2012-08-08

Using Motion Fields to Estimate Video Utility and Detect GPS Spoofing

Brandon T. Carroll
Brigham Young University - Provo

Using Motion Fields to Estimate Video Utility and Detect GPS Spoofing

Brandon Carroll

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Randal W. Beard, Chair
Bryan S. Morse
Dah Jye Lee

Department of Electrical and Computer Engineering

Brigham Young University

December 2012

ABSTRACT

Using Motion Fields to Estimate Video Utility and Detect GPS Spoofing

Brandon Carroll
Department of Electrical and Computer Engineering
Master of Science

This work explores two areas of research. The first is the development of a video utility metric for use in aerial surveillance and reconnaissance tasks. To our knowledge, metrics that compute how useful aerial video is to a human in the context of performing tasks like detection, recognition, or identification (DRI) do not exist. However, the Targeting Task Performance (TTP) metric was previously developed to estimate the usefulness of still images for DRI tasks. We modify and extend the TTP metric to create a similar metric for video, called Video Targeting Task Performance (VTTP). The VTTP metric accounts for various things like the amount of lighting, motion blur, human vision, and the size of an object in the image. VTTP can also be predictively calculated to estimate the utility that a proposed flight path will yield. This allows it to be used to help automate path planning so that operators are able to devote more of their attention to DRI. We have used the metric to plan and fly actual paths. We also carried out a small user study that verified that VTTP correlates with subjective human assessment of video.

The second area of research explores a new method of detecting GPS spoofing on an unmanned aerial system (UAS) equipped with a camera and a terrain elevation map. Spoofing allows an attacker to remotely tamper with the position, time, and velocity readings output by a GPS receiver. This tampering can throw off the UAS's state estimates, but the optical flow through the camera still depends on the actual movement of the UAS. We develop a method of detecting spoofing by calculating the expected optical flow based on the state estimates and comparing it against the actual optical flow. If the UAS is successfully spoofed to a different location, then the detector can also be triggered by differences in the terrain between where the UAS actually is and where it thinks it is. We tested the spoofing detector in simulation, and found that it works well in some scenarios.

Keywords: motion field, optical flow, video utility, surveillance, reconnaissance, GPS spoofing, fault detection

ACKNOWLEDGMENTS

Many people contributed to the work presented in this thesis. My adviser, Randy Beard, provided guidance and encouragement all along the way.

Bryan Morse, Peter Niedfeldt, Joel Howard, Dallin Briggs, and Stephen Pledgie were heavily involved in the work related to the VTTP metric. Bryan Morse provided general guidance and helped develop the use of the Lambertian lighting model as a way to estimate potential contrast. Peter Niedfeldt developed a planning algorithm based on VTTP, directed the flight tests, and organized the user study. Joel Howard helped compute lighting maps, investigated the possibility of updating the lighting maps, and helped extract video clips for use in the user study. Dallin Briggs served as a pilot and maintained the hardware. Stephen Pledgie provided expertise on the TTP metric as well as code to compute it.

Tim McLain and Brandon Cannon provided help with the work on GPS spoofing detection. The flight simulator used to test the GPS spoofing detector was based on Randy Beard and Tim McLain's work in *Small Unmanned Aircraft: Theory and Practice* [1]. Brandon Cannon pointed me toward papers that explained the innovations approach to fault detection and helped me work through ideas.

Finally, my parents Quinn and Margaret Carroll have given me the love, support, and inspiration I have needed to pursue higher education. I offer my gratitude to them and to all the others listed here. I could not have done this without their help.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The steadily increasing capabilities and sophistication of modern unmanned aerial systems (UASs) make them an appealing solution to many problems. The recently passed FAA Modernization and Reform Act of 2012, which requires the development of "a comprehensive plan to safely accelerate the integration of civil unmanned aircraft systems into the national airspace system" [2], underscores the importance of UAS research in the United States.

This thesis describes research done in two areas involving UASs. The first area is assessing the utility of aerial video for tasks such as surveillance and reconnaissance. The second area is using a video camera to detect GPS spoofing attacks on a UAS. Descriptions of these two areas and statements on the contributions of this work are given below.

1.1 The Video Targeting Task Performance Metric

1.1.1 Motivation

The use of UASs can provide an expanded capability to strategically position sensors in various tasks. Many of these tasks can be dull or dangerous. During the Fukushima disaster in Japan, a team of pilots remotely flew a Honeywell T-Hawk UAS through the facility several times to collect information on the conditions in areas where humans could not go [3]. The US military also makes extensive use of drones in surveillance and reconnaissance missions, thus reducing the workload of pilots and protecting their lives from unnecessary risk [4].

Although UASs do not carry passengers, they are rarely fully autonomous. Humans are typically needed to guide the UAS and inspect the data it collects. Much research on UASs is focused toward increasing the autonomy of the UAS and decreasing the workload of its human operators [5].

We specifically consider surveillance and reconnaissance scenarios. In these scenarios, the central objective is to perform detection, recognition, or identification (DRI) tasks. When using UASs, a human operator typically performs the DRI task by inspecting the video stream coming from the UAS. Meanwhile, the UAS needs to fly a path that gives good video coverage of the sensitive area, perhaps prioritizing some areas above others. If an operator needs to manage the flight path of one or more UASs, then his attention will be distracted from performing the DRI task.

In uncontrolled environments, humans are much better at vision tasks like DRI than computers. Thus, it is best to free any human operators from managing the flight paths of the UASs so that they can focus exclusively on the video stream. Many different types of path planning algorithms exist that could be used to automatically plan the paths of the UASs. The ideal choice would be a planning algorithm that maximizes the probability that a human watching the video will succeed at the DRI task. However, there must be a way to estimate this probability before a planning algorithm can attempt to maximize it.

Metrics that attempt to estimate the probability of DRI for aerial video do not exist. Morse el al. proposed a see-ability calculation for aerial video that could potentially be used in such a planning algorithm [6]. However, they were not explicitly attempting to model the probability of DRI and their work was geared toward creating coverage maps, not planning paths.

Without a metric to estimate the probability of DRI, planning algorithms used in surveillance and reconnaissance are ignorant of the effects that their planned paths have on the success of the DRI task. Development of such a metric has the potential to increase both the autonomy and the performance of UASs used in surveillance and reconnaissance. The metric would also have potential applications in many other tasks that use video captured by a UAS.

1.1.2 Contributions

This work contributes to the state of the art by developing a video utility metric that estimates the probability of DRI for aerial video. Because the metric is based on the well-established Targeting Task Performance (TTP) metric for still images [7], we call it the

Video Targeting Task Performance (VTTP) metric. VTTP allows automated path planning algorithms to find paths that maximize the chances of success in surveillance, reconnaissance, and other similar tasks. These paths can automatically balance video quality versus coverage and account for different priority levels in different areas. Automating path planning also relieves the operators from having to manage the UASs, allowing them to focus on the video. During or after flight, VTTP can be used to evaluate the actual performance achieved so that future plans can be adjusted accordingly.

The development of the VTTP metric was a collaborative effort. My specific contributions include the development of the motion field and occlusion detection algorithms. Peter Niedfeldt developed the path planning algorithm and the observer viewing model. Bryan Morse and Joel Howard contributed the lighting model and investigated the possibility of updating the lighting map. Stephen Pledgie provided expertise on TTP as well as code to calculate it. We were all involved in the flight tests and the user study used to validate the metric.

1.2 GPS Spoofing

1.2.1 Motivation

The Global Positioning System (GPS) is used in countless military and civilian applications. Although GPS includes a secured military band, the vast majority of its users are restricted to using the unsecured civilian band. Since all the information about the civilian band signal is available publicly, it is possible for a malicious individual to construct and broadcast fake civilian band signals that would mislead GPS receivers. This type of attack is known as spoofing. Although broadcasting fake GPS signals is illegal, the equipment necessary to do it is readily available at relatively low cost [8].

Few GPS spoofing attacks have been reported to date, but their potential to be lucrative and damaging increases as society becomes more dependent on GPS. The Volpe report prepared for the Department of Transportation in 2001 states that even fairly small errors in GPS could cause collisions between ships navigating narrow channels or run a barge into a highway bridge during inclement weather. It also notes an instance where a military helicopter convoy was led off course by faulty GPS readings despite obvious visual cues. The

report repeatedly highlights the need to investigate anti-spoofing technologies and inform GPS users of the risks [9].

For the past few years, Todd E. Humphreys at the University of Texas at Austin has also been publishing warnings about the vulnerability of the GPS system to spoofing. He has shown that spoofing attacks could be used to damage the power grid [10], interfere with cell phone service [11], or game the stock market [12]. More recently, Humphreys' Radionavigation Lab and Adaptive Flight, Inc. successfully demonstrated a GPS spoofing attack that caused an Adaptive Flight Hornet Mini UAV to mistakenly lower its altitude because it thought it was higher than it really was [13, 14].

Several people have proposed countermeasures against GPS spoofing [15, 16], but it has been shown that virtually all of them are vulnerable to sophisticated spoofing attacks [17]. Although many of the countermeasures are simple and would increase the difficulty of successfully spoofing without triggering alarms, none of them have been implemented in typical GPS receivers.

Ultimately, the best defense against spoofing would require modification of the entire GPS system to include cryptographic authentication. Unfortunately, it would probably take many years for such upgrades to be completed. Meanwhile, the only defense is to harden GPS receivers and the systems they depend on against spoofing.

1.2.2 Contributions

This work contributes to the state of the art by developing an algorithm that detects GPS spoofing by comparing expected optical flow against actual optical flow. The expected optical flow is calculated using the estimated states of the UAS and a terrain elevation map. Spoofing that alters the state estimates of the aircraft can trigger the detector by throwing off the expected optic flow. The detector is also triggered if there are large enough differences in the terrain between the actual location of the UAS and where it thinks it is. This provides a measure of defense against even the most sophisticated types of spoofing attacks.

1.3 Overview

Chapter 2 gives a brief introduction to some of the notational conventions used throughout the thesis, defines the coordinate frames of a UAS, and defines the state variables that describe the where the UAS is and how it is moving. Chapter 3 derives the mathematical equations used to calculate a motion field from terrain data and the states of a UAS. The motion field calculations are utilized in both the work on the video utility metric and the work on detecting GPS spoofing. Chapter 4 describes the VTTP aerial video utility metric and how it can be used in surveillance and reconnaissance applications. Chapter 5 presents our work on the algorithm to detect GPS spoofing on a UAS by comparing expected motion field calculations against the optical flow observed by the camera. Chapter 6 reviews the contributions of this work and lists some potential future work.

Chapter 2

Notation and Definitions

2.1 Introduction

This chapter introduces notational conventions, coordinate frame definitions, and state variable definitions that are used throughout the rest of this work. The coordinate frame definitions are useful in expressing the locations of objects from different points of view. The state variables describe where a UAS is located, how it is oriented, and how it is moving.

2.2 Notation

The following list describes some of the notation used in this work:

$\dot{x}$	Dots over variable names indicate the time derivative of that variable.
c_θ	Shorthand notation for $\cos(\theta)$.
s_θ	Shorthand notation for $\sin(\theta)$.
$\mathbf{x}$	Bold, lowercase letters represent column vectors.
$\mathbf{R}$	Bold, uppercase letters represent matrices.

2.3 Coordinate Frames

We use the coordinate frame definitions given in *Small Unmanned Aircraft: Theory and Practice* [1]. These coordinate frames follow the standard practice in aviation of using north east down (NED) coordinates. Units of length are measured in meters and angles are expressed in radians.

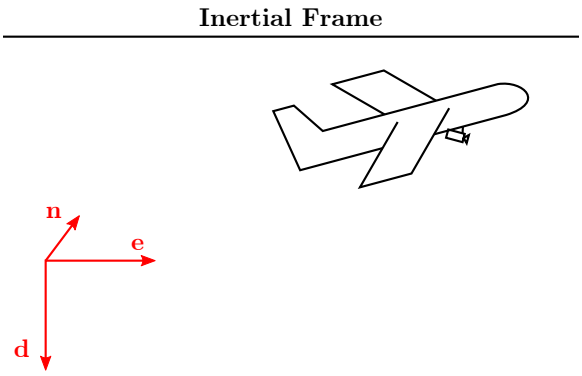

Figure 2.1: The inertial coordinate system. The red coordinate axes in the lower left are located at the south west corner of the current UTM zone and at the height from which the UAS was launched. Coordinates are given in meters to the north, east, and down from this origin. The terrain data and the absolute position of the UAS are given in inertial frame coordinates.

2.3.1 The Inertial Frame

The inertial coordinate frame provides an absolute reference for where things are located. For this work, we use the Universal Transverse Mercator (UTM) coordinate system as the inertial frame. The specifications of the UTM coordinate system are given in *The Universal Grids: Universal Transverse Mercator (UTM) and Universal Polar Stereographics (UPS)* [18]. UTM divides the surface area of the globe into a grid of zones, each designated with a letter and a number. Within each zone, coordinates are given in meters to the north and to the east of the south west corner of the zone. This corner serves as the origin of the inertial coordinate system.

Since UTM only specifies 2D coordinates, we need to define a reference from which to measure altitude. When we launch a UAS, we first zero the static pressure sensor that measures altitude. Thus, the UAS reports altitudes measured relative to the launch point. Out of convenience, we set the altitude of the inertial coordinate system to match the altitude at the launch point. Figure 2.1 illustrates the inertial coordinate frame.

2.3.2 The Vehicle Frame

The vehicle coordinate frame represents where things are located to the north, to the east, or below the current position of the UAS. Its coordinate axes are parallel to those of the inertial frame, but the origin is translated to the center of mass of the UAS. Figure 2.2 shows

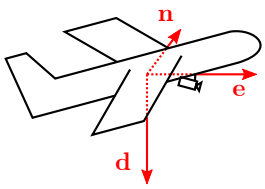

Figure 2.2: The vehicle coordinate frame. The origin is located at the center of mass of the UAS. The axes are oriented to point directly north, east, and down regardless of how the UAS is rotated.

how the origin of the vehicle frame is centered on the UAS, but the axes are still aligned in the north, east, and down directions.

2.3.3 The Body Frame

The body frame represents where things are located relative to both the position and the orientation of the UAS. It defines how things would be seen from the perspective of a pilot in the aircraft. Like the vehicle frame, the origin of the body frame is located at the center of mass of the UAS. However, the axes are rotated such that the x axis points out the nose of the aircraft, the y axis points out the right wing of the aircraft, and the z axis points out the bottom of the aircraft. The roll, pitch, and yaw angles define the rotations necessary to convert between the vehicle and the body frames. The body coordinate frame is depicted in Figure 2.3.

2.3.4 The Gimbal Frame

The gimbal frame expresses the locations of objects from the point of view of the camera. We assume that the camera's optical center is located at the center of mass of the UAS. Although this is probably not the case, the differences caused by the camera's offset are usually negligible. As long as the offset is small compared to the distances to the objects that the camera is looking at, this assumption is good.

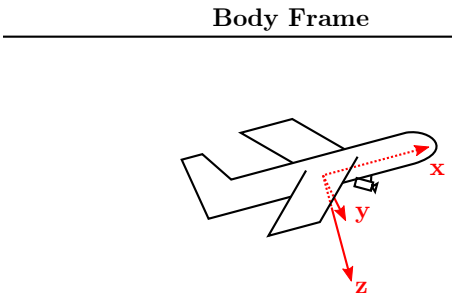

Figure 2.3: The body coordinate frame. The origin of the coordinate frame is located at the center of mass of the UAS. The x, y, and z axes are oriented to point out the nose of the aircraft, out the right wing of the aircraft, and out the bottom of the aircraft, respectively. This coordinate frame represents where things would be located from the perspective of a pilot.

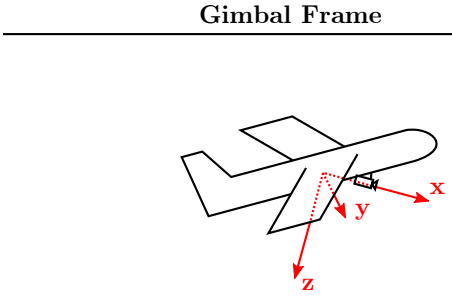

Figure 2.4: The gimbal coordinate frame. The origin is located at the center of mass of the UAS. Although the optical center of the camera may not actually be located at the center of mass of the UAS, we make the simplifying assumption that it is. The x, y, and z axes point out the front of the camera, out the right side of the camera, and out the bottom of the camera, respectively. This coordinate frame represents locations relative to the camera's point of view.

Given the above assumption, we place the origin of the gimbal frame at the center of mass of the aircraft. The axes are oriented such that the x axis coincides with the optical axis of the camera, the y axis points out the right side of the camera, and the z axis points out the bottom of the camera. The azimuth and elevation angles of the camera define the rotations necessary to convert between the body frame and the gimbal frame. Figure 2.4 depicts the gimbal frame.

2.3.5 The Image Frame

The image frame expresses the two-dimensional pixel location of an object in the image plane. The origin is the upper left pixel in the camera image. The x axis points to the right and the y axis points down.

2.4 UAS States

The state variables of a UAS describe its location, its orientation, and how it is moving. We denote the UAS state vector as $\mathbf{x}$. It is defined as $\mathbf{x} = (n, e, d, u, v, w, \phi, \theta, \psi, p, q, r)^\top$, where $(n, e, d)^\top$ is the inertial position of the UAS in north east down coordinates, $(u, v, w)^\top$ is the velocity vector expressed in body frame coordinates, ϕ is the roll angle, θ is the pitch, ψ is the yaw, and $(p, q, r)^\top$ are the angular rates expressed in body frame coordinates. An extended Kalman filter (EKF) and various low-pass filters running on board the UAS estimate $\mathbf{x}$ based on noisy sensor measurements.

If the roll, pitch, and yaw angles were all zeroed, then the aircraft would be level and pointed directly north. The yaw angle increases as the nose of the aircraft is turned to the right. The pitch angle increases as the nose of the aircraft tips upward. The roll angle increases as the right wing tip lowers and the left raises. These all constitute right-handed rotations around an axis in each successive coordinate frame.

If the UAS is equipped with a gimballed camera, then the azimuth, α_{az}, and elevation, α_{el}, angles as well as their angular rates become part of the state. Zeroed azimuth and elevation angles indicate that the camera is pointing directly out the nose of the aircraft. The azimuth angle increases as the camera pans to the right relative to the body of the aircraft. The elevation angle decreases as the camera tips downward. The azimuth and elevation rotations are also right-handed. If the camera has zooming capabilities, then the focal length and rate of change of the focal length are also added to the state.

Chapter 3

Motion Field

3.1 Introduction

A motion field is an ideal, geometric calculation of the motion of three-dimensional points as projected into the two-dimensional image plane of a camera. Figure 3.1 shows an example motion field calculated for an aircraft flying over slightly hilly terrain. Motion field calculations are an integral part of the video utility metric presented in Chapter 4 as well as the GPS spoofing detection system presented in Chapter 5. This chapter derives the equations necessary to calculate a motion field from terrain elevation data and the states of a camera-equipped UAS.

3.2 Elevation Maps

The first step in calculating a motion field is choosing the 3D points for which to calculate the motion. These points are somewhat analogous to the feature points chosen by an optical flow algorithm. In this work, motion fields are always calculated for terrain elevation data points viewed from the perspective of a UAS flying overhead.

All of the actual terrain elevation data and imagery used in this work was downloaded from the US Geological Survey's Seamless Data Warehouse. The elevation data came from the National Elevation Dataset and the orthoimagery came from the National Agriculture Imagery Program. The elevation data consists of elevation values sampled over a regular grid of latitude and longitude coordinates. We convert these coordinates into the UTM coordinate system, which we use as the inertial coordinate frame as described in Section 2.3.1.

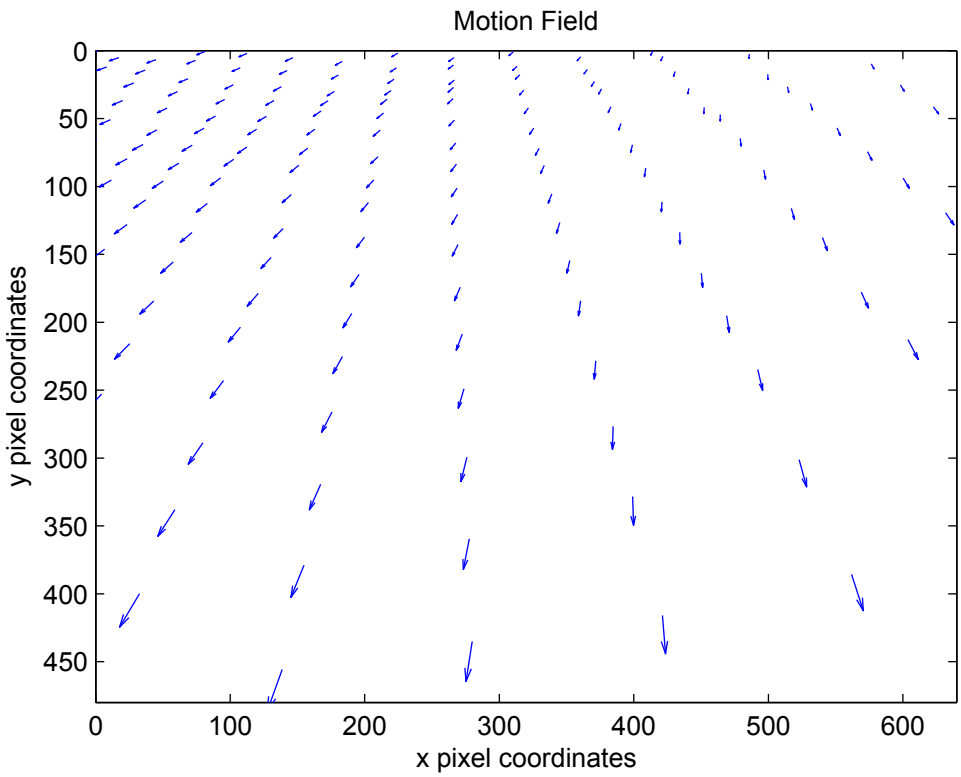

Figure 3.1: A example of a motion field calculated for an aircraft flying over slightly hilly terrain while gradually turning to the right. The camera is pointed downward and forward relative to the body of the aircraft. The base of each vector represents where a terrain point from the elevation map projected into the camera frame. The tip of each vector points to where that point will move in one camera frame's time given the current motion of the aircraft.

3.3 Derivation of the Motion Field Equations

Our derivation of the motion field equations is based on the equation used in computer graphics to render three-dimensional points on a two-dimensional screen. This equation uses matrices to transform points into different coordinate frames. However, these transformations require both translation and rotation. Matrices of the same dimensionality as the points they operate on can rotate and skew the points, but cannot translate them.

To facilitate translation of the points via matrix multiplications, computers append a fourth, homogeneous coordinate to the points, thus taking them from $\mathbb{R}^3$ into the projective space $\mathbb{P}^3$. The addition of the homogeneous coordinate allows matrices in $\mathbb{R}^{4\times4}$ to accomplish translations by skewing the first three coordinates of the points with respect to the fourth,

homogeneous dimension. To make these translations have a 1:1 scaling, the value of the homogeneous coordinate is always set to 1.

The computer graphics equation takes as parameters the inertial frame coordinates of a point of interest and the position and orientation of the camera. It outputs the pixel coordinates that the point projects to in the image frame. However, we need both the location and velocity of each point in the image frame to construct a motion field. To determine the velocity of each point in the image frame, we differentiate the computer graphics equation with respect to time.

Since the computer graphics equation involves a product of many matrices that depend on time, its derivative with respect to time would be lengthy if written as a single equation. To simplify the notation, we first construct matrices that represent the transformations between the coordinate frames defined in Section 2.3. These matrices and their time derivatives are presented in the order in which they are applied to the points being projected. Then, the final motion field equations are given in terms of these transformation matrices and their derivatives.

3.3.1 Translation from the Inertial Frame to the Vehicle Frame

The first step in projecting three-dimensional terrain points into the image plane is calculating the locations of the points in the UAS vehicle frame. The vehicle frame's axes are aligned with the inertial NED coordinate system's axes, but its origin is located at the center of mass of the aircraft. Thus, the transformation from the inertial to the vehicle frame is a simple translation given by the matrix

$$\mathbf{T}_{\mathrm{i}}^{\mathrm{v}} = \begin{bmatrix} 1 & 0 & 0 & -n \\ 0 & 1 & 0 & -e \\ 0 & 0 & 1 & -d \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.1}$$

where $(n, e, d)^\top$ are the inertial coordinates of the center of mass of the aircraft. Since the inertial position of the aircraft changes over time, the derivative of Equation (3.1) must also

be computed. It is given by

$$\dot{\mathbf{T}}_{\mathrm{i}}^{\mathrm{v}} = \begin{bmatrix} 0 & 0 & 0 & -\dot{n} \\ 0 & 0 & 0 & -\dot{e} \\ 0 & 0 & 0 & -\dot{d} \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.2}$$

where $(\dot{n}, \dot{e}, \dot{d})^\top$ is the velocity of the UAS in the inertial frame.

Since the EKF estimates the velocity of the UAS in body frame coordinates, $(\dot{n}, \dot{e}, \dot{d})^\top$ must be obtained by rotating the estimated velocity back into the inertial frame. As given by [1], the equation to do this is

$$\begin{bmatrix} \dot{n} \\ \dot{e} \\ \dot{d} \end{bmatrix} = \begin{bmatrix} \mathrm{c}_\theta \mathrm{c}_\psi & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{c}_\psi - \mathrm{c}_\phi \mathrm{s}_\psi & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{c}_\psi + \mathrm{s}_\phi \mathrm{s}_\psi \\ \mathrm{c}_\theta \mathrm{s}_\psi & \mathrm{s}_\phi \mathrm{s}_\theta \mathrm{s}_\psi + \mathrm{c}_\phi \mathrm{c}_\psi & \mathrm{c}_\phi \mathrm{s}_\theta \mathrm{s}_\psi - \mathrm{s}_\phi \mathrm{c}_\psi \\ -\mathrm{s}_\theta & \mathrm{s}_\phi \mathrm{c}_\theta & \mathrm{c}_\phi \mathrm{c}_\theta \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \tag{3.3}$$

where $(u, v, w)^\top$ is the estimated body frame velocity.

3.3.2 Rotation from the Vehicle Frame to the Body Frame

After the terrain points have been translated to the vehicle frame, they must be rotated into the body frame of the UAS. This is accomplished by rotating the points by the yaw angle of the aircraft, then by the pitch angle, and finally by the roll angle. Since we are converting between coordinate frames instead of rotating points within a coordinate frame, we must use left-handed rotation matrices. The yaw, pitch, and roll rotation matrices are defined as

$$\mathbf{R}_\psi = \begin{bmatrix} \mathrm{c}_\psi & \mathrm{s}_\psi & 0 & 0 \\ -\mathrm{s}_\psi & \mathrm{c}_\psi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.4}$$

$$\mathbf{R}_\theta = \begin{bmatrix} \mathrm{c}_\theta & 0 & -\mathrm{s}_\theta & 0 \\ 0 & 1 & 0 & 0 \\ \mathrm{s}_\theta & 0 & \mathrm{c}_\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.5}$$

and

$$\mathbf{R}_\phi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \mathrm{c}_\phi & \mathrm{s}_\phi & 0 \\ 0 & -\mathrm{s}_\phi & \mathrm{c}_\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.6}$$

We combine the three rotations into a single, vehicle-to-body rotation matrix,

$$\mathbf{R}_{\mathrm{v}}^{\mathrm{b}} = \mathbf{R}_\phi \mathbf{R}_\theta \mathbf{R}_\psi. \tag{3.7}$$

Because the orientation of the UAS changes during flight, the time derivatives of the rotation matrices are also needed. They are given by

$$\dot{\mathbf{R}}_\psi = \begin{bmatrix} -\dot{\psi}\mathrm{s}_\psi & \dot{\psi}\mathrm{c}_\psi & 0 & 0 \\ -\dot{\psi}\mathrm{c}_\psi & -\dot{\psi}\mathrm{s}_\psi & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.8}$$

$$\dot{\mathbf{R}}_\theta = \begin{bmatrix} -\dot{\theta}\mathrm{s}_\theta & 0 & -\dot{\theta}\mathrm{c}_\theta & 0 \\ 0 & 0 & 0 & 0 \\ \dot{\theta}\mathrm{c}_\theta & 0 & -\dot{\theta}\mathrm{s}_\theta & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.9}$$

and

$$\dot{\mathbf{R}}_\phi = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & -\dot{\phi}\mathrm{s}_\phi & \dot{\phi}\mathrm{c}_\phi & 0 \\ 0 & -\dot{\phi}\mathrm{c}_\phi & -\dot{\phi}\mathrm{s}_\phi & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.10}$$

where $\dot{\psi}$, $\dot{\theta}$, and $\dot{\phi}$ are the angular rates of the UAS each expressed in its own successive coordinate frame. Applying the product rule to take the derivative of Equation (3.7) gives

$$\dot{\mathbf{R}}_{\mathrm{v}}^{\mathrm{b}} = \dot{\mathbf{R}}_{\phi}\mathbf{R}_{\theta}\mathbf{R}_{\psi} + \mathbf{R}_{\phi}\dot{\mathbf{R}}_{\theta}\mathbf{R}_{\psi} + \mathbf{R}_{\phi}\mathbf{R}_{\theta}\dot{\mathbf{R}}_{\psi}. \tag{3.11}$$

The angular rates p, q, and r estimated by the EKF are all expressed in the body frame of the aircraft, and thus must be transformed to obtain $\dot{\psi}$, $\dot{\theta}$, and $\dot{\phi}$. The necessary transformation is given by

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \tag{3.12}$$

where p, q, and r are the body-frame angular rates [1].

3.3.3 Rotation from the Body Frame to the Gimbal Frame

At this point, we make the simplifying assumption that the optical center of the camera coincides with the center of mass of the UAS. In reality, it is not likely that the center of mass and the optical center will be perfectly aligned. However, the distance between them is often small enough that it has a negligible effect on the motion field calculations, particularly for small UAS. Even for a large aircraft with a camera mounted far out on a wing, the difference would be small unless the aircraft was flying particularly close to the terrain. Without this assumption, another translation matrix representing the body frame difference in location would need to be inserted into all the equations between the $\mathbf{R}_{\mathrm{v}}^{\mathrm{b}}$ and $\mathbf{R}_{\mathrm{b}}^{\mathrm{g}}$ matrices or their derivatives.

Once the terrain points have been expressed in the body frame, they must then be rotated by the azimuth and elevation angles at which the camera is oriented relative to the body of the UAS. For generality, we derive the equations under the assumption that the UAS is equipped with a gimballed camera. If a fixed camera is used, the azimuth and elevation can be set according to how the camera is mounted on the aircraft and their time derivatives

can be set to zero. The azimuth and elevation rotation matrices are

$$\mathbf{R}_{\text{az}} = \begin{bmatrix} \text{c}_{\alpha_{\text{az}}} & \text{s}_{\alpha_{\text{az}}} & 0 & 0 \\ -\text{s}_{\alpha_{\text{az}}} & \text{c}_{\alpha_{\text{az}}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.13}$$

and

$$\mathbf{R}_{\text{el}} = \begin{bmatrix} \text{c}_{\alpha_{\text{el}}} & 0 & \text{s}_{\alpha_{\text{el}}} & 0 \\ 0 & 1 & 0 & 0 \\ -\text{s}_{\alpha_{\text{el}}} & 0 & \text{c}_{\alpha_{\text{el}}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.14}$$

We combine the azimuth and elevation rotation matrices into a single, body-to-gimbal rotation matrix,

$$\mathbf{R}_{\text{b}}^{\text{g}} = \mathbf{R}_{\text{el}}\mathbf{R}_{\text{az}}. \tag{3.15}$$

If the camera is gimballed, the time derivatives of these rotations are also needed. They are given by

$$\dot{\mathbf{R}}_{\text{az}} = \begin{bmatrix} -\dot{\alpha}_{\text{az}}\text{s}_{\alpha_{\text{az}}} & \dot{\alpha}_{\text{az}}\text{c}_{\alpha_{\text{az}}} & 0 & 0 \\ -\dot{\alpha}_{\text{az}}\text{c}_{\alpha_{\text{az}}} & -\dot{\alpha}_{\text{az}}\text{s}_{\alpha_{\text{az}}} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.16}$$

and

$$\dot{\mathbf{R}}_{\text{el}} = \begin{bmatrix} -\dot{\alpha}_{\text{el}}\text{s}_{\alpha_{\text{el}}} & 0 & \dot{\alpha}_{\text{el}}\text{c}_{\alpha_{\text{el}}} & 0 \\ 0 & 0 & 0 & 0 \\ -\dot{\alpha}_{\text{el}}\text{c}_{\alpha_{\text{el}}} & 0 & -\dot{\alpha}_{\text{el}}\text{s}_{\alpha_{\text{el}}} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{3.17}$$

where $\dot{\alpha}_{\text{az}}$ and $\dot{\alpha}_{\text{el}}$ are the rates of change of the azimuth and elevation angles. The derivative of $\mathbf{R}_{\text{b}}^{\text{g}}$ is obtained by applying the product rule to Equation (3.15), which yields

$$\dot{\mathbf{R}}_{\text{b}}^{\text{g}} = \dot{\mathbf{R}}_{\text{el}}\mathbf{R}_{\text{az}} + \mathbf{R}_{\text{el}}\dot{\mathbf{R}}_{\text{az}}. \tag{3.18}$$

3.3.4 Projecting into the Image Plane

To project the terrain points from the gimbal frame into the image plane, we multiply them by a projection matrix and a camera matrix. Usually, the projection matrix simply takes the points from the projective space $\mathbb{P}^3$ to the projective space $\mathbb{P}^2$ by discarding the fourth coordinate. However, in this case it is necessary to reorder the axes as well so that they conform to conventions for image coordinates. The first image plane axis points to the right and corresponds to the second gimbal frame axis, which points out the right side of the camera. The second image plane axis points down and corresponds to the third gimbal frame axis, which points out the bottom of the camera. The third dimension becomes the new homogeneous coordinate and corresponds to the first axis in the gimbal frame, or the depth of the points relative to the camera. The projection matrix that reorders the axes appropriately is

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{3.19}$$

We use a standard camera matrix, which assumes the pinhole camera model. This matrix is

$$\mathbf{C} = \begin{bmatrix} \frac{f}{d_\mathrm{x}} & s_\theta & o_\mathrm{x} \\ 0 & \frac{f}{d_\mathrm{y}} & o_\mathrm{y} \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.20}$$

where f is the focal length in meters, d_x and d_y are the dimensions of a single pixel on the camera's sensor in meters, s_θ is the camera's skew factor, and $(o_\mathrm{x}, o_\mathrm{y})^\top$ is the pixel coordinate of the optical center of the image. For greater accuracy, the camera can be calibrated using a software library like OpenCV. The calibration data can then be used to undistort the camera's output so that it better matches the pinhole camera model. Together, the projection and camera matrices bring the terrain points into the projective space $\mathbb{P}^2$ that represents their pixel locations in the image plane.

For generality, we assume that the camera on the UAS is capable of zooming in and out. If this is the case, then we also need to take the rate of change of the focal length into

account. We do this by differentiating the camera matrix to get

$$\dot{\mathbf{C}} = \begin{bmatrix} \frac{\dot{f}}{d_{\mathrm{x}}} & 0 & 0 \\ 0 & \frac{\dot{f}}{d_{\mathrm{y}}} & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tag{3.21}$$

where $\dot{f}$ is the rate of change of the focal length in meters per second. If the UAS has a fixed zoom camera, the focal length will be constant and $\dot{\mathbf{C}}$ can be set to zero.

3.3.5 The Motion Field Equations

To compute a motion field, we need to calculate the image frame location of the flow vector for each terrain point as well as the components of the flow in the x and y directions. The location of the flow vector for each terrain point is found directly from the computer graphics equation that projects the three-dimensional point into the two-dimensional image seen by the camera. This equation, adapted specifically for rendering from the point of view of a UAS, is

$$\begin{bmatrix} t_{\mathrm{x}} \\ t_{\mathrm{y}} \\ t_{\mathrm{z}} \end{bmatrix} = \mathbf{C}\mathbf{P}\mathbf{R}_{\mathrm{b}}^{\mathrm{g}}\mathbf{R}_{\mathrm{v}}^{\mathrm{b}}\mathbf{T}_{\mathrm{i}}^{\mathrm{v}} \begin{bmatrix} t_{\mathrm{n}} \\ t_{\mathrm{e}} \\ t_{\mathrm{d}} \\ 1 \end{bmatrix}, \tag{3.22}$$

where $(t_{\mathrm{n}}, t_{\mathrm{e}}, t_{\mathrm{d}})^\top$ is the inertial location of the terrain point of interest, $\mathbf{T}_{\mathrm{i}}^{\mathrm{v}}$ is the translation matrix from the inertial frame to the vehicle frame given in Equation (3.1), $\mathbf{R}_{\mathrm{v}}^{\mathrm{b}}$ is the rotation matrix from the vehicle to the body frame given in Equation (3.7), $\mathbf{R}_{\mathrm{b}}^{\mathrm{g}}$ is the rotation matrix from the body frame to the gimbal frame given in Equation (3.15), $\mathbf{P}$ is the projection matrix given in Equation (3.19), $\mathbf{C}$ is the camera matrix given in Equation (3.20), and $(t_{\mathrm{x}}, t_{\mathrm{y}}, t_{\mathrm{z}})^\top$ is the image plane location of the point of interest expressed in homogeneous coordinates. To convert $(t_{\mathrm{x}}, t_{\mathrm{y}}, t_{\mathrm{z}})^\top$ from homogeneous coordinates in $\mathbb{P}^2$ back into actual coordinates in $\mathbb{R}^2$, we simply need to divide by t_{z} and discard the homogeneous coordinate. Thus, the actual

pixel location of the flow vector for the specified terrain point is

$$\begin{bmatrix} m_{\mathrm{x}} \\ m_{\mathrm{y}} \end{bmatrix} = f_{xy}(\mathbf{x}, \mathbf{t}) \triangleq \frac{1}{t_{\mathrm{z}}} \begin{bmatrix} t_{\mathrm{x}} \\ t_{\mathrm{y}} \end{bmatrix}, \tag{3.23}$$

where $(m_{\mathrm{x}}, m_{\mathrm{y}})^\top$ is the location of the flow vector in the image, $f_{xy}(\cdot, \cdot)$ is the motion field function, $\mathbf{t} \triangleq (t_{\mathrm{n}}, t_{\mathrm{e}}, t_{\mathrm{d}})^\top$ is the inertial location of the terrain point, and $(t_{\mathrm{x}}, t_{\mathrm{y}}, t_{\mathrm{z}})^\top$ are the homogeneous image plane coordinates of the point given by Equation (3.22).

Usually, not all terrain points will fall within the field of view of the camera. To save computation, the points outside of the viewing frustum can be discarded prior to calculating the motion field. Alternatively, Equation (3.23) can be calculated for all the points, and then the ones that fall outside the boundaries of the image or have negative values for t_{z} can be discarded. A negative value for t_{z} indicates that the point is located behind the camera.

Since a terrain point's position in the image frame is given by Equations (3.22) and (3.23), we can find its velocity in the image frame by taking the time derivatives of those equations. We multiply the velocity by the camera's frame period, T, to convert the flow rate from pixels per second to pixels per frame so that it matches the output of typical optical flow algorithms. The time derivative of Equation (3.22) is

$$\begin{bmatrix} \dot{t}_{\mathrm{x}} \\ \dot{t}_{\mathrm{y}} \\ \dot{t}_{\mathrm{z}} \end{bmatrix} = \left(\dot{\mathbf{C}}\mathbf{P}\mathbf{R}_{\mathrm{b}}^{\mathrm{g}}\mathbf{R}_{\mathrm{v}}^{\mathrm{b}}\mathbf{T}_{\mathrm{i}}^{\mathrm{v}} + \mathbf{C}\mathbf{P}\left(\dot{\mathbf{R}}_{\mathrm{b}}^{\mathrm{g}}\mathbf{R}_{\mathrm{v}}^{\mathrm{b}}\mathbf{T}_{\mathrm{i}}^{\mathrm{v}} + \mathbf{R}_{\mathrm{b}}^{\mathrm{g}}\dot{\mathbf{R}}_{\mathrm{v}}^{\mathrm{b}}\mathbf{T}_{\mathrm{i}}^{\mathrm{v}} + \mathbf{R}_{\mathrm{b}}^{\mathrm{g}}\mathbf{R}_{\mathrm{v}}^{\mathrm{b}}\dot{\mathbf{T}}_{\mathrm{i}}^{\mathrm{v}} \right) \right) \begin{bmatrix} t_{\mathrm{n}} \\ t_{\mathrm{e}} \\ t_{\mathrm{d}} \\ 1 \end{bmatrix}, \tag{3.24}$$

where $\mathbf{T}_{\mathrm{i}}^{\mathrm{v}}$, $\dot{\mathbf{T}}_{\mathrm{i}}^{\mathrm{v}}$, $\mathbf{R}_{\mathrm{v}}^{\mathrm{b}}$, $\dot{\mathbf{R}}_{\mathrm{v}}^{\mathrm{b}}$, $\mathbf{R}_{\mathrm{b}}^{\mathrm{g}}$, $\dot{\mathbf{R}}_{\mathrm{b}}^{\mathrm{g}}$, $\mathbf{P}$, $\mathbf{C}$, and $\dot{\mathbf{C}}$ are given in Equations (3.1), (3.2), (3.7), (3.11), (3.15) and (3.18)–(3.21). Using the quotient rule to differentiate Equation (3.23) and multiplying by T, we get

$$\begin{bmatrix} m_{\mathrm{u}} \\ m_{\mathrm{v}} \end{bmatrix} = f_{uv}(\mathbf{x}, \mathbf{t}) \triangleq \frac{T}{t_{\mathrm{z}}^2} \begin{bmatrix} t_{\mathrm{z}}\dot{t}_{\mathrm{x}} + \dot{t}_{\mathrm{z}}t_{\mathrm{x}} \\ t_{\mathrm{z}}\dot{t}_{\mathrm{y}} + \dot{t}_{\mathrm{z}}t_{\mathrm{y}} \end{bmatrix}, \tag{3.25}$$

where $(m_u, m_v)^\top$ is the motion field vector, $f_{uv}(\cdot, \cdot)$ is the motion field function that calculates a flow vector, and $(\dot{t}_x, \dot{t}_y, \dot{t}_z)^\top$ are given by Equation (3.24).

Taken together, the locations of the flow vectors for each point from Equation (3.23) and the flow vectors themselves from Equation (3.25) form the entirety of the motion field. Graphs like Figure 3.1 can be produced simply by drawing each flow vector at it's corresponding location.

3.4 Occlusion Detection

When calculating a motion field, it is often desirable to include only those points that are actually visible to the camera. For example, when viewing a hill from the side, the terrain points on the back side of the hill should be removed from the motion field so that they do not garble the motion field of the points on the front side of the hill. This makes the motion field a better representation of the actual optical flow that would be seen in that situation.

To determine if a point is visible from a given camera position, we examine the places where the line of sight passes over the grid lines of the terrain data. If the line of sight is higher than the terrain at every grid line crossing, then the point is visible. Since the terrain data is discretely sampled along each grid line, we use linear interpolation to determine the elevation of the terrain at the cross-over points. Figure 3.2 illustrates how the algorithm works for two sample lines of sight.

If the point under consideration is not itself a terrain point, then a bilinear interpolation of the terrain at the north/east coordinate of the point should also be performed to determine if the point itself is underground. This covers the case where the aircraft is directly above the point, resulting in a vertical line of sight that does not intersect with any north/east grid lines.

It should also be noted that the grid lines of actual elevation data are not perfectly aligned with the north and east axes of the UTM coordinate system. The misalignment is caused by the treatment of each UTM zone as a flat surface, when in reality the surface of the earth is curved. Thus, it is necessary to determine the equation for each grid line before locating the intersections.

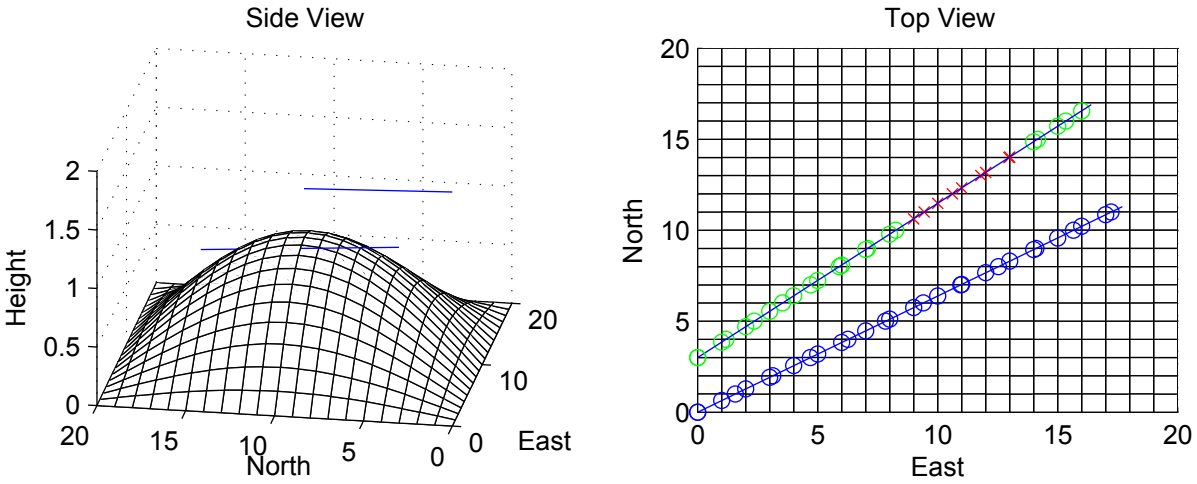

Figure 3.2: Side and top views of an occlusion checking scenario. Some terrain and two example lines of sight are shown, one of which is occluded by the hill. The collision detection algorithm compares the height of the line of sight against the height of the terrain at each place where the line of sight passes over a grid line. These locations are marked with circles and 'x's in the top view. Circles denote points along the line of sight that are occlusion-free, while 'x's denote points along the line of sight that are occluded by the terrain. Linear interpolation is used to determine the height of the terrain along the grid lines between the elevation data points.

3.5 Potential Optimizations

Since the motion field calculations are similar to typical computer graphics calculations, they lend themselves well to parallelization. The motion field and occlusion test calculations for each point are independent of each other and of all the other points. If enough processors were available, each of these calculations could be farmed out to a different processor to be run simultaneously. Our C++ and Matlab implementations do not attempt to parallelize the computations beyond what Matlab does internally. Even so, this part of our code runs quickly enough to be used in real-time. However, optimizations may become important in situations where larger maps or higher resolution data is used. Moving the computation to a GPU, which is specifically designed to perform large batches of matrix multiplications in parallel, could potentially speed up these calculations by a couple orders of magnitude.

3.6 Optical Flow

Dozens of different optical flow algorithms have been developed [19]. For this work, we use the type of optical flow algorithm that tracks a sparse set of features from one video frame to the next. Typically, two successive video frames are considered at a time. A feature detecting algorithm is run on the first video frame to find distinctive visual features, like corners. The optical flow algorithm then searches the second video frame for those same features. It outputs the location and the movement from one frame to the next of each feature for which it found a match. A more detailed discussion of feature trackers and optical flow algorithms can be found in [20].

Motion fields and optical flow are similar in that they both yield vector fields that represent how points are moving through the image plane of a camera. However, the methods used to calculate them are completely different. Motion field calculations rely entirely on a geometric understanding of the scene, while optical flow calculations rely entirely on the video sequence. The accuracy of a motion field calculation depends on the accuracy of the data giving the location and motion of the camera and of the points in view. The accuracy of an optical flow calculation depends on how well the features in the video can be localized and how accurately they are matched.

3.7 Summary

This chapter derived motion field equations specifically for a UAS equipped with a gimballed, zooming camera. These equations depend on an elevation map and the states measured by typical sensors on a UAS. In Chapter 4, the motion field equations are used to predict the amount of blur a planned flight path will induce on the video. In Chapter 5, the motion field equations are used to calculate an expected optical flow field using the estimated states of the UAS as well as to simulate the actual optical flow using the true states of the UAS.

In addition to the motion field equations, an efficient occlusion detection algorithm was developed to determine if the terrain occludes the view between the UAS and a given point. This algorithm is used to remove points from the motion field that fall within the viewing frustum of the camera, but are not actually visible. If not removed, the motion

of these points would corrupt the motion field of the visible points in front of them. The occlusion detecting algorithm is also used in Chapter 4 to determine which points on the terrain are directly lit by sunlight and which are shadowed.

Chapter 4

The Video Targeting Task Performance Metric

4.1 Introduction

UASs are often used to gather information in situations where it would be difficult or dangerous for a human to do so. Although UASs are becoming increasingly sophisticated, they still typically need a human to manage them in one way or another. Path planning algorithms can reduce the UAS's dependence on a human operator. But for a path planning algorithm to be as effective as possible, it must be able to accurately predict and evaluate its performance in the context of the mission goals. Otherwise, it will ignorantly plan paths without knowledge of how those paths affect the success or failure of the mission.

The video utility metric developed in this chapter is focused toward improving the autonomy and performance of UASs in surveillance and reconnaissance tasks. For these scenarios, it is important for the video to be high enough quality that the operator can successfully perform the given detection, recognition, or identification (DRI) task. Ideally, we would like to calculate the operator's actual probability of DRI, assuming that an object of interest is located at a given location that was seen in the video. We are not actually tracking any particular object. Rather, we wish to have a high confidence that the operator would have detected an object of interest in the video if it had been present.

In reality, there are far too many factors that influence an operator's true probability of DRI for this to be practical. These factors can include things like the operator's level of expertise, alertness, and visual acuity. However, we hope to get a rough estimate that is strongly correlated with the true probability.

In surveillance and reconnaissance scenarios, it is best if a video utility metric can be calculated predictively for a path that has not yet been flown. Then it can be used to inform an automated path planning algorithm of the effects that a proposed flight path has on the

video quality. This allows the planning algorithm to maximize the predicted video utility over the flight path, thus increasing the chances of success in the DRI task. If the metric cannot be calculated predictively, then the path must be arbitrarily chosen according to the constraints of the planner or the opinions of a human operator.

Evaluating the utility of the video as it is acquired is also useful because it provides information about the value of the actual data. This information can be used to close the loop for future path planning so that appropriate adjustments can be made to the flight path based on what was and was not seen well. For instance, if the UAS deviated from its course, or if the video of an important area was blurry, then the planner could adjust to revisit the area soon.

The video utility metric presented in this work can be used both to plan flight paths and evaluate them afterward. A path planner was developed in conjunction with the utility metric, although it is not covered in depth here. We have used the planner to plan paths based on the metric, and we have flown those paths in test areas. We also performed a small user study to verify that the utility metric is correlated with subjective human assessment of video quality.

My primary contributions to the video utility metric are the development of the motion field and occlusion detection algorithms. Bryan Morse and Joel Howard contributed the lighting model and experimented with updating the lighting map. The path planning algorithm and observer viewing model were developed by Peter Niedfeldt. Stephen Pledgie helped with the computation of TTP, and all of us helped with the flight tests and user study.

4.2 Literature Review

UAS research often seeks to increase the autonomy of UASs and decrease the workload of human operators [5]. When operators must manage the flight path of one or more UASs, they are unable to focus as much attention on performing the DRI tasks that are the underlying reason for the mission. Operators may also find that it can be difficult to keep track of which areas they have seen or how long it has been since they saw an area. The coverage maps developed by Morse et al. sought to solve this situational awareness problem,

but did not free the operator from manually managing the flight path [6]. Development of video utility metrics that can be used for both planning and evaluation can potentially allow operators to focus exclusively on performing DRI.

Video utility metrics could be used in a wide variety of civilian and military UAS applications. These include surveillance [21, 22], reconnaissance [23, 22], border patrol [4], target tracking [24, 5, 25], and wilderness search and rescue [26, 27, 28, 29, 30].

Still image utility metrics geared toward DRI tasks have been developed in the past. Johnson's criteria, published in 1958, provides basic rules of thumb on how many line pairs of resolution across an object are necessary to perform detection, recognition, or identification. The criteria list empirically determined minimum threshold resolutions with 50 percent probability that a person would succeed for each DRI task [31, 32].

In 2004, Vollmerhausen and Jacobs published the Targeting Task Performance (TTP) metric, which evaluates the quality of still images for performing target acquisition tasks. TTP basically calculates the probability of success for DRI given a still image. This metric was inspired by the Johnson criteria and sought to overcome some of its limitations, such as assumption that the highest visible spatial frequency determines the quality of an image. The TTP metric models the entire imaging process, including characteristics of the imaging hardware and human vision. Specifically, it calculates the effects of the contrast threshold function of the imager and of the human eye on target acquisition performance. The TTP metric has been thoroughly validated and is well-established [7]. Because the process of capturing a still image is similar to that of capturing video, we base the video utility metric on TTP.

The TTP metric assumes knowledge of many parameters that are often not known in surveillance and reconnaissance missions, such as the detailed lighting characteristics of an object and the minimal resolvable contrast (MRC) curve of the sensor-observer system. Niedfeldt et al. calculate the TTP metric using an approximation of the MRC curve and supply reasonable values for several of its parameters [33]. We employ the same methods used by Niedfeldt et al. and introduce new ways to estimate other parameters that may not be known.

The only other aerial video utility metric that we are aware of is the video coverage quality map developed by Morse et al. They calculate a seeability value for each spot on the terrain based on the resolution at which it was seen, the number of times it has been seen, and the diversity of angles from which it has been seen. They also perform georegistration to align the images to the terrain when creating the coverage map [6]. Although similar to our work in principle, there are several differences between the two. Our work does not consider the variety of angles from which a point has been seen and does not perform georegistration. Morse et al. do not account for motion blur, lighting, the size of an object, or characteristics of human vision. One of the largest differences between the two is that the coverage quality map focuses on improving situational awareness so that a human can plan better paths to get good coverage. Our work focuses on enabling the computer to plan paths that ensure good coverage so that the operators can focus on inspecting the video.

Many video quality metrics have been proposed in the field of video processing [34, 35, 36, 37, 38, 39]. However, these are designed to evaluate the effects of video compression, transmission, restoration, enhancement, etc. Some require the input of both a pristine and a degraded video for comparison. Although some of the types of things these metrics evaluate may be of use in our situation, they are not the primary factors that determine success in a DRI task. An up-close, noisy video can be of far more value than a pristine video that was taken too far away to see objects of interest.

4.3 The VTTP Metric

We would like to model the probability of DRI for each spot on the terrain as it passes through the video stream from the UAS. To do this, we propose the Video Targeting Task Performance (VTTP) metric, which estimates how well an operator can see each location by viewing the video. VTTP depends on a number of parameters that we group together in a parameter set, $\mathcal{P}$.

The VTTP metric is based on a TTP calculation for each location on the terrain visible in each frame of the video. Since the TTP metric was designed for static images, it does not take into account various aspects of the video imaging process. The VTTP metric modifies the TTP calculations with a couple of additional components that take into

Table 4.1: Requirements to calculate each component of VTTP.

Component	Elevation map	Predicted Telemetry	Actual Telemetry	Captured video	Object dimensions	Sun position	Ambient & direct lighting	Viewing model
Predicted								
TTP_{p}	•	•			•			
Lighting (L_{p})	•					•	•	
Motion (M_{p})	•	•						
Observer Viewing Model (O)		•						•
Actual								
TTP_{a}	•		•		•			
Motion (M_{a})				•				
Observer Viewing Model (O)			•					•

account some of the aspects of video that TTP does not. These components can be included in the calculation if the information necessary to compute them is available, or excluded otherwise. We define $\mathcal{P}$ as a set of parameters that will be used throughout the TTP and VTTP calculations.

Table 4.1 specifies the information needed to calculate the components. In this work, we develop two VTTP components: a motion factor M and an observer viewing model O. We present methods to calculate these components both during the planning phase as well as after flight.

In addition to developing the two VTTP components, we made a few other necessary modifications to the TTP calculation. First, TTP requires the calculation of the object to background contrast. However, the lighting coefficients that are necessary to calculate the object to background contrast are rarely known. We develop an alternative method to estimate the contrast that does not require any prior knowledge about the lighting characteristics of an object or the terrain around it.

Second, the TTP metric does not have a built-in mechanism of determining what areas of the terrain are in view. Such a mechanism is not necessary because TTP operates on still images. The VTTP metric, on the other hand, is constantly being calculated while

the camera moves and parts of the terrain come in and out of view. Because of this, it is impractical to manually specify the area that each subsequent video frame covers. Using the states $\mathbf{x}$ of the UAS and the terrain map contained in the parameter set $\mathcal{P}$, we can automatically determine which terrain points fall within the viewing frustum for each frame. Furthermore, we use the occlusion detection algorithm from Section 3.4 to determine which points in the viewing frustum are occluded by the terrain and which are not. This prevents the algorithm, for example, from mistakenly thinking that it saw points on the backside of a hill.

Some parts of the VTTP metric must be calculated differently depending on whether they are being predicted for a pre-planned trajectory, or whether they are being calculated for actual flight data and collected video. We distinguish the method used to calculate these values by adding a subscript 'p' for predicted values and a subscript 'a' for actual values.

The VTTP and TTP metrics estimate the probability of DRI assuming that an object of interest exists at a given location. Although an object will not usually be present at a given location, a high VTTP or TTP value indicates confidence that it would be detected if it were present. Thus, factors like the area of an object in the image plane and the contrast between an object and the background are calculated as if an object was located at the terrain point under consideration.

4.3.1 Lighting

The TTP calculation depends on the contrast between an object and the background. However, the lighting coefficients of an object and terrain necessary to calculate this contrast are typically not known. One factor that always affects the contrast between an object and the background is the amount of light present. When the scene is brightly lit, there be more contrast than when it is dimly lit. Thus, in the absence of knowledge of the lighting coefficients, we use a model of the amount of lighting as an estimate of contrast. This at least gives us a value that is proportional to the actual contrast.

To estimate the amount of lighting, we use the Lambertian lighting model. This model accounts for both direct and ambient lighting. The direct lighting specifies the amount of light coming directly from a light source like the sun. Surfaces facing directly toward the

light source will be lit the brightest. As a surface rotates away from the direction from which the light is coming, its lighting drops off as the cosine of the angle between the surface normal and the vector pointing towards the light source. If the surface falls within the shadow of another object, then it is not directly lit. Thus, we calculate the direct lighting for a spot on the terrain as

$$D(z, \mathcal{P}) = S(z, \mathcal{P})(L_d^\top N(z, \mathcal{P})), \tag{4.1}$$

where z is an index specifying a point in the terrain map, $S(z)$ is 0 if the terrain point falls in a shadow and 1 otherwise, L_d is a unit vector pointing toward the sun, and $N(z)$ is the unit normal of the terrain at location z. Both $S(z)$ and L_d assume knowledge of the current position of the sun. This information is readily available from astronomic data or from the National Oceanic and Atmospheric Administration's online solar calculator. The sun's position could also be estimated by the user.

The ambient lighting component of the Lambertian model represents the amount of light that is scattered randomly throughout the scene, primarily due to atmospheric conditions. We define a parameter A that the user sets to indicate the approximate ratio of ambient and direct lighting at the time of flight. On a very cloudy day, A would be set close to 1 to indicate that most of the lighting is ambient. In this case, the terrain does not cast any strong shadows. On a clear and sunny day, A would be set to a relatively low number, like 0.2. This indicates the strong presence of direct lighting and a stark contrast between the areas that are shadowed and those that are not. Setting A to zero would indicate the complete absence of ambient lighting. However, this is not realistic as there are always at least some atmospheric scattering effects even on a clear day.

The illumination $L(z, \mathcal{P})$ of a point on the terrain, which we use as an estimate of the contrast $C(z, \mathcal{P})$, is given by the Lambertian model to be

$$C(z, \mathcal{P}) \approx L(z, \mathcal{P}) = (1 - A)D(z) + A, \tag{4.2}$$

where $D(z)$ is the direct lighting from Equation (4.1) and A is the user-specified ratio of ambient and direct lighting, contained in the parameter set $\mathcal{P}$.

Figure 4.1: Orthoimagery of an area near Eureka, UT (left), the calculated lighting map overlaid on the orthoimagery (middle), and the lighting map $L(z, \mathcal{P})$ (right). This was one of the areas we used to test flight paths planned using the VTTP metric. The lighting map was computed for 8:20 a.m. on Sep. 15, 2011, with $A = 0.2$.

Since the sun moves slowly across the sky, its position can be assumed to be constant over short periods of time. When this assumption is made, the lighting at each point on the terrain can be pre-computed and stored in a lighting map contained in the parameter set $\mathcal{P}$. This can save computation by avoiding the recalculation of the lighting for terrain points that are in the field of view over multiple frames of the video. For lengthy flights, the lighting map could be periodically recomputed to keep it current with the position of the sun. Figure 4.1 shows a sample lighting map calculated for an area near Eureka, UT, along with the orthoimagery of the area.

We used the same method to estimate the contrast for both planning and evaluation after flight. We experimented with estimating the actual lighting based on the intensity values of the pixels in the video. We also tried using these estimates to update the lighting map on the fly. However, we found that the auto-exposure of the camera corrupted these estimates. When the camera was pointed into a shadowed area, the auto-exposure would adjust until the video actually came out brighter overall than video of areas lit by direct sunlight. Although the video of shadowed areas was artificially brighter, the contrast and quality of the video was lower due to the lower levels of light. Since our platform did not provide the ability to query the current exposure settings of the camera, we abandoned efforts to estimate the lighting based on the video. If the exposure settings were available, it might

be possible to adjust the pixel intensities according to the exposure to get a better estimate of the actual amount of light present in the areas seen by the video.

4.3.2 Calculating TTP

We follow the method put forth by Niedfeldt et al. to calculate TTP for each point on the terrain in the field of view of the UAS. The equations and parameter values presented in this section are based on their work [33]. Following their method allows us to approximate the MRC curve for the sensor-observer system instead of having to determine it through the process given in *The Infrared and Electro-optical Systems Handbook* [40]. The MRC curve specifies the minimum contrast required for a human to resolve a given spatial frequency [7].

To calculate TTP, we first compute the integral

$$\Phi(z, \mathcal{P}) = \int_{\kappa_{\text{low}}}^{\kappa_{\text{cut}}} \left(\frac{C(z, \mathcal{P})}{MRC(\kappa, \mathcal{P})} \right)^{\frac{1}{2}} \, \mathrm{d}\kappa, \tag{4.3}$$

where C is the contrast between an object and the background at terrain point z, $MRC(\kappa, \mathcal{P})$ is the minimum resolvable contrast curve, and κ_{low} and κ_{cut} specify the range of relevant spatial frequencies. A spatial frequency κ is considered relevant if $C(z, \mathcal{P}) > MRC(\kappa, \mathcal{P})$, meaning there is excess contrast beyond the minimum resolvable contrast at that spatial frequency. In practice, κ_{low} is very close to zero [7]. The contrast C can be calculated as specified in Niedfeldt et al. [33] if the lighting coefficients of an object and terrain are known, or it can be approximated using Equation (4.2) if the coefficients are not known. The MRC curve is approximated to be

$$MRC(\kappa, \mathcal{P}) \approx \exp\left(\frac{-b + \sqrt{b^2 - 4c(a - \kappa \measuredangle_{\text{pixel}})}}{2c} \right), \tag{4.4}$$

where $\measuredangle_{\text{pixel}}$ is the field of view of a pixel, $a = 1.4445$, $b = .0742$, and $c = -0.0235$. The values for a, b, and c are considered part of the parameter set $\mathcal{P}$ and are provided in [33]. The TTP metric is then calculated as

$$TTP(z, \mathbf{x}, \mathcal{P}) = \frac{\left(N_{\text{resolved}}(z, \mathbf{x}, \mathcal{P})/N_{50}\right)^{E(z, \mathbf{x}, \mathcal{P})}}{1 + \left(N_{\text{resolved}}(z, \mathbf{x}, \mathcal{P})/N_{50}\right)^{E(z, \mathbf{x}, \mathcal{P})}}, \tag{4.5}$$

where

$$E(z, \mathbf{x}, \mathcal{P}) = u + v^{(N_{\text{resolved}}(z,\mathbf{x},\mathcal{P})/N_{50})}$$

and

$$N_{\text{resolved}}(z, \mathbf{x}, \mathcal{P}) = \frac{\sqrt{A_{\text{obj}}(z, \mathbf{x}, \mathcal{P})}\Phi(z, \mathbf{x}, \mathcal{P})}{s(z, \mathbf{x}, \mathcal{P})},$$

and where $A_{\text{obj}}(z, \mathbf{x}, \mathcal{P})$ is the area of an object in the image plane, $s(z, \mathbf{x}, \mathcal{P})$ is the slant range between the UAS and the terrain point, $\mathbf{x}$ is the state vector of the UAS, $u = 1.33$, $v = 0.23$, and $N_{50} = 30$. The values for u and v are statistically determined from field and lab data, and N_{50} represents the empirically determined average number of cycles required for analysts to successfully identify objects 50% of the time. The values for u, v, and N_{50} are considered part of the parameter set $\mathcal{P}$ and are provided in [33]. The value of $TTP(z, \mathbf{x}, \mathcal{P})$ represents the probability that the operator would succeed at the DRI task using a still image taken from the UAS if an object were located at the terrain point z.

4.3.3 Motion

The speed at which a point moves through the image plane is directly proportional to the amount of blurring of that point in the video. This blurring degrades the quality of the video and lowers the probability of DRI. The motion component of the VTTP metric models this effect. Since the motion of an object is unknown, we can only account for the blurring induced by the motion of the UAS and the camera gimbal.

To calculate the motion component of the VTTP metric, we must first determine the speed at which each point in view is moving through the image plane. We can use the motion field equations to do this prior to a flight if we can predict the UAS states along the planned flight path. We assume that the UAS flies at a constant altitude and velocity. We also assume that it maintains a constant turning rate between waypoints. Using the methods given in [1], we calculate the necessary pitch and roll angles for the UAS to maintain its altitude and make a constant rate turn between waypoints. The planned camera path is constructed by associating a camera waypoint with each flight path waypoint. The camera waypoint specifies the spot on the ground that the camera should be pointed towards when the aircraft arrives at the corresponding flight path waypoint. Between waypoints, the gimbal

attempts to orient the camera such that the look-at point linearly interpolates between the specified camera waypoints. Given these constraints, we can calculate all the states of the UAS for an ideal flight. With the ability to estimate all the states of the UAS at any point along the planned path, we can then use the motion field equation given in Equation (3.25) to predict the amount of blur that the flight plan causes for each terrain point in view.

Since the UAS is never able to perfectly follow the planned path, it is necessary to use actual flight data when calculating the actual motion. The motion field equations can be used to calculate the motion using the state estimates from the EKF. However, there are three problems with this method. First, the state estimates are somewhat noisy due to the noise in the sensors upon which they are based. Second, our platform was only able to transmit state estimates at a relatively low rate, sometimes as little as 2 or 3 Hz. Thus, we would have to introduce even more error by interpolating the state estimates up to same rate at which we received video frames, 15 Hz. Finally, our platform transmitted video and state estimates using two different transmitters. The video frames were not timestamped until they were received by the base station computer. Thus, the recorded video and state estimates were not well synced, sometimes exhibiting offsets of more than a second. We improved the situation by manually syncing the video and telemetry prior to evaluating each flight, but were only able to sync them to within a few hundred milliseconds.

Because of these problems, we found that running an optical flow algorithm on the recorded video provided much more accurate estimates of the motion of points through the image plane. We used the pyramidal Lucas-Kanade optical flow algorithm provided by OpenCV [41, 42]. To get a relatively even spread of features across each frame, we configured the algorithm to search for 60 features spaced a minimum distance of 60 pixels apart. We used a 60×60 search window, a quality level of 0.01, and 6 pyramidal levels. This configuration seemed to work well for all of the video that we collected.

For each frame of the video, we average the magnitudes of the calculated optical flow vectors. This average is used to calculate the motion component for all of the points in the field of view. More accuracy could perhaps be achieved by interpolating the flow field to the locations of each terrain point in the frame, but the gain would likely be canceled out by the noise in the state estimates and their misalignment with the video. Furthermore, the

vast majority of the video exhibited relatively uniform motion across the entire frame and averaging made the algorithm more robust to outliers.

To model the effects of motion on the probability of DRI, we empirically chose a threshold amount of motion m_{T} based on our examination of video from flight tests. When the amount of motion of a point exceeds this threshold, we consider that frame of video too blurry to be of use and zero out the VTTP metric. When the point is not moving, the motion component does not modify its VTTP value. We linearly interpolate the effect of motion between these two extremes. Thus, we calculate the predicted motion factor as

$$M_{\mathrm{p}}(\xi, \mathcal{P}) = \max\left(0, 1 - \frac{\sqrt{m_{\mathrm{u}}^2 + m_{\mathrm{v}}^2}}{m_{\mathrm{T}}}\right), \tag{4.6}$$

where $(m_{\mathrm{u}}, m_{\mathrm{v}})^\top$ is the calculated motion from Equation (3.25) and m_{T} is the threshold amount of motion measured in pixels per frame. We set $m_{\mathrm{T}} = 40$ pixels per frame and include it in the parameter set $\mathcal{P}$. The actual motion factor is given by

$$M_{\mathrm{a}}(v, \mathcal{P}) = \max\left(0, 1 - \frac{1}{n m_{\mathrm{T}}} \sum_{i=1}^{n} \sqrt{m_{\mathrm{u},i}^2 + m_{\mathrm{v},i}^2}\right), \tag{4.7}$$

where v is the video stream, n is the number of tracked features, and $(m_{\mathrm{u},i}, m_{\mathrm{v},i})^\top$ is the motion of the ith feature between the two frames.

4.3.4 Observer Viewing Model

The TTP metric calculates a probability of DRI that assumes that the viewer has an indefinite amount of time to examine the still image. When dealing with video, the operator does not have time to thoroughly inspect every spot in every frame before moving on to the next. Rather, the operator is most likely to focus his attention on the middle area of the video most of the time. If an object passes through a corner or along an edge of the video, the operator will be less likely to detect it because he is less likely to be looking at that area of the video frame. Points that pass near the boundaries of the image will also tend to be in the field of view of the camera for a shorter amount of time, which further limits the operator's ability to perform DRI for those locations.

We introduce an observer viewing model O to account for this aspect of performing DRI with video. This viewing model is a Gaussian shaped discount factor applied to the VTTP based on where the point under consideration is located in the video frame. The factor peaks at a value of one in the center of the image and falls off towards the edges and corners. It is given by

$$O(\xi, \mathcal{P}) = \exp\left(-\frac{1}{2}((x,y)^\top - (o_\mathrm{x}, o_\mathrm{y})^\top)^\top \Sigma_O^{-1}((x,y)^\top - (o_\mathrm{x}, o_\mathrm{y})^\top)\right), \tag{4.8}$$

where $(x,y)^\top$ is the location of the point projected into the image plane, $(o_\mathrm{x}, o_\mathrm{y})^\top$ is the coordinate of the center of the image, and Σ_O is the covariance matrix of the Gaussian. The covariance matrix is included in the parameter set $\mathcal{P}$.

4.3.5 Calculating VTTP

The equation used to predict VTTP during the path planning stage is

$$VTTP_\mathrm{p}(z, \xi, \mathcal{P}) = TTP(z, \xi, \mathcal{P}) M_\mathrm{p}(\xi, \mathcal{P}) O(\xi, \mathcal{P}), \tag{4.9}$$

where ξ is the predicted state of the UAS at a point along the planned flight path, $TTP(z, \xi, \mathcal{P})$ is the TTP value calculated from Equation (4.5), $M_\mathrm{p}(\xi, \mathcal{P})$ is the predicted motion component calculated from Equation (4.6), and $O(\xi, \mathcal{P})$ is the observer viewing model given in Equation (4.8). Once the actual video has been obtained, the actual VTTP is

$$VTTP_\mathrm{a}(z, \mathbf{x}, \mathcal{P}) = TTP(z, \xi, \mathcal{P}) M_\mathrm{a}(v) O(\mathbf{x}, \mathcal{P}), \tag{4.10}$$

where $\mathbf{x}$ is the state of the UAS, v is the video stream, and $M_\mathrm{a}(v)$ is the actual motion component calculated from Equation (4.7).

4.4 Using VTTP for Path Planning

The VTTP metric cannot be used alone to plan paths. If it were, the planner would likely find the spot with the best viewing conditions and stare at it indefinitely. Rather, it

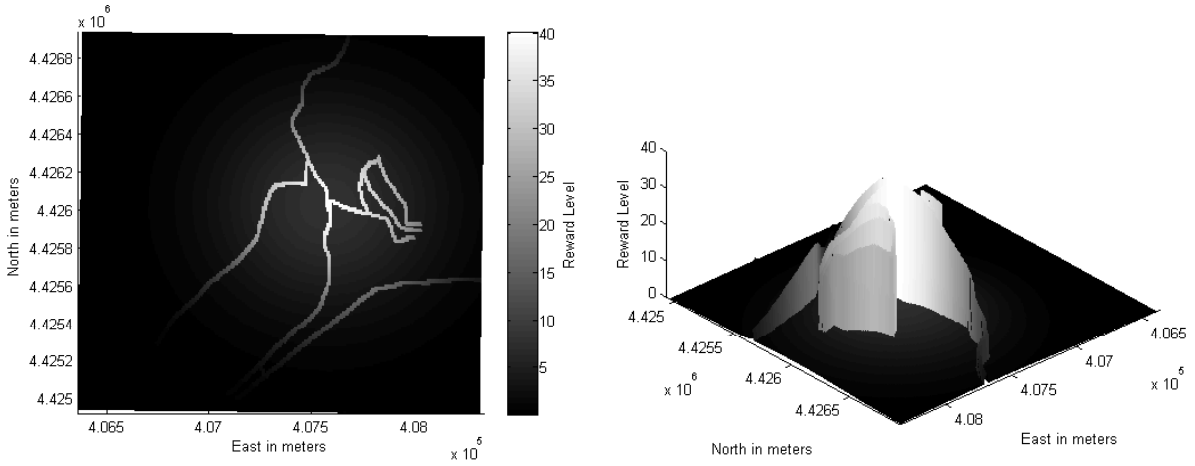

Figure 4.2: Top and side views of an example reward map that has grown over a period of 20 seconds. The reward levels grow at a much faster rate along the roads in the area to prioritize them in the surveillance task.

must be used in conjunction with some other goal or reward mechanism designed to achieve the desired behavior.

For our tests, we assumed a scenario where the UAS is performing surveillance around a sensitive area, like a military base or nuclear reactor. We set up a Gaussian shaped reward map that assigns a reward to each terrain point in the area. The reward is large in sensitive areas and grows over time. The growth rates are greatly increased along roads in the area to cause them to be prioritized in the surveillance. Figure 4.2 plots an example reward map that was allowed to grow over 20 seconds. This map covers the same area as was shown for the orthoimagery and lighting map in Figure 4.1.

The reward map tells the system what the value of seeing different points in the area is, while the VTTP metric tells it how well those points were seen. When the UAS sees a point during its flight, it consumes a portion of the reward for that point depending on how well the VTTP metric says the point was seen. Thus, little reward remains once a point has been seen well and the system is encouraged to move on to inspect other areas with more reward. While the UAS inspects other areas, the reward for the point slowly grows back until there is once again a strong incentive to view it again. The path planning algorithm seeks to maximize the amount of reward consumed. This setup allows the system to continuously perform surveillance without requiring a human to direct the UAS. It also provides a way for

operators to prioritize what areas are most important by adjusting the structure and growth of the reward map.

4.5 Validation and Results

We provide two preliminary validations of the usefulness of the VTTP metric. First, we show that the VTTP values calculated during the planning phase are indicative of the actual values achieved when the path is flown. This indicates that searching for paths that maximize the predicted VTTP will also maximize the achieved VTTP. Second, we performed a small user study that verifies that the VTTP metric correlates with human assessment of the quality of video for a DRI task.

4.5.1 Predicted vs. Actual VTTP

We verify that the predicted VTTP is indicative of the achieved VTTP by selecting a point and graphing both the predicted and achieved values for that point over time. For added insight, we also calculate the VTTP with the actual estimated states from the test flight, but using the motion field instead of optical flow. This allows us to compare how well the motion field predicts the optical flow, given the actual states. We denote this value as $VTTP_{\mathrm{p}^+}$, and it is given by the equation

$$VTTP_{\mathrm{p}^+}(z, \mathbf{x}, \mathcal{P}) = TTP(z, \mathbf{x}, \mathcal{P}) M_{\mathrm{p}}(\mathbf{x}, \mathcal{P}) O(\mathbf{x}, \mathcal{P}). \tag{4.11}$$

Figures 4.3 and 4.4 show the predicted and actual calculations for two different points on the terrain observed during two different flights. Note that the predicted value, $VTTP_{\mathrm{p}}$, effectively serves as a cap on the achieved value, $VTTP_{\mathrm{a}}$. This makes sense because the UAS is unable to perfectly follow the planned path, which should maximize VTTP. Therefore, deviations from the path tend to lower the value of $VTTP_{\mathrm{a}}$ relative to $VTTP_{\mathrm{p}}$, which assumes perfect path following.

Figures 4.3 and 4.4 also show that the $VTTP_{\mathrm{p}^+}$ value very closely follows $VTTP_{\mathrm{a}}$. This indicates that the motion field calculations closely agree with the output of the optical flow algorithm. Close inspection of the graphs reveals that $VTTP_{\mathrm{p}^+}$ varies more smoothly

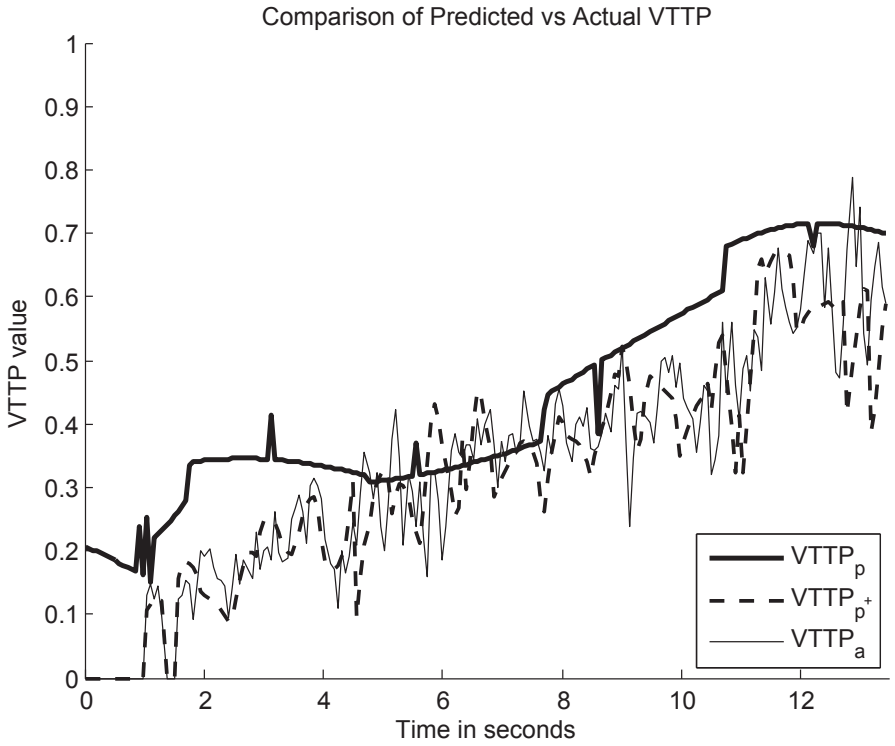

Figure 4.3: Comparison of the predicted $VTTP_{\mathrm{p}}$ for a terrain point seen by a planned flight path versus the actual $VTTP_{\mathrm{a}}$ achieved during flight. The graph also shows the $VTTP_{\mathrm{p}^+}$ curve, which is calculated like $VTTP_{\mathrm{a}}$ except that the motion field equations are used in place of optical flow. The closeness of the $VTTP_{\mathrm{p}^+}$ and $VTTP_{\mathrm{a}}$ curves verifies that the motion field calculations give comparable results to the optical flow algorithm. Note however, that the optical flow captures higher frequency motion of the UAS because the transmitted video has a higher sampling rate than the transmitted state estimates. The $VTTP_{\mathrm{p}}$ curve is consistently higher than the curves based on actual flight data because it assumes the planned path is followed perfectly. However, the $VTTP_{\mathrm{a}}$ curve clearly follows $VTTP_{\mathrm{p}}$, indicating that a path planner that maximizes $VTTP_{\mathrm{p}}$ will also maximize, or at least improve, $VTTP_{\mathrm{a}}$.

than $VTTP_{\mathrm{a}}$, which is more noisy and spiky. This reflects the previously mentioned limitation of our platform's rate of transmission of state estimate data, which sometimes dropped as low as 2 or 3 Hz. Since video frames were received at the higher rate of 15 Hz, the optical flow is able to capture the higher frequency motion of the UAS better than the transmitted state estimates.

On occasion, $VTTP_{\mathrm{a}}$ does actually spike above $VTTP_{\mathrm{p}}$. This is because the shakiness of the camera motion can cause there to be more or less motion blur for a few frames than

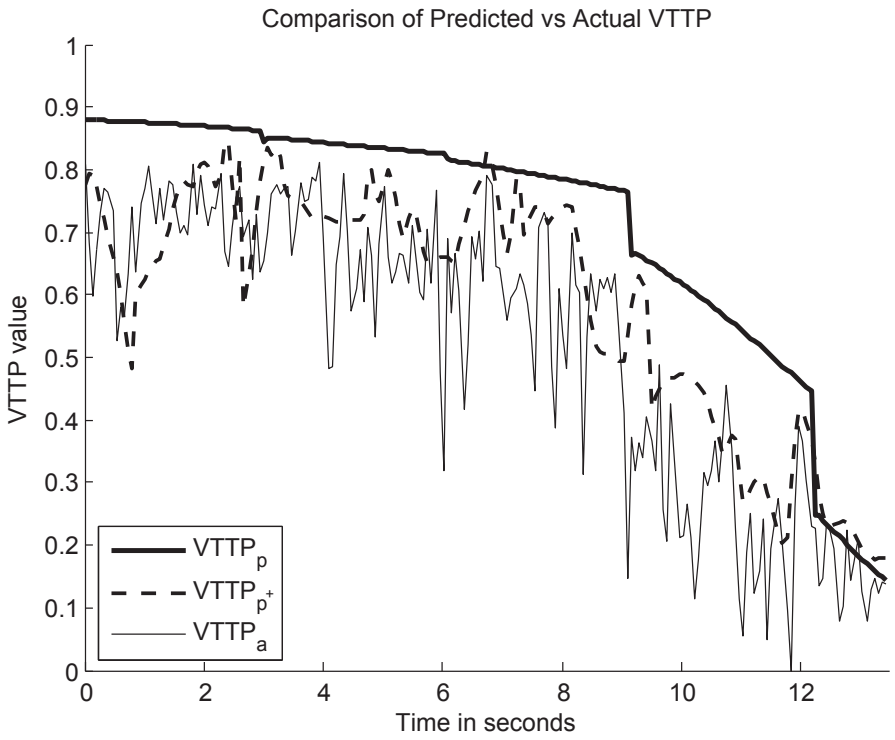

Figure 4.4: Another comparison of $VTTP_{\mathrm{p}}$, $VTTP_{\mathrm{p}^+}$, and $VTTP_{\mathrm{a}}$, but for a different terrain point seen during a different test flight. This graph provides additional data supporting the analysis made in Figure 4.3.

there would be if the camera moved smoothly along the planned path. So while $VTTP_{\mathrm{a}}$ may spike above $VTTP_{\mathrm{p}}$ momentarily, its average value never exceeds $VTTP_{\mathrm{p}}$.

The $VTTP_{\mathrm{p}}$ values exhibit occasional steps or spikes in what is otherwise a smooth curve. This behavior is caused by our assumptions about how the UAS travels between waypoints. If the UAS maintains a constant turn rate and roll angle between waypoints, then there is a step in the assumed roll angle as the UAS passes through a waypoint and transitions from one path segment to another. This step in the roll angle also requires steps in the gimbal angles to keep the camera pointed at the same location. Spikes could also be caused by the limitations of the gimbal motion of the camera. If the elevation angle of the gimbal ever passes close to 90 degrees, the gimbal may need to spin rapidly to keep tracking the camera path. The azimuth angle of the camera's gimbal was also limited between -180 degrees and 180 degrees, which can cause the gimbal to unwind by spinning if it tries to track a path that passes directly behind the UAS.

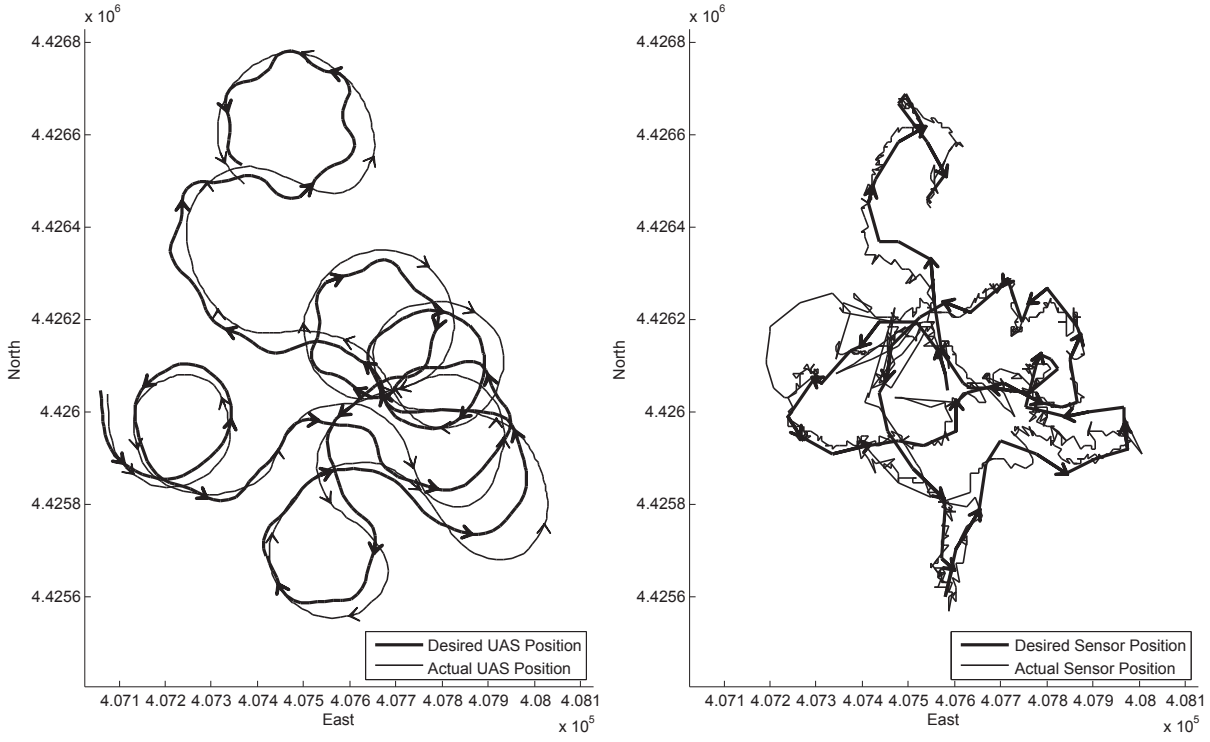

Figure 4.5: A comparison of the planned and actual flight path of the UAS (left) and a comparison of the planned and actual camera path (right). The effects of a wind blowing to the south east are evident in the deviations of the actual flight path from the planned path. Also, note the shakiness of the actual camera path due to wind and other disturbances on the aircraft. This shakiness is evident in the $VTTP_{\mathrm{a}}$ values plotted in Figures 4.3 and 4.4.

Figure 4.5 shows the how the actual flight and camera paths deviate from the planned paths. Note the shakiness of the actual camera path caused by disturbances like wind. This corresponds to the noisy $VTTP_{\mathrm{a}}$ values in Figures 4.3 and 4.4.

4.5.2 User Study

The ultimate goal of VTTP metric is to improve the performance of DRI tasks. Since comparable, established metrics do not exist, the effectiveness of VTTP can only be established based upon subjective human evaluation and data collected through user studies. Ideally, a large amount of data would be collected for trained UAS operators performing DRI tasks on video from various situations. The VTTP metric would also be calculated on these videos and the results would be statistically compared against the performance of the human operators. This would quantify how well the VTTP metric predicts actual DRI performance.

Unfortunately, our resources are limited and we do not have access to trained UAS operators. As such, we are only able to provide a preliminary verification of the VTTP metric based on a small user study that we performed. We extracted ten video clips taken from our test flights for the user study. Each clip lasted for about 10 to 14 seconds and contained a charcoal-gray extended cab truck in the field of view for the duration of the clip. The clips were extracted from a variety of scenarios, although the truck was stationary in all of them. These included near and far viewing distances, different viewing angles, shadowed and sunny locations, and on-road and off-road locations.

In the user study, we administered a test developed by Likert [43] to twenty participants. All of the participants were college students and many were members of the BYU MAGICC Lab, where this research was conducted. However, students directly involved in the project did not participate in the study. The participants were asked to rate how hard it was to detect and identify the truck in each video clip on a five point rating scale. The five ratings ranged from very easy to very hard. The order in which each participant viewed the video clips was also randomized.

Because participants will have individual biases, it is impossible to directly correlate user ratings with the VTTP metric. Following Likert's method [43], we instead calculate the pairwise differences between a participant's ratings of all the video clips. Similarly, we calculate the pairwise differences in the VTTP values calculated for the location of the truck in each of the video clips. When the differences in the participants' scores for two video clips are large, we would expect that the differences in VTTP between the two clips should also be large.

While analyzing the results of the user study, we identified three video clips that skewed the results due to modeling error in the VTTP metric. Two of these clips were captured when the UAS was near the truck and the camera was zoomed to its maximum level. The high level of zoom combined with the shakiness of the flight caused large amounts of motion blur in the images. The magnitude of this motion was often close to or greater than the motion threshold, thus making the calculated VTTP values very small. However, the truck was so large in the field of view of the video that the user study participants found it easy to detect and identify despite the large amount of blur. This indicates that

Table 4.2: User study results comparing the user preference differences and the corresponding VTTP differences between pairwise combinations of video clips. Notice that as the preference difference increases, so does the mean of the TTP differences for those same videos.

Preference Difference	Mean VTTP Difference	Std. Deviation of VTTP Difference	Number of Samples	Significance (significant if $p < .05$
0	3.83×10^{-19}	0.0940	290	$p < 0.0001$
1	0.0534	0.1217	153	$p < 0.0001$
2	0.1178	0.1191	110	$p < 0.0001$
3	0.1631	0.1131	63	$p = 0.0152$
4	0.2006	0.0688	19	$p = 0.1757$

the assumption of a constant threshold for motion blur is not good in all situations. Instead, the motion threshold should probably scale proportionally with the dimensions of an object in the image plane. The third clip received a high VTTP rating, but the participants found it difficult to distinguish the truck from some large bushes in the surrounding area. This indicates that the VTTP metric could benefit from the addition of a component that accounts for the amount of visual clutter in the scene.

Table 4.2 summarizes the results of the study after removing the outlier video clips. Note that the preference difference and the mean VTTP difference increase together. The data presented in the top four rows all provide statistically significant evidence of the correlation between VTTP and user preference. The p-value for the bottom row is large mainly because the number of samples is small. There were fewer pairwise combinations of videos with a preference difference of four, which requires that one video clip was rated five and the other was rated one.

In addition to the results of the user study, we personally found that the paths planned based on VTTP reduced the user workload and provided higher quality video than could be obtained by manually managing the flight path and camera gimbal.

Chapter 5

Detecting GPS Spoofing

5.1 Introduction

Modern society and infrastructure have become dependent on GPS in countless ways. However, the civilian band GPS signal used in almost all cases is open and vulnerable to attack. GPS spoofers can construct and broadcast fake GPS signals that will spoof GPS receivers to provide false readings. Although this vulnerability has been known for years, virtually nothing has been done by manufacturers of GPS receivers to protect against it.

The absence of even the most basic anti-spoofing technology in modern GPS receivers is likely explained by the rarity of reported GPS spoofing attacks. Manufacturers likely do not wish to spend extra effort or money to implement spoofing countermeasures when spoofing has not been a problem. Unfortunately, our dependence on GPS has long since passed the point where the first significant spoofing incident could cost human lives. Even in 2001, the Volpe report warned about the potential for GPS spoofers to cause collisions between ships or to run a barge into a bridge during rush-hour [9].

Various countermeasures for GPS spoofing attacks have been proposed [15, 16], but most of them are either ineffective against sophisticated spoofing or require fundamental alteration of the way the GPS system works. The Volpe report notes that the best proposed countermeasure that would work with the current GPS system is using multiple antennas to calculate the directions from which the GPS signals are coming. However, this method could require baselines on the order of meters between the antennas and is very susceptible to multipath errors [9]. For small aircraft with limited payloads, using multiple antennas may not be practical.

This work proposes a new method of detecting GPS spoofing on a UAS. It relies on sensors that many UASs are already equipped with and works with the current GPS system.

The Iran-US RQ-170 incident highlights the importance of preventing a UAS from being led astray. Iran captured a downed US RQ-170 Sentinel drone that had only suffered minimal damage on December 4, 2011. Iranian scientists claimed that they downed the drone by spoofing its GPS and tricking it into landing as if it were at its home base [44]. Iran later announced that it was reverse engineering the drone and had extracted sensitive intelligence data [45]. Whether or not all of the Iranian claims are true, the incident placed advanced technology and sensitive military data into Iranian hands. Developing and implementing measures to detect spoofed GPS readings aboard a UAS can help guard against similar incidents.

5.1.1 How GPS Works

The information presented in this section is based on *The Navstar GPS User Equipment Introduction*, which explains the operation of the GPS system in detail [46]. Similar but more concise information is available at `www.gps.gov`.

The GPS system consists of a constellation of satellites and several ground stations that keep them calibrated. The satellite orbits are arranged to ensure that at least four satellites are always visible from nearly any point on the surface of the earth. The satellites broadcast signals on two different frequencies—the military band and the civilian band. The military band signal is encrypted and reserved for military use, while the civilian band is open to everyone. Since all of the GPS satellites transmit at the same frequency, each one modulates its signal with a unique Coarse/Acquisition (C/A) code. This allows GPS receivers to separate out all the different satellite signals using code division multiple access (CDMA).

The signal from each GPS satellite contains the current time from an atomic clock on board the satellite as well as ephemeris data that gives the satellite's position and trajectory. GPS receivers calculate their distance from each satellite by multiplying the amount of time it took for the signal to travel from the satellite to the receiver by the speed of light. Thus, each satellite's signal allows the receiver to restrict its position to a sphere centered on the satellite's position. The intersection of three such spheres narrows the position of the receiver to two points, as can be seen in Figure 5.1. One of these two points will be near the surface of the earth, while the other will be out in space. The receiver assumes that the position

closest to the surface of the earth is correct. This process of calculating a position based on distance measurements from other points is known as trilateration.

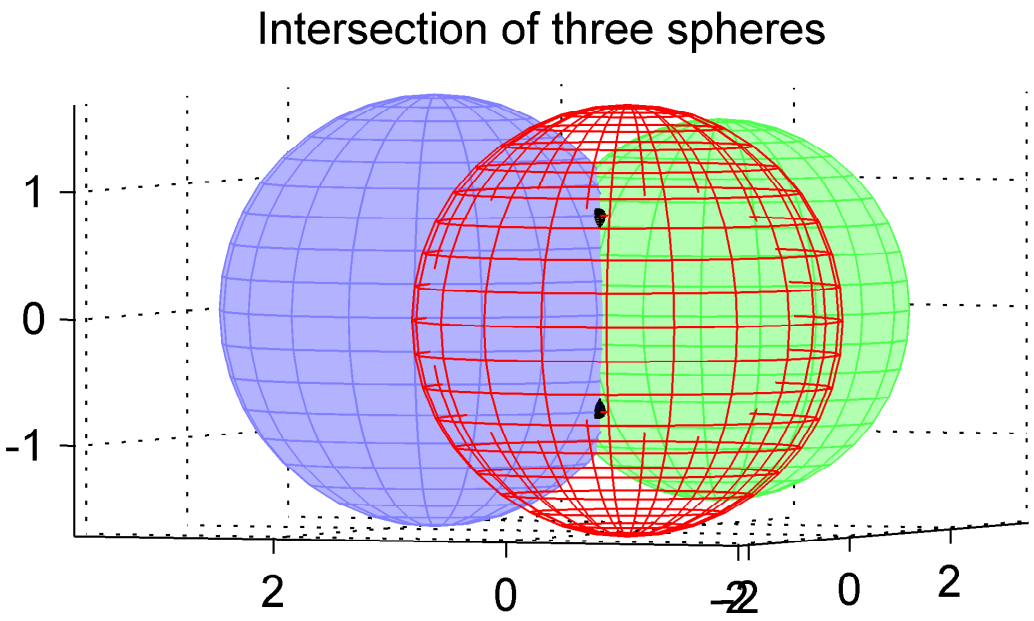

Figure 5.1: Three overlapping spheres intersect at two points, marked by black spots above. If these spheres are centered on satellites above the surface of the earth, then the lower intersection will be near the surface of the earth while the upper one will be out in space.

If the receiver's internal clock was precisely synced with the GPS system, then the signals from three satellites would be sufficient to establish its location. However, GPS receivers have inaccurate internal clocks that are not synced with the atomic clocks on board the satellites. This prevents receivers from being able to subtract the time the signal was transmitted from their internal clock to calculate the amount of time it took for the signal to travel to the receiver. Because of this, a fourth satellite signal is required to allow the receiver to simultaneously solve for both its position and the current GPS time. If more than four satellites signals are available, then the receiver can improve accuracy by using a least squares approach to solve the overdetermined system.

In addition to calculating their position, many GPS receivers also output velocity. Rather than using numerical differentiation to calculate velocity, GPS receivers use Doppler shifts between the received signals. When searching for satellite signals, the receiver scans

a range of frequencies around the specified transmission frequency to locate the actual Doppler-shifted signals. Once located, the receiver locks onto a satellite's signal and tracks its frequency, so the Doppler shift is always known. The receiver calculates its relative speed toward or away from each satellite based on the Doppler shift for that satellite's signal, much like a policeman's radar gun. Given these speed measurements and its position, the receiver is able to construct its three-dimensional velocity vector.

5.2 Literature Review

One of the earlier warnings about the potential for GPS spoofing was given in 1996 when the *Navstar GPS User Equipment Introduction* noted that the C/A code is not protected against spoofing [46]. However, this single sentence statement was buried in over 200 pages of information and little was published about GPS spoofing for the next several years.

In 2001, the John A. Volpe National Transportation Systems Center prepared a vulnerability assessment report on GPS for the Department of Transportation. This report indicates that the GPS system is vulnerable to jamming, spoofing, and meaconing—an attack that involves receiving, delaying, and retransmitting actual GPS signals to confuse receivers. It warns that even small errors in GPS could be enough to cause collisions of ships with each other or with bridges in bad weather. The authors also cite an incident where a helicopter convoy flew off course because the pilots trusted faulty GPS readings and ignored readily apparent visual cues. Furthermore, they found that virtually all anti-jamming equipment could be thwarted by a spoofing attack. Even many of the countermeasures proposed specifically for spoofing, like monitoring the power of the GPS signal or watching for loss of satellite lock, could be defeated by sophisticated spoofing attacks. They noted that the range gate of typical GPS receivers spans about 300 meters, meaning that an attacker would only need a rough estimate of his victim's position to be able to spoof without causing loss of lock. Most of the countermeasures that could not be defeated by spoofing involved fundamental changes to how the GPS system works, like introducing cryptographic authentication to the signals broadcast by the satellites. The report notes that the best countermeasure that could be implemented immediately was the use of multiple antennas to distinguish the angles from which the GPS signals arrive. But, that countermeasure is susceptible to multipath errors

and can require careful placement of antennas at least meters apart. Throughout the entire report, the authors repeatedly emphasize the need to investigate anti-spoofing technologies and inform users of the potential risks [9].

In 2002, Warner and Johnston successfully demonstrated a simple spoofing attack. They rented a GPS simulator for $1000 and attached an amplifier and an antenna. When assembled, their spoofer's signal strength was under 1×10^{-10} W. Their tests involved manually breaking the receiver's lock on the real GPS signals with a metal wastebasket, and then keeping the spoofer near the receiver long enough for the receiver to lock on to the spoofed signal. It typically took the receiver about two minutes to lock on to the fake signal, after which spoofer could be moved up to about 30 feet away. They note that a higher powered transmitter would have enabled spoofing over greater distances [8].

In 2003, Warner and Johnston followed up on their work by proposing various spoofing countermeasures, some of which had also been discussed in the Volpe report. These countermeasures included monitoring absolute signal strength, relative signal strength, individual satellite signal strengths, the number of satellites present, and the time. The idea was to look for any sudden jumps or unrealistic values that might be indicative of spoofing. They also proposed attaching an accelerometer and a compass to the receiver to verify its relative motion [15].

Several years later in 2008, Humphreys et al. published a more thorough assessment of the spoofing threat. They first identified three levels of sophistication for spoofing attacks: simplistic, intermediate, and sophisticated. A simplistic spoofing attack consists of broadcasting the signal from a GPS simulator toward the target receiver, like Warner and Johnston did in 2002. This type of attack is easy to detect because the spoofed signal is not synchronized with the actual GPS signal. It would likely trigger many of the simple countermeasures proposed by using excessive power, causing the receiver to lose lock, and causing a jump in the receiver's outputs. An intermediate spoofing attack requires knowledge of the victim's position and velocity so that the spoofing signals can be aligned with the actual GPS signals. Once aligned, the signal strength is gradually increased until the GPS receiver's lock can be smoothly pulled away from the true signals. This method of attack could defeat all the previously proposed countermeasures that do not involve altering the GPS system, except

for using multiple antennas to distinguish the angles of arrival. A sophisticated spoofing attack involves individually tailoring a fake signal for each antenna of a device that uses angle of arrival discrimination. This would likely require the spoofer to have physical access to each antenna. Cryptographic authentication is the only proposed countermeasure that can currently guard against a sophisticated spoofing attack. The authors of the paper successfully carried out an intermediate level spoofing attack using a receiver hard-wired to a spoofer. They propose two new countermeasures that involve monitoring the latency of data bits in the GPS signal and checking for the vestigial true signal that remains when the spoofer starts pulling the receiver's lock away. These countermeasures increase security, but could still be overcome by an attacker that could precisely align the fake signal with the real one and that knows the position of the receiver well enough to suppress the true signal at that location [17].

In 2009, Xi-jun et al. analyzed techniques of forging GPS signals [47] and proposed modifying the GPS system to embed encrypted messages for authentication [48]. The next year, Cavaleri et al. proposed a signal quality monitoring approach to detecting GPS spoofing in much the same way that other interferences like multipath are detected [49]. Humphreys and Bhatti also prototyped a method to upgrade existing GPS receivers to harden them against spoofing by fusing the information from various other sensors, like an inertial navigation system and a frequency reference [50].

In 2011, Shepard and Humphreys characterized the response of four different receivers used in different applications to spoofing. They demonstrated that spoofing could be used to disrupt cell phone service or cause problems in the power grid due to the reliance of these utilities on GPS timing. They also showed that the spoofing signal only needs to be approximately 1.1 times as powerful as the actual GPS signal to consistently capture a receiver's lock. Since this ratio falls within the natural variations of the GPS signal power, such a spoofing attack would not be detected by many of the previously proposed countermeasures [11]. Tippenhauer et al. also explored the requirements to spoof multiple GPS receivers simultaneously from afar [16]. Wesson et al. proposed another modification to the GPS signal to introduce cryptographic authentication, but noted that such a change is unlikely in the next decade [51]. Humphreys published a white paper warning about the

potential for GPS spoofing to be used to game financial markets by altering the timestamps for trades [12]. Later in the year, Iran captured a US RQ-170 Sentinel drone and flaunted it in their media. An Iranian engineer reportedly claimed that they spoofed the drone's GPS and got it to land as if it were back at its home base [44, 45].

In 2012, Shepard et al. evaluated the vulnerability of the power grid to GPS spoofing and determined that it could possibly cause a large scale blackout [10]. Jahromi et al. proposed a modification of GPS receiver circuitry that would allow it to measure the absolute power of correlation peaks and potentially detect spoofing [52]. Also, Adaptive Flight, Inc. worked with Humphreys' Radionavigation Laboratory to determine if an unmanned aerial vehicle (UAV) could be led off course via GPS spoofing. In a test with representatives from the Federal Aviation Administration and the Department of Homeland Security present, they successfully altered the behavior of an Adaptive Flight Hornet Mini UAV [13, 14]. This test led to Humphreys testifying before Congress about the vulnerabilities of the GPS system and the need for anti-spoofing technology going forward [53].

In this work, we investigate the use of a video camera to detect GPS spoofing. The idea is to run an optical flow algorithm on the video from the camera and compare its output against an expected optical flow field calculated using the motion field equations presented in Chapter 3. To our knowledge, video cameras have not yet been used to detect spoofing.

5.3 Problem Statement

This work explores a novel approach for a UAS to detect spoofed GPS readings using an electro-optical camera, an inertial measurement unit (IMU), and a digital elevation map. The method developed could also detect faulty GPS readings, but we focus on the spoofing scenario. The IMU's measurements of how the UAS is moving coupled with the terrain elevation map allow us to calculate a motion field for the points in its field of view. An optical flow algorithm can also be run on the video to obtain a field representing the actual motion of features through the image plane. Our spoofing detector compares the optical flow against the motion field and signals spoofing if the difference is too great.

The algorithm has the potential to detect a spoofing attack or fault under two different scenarios. The first is where the false GPS readings throw off the UAS's estimates of its

movement enough that the motion field does not agree with what the camera is seeing. Previously proposed countermeasures, like attaching an IMU to a GPS receiver, could also detect this type of spoofing. A sophisticated attacker can thwart this detection by making small, incremental changes to the GPS readings so that the detection threshold is not exceeded. Over time, the changes could accumulate to introduce significant error in the UAS's position estimates. Although this type of spoofing detection can be overcome, it at least makes it more difficult for an attacker to succeed because he must have some knowledge of the position and velocity of the receiver.

The second scenario is where a spoofer has already successfully taken control of the UAS's position estimate, but the terrain between the spoofed and actual positions becomes different enough to trigger the alarm. This of course depends on variation in the terrain and would not work in a large area of uniformly flat terrain. Given enough variance in the terrain, the algorithm will detect the spoofing regardless of how gradually the attacker pulls the state estimates away from the truth. Figure 5.2 illustrates how a difference in terrain height between a spoofed and actual location can cause differences in the motion field and optical flow.

Noise in the sensor measurements and optical flow can also cause some mismatch between the fields. However, these effects can be mitigated through low-pass filtering. Outliers in the optical flow field caused by bad feature correspondences can also be detected and discarded using a number of different algorithms, such as RANSAC.

The method we have developed to detect GPS spoofing will not work in all situations. For instance, if the UAS flies high enough or slow enough, differences in the flow caused by the shape of the terrain will not be discernible. Also, if optical flow cannot be calculated reliably, the method will fail. This could happen over featureless terrain like snow or over bodies of water.

5.4 Flight Simulator

The spoofing detection algorithm was developed in a flight simulator written using Matlab and Simulink. The flight simulator was originally developed using the methods and dynamics presented in *Small Unmanned Aircraft: Theory and Practice* [1]. The program

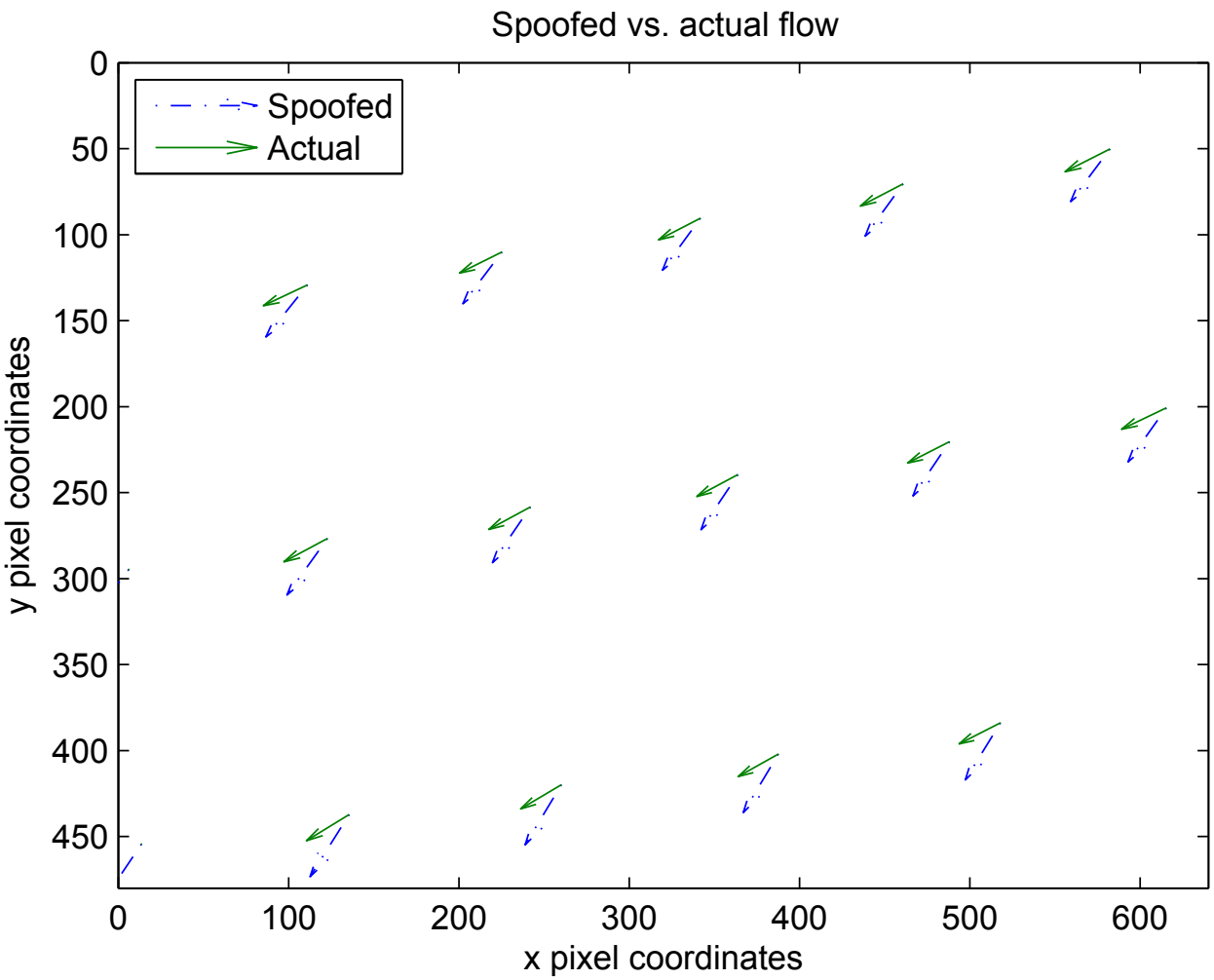

Figure 5.2: The motion field calculated from a spoofed location compared with the optical flow calculated from the UAS's actual states. The UAS was traveling at about 10 m/s and only its position estimate was tampered with. The terrain in the spoofed location was about 30 m below the UAS, while the terrain in the actual location was about 100 m below the UAS. A large difference in terrain height was used to make the difference in the flows clearly visible. The leftward component of the flow was caused by rolling motion of the aircraft. The diagram represents the flow over one tenth of a second.

simulates noisy sensor measurements from accelerometers, gyros, pressure sensors, and GPS. An extended Kalman filter (EKF) and several low-pass filters are used to to estimate the states of the aircraft based on the sensor measurements and dynamics.

We added two different places where values in the simulator can be tampered with to simulate spoofing. The first is a function that can modify the simulated GPS measurements before they are sent to the EKF. This allows us to test the first scenario where the spoofing

throws off the estimated states enough that the motion field does not agree with the optical flow. We can experiment with the amount of change that can be made to the velocity of the aircraft to determine what is and is not detectible.

To test the scenario where differences in the terrain become large enough to trigger the spoofing detector, we simply add an offset to the UAS's true location when using it to simulate optical flow. Since this does not disturb anything in the system other than calculating the optical flow field for a different location, the only way it can be detected is by noticing differences in the flow caused by the different terrain. This allows us to experiment with how different the terrain must be before the spoofing is detected.

5.5 Detecting Spoofing

To detect spoofing, we first calculate the motion field and optical flow. We then interpolate one field to the other for comparison and calculate an average innovation vector for the video frame. These innovation vectors form a signal that we low-pass filter and compare against a threshold to detect spoofing. These steps are detailed in the following sections.

5.5.1 Calculating the Motion Field

We calculate the motion field, or expected flow, of the terrain data as

$$\begin{bmatrix} \hat{m}_{\mathrm{x}i} \\ \hat{m}_{\mathrm{y}i} \end{bmatrix} = f_{xy}(\hat{\mathbf{x}}, \mathbf{t}_i) \qquad \text{for } i \in \{1, 2, \ldots, N_{\hat{m}}\} \tag{5.1}$$

and

$$\begin{bmatrix} \hat{m}_{\mathrm{u}i} \\ \hat{m}_{\mathrm{v}i} \end{bmatrix} = f_{uv}(\hat{\mathbf{x}}, \mathbf{t}_i) \qquad \text{for } i \in \{1, 2, \ldots, N_{\hat{m}}\}, \tag{5.2}$$

where $(\hat{m}_{\mathrm{x}i}, \hat{m}_{\mathrm{y}i})^\top$ is the location of the flow vector for the ith terrain point in view from the estimated location, $(\hat{m}_{\mathrm{u}i}, \hat{m}_{\mathrm{v}i})^\top$ is the corresponding flow vector, $N_{\hat{m}}$ is the number of terrain points in view from the estimated location, and the functions $f_{xy}(\cdot, \cdot)$ and $f_{uv}(\cdot, \cdot)$

were previously defined in Equations (3.23) and (3.25). We use the estimated states $\hat{\mathbf{x}}$ from the EKF because they are representative of the best estimate we would have of the aircraft's states in real life, where the true states are not known. Let

$$\hat{\mathbf{m}}_{\mathrm{x}} = (\hat{m}_{\mathrm{x}1}, \ldots, \hat{m}_{\mathrm{x}N_{\hat{m}}})^\top, \tag{5.3}$$

$$\hat{\mathbf{m}}_{\mathrm{y}} = (\hat{m}_{\mathrm{y}1}, \ldots, \hat{m}_{\mathrm{y}N_{\hat{m}}})^\top, \tag{5.4}$$

$$\hat{\mathbf{m}}_{\mathrm{u}} = (\hat{m}_{\mathrm{u}1}, \ldots, \hat{m}_{\mathrm{u}N_{\hat{m}}})^\top, \tag{5.5}$$

and

$$\hat{\mathbf{m}}_{\mathrm{v}} = (\hat{m}_{\mathrm{v}1}, \ldots, \hat{m}_{\mathrm{v}N_{\hat{m}}})^\top. \tag{5.6}$$

5.5.2 Simulating Optical Flow

Since the simulator does not render the camera's view, we do not actually have any video to run an optical flow algorithm on. However, if a camera is rigidly attached to a UAS and pointed toward static objects like terrain, then all of the optical flow seen by the camera is caused by the motion of the UAS. In simulation, we have access to the true states of the aircraft, which describe exactly how it is moving. Thus, we can simulate optical flow by using true states to calculate the motion field. It should be noted that this calculation gives results that are exact to within the accuracy of the data types used and the computations performed on them. In reality, an optical flow algorithm may not yield such sub-pixel accuracy. Because of this, we round the location of each calculated flow vector to the nearest pixel and its u and v flow components to the nearest pixel per frame. Thus, the simulated optical flow is given by

$$\begin{bmatrix} m_{\mathrm{x}i} \\ m_{\mathrm{y}i} \end{bmatrix} = round\big(f_{xy}(\mathbf{x}, \mathbf{t}_i)\big) \qquad \text{for } i \in \{1, 2, \ldots, N_m\} \tag{5.7}$$

and

$$\begin{bmatrix} m_{\mathrm{u}i} \\ m_{\mathrm{v}i} \end{bmatrix} = round\big(f_{uv}(\mathbf{x}, \mathbf{t}_i)\big) \qquad \text{for } i \in \{1, 2, \ldots, N_m\}, \tag{5.8}$$

where $(m_{\mathrm{x}i}, m_{\mathrm{y}i})^\top$ is the location of the flow vector for the ith terrain point in view from the true location, $(m_{\mathrm{u}i}, m_{\mathrm{v}i})^\top$ is the corresponding flow vector, and N_m is the number of terrain points in view from the true location. Let

$$\mathbf{m}_{\mathrm{x}} = (m_{\mathrm{x}1}, \ldots, m_{\mathrm{x}N_m})^\top, \tag{5.9}$$

$$\mathbf{m}_{\mathrm{y}} = (m_{\mathrm{y}1}, \ldots, m_{\mathrm{y}N_m})^\top, \tag{5.10}$$

$$\mathbf{m}_{\mathrm{u}} = (m_{\mathrm{u}1}, \ldots, m_{\mathrm{u}N_m})^\top, \tag{5.11}$$

and

$$\mathbf{m}_{\mathrm{v}} = (m_{\mathrm{v}1}, \ldots, m_{\mathrm{v}N_m})^\top. \tag{5.12}$$

We also experimented with adding Gaussian noise with a standard deviation of 0.5 pixels to the optical flow instead of rounding. However, this change did not cause any noticeable difference in the performance of the detector. All the results presented in this work used the rounding approach.

5.5.3 Comparing the Motion Field and Optical Flow

To detect spoofing, we need a way to compare an optical flow field to a motion field. The first problem that arises is that the features tracked by the optical flow algorithm are not located at the same places as the motion field vectors. Thus, it is necessary to interpolate the flow data from one of the fields to the locations of the data points of the other field. We chose to interpolate the optical flow data to the locations of motion field data points. This choice helps ensure that data is compared at places where terrain should be located. If the camera is angled up far enough to see part of the sky, then an optical flow algorithm might

find and track features in a cloud. By only making comparisons at places where terrain should be located, we minimize the chances that such situations will throw off the algorithm.

To perform the interpolation, we use Matlab's `griddata` function, which conveniently takes a two-dimensional field of data and linearly interpolates it to another set of two-dimensional locations. Since the flow vectors at each point are two dimensional, we must call this function once for each dimension. So, we have

$$\mathbf{m}_{\hat{\mathrm{u}}} = \texttt{griddata}(\mathbf{m}_{\mathrm{x}}, \mathbf{m}_{\mathrm{y}}, \mathbf{m}_{\mathrm{u}}, \hat{\mathbf{m}}_{\mathrm{x}}, \hat{\mathbf{m}}_{\mathrm{y}}) \tag{5.13}$$

and

$$\mathbf{m}_{\hat{\mathrm{v}}} = \texttt{griddata}(\mathbf{m}_{\mathrm{x}}, \mathbf{m}_{\mathrm{y}}, \mathbf{m}_{\mathrm{v}}, \hat{\mathbf{m}}_{\mathrm{x}}, \hat{\mathbf{m}}_{\mathrm{y}}), \tag{5.14}$$

where $\mathbf{m}_{\hat{\mathrm{u}}}$ and $\mathbf{m}_{\hat{\mathrm{v}}}$ are optical flow vectors interpolated to the motion field points. The `griddata` function returns `NaN`s for any requested points that fall outside the convex hull of the data to be interpreted. We simply discard these points since they do not have data available from both fields to make a comparison. Out of convenience, we assume in the rest of the chapter that the corresponding elements of $\mathbf{m}_{\hat{\mathrm{u}}}$, $\mathbf{m}_{\hat{\mathrm{v}}}$, $\hat{\mathbf{m}}_{\mathrm{u}}$, and $\hat{\mathbf{m}}_{\mathrm{y}}$ have been discarded if a `NaN` is present. We define N to be the number of elements in one of these vectors, noting that they are now the same size.

With both optical flow and motion field vectors located at the same points in the image, we can now calculate an innovation vector for each frame. We do this by taking the average difference between the measured optical flow and the estimated motion field,

$$\tilde{m}_{\mathrm{u}j} = \frac{1}{N}\sum_{i=1}^{N}(m_{\hat{\mathrm{u}}i} - \hat{m}_{\mathrm{u}i}) \tag{5.15}$$

and

$$\tilde{m}_{\mathrm{v}j} = \frac{1}{N}\sum_{i=1}^{N}(m_{\hat{\mathrm{v}}i} - \hat{m}_{\mathrm{v}i}), \tag{5.16}$$

where $(\tilde{m}_{\mathrm{u}j}, \tilde{m}_{\mathrm{v}j})^\top$ is the innovation vector for the jth frame. Over time, the innovation vectors from each frame form an innovation signal that should be zero mean if spoofing has not occurred.

5.5.4 Detecting Spoofing

To detect spoofing, we look at the low frequency components of the innovation signal. If an attacker wants to divert the UAS toward a certain location, he must consistently bias the spoofed GPS readings in the opposite direction. This will cause the UAS to mistakenly correct its course in the desired direction. This consistent bias will show up as a low frequency component in the innovation signal. If the UAS has already been led off course, then differences in the height of the terrain between the spoofed and actual location will also show up as a low frequency component of the innovation signal. If the actual terrain below the UAS is higher than the terrain at the spoofed location, then sequence of innovation vectors will consistently have a larger downward component than expected.

We low-pass filter the innovation signal by taking a windowed average over time. This suppresses the high-frequency variations caused by noise in the state estimates and in the optical flow. The algorithm then signals that spoofing has occurred if the magnitude of the filtered innovation signal exceeds a predetermined threshold. The magnitude of the low-pass filtered innovation signal is

$$|\tilde{m}_{\mathrm{w}}| = \frac{1}{W}\sqrt{\left(\sum_{i=j-W+1}^{j} \tilde{m}_{\mathrm{u}i}\right)^2 + \left(\sum_{i=j-W+1}^{j} \tilde{m}_{\mathrm{v}i}\right)^2}, \tag{5.17}$$

where j is the index of the current frame and W is the number of frames in the window. We used a window size of 20 seconds, or 300 frames. The detector signals spoofing if $|\tilde{m}_{\mathrm{w}}| > m_{\max}$, where $m_{\max}$ is the predetermined threshold.

The threshold $m_{\max}$ can be chosen empirically by gathering flight data that is known to not be spoofed. It should be chosen to be larger than the magnitude of most or all of the vectors in the low-pass filtered innovation signal from the data. Smaller values for the threshold will make the detector more sensitive to spoofing, but will also trigger more false positives.

The threshold can also be chosen based on the amount of difference in the terrain height that should trigger spoofing. Assuming straight and level flight with a nadir oriented camera, the threshold can be calculated as the expected difference in flow between the two terrain heights. This is given by

$$m_{\max} = Tfu\left(\frac{1}{d_{\mathrm{y}}h} - \frac{1}{d_{\mathrm{y}}(h+d)}\right), \tag{5.18}$$

where T is the frame period of the camera, f is the focal length, u is the speed of the aircraft, d_{y} is the y dimension of a single pixel on the camera sensor, h is the expected height of the aircraft, and d is the difference in terrain height that should trigger the detector. However, this method does not give any intuition on what the resulting false positive rate will be.

Ideally, it would be possible to choose a threshold that provides the desired balance between the false positive rate and the true positive rate. However, the true positive rate depends on the characteristics of the spoofing attack and the differences in the terrain between the spoofed and actual locations. These are typically not known, so we can only tune our threshold based on the observed false positive rate.

5.6 Results

We simulated the GPS spoofing detector in several different situations. First, we evaluate its ability to detect corrupted state estimates by adding a constant bias to the ground speed reported by the GPS receiver. Then, we test its sensitivity to differences in terrain height by modifying the estimated height of the UAS over flat ground. Finally, we test the ability of the algorithm to detect differences in the terrain for randomly generated flight paths over randomly generated terrain.

For our first tests, we commanded the UAS to fly at 10 m/s along a straight and level path 100 m above perfectly flat ground. We corrupted the ground speed reported by the GPS receiver by adding a constant bias. This biased measurement was then passed in to the EKF that estimates the states of the UAS. Our threshold was set to $m_{\max} = 0.248$, which was empirically chosen to be small, but without causing many false positives. This threshold corresponds to a 4.3 m difference in the height of the terrain in this scenario. We also ran

Table 5.1: True positives vs. the amount of velocity spoofing.

Modification to GPS velocity (m/s)	True positives (out of 1502 frames) UAS height = 100 m	True positives (out of 1502 frames) UAS height = 300 m
0.05	0	0
0.10	0	20
0.15	133	22
0.20	99	1
0.25	554	1
0.30	763	7
0.35	624	0
0.40	1024	2
0.45	885	123
0.50	1206	92
0.55	988	166
0.60	1223	27
0.65	1188	11
0.70	1170	334
0.75	1219	374
0.80	1225	225
0.85	1222	568
0.90	1250	433
0.95	1230	544
1.00	1278	852

similar simulations but with the UAS at 300 m above the terrain. Table 5.1 enumerates the number of video frames for which the spoofing detector was triggered for each bias in velocity. The data for each simulation was collected over 1502 video frames, or about 100 s at a frame rate of 15 Hz. For a height of 100 m, the detector starts being triggered fairly consistently when the bias on the velocity is greater than or equal to 0.25 m/s. In this case, the detector works quite well as it can be triggered by a bias that is just one fortieth of the typical velocity of the aircraft. At higher altitudes, the effects of the velocity of the UAS on the optical flow are lessened and only larger biases can be detected.

To get a clear picture of how differences in the height of the terrain affect the detector, we again commanded a straight and level path over flat terrain. For each flight, we ran the detection algorithm on both the uncorrupted state estimates and a spoofed state estimate. The spoofed estimate was created by directly changing the estimated location of the plane to

Table 5.2: True positives detected over 1502 frames.

Height difference (%) \ Height (m):	75	100	150	200	300	400
5%	1063	351	5	7	0	3
10%	1502	1502	1498	632	180	12
20%	1502	1502	1502	1502	1352	524
40%	1502	1502	1502	1502	1502	1494

be above flat terrain at a different height. Since no other state estimates were changed, the differences in terrain height were the only factor that could cause the detector to be triggered more than normal. We varied the height that the UAS flew above the terrain as well as the percentage difference in the height between the actual and spoofed locations. Table 5.2 shows the number of times the spoofing detector was triggered under spoofed conditions. For example, the detector was triggered 351 times when the UAS was 100 m above the ground, but had been spoofed to think it was 105 m above the ground. Since each simulation lasted for 1502 frames, a perfect detector would have been triggered 1502 times for every spot in Table 5.2. The data show that the algorithm grows less effective as the height of the UAS increases. This is because the effect of the difference in terrain height on the optical flow varies inversely proportional to the distance of the UAS from the terrain. Increasing the speed of the UAS as it flies higher could help counter this effect. Although this effect degrades performance at high altitudes, it could actually help against a spoofer trying to force the UAS to land. As the UAS gets closer to the terrain, the algorithm would become more likely to detect even small discrepancies in the terrain height.

Table 5.3 shows the number of times the detector was falsely triggered under normal conditions for the same flights as are listed in Table 5.2. Note that the spoofed height difference has no effect on the data in Table 5.3 because the uncorrupted state estimates were used by the detector. Across all 20 flights, false positives were only triggered for a total of 77 out of 36048 frames, or 0.21% of the time. Also, the false positive rate does not seem to be correlated with the height of the UAS above the terrain. This indicates that the same threshold will yield similar false positive rates over a broad range of different altitudes. Thus, new thresholds do not need to be chosen to maintain the same false positive rate when the altitude changes.

Table 5.3: False positives detected over 1502 frames.

Height difference (%) \ Height (m):	75	100	150	200	300	400
5%	5	0	1	0	0	5
10%	14	0	9	6	1	4
20%	0	0	0	5	0	0
40%	13	1	0	0	1	12

A sophisticated spoofer could lessen the chances that the terrain height would trigger the detector by adjusting the GPS height to get the UAS to fly at the same height above the actual terrain as it thinks it is above the terrain at the spoofed location. However, the slope of the terrain could require the spoofer to make changes in the altitude too rapidly to be undetected. Also, different shapes of terrain will still cause discrepancies in the flow fields. Finally, this type of spoofing can easily be countered with a static pressure sensor from which altitude can be estimated. UASs are often already equipped with these sensors.

To evaluate how the GPS spoofing detector would perform in actual flights, we ran simulations with randomly generated flight paths over randomly generated terrain. Two examples of these randomly generated flight paths and terrain can be seen in Figure 5.3. To generate the random terrain, we repeatedly picked random lines across an originally flat map and shifted everything on one side of the line up or down. This algorithm to generate random terrain is known as the fault algorithm. After generating the terrain, we scaled it into a set height range.

We found that we got higher numbers of false positives when we used the same threshold with the randomly generated paths as we did with the straight and level paths from before. The increased amount of angular motion required to make the frequent turns in the randomly generated paths introduces more noise into the state estimates. The increased noise slightly increases the magnitude of the low-pass filtered innovation, thus triggering more false positives. We picked a new threshold of $m_{\max} = .341$ for the randomly generated paths. This threshold corresponds to a terrain height difference of 6 m when flying 100 m above the ground at 10 m/s.

For the simulations with random paths and terrain, we ran our detector on both normal state estimates and spoofed state estimates. The spoofing was done by subtracting

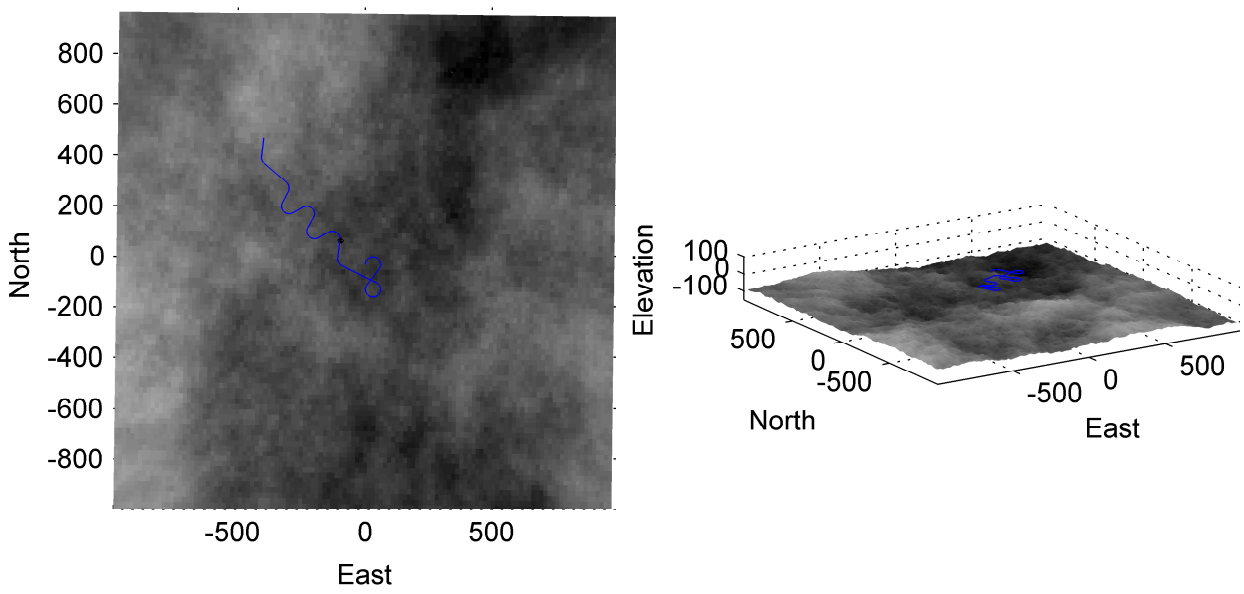

Figure 5.3: Two examples of a randomly generated flight path over randomly generated terrain. One is shown from the top, while the other is shown from the side. The random paths were generated using a random walk with Dubins paths between the waypoints. The random terrain was generated using the fault algorithm. Lighter colored terrain indicates higher elevation.

400 m from the estimated east coordinate of the UAS. No other estimates were modified, so the detector had to rely on terrain differences to detect spoofing. Table 5.4 lists the number of times the detector was correctly triggered for the spoofed state estimates and falsely triggered for the normal state estimates over 20 flights. The terrain varied in height between 60 and 160 m below the height of the UAS. Across all 20 flights, the detector was correctly triggered 76.25% of the time when spoofing was present. It was falsely triggered only 0.78% of the time when there was no spoofing. Flight 12 and flight 13 have exceptionally low true positive counts in Table 5.4. This is likely because the flight paths stayed in a relatively small area and the terrain in that area and the corresponding spoofed area was similar in height. Overall, the detector performed quite well in this situation.

We performed similar simulations with terrain that varied in height between 150 and 250 m below the height of the UAS. The results are given in Table 5.5. As we saw before, the increased height does not have a strong effect on the false positive rate, but degrades the true positive rate when spoofing is present. The frequent turns in the flight path similarly degrade the true positive rate because the camera views the terrain at an angle during the turns.

Table 5.4: True and false positives for random flight paths above random terrain between 60 and 160 meters below.

Flight	1	2	3	4	5	6	7	8	9	10	11	12
TP	1149	1434	1248	874	1502	1502	1502	1053	1502	1502	1502	52
FP	22	7	4	21	27	2	2	7	18	7	0	5

Flight	13	14	15	16	17	18	19	20	Average	% of all frames
TP	198	1370	1423	1350	845	1184	937	776	1145.3	76.25%
FP	7	7	11	42	14	13	3	16	11.75	0.78%

Table 5.5: True and false positives for random flight paths above random terrain between 150 and 250 meters below.

Flight number	1	2	3	4	5	6	7	8	9	10	11	12	13
True Positives	104	15	359	35	90	470	31	5	15	11	1	689	1265
False Positives	22	7	4	21	27	2	2	7	18	7	0	5	7

Flight number	14	15	16	17	18	19	20	Average	% of all frames
True Positives	1	0	868	721	17	9	405	255.6	17.01%
False Positives	7	11	42	14	13	3	16	5.5	0.37%

This effectively increases the distance between the camera and the terrain, thus decreasing the effects of terrain differences on the optical flow. Despite this degradation in performance, the detector still performed well for a few of the flights in Table 5.5.

5.7 Conclusion

Although the GPS spoofing detection algorithm we developed does not work in all situations, it has the potential to detect spoofing in some situations where no other proposed countermeasure would be effective. Our simulations indicate that it could be effective in the types of scenarios typically encountered by small UASs. To provide the best protection against spoofing, the detection algorithm should be used in conjunction with several of the other spoofing countermeasures that have been proposed.

Chapter 6

Conclusion

6.1 The VTTP Metric

Automating path planning can reduce the workload of UAS operators. However, to perform well in a given application, path planning algorithms need some kind of metric with which they can predict and evaluate their performance. Although some basic metrics like coverage have been used in surveillance, reconnaissance, and other similar scenarios, there have not previously been any that estimated the probability of success in the DRI task. Furthermore, improving performance in surveillance, reconnaissance, and search and rescue missions can ultimately result in saved lives.

6.1.1 Contributions

VTTP is the first metric designed to estimate the probability of DRI for aerial video. It extends the TTP metric to account for factors like motion blur and the limited amount of time that an observer has to view a location before it passes out of the video frame. VTTP also includes methods to estimate parameters of the TTP metric that are typically unknown in surveillance and reconnaissance scenarios.

The VTTP metric was designed specifically to be used for both path planning and evaluation of actual performance during or after flight. The ability to calculate VTTP in both situations allows the loop to be closed around the path planner so that it can adjust future paths based on actual performance. For instance, it could revisit a high-priority area that was not seen well because the UAS was blown of course or because the video was blurry.

We also provided a preliminary validation of the VTTP metric. A small user study verified that VTTP values are correlated with subjective human evaluation of the quality of the video for DRI tasks. Although achieved VTTP tends to be lower than predicted VTTP

due to deviations from the planned flight path, the data shows that achieved VTTP follows predicted VTTP fairly closely. Thus, by maximizing the predicted VTTP over the planned path, we increase the achieved VTTP and improve the observer's ability to perform the DRI task.

6.1.2 Future Work

There are many improvements that could be made to the VTTP metric in future work. As we discovered after performing a user study, the threshold used in computing the motion component of the metric should vary according to the size of the object in the image plane. The user study also highlighted how clutter in the scene can increase the difficulty of DRI. Various image clutter metrics have been proposed [54, 55, 56, 57] that could potentially be used to account for these effects in the VTTP metric.

As mentioned in Section 4.3.1, the actual lighting of areas in the video could potentially be estimated if the exposure of the camera can be accounted for. These estimates could then be used to update the lighting map based on actual observations.

Performing georegistration on the video frames similar to Morse et al. [6] could yield significant improvements in accuracy when determining what terrain points are in the field of view and where they are located. Information gleaned from the georegistration could also be fed back into the EKF to improve the UAS state estimates.

Finally, the VTTP metric could benefit from a more thorough validation based on a larger user study involving trained personnel. An in-depth study could validate the metric as a whole as well as validating its individual components. Such a study could also provide better insights on how the various parameters used throughout the metric should be tuned.

6.2 Detecting GPS Spoofing

Millions of people and significant parts of the infrastructure rely on the civilian band GPS system every day. Although the vulnerability of the civilian band to spoofing attacks has been known for many years, very little has been done to harden receivers against this kind of attack. In 2011, Shepard and Humphreys sought out four different types of receivers, including ones used in infrastructure, and spoofed them all [11]. Although few significant

spoofing attacks have been reported so far, their potential to cause damage and loss of life is great.

6.2.1 Contributions

The GPS spoofing detector we developed for UASs is the first to use a camera and an elevation map to detect spoofing. Unlike other countermeasures, the spoofing detector is limited to use on aircraft. However, it has the potential to detect any spoofing attack, regardless of speed or sophistication, if there is enough variation in the terrain. We are unaware of any other proposed countermeasures that have potential to defeat a slow, sophisticated attack as described by Humphreys et al. [17].

Even without variation in the terrain, the spoofing detector can detect less sophisticated attacks that alter the estimated states of the UAS too abruptly. This makes an attacker's job more difficult because he must know the location and velocity of the aircraft well enough to produce a spoofing signal that is at least somewhat close to the true states. Furthermore, he must maintain this knowledge over the duration of the spoofing attack.

The detection algorithm is particularly suited for UASs because it relies on sensors that most UASs are already equipped with. The computational requirements are also within reach of UAS hardware, particularly if a GPU is on board.

6.2.2 Future Work

Due to time constraints and the limitations of the hardware available to us, we were unable to implement and test the spoofing detection algorithm on hardware. Although the simulation results look promising, a working hardware demonstration would provide a much stronger validation of the algorithm. Also, more extensive experimentation could be performed to determine what the best length for the averaging window is.

Our spoofing detection algorithm currently requires the user to set an empirically determined threshold. Ideally, this threshold could be determined probabilistically. For example, the covariances on the state estimates could be propagated through the motion field equations and added to the covariance of the optical flow to estimate the covariance of the innovation signal. We attempted to do this by numerically calculating the Jacobian of

the motion field and using it to propagate the covariances. However, we were unable to get satisfactory results in the time available. Our attempt yielded a threshold that was too low and caused constant false positives. This could be due to inaccuracies in the covariances of the state estimates, a bad propagation model, the non-linearities of the motion field equations, or various other causes. Additional work would be needed to explore the possibility of setting a threshold in this manner.

Future work could also explore the possibility of using a visual odometry algorithm to detect GPS spoofing. The computational requirements would be higher for a visual odometry algorithm than they are for optical flow and motion fields, but the visual odometry could also be used to improve the state estimates of the UAS. However, flying outdoors and far above the terrain may also make visual odometry less feasible.

Bibliography

[1] R. W. Beard and T. W. McLain, *Small Unmanned Aircraft: Theory and Practice*. Princeton University Press, 2012. iii, 6, 14, 16, 34, 52

[2] 112th Congress, "FAA modernization and reform act," U.S. Government Printing Office, Feb. 2012. 1

[3] A. Madrigal, "Inside the drone missions to fukushima," The Atlantic news article, Apr. 2011. 1

[4] C. Bolkcom, "Homeland security: Unmanned aerial vehicles and border surveillance," Library of Congress Congressional Research Service, CRS Report for Congress, Dec. 2004. 1, 27

[5] F. Rafi, S. Khan, K. Shafiq, and M. Shah, "Autonomous target following by unmanned aerial vehicles," in *Proceedings of SPIE Defence and Security Symposium*, 2006, pp. 6230–6242. 1, 26, 27

[6] B. S. Morse, C. H. Engh, and M. A. Goodrich, "UAV video coverage quality maps and prioritized indexing for wilderness search and rescue," in *Proceedings of the 5th ACM/IEEE international conference on Human-robot interaction*. Piscataway, NJ, USA: IEEE Press, 2010, pp. 227–234. 2, 27, 28, 66

[7] R. H. Vollmerhausen and E. Jacobs, "The targeting task performance (TTP) metric: A new model for predicting target acquisition performance," U.S. Army CERDEC, Tech. Rep. AMSEL-NV-TR-230, Apr. 2004. 2, 27, 33

[8] J. S. Warner and R. G. Johnston, "A simple demonstration that the global positioning system (GPS) is vulnerable to spoofing," *The Journal of Security Administration*, vol. 25, pp. 19–28, 2002. 3, 49

[9] John A. Volpe National Transportation Systems Center, "Vulnerability assessment of the transportation infrastructure relying on the global positioning system," U.S. Department of Transportation, Tech. Rep., Aug. 2001. 4, 45, 49

[10] D. P. Shepard, T. E. Humphreys, and A. A. Fansler, "Evaluation of the vulnerability of phasor measurement units to GPS spoofing attacks," in *Sixth Annual IFIP WG 11.10 International Conference on Critical Infrastructure Protection*, 2012. 4, 51

[11] D. P. Shepard and T. E. Humphreys, "Characterization of receiver response to spoofing attacks," in *Proceedings of ION GNSS*, 2011. 4, 50, 66

[12] T. E. Humphreys, "GPS spoofing and the financial sector," The University of Texas at Austin Radionavigation Laboratory, White Paper, Jun. 2011. 4, 51

[13] Press, "Adaptive Flight works with University of Texas for GPS spoofing tests - announces GPS denied software," sUAS News article, Jul. 2012, retrieved 17 July 2012. 4, 51

[14] G. Mortimer, "University of Texas at Austin researchers demonstrate first successful spoofing of UAVs," sUAS News article, Jun. 2012. 4, 51

[15] J. S. Warner and R. G. Johnston, "GPS spoofing countermeasures," Los Alamos National Lab, Tech. Rep. LA-UR-03-6163, 2003. 4, 45, 49

[16] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, "On the requirements for successful GPS spoofing attacks," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 75–86. 4, 45, 50

[17] T. E. Humphreys, B. M. Ledvina, M. L. Psiaki, B. W. O'Hanlon, and P. M. Kintner, Jr., "Assessing the spoofing threat: Development of a portable GPS civilian spoofer," in *Proceedings of the 21st International Technical Meeting of the Satellite Division of The Institute of Navigation*, Savannah, GA, Sep. 2008, pp. 2314–2325. 4, 50, 67

[18] J. W. Hager, J. F. Behensky, and B. W. Drew, *The Universal Grids: Universal Transverse Mercator (UTM) and Universal Polar Stereographic (UPS)*, Defense Mapping Agency, Fairfax, VA, Sep. 1989. 7

[19] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *International Journal of Computer Vision*, vol. 92, no. 1, pp. 1–31, 2011. 23

[20] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer, Sep. 2010. 23

[21] M. Kontitsis, K. P. Valavanis, and N. Tsourveloudis, "A UAV vision system for airborne surveillance," in *Proceedings of the IEEE International Conference on Robotics and Automation*, vol. 1, 2004, pp. 77–83. 27

[22] R. Stevens, A. Cordes, R. L. Nelson, F. A. Sadjadi, and J. R. Braegelmann, "Improved aerial surveillance using onboard image processing," SPIE Newsroom article, Oct. 2008. 27

[23] N. Ceccarelli, J. J. Enright, E. Frazzoli, S. J. Rasmussen, and C. J. Schumacher, "Micro uav path planning for reconnaissance in wind," in *Proceedings of the American Control Conference*, Jul. 2007, pp. 5310 –5315. 27

[24] J. Kim and Y. Kim, "Moving ground target tracking in dense obstacle areas using UAVs," in *Proceedings of IFAC World Congress*, 2008. 27

[25] M. Quigley, M. A. Goodrich, S. Griffiths, A. Eldredge, and R. W. Beard, "Target acquisition, localization, and surveillance using a fixed-wing mini-UAV and gimbaled camera," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Apr. 2005, pp. 2600–2605. 27

[26] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte, "Coordinated decentralized search for a lost target in a Bayesian world," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, Oct. 2003, pp. 48–53. 27

[27] T. J. Setnicka and K. Andrasko, *Wilderness Search and Rescue: A Complete Handbook*. Boston, MA: Appalachian Mountain Club, 1980. 27

[28] S. Hansen, T. McLain, and M. Goodrich, "Probabilistic searching using a small unmanned aerial vehicle," in *Proceedings of AIAA Infotech@Aerospace*, 2007. 27

[29] L. Lin and M. Goodrich, "UAV intelligent path planning for wilderness search and rescue," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2009, pp. 709 –714. 27

[30] M. A. Goodrich, B. S. Morse, C. Engh, J. L. Cooper, and J. A. Adams, "Towards using unmanned aerial vehicles (UAVs) in wilderness search and rescue: Lessons from field trials," *Interaction Studies*, vol. 10, no. 3, pp. 453–478, 2009. 27

[31] J. Johnson, "Analysis of image forming systems," *Image Intensifier Symposium*, vol. AD 220160, pp. 244–273, 1958. 27

[32] J. Johnson and W. R. Lawson, "Performance modeling methods and problems," in *Proceedings of IRIS Specialty Group on Imaging*. Michigan: InfraRed Analysis Service (IRIS), Jan. 1974, pp. 40–44. 27

[33] P. Niedfeldt, R. Beard, B. Morse, and S. Pledgie, "Integrated sensor guidance using probability of object identification," in *Proceedings of the American Control Conference*, Jul. 2010, pp. 788 –793. 27, 33, 34

[34] M. Pinson and S. Wolf, "A new standardized method for objectively measuring video quality," *IEEE Transactions on Broadcasting*, vol. 50, no. 3, pp. 312 – 322, Sep. 2004. 28

[35] C. J. B. Lambrecht and O. Verscheure, "Perceptual quality measure using a spatio-temporal model of the human visual system," in *Proceedings of the SPIE*, vol. 2668, Mar. 1996, pp. 450–461. 28

[36] Z. Wang, L. Lu, and A. C. Bovik, "Video quality assessment based on structural distortion measurement," *Signal Processing: Image Communication*, vol. 19, no. 2, pp. 121–132, Feb. 2004. 28

[37] Z. Wang and A. Bovik, "A universal image quality index," *IEEE Signal Processing Letters*, vol. 9, no. 3, pp. 81–84, Mar. 2002. 28

[38] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, Apr. 2004. 28

[39] N. Damera-Venkata, T. Kite, W. Geisler, B. Evans, and A. Bovik, "Image quality assessment based on a degradation model," *IEEE Transactions on Image Processing*, vol. 9, no. 4, pp. 636–650, Apr. 2000. 28

[40] G. J. Zissis, J. S. Accetta, and D. L. Shumaker, Eds., *The infrared and electro-optical systems handbook*. Infrared Information and Analysis Center, 1993. 33

[41] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 1994, pp. 593–600. 35

[42] J. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker: Description of the algorithm," Intel Corporation Microprocessor Research Labs, Tech. Rep., 1999. 35

[43] R. Likert, "A technique for the measurement of attitudes," *Archives of Psychology*, vol. 22, no. 140, pp. 1–55, 1932. 43

[44] S. Peterson, "Exclusive: Iran hijacked US drone, says Iranian engineer," The Christian Science Monitor news article, Dec. 2011, retrieved 12 July 2012. 46, 51

[45] Press TV, "Iran airs footage of downed US drone," Press TV news article, Press TV, Dec. 2011, retrieved 12 July 2012. 46, 51

[46] United States Air Force Navstar Global Positioning System Program Office and Navstar Seminars, *Navstar GPS user equipment introduction*, Sep. 1996. 46, 48

[47] C. Xi-jun, C. Ke-jin, X. Jiang-ning, and L. Bao, "Analysis on forgery patterns for GPS civil spoofing signals," in *Fourth International Conference on Computer Sciences and Convergence Information Technology*, Nov. 2009, pp. 353 –356. 50

[48] C. Xi-jun, X. Jiang-ning, C. Ke-jin, and W. Jie, "An authenticity verification scheme based on hidden messages for current civilian GPS signals," in *Fourth International Conference on Computer Sciences and Convergence Information Technology*, Nov. 2009, pp. 345 –352. 50

[49] A. Cavaleri, B. Motella, M. Pini, and M. Fantino, "Detection of spoofed gps signals at code and carrier tracking level," in *Fifth ESA Workshop on Satellite Navigation Technologies and European Workshop on GNSS Signals and Signal Processing*, Dec. 2010, pp. 1–6. 50

[50] T. E. Humphreys and J. A. Bhatti, "The GPS assimilator: a method for upgrading existing GPS user equipment to improve accuracy, robustness, and resistance to spoofing," in *Proceedings of the 23rd International Technical Meeting of The Satellite Division of the Institute of Navigation*, Sep. 2010, pp. 1942–1952. 50

[51] K. Wesson, M. Rothlisberger, and T. Humphreys, "Practical cryptographic civil GPS signal authentication," in *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation*, Sep. 2011, pp. 3335–3345. 50

[52] A. J. Jahromi, A. Broumandan, J. Nielsen, and G. Lachapelle, "GPS spoofer countermeasure effectiveness based on signal strength, noise power, and c/n_0 measurements," *International Journal of Satellite Communications and Networking*, vol. 30, no. 4, pp. 181–191, Jun. 2012. 51

[53] H. J. DeCiutiis, "UT professor's UAV hacking brings him before Congress," The Daily Texan news article, Jul. 2012. 51

[54] R. Rosenholtz, Y. Li, and L. Nakano, "Measuring visual clutter," *Journal of Vision*, vol. 7, no. 2, pp. 1–22, Aug. 2007, article 17. 66

[55] M. J. Bravo and H. Farid, "A scale invariant measure of clutter," *Journal of Vision*, vol. 8, no. 1, pp. 1–9, Jan. 2008, article 23. 66

[56] H. Chang and J. Zhang, "Detection probability and detection time using clutter metrics," *Infrared Physics and Technology*, vol. 51, no. 2, pp. 83–90, Oct. 2007. 66

[57] J. M. Henderson, M. Chanceaux, and T. J. Smith, "The influence of clutter on real-world scene search: Evidence from search efficiency and eye movements," *Journal of Vision*, vol. 9, no. 1, pp. 1–8, Jan. 2009, article 32. 66