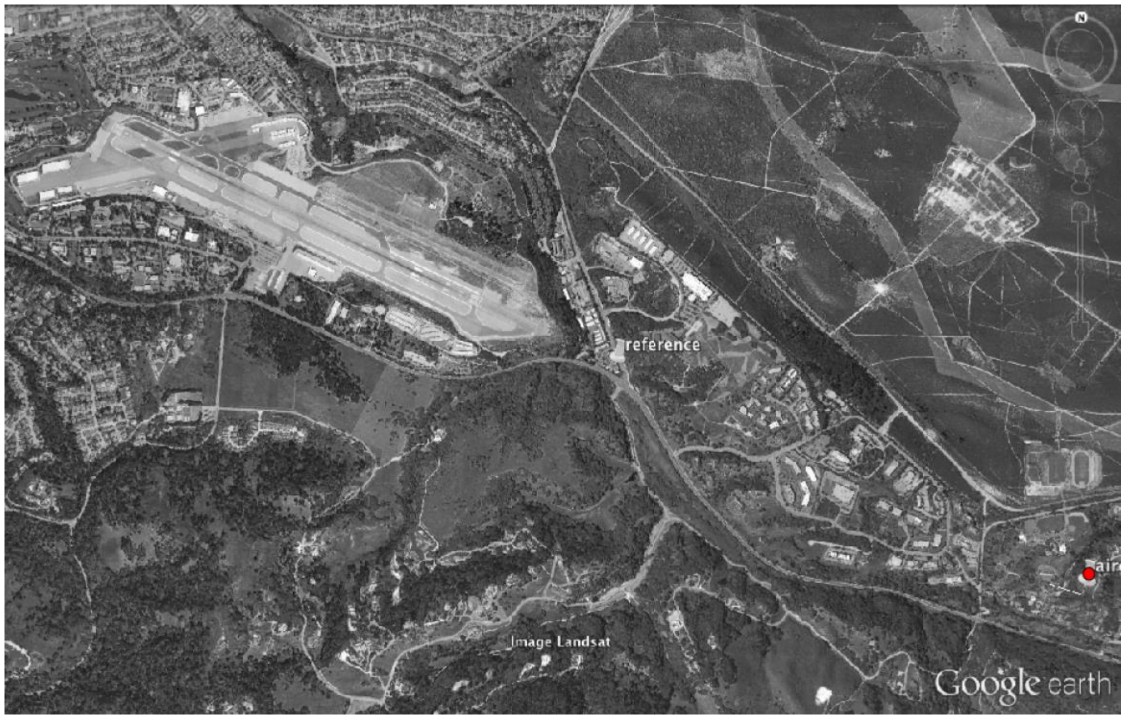

Figure 22. Monterey Regional Airport and Aircraft Positon (Red Dot)

Adapted from [11]: "Monterey Regional Airport." 2016. [Online]. Available: https://www.google.com/maps/@36.5857303,-121.8427775,2790m/data=!3m1!1e3 Accessed Jan. 22, 2016.

The algorithm presents an image by using the real location of the aircraft, its simulated navigational data and the camera orientation angles. The image portrays what the camera would show if the aircraft was flying in the location indicated in Figures 21 and 22 with the camera pointing toward the airport. These simulated views of the aircraft camera are shown in Figures 23 and 24, respectively.

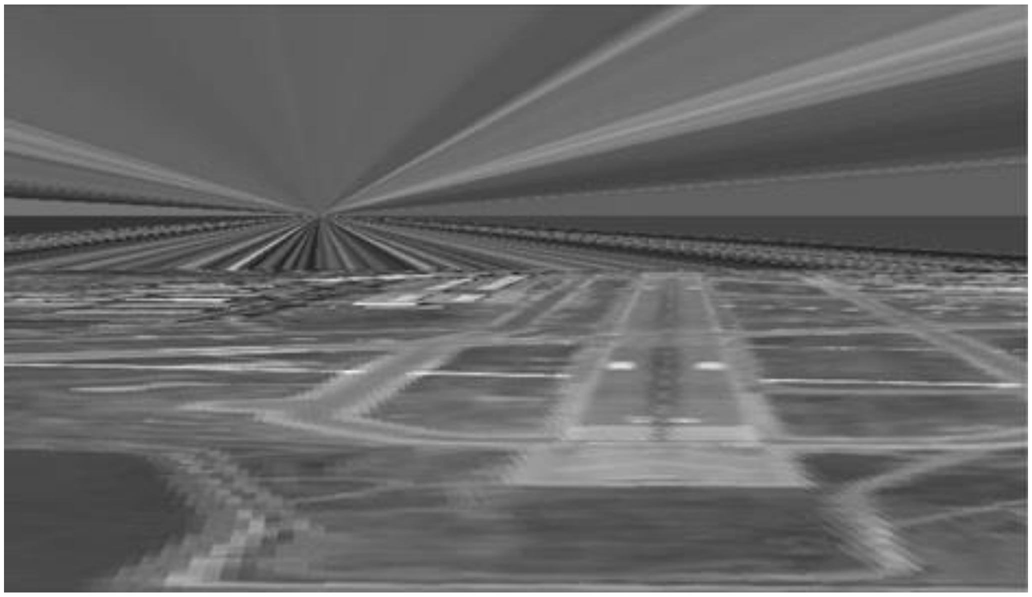

Figure 23. Virtual Camera View of Salinas Municipal Airport

Adapted from [7]: “Salinas Municipal Airport.” 2016. [Online]. Available: https://www.google.com/maps/@36.6612306,-121.6066075,2805m/data=!3m1!1e3. Accessed Jan. 22, 2016.

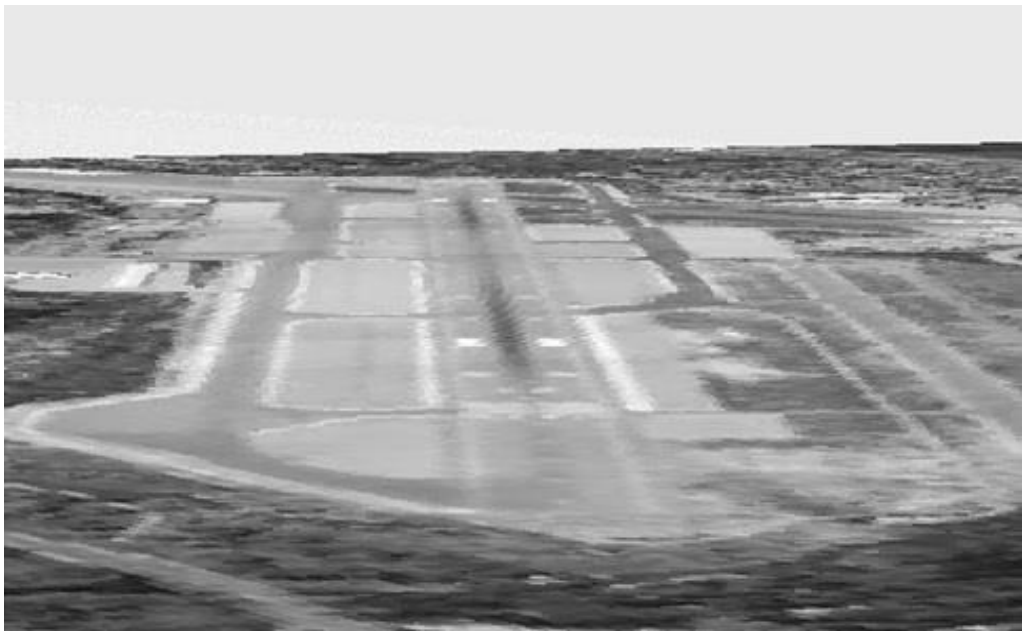

Figure 24. Virtual Camera View of Monterey Regional Airport

Adapted from [11]: “Monterey Regional Airport.” 2016. [Online]. Available: https://www.google.com/maps/@36.5857303,-121.8427775,2790m/data=!3m1!1e3 Accessed Jan. 22, 2016.

Once the virtual view is presented, we need to validate the matching of the edges of the runway in the map with the same edges in the virtual view. To do so, we add four points, which form two red lines in the map frame, as shown in Figures 25 and 26.

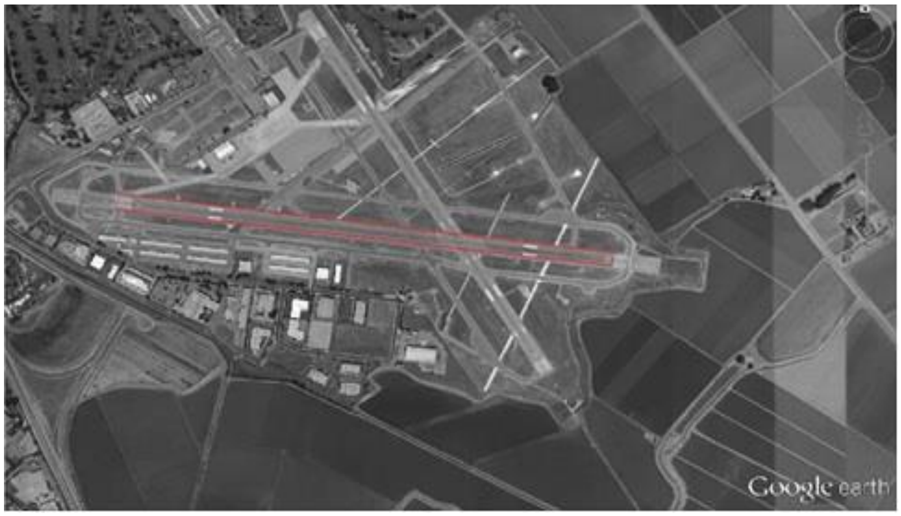

Figure 25. Salinas Municipal Airport with Runway Edges Highlighted in Red

Adapted from [7]: "Salinas Municipal Airport." 2016. [Online]. Available: https://www.google.com/maps/@36.6612306,-121.6066075,2805m/data=!3m1!1e3. Accessed Jan. 22, 2016.

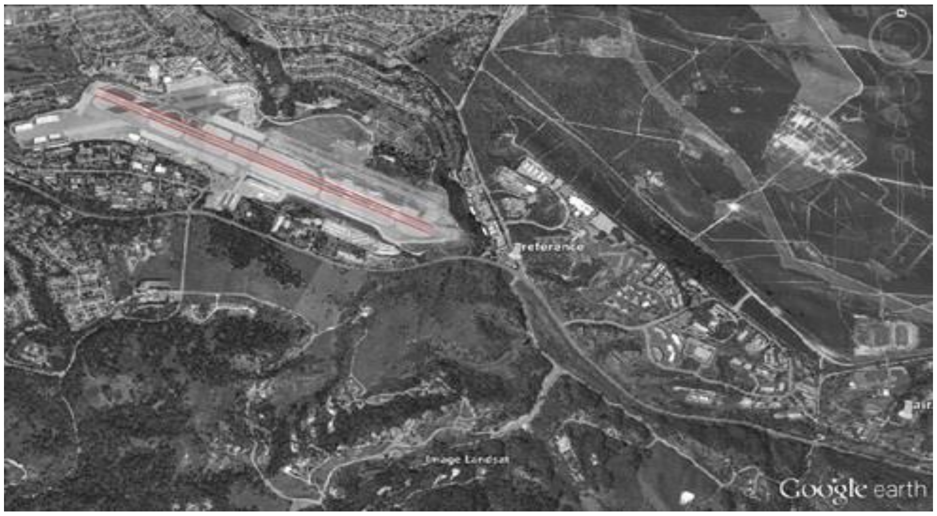

Figure 26. Monterey Regional Airport with Runway Edges Highlighted in Red

Adapted from [11]: "Monterey Regional Airport." 2016. [Online]. Available: https://www.google.com/maps/@36.5857303,-121.8427775,2790m/data=!3m1!1e3 Accessed Jan. 22, 2016.

The algorithm takes these four points on the map frame and performs a transformation to get the same points in the virtual image frame. A virtual view of the camera frame (highlighted in yellow) superimposed on the map frame is shown in Figures 27 and 28.

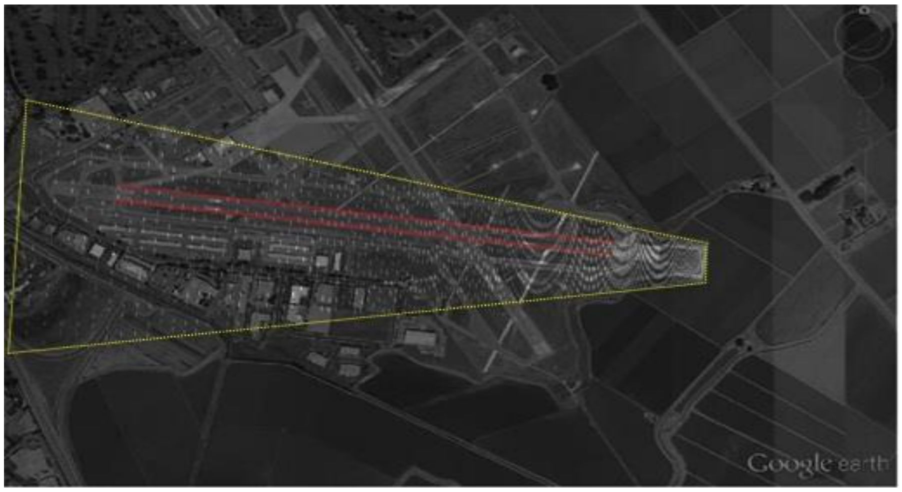

Figure 27. Virtual Frame (Yellow) of the Camera on Salinas Municipal Airport

Adapted from [7]: "Salinas Municipal Airport." 2016. [Online]. Available: https://www.google.com/maps/@36.6612306,-121.6066075,2805m/data=!3m1!1e3. Accessed Jan. 22, 2016.

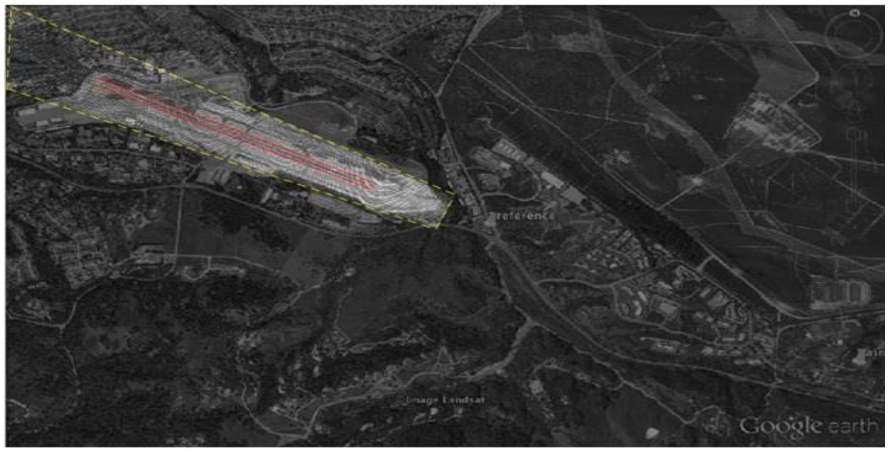

Figure 28. Virtual Frame (Yellow) of the Camera on Monterey Regional Airport

Adapted from [11]: "Monterey Regional Airport." 2016. [Online]. Available: https://www.google.com/maps/@36.5857303,-121.8427775,2790m/data=!3m1!1e3 Accessed Jan. 22, 2016.

In Figures 29 and 30, the virtual view of the camera is depicted with the runway edges highlighted in red. As illustrated, the correspondence between the map frame runway and the virtual image frame runway was achieved.

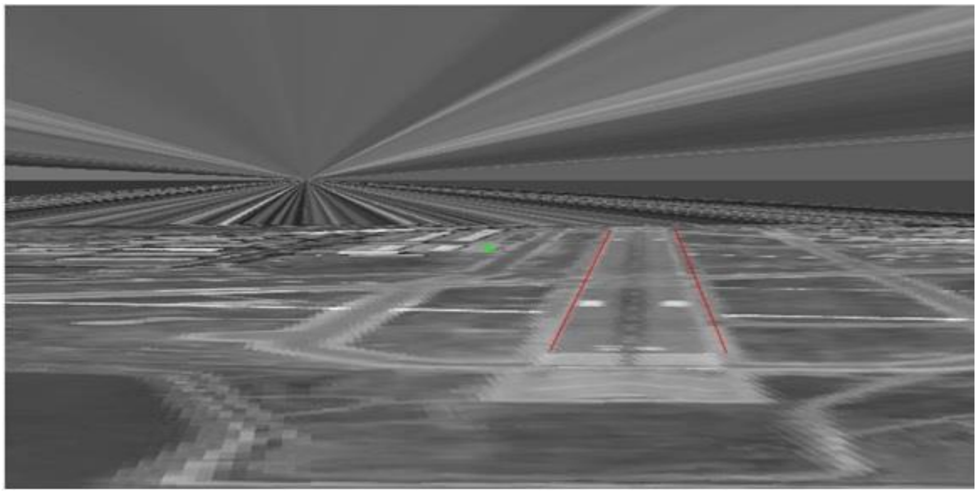

Figure 29. Runway Detected in the Camera Virtual View of Salinas Municipal Airport

Adapted from [7]: "Salinas Municipal Airport." 2016. [Online]. Available: https://www.google.com/maps/@36.6612306,-121.6066075,2805m/data=!3m1!1e3. Accessed Jan. 22, 2016.

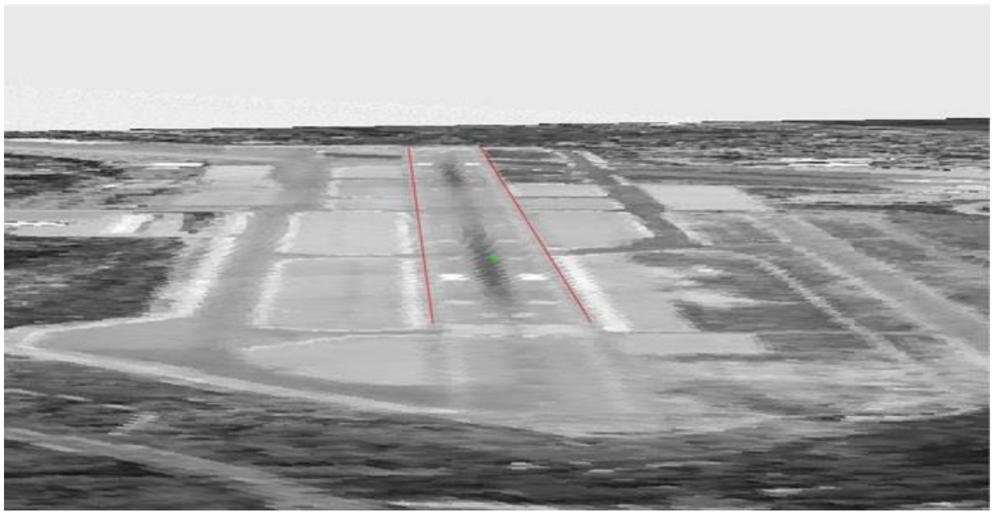

Figure 30. Runway Detected in the Camera Virtual View of Monterey Regional Airport

Adapted from [11]: "Monterey Regional Airport." 2016. [Online]. Available: https://www.google.com/maps/@36.5857303,-121.8427775,2790m/data=!3m1!1e3 Accessed Jan. 22, 2016.

We have described how the rotation matrices and the equations that relate the points on the map frame with those on the camera image frame are implemented in the MATLAB algorithm and shown that it is feasible to match the runway edges on the map frame with their corresponding edges in the image frame.

The values selected for the simulated navigational data were purposely set equal to the ones from the real experiment. By doing this, are able to compare the results from this simulation with the real experiment that is described in the next chapter.

# IV. MAP AND REAL IMAGE MATCHING

We have observed how the correspondence between the map and the virtual image is achieved in an ideal simulated environment. In this chapter, we describe how to associate the map frame and the real image frame.

Based on Chapter III and equations (46) and (47), the rotations performed by the developed function called map2image to go from the map frame to the image frame are essentially the same. The key variation is that we are using real parameters, namely navigational data of the aircraft (roll, pitch, and yaw), orientation angles of the camera (roll, tilt, and pan) and position of the aircraft (latitude, longitude, and altitude).

The real navigational data and camera orientation angles were obtained from another MATLAB algorithm that extracts the metadata from the frames of a video recorded by a TASE200 camera, which is mounted on a Cessna 206 airplane. Data for Salinas and Monterey airports were collected on several flights performed in 2015.

It is important to take into account that the input value of the aircraft position (posNEAw) for the MATLAB function map2image should be given in meters with respect to a defined reference in the map frame. This reference could be anywhere in the map, but for convenience, we utilized the map center.

To find this particular variable we proceeded as follows: first, we obtained the aircraft latitude and longitude from the data, and then we took the difference in latitude and longitude between this position and the reference and converted the result into meters. This should be the input value for the aircraft position.

## A. REAL DATA COLLECTION

In June and December 2015, the Cessna 206 airplane made several flights with a mounted TASE200 daylight and infrared camera, manufactured by Cloud Cap Technology, which is illustrated in Figure 31.

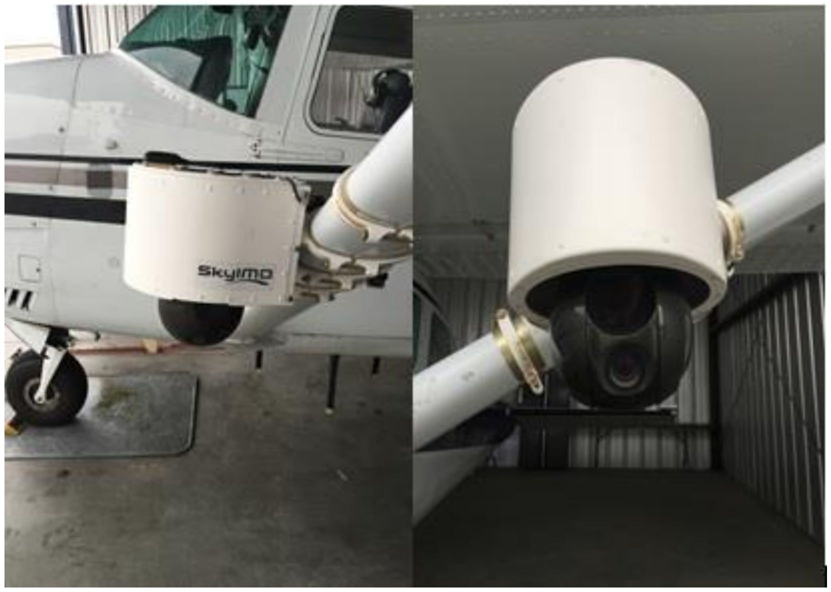

Figure 31. Camera TASE200 Mounted on the Cessna 206 Airplane

The main characteristics of the camera are shown in Table 1.

Table 1. TASE200 Camera Specifications

| MECHANICAL SPECIFICATIONS | |
|---|---|
| Diameter | 4.4 inches |
| Height | 7.5 inches |
| Weight | 2.34 lbs |
| PERFORMANCE | |
| Use | Daylight and infrared imaging |
| Pan limits | 360˚ continuous |
| Tilt limits | +23˚/-203˚ |
| IR camera | Resolution:640x480<br>HFOV: 10.5˚ |
| Daylight camera | Optical zoom: 31x<br>HFOV:55.7˚-1.94˚ |

Adapted from [9] *Cloud Cap Technology TASE 200 camera*. [Online]. Available: http://www.cloudcaptech.com/products/detail/tase-200. Accessed Jan. 25, 2016.

During these flights, several videos of approaches to the Salinas and Monterey airports were recorded. The videos were then processed by using the MATLAB algorithm called ReadingJpegSeries.m that reads each of the video frames separately and extracts useful navigational information (metadata) about the flights.

### 1. Salinas Municipal Airport Approximation

The two-minute video of the approach to the Salinas Municipal Airport was composed of 638 frames. As a reference, the data generated from the first ten frames is shown in Tables 2 and 3.

Table 2. Aircraft and Camera Orientation Angles for the First Ten Frames of the Salinas Municipal Airport Approximation

| Frame | Aircraft orientation | | | Camera orientation | | |
|---|---|---|---|---|---|---|
| | Roll | Pitch | Yaw | Roll | Tilt | Pan |
| 1 | -1.5985° | -19.7326° | -78.5525° | 2.7437° | -2.5517° | -91.0439° |
| 2 | -1.5985° | -19.7326° | -78.5525° | 2.7437° | -2.5517° | -91.0439° |
| 3 | -1.9938° | -19.6925° | -78.7301° | 2.4309° | -2.4947° | -91.5041° |
| 4 | -1.9938° | -19.6925° | -78.7301° | 2.4309° | -2.4947° | -91.5041° |
| 5 | -1.9938° | -19.6925° | -78.7301° | 2.4309° | -2.4947° | -91.5041° |
| 6 | -1.9938° | -19.6925° | -78.7301° | 2.4309° | -2.4947° | -91.5041° |
| 7 | -1.9938° | -19.6925° | -78.7301° | 2.4309° | -2.4947° | -91.5041° |
| 8 | -1.9938° | -19.6925° | -78.7301° | 2.4309° | -2.4947° | -91.5041° |
| 9 | -1.9938° | -19.6925° | -78.7301° | 2.4309° | -2.4947° | -91.5041° |
| 10 | -1.9938° | -19.6925° | -78.7301° | 2.4309° | -2.4947° | -91.5041° |

Table 3. Aircraft Position and Horizontal Field of View of the Camera for the First Ten Frames of the Salinas Municipal Airport Approximation

| Frame | Aircraft Position | | | HFOV |
|---|---|---|---|---|
| | Altitude (m) | Latitude | Longitude | |
| 1 | 60.96 | 36.6601° | -121.5906° | 20.83° |
| 2 | 60.96 | 36.6601° | -121.5906° | 20.83° |
| 3 | 59.97 | 36.6601° | -121.5907° | 20.83° |
| 4 | 59.97 | 36.6601° | -121.5907° | 20.83° |
| 5 | 59.97 | 36.6601° | -121.5907° | 20.83° |
| 6 | 59.97 | 36.6601° | -121.5907° | 20.83° |
| 7 | 59.97 | 36.6601° | -121.5907° | 20.83° |
| 8 | 59.97 | 36.6601° | -121.5907° | 20.83° |
| 9 | 59.97 | 36.6601° | -121.5907° | 20.83° |
| 10 | 59.97 | 36.6601° | -121.5907° | 20.83° |

The overall altitude and heading of the aircraft during the length of the video (638 frames) is depicted in Figure 32.

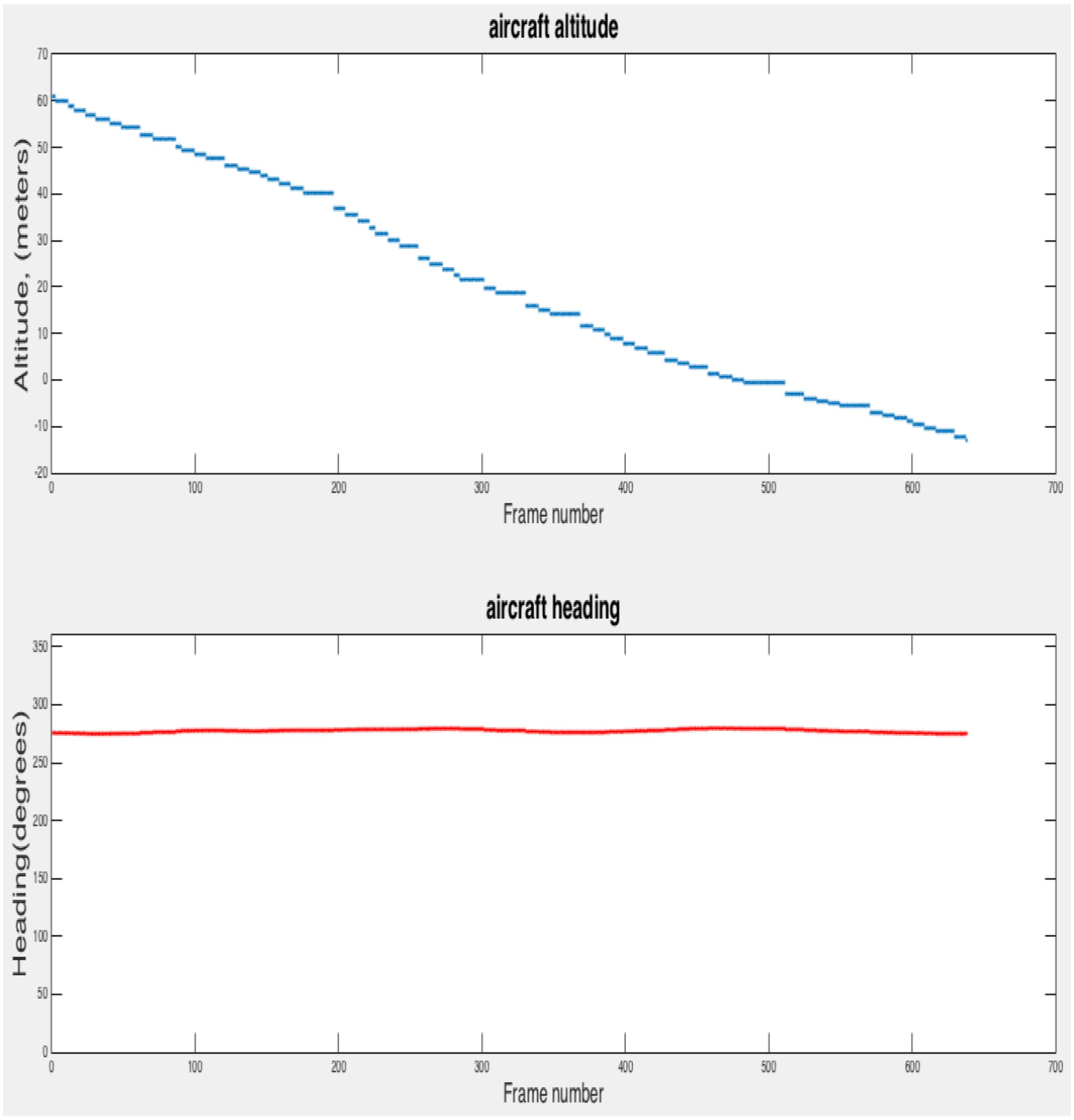

Figure 32. Aircraft Altitude and Heading for All the Frames of the Salinas Municipal Airport Approximation

Equally, the aircraft and the camera orientation angles are depicted in Figure 33.

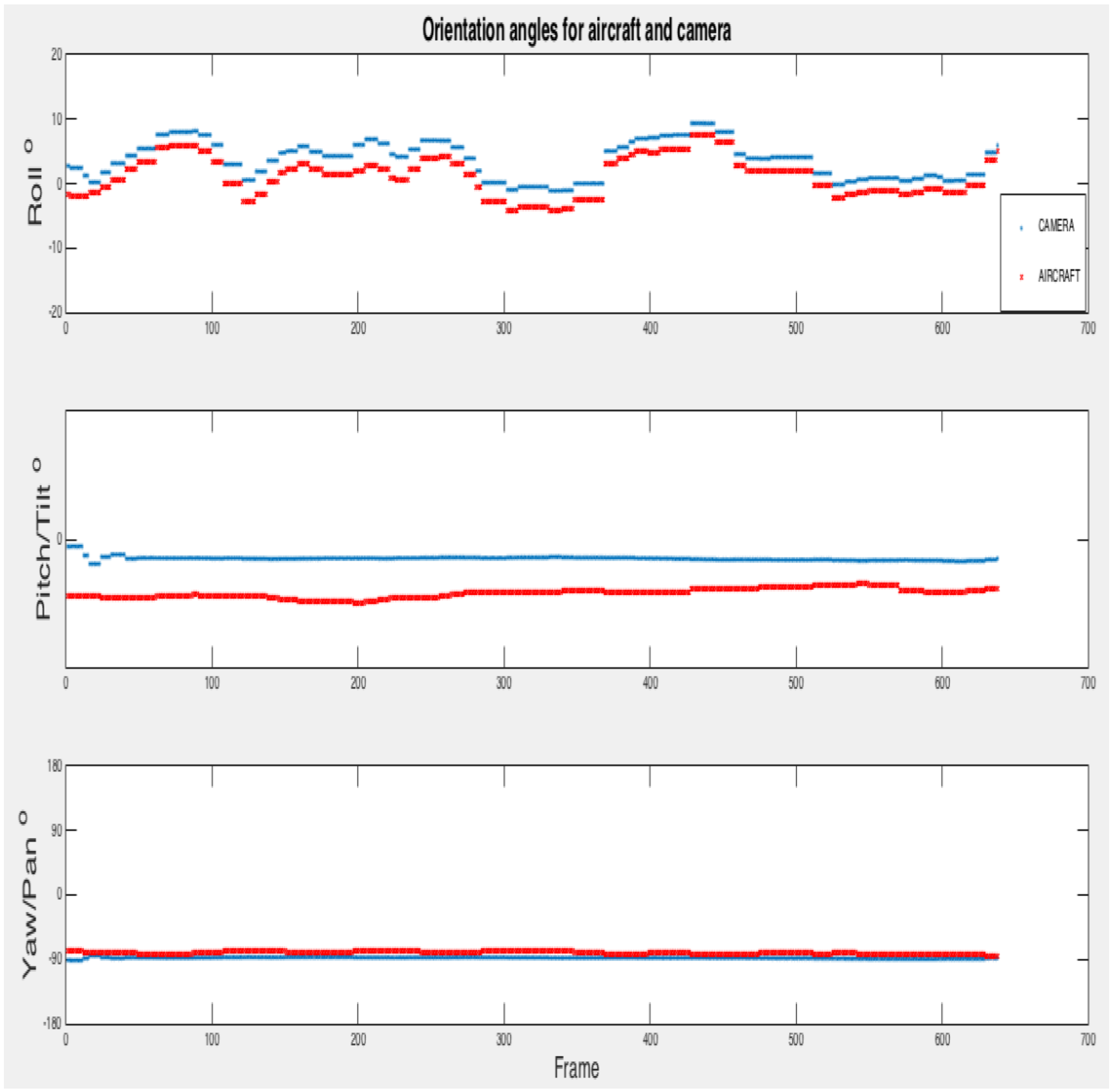

Figure 33. Aircraft and Camera Orientation Angles for All the Frames of the Salinas Municipal Airport Approximation

## 2. Monterey Regional Airport Approximation

Similar to the procedure for Salinas Municipal Airport, a two-minute video of the approach to the Monterey airport was also recorded. This video is composed of 3862 frames. The data generated from the first ten frames of this approach is shown in Tables 4 and 5.

Table 4. Aircraft and Camera Orientation Angles for the First Ten Frames of the Monterey Regional Airport Approximation

| Frame | Aircraft orientation | | | Camera orientation | | |
|---|---|---|---|---|---|---|
| | Roll | Pitch | Yaw | Roll | Tilt | Pan |
| 1 | 7.9412° | -21.9672° | -92.9452° | 1.7245 | -4.0699° | -76.6173 |
| 2 | 7.9412° | -21.9672° | -92.9452° | 1.7245 | -4.0699° | -76.6173 |
| 3 | 7.9412° | -21.9672° | -92.9452° | 1.7245 | -4.0699° | -76.6173 |
| 4 | 7.9412° | -21.9672° | -92.9452° | 1.7245 | -4.0699° | -76.6173 |
| 5 | 7.9412° | -21.9672° | -92.9452° | 1.7245 | -4.0699° | -76.6173 |
| 6 | 7.9412° | -21.9672° | -92.9452° | 1.7245 | -4.0699° | -76.6173 |
| 7 | 7.9412° | -21.9672° | -92.9452° | 1.7245 | -4.0699° | -76.6173 |
| 8 | 7.9412° | -21.9672° | -92.9452° | 1.7245 | -4.0699° | -76.6173 |
| 9 | 7.3682° | -21.8469° | -93.1400° | 1.0947 | -4.1112° | -76.6300 |
| 10 | 7.3682° | -21.8469° | -93.1400° | 1.0947 | -4.1112° | -76.6300 |

Table 5. Aircraft Position and Horizontal Field of View of the Camera for the First Ten Frames of the Monterey Regional Airport Approximation

| Frame | Aircraft Position | | | HFOV |
|---|---|---|---|---|
| | Altitude (m) | Latitude | Longitude | |
| 1 | 460.50 | 36.5642° | -121.7753° | 6.2338° |
| 2 | 460.50 | 36.5642° | -121.7753° | 6.2338° |
| 3 | 460.50 | 36.5642° | -121.7753° | 6.2338° |
| 4 | 460.50 | 36.5642° | -121.7753° | 6.2338° |
| 5 | 460.50 | 36.5642° | -121.7753° | 6.2338° |
| 6 | 460.50 | 36.5642° | -121.7753° | 6.2338° |
| 7 | 460.50 | 36.5642° | -121.7753° | 6.2338° |
| 8 | 460.50 | 36.5642° | -121.7753° | 6.2338° |
| 9 | 459.12 | 36.5642° | -121.7754° | 6.2338° |
| 10 | 459.12 | 36.5642° | -121.7754° | 6.2338° |

The overall altitude and heading of the aircraft during the length of the video is shown in Figure 34.

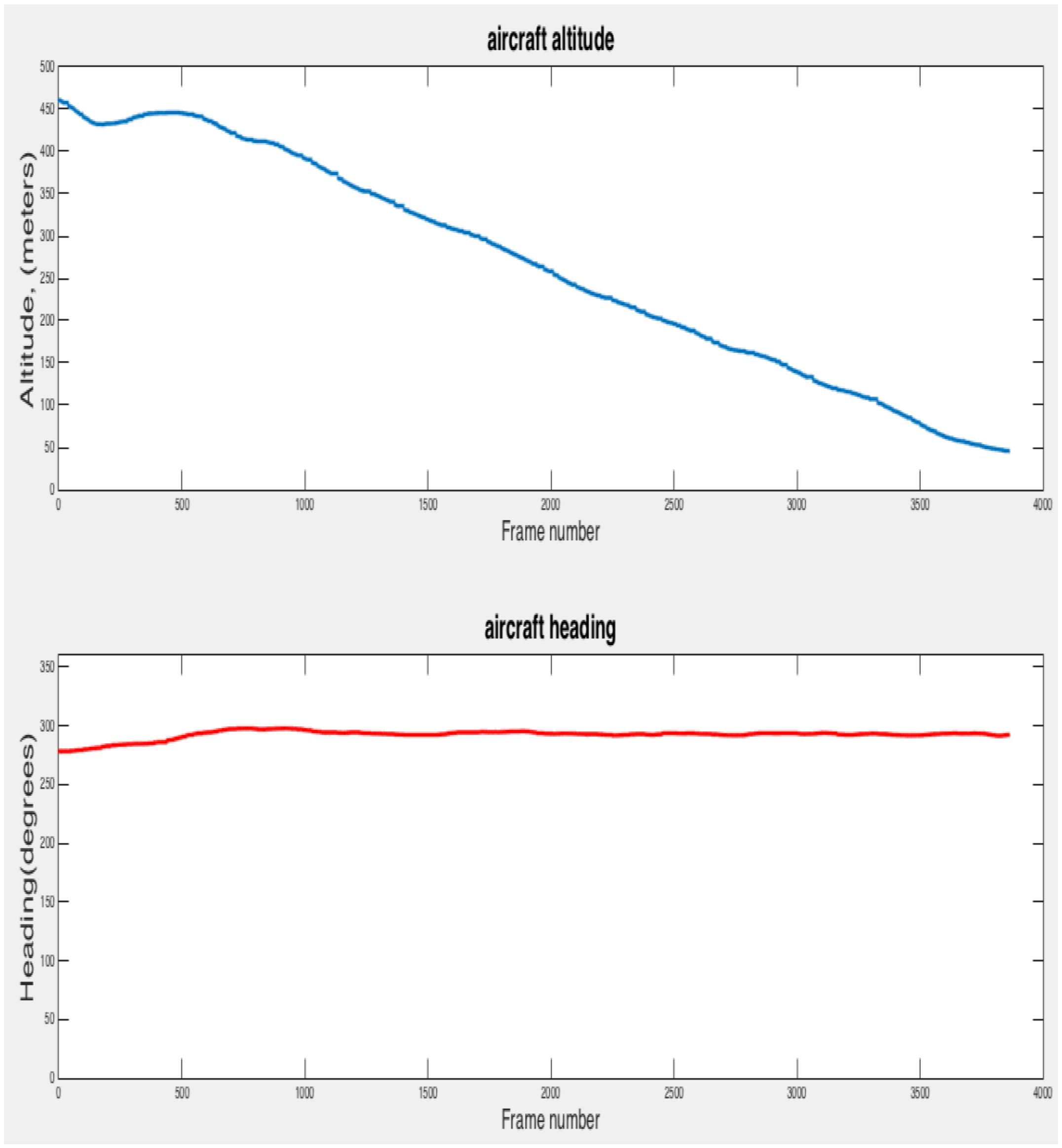

Figure 34. Aircraft Altitude and Heading for All the Frames of the Monterey Regional Airport Approximation

Likewise, the orientation angles of camera and aircraft for the Monterey approach are depicted in Figure 35.

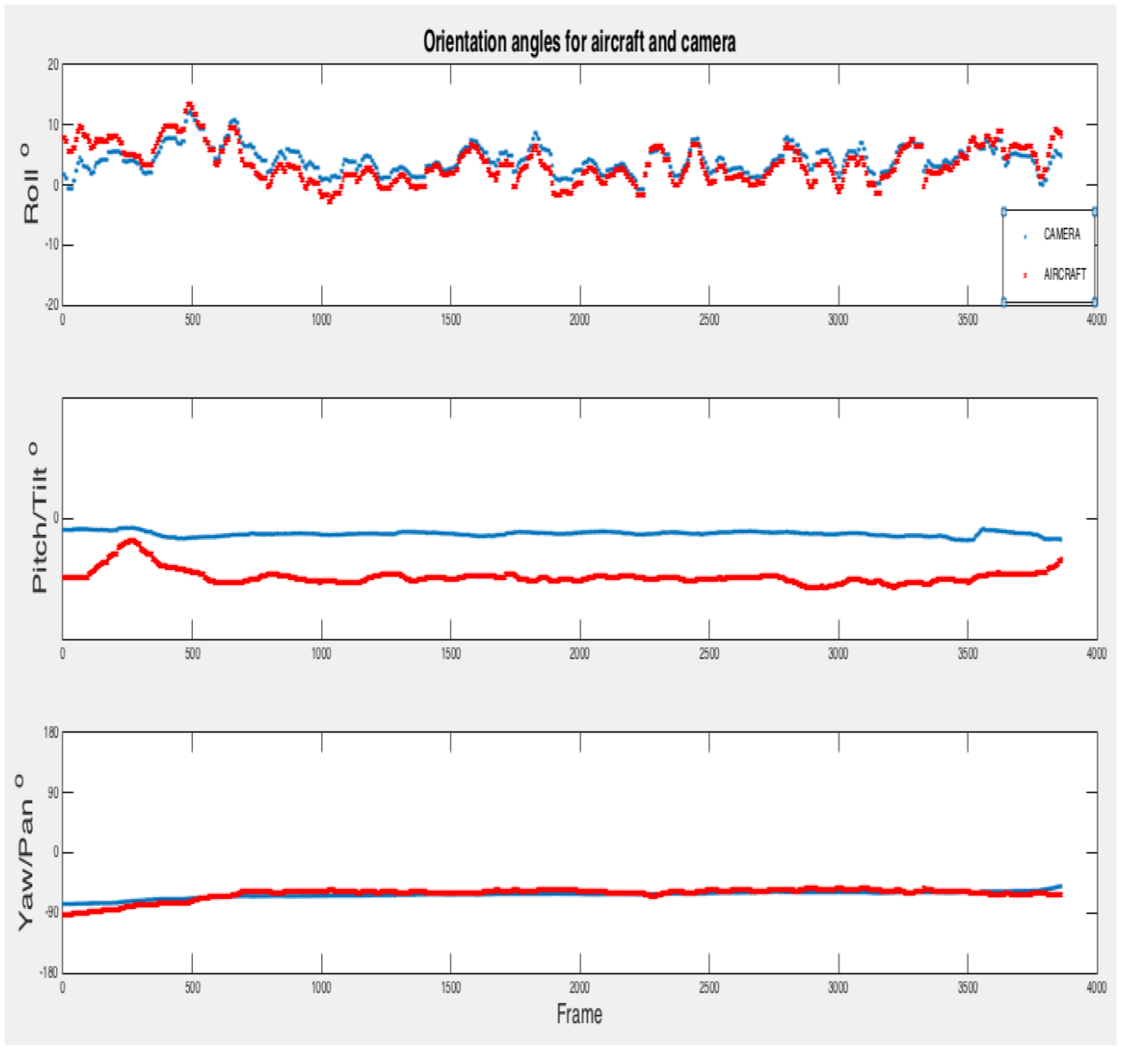

Figure 35. Aircraft and Camera Orientation Angles for All the Frames of the Monterey Regional Airport Approximation

## B. REAL DATA EXPERIMENT

Once the navigational information had been gathered from the metadata, we extracted the parameters needed for the MATLAB algorithm in order to perform the map and real-image runway matching. For this implementation, we started by defining four points on the map frame as described in page 32 of Chapter III in order to generate the two lines of the runway. Next, we substituted the real values from Tables 2, 3, 4 and 5 into the variables needed for the algorithm.

With this data, the algorithm performed a transformation from the map frame to the image frame, projecting the runway edges into the real image as depicted in Figures 36 and 37.

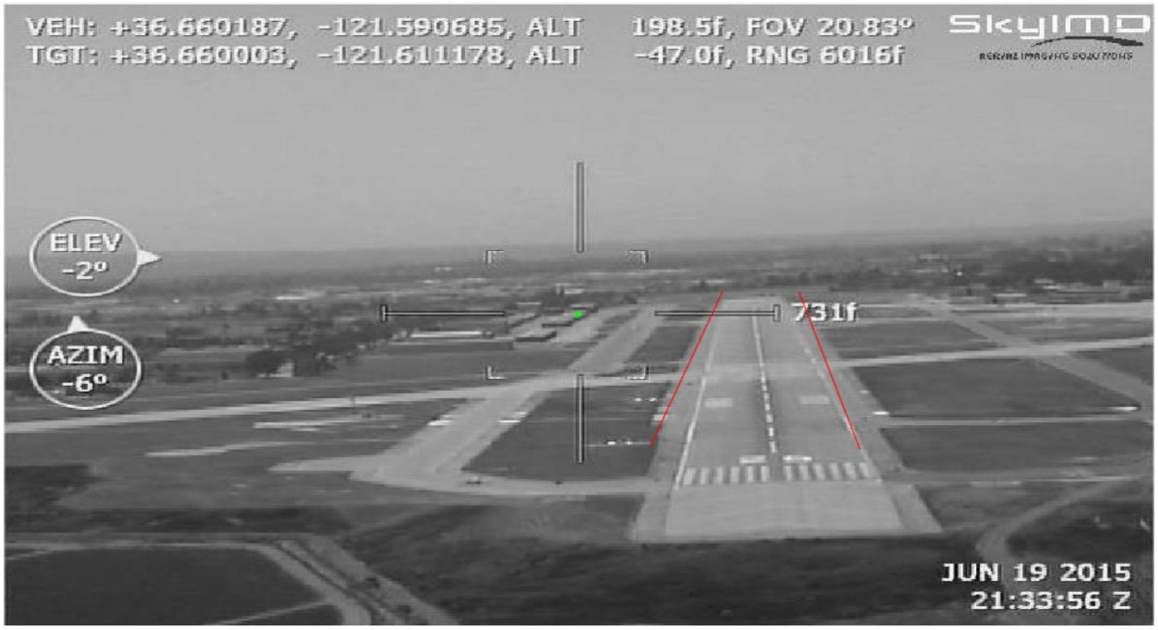
VEH: +36.660187, -121.590685, ALT 198.5f, FOV 20.83°
TGT: +36.660003, -121.611178, ALT -47.0f, RNG 6016f
SkyIMD
ELEV -2°
AZIM -6°
731f
JUN 19 2015
21:33:56 Z

Figure 36. Runway Detected on the Real Camera View from Salinas Municipal Airport

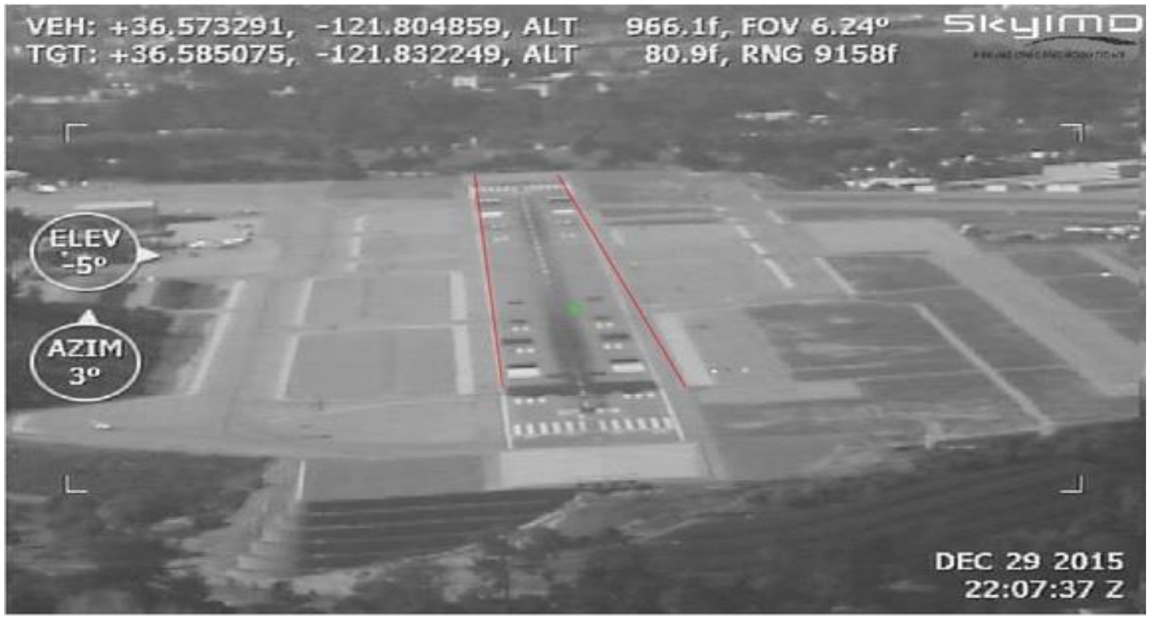
VEH: +36.573291, -121.804859, ALT 966.1f, FOV 6.24°
TGT: +36.585075, -121.832249, ALT 80.9f, RNG 9158f
SkyIMD
ELEV -5°
AZIM 3°
DEC 29 2015
22:07:37 Z

Figure 37. Runway Detected on the Real Camera View from Monterey Regional Airport

If we compare the results of the simulation against the experimental implementation, we notice that there is a slight mismatch when projecting the map runway lines into the real frame. In other words, the projected lines do not lie exactly on the image runway.

This discrepancy is due to several factors, including the presence of sensor noise as well as a simplifying assumption that the aircraft roll is close to zero. This assumption was made because aircrafts usually have small roll angles when they are approaching a runway. This simplified the equations derived in Chapter II.

The comparison between the simulations and the real experiments is illustrated in Figures 38 and 39, where it is shown that the estimated line boundaries of the runway must be corrected. This operation is performed by the Hough transform of the image and estimating the closest features in the image.

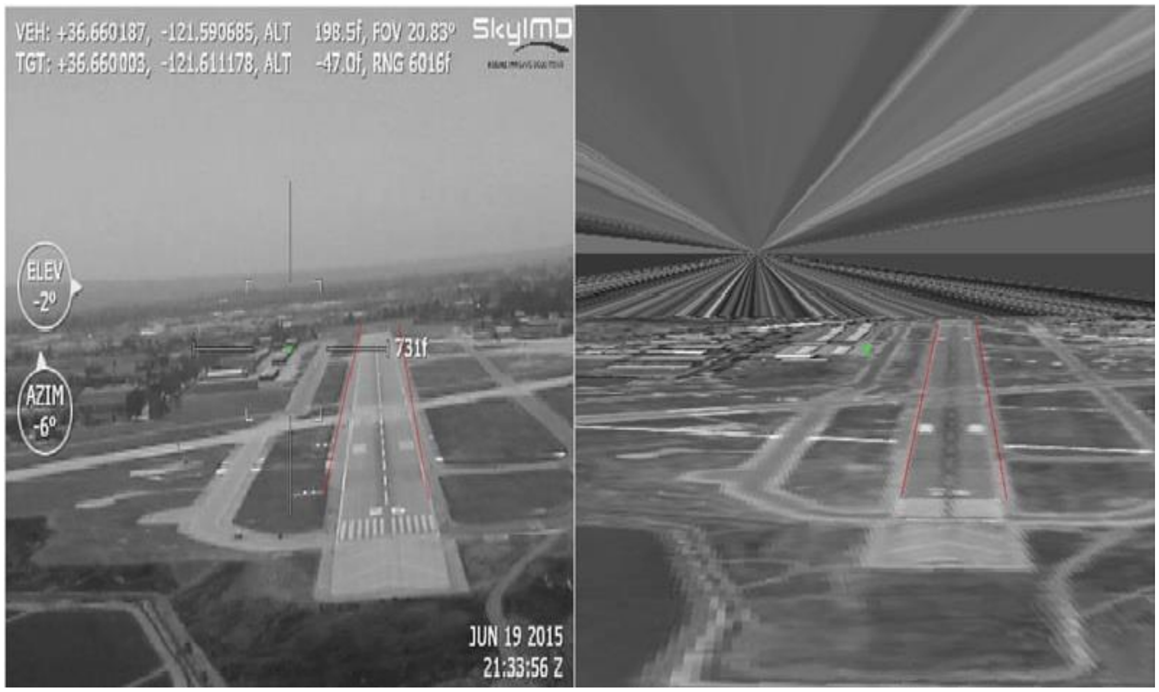
VEH: +36.660187, -121.590685, ALT 198.5f, FOV 20.83°
TGT: +36.660003, -121.611178, ALT -47.0f, RNG 6016f
SkyIMD
ELEV -2°
AZIM -6°
731f
JUN 19 2015
21:33:56 Z

Figure 38. Line Projection Mismatch in Real Implementation (Left) and Match in Simulation (Right) for Salinas Municipal Airport

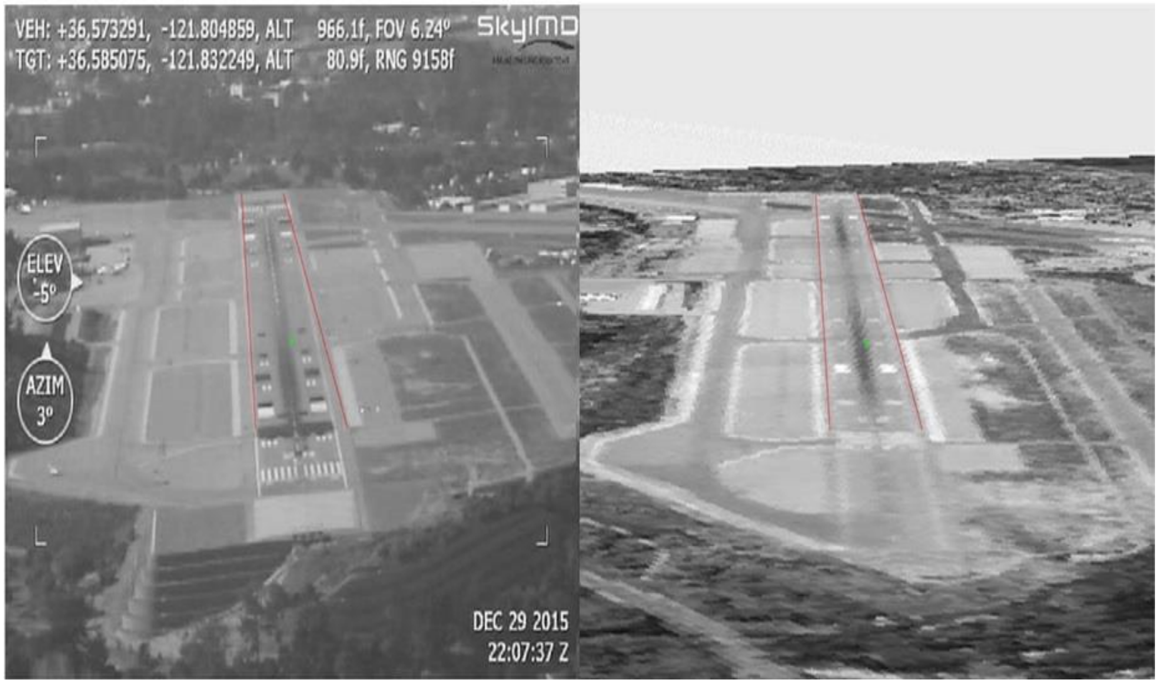
VEH: +36,573291, -121.804859, ALT 966.1f, FOV 6.24°
TGT: +36,585075, -121.832249, ALT 80.9f, RNG 9158f
SkyIMD
ELEV -5°
AZIM 3°
DEC 29 2015
22:07:37 Z

Figure 39. Line Projection Mismatch in Real Implementation (Left) and Match in Simulation (Right) for Monterey Regional Airport

### 1. Matching Line Parameters $\alpha$ and $\beta$

As discussed in Chapter II, the equations that describe the relationship between the line in the map frame and the line in the image frame were defined as

$$\beta = \frac{\overline{f}(\sin(\delta)h + \cos(\delta)b)}{\cos(\delta)h - \sin(\delta)b}$$
$$\alpha = \frac{ha}{\cos(\delta)h - \sin(\delta)b}$$
$$a = \frac{\overline{f}}{\cos(\delta)\overline{f} + \sin(\delta)\beta}\alpha$$
$$b = \frac{\cos(\delta)\beta - \sin(\delta)\overline{f}}{\cos(\delta)\overline{f} + \sin(\delta)\beta}h \; .$$

These equations were implemented within the main MATLAB algorithm to test and validate them. The percentage of error between slope α and intercept β obtained by using (34) and (35) and their values when using two points of the projected lines to calculate α and β are shown in Table 6 and Table 7.

Table 6. Comparison between Slope α and Intercept β Values for Salinas Municipal Airport Using Two Different Methods

| Using equations (19) and (20) | | Calculating slope and intercept | | Error (%) |
|---|---|---|---|---|
| Parameter | Value | Parameter | Value | |
| α | -2.6366 | α | -2.6364 | 0.0099 |
| β | 213.2586 | β | 212.3636 | 0.4214 |

Table 7. Comparison between Slope α and Intercept β Values for Monterey Regional Airport Using Two Different Methods

| Using equations (19) and (20) | | Calculating slope and intercept | | Error (%) |
|---|---|---|---|---|
| Parameter | Value | Parameter | Value | |
| α | -11.1278 | α | -11.1538 | 0.2340 |
| β | -1096..3 | β | -1098.8 | 0.2217 |

## C. MISMATCH CORRECTION

As described in Chapter II, the Hough transform algorithm is an image processing technique that detects linear features. The correction presented here is based on the fact that the Hough transform performs the line detection on the real image about the edges that are closer to the projected lines.

The use of the Hough transform in conjunction with the estimate from map information helps to eliminate line features that are not related to the specific target under consideration (a runway in this case) such as highways, roads and lines of buildings. In this way, once the parameters of the runway are estimated, a search can be performed by the Hough algorithm to evaluate the image feature closest to the one estimated from the map.

This procedure helps to detect just the lines of interest on the image that have similar angles to the projected ones. The detected lines that are closest to the projected ones are the actual runway lines.

Once we determine the angles of the projected lines by inspection, $\theta_1$ and $\theta_2$, we need to perform another step prior to the Hough transform. This stage executes an edge detection algorithm on the real image, and from there, the line detection becomes more efficient. The result obtained on the image after this previous process is implemented is shown in Figure 40.

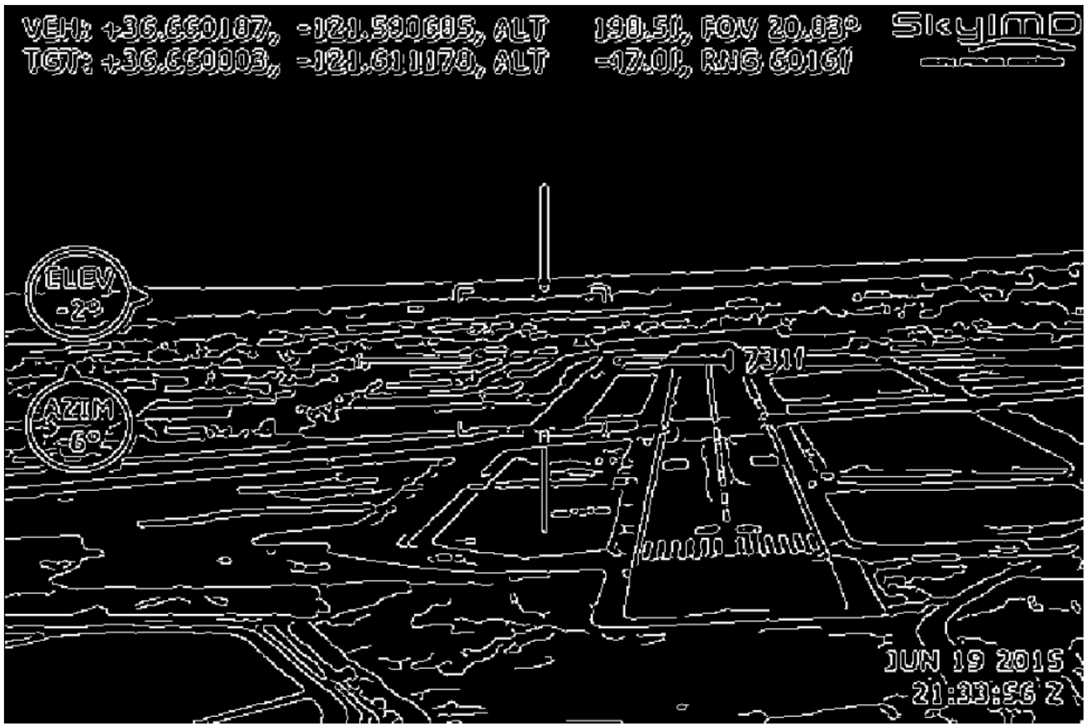
VEH: +36.660187, -121.590605, ALT 190.51, FOV 20.03°
TGT: +36.660003, -121.611178, ALT -17.01, RNG 6016
ELEV -2°
AZIM -6°
JUN 19 2015
21:33:56 Z

Figure 40. Edge Detection in the Real Image of Salinas Municipal Airport

The Hough transform can now be performed on the image depicted in Figure 40 for the lines closer to the angles $\theta_1$ and $\theta_2$. For angle $\theta_1$, the Hough transform detects the left external line of the runway, which is certainly closer to the projected left line (in red); this circumstance is shown in Figure 41.

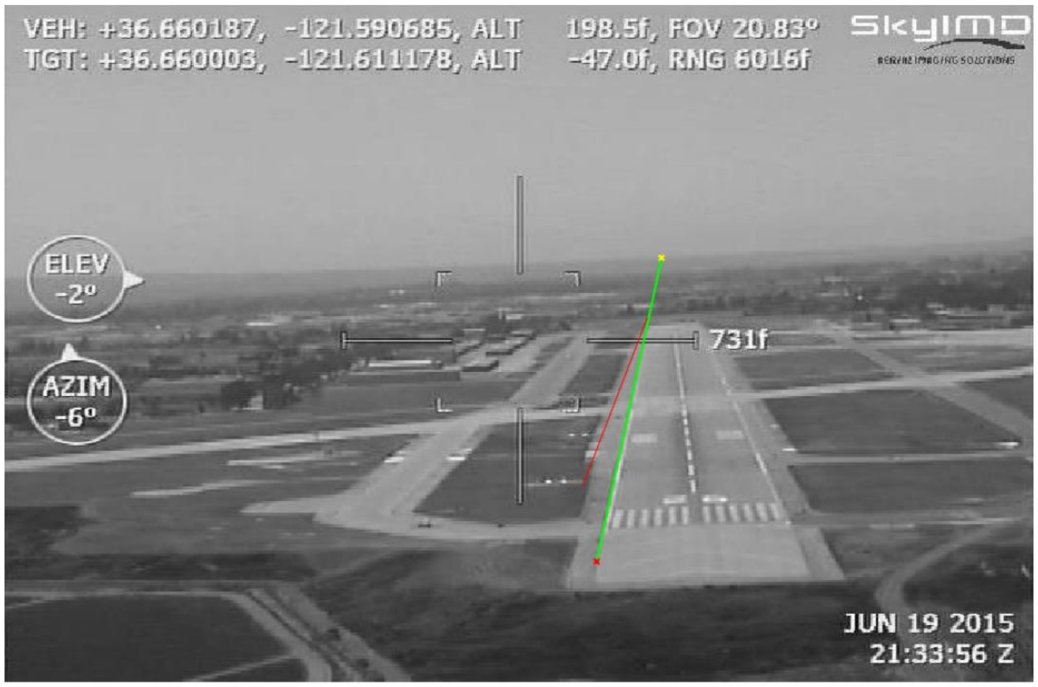
VEH: +36.660187, -121.590685, ALT 198.5f, FOV 20.83°
TGT: +36.660003, -121.611178, ALT -47.0f, RNG 6016f
SkyIMD
ELEV -2°
AZIM -6°
731f
JUN 19 2015
21:33:56 Z

Figure 41. Left Edge (Green) of Salinas Municipal Runway Detected by the Hough Transform Closer to the Projected Left Line (Red)

Likewise, for angle $\theta_{2,}$ the algorithm detects the right external line of the runway, which is in fact closer to the projected right line. This detection is illustrated in Figure 42.

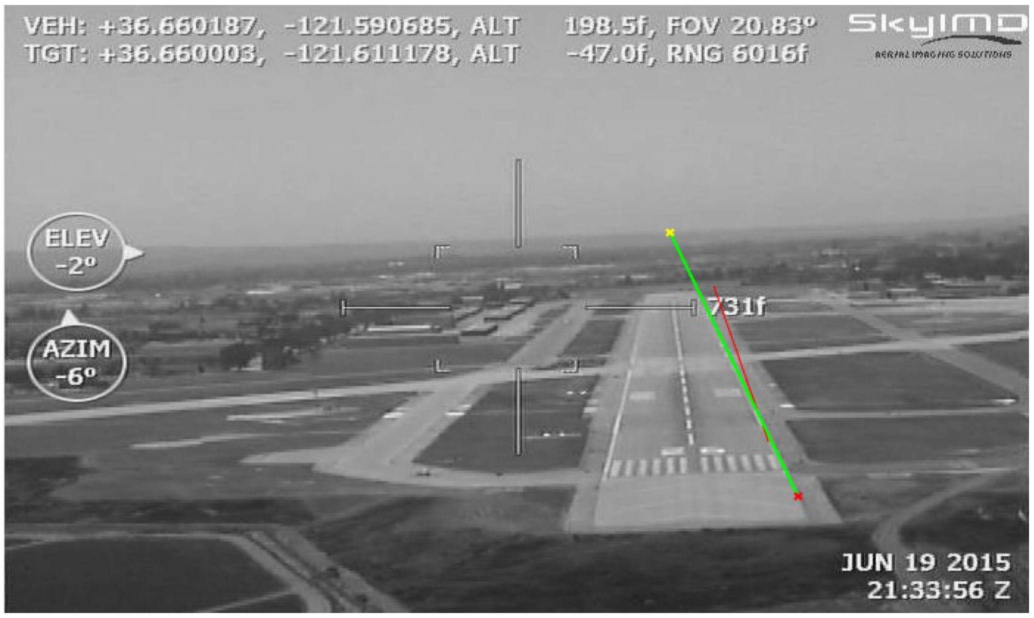
VEH: +36.660187, -121.590685, ALT 198.5f, FOV 20.83º
TGT: +36.660003, -121.611178, ALT -47.0f, RNG 6016f
ELEV -2º
AZIM -6º
731f
JUN 19 2015
21:33:56 Z

Figure 42. Right Edge (Green) of Salinas Municipal Runway Detected by the Hough Transform Closer to the Projected Right Line (Red)

By adding together both Hough transforms results, we can correct the mismatch and make the detection of the runway as shown in Figure 43.

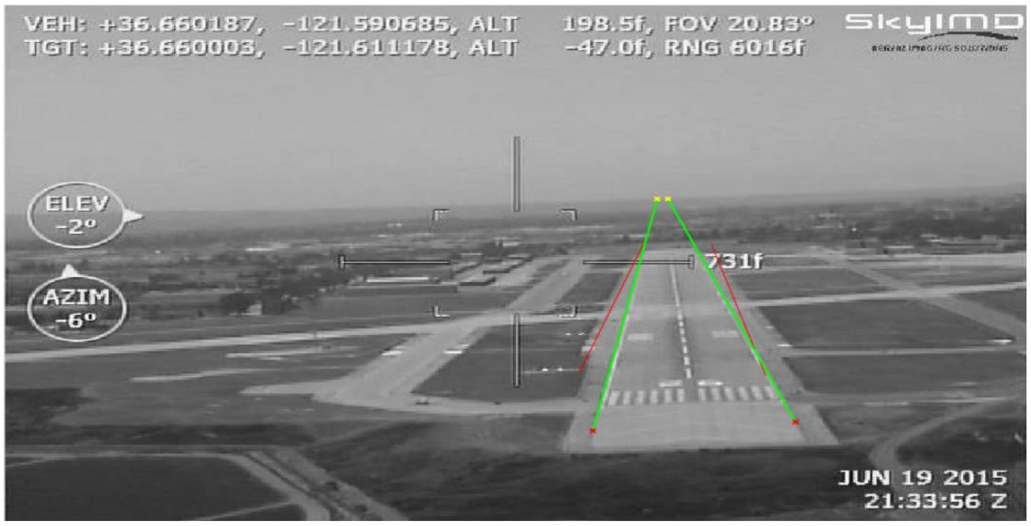
VEH: +36.660187, -121.590685, ALT 198.5f, FOV 20.83º
TGT: +36.660003, -121.611178, ALT -47.0f, RNG 6016f
ELEV -2º
AZIM -6º
731f
JUN 19 2015
21:33:56 Z

Figure 43. Mismatch in Projected Lines (Red) and Correction by Hough Transform (Green) in Salinas Municipal Airport

The same procedure is applied to the Monterey airport case. Recalling from Figure 37, we notice that the mismatch is just the right side of the runway; therefore, the Hough transform is performed only on this region. The right edge of the runway is subsequently detected as illustrated in Figure 44.

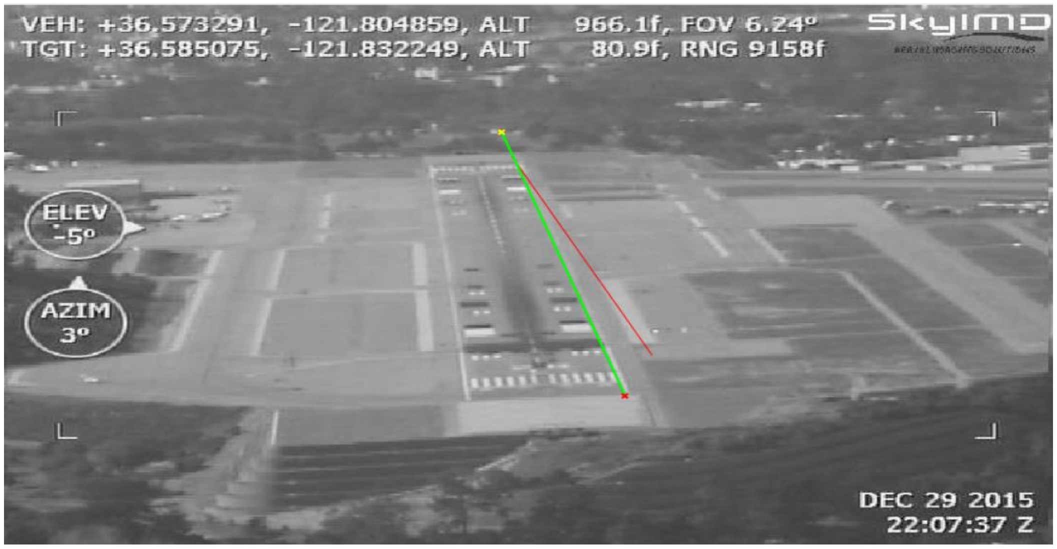
VEH: +36.573291, -121.804859, ALT 966.1f, FOV 6.24°
TGT: +36.585075, -121.832249, ALT 80.9f, RNG 9158f
SkyIMD
ELEV -5°
AZIM 3°
DEC 29 2015
22:07:37 Z

Figure 44. Mismatch in Right Projected Line and Correction by Hough Transform in Monterey Regional Airport

From the results on both airports, we observe that after the Hough transform correction, the runway detection was achieved in the experimental implementation. In addition, the results were better for the Monterey airport, which only needed the correction for one side of the runway, while the Salinas Municipal Airport data required the correction on both sides.

# V. CONCLUSIONS AND FUTURE WORK

## A. CONCLUSIONS

The main objective of this thesis was to present the development and implementation of a methodology that performs the detection of a runway from map, video and aircraft navigational data. The algorithm was tested in a simulation method where ideal conditions were set, and as a result, it achieved an accurate runway detection.

This methodology was then verified using actual video and aircraft data. The results were not as precise as the simulation due to factors such as noise present in the sensors as well as assumptions made on the aircraft roll behavior, which was assumed to be zero in order to simplify the equations derived in Chapter II. These discrepancies were corrected through the use of the Hough transform for linear features, which resulted in accurate detection of the runway.

As described in Chapter I, many approaches perform runway detection by image-processing techniques on video frames or satellite images. The approach presented in this thesis differs from others in that it uses the satellite map of the runway that we want to detect, the aircraft data, the camera data, and the video frames (image). The use of these four tools in the algorithm provide more data and improves upon the runway detection that can be achieved.

## B. FUTURE WORK

The work done in this thesis is the initial contribution to perform the detection of runways using not just an image by itself but data from other sources. Future work could address the suppression of the mismatching when the algorithm is implemented using real world video. By doing this, the algorithm will not need to perform the Hough transform, saving computational time.

The methodology presented here used one frame of video to achieve runway detection. It would be interesting to implement the algorithm on a group of frames in such a way that the detection results could be displayed on a sequence of video frames.

Also in the future, this method could be implemented to perform the runway detection in real time.

THIS PAGE INTENTIONALLY LEFT BLANK

# APPENDIX

This Appendix includes the MATLAB code implemented in this thesis.

```
%%///////////////////////////////////////////////////////////////////////
%Roberto Cristi /Jose Espinosa. Naval Postgraduate School ECE
Department.
%March 2016.
%This program has four parts:
%Part 1: Show image from map for different aircraft and camera
positions and orientations.
%Part 2: Show correspondence of lines between map and image.
%Part 3 : Compare parameters of corresponding lines in map and image:
%Part 4: Test Hough transform (to correct the matching map-real frame).
%%///////////////////////////////////////////////////////////////////////
clear
close all

%Load GoogleMap data.
Imap=imread('runway_map.jpg');
Imap=rgb2gray(Imap);
Nmap=size(Imap);

%Camera Parameters.
FOV=20.87; %Field of View
N=[464,696];%[rows, columns] of image.

%Aircraft rotation angles.
%Enter roll, pitch and yaw angles of aircraft in degrees.
thR =0; thP = 88; thY = -86;

%Camera rotation angles.
%Enter roll, tilt and pan angles of camera in degrees.
throll=0; thtilt=0; thpan=0;

%World frame: [North-East-Down (NED)]
%Aircraft Position  [North, East] (in pixels).Origin is center of
%the map divided by the scale.
Tne=[-97.543;1213.602]/1.25;

%Aircraft height (in meters) divided by the scale.
h=60/1.25;

%Scale of map (meters per pixel).
scale=1.0;

%% PART 1: Image from Map

%Compute fbar.
fbar=N(2)/(2*tan(FOV*pi/360));
```

```
%Aircraft position in World Frame (in meters with respect to center of
map):
Tw=[Tne; h];

thRPYaw=[thR; thP; thY]; %Roll, pitch and yaw of aircraft.
thRTPca=[throll; thtilt; thpan];%Roll, tilt and pan of camera.
posNEAw=Tw; %Position of aircraft.

%Compute points on map corresponding to pixels in image.
XYm=image2map(N, FOV, thRPYaw, thRTPca, posNEAw, scale);

%Highlight them on the map
im_min=min(XYm(1,:)+Nmap(1)/2); im_max=max(XYm(1,:)+Nmap(1)/2);
jm_min=min(XYm(2,:)+Nmap(2)/2); jm_max=max(XYm(2,:)+Nmap(2)/2);
Area_spanned=0.5*ones(Nmap);

%Compute intensities from map by interpolation (closest point in map
grid).
for i=1:N(1),
    for j=1:N(2),
        k=(j-1)*N(1)+i;  %Add shift from center.
        im=XYm(1,k)+(Nmap(1)/2);
        jm=XYm(2,k)+(Nmap(2)/2);
        im=max([im,1]); im=min(im, Nmap(1));
        jm=max([jm,1]); jm=min(jm, Nmap(2));
        Image(i,j)=Imap(uint16(im),uint16(jm));
        Area_spanned(uint16(im),uint16(jm))=1.0;
    end
end

%% PART 2: Correspondences between lines in map and image.
%Enter two points (A,B) on the map (origin left-upper corner, row and
%column).

Am=[593; 288]; Bm=[758; 1593]; % two points on one side of runway
Cm=[548; 295]; Dm=[722; 1602]; % two points on other side of runway

%Display on map.
Area_spanned=Area_spanned.*single(Imap);
figure, imshow(Area_spanned,[0,255])
hold on
XY=[Am, Bm, Cm, Dm];
plot(XY(2,1:2), XY(1,1:2), 'r') % line 1
plot(XY(2,3:4), XY(1,3:4), 'r') % line 2
XYcentermap=round(Nmap/2);  % origin of map
XYcenterscene=image2map2(N, N'/2, FOV, thRPYaw, thRTPca, posNEAw,
scale);
XYcenterscene=XYcenterscene+Nmap'/2;

%Display on Images (Estimated and Actual Frame).
%Convert to map coordinates (with respect to center of map).
Am=round(Am-Nmap'/2); Bm=round(Bm-Nmap'/2);
Cm=round(Cm-Nmap'/2); Dm=round(Dm-Nmap'/2);
XYm=[Am, Bm, Cm, Dm];
```

```
XYi=map2image(XYm,thRPYaw, thRTPca, posNEAw, N, FOV,scale);

%Estimated Image.
figure, imshow(Image),
title('estimated frame')
hold on
XYcenterimage=round(N/2);  % origin of map.
plot(XYcenterimage(2), XYcenterimage(1),'*g')
plot(XYi(2,1:2), XYi(1,1:2), 'r') % line 1.
plot(XYi(2,3:4), XYi(1,3:4), 'r') % line 2.

%Actual image.
Frame=imread('Test1.jpg');  %to compare with actual image.
Frame=rgb2gray(Frame);
figure, imshow(Frame),
title('actual frame')
hold on
XYcenterimage=round(N/2);  % origin of map.
plot(XYcenterimage(2), XYcenterimage(1),'*g')
plot(XYi(2,1:2), XYi(1,1:2), 'r') % line 1.
plot(XYi(2,3:4), XYi(1,3:4), 'r') % line 2.

%% PART 3 : Compare parameters of correponding lines in map and image:
%Line in map y=ax+b  (x=horizontal(column),
y=vertical(row),origin=center)

YXm=[XYm(2,:); XYm(1,:)]; % switch XY with YX.

%Compute "a" and "b" parameters from 2 points on the line in the map.
a=(YXm(2,1)-YXm(2,2))/(YXm(1,1)-YXm(1,2));
b=(YXm(2,1)-a*YXm(1,1))/2 + (YXm(2,2)-a*YXm(1,2))/2;

%Correct "a," "b" for shift.
as=a; bs=b+a*Tne(2)+Tne(1);
%Correct "a," "b for rotation.
sY=sin(thY*pi/180); cY=cos(thY*pi/180);
ar=(as*cY-sY)/(cY+as*sY); br=bs/(cY+as*sY);

%Line in image y=alpha x + beta
%Use the corresponding points  XYi on the image.
%First shift  to the reference in the center
XYi(:,1)=round(XYi(:,1)-N'/2); XYi(:,2)=round(XYi(:,2)-N'/2);

YXi=[XYi(2,:); XYi(1,:)]; % switch XY with YX.
%Compute "alpha" and "beta."
alpha=(YXi(2,1)-YXi(2,2))/(YXi(1,1)-YXi(1,2));
beta=(YXi(2,1)-alpha*YXi(1,1))/2 + (YXi(2,2)-alpha*YXi(1,2))/2;

%Verify correspondence alpa_hat and beta_hat with computed "alpha" and
"beta."
sP=sin(thP*pi/180); cP=cos(thP*pi/180);
alpha_hat=h*ar/(cP*h-sP*br);
beta_hat=fbar*(sP*h+cP*br)/(cP*h-sP*br);
```

```
%Should get alpha close to alpha_hat and beta close to beta_hat.

%Error percent between both calculaitons.
alpha_errorpercent=100*abs((alpha-alpha_hat)/alpha);
beta_errorpercent=100*abs((beta-beta_hat)/beta);

%% PART 4: Test Hough transform (to correct the matching map-real
frame).

%Left side of the runway
runway = rgb2gray(imread('Test1.jpg'));
X0=runway;
X=single(X0);
Y=edge(X,'canny',[]);
figure
imshow(Y,[0, max(max(Y))]);
[H, theta, rho]=hough(Y,'Theta',11.5:0.5:12.5);
figure
imshow(imadjust(mat2gray(H)),'XData',theta,'YData',rho,
'InitialMagnification','fit');
title('Hough Transform');
xlabel('\theta'), ylabel('\rho');
axis on,
axis normal,
hold on;
colormap(gray);

%Find peaks.
F  = houghpeaks(H,3);
hold on
plot(theta(F(:,2)),rho(F(:,1)),'s','color','green');

%Lines (show the correct lines of the runway in the real frame).
lines = houghlines(Y, theta, rho, F);
figure
imshow(X0), hold on

Am=[593; 288]; Bm=[758; 1593]; % Two points on one side of runway.
Cm=[548; 295]; Dm=[722; 1602]; % Two points on other side of runway.
Am=round(Am-Nmap'/2); Bm=round(Bm-Nmap'/2);
Cm=round(Cm-Nmap'/2); Dm=round(Dm-Nmap'/2);
XYm=[Am, Bm, Cm, Dm];
XYi=map2image(XYm,thRPYaw, thRTPca, posNEAw, N, FOV,scale);
plot(XYi(2,1:2), XYi(1,1:2), 'r') % line 1.

max_len = 0;
       for k = 1:length(lines)
           xy = [lines(k).point1; lines(k).point2];
           plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

           %Plot beginnings and ends of lines.
           plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
           plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
           %Determine the endpoints of the longest line segment.
           len = norm(lines(k).point1 - lines(k).point2);
           if (len > max_len)
              max_len = len;
              xy_long = xy;
           end
       end
title('left side')

%Right side of the runway.
[H, theta, rho]=hough(Y,'Theta',-23.5:0.5:-22.5);
figure
imshow(imadjust(mat2gray(H)),'XData',theta,'YData',rho,'InitialMagnific
ation','fit');
title('Hough Transform');
xlabel('\theta'), ylabel('\rho');
axis on,
axis normal,
hold on;
colormap(hot);

%Find peaks.
F  = houghpeaks(H,3);
hold on
plot(theta(F(:,2)),rho(F(:,1)),'s','color','green');

%Lines (show the correct lines of the runway in the real frame).
lines = houghlines(Y, theta, rho, F);
figure
imshow(X0), hold on

Am=[593; 288]; Bm=[758; 1593]; % Two points on one side of runway.
Cm=[548; 295]; Dm=[722; 1602]; % Two points on other side of runway.
Am=round(Am-Nmap'/2); Bm=round(Bm-Nmap'/2);
Cm=round(Cm-Nmap'/2); Dm=round(Dm-Nmap'/2);
XYm=[Am, Bm, Cm, Dm];
XYi=map2image(XYm,thRPYaw, thRTPca, posNEAw, N, FOV,scale);
plot(XYi(2,3:4), XYi(1,3:4), 'r') % line 2.

max_len = 0;
       for k = 1:length(lines)
           xy = [lines(k).point1; lines(k).point2];
           plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

         %Plot beginnings and ends of lines.
         plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
         plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');

         %Determine the endpoints of the longest line segment.
         len = norm(lines(k).point1 - lines(k).point2);
         if (len > max_len)
             max_len = len;
             xy_long = xy;
```

```
        end
      end

title('right side')



%%//////////////////////////////////////////////////////////////////////
%Roberto Cristi /Jose Espinosa. Naval Postgraduate School ECE
Department.
%March 2016.
%This function converts points on map frame to points on image frame.
%%//////////////////////////////////////////////////////////////////////
function XYi=map2image(XYm,thRPYaw, thRTPca, posNEAw, N, FOV, scale)
%XYi=map2image(XYm,thRPYaw, thRTPca, posNEAw, N, FOV, scale)
%XYi = 2xNpoints positions in Image Frame.
%XYm = 2xNpoints positions in Map Frame.
%thRPYaw = 3x1 Roll, Pitch Yaw angles(deg.)World to Aircraft Frames.
%thRTPca = 3x1 Roll, Tilt, Pitch angles(deg.) Aircraft to Camera
Frames.
%posNEAw = 3x1 North,East,Altitude of Aircraft in World Frame.
%FOV=Field of View of Camera in degrees.
%N=[number of rows, number of columns] of image. It has to be EVEN.
%scale = length, in meters, of one pixel in the map (meters/pixel).
%Map Frame: [Row (down), Column (right), Elevation (up)].
%World Frame: [North, East, Down].
%Aircraft Frame: [Roll (forward), Pitch (right), Yaw (down)].
%Camera Frame: [Roll (forward), Tilt (right), Pan (down)].
%Image Frame: Rows (Down), Columns (Left) , Down (View direction).
%Rotation Matrices (Rba=Rotation from frame "a" to frame "b").

%Map frame to world frame and viceversa.
Rwm=diag([-1, 1, -1]); Rmw=Rwm';

%Aircraft frame to world frame and viceversa.
thR=thRPYaw(1); thP=thRPYaw(2); thY=thRPYaw(3);
thR=thR*pi/180; thP=thP*pi/180; thY=thY*pi/180;

Ry=[cos(thY), sin(thY), 0;
    -sin(thY), cos(thY), 0;
    0,0,1];

Rp=[cos(thP), 0, sin(thP);
    0,1,0;
    -sin(thP), 0, cos(thP)];

Rr=[1,0,0;
    0 cos(thR), -sin(thR);
    0, sin(thR), cos(thR)];

Raw=Rr*Rp*Ry;        Rwa=Raw';

%Aircraft frame to camera frame and viceversa.
throll=thRTPca(1); thtilt=thRTPca(2); thpan=thRTPca(3);
```

```
throll=throll*pi/180; thtilt=thtilt*pi/180; thpan=thpan*pi/180;

Rpan=[cos(thpan), sin(thpan), 0;
    -sin(thpan), cos(thpan), 0;
    0,0,1];

Rtilt=[cos(thtilt), 0, -sin(thtilt);
    0,1,0;
    sin(thtilt), 0, cos(thtilt)];

Rroll=[1,0,0;
    0 cos(throll), -sin(throll);
    0, sin(throll), cos(throll)];

Rca=Rroll*Rtilt*Rpan;  Rac=Rca';

%Camera frame to image frame and viceversa.
Ric=diag([-1, 1, -1]); Rci=Ric';

%Rotation Map frame to image frame and viceversa.
Rim=Ric*Rca*Raw*Rwm;   Rmi=Rim';

%% Transformation
% Convert aircraft position from meters to pixels
posNEAw=posNEAw/scale; % position of aicraft in pixels
Npoints=size(XYm,2);
Tm=Rmw*posNEAw;
XYm(1,:)=XYm(1,:)-Tm(1);
XYm(2,:)=XYm(2,:)-Tm(2);
XYZi=Rim*[XYm; -Tm(3)*ones(1,Npoints)];
fbar=N(2)/(2*tan(FOV*pi/360));
w=fbar./XYZi(3,:);
XYi(1,:)=XYZi(1,:).*w+N(1)/2;
XYi(2,:)=XYZi(2,:).*w+N(2)/2;




%%//////////////////////////////////////////////////////////////////////
%Roberto Cristi /Jose Espinosa. Naval Postgratuade School ECE
Department.
%March 2016.
%This function converts points on image frame to points on map frame.
%%//////////////////////////////////////////////////////////////////////
function XYm=image2map(N, FOV, thRPYaw, thRTPca, posNEAw, scale)
%XYm=image2map(N, FOV, thRPYaw, thRTPca, posNEAw,scale).
%XYZi = 3xNpoints positions in Image Frame.
%XYm = 2xNpoints positions in Map Frame.
%thRPYaw = 3x1 Roll, Pitch Yaw angles (deg.)World to Aircraft Frames.
%thRTPca = 3x1 Roll, Tilt, Pitch angles (deg. Aircraft to Camera
Frames.
%posNEAw = 3x1 North,East,Altitude of Aircraft in World Frame.
%FOV=Field of View of Camera in degrees.
%N=[number of rows, number of columns] of image. It has to be EVEN.
```

```
%scale = size of pixel in the map, in meters/pixel.
%Map Frame: [Row (down), Column (right), Elevation (up)].
%World Frame: [North, East, Down].
%Aircraft Frame: [Roll (forward), Pitch (right), Yaw (down)].
%Camera Frame: [Roll (forward), Tilt (right), Pan (down)].
%Image Frame: Rows (Down), Columns (Left) , Down (View direction).
%Rotation Matrices (Rba=Rotation from frame “a” to frame “b”).

%Map frame to world frame and viceversa.
Rwm=diag([-1, 1, -1]); Rmw=Rwm’; % Map to World and viceversa

%Aircraft frame to world frame and viceversa.
thR=thRPYaw(1); thP=thRPYaw(2); thY=thRPYaw(3);
thR=thR*pi/180; thP=thP*pi/180; thY=thY*pi/180;

Ry=[cos(thY), sin(thY), 0;
    -sin(thY), cos(thY), 0;
    0,0,1];

Rp=[cos(thP), 0, sin(thP);
    0,1,0;
    -sin(thP), 0, cos(thP)];

Rr=[1,0,0;
    0 cos(thR), sin(thR);
    0, -sin(thR), cos(thR)];

Raw=Rr*Rp*Ry;        Rwa=Raw’;

%Aircraft frame to camera frame and viceversa.
throll=thRTPca(1); thtilt=thRTPca(2); thpan=thRTPca(3);
throll=throll*pi/180; thtilt=thtilt*pi/180; thpan=thpan*pi/180;

Rpan=[cos(thpan), sin(thpan), 0;
    -sin(thpan), cos(thpan), 0;
    0,0,1];

Rtilt=[cos(thtilt), 0, sin(thtilt);
    0,1,0;
    -sin(thtilt), 0, cos(thtilt)];

Rroll=[1,0,0;
    0 cos(throll), sin(throll);
    0, -sin(throll), cos(throll)];

Rca=Rroll*Rtilt*Rpan;  Rac=Rca’;

%Camera frame to image frame and viceversa.
Ric=diag([-1, 1, -1]); Rci=Ric’;

%Rotation Map frame to image frame and viceversa.
Rim=Ric*Rca*Raw*Rwm;   Rmi=Rim’;
```

```
%% Transformation
% Convert aircraft position from meters to pixels
posNEAw=posNEAw/scale;   % position of aicraft in pixels
L1=round(N(1)/2); N(1)=2*L1; L2=round(N(2)/2); N(2)=2*L2;
Ii=reshape((-L1:L1-1)'*ones(1,N(2)), 1, N(1)*N(2));
Ji=reshape(ones(N(1),1)*(-L2:L2-1), 1, N(1)*N(2));
fbar=N(2)/(2*tan(FOV*pi/360));
XYZi=[Ii; Ji; fbar*ones(1,N(1)*N(2))];
Tm=Rmw*posNEAw;
XYZm=Rmi*XYZi;
w=-Tm(3)./XYZm(3,:);
XYm=XYZm(1:2,:).*[w;w];
XYm(1,:)=XYm(1,:)+Tm(1);
XYm(2,:)=XYm(2,:)+Tm(2);


%%///////////////////////////////////////////////////////////////////////
%Roberto Cristi /Jose Espinosa. Naval Postgratuade School ECE
Department.
%March 2016.
%This function presents de virtual view of the camera.
%%///////////////////////////////////////////////////////////////////////
function XYm=image2map2(N, XYi, FOV, thRPYaw, thRTPca, posNEAw, scale)
%XYm=image2map(N, XYi, FOV, thRPYaw, thRTPca, posNEAw, scale);
%XYi = 2xNpoints positions in Image Frame
%XYm = 2xNpoints positions in Map Frame
%thRPYaw = 3x1 Roll, Pitch Yaw angles(deg.)World to Aircraft Frames
%thRTPca = 3x1 Roll, Tilt, Pitch angles(deg.) Aircraft to Camera Frames
%posNEAw = 3x1 North,East,Altitude of Aircraft in World Frame.
%FOV=Field of View of Camera in degrees.
%N=[number of rows, number of columns] of image. It has to be EVEN.
%scale = length, in meters, of one pixel in the map (meters/pixel).
%Map Frame: [Row (down), Column (right), Elevation (up)].
%World Frame: [North, East, Down].
%Aircraft Frame: [Roll (forward), Pitch (right), Yaw (down)].
%Camera Frame: [Roll (forward), Tilt (right), Pan (down)].
%Image Frame: Rows (Down), Columns (Left) , Down (View direction).
%Rotation Matrices (Rba=Rotation from frame “a” to frame “b”).

%Map frame to world frame and viceversa.
Rwm=diag([-1, 1, -1]); Rmw=Rwm’;

%Aircraft frame to world frame and viceversa.
thR=thRPYaw(1); thP=thRPYaw(2); thY=thRPYaw(3);
thR=thR*pi/180; thP=thP*pi/180; thY=thY*pi/180;

Ry=[cos(thY), sin(thY), 0;
    -sin(thY), cos(thY), 0;
    0,0,1];

Rp=[cos(thP), 0, sin(thP);
    0,1,0;
    -sin(thP), 0, cos(thP)];
```

```
Rr=[1,0,0;
    0 cos(thR), -sin(thR);
    0, sin(thR), cos(thR)];

Raw=Rr*Rp*Ry;        Rwa=Raw';

%Aircraft frame to camera frame and viceversa.
throll=thRTPca(1); thtilt=thRTPca(2); thpan=thRTPca(3);
throll=throll*pi/180; thtilt=thtilt*pi/180; thpan=thpan*pi/180;

Rpan=[cos(thpan), sin(thpan), 0;
    -sin(thpan), cos(thpan), 0;
    0,0,1];

Rtilt=[cos(thtilt), 0, sin(thtilt);
    0,1,0;
    -sin(thtilt), 0, cos(thtilt)];

Rroll=[1,0,0;
    0 cos(throll), -sin(throll);
    0, sin(throll), cos(throll)];

Rca=Rroll*Rtilt*Rpan;  Rac=Rca';

%Camera frame to image frame and viceversa.
Ric=diag([-1, 1, -1]); Rci=Ric';

%Rotation Map frame to image frame and viceversa.
Rim=Ric*Rca*Raw*Rwm;   Rmi=Rim';

%% Transformation
% Convert aircraft position from meters to pixels
posNEAw=posNEAw/scale;   % position of aicraft in pixels
Npoints=size(XYi,2);
fbar=N(2)/(2*tan(FOV*pi/360));
XYi(1,:)=XYi(1,:)-N(1)/2;
XYi(2,:)=XYi(2,:)-N(2)/2;
XYZi=[XYi; fbar*ones(1,Npoints)];
Tm=Rmw*posNEAw;
XYZm=Rmi*XYZi;
w=-Tm(3)./XYZm(3,:);
XYm=XYZm(1:2,:).*[w;w];
XYm(1,:)=XYm(1,:)+Tm(1);
XYm(2,:)=XYm(2,:)+Tm(2
```

# LIST OF REFERENCES

[1] Ö. Aytekin, U. Zöngür, and U. Halici, “Texture-based airport runway detection,” *IEEE Geosci.Remote Sensing Lett.*, vol. 10, no. 3, pp. 471–475, May 2013.

[2] Robert E. Schapire, “A brief introduction to boosting,” in *Proceedings 16th International Joint Conference on Artificial Intelligence*, 1999, vol. 2, pp. 1401–1406.

[3] A. Miller, M. Shah, and D. Harper, “Landing a UAV on a runway using image registration,” in *Proceedings International Conference on Robotics and Automation*, Pasadena, CA, 2008, pp. 182–187.

[4] J. Shang, and Z. Shi, “Vision-based runway recognition for UAV autonomous landing,” *Int. Journal of Computer Sci. and Network Security*, vol. 7, no. 3, pp. 112–117, Mar. 2007.

[5] R. Hartley and A. Zisserman. (2004). *Multiple View Geometry in Computer Vision*. [Online]. Available: https://itun.es/us/_pvzW.l

[6] O. J. Woodman, “An introduction to inertial navigation,” Cambridge Univ. Comp. Lab., Cambridge, England, Tech. Report. UCAM-CL-TR-696, Aug. 2007.

[7] “*Salinas Municipal Airport*.” 2016. [Online]. Available: https://www.google.com/maps/@36.6612306,121.6066075,2805m/data=!3m1!1e3. Accessed Jan. 22, 2016.

[8] *Understanding Euler Angles*. [Online]. Available: http://www.chrobotics.com/library/understanding-euler-angles. Accessed Jan. 22, 2016.

[9] *Cloud Cap Technology TASE* 200 *camera*. [Online]. Available: http://www.cloudcaptech.com/products/detail/tase-200. Accessed Jan. 25, 2016.

[10] D. H. Ballard, C. M. Brown, *Computer Vision*, 1st. ed. Englewood Cliffs, NJ: Prentice Hall, 1982, pp. 123–131.

[11] “*Monterey Regional Airport*.” 2016. [Online]. Available: https://www.google.com/maps/@36.5857303,121.8427775,2790m/data=!3m1!1e3. Accessed Jan. 22, 2016.

THIS PAGE INTENTIONALLY LEFT BLANK

## INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California