2005-03-02

# Real-time Evaluation of Vision-based Navigation for Autonomous Landing of a Rotorcraft Unmanned Aerial Vehicle in a Non-cooperative Environment

Dale D. Rowley
*Brigham Young University - Provo*

# REAL-TIME EVALUATION OF VISION-BASED NAVIGATION FOR AUTONOMOUS LANDING OF A ROTORCRAFT UNMANNED AERIAL VEHICLE IN A NON-COOPERATIVE ENVIRONMENT

by

Dale D. Rowley

A thesis submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

Brigham Young University

April 2005

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Dale D. Rowley

This thesis has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

| | |
|---|---|
| Date | Timothy W. McLain, Chair |
| Date | Randal W. Beard |
| Date | W. Jerry Bowman |

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the thesis of Dale D. Rowley in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Timothy W. McLain
Chair, Graduate Committee

Accepted for the Department

Matthew Jones
Graduate Coordinator

Accepted for the College

Douglas M. Chabries
Dean, Ira A. Fulton College of
Engineering and Technology

ABSTRACT

REAL-TIME EVALUATION OF VISION-BASED NAVIGATION FOR AUTONOMOUS LANDING OF A ROTORCRAFT UNMANNED AERIAL VEHICLE IN A NON-COOPERATIVE ENVIRONMENT

Dale D. Rowley

Department of Mechanical Engineering

Master of Science

Landing a rotorcraft unmanned aerial vehicle (RUAV) without human supervision is a capapability that would significantly broaden the usefulness of UAVs. The benefits are even greater if the functionality is expanded to involve landing sites with unknown terrain and a lack of GPS or other positioning aids. Examples of these types of non-cooperative environments could range from remote mountainous regions to an urban building rooftop or a cluttered parking lot.

The research of this thesis builds upon an approach that was initiated at NASA Ames Research Center to advance technology in the landing phase of RUAV operations. The approach consists of applying JPL's binocular stereo ranging algorithm to identify a landing site free of hazardous terrain. JPL's monocular feature tracking

algorithm is then applied to keep track of the chosen landing point in subsequent camera images. Finally, a position-estimation routine makes use of the tracking output to estimate the rotorcraft's position relative to the landing point. These position estimates make it possible to guide the rotorcraft toward, and land at, the safe landing site.

This methodology is implemented in simulation within the context of a fully-autonomous RUAV mission. Performance metrics are defined and tests are carried out in simulation to independently evaluate the performance of each algorithm. The stereo ranging algorithm is shown to successfully identify a safe landing point on average 70%-90% of the time in a cluttered parking lot scenario. The tracking algorithm is demonstrated to be robust under extreme operating conditions, and lead to a position-estimation error of less than 1 meter during a 2-minute hover at 12 meters above the ground. Preliminary tests with actual flight hardware are done to confirm the validity of these results, and to prepare for demonstrations and testing in flight.

ACKNOWLEDGMENTS

This thesis reflects the support and influence of several notable individuals. I am grateful to the members of my committee, Dr. Timothy McLain, Dr. Randal Beard, and Dr. Jerry Bowman for the personal interest they have shown throughout my education at BYU. I would also like to thank Dr. Colin Theodore and Dr. Mark Tischler for their professional direction and criticism, and for providing the opportunity for me to participate in this research. My wife, Emily, and my parents, Leon and Becky, contributed to this thesis in ways less tangible, but just as significant, for which I express my abiding appreciation.

# Contents

## List of Tables

## List of Figures

# Chapter 1

# Introduction

Although **R**otorcraft **U**nmanned **A**erial **V**ehicles (RUAVs) have been used in various applications for quite some time now, in reality very few of them are truly unmanned. Whether the vehicle is being employed for crop dusting, traffic monitoring, terrain mapping, reconnaissance, equipment transportation, search-and-rescue, or any other environment monitoring, the vehicle invariably requires human attention or control to some degree while it is in the air. Since human control is often inconsistent, inadequate, and/or expensive, a completely autonomous vehicle would offer substantial cost and performance benefits in nearly all of these applications.

When endeavoring to automate an RUAV mission, some of the most complex operations are encountered in the landing phase. A safe landing requires a high degree of intuition concerning the immediate environment, as well as the ability to adapt to unfamiliar situations. These attributes are extremely difficult to encode in a computer program. However, the ability to execute the landing task autonomously would play an important role in several aspects of a UAV mission. Foremost, completion of a mission is usually marked by the safe return of the vehicle to solid ground. Second, emergency landings may be an integral part of contingency handling in the event of a failure or unsafe operating conditions. Finally, landings may be scheduled as routine steps in a mission to conserve resources or accomplish other mission objectives.

## 1.1 Problem Statement

In order to eliminate the need for human supervision, an RUAV must be capable of at least three essential tasks: take-off, navigation, and landing. Of these

three efforts, an autonomous landing is perhaps the most complex. As a result, the **P**recision **A**utonomous **L**anding **A**daptive **C**ontrol **E**xperiment (PALACE) was conceived with the goal of advancing RUAV autonomous-landing technology. Within this project it is assumed that there will be no a priori information concerning the landing area, and that no GPS signal or other navigational aid will be available. By restricting operations to these types of non-cooperative environments, the usefulness of RUAVs is expanded to fulfill missions that are difficult or impossible with current technology. Some examples of non-cooperative environments could include remote mountainous regions, an urban building rooftop, or a cluttered parking lot.

The work of this thesis is founded upon, and a continuation of, the work that was initiated by a previous BYU student on the PALACE project. Joshua Hintze's work (reference [1]) began with the goal of assembling algorithms provided by the Machine Vision Group at JPL to demonstrate an autonomous RUAV landing in simulation. As a result of Hintze's research, several contributions were made to the PALACE project. First, JPL's stereo-ranging and monocular-tracking algorithms were implemented at NASA Ames Research Center in the RIPTIDE simulation environment. The purpose of these two algorithms was to process camera images to identify a safe landing site, and then keep track of the chosen landing point as the aircraft descended toward it. One significant outcome of this work was the validation of the algorithms' operation using artificial camera imagery from simulation. This suggested that machine vision may lay hold on the same benefits that other projects gain from simulation-based experiments.

Second, Hintze implemented a method for vision-based position estimation using the JPL monocular tracking algorithm and measurements from a laser rangefinder onboard the vehicle. This method was implemented in the RIPTIDE environment with a simulated laser rangefinder, and was shown to produce position estimates accurate enough for use in an autonomous navigation system. The aircraft's navigation system was adapted to maneuver the vehicle toward the landing point by using the position-estimation feedback instead of GPS. By combining this functionality with that of the stereo-ranging and monocular-tracking algorithms, Hintze simulated an

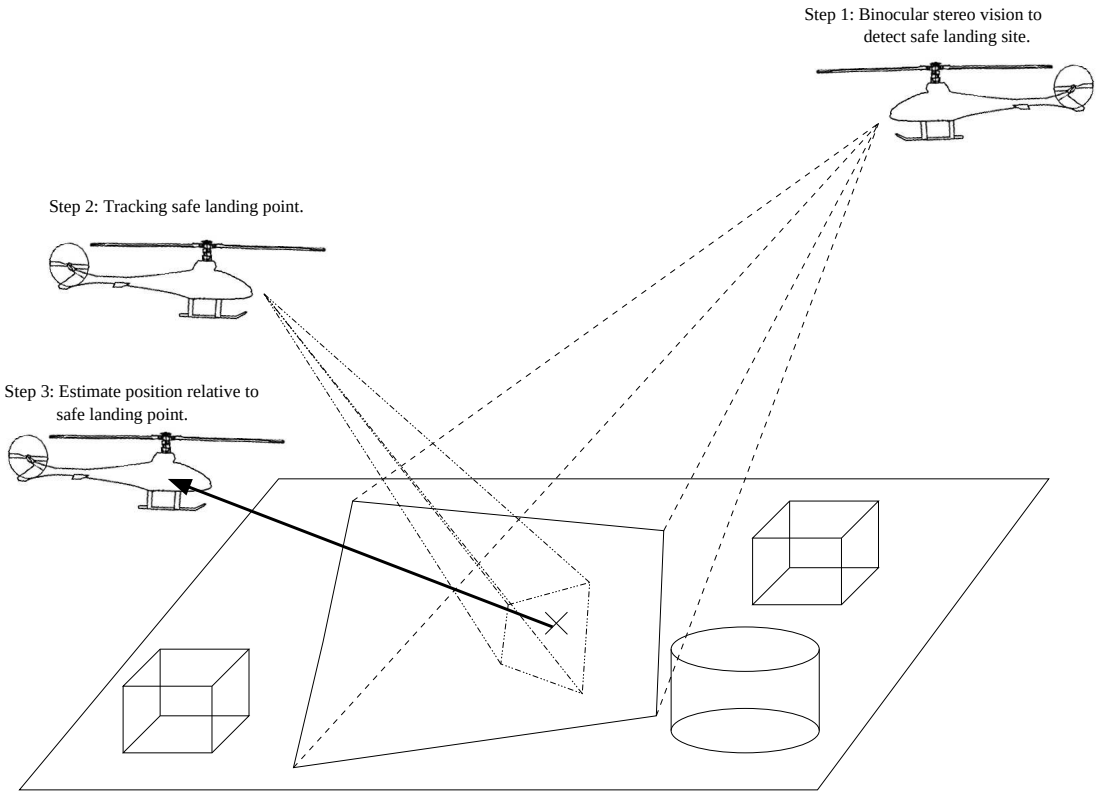



Figure 1.1: Summary of tasks performed by the vision-based algorithms.

autonomous RUAV landing in the RIPTIDE environment and illustrated how the software would fulfill the PALACE objectives. A summary of these algorithms' functionality is presented in Figure 1.1. Hintze further confirmed these conclusions with preliminary tests of the vision algorithms by post-processing data that was collected in flight by a real RUAV.

Hintze's research provided promising results, but the focus of it was on the implementation and demonstration of the feasibility of the vision-based approach. Hintze's research did not include a detailed evaluation of the system, and the implementation was characterized by several limitations. The sensors that were replicated in the simulation environment were implemented as idealized sensors without any noise models. Also, default values were used for many of the algorithm parameters. When default values for certain parameters led to unacceptable results, the values were adjusted until good performance was achieved. In some cases, this meant

that unreasonable settings were necessary to make the vision algorithms work, which implied a need to better understand the capabilities of the algorithms. The most significant limitation was the resulting aircraft instability that occurred when using the position-estimation feedback. This was temporarily resolved by reducing the control law gains and modifying the Kalman filter setup to handle the position-estimate signal characteristics. However, this produced sluggish aircraft motion and extra complexity in the Kalman filtering setup.

## 1.2 Objectives

The goal of this thesis is to fully-automate a typical RUAV landing in a non-cooperative environment. The landing procedure begins as the aircraft reaches the general landing site, about 30 meters directly over the nominal landing point, and will consist of the following steps:

1. Calculate an optimum aircraft heading based on environment conditions (i.e., wind direction, turbulence, sun angle, etc.) so as to maximize the quality of camera images.

2. Survey the terrain using the stereo-ranging algorithm, and analyze the data to select a safe landing site. If no safe landing site can be found, then continue the survey in an outward spiral pattern until a suitable point is identified.

3. Use the monocular-tracking and position-estimation algorithms to maintain a knowledge of where the point is relative to the aircraft so that it may descend toward the chosen landing site.

4. After arriving within a few feet of the landing point, invoke control logic to ensure that the aircraft is safely planted on the ground.

As of April 2004, Hintze demonstrated specific computer vision techniques that could be used to perform the critical landing tasks at the end of an RUAV mission. The objective of this thesis is to build upon Hintze's foundation by unifying the landing algorithms and fitting them into a larger autonomous framework so that a

comprehensive evaluation can be carried out. This will be done by implementing an autonomous landing within the context of a full-mission simulation. Mission parameters, such as waypoints, speeds, contingency actions, and the nominal landing point, will be communicated to the aircraft, after which the aircraft will fly the designated mission from take-off to landing without any human guidance. The full-mission simulation framework will be used to test each of the vision algorithms. Hintze's simplifications and assumptions will be examined with the intent of optimizing performance. These tests will contribute to a better understanding of the algorithms, and will be designed to quantify their performance and identify weaknesses or limitations. This work will culminate with the demonstration of a real-time, fully-autonomous RUAV landing in a simulation environment.

Some work will also be done in parallel to further the ultimate objective of the PALACE project: the fully-autonomous landing of a real rotorcraft UAV. The hardware that was chosen to be used for the PALACE project is the Yamaha RMAX helicopter UAV, which is operated by the ARP (**A**utonomous **R**otorcraft **P**roject) group at NASA Ames. The scope of this hardware implementation effort will be relatively limited, and will begin at the integration of a laser rangefinder with the existing RMAX hardware. The objective will be to demonstrate the individual real-time execution of the stereo-ranging, monocular-tracking, and position-estimation algorithms on the RMAX helicopter to confirm the algorithms' feasibility, and to prepare for future development and testing for flight demonstrations.

### 1.3 Related Work

Due to the multifarious practical applications of autonomous UAVs, the research literature on this topic covers every imaginable aspect of the field. There are many accounts of autonomous fixed-wing and rotorcraft landings. Most research in this area is likely to discuss, to some extent, one or more of the following core problems: avoidance of landing site hazards, tracking the landing point, and position estimation. However, Hintze's work ([1, 2]) is the only published research that was

found to address all of these problems specifically for landing RUAVs autonomously in a GPS-denied, non-cooperative environment.

### 1.3.1 Landing Site Hazard Avoidance

Most research in autonomous UAV landing tends to simplify the problem by assuming a specially-prepared landing environment. For example, in [3] computer vision systems are used to land an unmanned helicopter by identifying an "H" pattern painted on the landing pad. In this case, finding a safe landing site is reduced to pattern recognition involving a predetermined shape. This method is reported to work very well, with an average final position error of 40 cm from the center of the landing pad, and an average orientation error of 7 degrees. In [4], the Sierra Nevada Corporation describes a very reliable system involving an all-weather ground station with a **M**illi**m**eter-**W**ave (MMW) radar to track and guide UAVs to a predefined landing area, thus helping them avoid unsafe terrain. This system claims 400 successful UAV landings since 2001, and a success rate greater than 99.95%. One other common approach is to simply provide the GPS coordinates of a site that is known to be free of obstacles.

Those who do attempt to address the landing hazard avoidance problem almost always fall back on the large amount of general research that has been done to provide for obstacle detection and avoidance in an aerial vehicle [5, 6]. Almost all approaches involve some variation of stereo-ranging technology, laser scanning, or MMW radar. These techniques can be classified as either passive or active.

Passive approaches only operate on information that is already present in the environment. For example, stereo analyses are based on two or more camera images that are usually generated from ambient light. One type of stereo vision is binocular analysis, which attempts to deduce 3D information about the world by comparing two images taken simultaneously by cameras with a known configuration. In contrast, motion stereo analysis has the same objective, except that it compares several images taken over time by a moving camera, assuming the movement of the camera between each frame is known.

A representative hazard-avoidance approach using stereo methods is presented in [7]. The authors recommend the combination of binocular and motion stereo to detect obstacles and present the information to a pilot during low-altitude helicopter flight. The paper presents a comparison of actual distances to estimated ranges which shows an average error of about 25%. Stereo methods are a popular means for hazard avoidance because they are inexpensive and provide much information about the environment all at once. However, they do have a significant weakness in the lack of resolution, which is usually manifest as a trade-off between useful range, field of view, and accuracy.

The weaknesses of passive approaches sometimes necessitate the use of active techniques. An active technique is one that modifies the environment in some way in order to obtain information about it, such as sonar, laser, and MMW radar ranging. In [8], computer simulations demonstrated how LIDAR (**Li**ght **D**etection **A**nd **R**anging) might be used to select a safe landing site from a high altitude in the Mars atmosphere. A success rate greater than 93% is reported for trial runs in simulation. Although active methods afford greater resolution than stereo methods, there are several disadvantages. Since these methods modify the environment in the process of taking a measurement, they are usually characterized by a high electrical power consumption in comparison to passive approaches. Other disadvantages may include cooling requirements, greater weight, slower scanning rate, a more narrow field of view, and easier detection by unfriendly parties.

### 1.3.2 Tracking

As with obstacle detection, significant work has been done to explore methods for tracking features in camera images over time. There is a great diversity of techniques and many possibilities for variations on each technique, but most work in this area can be categorized as feature-based or optical-flow-based tracking.

Feature-based tracking seeks to identify a distinctive feature in an image with the hope that a brute-force search through subsequent images will locate a match with a high level of confidence. In [9], a method is proposed to help identify image features

that will be most easily tracked, and therefore improve the overall performance of the tracking algorithm. References [10, 11] describe another feature-based tracking implementation as a key element of a visual odometer that was tested in a helicopter UAV.

In contrast, optical flow methods ([12]) can be used to track features that are not necessarily distinctive. Under most circumstances, optical flow can be visualized as a 2D representation of the 3D velocities of all objects in the camera image. By assuming some constraints, partial differential equations can be formed to describe motion that would be apparent in the image. The solution of these equations can then be used to predict the locations of pixels representing features that are being tracked. Other similar numerical methods are available to accomplish the same task (see [13]). Although many examples are available to illustrate the application of both feature-based and optical-flow tracking methods, no common standards or metrics were found that might be used to compare their effectiveness.

### 1.3.3 Position Estimation

Vehicle position estimation is one other critical task in an autonomous UAV landing. Fortunately, the advent of GPS has provided a simple, affordable solution, and most research in autonomous UAV landing relies on GPS to some degree. However, one stipulation of the PALACE project is that the landing environment will preclude the use of GPS or any other similar positioning signals. Reasons for this restriction may vary from signal jamming to obstacles that occlude positioning signals. Vision-based techniques are once again an attractive alternative.

Many researchers have suggested computer-vision systems to provide position information. In [14, 15, 10, 11], motion stereo is combined with inertial measurements to produce a more accurate estimate of the vehicle's changing position. The specific methods in articles such as these invariably use feature-tracking with the assumption that the tracked features are fixed with respect to the world frame. Since the combination of vehicle translation and rotation can lead to ambiguities when analyzing motion in a sequence of camera images, the vehicle's attitude is usually measured

and used to correct the images so that vehicle translation can be inferred from the images. For the implementation described in [10], the error in the lateral position estimate is less than 1.7 meters based on images from about 4 meters above the ground. Although this type of position-estimation procedure is reported to work sufficiently well for most purposes, there is usually a noticeable lack of accuracy that grows with the distance squared from the tracked features.

## 1.4 Contributions

The goal of the PALACE project is to demonstrate a fully-autonomous RUAV landing in a non-cooperative environment, thereby opening the doors to a multitude of applications that would reduce human error, lower operation costs, increase efficiency, and broaden RUAV serviceability. This thesis makes the following contributions toward the accomplishment of this goal:

- Develop a landing procedure that is based on JPL vision algorithms to identify a safe landing point and guide the rotorcraft as it descends toward, and lands at, the chosen landing point.
- Demonstrate in simulation how the landing procedure is implemented in the context of a fully-autonomous RUAV mission.
- Show that artificial imagery from simulation can be used to test the vision algorithms and quantitatively predict how the algorithms will perform in RUAV flight hardware.

## 1.5 Outline

This thesis will discuss the accomplishments of the PALACE project in the following order. Chapter 2 will present in detail the purpose, basic concepts, and application behind the machine vision algorithms. This chapter will also define important algorithm parameters and terminology that will be used throughout the remaining chapters. Chapter 3 will discuss the simulation environment that was built to demonstrate a typical autonomous RUAV mission, including a fully-autonomous landing.

This will involve a description of the Mission Manager module that was developed to unify and execute the vision algorithms, as well as a report of the implementation issues that were encountered. The results of simulation testing of the vision algorithms will be presented in Chapter 4, including an evaluation of the performance of the three algorithms. Chapter 5 will document the implementation of each of the algorithms in real-time on the Yamaha RMAX helicopter. Conclusions will appear in Chapter 6, as well as a discussion of work remaining to fulfill the PALACE project goals.

# Chapter 2

# Vision Technologies

Landing a UAV autonomously in an unknown environment is an ambitious goal for the simple reason that the problem is based on very few assumptions; there is a high degree of spontaneity and a UAV must be capable of reacting to a wide variety of conditions. The most natural way to tackle such a problem is to analyze how it is done by a human, and then attempt to translate that knowledge into a sequence of well-defined steps that are amenable to computerization.

The difficulty that arises in applying vision technologies is that a human has experience and intuition that are not readily transferred to a computer program. For example, given an aerial view of a truck, a human would easily recognize it, reject it as an unsuitable landing surface, and use it to estimate sizes of other less-familiar objects in the image. To compensate for a computer's obliviousness, it must be given extra data, such as a second camera image, to replace human intuition with explicit calculations. The next three sections will introduce details of how the vision algorithms process this data to perform the tasks of landing-site selection and position estimation.

## 2.1 Stereo Vision

Perhaps the most difficult task in automating a UAV landing is choosing a suitable point to land in an obstacle field. Stereo machine vision is the tool that was selected to fulfill this objective. This section gives an overview of the stereo algorithm that was developed by JPL, which includes a discussion of camera calibration, image processing for range map generation, and safe landing area determination.

The primary goal of stereo vision analysis is to determine the 3D form of an environment that is represented by 2D images. One image is insufficient for this type of deduction, as can be seen in the fact that a small, proximate sphere will appear the same as a large, distant sphere in the same image. However, if another photo is taken of the same scene from a slightly different position or angle, then the two images can be compared and the distance can be estimated between the camera and the objects represented in each pixel from the images. The result is a 3D point cloud representation of the terrain, where each point is representative of one pixel from the original image. The final step in our application of stereo ranging is to examine this point cloud range map to select the optimum landing point based on the estimated slope and roughness of each neighborhood of points.

### 2.1.1 Camera Calibration

JPL's stereo algorithm assumes a binocular stereo camera arrangement where the image planes of both cameras are more or less coplanar with only a horizontal offset to separate them. The magnitude of the horizontal offset is known as the stereo baseline. In stereo ranging, extrinsic parameters such as the relative displacement and orientation of the cameras will determine to a great extent the similarity of both of the camera images. For example, reducing the stereo baseline will increase the similarity of the images. Intrinsic characteristics of each camera, such as focal length and lens distortion qualities, will also have some influence. When comparing stereo images, it is critical to take these qualities into account in order to accurately deduce 3D structure from the images. The process of obtaining quantities for these stereo parameters is known as camera calibration [16].

With the aid of computer software from JPL, the camera calibration process is reduced to a simple, well-defined procedure. Instead of directly measuring all of the required information, images were taken of a special calibration board and all necessary information was extracted from the images. The calibration board consists of a high-contrast grid of dots with an arbitrary, but precise, size and spacing (see Figure 2.1). The calibration steps were performed as follows:

1. Aim the cameras (which are anchored to and separated by a rigid bar) at the calibration board. Capture images from both cameras with the calibration board placed at different distances and angles from the cameras (see Figures 2.1 and 2.2). Some images must be taken with the board appearing in each corner of the camera images so that distortion parameters can be calculated (see Section 2.1.1).

2. Feed the camera images into a program that locates and outputs the $x, y$ pixel coordinates of the centroid of each dot in all of the images.

3. Feed the centroid pixel coordinates into a program that calculates a Tsai camera model (to be described shortly).

4. Feed the Tsai camera model, and a knowledge of the dot spacing on the calibration board, into a third program that reverse-projects the centroid pixel coordinates to obtain 3D world coordinates of each dot relative to the cameras.

5. Use the 3D world coordinates and original pixel coordinates to calculate the final CAHVOR camera model.

The end result of the camera calibration process is a CAHVOR camera model, developed by Yakimovsky and Cunningham [17] to encapsulate all information necessary to do a stereo analysis on images taken with the given camera setup. The following sections will first present the Tsai model that was used as an intermediate result in the calibration procedure listed above. The CAHVOR model will then be explained and compared to the Tsai model.

**Tsai Camera Model**

The Tsai model was developed by Roger Tsai [18] based on the pin-hole camera model, which is a simplified representation of how light from a 3D object passes through a camera lens and is mapped onto a 2D image plane. If the camera lens is represented by a pin hole, then the first Tsai model parameter - the focal length - is illustrated as in Figure 2.3. Given this information, as well as the camera position

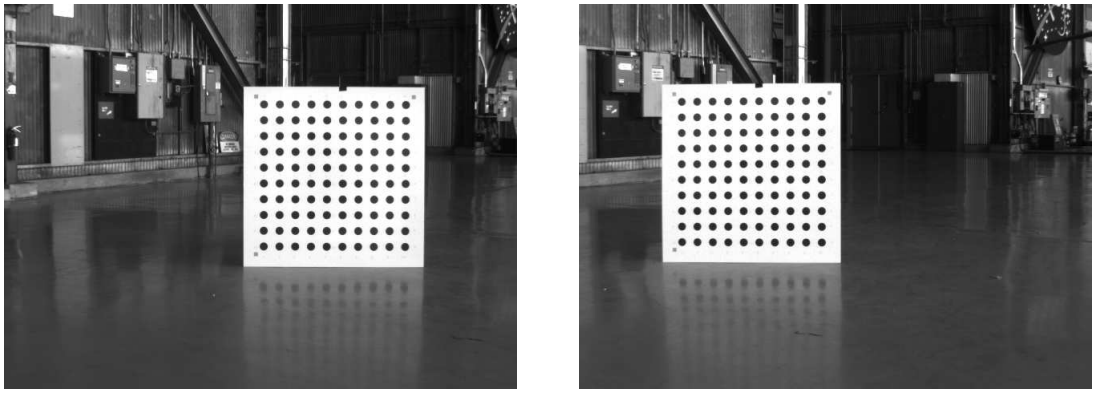


Figure 2.1: Sample calibration images from left and right cameras.

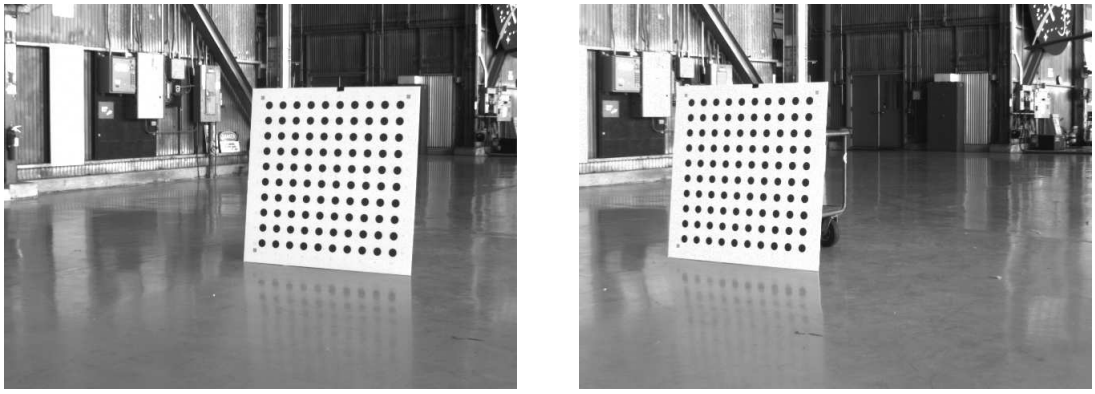


Figure 2.2: Calibration images with calibration board at an oblique angle.

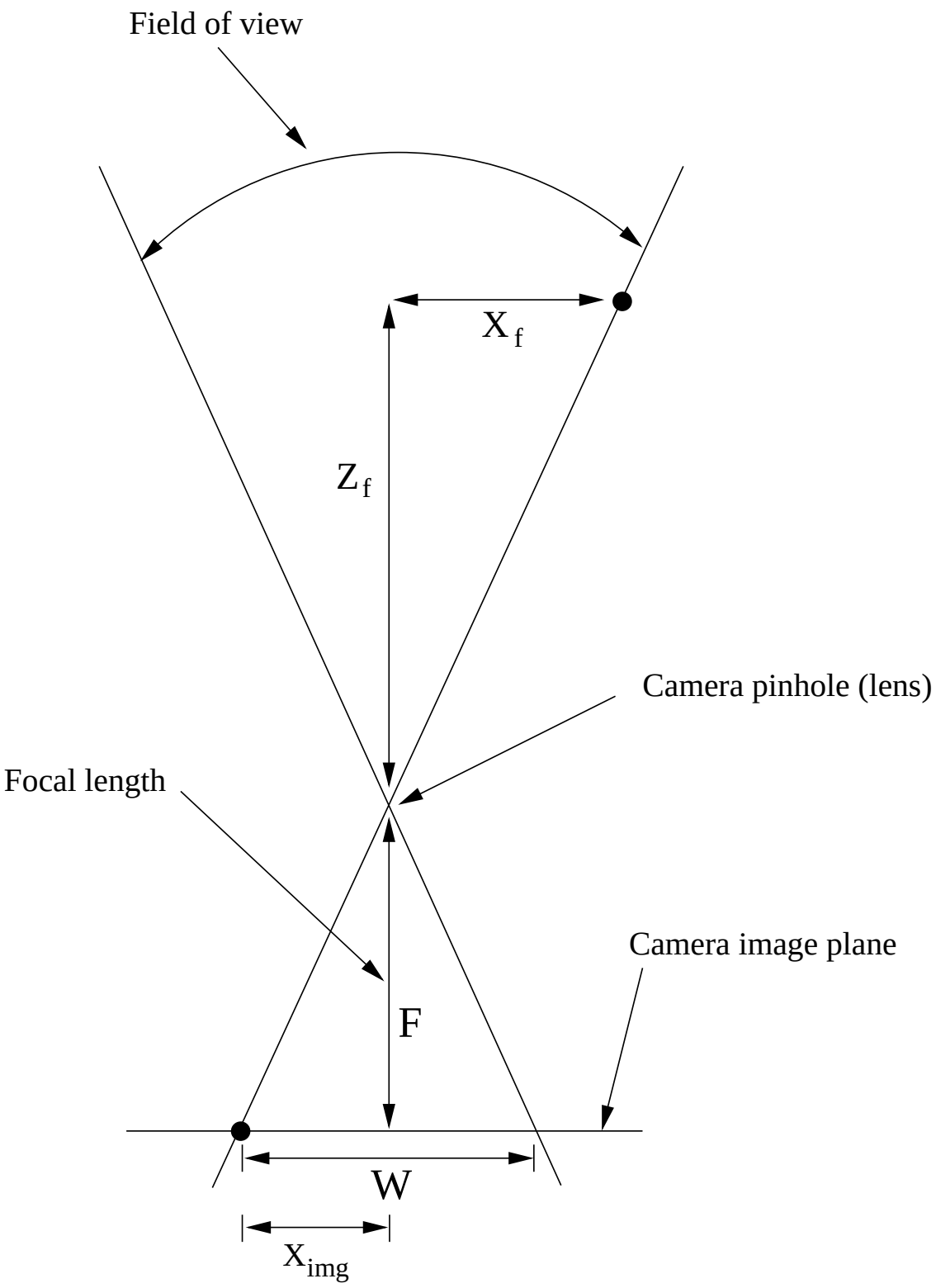



Figure 2.3: Pinhole camera model diagram.

Table 2.1: Tsai camera model parameters.

| | |
|---|---|
| $f$ | camera focal length |
| $\kappa$ | first-order radial lens distortion coefficient |
| $C_X$, $C_Y$ | $x, y$ coordinates of the center of the radial lens distortion |
| $S_X$ | pixel skew factor |
| $R_X, R_Y, R_Z$ | rotation angles that describe the transformation from the inertial to the camera coordinate system |
| $T_X, T_Y, T_Z$ | translation magnitudes that describe the transformation from the inertial to the camera coordinate system |

and orientation in world coordinates, the Tsai model maps a 3D point to an $x, y$ coordinate on the image plane. This projection is represented as

$$\begin{bmatrix} f & S_X & C_X \\ 0 & f & C_Y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} x_{img} \\ y_{img} \\ 1 \end{bmatrix} \tag{2.1}$$

where $x_w$, $y_w$, and $z_w$ are the 3D point coordinates in the world frame, and $x_{img}$ and $y_{img}$ are the projected 2D image plane coordinates. The $R$ sub-matrix is the Euler transformation matrix for the Euler rotation angles $R_X$, $R_Y$, and $R_Z$ (see Equation 2.11); the $T$ sub-matrix is the vector of translation magnitudes $[T_X \quad T_Y \quad T_Z]^T$; and all other variables are the Tsai model parameters listed in Table 2.1. In reality, most camera lenses are imperfect, which is manifest to a greater or lesser degree as a "fish-eye" distortion in the image. This is usually modeled as a radial distortion, and the Tsai model parameter $\kappa$ is used along with a 3rd-order polynomial to correct the image distortion before Equation 2.1 can be applied.

### CAHVOR Camera Model

The CAHVOR camera model is simply a different way of representing the same information stored in the Tsai camera model. The fundamental difference is that the Tsai model is based on certain assumptions that may not always be true.

Table 2.2: CAHVOR camera model parameters.

| | |
|---|---|
| **C** | Center of focus; the 3D coordinates of the pinhole focus point |
| **A** | The vector normal to the sensor plane |
| **H** | Horizontal information vector |
| **V** | Vertical information vector |
| **O** | Optical axis used only for lens-distortion correction |
| **R** | Radial lens distortion coefficients |

Since the CAHVOR model does not make these assumptions, it can theoretically lead to a better camera model, and therefore, better stereo results.

As shown in Table 2.2, CAHVOR is an acrostic where each letter is a mnemonic for a vector that encodes camera characteristics. An explanation and derivation of these vectors is given in [19]. When camera distortion information is not needed, it is not uncommon to discard it and only work with a CAHV model. Since the simulation environment involves "perfect" cameras oriented precisely with respect to each other, camera calibration was not necessary. The values for the CAHVOR vectors were easily established based on a knowledge of their definitions (see Table 2.2). The following equations are CAHVOR analogs for the Tsai equation (2.1) where $P$ is the world point $[x_w \quad y_w \quad z_w]$, and the other variables are the CAHV vectors defined in Table 2.2:

$$x_{img} = \frac{(P - C) \cdot H}{(P - C) \cdot A} \tag{2.2}$$

$$y_{img} = \frac{(P - C) \cdot V}{(P - C) \cdot A} \tag{2.3}$$

### 2.1.2 Stereo Image Processing

After the cameras' characteristics have been encoded in a camera model, the model is applied to the stereo images to produce a range map. An inaccurate model will lead to spotty, imprecise range data. The process of generating a range map involves the following steps:

1. Image reduction.

2. Image filtration.

3. Image rectification.

4. Disparity map computation.

5. Range map generation.

The purpose of the first three steps is to condition the images in order to simplify the computations in the fourth step, and to avoid potential problems. First, the size of the image is reduced. Although this step is optional, it is frequently done to help remove noise from the images, and to decrease computation time. JPL's stereo algorithms offer the option of several different levels of reduction, represented as pyramid levels where each level reduces the image size by a factor of 2. Pyramid level 0 corresponds to the original image (640×480 pixels in this thesis), and pyramid levels 1 and 2 would represent reduced image sizes of 320×240 and 160×120 pixels respectively. The desired level of reduction is usually only dependent on the available processing power and the required range map resolution.

The next step is to filter the images. If the two images differed in brightness or noise characteristics due to slightly different camera positions or idiosyncrasies, then it might be difficult to compare the two images. The JPL stereo algorithm offers the option of applying a Laplacian or a bilateral filter to help mitigate these effects. This thesis settled upon the Laplacian filter since it seemed to lead to slightly better results with images from simulation.

At this point, although the images have been reduced and filtered, they are not yet suitable for comparison. The third step consists of applying the CAHVOR camera model information to remove lens distortion effects and correct for small optical axis alignment errors. The goal of this step is to produce images whose corresponding features lie on the same horizontal row of pixels, thus reducing the image-comparison task to a search within only one row of pixels in the other image (see next step).

The fourth step is the heart of the stereo analysis and the justification for the preceding steps. As mentioned earlier, it is assumed that the cameras' optical axes

are parallel and that the second camera is only offset horizontally from the first. In this case, an object will appear at the same vertical location in both images, but the horizontal location will differ in each image. For example, Figure 2.4 shows horizontal strips from two images that were produced by cameras with a 1 meter stereo baseline. Note that the corner of the box in the left camera image is shifted roughly 60 pixels to the right of where the same box corner appears in the right image. This offset is known as the pixel disparity, and is dependent on the distance between the object and the cameras.

Figure 2.5, shows approximate disparities observed in simulation for objects at different ranges, and illustrates that objects closer to the cameras will exemplify a larger disparity than distant objects. Step four in the stereo algorithm consists of matching pixels in both images and calculating a disparity map. This disparity map will contain a disparity value for every pixel in the left image that was matched in the right image. With JPL's stereo algorithm it is possible to specify a maximum disparity, which reduces processing time by instructing the algorithm to only search for matches within the region of the image constrained by the specified maximum disparity.

It is possible under some circumstances that pixels in the left image will not be identified in the right image. For example, an object may appear in one image, but stand occluded in the other. Or the images might contain regions of homogeneous intensities that make the matching task ambiguous. It is impossible to calculate disparities for these pixels, so they are ignored. For the remaining pixels, the disparity value is fed into the camera model to calculate 3D points representative of each pixel. The collection of these 3D points is the desired output range map. Pixels that lack disparity values are manifest as holes in the range map.

### 2.1.3 Safe Landing Area Determination

The final step in selecting a landing point is to extract data from the range map to identify hazard-free areas. This is done using JPL's SLAD (**S**afe **L**anding **A**rea **D**etermination) algorithm [8]. The primary inputs to this algorithm are the

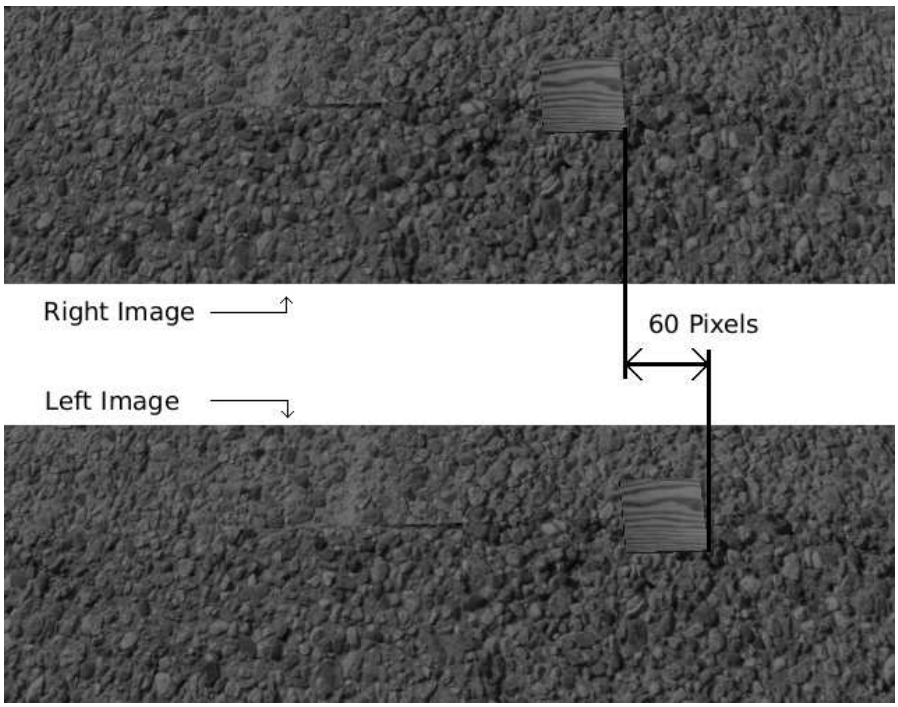



Figure 2.4: Sections of left and right images illustrating disparity.

stereo range map and three constraints: 1) maximum surface roughness, 2) maximum surface slope, and 3) the minimum distance from hazards, where hazards are defined as terrain that violates the first two constraints. The output is an $x$,$y$ pixel coordinate corresponding to the optimum landing point represented in the left-camera image. This section provides a general overview of the inner workings of the SLAD algorithm.

### SLAD Maps

The objective of the SLAD algorithm is to find regions in the range map that satisfy the three constraints of roughness, slope, and distance from hazards. The first step is to re-sample the range map data points into a regular grid using bilinear interpolation. The grid must be square, and will be arbitrarily set to a size of 400×400 data points in this thesis. The result is an elevation map $Z(r, c)$ representing the elevation at the intersection of row and column $r$,$c$ where the rows and columns are spaced equally over the terrain. Re-sampling the data points into a regular grid

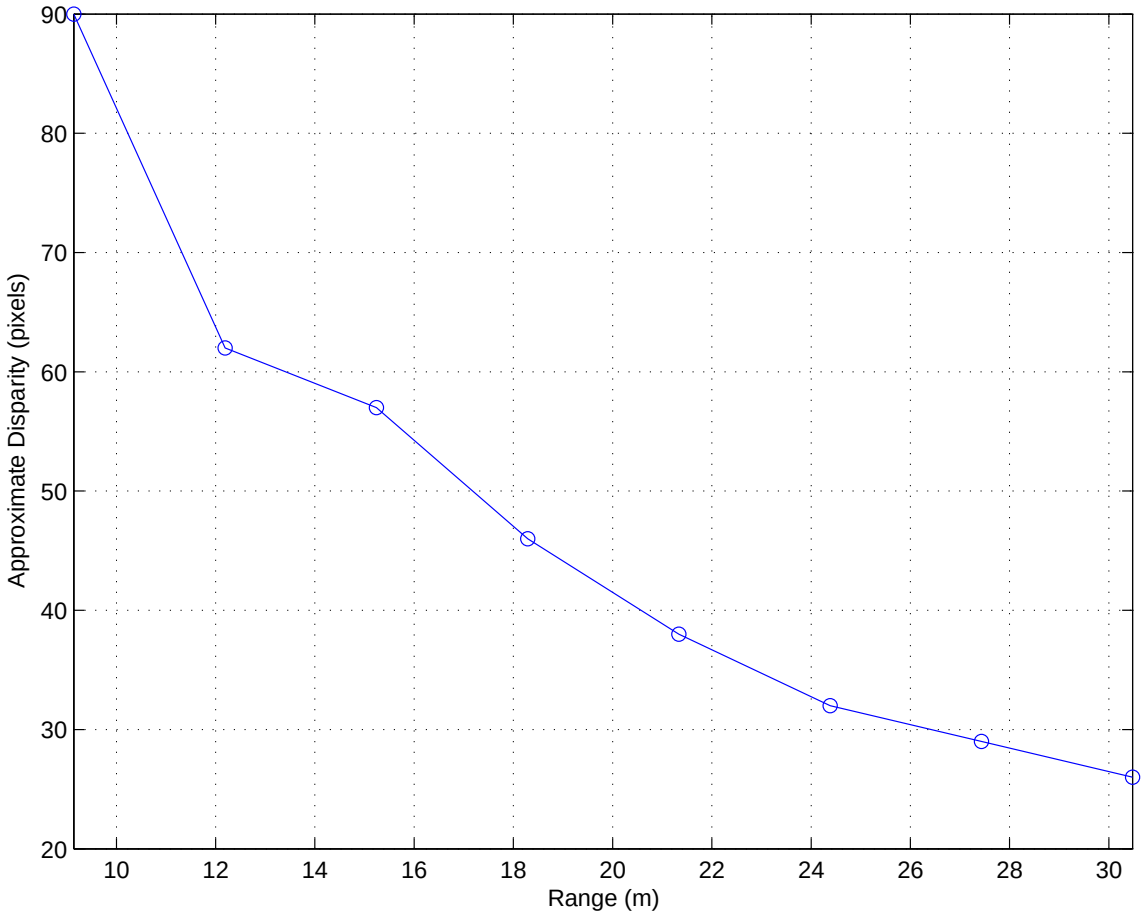



Figure 2.5: Plot of average disparities observed from different altitudes in simulation.

reduces processing time by eliminating the need to account for irregular spacing of the range map data points in subsequent calculations.

Once the elevation map has been generated, the next step is to identify landing hazards in the terrain. A hazard is defined to be a region of points where the slope or roughness exceeds the given maximum values. In order to calculate the slope at each point in the elevation map $Z(r, c)$, a plane is fitted to a grid of neighboring points, and the slope is recorded as the angle between the vertical direction and the normal to the plane. The roughness is then calculated by subtracting the original elevation $Z(r, c)$ from the elevation of the fitted plane at that same point. The results of these calculations are a slope map $A(r, c)$, and a roughness map $R(r, c)$.

The roughness and slope maps combined will contain enough information to identify all image regions that represent a safe landing site, which is illustrated in the safe-distance map. But in order to locate the *best* landing point, the data from the safe-distance map is converted into a cost map. This is done by setting the cost map $C(r, c)$ to 1.0 wherever the roughness or slope constraint is exceeded, or where the point $r, c$ is under the minimum distance $D_{min}$ from an offending point. All other points are assigned a cost as the normalized product of the slope and roughness values, which causes the best landing areas to be assigned the smallest values. In other words if

$$\left.\begin{aligned} A(r,c) &> A_{max} \\ R(r,c) &> R_{max} \\ D(r,c) &< D_{min} \end{aligned}\right\} C(r,c) = 1.0 \tag{2.4}$$

else

$$C(r,c) = \frac{A(r,c)R(r,c)}{A_{max}R_{max}} \tag{2.5}$$

Finally, the cost map is smoothed by averaging the cost at each point with its neighbors. The optimum landing point is then defined as the point with the lowest cost, and the CAHVOR camera model is used to return the $x$,$y$ coordinate of the pixel in the left camera image that would represent the selected optimum landing point. As an example, Figure 2.6 shows a typical scene taken from a simulation environment, with an "X" painted on the landing point that was selected by the SLAD algorithm. Figure 2.7 shows the intermediate SLAD maps that were used in identifying the optimum landing point.

In summary, the stereo ranging and SLAD processes represent a considerably complex analysis (outlined in Figure 2.8) of two images to select a safe landing point. In return for this complexity, the processes demonstrate the ability to successfully analyze a vast array of potential scenarios and entirely automate a task which normally requires a high degree of human intuition and supervision.

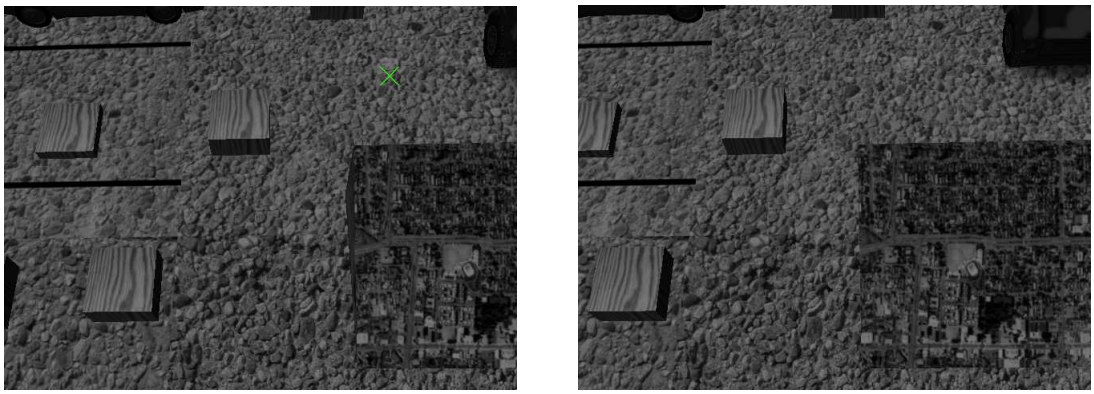
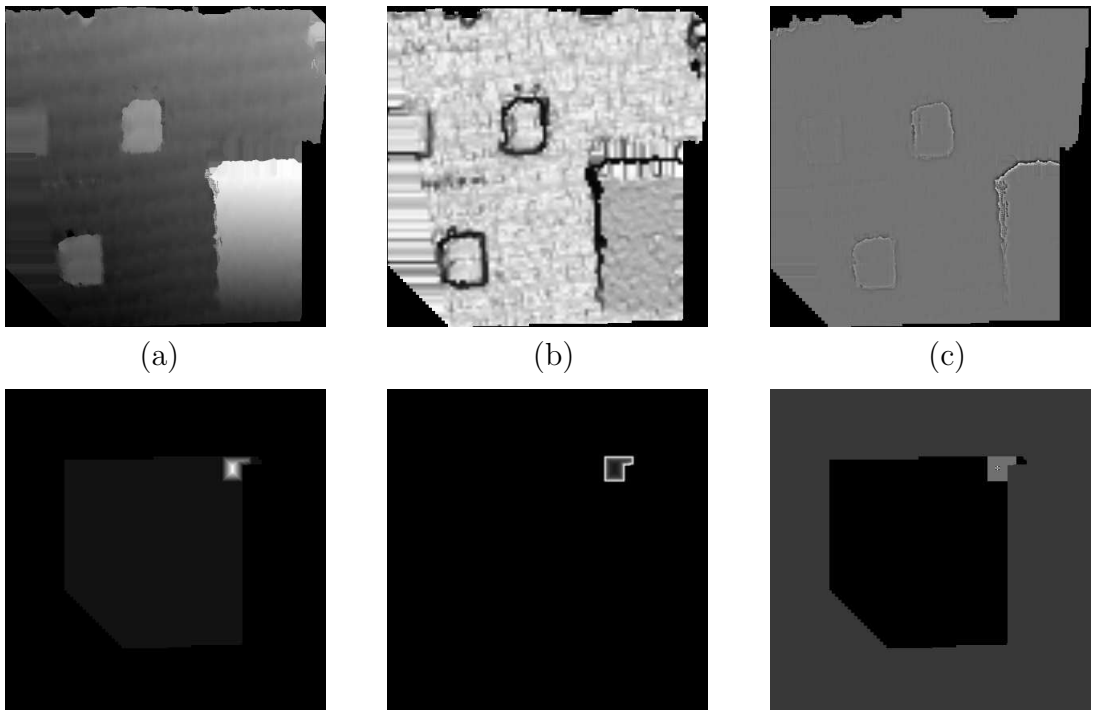


Figure 2.6: Left and right images of a simulation scenario, with an "X" marking the landing point chosen by the SLAD algorithm.



(a) (b) (c)

(d) (e) (f)

Figure 2.7: Intermediate SLAD maps: (a) elevation map, (b) slope map, (c) roughness map, (d) safe-distance map, (e) cost map, and (f) safe-landing map.

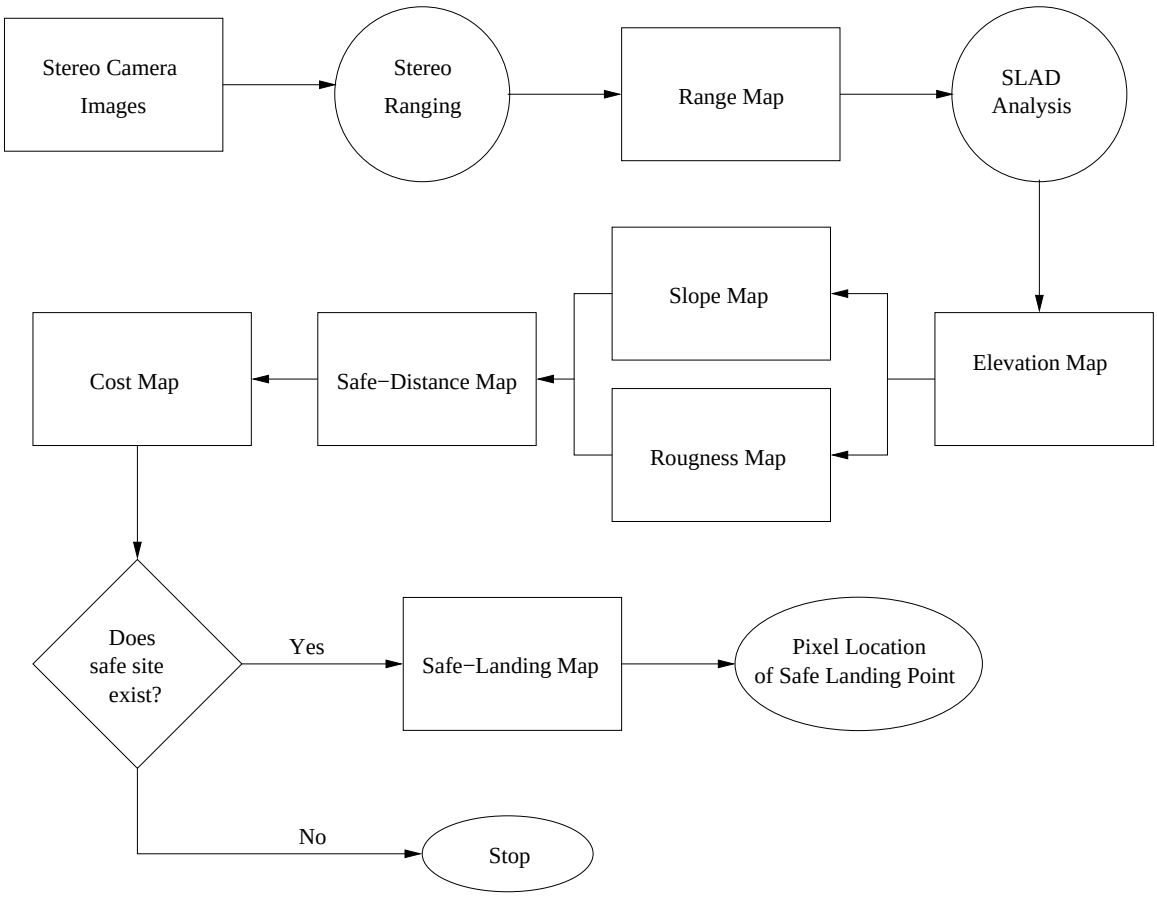



Figure 2.8: Stereo-ranging / SLAD analysis procedure.

## 2.2 Monocular Feature Tracking

After the SLAD algorithm returns the pixel coordinate of the chosen landing point, the next step is to maintain a current knowledge of the vehicle's position with respect to the landing site until the aircraft has touched down. This functionality is provided by two distinct algorithms: feature tracking and position estimation. The feature tracking algorithm is outlined as follows:

1. Initialization:
   (a) Provide a camera image and coordinates of a rectangular region in the image.
   (b) Search a rectangular region for the best feature to track.
   (c) Record the template window and pixel coordinate of the center pixel.
2. Provide a new camera image.
3. Convolve the old template within search window of new image.
4. Return coherence and pixel coordinates representing the highest convolution score.
5. If a match was found, then replace the old template window in memory with the high-scoring region from new image.

The tracking algorithm is initialized by providing a grayscale camera image, along with pixel coordinates that define a rectangular region of the image (see Figure 2.9). The algorithm searches within the rectangular region (a) for the feature (b) that will be most-easily recognized in subsequent frames. Since one pixel alone is generally not unique enough, it is necessary to define a feature as a small region of pixels (c) containing a unique pattern. This region will be termed the template window, the size of which may be defined by the user. The next call to the tracking algorithm is submitted with a new camera image, and the algorithm returns the

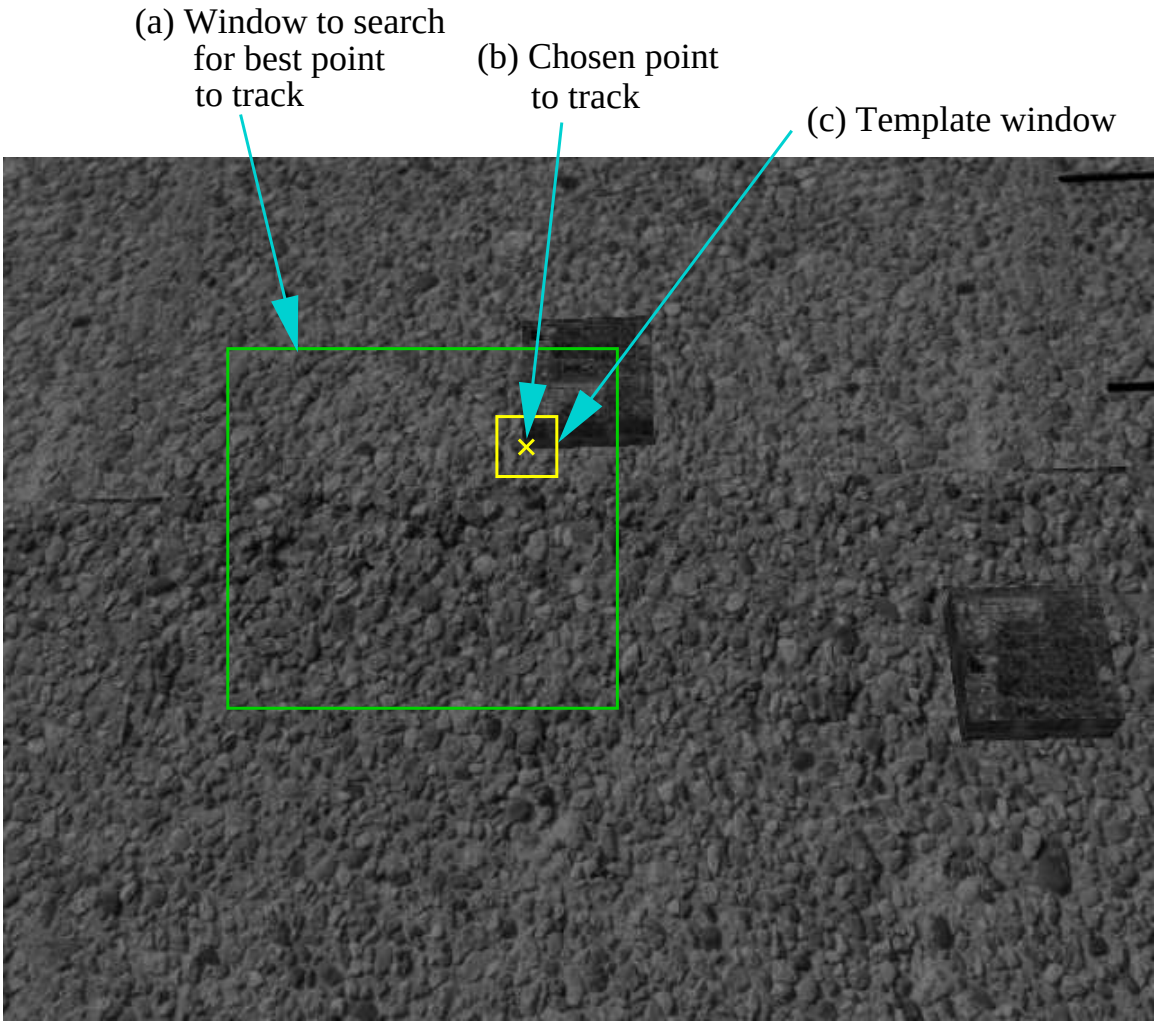



Figure 2.9: Initialization of tracking algorithm and selection of feature to be tracked.

$x, y$ coordinate (in the new image) that best matches the template window from the previous image.

In more detail, the comparison of the old and new images is done by convolution, a process in which the intensities of each pixel in the old template window are compared to each pixel within a same-sized region in the new image. An overall score is assigned to the comparison of these regions from the two images. Figure 2.10 illustrates how this comparison is repeated between the template window and other regions in the new image until a comparison has been done with every possible region within a predefined search window in the new image. The $x, y$ coordinates that

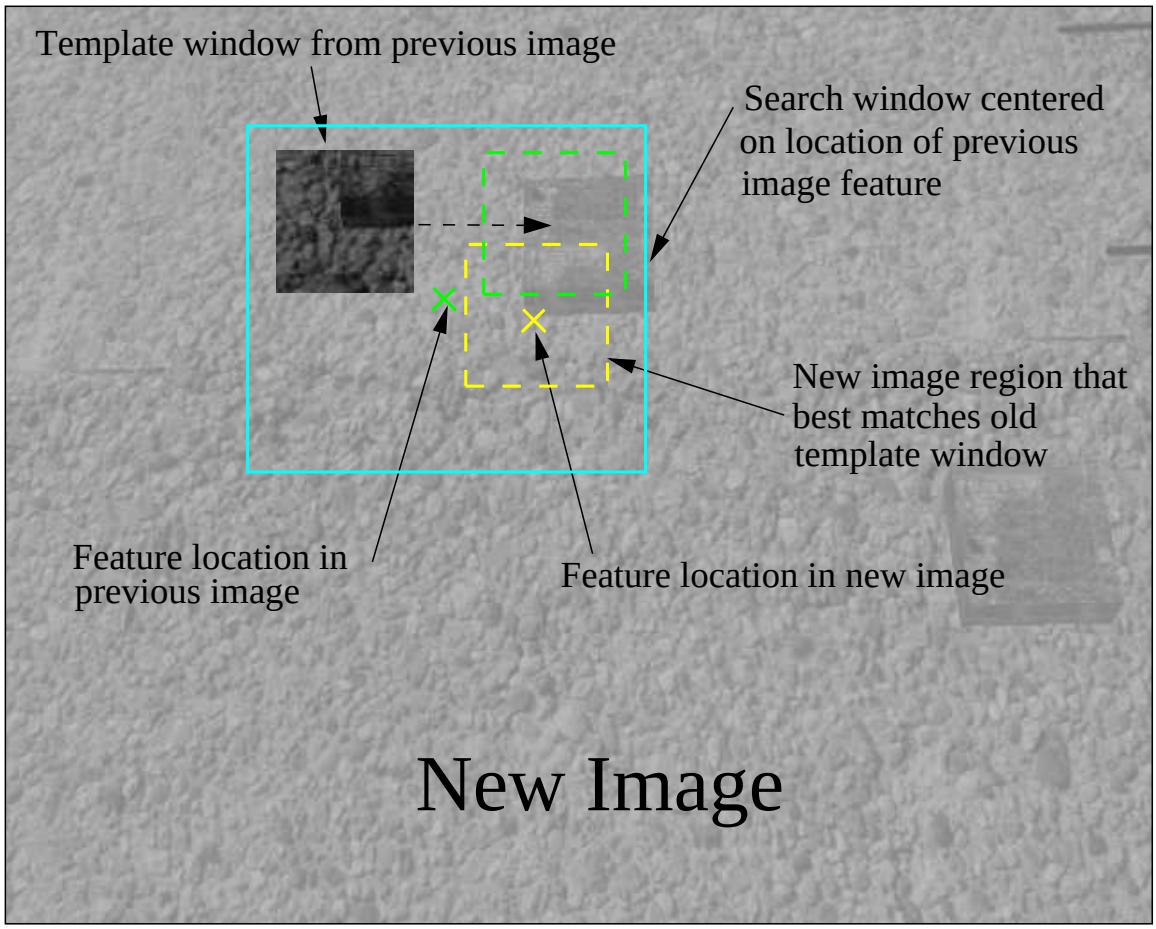



Figure 2.10: Convolution search procedure to find location of feature in new image.

represent the comparison resulting in the highest score are assumed to be the coordinates of the matching feature in the new image. A coherence value is also returned to indicate the level of confidence that a good match was found.

When attempting to find a match in the new image, there are two possible outcomes. If the returned coherence value is 0, then the tracking routine failed to find a match in the new image. In this case, the new image is discarded in hopes that a successful match will be found in the next image. On the other hand, if the coherence is greater than 0, then the tracking routine assumes that it found a match and the old template window is replaced in memory by the comparable region from the new

image. At this point, the algorithm is prepared to repeat the tracking procedure when it is provided with the next camera image.

When initializing the tracking algorithm, instead of instructing it to track the $x, y$ coordinate of the SLAD output pixel, it is given some freedom to choose to track a nearby pixel instead. The reasoning behind this method is that SLAD may select a pixel in a region of the image that does not contain unique features. Therefore, the tracking algorithm is allowed to begin tracking a nearby pixel that may be more easily recognized in future images. Obviously, the tracker must also be restricted enough that it will not start tracking a distant pixel that corresponds to an unsafe landing point.

## 2.3 Monocular Position Estimation

The final step to enabling an autonomous landing is to use the image pixel coordinate (output from the tracking algorithm) of the landing site to guide the aircraft toward the safe landing point. The approach used here is to convert the image pixel coordinate of the landing site into a pseudo-GPS **m**onocular **p**osition **e**stimate (MPE) that would take the place of the usual GPS signal input. This section will give an overview of the complete derivation of this method, which was originally developed by Hintze and presented in [1].

### 2.3.1 Coordinate Systems

There are four coordinate systems that are involved in the calculation of the aircraft's position. The first is represented by the camera image plane, which is a 2D coordinate system as defined in Figure 2.11. Next, the camera coordinate system is defined according to Figure 2.12. Finally, the helicopter and inertial coordinate systems are depicted in Figure 2.13.

In reality the camera is usually mounted on the side of the helicopter, but for this derivation it is assumed that the origins of the camera frame and helicopter frame are coincident. It is also assumed that the camera $x$-axis and the helicopter $y$-axis are collinear. The optical axis of the camera is allowed a variable pitch offset

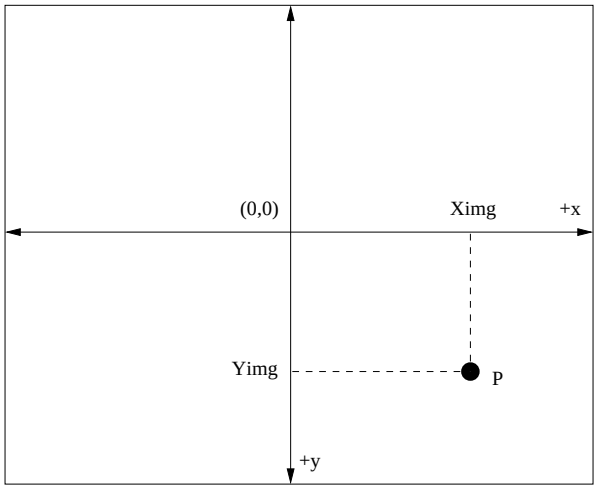



Figure 2.11: Image plane coordinate system.

$\theta_c$ from the helicopter $x$-axis so that the camera may be pointed toward the ground. This arrangement of axes is depicted in Figure 2.12. These assumptions simplify the position calculations, and are justified by noting that the camera offset from the helicopter center of gravity is relatively small, and inconsequential to the task at hand.

### 2.3.2 Derivation

It was noted in Section 2.1 that one camera image is insufficient to deduce 3D information about objects in the image. In order to estimate the aircraft position relative to a feature in a single image, additional information must be provided. The position-estimation method used in this thesis is based on these two assumptions:

1. The distance is known from the camera to the feature in the center of the camera image.

2. The center-of-image feature, and the feature that is being tracked, are at the same altitude.

The first assumption is supported on the Yamaha RMAX helicopter by mounting a laser rangefinder. A similar arrangement is made in simulation by creating a

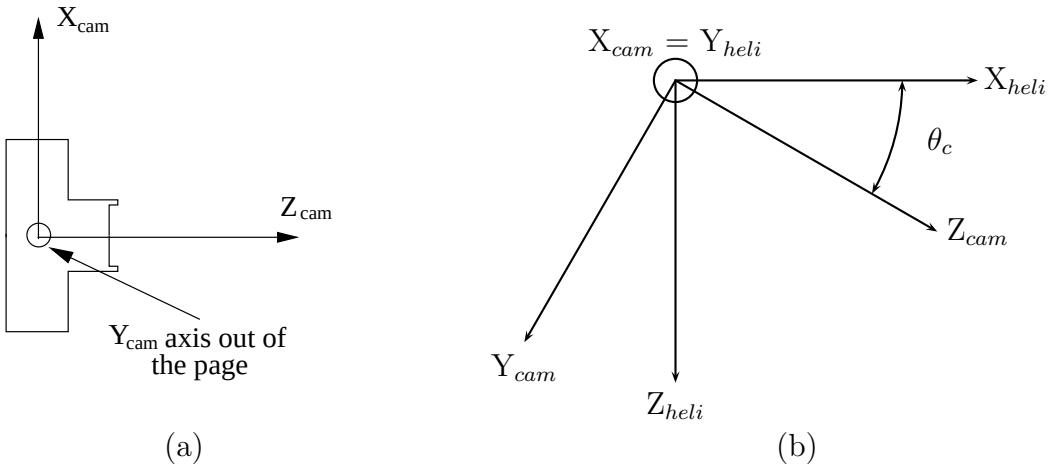



Figure 2.12: (a) Camera coordinate system. (b) Camera and helicopter coordinate systems.

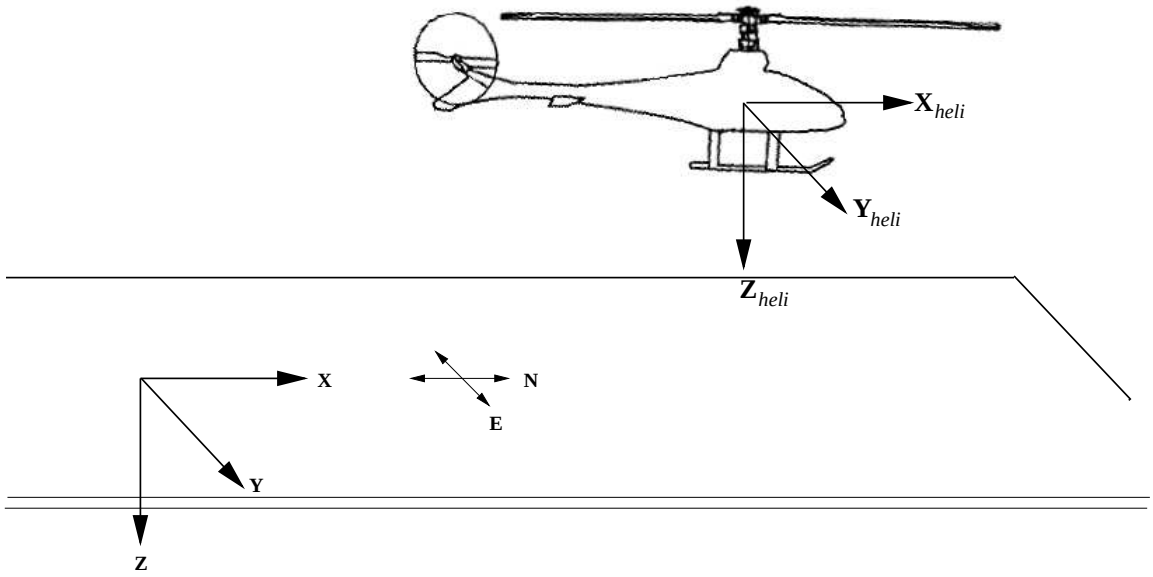



Figure 2.13: Helicopter and inertial coordinate systems.

simulated laser rangefinder. The second assumption may not be completely accurate depending on the scenario, but it is assumed that the error will be small enough that it will not have a significant influence on the landing performance.

The general approach to estimating the aircraft position is to calculate the "view" vector $[X_f \quad Y_f \quad Z_f]$ from the camera to the feature that is being tracked. Once this is known, Euler transformations can be applied to express this vector in the inertial frame. Since this thesis assumes that a GPS signal is available during the initial selection of the landing point, this view vector may be calculated and then added to the known GPS position of the aircraft to estimate the GPS position of the selected landing point. Thereafter, it is assumed that GPS is no longer available, and the position of the aircraft is estimated by calculating the view vector and subtracting it from the estimated position of the landing point. This method raises the concern that the vehicle position estimates will diverge from the true GPS coordinates over time. However, this is irrelevant because the position estimate is only a means to enable the aircraft to fly toward the safe landing point.

The view vector is calculated based on the camera model that was illustrated in Figure 2.3. By the law of similar triangles, we see that

$$X_f = \frac{x_{img}}{F} Z_f \tag{2.6}$$

$$Y_f = \frac{y_{img}}{F} Z_f \tag{2.7}$$

where $F$ is the camera focal length. This value is obtained from the camera manufacturer specifications, or in the case of simulation cameras it may be derived from Figure 2.3 as

$$F = \frac{\frac{W}{2}}{\tan \frac{\alpha_{fov}}{2}} \tag{2.8}$$

where $\alpha_{fov}$ is the camera field of view.

Since $x_{img}$ and $y_{img}$ are the pixel coordinates output by the feature tracker, $Z_f$ is the last value that must be determined. It is noted that a vector $\mathbf{P_c} = [X_f \quad Y_f \quad Z_f]$

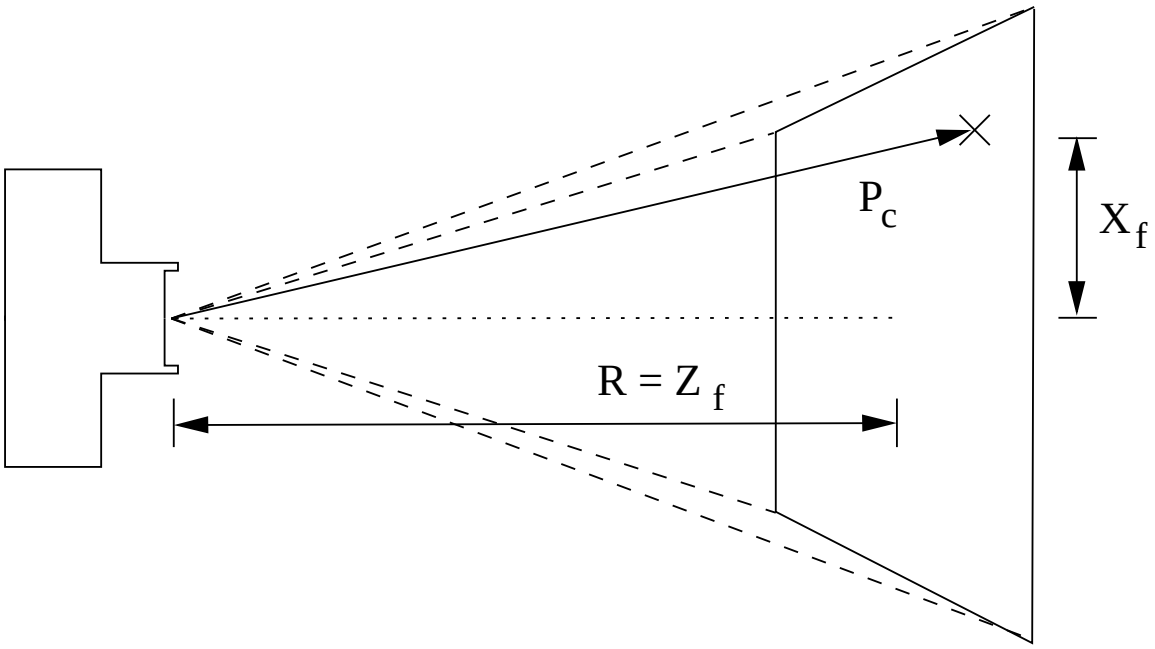



Figure 2.14: Illustration of a vector $\mathbf{P_c}$ in the camera frame.

in the camera frame (see Figure 2.14) may be transformed into a vector $\mathbf{P_h}$ in the helicopter frame by

$$\mathbf{P_h} = \begin{bmatrix} X_h \\ Y_h \\ Z_h \end{bmatrix} = \begin{bmatrix} 0 & \sin\theta_c & \cos\theta_c \\ 1 & 0 & 0 \\ 0 & \cos\theta_c & -\sin\theta_c \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \tag{2.9}$$

and therefore

$$Z_h = Y_c \cos\theta_c - Z_c \sin\theta_c \tag{2.10}$$

Similarly, a vector $\mathbf{P_h}$ in the helicopter frame is transformed into $\mathbf{P_i}$ in the inertial frame using a standard Euler rotation matrix [20] defined as

$$E(\phi,\theta,\psi) = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi + s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{2.11}$$

where $c\theta = \cos\theta$ and $s\theta = \sin\theta$. The result of Equation 2.9 is multiplied by $E(\phi, \theta, \psi)$ to complete the transformation of $\mathbf{P_c}$ to the vector $\mathbf{P_i}$ in the inertial frame. After substituting from Equation 2.6 and simplifying, the $z$-component $Z_i$ of this vector expands to

$$Z_i = Z_c \left[ -s(\theta) \left( c(\theta_c) + \frac{y_{img}}{F} s(\theta_c) \right) + c(\theta)\, s(\phi) \frac{x_{img}}{F} + c(\theta)\, c(\phi) \left( -s(\theta_c) + \frac{y_{img}}{F} c(\theta_c) \right) \right] \tag{2.12}$$

Since our objective is to calculate the $z$-component $Z_f$ of the view vector in the camera frame, then the substitution $Z_f = Z_c$ is made in Equation 2.12 and the equation can be solved for $Z_f$ if $Z_i$ is known. $Z_i$ is be calculated by noting that the vector in the camera frame to the center-of-image point is $[0 \quad 0 \quad D_r]$ where $D_r$ is the distance obtained from the laser rangefinder. Substituting $Z_c = D_r$, Equation 2.12 reduces to

$$Z_i = -D_r (\sin\theta \cos\theta_c + \cos\phi \cos\theta \sin\theta_c) \tag{2.13}$$

If the aircraft roll and pitch are 0 degrees, then Equation 2.13 reduces to $Z_i = -\sin(\theta_c) D_r$, which is basic trigonometry of a right triangle with the rangefinder distance as the hypotenuse. The minus sign is present since the camera pitch angle $\theta_c$ is defined to be negative when rotated downward. Equation 2.13 is substituted into Equation 2.12, and the formula is solved for the remaining unknown $Z_f$. This value can then be substituted into Equation 2.6 to calculate the view vector for the feature that is being tracked. Finally, the view vector is transformed into the inertial frame and applied as explained earlier to estimate the position of the aircraft.

# Chapter 3

# PALACE Full-Mission Simulation

As the PALACE mission is significantly complex and involves some risk (i.e., implementing new algorithms in expensive hardware), it is expedient that the concept first be proven in a realistic simulation environment. Simulations also offer the significant advantage of quick development iterations using minimal resources, as well as an easy implementation of Monte Carlo experiments to quantify performance. This section will describe the simulation environment, with most attention given to the Mission Manager module developed for this thesis to unify the vision algorithms and supply the level of autonomy required by the PALACE project. The chapter will conclude with some discussion of implementation issues.

## 3.1 Simulation Tools

The simulation environment developed for the PALACE project is an integration of several pieces of software, most of which originated at NASA Ames or JPL. Figure 3.1 is a summary of how these pieces fit together. The following sections will briefly describe each one.

### 3.1.1 RIPTIDE

In a simulation environment, the first tool that is required is a 3D display program to render the graphics to represent a realistic environment. The **R**eal-Time **I**nteractive **P**rototype **T**echnology **I**ntegration / **D**evelopment **E**nvironment (RIPTIDE) software is an extremely flexible 3D display program developed at NASA

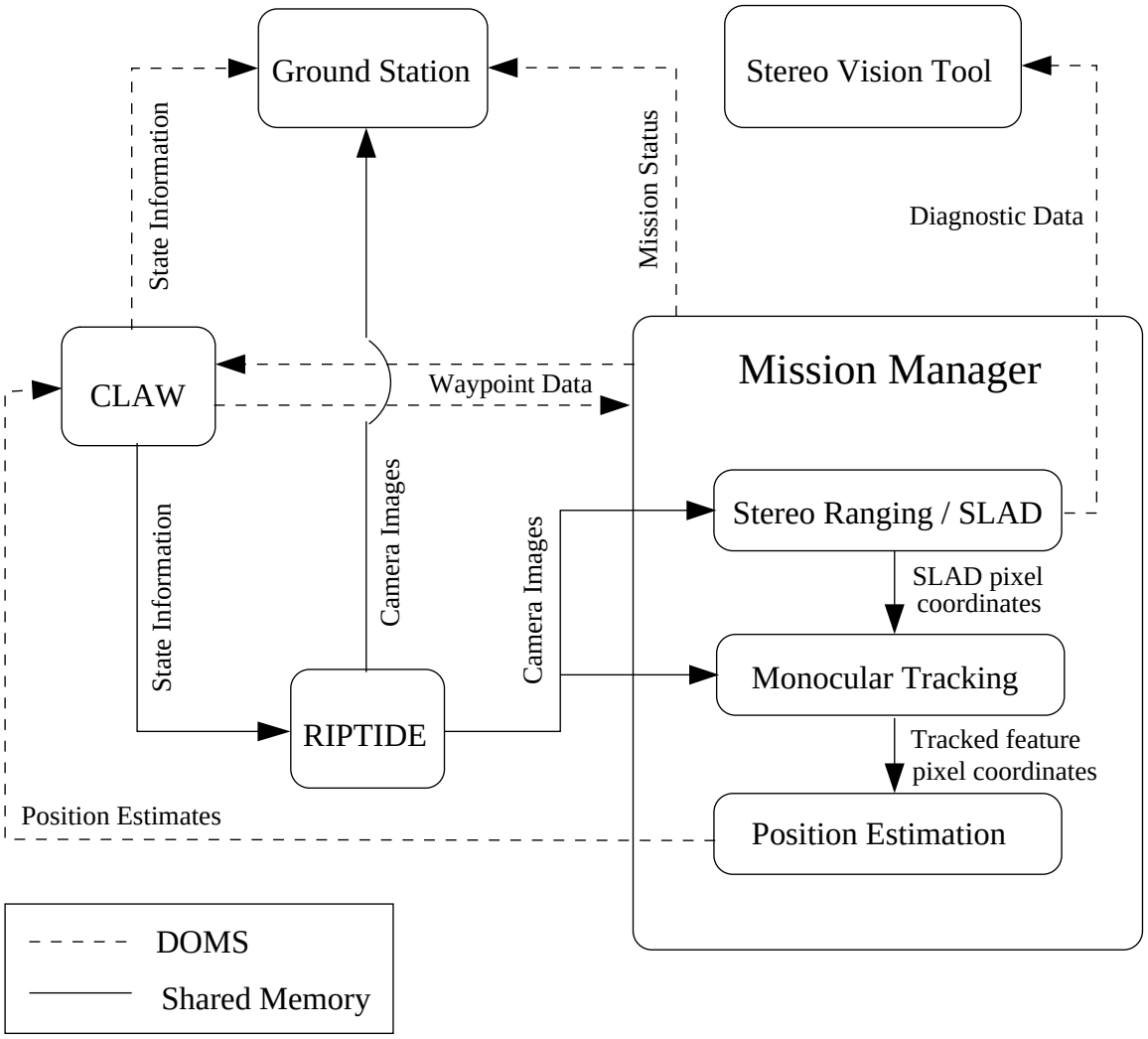



Figure 3.1: Diagram of inter-process communication within the simulation environment.

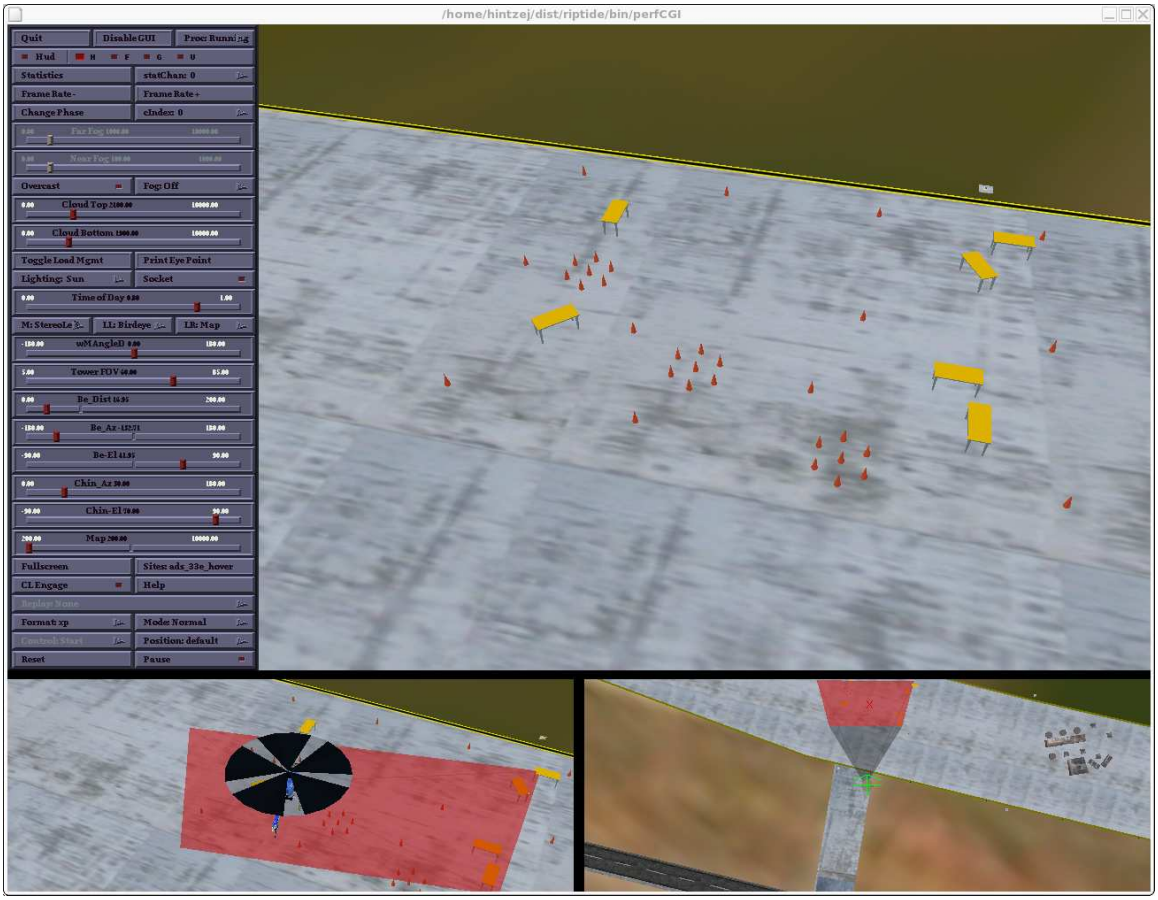



Figure 3.2: RIPTIDE screenshot showing stereo camera views under the cockpit view.

Ames for the "evaluation of a notional control law design using a high-fidelity non-linear mathematical model" [21]. The program offers an extensive database, complete with buildings, airports, trees, etc. The RIPTIDE screenshot in Figure 3.2 depicts a cluttered runway scene as it would be seen from multiple viewpoints, with the red trapezoid marking the terrain that is visible to the cameras. A parking lot scene was also added to the database specifically for testing the vision algorithms within the context of a PALACE mission. RIPTIDE makes extensive use of shared memory to provide for inter-process communication. This facilitates development of the math model and control laws as separate processes that continually update RIPTIDE on the state of the vehicle so that the scene appears as it would be seen from the aircraft.

RIPTIDE offers the ability to include sensor models in the simulation so that actual sensor views are returned. The PALACE project uses a pair of stereo cameras attached to the aircraft. Configuration parameters permit changing the placement and orientation of the cameras relative to the vehicle's center of gravity. When these cameras are enabled in the simulation, RIPTIDE renders the scenes viewed from the position of both cameras, and stores the images in shared memory. These images can then be accessed by other processes. Finally, the simulation includes a laser rangefinder that is collinear with the left-camera optical axis, and measures the distance from the camera focal point to the object represented in the center of the camera image.

### 3.1.2 CLAW

The **C**ontrol **LAW**s (CLAW) program, developed at NASA Ames as part of the Autonomous Rotorcraft Project, includes a high-fidelity vehicle dynamics model as well as the inner and outer loop control laws. The CLAW program models the dynamics of the Yamaha RMAX helicopter, which were determined using the system-identification methods described in [22]. This model enables ground-based testing and makes it possible for RIPTIDE to render a realistic view of what would be seen by the RMAX in flight.

The CLAW program also provides other functionality required by the PALACE project. The autonomous navigation system is perhaps the most critical. This feature consists of providing the CLAW program with a list of GPS waypoints which are processed to plan an efficient flight path from the first waypoint to the last, based on the aircraft capabilities [23]. This includes the management of aircraft velocities and accelerations while it is autonomously navigating the pre-planned flight path, as well as the provision for some contingency reactions.

As mentioned in Section 2.3.2, the navigation task normally relies on the presence of GPS signals, which CLAW feeds through a Kalman filter before producing the control efforts to maintain course. That section also described how GPS signals are replaced by MPE during the landing descent. Since the position estimates have

different noise characteristics, CLAW was retrofitted with a second Kalman filter to support the vision algorithms, as well as a switch to toggle between the two filters. This allowed the Kalman filters to be tuned separately for each source of position estimates. After a careful evaluation of the MPE algorithm, it was determined that one Kalman filter could handle both sources of position estimates (GPS and MPE). This will be discussed further in Section 3.4.

Another important CLAW capability includes procedures that have been developed recently for managing autonomous take-offs and landings. These procedures rely on data from other sensors such as sonar and switches on the skids to determine low-altitude height above ground and individual skid contacts with the ground. In addition to installing these sensors on the aircraft, they have been integrated in the RIPTIDE environment to support simulation testing and the contribution of these capabilities to the full-mission simulation.

### 3.1.3 DOMS

Since a UAV mission often involves many sensors simultaneously generating data, and functionality governed by multiple sources that rely on sensor data, the need arises for a mechanism of communication. The **D**istributed **O**pen **M**essaging **S**ystem (DOMS) software, also developed at Ames, was chosen to play this role in the PALACE project. The program provides for the generic definition of message structures. Processes may publicize their intentions to populate certain message structures with data and publish them, or they may subscribe to such messages that are published by other processes. The DOMS software then handles the details of delivering the data over TCP or UDP channels. The only thing two processes must have in common in order to communicate through this system is a **u**niform **r**esource **l**ocator (URL), in other words, a string that uniquely identifies the message structure that will be published or subscribed to.

The DOMS software is used heavily to support the needs of the PALACE project. The operation of the CLAW program depends on many sensors to observe and control the aircraft. It is constantly broadcasting this data so that other

programs may have access to it and record it for offline analysis. Communication with CLAW (e.g., position-estimate inputs to the second Kalman filter) is also done through DOMS. In actual flight, DOMS is the chief mechanism for sending data from the RMAX to the ground station, including camera image streaming and other miscellaneous process outputs. For the PALACE simulation, the situation is simplified somewhat by having all processes run on one computer. To conserve resources, data-intensive communications (such as camera images) are relayed via shared memory.

### 3.1.4 Ground Station

The PALACE project stipulated the need for ground-station software. As a result, the PALACE team developed a GUI to simplify the tasks of defining an RUAV mission and monitoring the RUAV's status during the mission. This software is only mentioned briefly here as it is involved in communications with the Mission Manager, which will be presented in the next section.

### 3.1.5 Stereo Display Tool

To simplify the testing of the Mission Manager, and especially the debugging of the stereo and SLAD algorithms, a tool was created to interact with the Mission Manager. The significance of this tool lies in the following features:

1. A display of bitmap images representing the intermediate SLAD hazard-analysis maps. A sample screenshot is provided in Figure 3.3.

2. An interactive 3D reconstruction of the landing site based on the range map output from the stereo ranging algorithm (see Figure 3.4).

## 3.2 Mission Manager

The Mission Manager is the simulation tool developed for this thesis to unify the functionality in CLAW, the ground station, and the vision algorithms to execute a fully-autonomous RUAV mission. The Mission Manager was designed and written

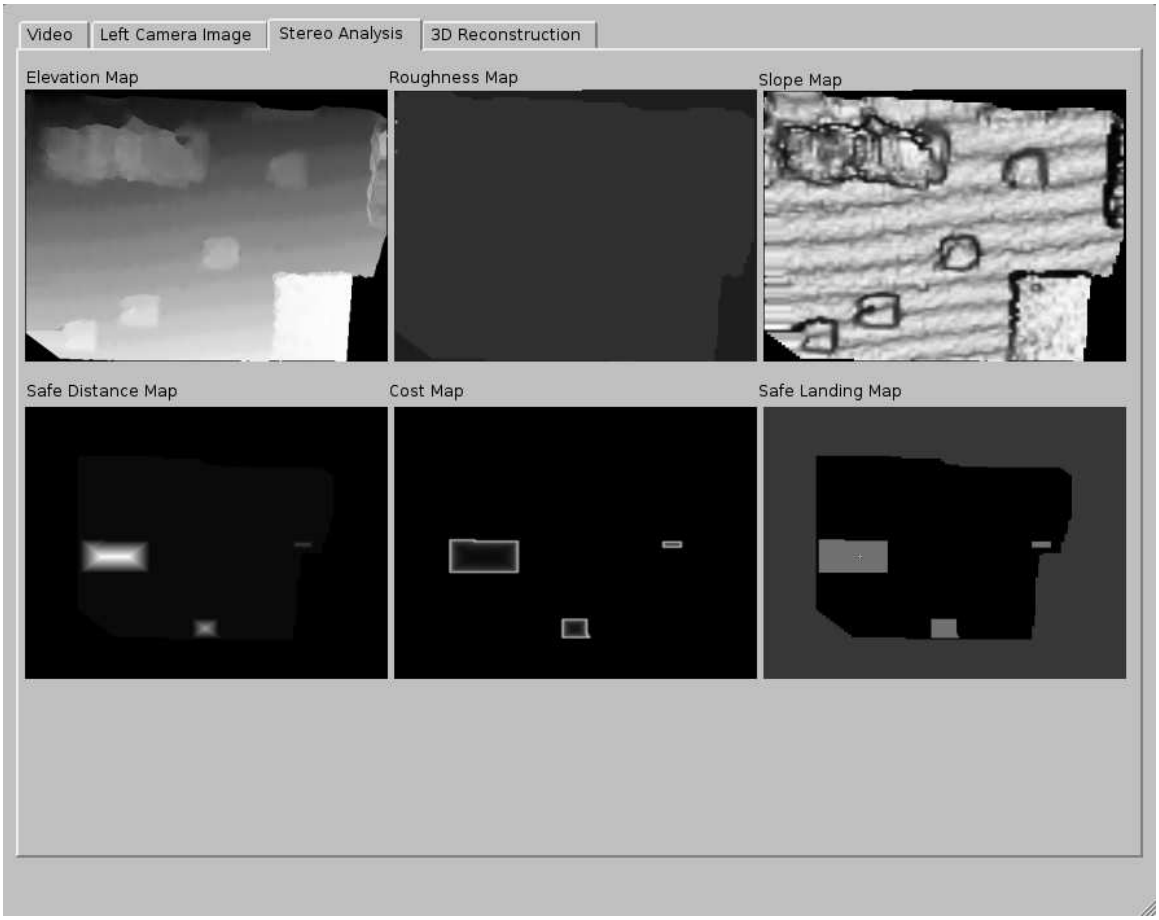



Figure 3.3: Screenshot of SLAD maps from Stereo Vision Tool.

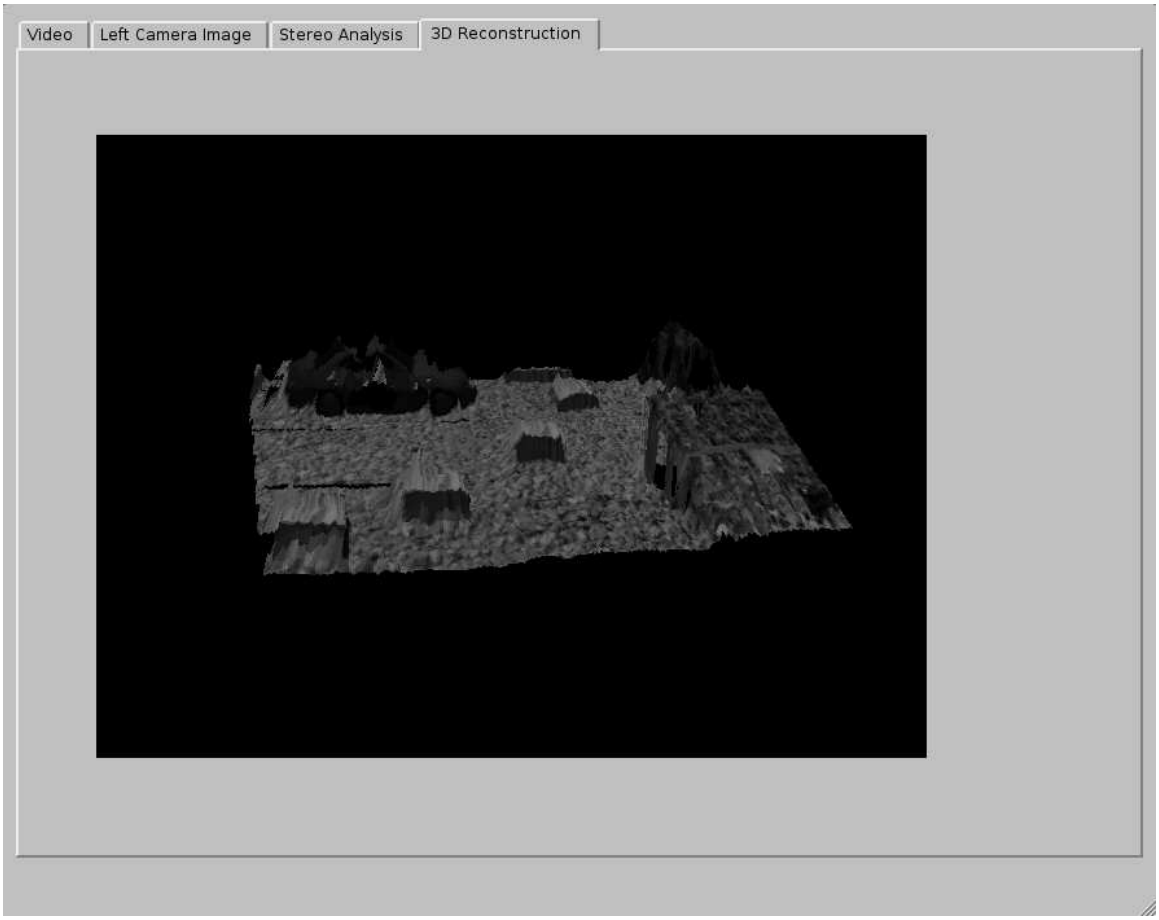



Figure 3.4: Screenshot of 3D reconstruction from Stereo Vision Tool.

in a modular fashion so that it may eventually be easily integrated with the actual RMAX helicopter hardware. The goal was to provide a framework for simple decision making and coordination to satisfy all of the functionality required for the PALACE project.

On command from the **P**ALACE **M**ission **O**perator (PMO), the aircraft takes off from a known location and ascends vertically to a predefined altitude to avoid collisions with other objects. The RUAV flies through a set of waypoints and stops at the final waypoint over the nominal landing area. The screenshots in Figures 3.3 and 3.4 show the helicopter at these two stages of the mission, namely, flying between waypoints and surveying the landing area. Finally, it follows an outward-spiral search pattern until a safe landing point is identified, and then descends and touches down at the selected safe landing point. Figure 3.4 shows the helicopter about to complete the landing task after having selected a safe landing point and descended to it under MPE feedback control. If the aircraft encounters any problems it must abort the mission, fly to the designated rally point, and await further commands from the PMO.

### 3.2.1 Architecture

The core features of the Mission Manager require it to make observations or receive communications and then take appropriate actions based on this data. In addition, the second CLAW Kalman filter is designed to receive position-estimate inputs at the nominal rate of 10 Hz. Thus the basic architecture of the Mission Manager is an infinite loop running at 10 Hz. Each cycle consists of performing the following sequence of actions:

1. Check for arrival of new DOMS messages.
2. Handle new action commands (i.e., begin/pause/abort the mission).
3. Handle new waypoint requests from the ground station.
4. Check for recent completion of a waypoint.
5. Check if a position estimate should be sent to CLAW.

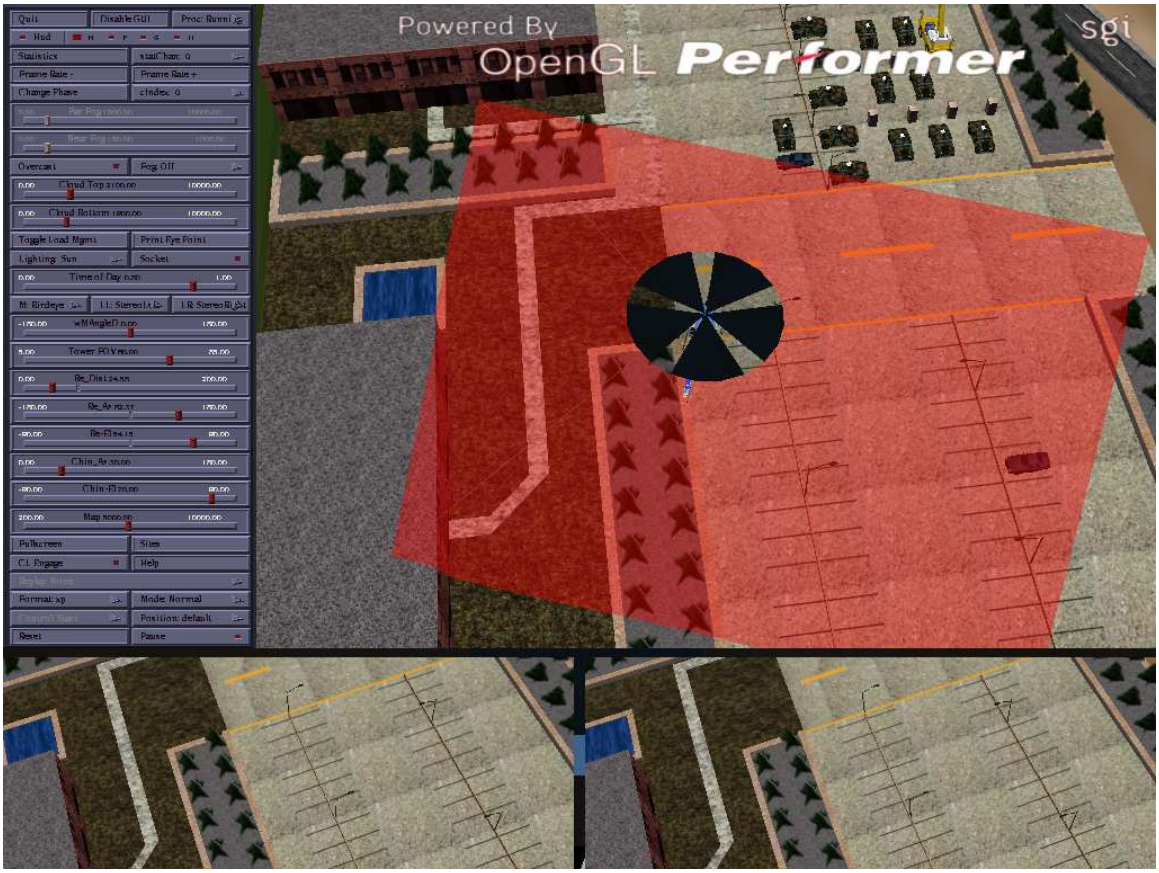



Figure 3.5: PALACE screenshot of the helicopter as it flies to the next waypoint.


Figure 3.6: PALACE screenshot of the helicopter as it reviews the landing area.


Figure 3.7: Screenshot of a PALACE mission showing the helicopter touching down at the landing point selected by the SLAD algorithm.

This list of actions makes it apparent that current PALACE missions are reduced to a definition in terms of waypoints. A mission begins when the ground station converts the PMO-defined mission data into a list of waypoints that are sent to the Mission Manager via DOMS, followed by a command to begin the mission. From that point on, the Mission Manager follows the outlined list of actions above to take-off, fly the waypoints, and land at the desired landing area, all without requiring any further input from the PMO. Figure 3.8 provides a flowchart representing the basic Mission Manager architecture. The following sections will describe the details of intermediate tasks which contribute to the success of the mission.

### 3.2.2 Communications

The first step in the main loop is to check for the arrival of new DOMS messages that are waiting to be processed. Since DOMS messaging is a central element in the coordination of a mission, Tables 3.1 and 3.2 summarize the messages that the Mission Manager subscribes to and publishes. These tables illustrate the nature of the Mission Manager's responsibilities. For example, CLAW communicates vehicle state information through the Dynamic State message, which the Mission Manager uses to calculate its height above ground and intermediate waypoint locations. Then it responds with the CLAW waypoint and CLAW Heading messages to command CLAW where to go. The other DOMS messages, such as the Laser Range message, represent other miscellaneous data traffic that is necessary to accomplish mission objectives.

The next step in the main loop is to handle new action commands. A "pause" command causes the Mission Manager to abort the current waypoint efforts and slow to a hover until given further instructions. A "begin" command instructs the Mission Manager to resume the mission where it left off, or to begin the mission if it has not already done so. An "abort" command causes the Mission Manager to immediately abort the current waypoint and calculate new waypoints that will take the vehicle vertically to a safe altitude. From there, it will fly to the rally point, and then descend vertically to the rally altitude and wait for further instructions.

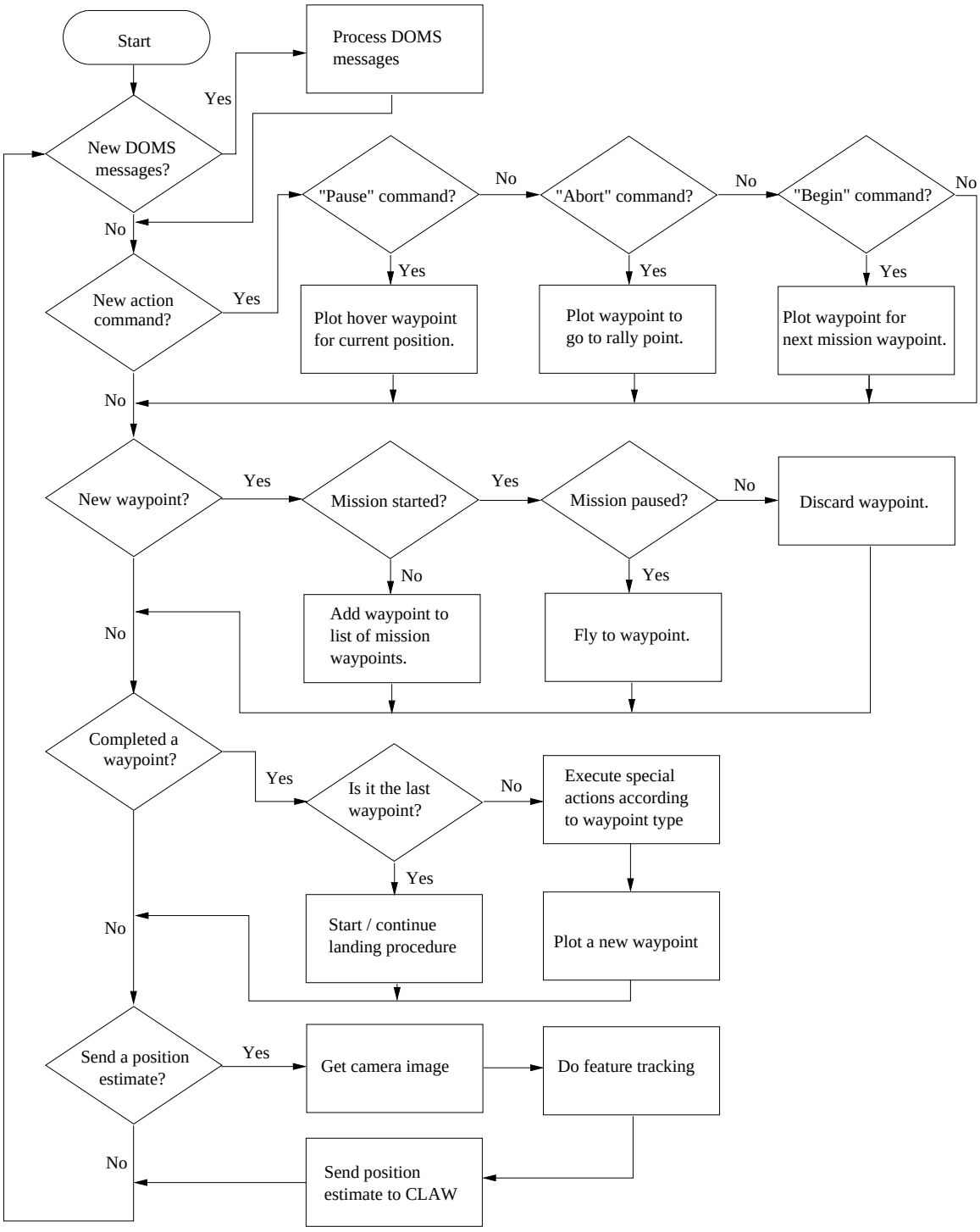



Figure 3.8: Flowchart illustrating the basic architecture of the Mission Manager.

Table 3.1: DOMS messages subscribed to by the Mission Manager.

| **Message Name** | **Description and Purpose** |
|---|---|
| Dynamic State | Contains vehicle position and attitudes from CLAW. Attitudes used in SLAD and position estimation. Positions used in the calculations of some intermediate waypoints during the landing procedure. |
| Mission Manager Waypoints | Contains mission waypoint information from the ground station. These messages define the mission that the Mission Manager will fly. |
| Mission Manager Autonomy | Message from the ground station indicating the level of autonomy desired by the PMO. Ground station allows PMO the option of telling the Mission Manager to pause at key points in the mission to offer the PMO a chance to approve some decisions made by the Mission Manager. |
| Fuzzy Logic Variables | Contains variable values that are used in the fuzzy-logic heading optimization (to be discussed shortly). Mission Manager initialization involves setting default values for these variables. This message is a provision for changing these default values. |
| Mission Manager Commands | Message from the ground station to tell the Mission Manager to begin, pause, or abort the mission. |
| Waypoint Achieved | Message from CLAW indicating that the aircraft has arrived at a requested waypoint. |
| Laser Range | Message from a process responsible for interfacing with the laser rangefinder. |

Table 3.2: DOMS messages published by the Mission Manager.

| **Message Name** | **Description and Purpose** |
|---|---|
| Position Estimation | Message sent to CLAW containing a GPS position estimate. |
| CLAW Waypoint | Message sent to CLAW containing waypoint data. Requests CLAW to fly to the given waypoint. |
| CLAW Heading | Message sent to CLAW containing heading data. Requests CLAW to assume a new heading. |
| Mission Manager Waypoint | Contains mission waypoint information. Sent to the ground station to notify it of the current mission status. |
| Fuzzy Logic Variables | Contains variable values that are used in the fuzzy-logic heading optimization (to be discussed shortly). Message sent to the ground station to notify it of heading optimization results. |
| SLAD Results | Message broadcast to alert other processes to availability of new SLAD results. In the future, this message may be used to send intermediate SLAD results to the ground station for display. |

Once any new action commands have been processed, the main loop handles new requests for mission waypoints. If the mission has not yet begun, the waypoints are added to the list of waypoints that define the mission. If the mission has commenced but has been paused by the PMO, then the ground station provides the PMO with primitive controls for manually maneuvering the aircraft. The ground station converts the PMO's inputs into waypoints and sends them to the Mission Manager. Since the mission has been paused, the Mission Manager recognizes that these waypoints are not part of the pre-planned mission, and simply relays them to CLAW. If new waypoints are received under any other circumstances, they are discarded since they are not pre-planned mission waypoints or valid PMO maneuvering requests.

The fourth step in the main loop, checking for recent completion of a waypoint, is what drives the mission. When the mission begins, the Mission Manager requests CLAW to fly to the first waypoint. Upon arrival, CLAW replies with a message that the waypoint has been completed. This is the Mission Manager's cue to decide what should happen next. Based on the Mission Manager's recollection of the instructions

for the waypoint that has just been completed, it may perform a specific action, such as point the cameras (i.e., the entire aircraft) in a certain direction and hover for a given amount of time, or it may simply continue without hesitation to the next waypoint. One more important possibility is that the Mission Manager has reached the last waypoint, in which case it will begin the landing procedure.

### 3.3 Landing Procedure

The main focus of this thesis is the application of the vision algorithms to the landing portion of an RUAV mission. Since this is the most difficult part of the mission, it is described in detail in this section. The landing procedure starts with the vehicle at about 30 meters directly over the nominal landing point. At this point, the Mission Manager invokes the landing procedure, which executes the following steps. These steps are summarized by the flowchart shown in Figure 3.9.

1. The vehicle flies to a specified height above the ground and orients the aircraft so that the nominal landing point appears in the center of the camera images. This allows the PMO to review the landing site. Before this can be done, the Mission Manager must determine the aircraft's current height above the ground. Assuming the laser range $R$ (see Figure 2.14), camera pitch $\theta_c$, and aircraft roll and pitch $\theta$,$\phi$ are known, the current height above ground $H$ is calculated as

$$H = R\cos\left(\frac{\pi}{2} + \theta_c + \phi\right)\cos\theta \tag{3.1}$$

   Note that this calculation is based on the assumption that that the ground is locally flat (i.e., the ground directly underneath the helicopter is at the same altitude as the point detected by the rangefinder). Next, given the current heading $\psi$ and northing and easting coordinates $x, y$ of the nominal landing point, new waypoint coordinates $x_n, y_n$ may be calculated to put the aircraft in a position so that the landing point is in the center of the image:

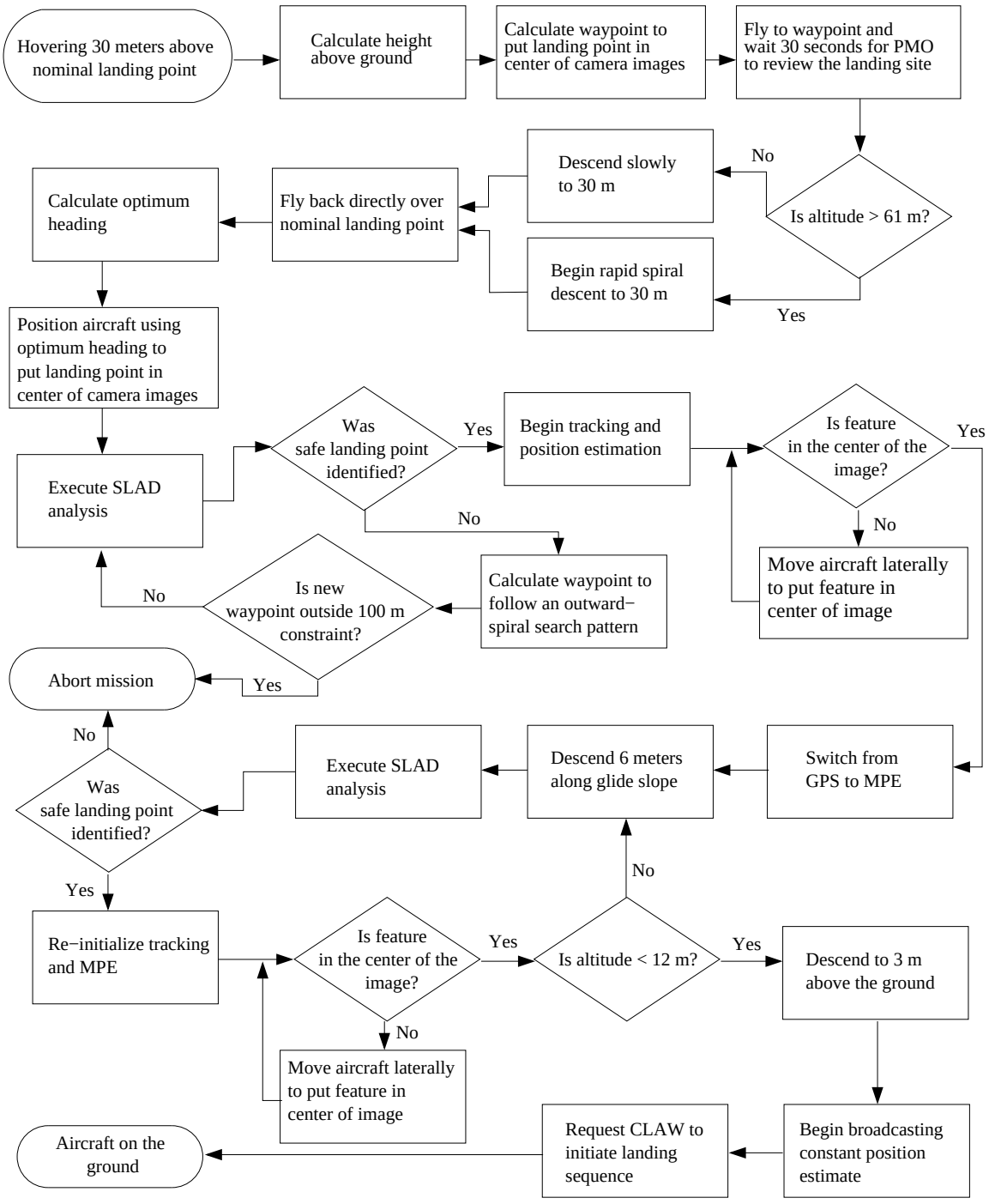



Figure 3.9: Flowchart summary of the landing procedure.

$$x_n = x - H \sin\psi \tan\left(\frac{\pi}{2} + \theta_c\right) \tag{3.2}$$

$$y_n = y - H \cos\psi \tan\left(\frac{\pi}{2} + \theta_c\right) \tag{3.3}$$

After orienting the aircraft, hover for 30 seconds to allow the PMO an opportunity to reject the landing site or take some other course of action. If no input is received, the Mission Manager will automatically continue with step 2.

2. Fly back directly over the nominal landing point. If the aircraft is more than 61 meters (200 ft) above the ground, then continue with the next step to begin a rapid spiral descent. Otherwise, descend vertically to 30.5 meters (100 ft) above the nominal landing point and skip to step 4.

3. Perform a rapid spiral descent down to roughly 30 meters above the ground. Since rapid vertical descent in a rotorcraft can be problematic, a series of waypoints are plotted in a general spiral pattern. The vehicle flies this pattern down to 30.5 meters (100 ft), and then adjusts its position laterally until it is directly over the nominal landing point.

4. Calculate the heading of the aircraft that will lead to optimum vision algorithm performance. Factors such as wind direction, turbulence, and sun angle are considered to keep the RUAV shadow out of the field of view of the cameras and to minimize the influence of turbulent disturbances. The effects of turbulence and wind direction will be discussed in Chapter 4. Use the new heading calculations to position the aircraft 30.5 meters above the ground where the nominal landing point will appear in the center of the camera images (see step 1).

5. Identify a safe place to land by executing the following steps:

   (a) Acquire new images from the stereo cameras.

   (b) Apply the stereo ranging algorithm to generate a range map.

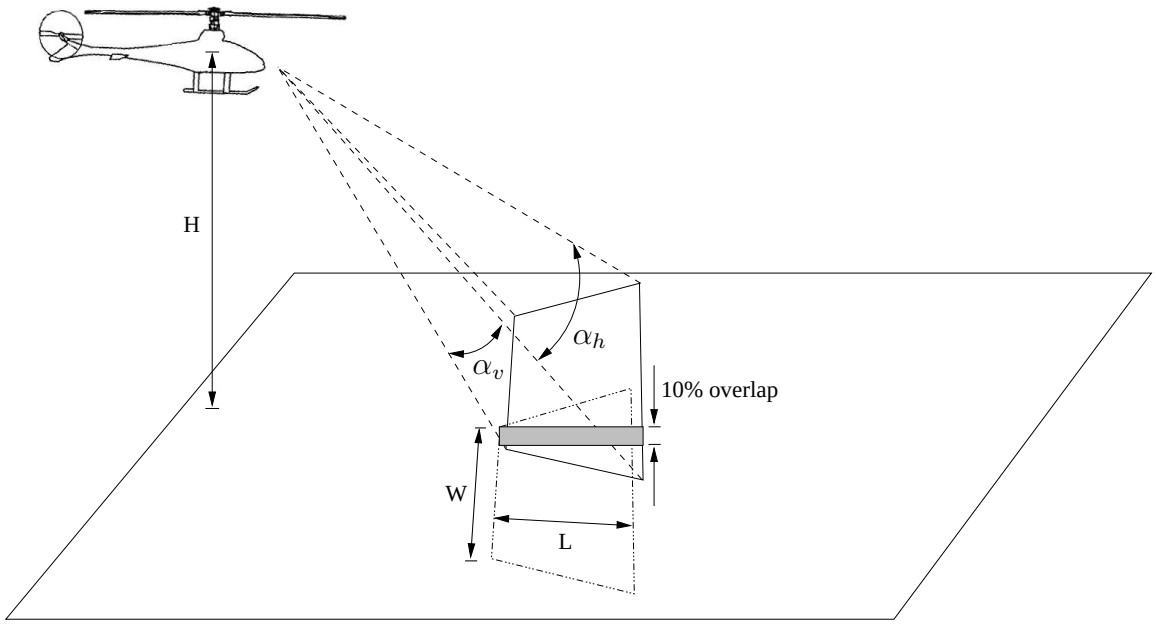



Figure 3.10: Illustration of parameters used in calculations to ensure 10% overlap of the images when surveying the terrain.

(c) Execute a SLAD analysis of the range map. If a safe landing point is found then skip to step 6 in the landing procedure.

(d) If a safe landing point has not been identified, then search in an outward-spiral pattern until one is found, or until the search pattern exceeds a radius of 100 meters from the nominal landing point and the mission is aborted. The search is done by re-checking the current height above ground $H$, and then calculating the amount of ground that is covered in the camera image width $w$ and height $l$ (see Figure 3.10 and Equations 3.4 and 3.5).

$$w = 2H \tan\left(\frac{\alpha_h}{2}\right) \cos\left(\frac{\pi}{2} + \theta_c - \frac{\alpha_v}{2}\right) \tag{3.4}$$

$$l = H\left[\tan\left(\frac{\pi}{2} + \theta_c + \frac{\alpha_v}{2}\right) - \tan\left(\frac{\pi}{2} + \theta_c - \frac{\alpha_v}{2}\right)\right] \tag{3.5}$$

In Equations 3.4 and 3.5, $\alpha_h$ and $\alpha_v$ represent angles for the horizontal and vertical camera field of view, which take on different values in our

simulation environment, but which are usually nearly equal for most real cameras. The resulting distances $w$ and $l$ are applied to calculate a new waypoint that will move the vehicle in the appropriate lateral direction so that the camera images at the new location will have a 10% overlap with images used in the previous SLAD analysis.

6. Prepare to descend toward the safe landing point that was selected in step 5 by doing the following:

   (a) Request the pixel coordinates in the left camera image that represent the selected safe landing point.

   (b) Use the pixel coordinates to initialize the position estimation algorithm.

   (c) Initialize the monocular tracking algorithm.

   (d) Begin feature tracking and broadcasting position estimates at the rate of 10 Hz.

   (e) If the landing point is not already in the center of the image, then reposition the aircraft laterally. This extra effort is desirable to encourage compliance with the assumption that the altitudes of the tracked landing point and center-of-image feature are equal (see Section 2.3.2 on MPE derivation).

7. Send a message to CLAW to switch over to the second Kalman filter to begin flying based on the position estimates it has been receiving, instead of relying on GPS.

8. Plot a waypoint 6 meters (20 ft) lower along a glide slope to keep the selected landing point in the center of the camera image.

9. Verify the safety of the landing point and prepare for continuing the descent by doing the following:

   (a) Acquire new images from the stereo cameras.

(b) Apply the stereo ranging algorithm to generate a range map.

(c) Execute a SLAD analysis of the range map. If no safe landing point is found, then abort the mission.

(d) Re-initialize the tracking and position-estimation algorithms using the new landing point.

(e) Continue tracking and broadcasting position estimates at 10 Hz.

(f) If the landing point is not already in the center of the image, then re-position the aircraft laterally.

10. Repeat steps 8 and 9 until the aircraft is less than 12 meters (40 ft) above the ground. Since the stereo methods have sufficient resolution from 12 meters to make a good final-landing decision, it is not necessary to do another SLAD analysis at a lower altitude. Instead, plot a waypoint to descend along the glide slope down to 3 meters (10 ft) above the ground.

11. When the vehicle reaches 3 meters, begin broadcasting a constant position estimate using the most recent estimate.

12. Since the CLAW landing sequence relies on a sonar that is only accurate under 2 meters, then plot another waypoint to descend vertically 1 meter and send a request for CLAW to initiate its landing sequence.

Figure 3.11 displays a simple summary of the steps that have been outlined for the landing procedure. In the event that it is necessary to abort the mission after step 7, it is not an option to simply discontinue the landing procedure and fly to the rally point. Since navigation is dependent on tracking the landing point, and since GPS is not available after step 7, the landing procedure must be reversed to take the aircraft back up to the point where the GPS signal was abandoned. Only at that point will the aircraft be allowed to re-acquire the GPS signal and fly to the rally point.

Two details of the landing procedure warrant some additional explanation. The first is concerned with action (d) under step 9 in the landing procedure. After

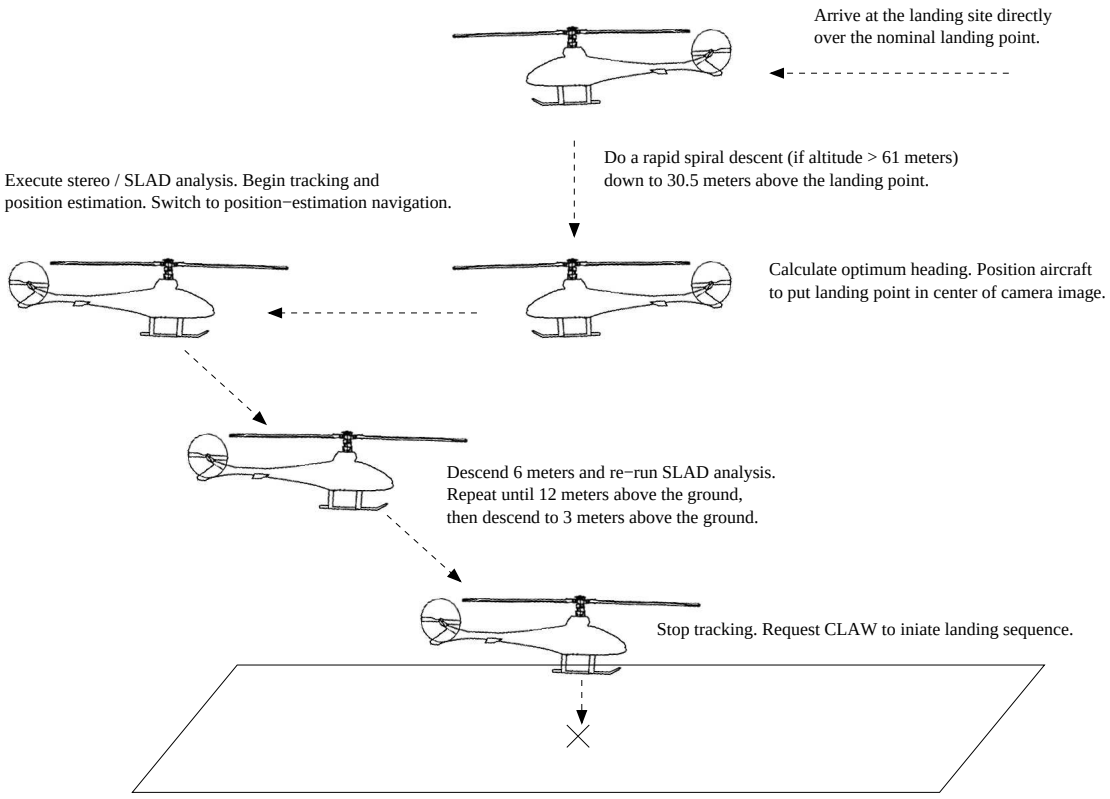



Figure 3.11: Summary of the landing procedure.

descending and re-running the SLAD routine to verify the safe landing point, it is possible that SLAD will select a different point than the one that was chosen previously. This is due to the fact that camera images will provide greater resolution at lower altitudes, which may make it possible for SLAD to detect hazards that were previously not discernible. However, if the tracking algorithm suddenly discards the old feature and begins tracking the new feature that was selected by SLAD, then a discontinuity will result in the MPE output, which in turn may provoke control efforts that can lead to abrupt aircraft motion. To avoid this problem, the MPE algorithm must be re-initialized by estimating the position of the new SLAD landing point. Since GPS is not available at this point, the position of the landing point is estimated relative to the current estimated position of the aircraft. Future vehicle position estimates would then be based on the estimated position of the new landing point, which eliminates the potential for discontinuities in the MPE output.

The second detail that needs further explanation is step 11 in the landing procedure. The need for this step is illustrated by pointing out that at a high altitude, the camera image captures a large amount of terrain, and changes in aircraft roll and pitch will only cause an image feature to move a few pixels in a set of chronological images. At lower altitudes, the camera's view will be restricted to much less terrain, and turbulence may cause an image feature to leave the image. As the aircraft descends while tracking the landing point, it will descend to a point where even small aircraft attitude changes will cause the image feature in one frame to move outside of the search window in the next frame, and the feature will be lost.

Section 4.2 discusses test results that reveal that this lower altitude limit will be found at roughly 1.8 meters above the ground for mild wind conditions, and about 2.5 meters in extreme wind conditions. Position estimation becomes unreliable below these points, so the simple solution was to broadcast a constant position estimate after the vehicle descends below 3 meters. Since CLAW uses a sonar for the touch-down sequence, it was made to base this last part of the descent on the altitude from the sonar reading, and only pay attention to the horizontal position from the Mission Manager position estimate. Because the horizontal position is reported to be constant

below a height of 3 meters, it is expected that the aircraft will drift to the side during the descent. But it is assumed that this drift will be minimal since this method will only be necessary for the last couple meters of the descent.

### 3.4 Simulation Implementation Details

The complete simulation environment and methods that have been presented in the previous section were successfully applied to demonstrate a fully-autonomous RUAV mission in simulation. When obstacles were added to the RIPTIDE parking lot scenario, the Mission Manager identified the hazards and guided the aircraft down to a safe landing point. This achievement encompasses several changes to remove simplifications that Hintze had made to make the vision algorithms work.

All of Hintze's work with the vision algorithms had been done with simulation camera images using a resolution of 640×240 pixels. This resolution was chosen by halving the height dimension of the preferred resolution (640×480 pixels) to help reduce CPU load incurred by rendering the stereo images. The irregular image resolution theoretically would not have any adverse effects on the functionality of the stereo or tracking algorithms, but it was decided, for the work of this thesis, to return to a 640×480 resolution since this is what would eventually be used in the final hardware PALACE demonstrations. It was reasoned that the benefits of a more realistic simulation outweighed the costs of slowing down RIPTIDE graphics rendering. In addition to making this change in the RIPTIDE stereo cameras, the simulation camera CAHVOR model was also updated to reflect the new camera resolution.

One other significant improvement in the simulation was related to an aircraft instability that often occurred when CLAW was instructed to switch over to using the Mission Manager position estimates for navigation, instead of the usual GPS signal. This instability had been a long-standing problem that was recognized early on by Hintze, who temporarily resolved the problem by changing CLAW to use a different set of aircraft control law gains and Kalman filter gains. These changes had several side effects, the most significant one being a reduction in the responsiveness of the aircraft. Because these changes included removing the integrator effects of the control

laws, it was also necessary to disable the simulation wind and gust model. Otherwise, the aircraft would not achieve a waypoint under conditions of steady disturbances. Finally, Hintze disabled all sensor and model noise, which resulted in marginally-stable navigation using the MPE output.

As the instability issue continued to resurface occasionally in the simulation implementation that was outlined in this chapter, a series of tests were carried out to help isolate the root cause of the problem. A careful side-by-side comparison of one set of data provided an essential clue. Figure 3.12 displays data that was recorded from one test where oscillations were elicited in the aircraft roll during hover. As shown in Figure 3.13, a 0.25-second delay was observed between the peaks of the aircraft roll oscillations and the peaks of the image pixel $x$-coordinate of the feature that was being tracked.

Assuming that the tracker was perfectly tracking a feature in the image, then the variation in the $x$-coordinate of the pixel over time should have been a direct result of, and therefore synchronized with, the aircraft roll behavior. This lack of synchronization was proven to be the cause of the instability by programmatically introducing a comparable delay in the roll angles that were used in calculating the position estimates, thereby manually synchronizing the two curves and then broadcasting a position estimate that was slightly delayed. The resulting performance was not ideal, due to the delay in the position estimate output, but the unstable behavior disappeared.

Further tests identified the source of the delay in the fact that the CPU was heavily-loaded, which was limiting RIPTIDE to only producing stereo images at 5 Hz. Furthermore, timing tests revealed that there was an average delay of 0.13 seconds between the time that CLAW broadcast aircraft attitudes and the time that the Mission Manager received them from DOMS and used them to calculate a position estimate. It was expected that another 0.13 seconds would be added to the delay as the Mission Manager sent CLAW a position estimate through DOMS so that CLAW could use the estimate to produce a control effort. Finally, it was noted that a similar delay would separate the graphics rendering in RIPTIDE (i.e., the two 640×480 stereo

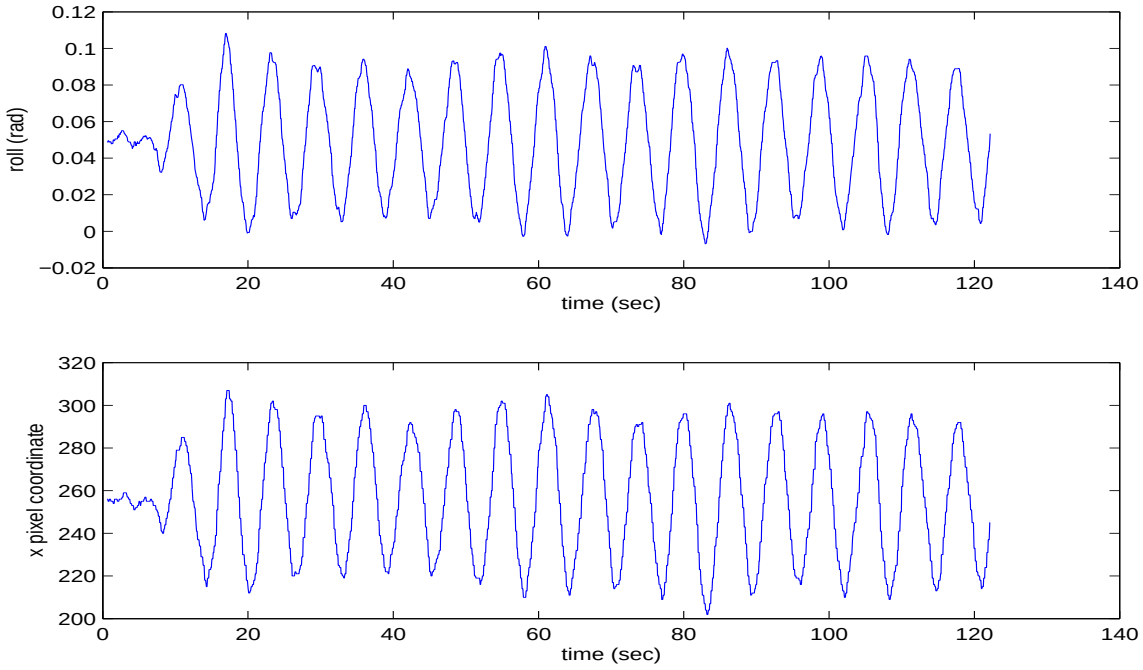



Figure 3.12: Test data showing variations in aircraft roll angle over time. This is compared to the tracking algorithm output of pixel $x$-coordinates over time.

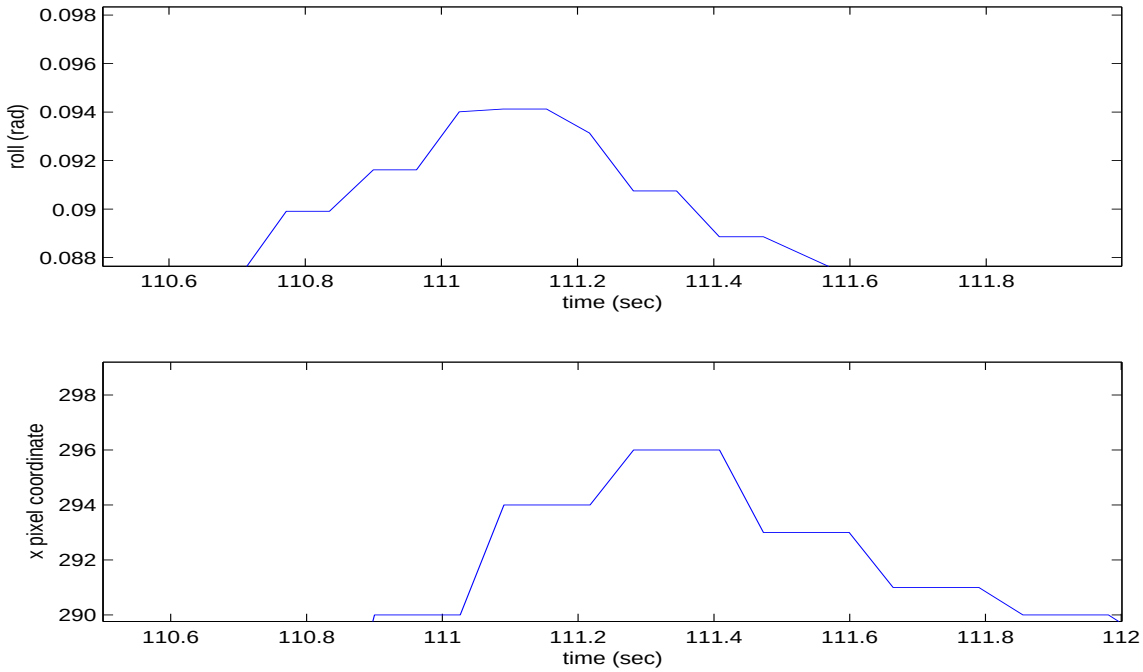



Figure 3.13: Zooming in on one of the peaks from Figure 3.12.

images) and the true state modeled in CLAW because a third process was being used to transfer CLAW vehicle states from DOMS messages to RIPTIDE shared memory.

Since stereo images were being rendered at 5 Hz and position estimation was being done at 10 Hz, part of the problem was that old images were being combined with more-recent aircraft attitude information to result in very slightly off-nominal position estimates. The control efforts that CLAW generated to correct the perceived error were delayed and actually contributed to any error instead of reducing it. This lack of synchronization led to oscillations in roll and pitch whose magnitudes grew quickly until the aircraft spun out of control and/or the limits of the math model were breached.

The solution to this problem involved several parts. First, collaboration with the RIPTIDE author led to improvements that helped RIPTIDE run more efficiently. The reduced CPU load in turn decreased the one-way DOMS message-transfer delay. Next, a CLAW parameter was adjusted so that aircraft attitude data would be sent to the Mission Manager at a higher rate, thus reducing the delay between when the attitude is measured and when it is used to calculate a position estimate. Finally, an intermediate process was eliminated, along with the associated DOMS communication delays, by changing CLAW to communicate directly with RIPTIDE through shared memory so that the graphics rendering would more closely reflect the model states.

These efforts not only resolved the instability issue, but also permitted a return to the same control law gains that are used during RMAX helicopter flights. The original control law gains provided renewed aircraft responsiveness in simulation, and allowed for re-enabling the wind and gust models. In addition, the model and sensor noise was restored without introducing any adverse effects in MPE navigation. Finally, the need for a second Kalman filter was removed, which increased confidence in the tracking and position-estimation algorithms.

# Chapter 4

## Simulation Evaluation Results

The focus of the PALACE project from the beginning has been placed on coordinating the landing phase of an autonomous RUAV mission. Although many factors have the potential for contributing to the success or failure of a PALACE autonomous landing, the most important components are the machine vision algorithms. The performance of the landing system as a whole will be greatly influenced by the performance of the individual vision-based algorithms.

This chapter presents the results of the testing that was done on each algorithm with the aid of the RIPTIDE simulation environment that was described in the previous chapter. The discussion begins with the details of the tests conducted on the stereo-ranging / SLAD algorithm, as well as a listing of potential restrictions imposed by this software. Attention will then be turned to the tracking and position-estimation algorithms. Since the tracking and position-estimation algorithms are closely connected, they will be discussed simultaneously.

Simulation testing conditions were established to define the scope of the tests, with each test motivated to identify optimum operating conditions as well as conditions that might compromise the success of the mission. Finally, metrics were constructed to facilitate an objective evaluation of the results from each test. Limited tests were also conducted on the RMAX helicopter hardware to confirm the validity of the simulation test results, but a presentation of these will be postponed until Chapter 5.

## 4.1 Stereo Ranging / SLAD Evaluation

The ultimate output of the SLAD analysis is very simple - either an $x, y$ pixel coordinate representing a safe landing point, or a declaration of the lack of a safe landing point. As illustrated in Chapter 2, the computations leading to this output are very complex and exhibit sensitivity to a number of factors. In order to prove the capabilities of this algorithm, the first step was to focus on the intermediate data. If the range map did not accurately represent the terrain, then the SLAD output may not be the optimum, or even a valid, landing point. Next, tests were conducted under a variety of conditions, including variations in obstacle density, camera angle, image pyramid level, and height above ground. The results of these tests are summarized and analyzed below.

### 4.1.1 Image Texture

The stereo algorithm is based on the assumption that a feature in one camera image can be matched with the same feature in the second camera image. It is clear that this assumption breaks down in situations where there are large regions of pixels that are indistinguishable. Figures 4.1, 4.2, and 4.3 show some examples of scenes in simulation that contain varying degrees of texture. Each camera image is accompanied by an overhead view of the resulting stereo range map, on which the original camera image is overlaid for reference.

The black holes in the range maps indicate a lack of range data due to non-unique patterns of pixel intensities in those regions of the original images. In other words, the features in those regions of the two images could not be correlated, which made it impossible to obtain 3D information for those pixels. Since nothing is known of these regions, they must be treated as regions of hazardous terrain. Some images, such as the image with the car (Figure 4.3), may contain sufficient range data to identify a safe landing point. The other images (Figures 4.1 and 4.2) do not contain sufficient range data due to the lack of texture in the original scene. These results indicate that careful attention to providing good textures in simulation is required to

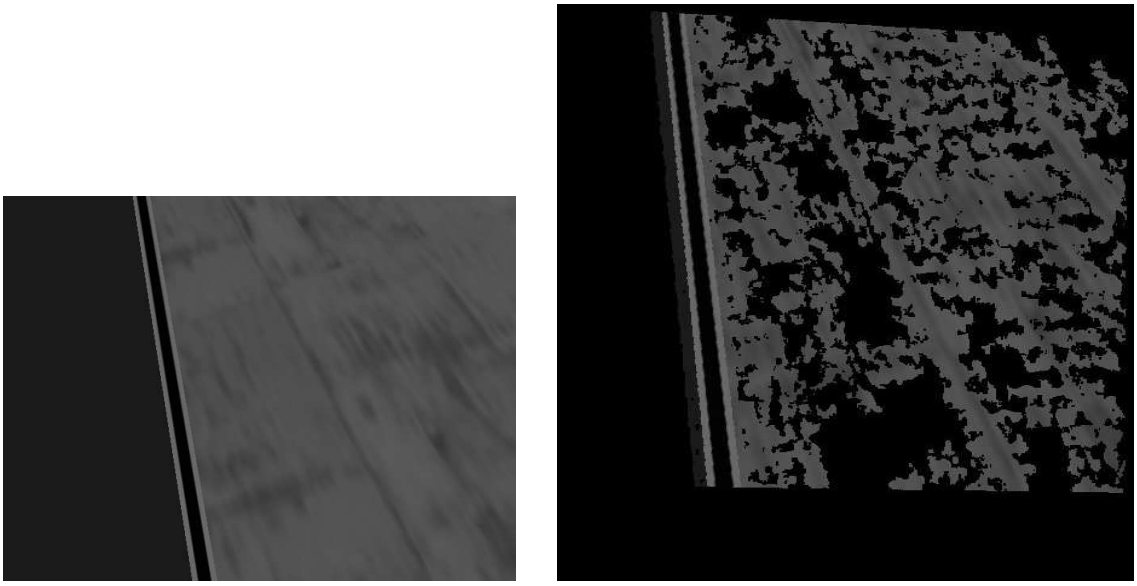


Figure 4.1: Left camera image and corresponding range map of a runway in simulation.

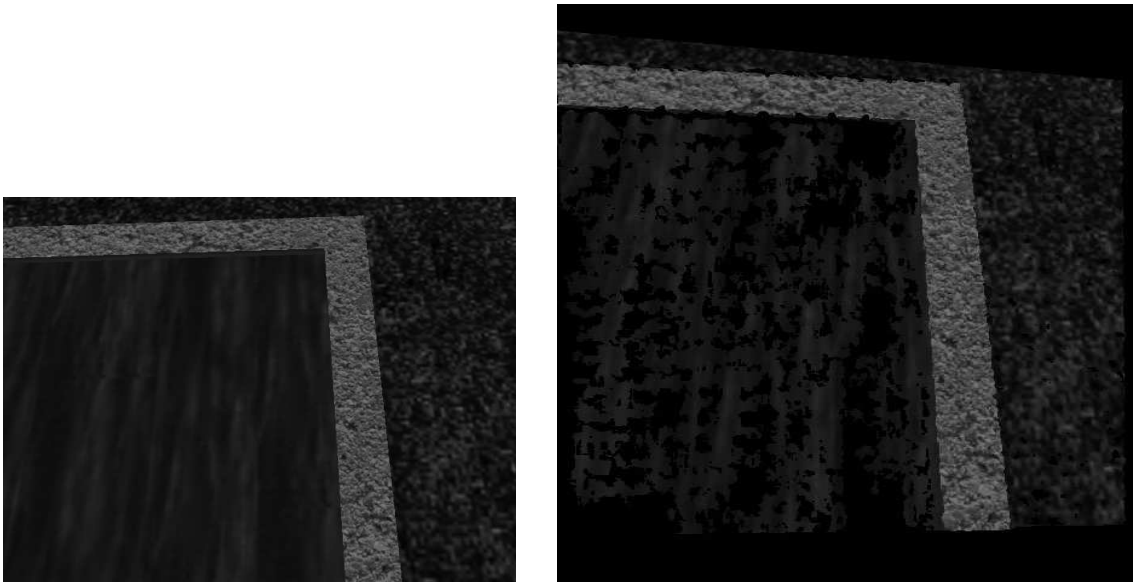


Figure 4.2: Left camera image and corresponding range map of a simulation water fountain.

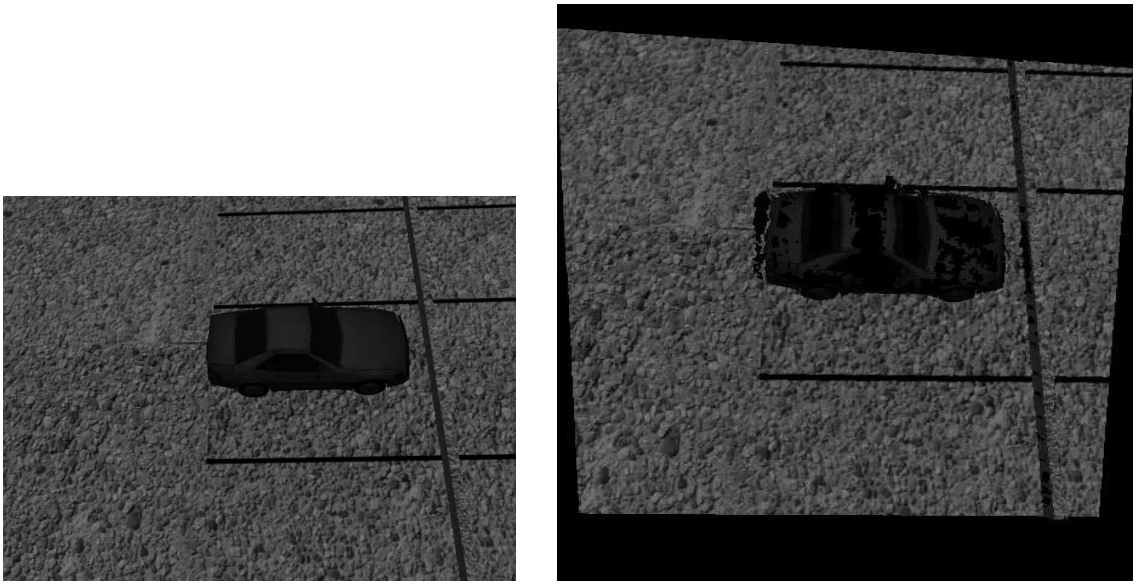


Figure 4.3: Left camera image and corresponding range map of a simulation parking lot.

get meaningful results from the stereo algorithms. The surface texture will also be important when the stereo vision algorithms are used in flight.

### 4.1.2 Range Map Accuracy

The first test of the stereo ranging algorithm was conducted to quantify the accuracy of the range map representation of the terrain. This test consisted of hovering at about 10 meters and pointing the cameras (30 degrees from vertical) at a box sitting on level ground about 12 meters away from the camera. Figure 4.4 contains a sample test image showing the box in the top-middle of the image, and a second inconsequential box on the side. A stereo analysis was executed with the box height set to values ranging from 5 cm to 45 cm in 5 cm increments. For each case, image pyramid level 1 was used (see Section 2.1.2), meaning that the camera image was reduced by a factor of 2. Figures 4.5 and 4.6 show representative range maps resulting from images of boxes at 15 and 45 cm.

In Figure 4.5, the side view of the range map illustrates the variation in the range data. In this figure, it can be seen that some obstacle is present, but it is

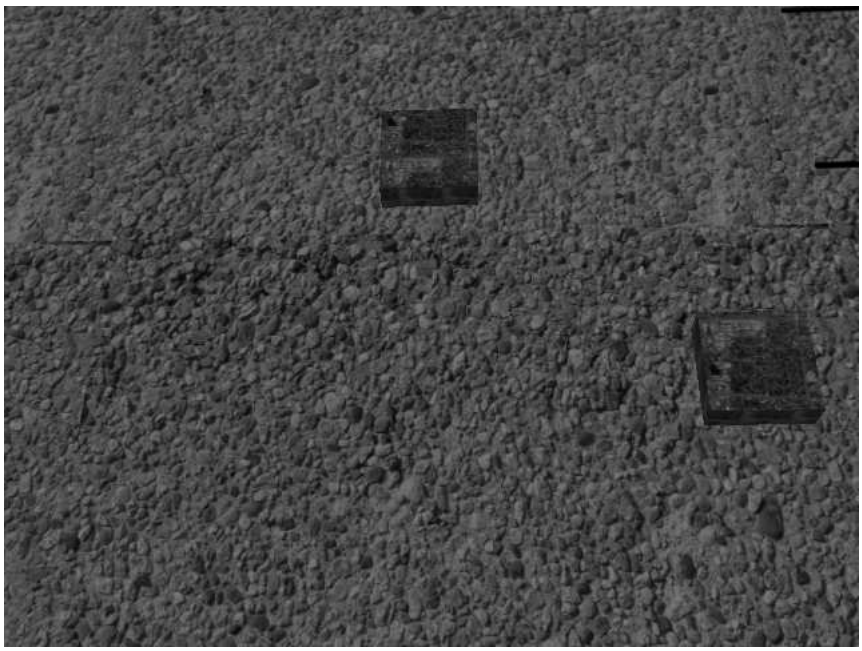


Figure 4.4: Sample image from tests to investigate range-map accuracy.

barely discernible from the variation in the surrounding points that represent the flat ground. In contrast, the presence of a taller box is more obvious, as shown in Figure 4.6. These images make it clear that the stereo ranging algorithm is successful at representing the general quality of the terrain with a resolution of about 15 cm from a distance of 12 meters.

To get an idea of the variation in the range data, a quick program was written to overlay the original camera image on the range map. The mouse was used to select a small square region of the image and the program would read all of the data from the corresponding region of the range map to report the maximum, minimum, and average elevations. This method was used to record the average elevations of the top of the box and a neighboring piece of ground. The height of the box was estimated as the difference between these two values. The actual height of the box was compared to the stereo estimate of the height, and the results are presented in Figure 4.7.

In this figure, the solid line represents the average elevation from the stereo analysis; the dashed line represents perfect estimates; and the error bars represent the minimum and maximum elevations that were reported from the range data. The

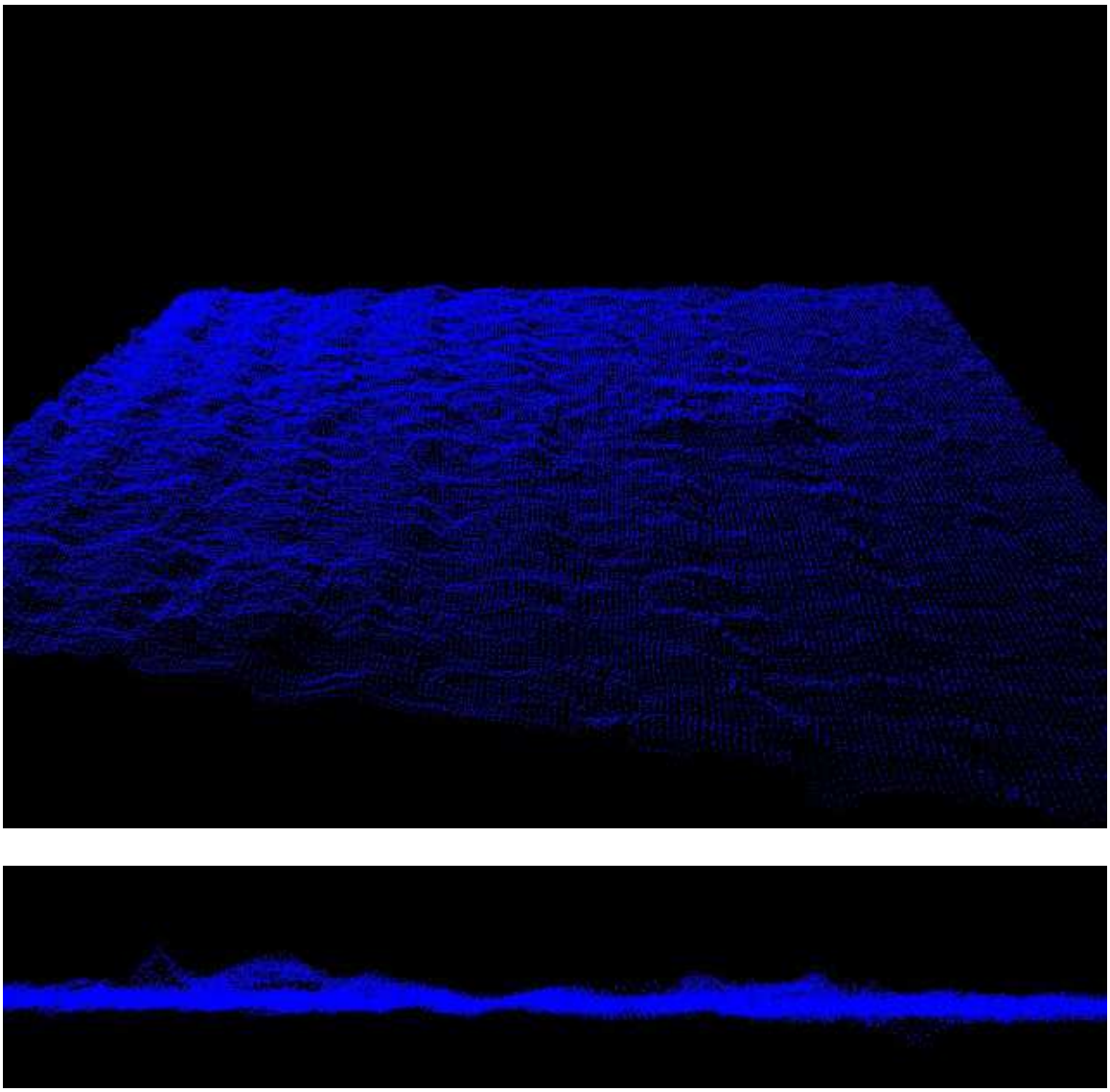


Figure 4.5: Top and side views of range map representing box height of 15 cm, based on an image reduction of 50%.

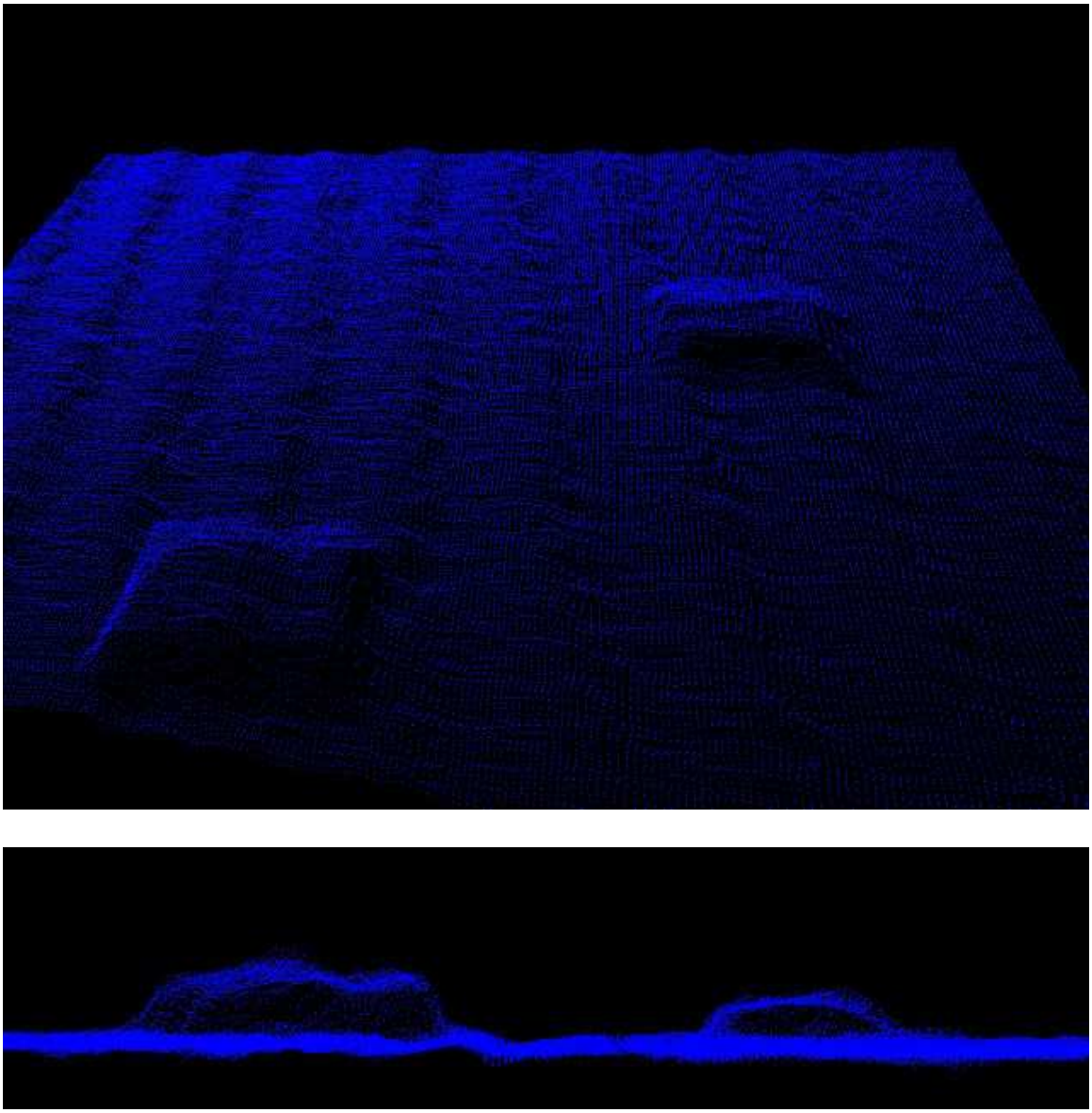


Figure 4.6: Top and side views of range map representing box height of 45 cm, based on an image reduction of 50%.

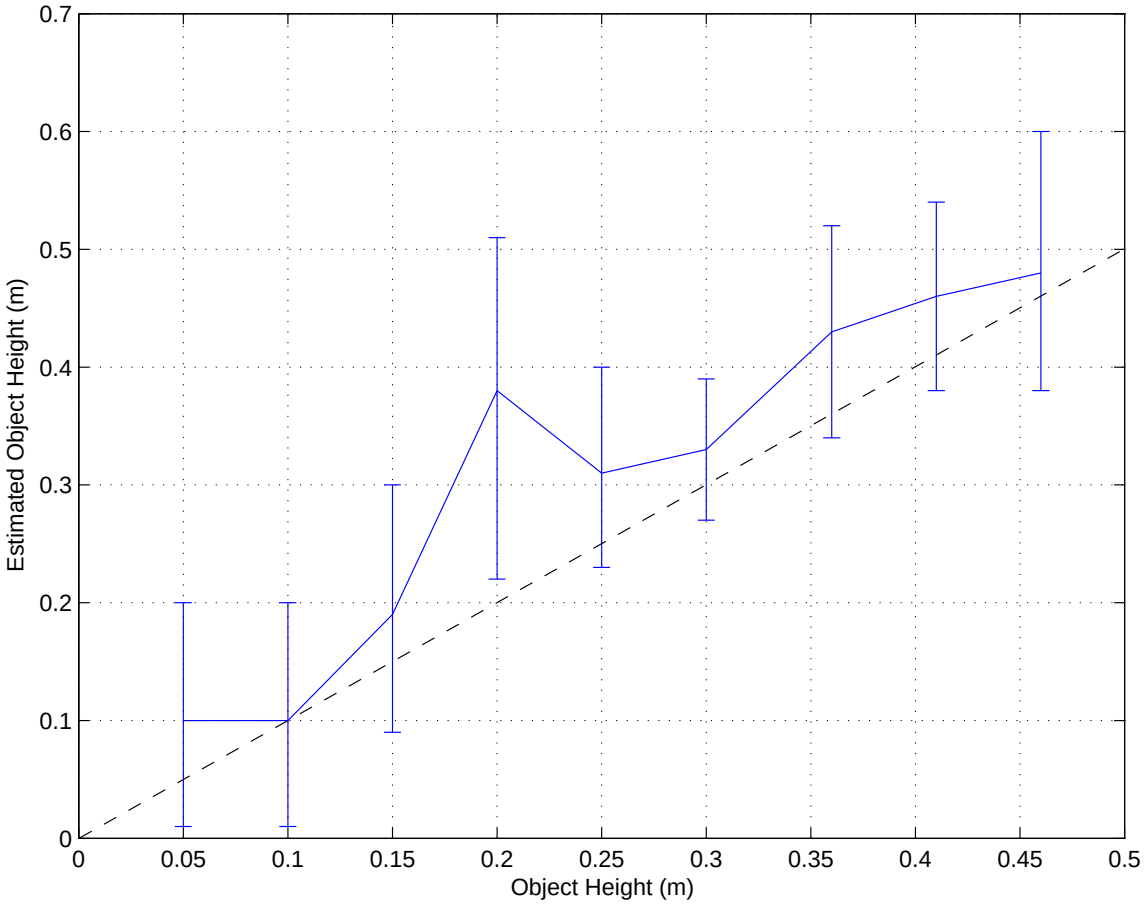



Figure 4.7: Test results for stereo estimation of box heights with pyramid level 1.

first thing that can be observed from the plot is that the box-height estimates are fairly accurate considering that increments of 5 cm are being measured from 12 meters away. Furthermore, if objects taller than 15 cm constitute a landing hazard, then the plot indicates that the range maps will represent the hazard with sufficient accuracy that the SLAD algorithm will be able to recognize the hazard. Finally, it should be noted that there is a point at which the variation in the range data for the flat ground is comparable to the height of an object sitting on the ground. In this case, the limit is around 10 cm, below which smaller objects will be completely indistinguishable from the ground.

In order to illustrate the effects of using pyramid level 0, the same tests were repeated with the only difference being that the images were not reduced before running a stereo analysis. These results are displayed in Figures 4.8 and 4.9.

Figure 4.8 shows that boxes as small as 10 cm are distinct from the ground when pyramid level 0 is used, whereas only objects larger than 15 cm could be distinguished with pyramid level 1 (Figure 4.5). Furthermore, by comparing the 15 cm box in Figure 4.5 to the same box height in Figure 4.9, it is clear the pyramid level 0 produces a much more crisp range map. These results are reflected in Figure 4.10 where the plot of minimum, maximum, and average elevations are more consistent and are characterized by 6% less variation on average.

The superior data quality from pyramid level 0 comes at the cost of processing time. The previous tests with pyramid level 1 only consumed about 2 seconds of processing time (3.3 GHz Pentium 4), whereas pyramid level 0 required 6 seconds. It was decided that all of the remaining stereo/SLAD tests would be conducted using pyramid level 1. This will give a more reasonable picture of the performance that will be available in flight since the RMAX helicopter flight computers are much less powerful than the desktop computer used for these tests in simulation. In addition, it may be questionable whether the increased accuracy is necessary since SLAD does not need to know the exact obstacle dimensions. It is only necessary to determine the presence of an obstacle so that it can be avoided.

These tests raised confidence in the stereo method's ability to recognize small hazards. The next step was to verify that the range maps accurately represent the locations of these hazards relative to the vehicle. A test was conducted similar to the previous test, except that the helicopter was moved to a different altitude before each stereo analysis and the box size was held constant. The cameras were pointed straight down at the box (see Figure 4.11 for a sample image), and the distance between the camera and the box was calculated by subtracting their GPS coordinates. This value was compared to the maximum, minimum, and average range to a small area in the middle of the box. The results can be found in Figure 4.12.

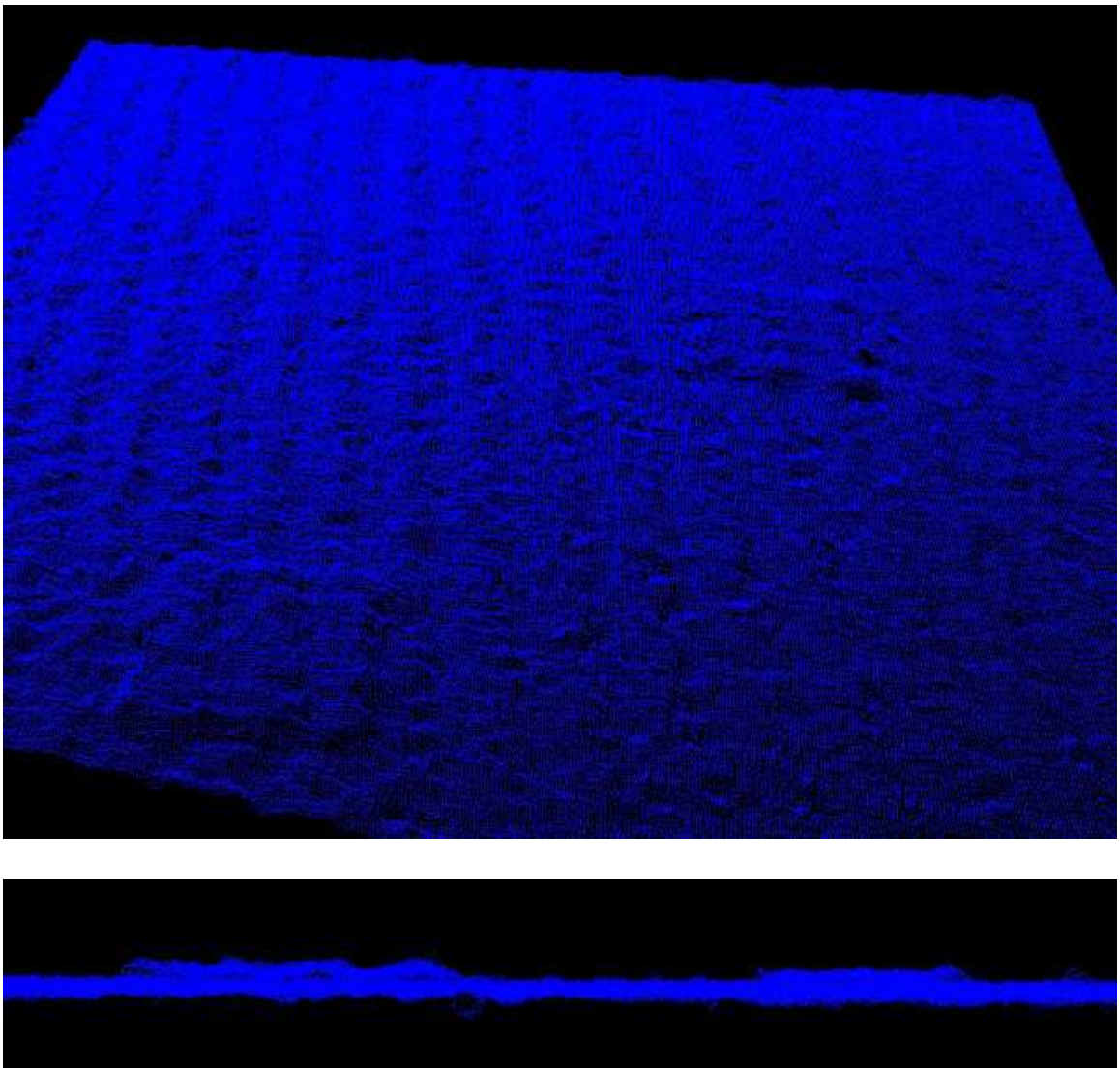


Figure 4.8: Top and side views of range map representing box height of 10 cm, based on the original image (no reduction).

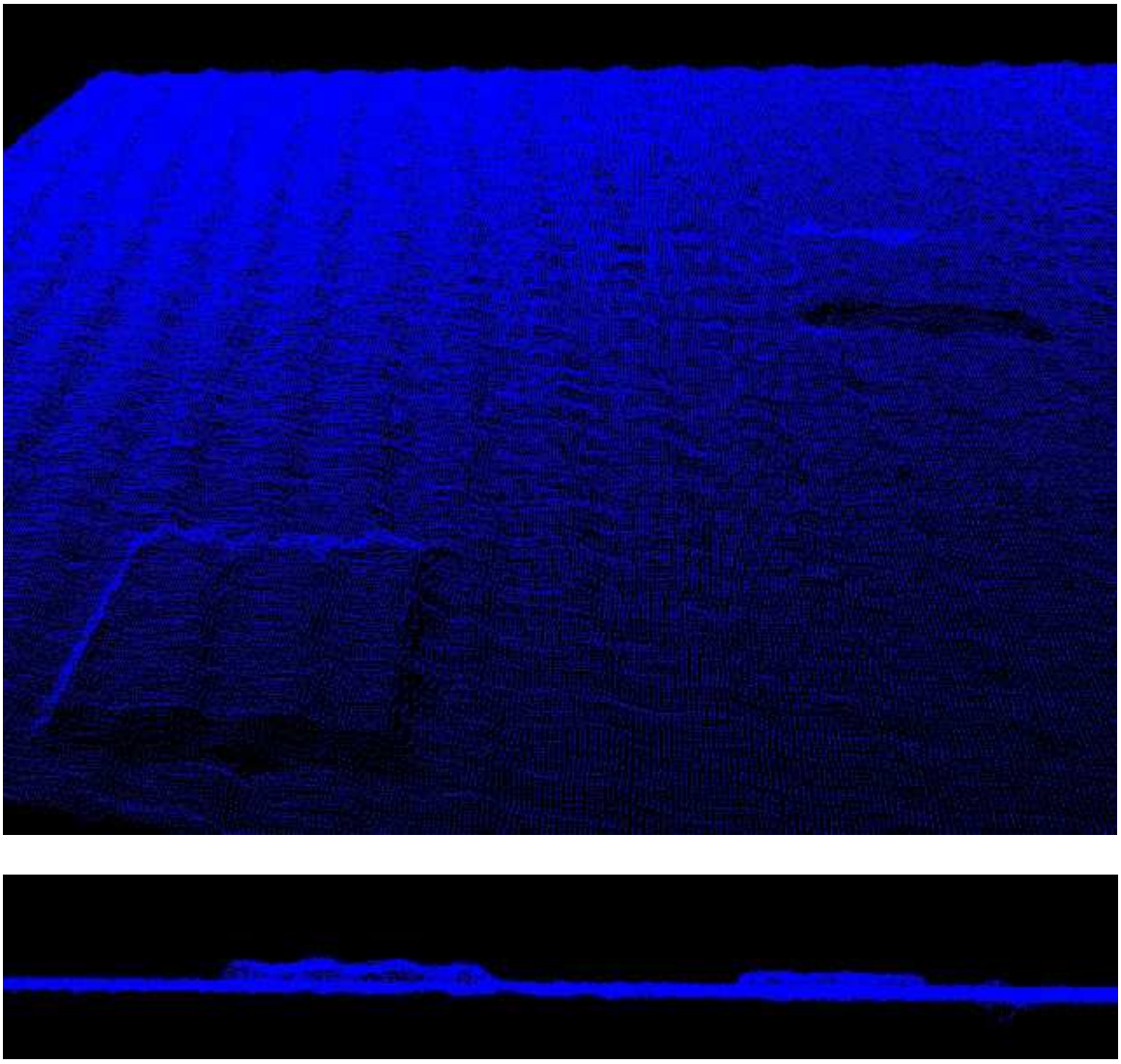


Figure 4.9: Top and side views of range map representing box height of 15 cm, based on the original image (no reduction).

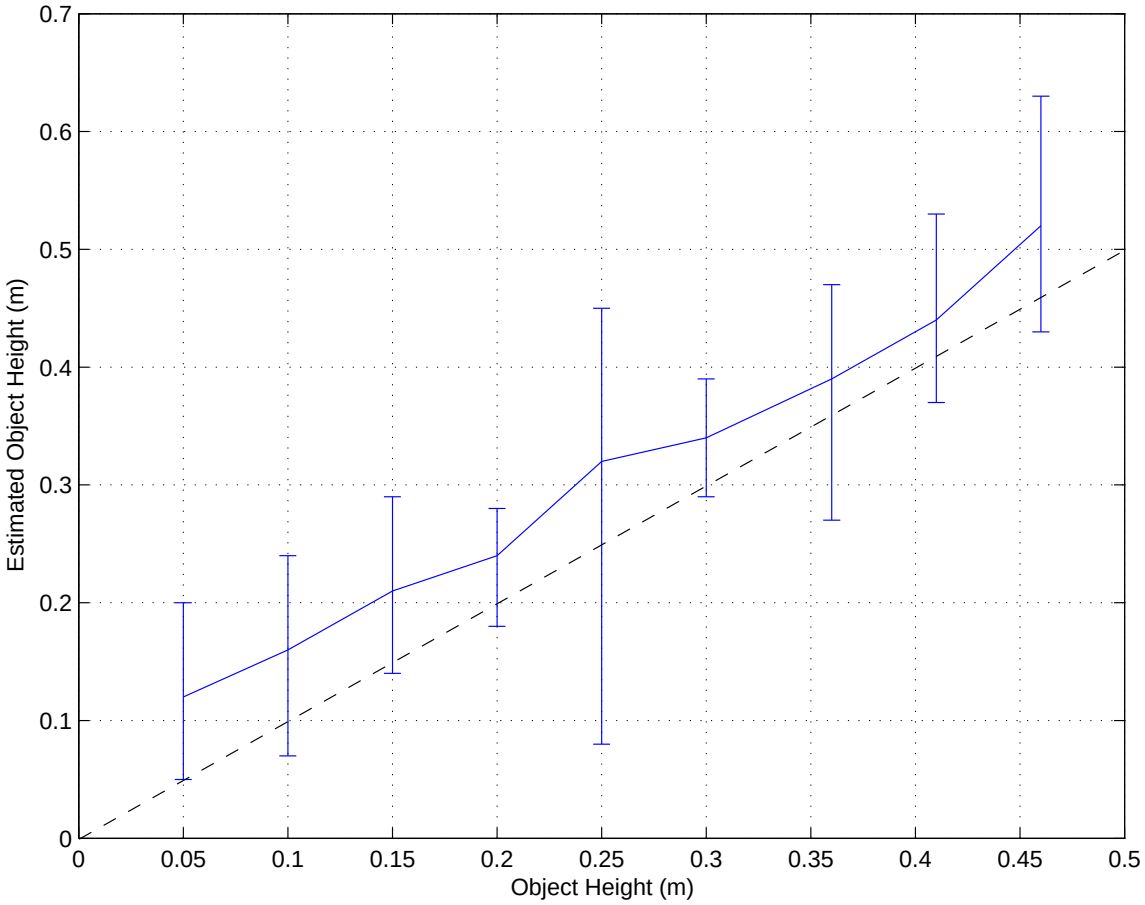



Figure 4.10: Test results for stereo estimation of box heights with pyramid level 0.

The PALACE mission anticipates normal operating altitudes (for the stereo algorithms) to fall within 10 m to 30 m, so the tests were limited to this range. The horizontal line represents the true range, and the error bars represent the local variation in the stereo range estimates. The plot indicates that the error between the average value and the actual range is within 1.3% of the true value, which is surprisingly accurate. However, it should be pointed out that the tests in simulation make use of perfect cameras and perfect camera models. Similar tests with flight hardware are described in Section 5.2. These tests are sufficient to confirm that the SLAD algorithm is being presented with sufficiently accurate range data to identify a safe landing site.

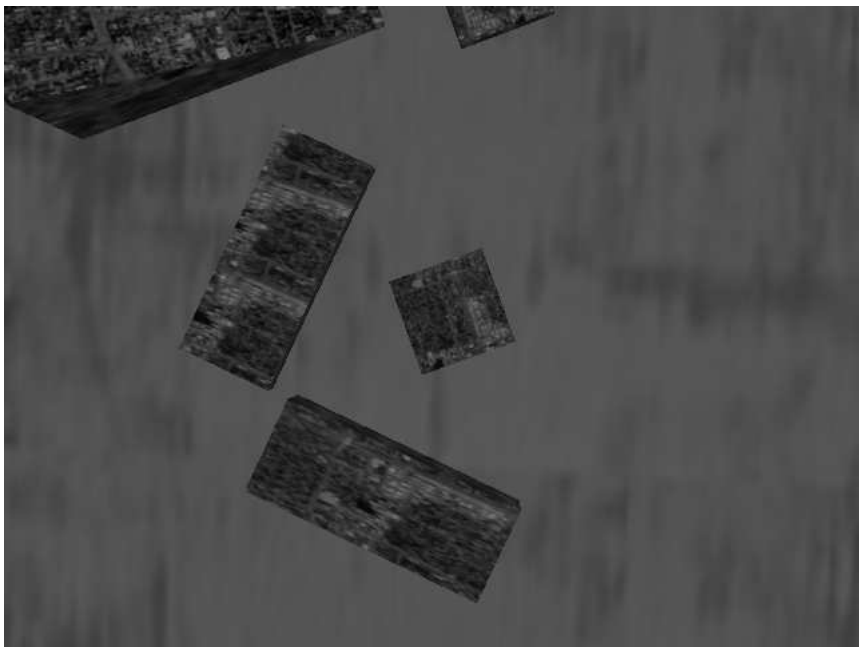


Figure 4.11: Sample image from a test to investigate range-map accuracy.

### 4.1.3 SLAD Performance Assessment

All of the remaining results in this section were designed to evaluate the robustness of the SLAD algorithm under different conditions. In each case, these evaluations were carried out under the following nominal conditions and allowing variation in only one of the variables:

- Max slope = 15 degrees
- Max roughness = 0.15 meters
- Lander base size = 3.0 meters diameter
- Elevation map grid size = 400×400 points
- Camera angle = 30 degrees from vertical
- Number of safe landing areas = 1
- Obstacle spacing ≈ 1.3 meters

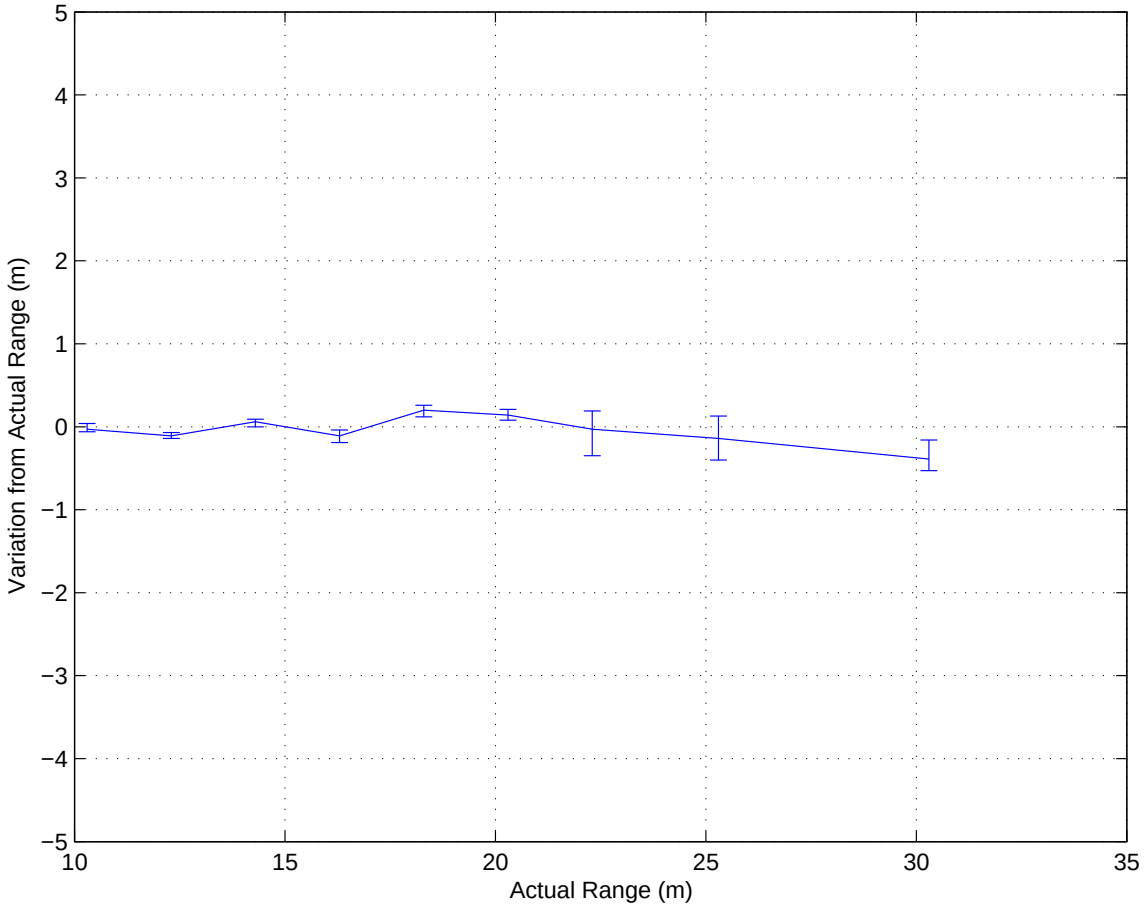



Figure 4.12: Test results for stereo range estimation.

- Height above ground ≈ 12 meters
- Lighting conditions = 80% of maximum simulation lighting

### Number of Safe Landing Areas

The first test was designed to observe the effects of having different numbers of safe landing areas. The scene was arranged in simulation so that there were no safe landing areas (see Figure 4.13), and then the boxes were moved or removed one at a time until there were three general landing areas to choose from (see Figure 4.14). For each case, the helicopter was instructed to carry out 30 SLAD analyses of the same scene with a five-second delay between each one. The test included maximum

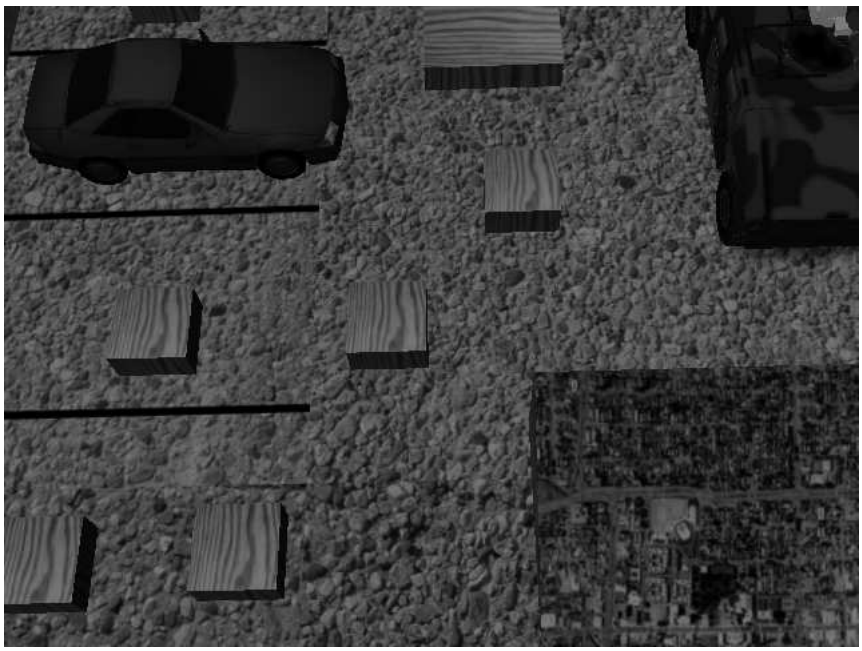


Figure 4.13: Sample camera image of landing site with no safe landing areas.

turbulence so that the stereo images for the 30 analyses would represent slightly different viewpoints of the same scene. The 30 points returned by SLAD were all plotted on one image (see Figure 4.15 for sample results), and a success ratio was assigned by human observation of the percentage of points that fell within a safe landing area. The number of points that fell in unsafe terrain were recorded as false positives, and the number of times SLAD reported no safe landing site were recorded as false negatives. The sum of these three figures will total 100%.

For the case where there were 0 safe landing points, there is potential for confusion because success is represented differently. Since the algorithm should be expected *not* to report a safe landing point, then success is measured from the number of times this actually occurred. If a point *was* selected, then it was recorded as a false positive since it falls within unsafe terrain. False negatives do not have any meaning for this case, and are shown as 0%.

Figure 4.16 shows that SLAD successfully identifies a safe landing point (or lack thereof) roughly 90% of the time overall. In the case of 1 and 2 safe landing areas, there were no instances of false positives, but SLAD occassionally reported that

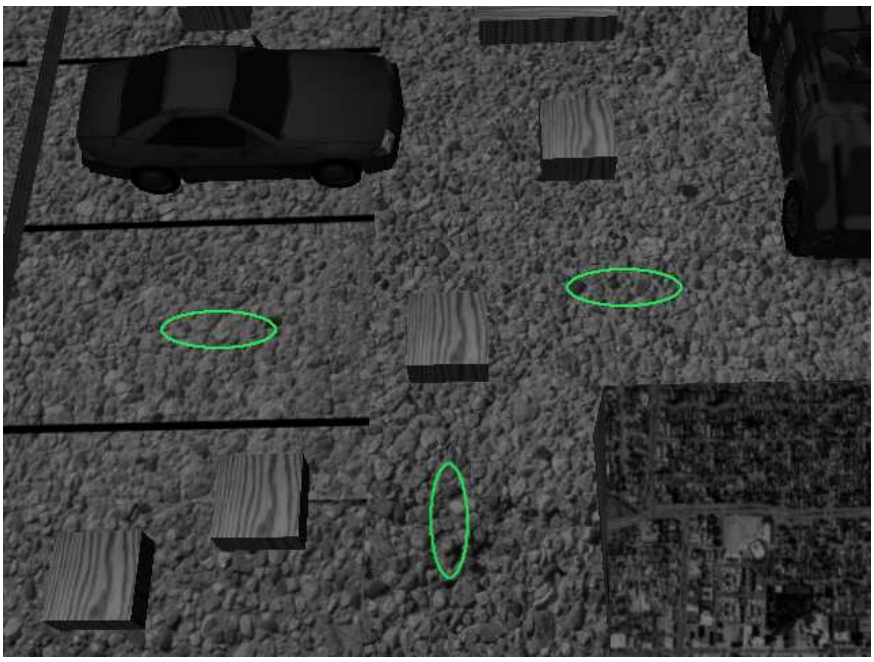


Figure 4.14: Sample camera image of landing site showing three general safe landing areas.

there were no safe landing areas. These cases involving SLAD failure were analyzed and it was found that the stereo analysis resulted in poor-quality range maps that contained many small holes and unusually-large variations in the range data. In the case of 0 safe landing areas, the SLAD algorithm erroneously reported a safe landing point about 15% of the time. Although the failure rate is fairly low, the consequences of incorrect judgements in an actual landing could jeopardize the mission. Further investigation will be needed to improve the success rate.

### Obstacle Spacing

The purpose of the next test was to evaluate the SLAD success rate as the obstacle spacing approaches the value for the SLAD constraint on minimum distance from hazards. In this test, the constraint is represented as the lander base size, which is the minimum diameter of hazard-free terrain necessary to constitute a safe landing area. The obstacle spacing is expressed as a percentage of the lander base

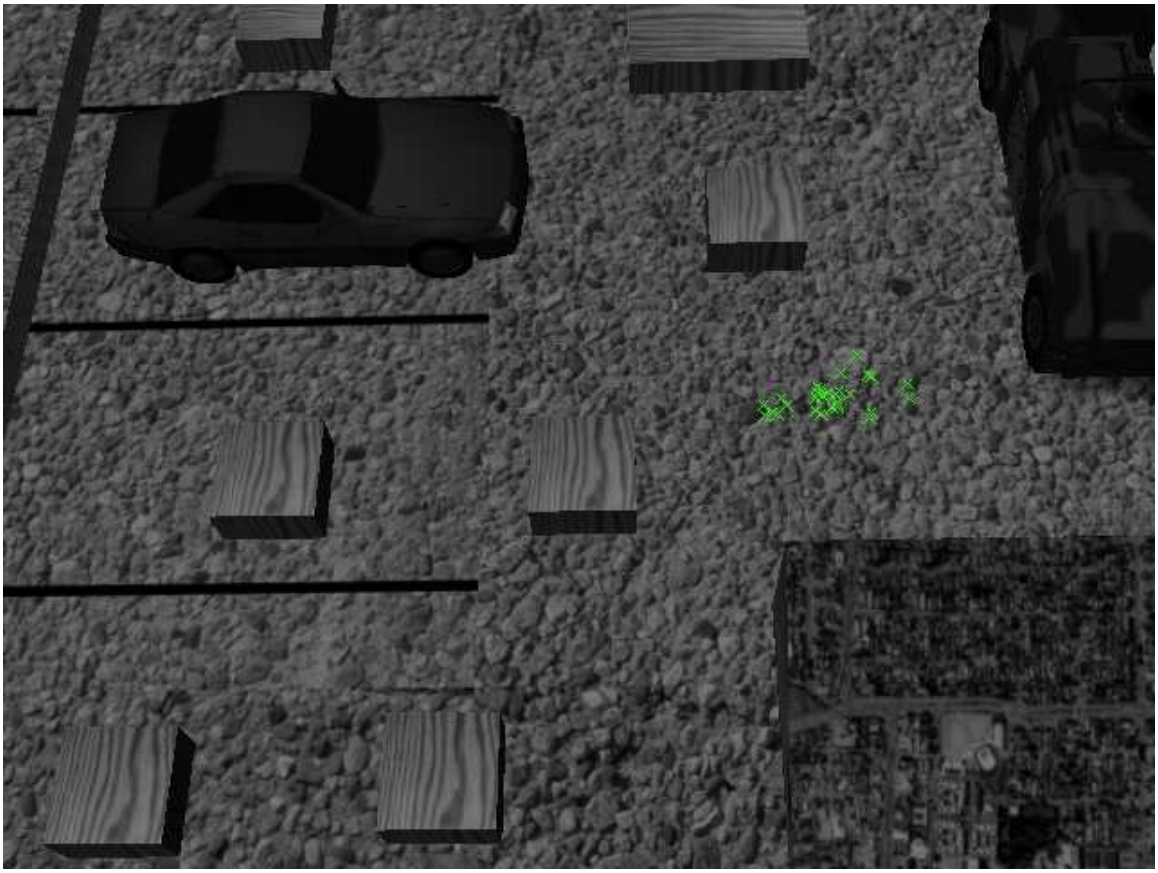


Figure 4.15: SLAD results for one test with one safe landing area.

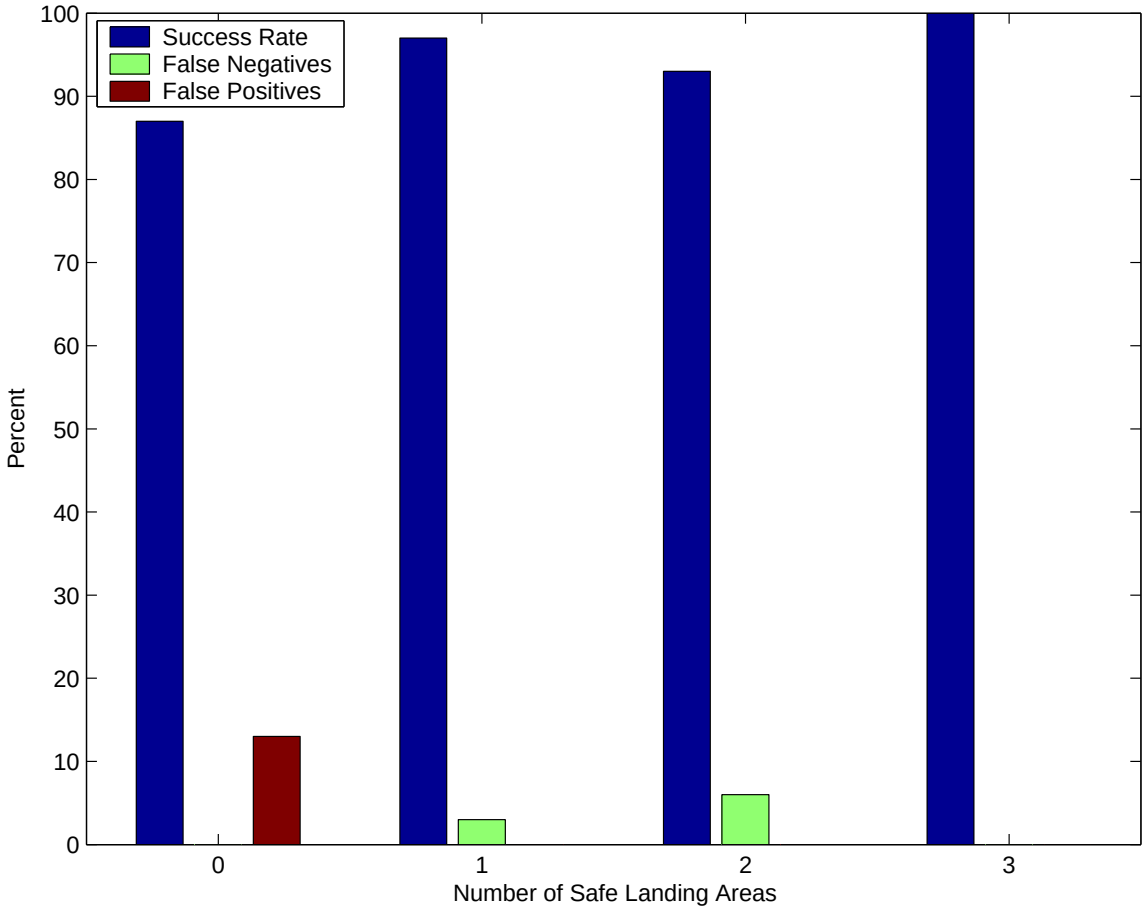



Figure 4.16: SLAD test results for different numbers of safe landing areas.

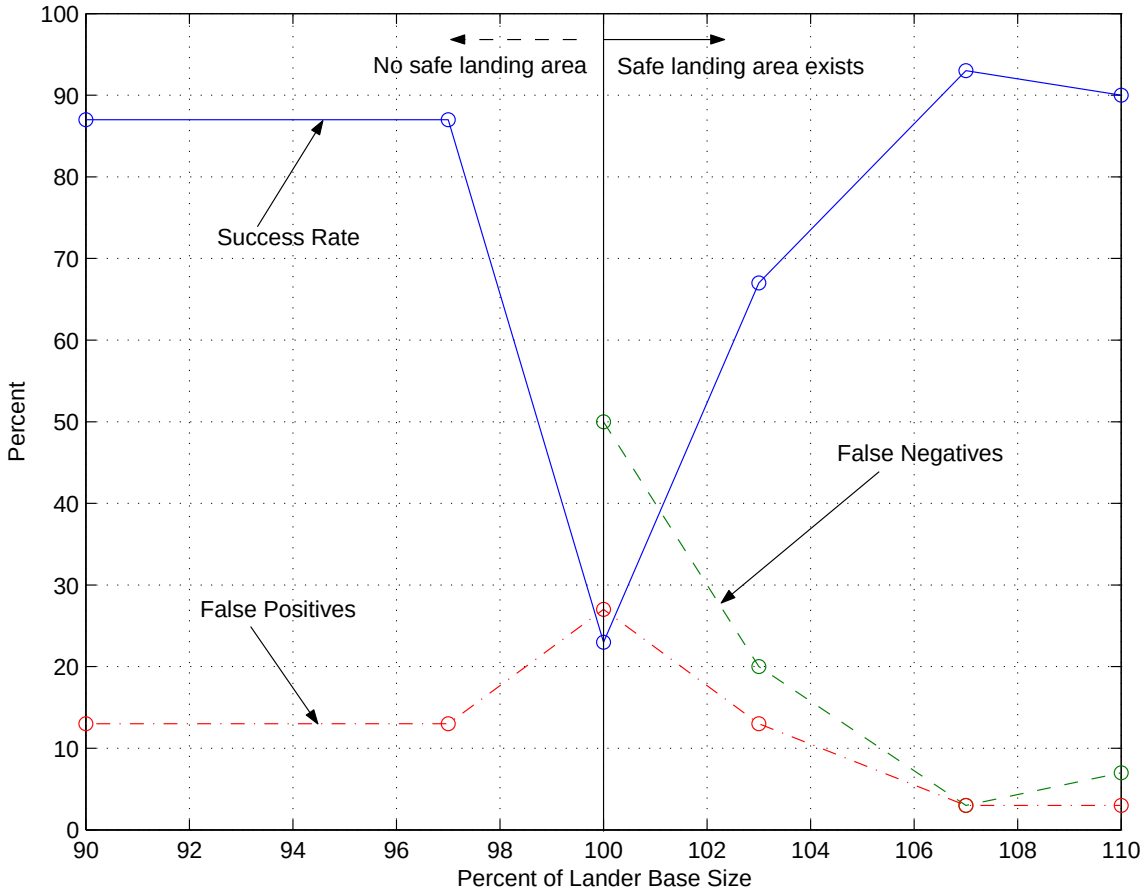



Figure 4.17: SLAD test results involving variation of obstacle spacing.

size, and data points for each obstacle spacing were represented as a success rate out of 30 analyses of the same scene.

The results are plotted in Figure 4.17, which contains a vertical line at an obstacle spacing of 100% of the lander base size. As with the previous tests, there is potential for confusion in this plot because conditions to the left of the vertical line represent an absence of a safe landing point. Therefore, in these cases success is measured as the number of times SLAD reported *no* safe landing point; a false positive is the number of times it *did* report a safe landing point; false negatives have no meaning and are left out.

This plot confirms previous results that SLAD is successful about 80-90% of the time when there is clearly the presence or absence of a safe landing point.

However, as the obstacle spacing approaches the boundaries of the constraint, the SLAD results are characterized by a fairly equal mix of successes, false positives, and false negatives, as if there was uncertainty. This can be attributed to the lack of precision that has been demonstrated in pyramid level 1 stereo analyses. The variation in the range map data points could cause SLAD to select a safe point some of the time, and report no safe landing point at other times. Fortunately, the failures occur at a relatively narrow margin of $\pm 3\%$ of the lander base size. Thus, the consequences of this indecision can be avoided in real missions by setting the minimum-distance constraint to a conservative (larger-than-necessary) value.

### Elevation Map Grid Resolution

The following SLAD test was designed to assess the effects of different elevation map grid resolutions. The test results plotted in Figure 4.18 show that performance generally improves with increasing grid resolution. When a low resolution is used, a high rate of false negatives are seen. This is explained by the fact that the resolution of the elevation map is low enough that it would rarely contain enough data points within a small region to establish that the landing area is safe. According to the figure, a grid resolution of at least 350 pixels is necessary for SLAD to run well. However, increasing the resolution requires additional processing time to analyze the extra data, as shown in Figure 4.19.

### Lighting Effects

The last data plot that will be presented is concerned with the effects of lighting conditions. The RIPTIDE simulation environment allowed easy adjustment of lighting conditions via a slider bar that ranged from 0% to 100% ambient light. Tests were carried out to assess the robustness of the SLAD algorithm with respect to lighting levels by reducing the nominal 80% value gradually to 0%. A sample image representing 0% light is provided in Figure 4.21.

The results are represented in Figure 4.20, which indicate that the SLAD performance is generally independent of the lighting levels used in this test. The

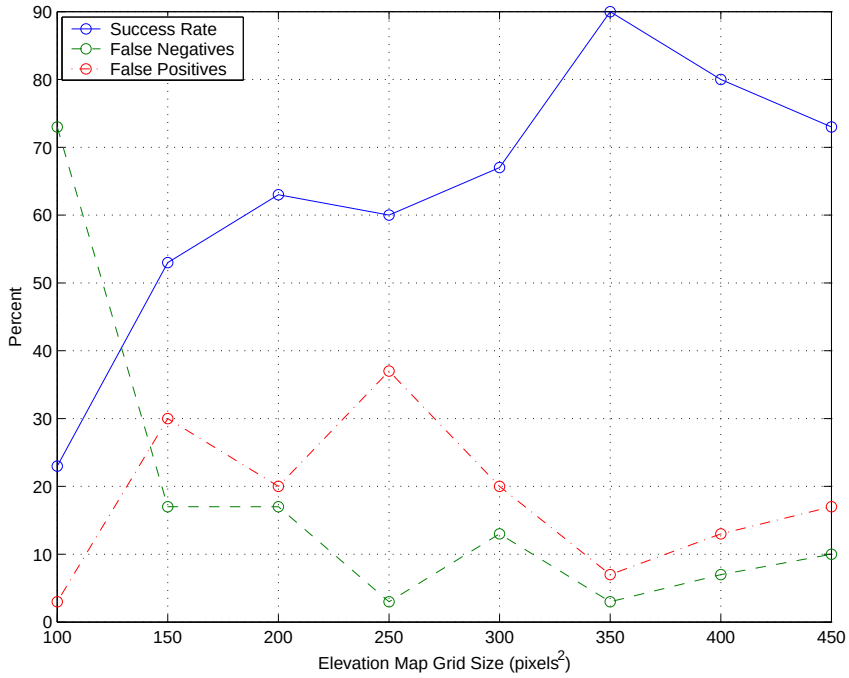



Figure 4.18: SLAD test results for different elevation map grid resolutions.

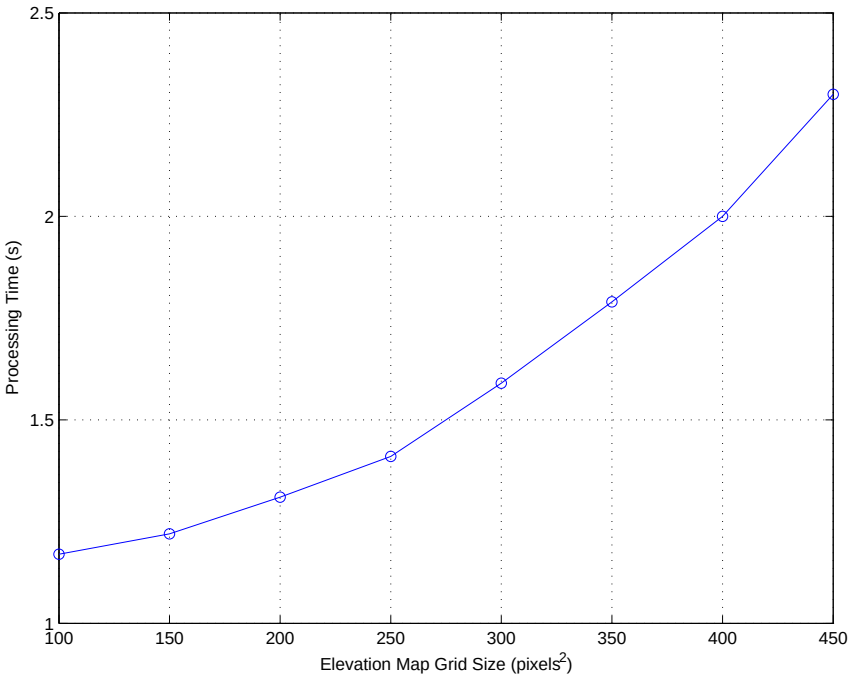



Figure 4.19: SLAD processing time for increasing elevation map grid resolutions.

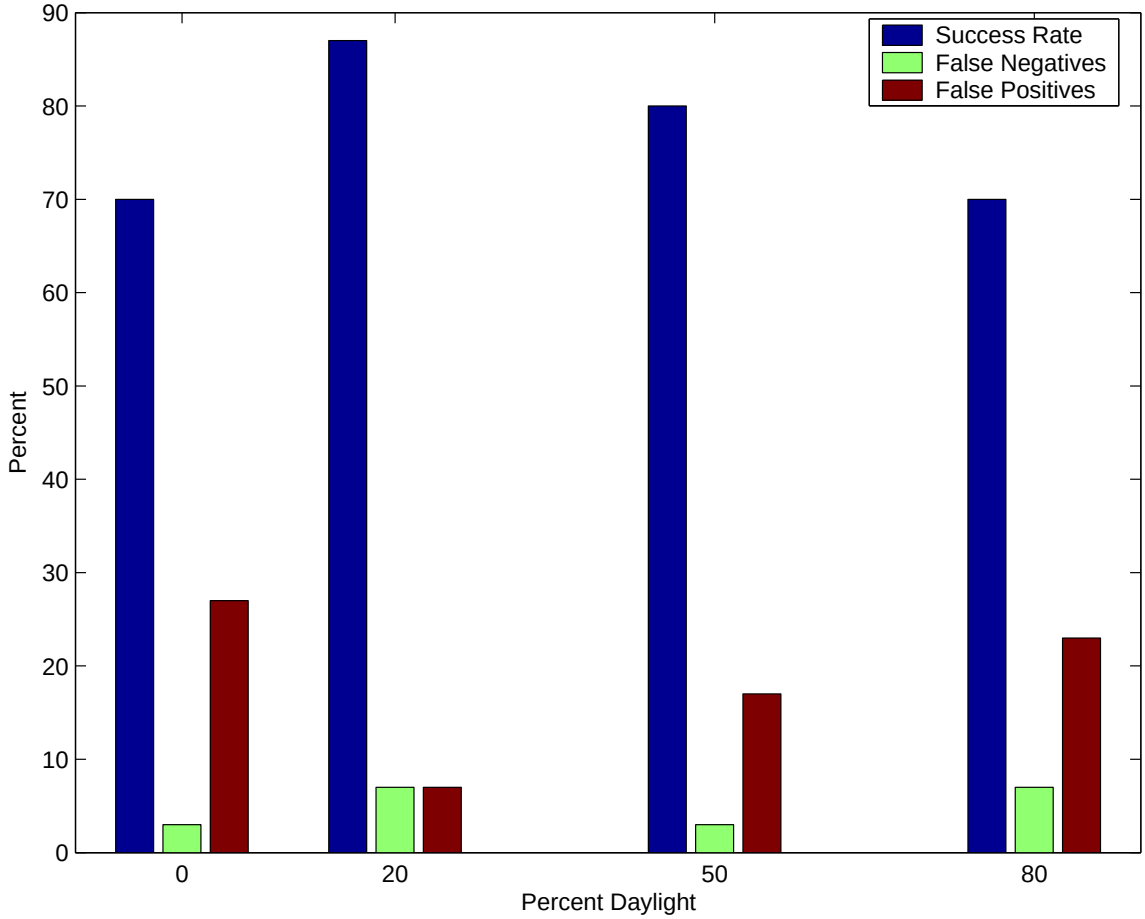



Figure 4.20: SLAD test results for different lighting levels.

Figure 4.21: Sample camera image corresponding to 0% lighting conditions.

suggested reason for these results is that although the lighting has dropped, the images retain most of the original characteristics. The only change is that there is a uniform decrease in pixel intensities. This will result in a higher probability of homogeneous regions with minimum intensity, but the higher-intensity regions will still be characterized with the same relative intensity variations from pixel-to-pixel. Thus, sufficient texture may still be present to support stereo analysis. These results do not claim good performance in perfect darkness, but they may suggest acceptable performance at dusk or in bright moonlight.

One other prominent feature in Figure 4.20 is the high rate of false positives. As mentioned earlier, further investigation will be needed to improve this aspect of the SLAD performance.

### Height Above Ground

The last test involved SLAD analyses at various altitudes ranging from 10 meters to 30 meters. In preliminary test results, it was found that the range maps quickly degraded at altitudes below 12 meters, as evidenced by an increasingly-sparse point cloud. Correspondence with JPL identified the problem in the default setting for the maximum-disparity parameter of the stereo algorithm. The default setting was set to 64 pixels, which prevented the stereo algorithm from searching a large enough region in the right-camera image to find a matching feature from the left-camera image. As shown in Figure 2.5, images taken from an altitude lower than 12 meters will be characterized by an average disparity greater than about 60 pixels, which borders on the the default maximum disparity limit set in the stereo configuration.

After the value was changed to 96 pixels, SLAD began producing good results under 12 meters. However, the tests resulted in a nearly-0% success rate above 12 meters, and a consistently-high rate of false positives (see Figure 4.22). The cause for this is attributed to the 0.15 meter roughness constraint. Although the constraint served well at low altitudes to instruct SLAD to reject 15 cm tall obstacles, at higher altitudes this constraint value became unreasonable because the variation in the estimated elevation of the flat ground exceeded this value, thus making it impossible to find *any* areas with a roughness less than 15 cm. The data from these tests clarified the need for future work to dynamically adjust the roughness constraint based on the height of the aircraft. Future work will also be needed to find out why SLAD was reporting safe landing points when it could not possibly find one that satisfied all of the constraints.

### 4.1.4 SLAD Testing Summary

The tests that have been discussed with regard to the SLAD algorithm show promising potential. The stereo ranging algorithm was proven to provide a range map that represents the terrain accurately enough to allow SLAD to identify a safe landing point. SLAD was also shown to be successful 70%-90% of the time under nominal conditions involving the clear presence or lack of a safe landing area. Data

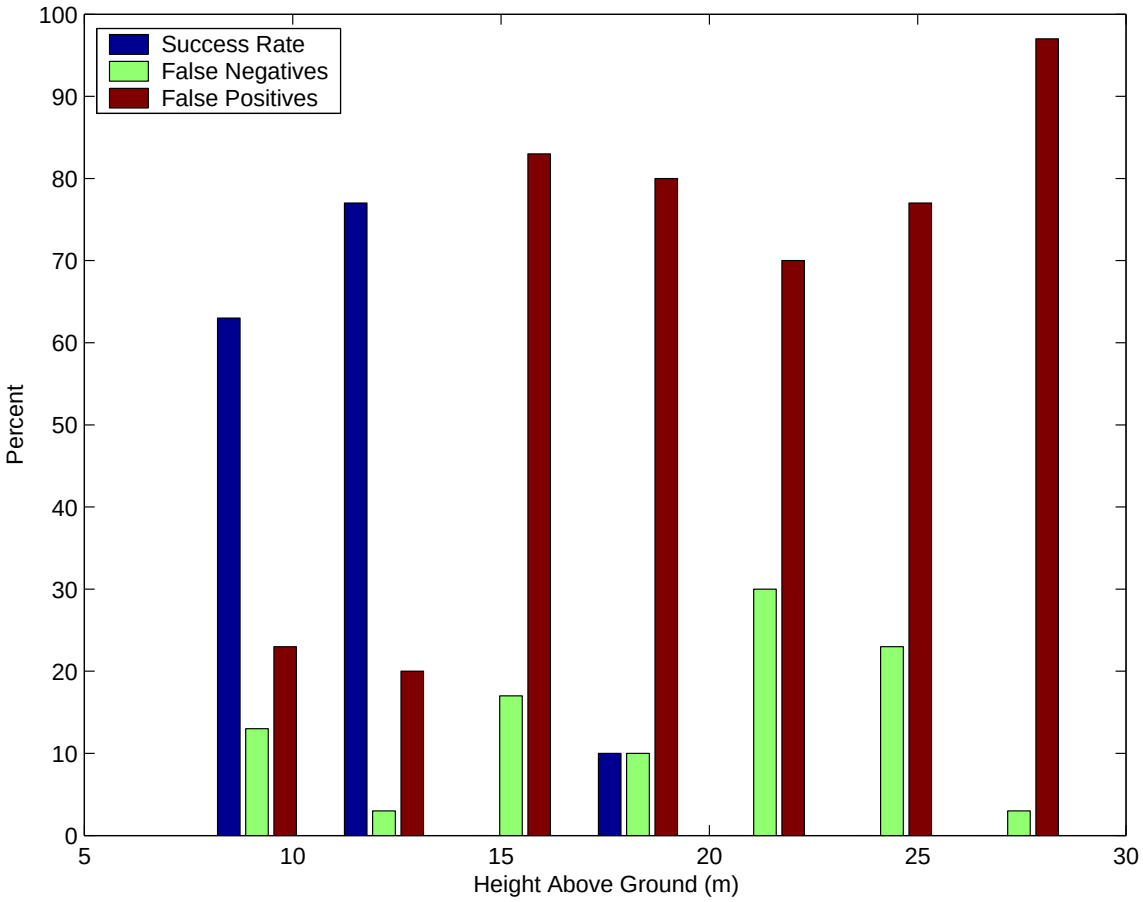



Figure 4.22: SLAD test results for different altitudes with maximum disparity of 96 pixels.

was presented that indicated good performance under low-lighting conditions, even in a cluttered landing area.

It is also clear that some anomalies have yet to be resolved in the SLAD software before applying it to real UAV missions. Test data showed the consistent presence of false negatives at the rate of 5%-10%. It was also common to see a slightly higher rate of false positives, meaning that SLAD sometimes chose landing points that violated one or more of the constraints. Finally, the need was shown for the Mission Manager to be adapted to support a dynamic roughness-constraint value, and for the minimum-distance constraint to be set to a conservative value relative to the actual helicopter landing footprint.

## 4.2 Tracking / Position-Estimation Evaluation

The last half of this chapter will present the results of tests conducted to evaluate the performance of the tracking and position-estimation algorithms. There are two important performance heuristics:

1. The proximity of the tracked features in two chronological camera frames. This heuristic is a measure of frame-to-frame performance, and directly affects the stability of the vehicle as position estimates are used to produce control efforts.

2. The proximity of the feature in the first frame to the feature that was tracked in another frame much later in time. This heuristic is a measure of overall performance over time. It accounts for gradual drift that would not cause aircraft instability, but would coax the aircraft into landing at a point other than the one selected by SLAD.

### 4.2.1 Tracking Metrics and Procedure

The metrics used to represent these heuristics are based on the observation that the position-estimation results are closely linked to the tracking performance. Assuming that the error from the position-estimation method itself is negligible, then

the error in the position estimate can be used to indirectly represent tracking performance. More specifically, the error in the position estimate at the end of a test run will be used to represent the second heuristic mentioned above. Furthermore, if the error in the position estimate is erratic (i.e., performance is poor relative to the first heuristic), then it will likely lead to control efforts that will result in aircraft instability. This metric will be used as a measure of the first heuristic, and will be expressed as the number of times that aircraft instability was observed under a given set of test conditions.

These metrics are based on the assumption that the error in the position-estimation method is negligible, and that all position-estimation error is attributed to poor tracking performance. To show that this is true, position-estimation test data is plotted in Figure 4.23. This data compares open-loop position estimates to the actual GPS position in each of the three world axes while the aircraft was commanded to hover in simulation. The data shows that position estimates are accurate to about 10 centimeters when hovering at an altitude of about 12 meters.

In order to illustrate how the MPE error is linked to tracking performance, an example of one test run is represented in Figure 4.24, which shows a camera image with a large "X" marking the feature that was selected for tracking in the first frame, and a small "X" marking the feature that was being tracked when the test was terminated two minutes later. After two minutes of tracking frames at 10 Hz, it is clear that the tracker has drifted a little and is tracking another feature that is about 10 pixels ($\sim$0.25 m) away from the original feature.

However, the position-estimation equations assume perfect tracking, and that the tracked feature is stationary. Since the tracked feature has moved, the aircraft will move with it to maintain a constant position estimate because it was commanded to hover and hold position. Figure 4.25 displays the estimated horizontal path of the vehicle over time, which is centered mostly on the starting point (the origin). This is what should be expected because the control laws were working to maintain a constant position in the presence of turbulence and steady wind.

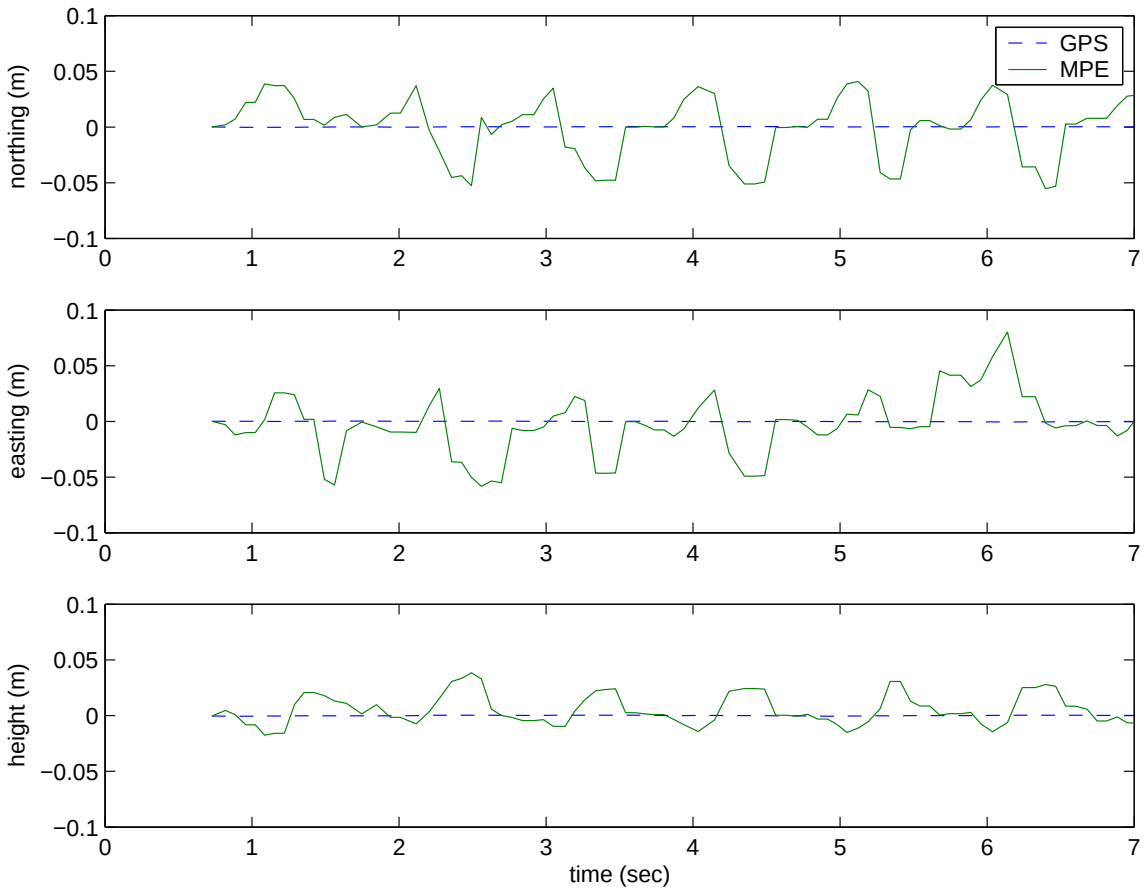



Figure 4.23: Open-loop test data illustrating MPE accuracy.

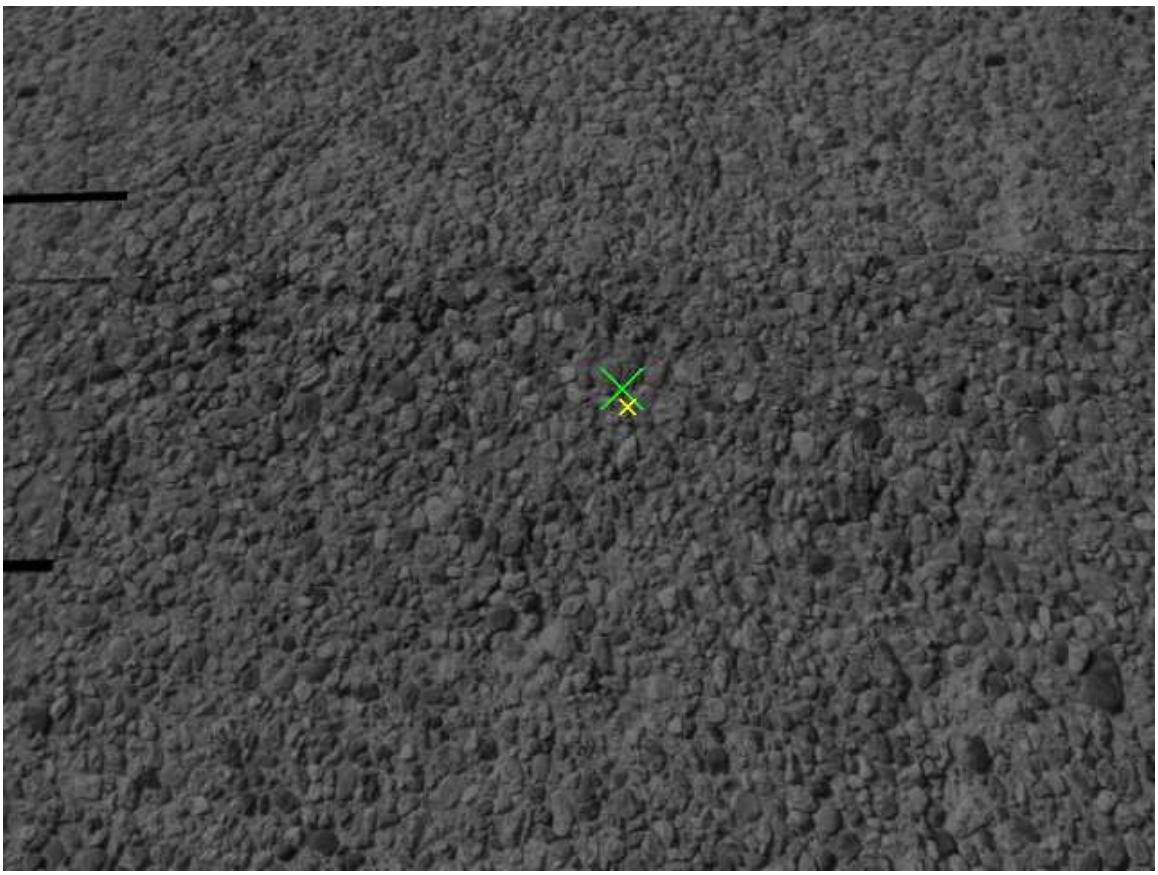


Figure 4.24: Sample camera image showing the initial (large "X") and the final (small "X") features from a two-minute test of the tracking algorithm.

However, the true aircraft position is drifting as a result of the tracking pixel drift. By subtracting the estimated aircraft position (Figure 4.25) from the true GPS position, a similar plot is generated (Figure 4.26) to represent the error in the position estimate over time. This plot shows that the error is more or less constant in the bottom-right region at the beginning of the test, but then migrates over time to the upper-left region of the plot. The change in the error from the beginning to the end of the test is seen to be 0.132 meters in the easting direction, and 0.216 meters in the northing direction.

In comparison, Figure 4.24 shows a displacement of 5 pixels in the $x$-direction and 9 pixels in the $y$-direction between the initial and final features. According to the equations from Section 2.3, and assuming no change in attitudes or altitude, a

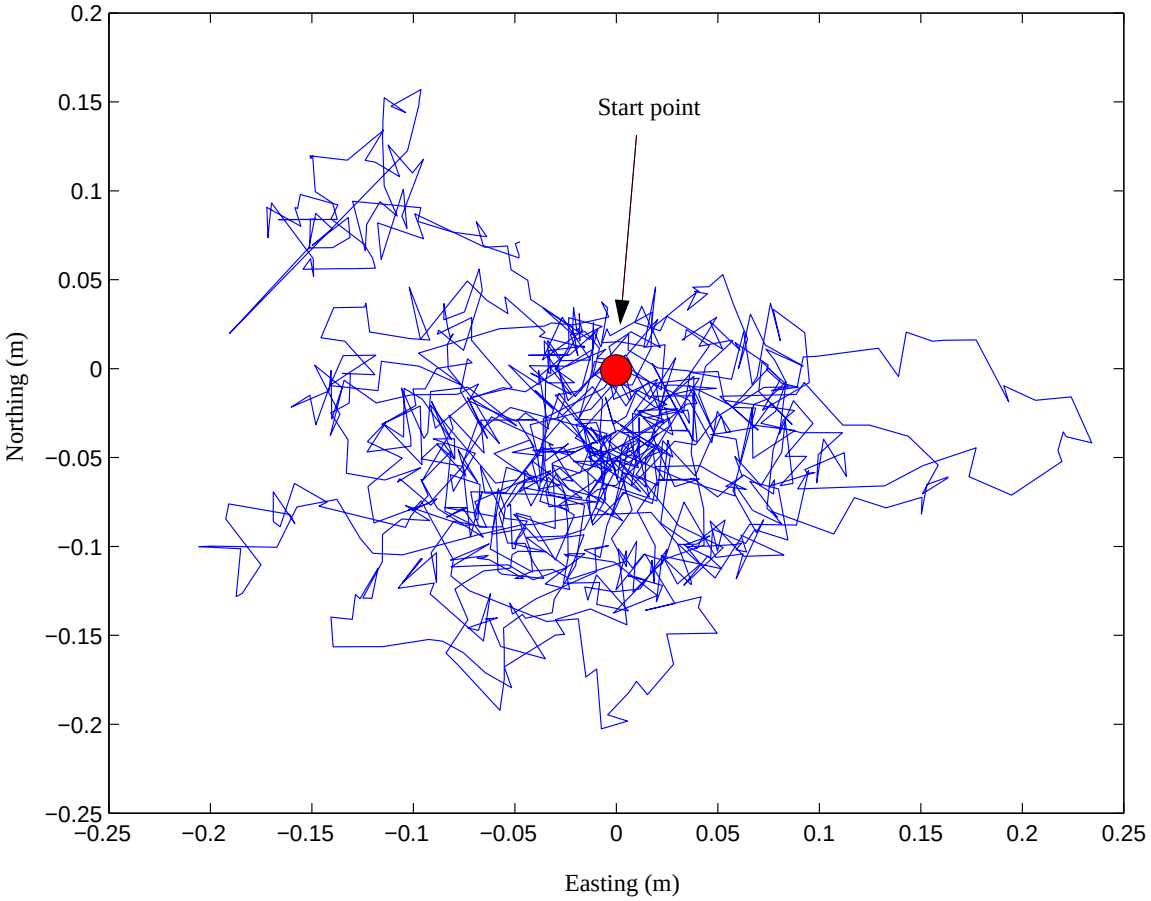



Figure 4.25: Deviation over time of estimated horizontal aircraft position from the starting point.

position displacement of 0.158 meters easting and 0.263 meters northing would be needed to cause a feature to move this many pixels in a camera image. These values closely match the actual position-estimate error that was reported in the previous paragraph, so we may conclude that the error in the position estimate is due to tracking drift. This justifies using the MPE error as a metric of tracking performance over time.

Two other metrics are used to provide additional insight into the tracking performance. These metrics are based on the *coherence*[1] value that is returned by

[1]The coherence is a number between 0 and 1 that represents a level of confidence that a good a match was found in one frame based on the previous frame, with 1 representing a perfect match.

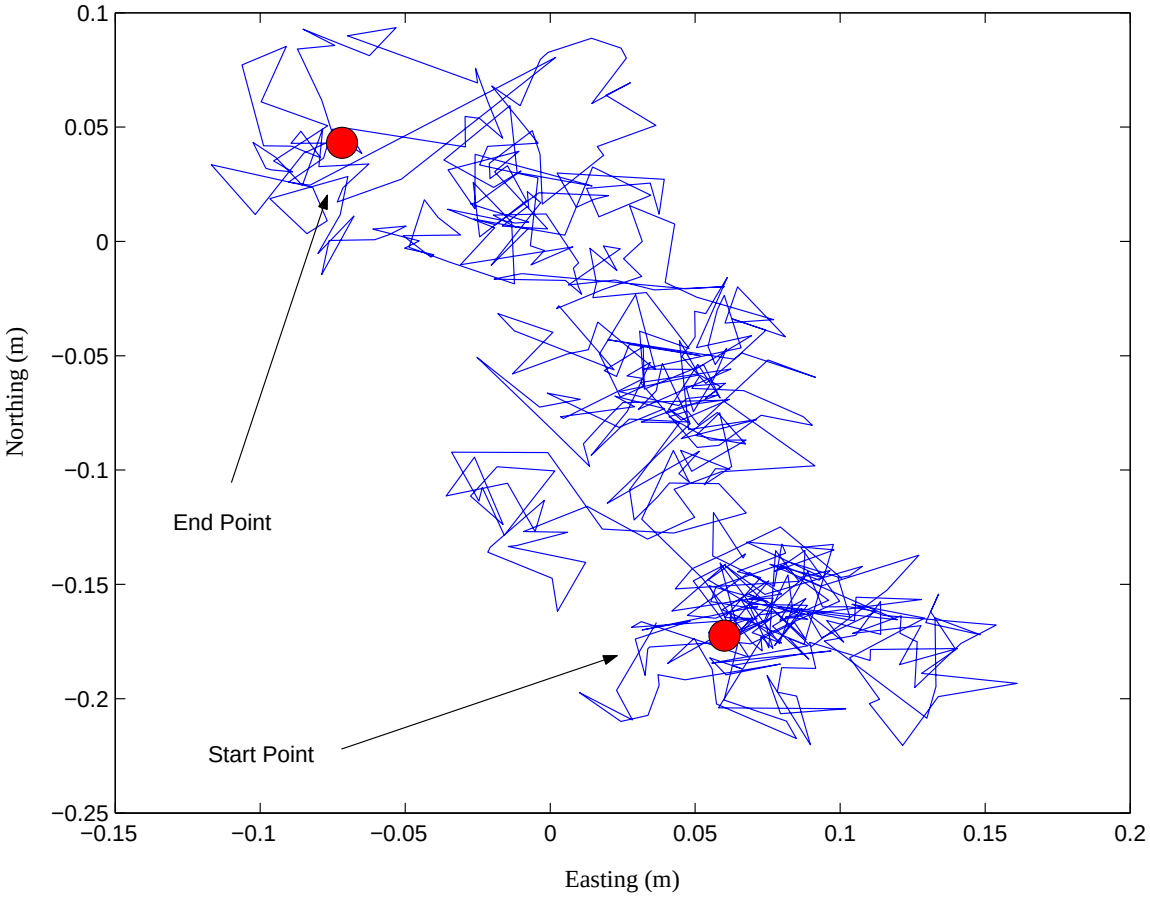



Figure 4.26: Error over time in the horizontal estimate of vehicle position.

the JPL algorithm after each tracking attempt. The statistics that will be recorded during the tracking tests are: 1) the minimum coherence during the test, and 2) the number of times the coherence drops below a certain threshold (referred to as the "coherence count"). The first metric indicates how poor the worst tracking attempt was, and the second metric is an indication of how often the algorithm is having difficulty finding a good match.

Most of the feature-tracking tests that are discussed in this section were conducted using a procedure as follows.

- Hover at 12 meters above the ground facing into the wind, select the best feature to track from a region in the center of the image, and then switch over

to navigation based on position estimation. The tracking window and search window sizes are set to 21 and 81 pixels respectively.

- Hold the hover for two minutes, then switch back to GPS navigation. Hover for a few seconds to allow transient motion to be damped out.
- Repeat the procedure five times under conditions of low, medium, and high turbulence for a total of 15 two-minute runs. Low, medium, and high turbulence conditions are defined to be 10%, 50%, and 90% of the maximum safe wind speed (4.5 m/s) and wind speed variations allowed for RMAX helicopter operation.
- Average or sum the metrics for each set of five runs to yield three data points, each one representing average performance at one turbulence level for the given test condition.
- If aircraft instability is detected, then the test run concludes prematurely and testing will resume with the next test run.

### 4.2.2 Wind Direction

The tracking test procedure was first applied to evaluate the effects of wind direction on tracking performance. The test conditions began with the aircraft facing directly into the wind (0 degrees relative to the wind), and recording metric values after each increment of 20 degrees in the heading relative to the wind. Figure 4.27 plots the test results for all four metrics relative to wind direction.

The coherence-count plot in Figure 4.27 shows that the number of low coherence values is greatest when the aircraft heading is close to right angles to the wind (90 and 270 degrees). The high number of low-coherence values suggests that the tracking algorithm is frequently processing frames with low confidence. The minimum coherence for low and medium turbulence is good at some of these angles, which indicates that the tracker was able to perform well under these conditions since the turbulence was not too extreme. But the minimum coherence was most consistently high at angles close to 0 degrees (heading into the wind).

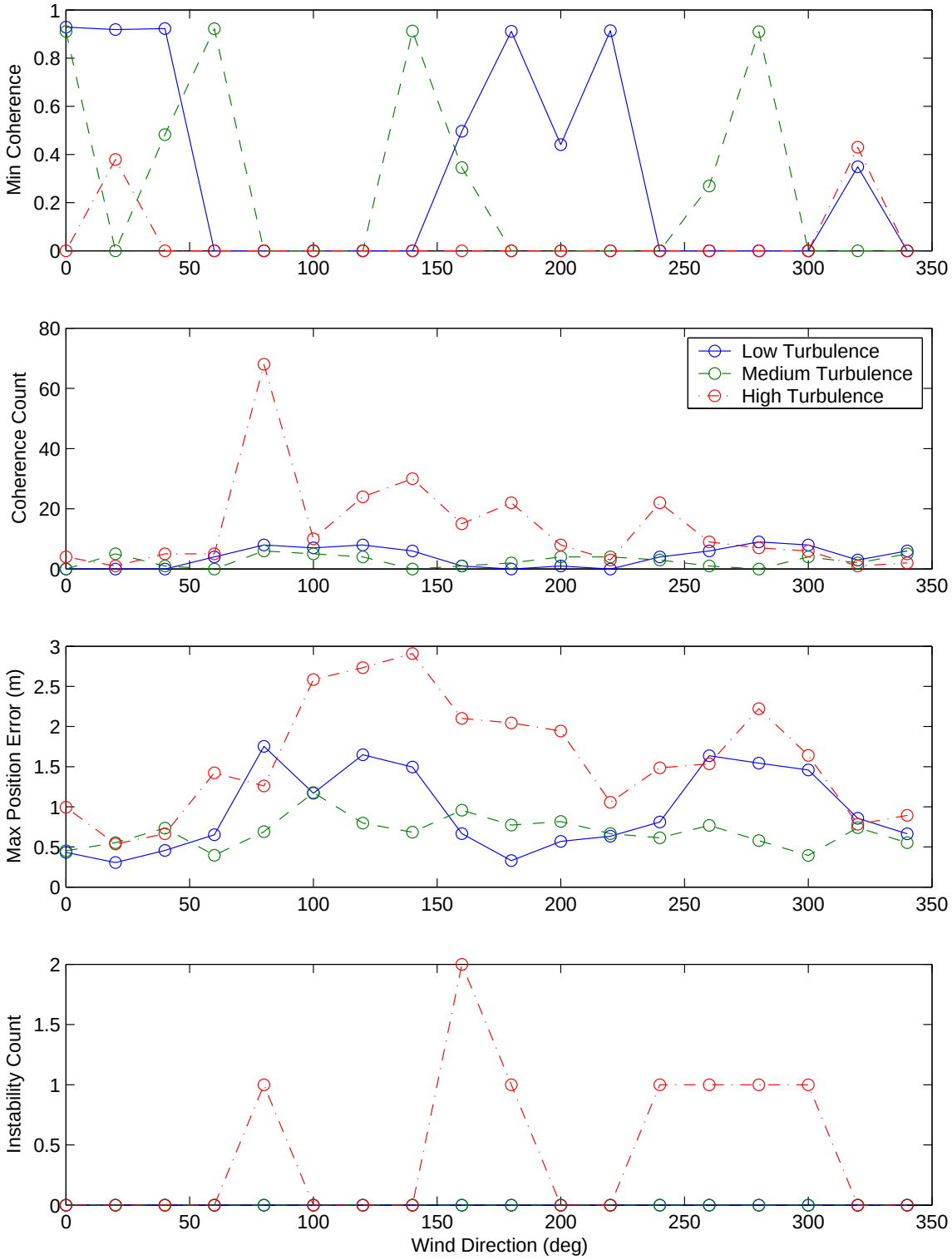



Figure 4.27: Test results of tracking performance vs aircraft heading relative to wind direction.

These results are explained by noting that when the aircraft is pointing perpendicular to the wind, the aircraft is more sensitive to yaw disturbances because the wind is blowing against the tail rotor, and more work is required to maintain heading. As described in Section 2.2, the tracking routine only searches a small portion of the image when searching for a matching feature in a new camera image. Extreme wind conditions may jar the aircraft sufficiently to occasionally move the feature outside of the search window in the next image, thus making it impossible to find the feature in the new image. According to Figure 4.27, tracking performance may be optimized by pointing the aircraft into the wind to reduce the influence that turbulence exerts on the camera images.

A look at the max-position-error plot adds confidence in these conclusions. The average maximum error in the position estimate tends to increase as the heading deviates from 0 degrees, indicating larger drift in the feature-tracking. It should be pointed out that this larger drift is not simply due to larger motions of the feature within the camera images from frame-to-frame. Whether a feature moves 5 pixels or 20 pixels from frame-to-frame does not influence the tracker's ability to find the feature, because it will examine *all* of the pixels within the search window. The cause for the larger drift is more likely attributed to the greater probability that the feature will move outside of the search window, and that the tracker will find a nearby, next-best match and begin tracking it.

Finally, the instability-count plot discloses the occurrence of a few instabilities at headings away from 0 degrees in high turbulence. This is probably a result of buffeting so violent that the feature-tracker was constantly losing sight of the feature, thereby forcing the position estimation to put out erratic or discontinuous values. Since the control system was not designed to handle this, it would provoke disastrous control efforts. The fact that only a few instabilities were recorded under extreme wind conditions may be viewed as indication of the tracking / position-estimation robustness, but nevertheless is a warning of high-risk conditions that should be avoided.

### 4.2.3 Tracking Height Above Ground

The next tests were conducted to evaluate how the tracking performance varies with altitude. These tests will also provide the lower limit under which tracking performance is degraded (see Section 3.3). Figure 4.28 shows the results of these tests by plotting the same metrics used in the previous tests, except that the instability count is omitted because no instabilities were observed during these tests.

The minimum-coherence plot provides strong evidence that tracking performs best at high altitudes. Since the minimum coherence values are high, we know that the algorithm did not process even one frame with a low level of confidence. Even in cases where a low minimum-coherence value is seen for high-altitude tracking, the coherence-count plot indicates that it was not a frequent occurrence. As mentioned in Section 3.3, one pixel in an image from a high altitude will cover more ground than one pixel at a low altitude. For this reason, turbulence will have less of an effect on camera images taken at higher altitudes.

This effect also explains the general upward trend in the max-position-error plot. Tracking drift of a few pixels at higher altitudes will generate a larger error in the position estimate than the same drift at low altitudes because it represents a shifting of the original point over a greater distance. Because of this, the MPE error cannot be used alone to compare tracking performance at different altitudes.

The most important observation, based on this data, is that tracking performance begins to suffer slightly as the aircraft descends below 6 meters. This is apparent in looking at the minimum coherence, which indicates that the coherence is likely to drop to 0 at least once in a two-minute span. However, the coherence-count plot suggests that these occurrences are relatively infrequent until the altitude drops below 3 meters.

Although no instabilities were observed during any of the tests, and although it appeared as though the control laws were succeeding in holding a constant position, the high count of low-coherence values implies a greater potential for discontinuous position estimates that may provoke unstable control efforts. This data leads to

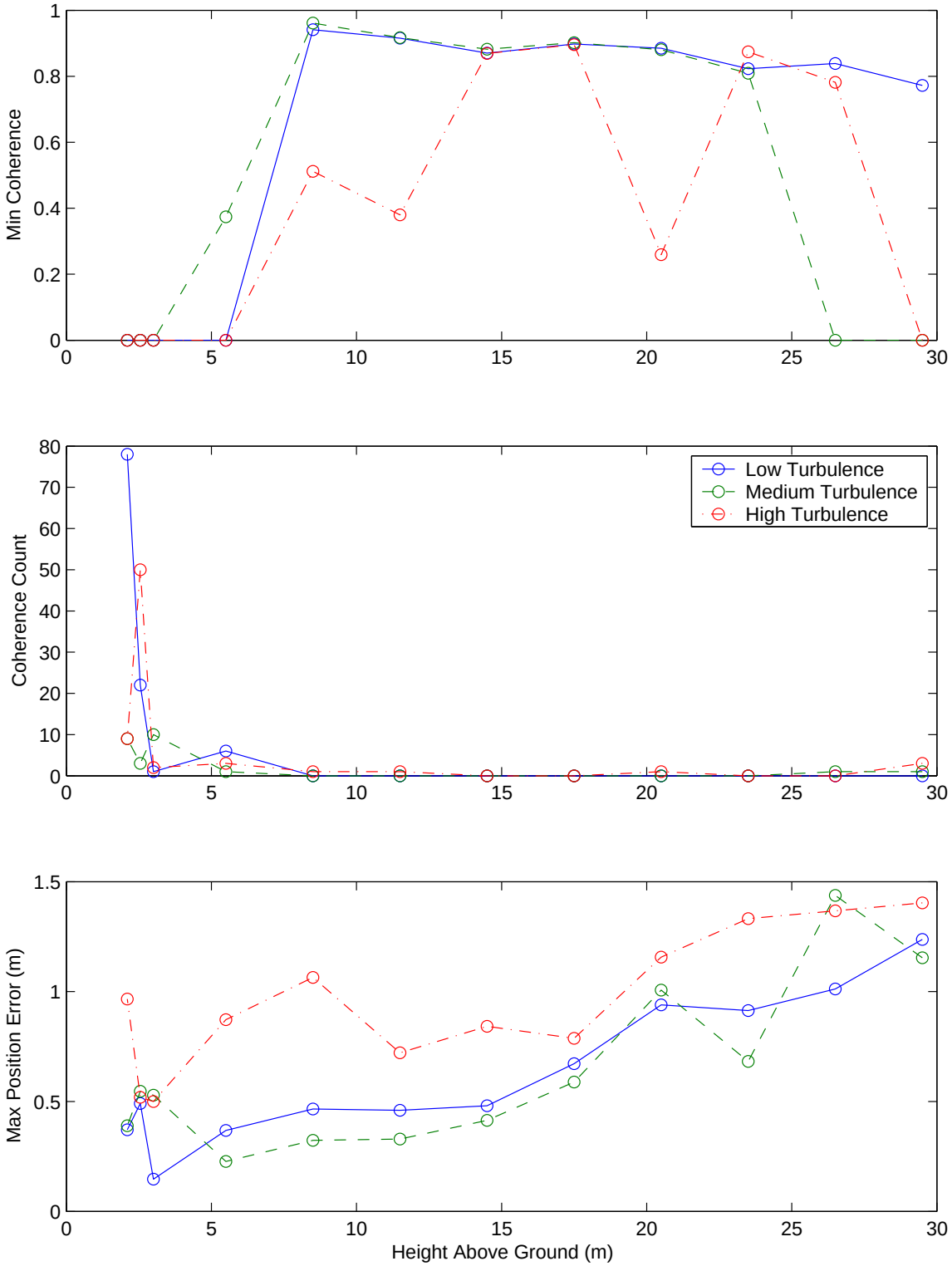



Figure 4.28: Test results of tracking performance vs height above ground.

the conclusion that safe conditions of operation will be found no lower than 2.5 to 3.0 meters above the ground.

### 4.2.4 Template Window and Search Window Sizes

Two primary parameters of the tracking algorithm are the sizes of the search window and the template window. A large search window generally translates to a greater probability that a matching feature will be found, but also incurs a greater demand on the CPU to search through a larger portion of the image. Similarly, a large template window will generally lead to better performance since there is a greater probability of bounding unique features, but this also demands more processing power.

In order to understand the results of the tests designed to quantify the influence of these parameters, the relationship between these window sizes must be clarified. In Section 2.2, it was explained that feature matching was done by convolution of the template window within the search window in the new image. For example, Figure 4.29 illustrates how a template window of 21×21 pixels is moved one pixel at a time and convolved at every possible location throughout an 81×81 pixel search window. This particular tracking implementation defines a window as a center pixel with window boundaries drawn a constant number of pixels away in the $x, y$ directions, hence the square windows with odd sizes. Based on the given definitions, Figure 4.29 shows the important conclusion that the effective search window size is dependent on the template size. With a template window size of 21 pixels, and a search window size of 81 pixels, the algorithm will only search a region of 61×61 pixels.

The testing procedure was applied as before, except that the search window size was varied from 41 to 141 pixels. Figure 4.30 displays the test results. The most notable feature in these plots is that no performance is gained by increasing the window size above 81 pixels. On the contrary, the last plot shows that the time required to process one frame grows from 2-3 milliseconds to about 40 milliseconds. For the latter case, the extra processing time is not critical as long as it is below 100 milliseconds, with a little extra margin, to allow normal processing at 10 Hz. But

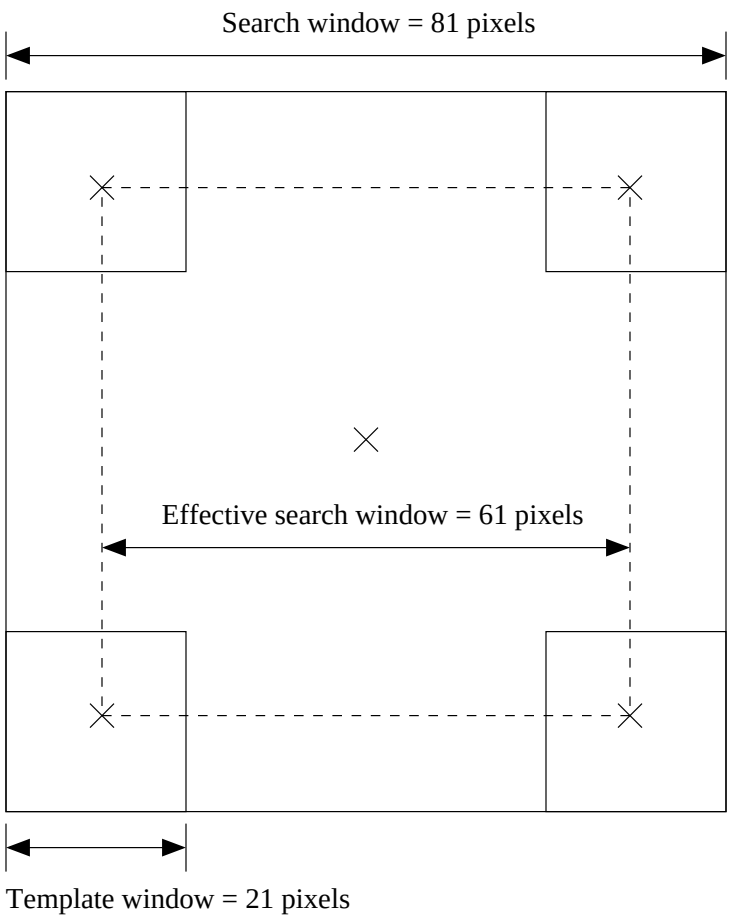



Figure 4.29: Illustration of how the effective search window is dependent on the template window size.

processing speed will likely be a crucial factor when the algorithm is implemented with the slow processors on the RMAX helicopter.

Assuming speed is a determining factor, it would be desirable to establish a lower functional limit on the search window size. It is clear from the instability count that a window size of 41 pixels is unacceptable in high turbulence - there is a large probability that the feature will move outside of the small search window before the next frame is processed. This is confirmed by the high number of low-coherence values recorded during this test. The MPE-error and instability-count plots show that this window size may be acceptable in low and medium turbulence, but the minimum-coherence plot indicates that the feature may have been lost at least once.

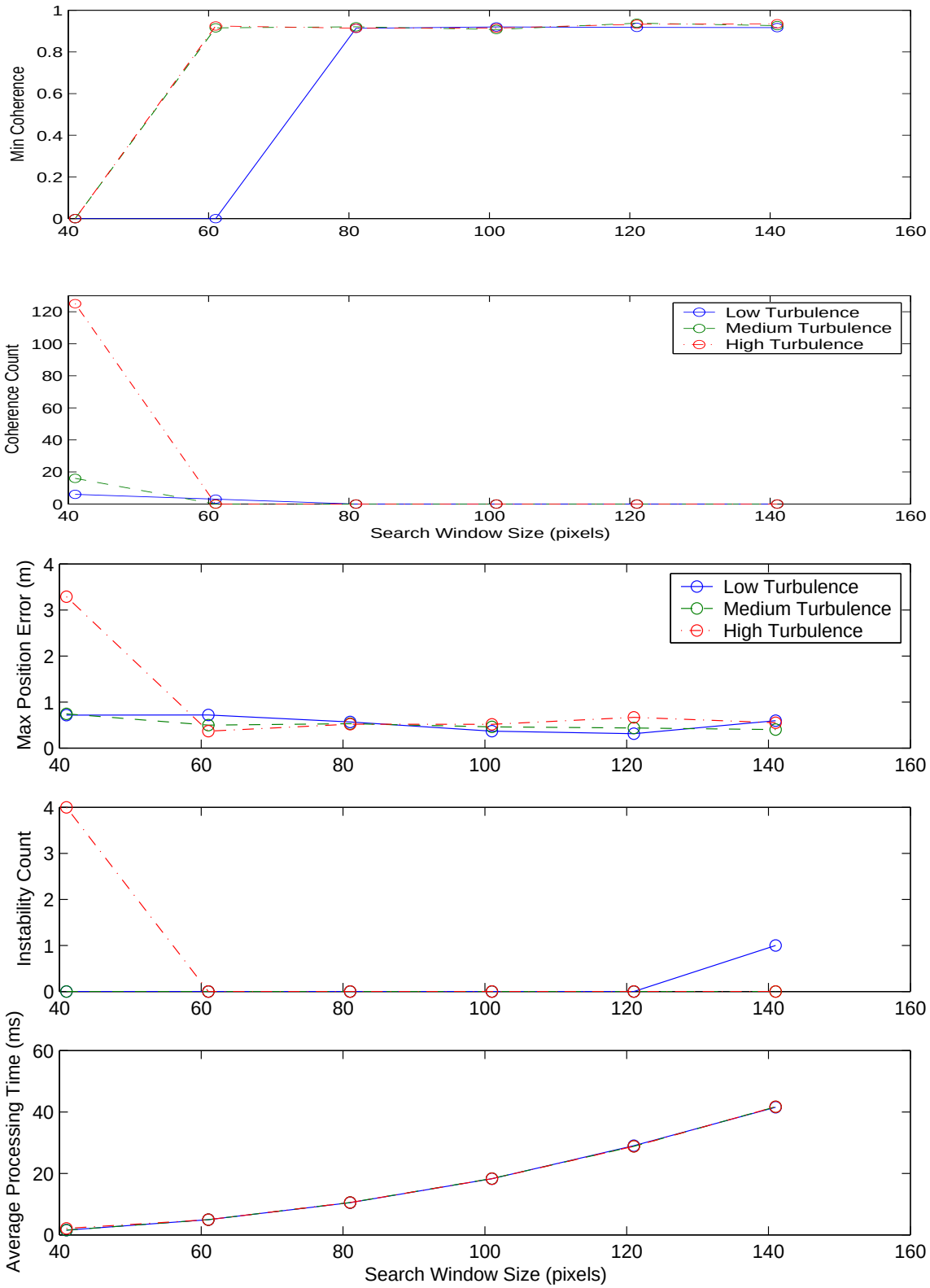



Figure 4.30: Test results of tracking performance vs search-window size.

At a window size of 61 pixels, all of the plots give signs of optimum performance, except for the case of low turbulence in the minimum-coherence plot, which indicates that the pixel may have been lost at least once. This is surprising because poor performance would be most likely observed in the case of high turbulence. Since performance in the high-turbulence case appears excellent (as shown in Figure 4.30, this was probably a rare mishap that can be disregarded. This is further supported by noting that the maximum error in the position estimate shows no adverse effects. Thus, a window size of 61 pixels is probably the best compromise between tracking performance and processing speed.

As further evidence, Figure 4.31 includes two more plots that show the maximum displacements of the pixel coordinates in the $x$ and $y$ directions from one frame to the next. In other words, while tracking a feature using a search window size of 101 pixels, the maximum-$\Delta$X plot shows that the $x$-coordinate of the pixel never changed more than 20 pixels from frame-to-frame. If this were always guaranteed to be the case, then an effective search window size of only 41 pixels would be necessary (to allow for a movement of at most 20 pixels to the left or the right). With an effective search window size of 41 pixels, and template size of 21 pixels, then a search window size as small as 61 pixels would have sufficed, meaning that processing time was wasted in searching through the extra pixels in the $101\times101$ pixel search window.

The plots in Figure 4.31 show that the maximum change in pixel coordinates is usually less than 20 pixels, and only exceeded 30 pixels in two cases. Therefore, a 61-pixel search window (with a 21-pixel template window) will almost always suffice to account for a movement of 20 pixels in any direction from one frame to the next. The fact that there are two cases that exceed 30 pixels is probably not cause for alarm because these plots display *maximum* $\Delta$X and $\Delta$Y over two-minute runs, and represent the cumulative results of hours of feature-tracking tests. Therefore, these outliers only represent very rare exceptions.

The next test was designed for a similar optimization of the template window size by varying it from 11 to 61 pixels. Although it was shown above that the optimum search window size was 61 pixels, the testing procedure that was defined in the

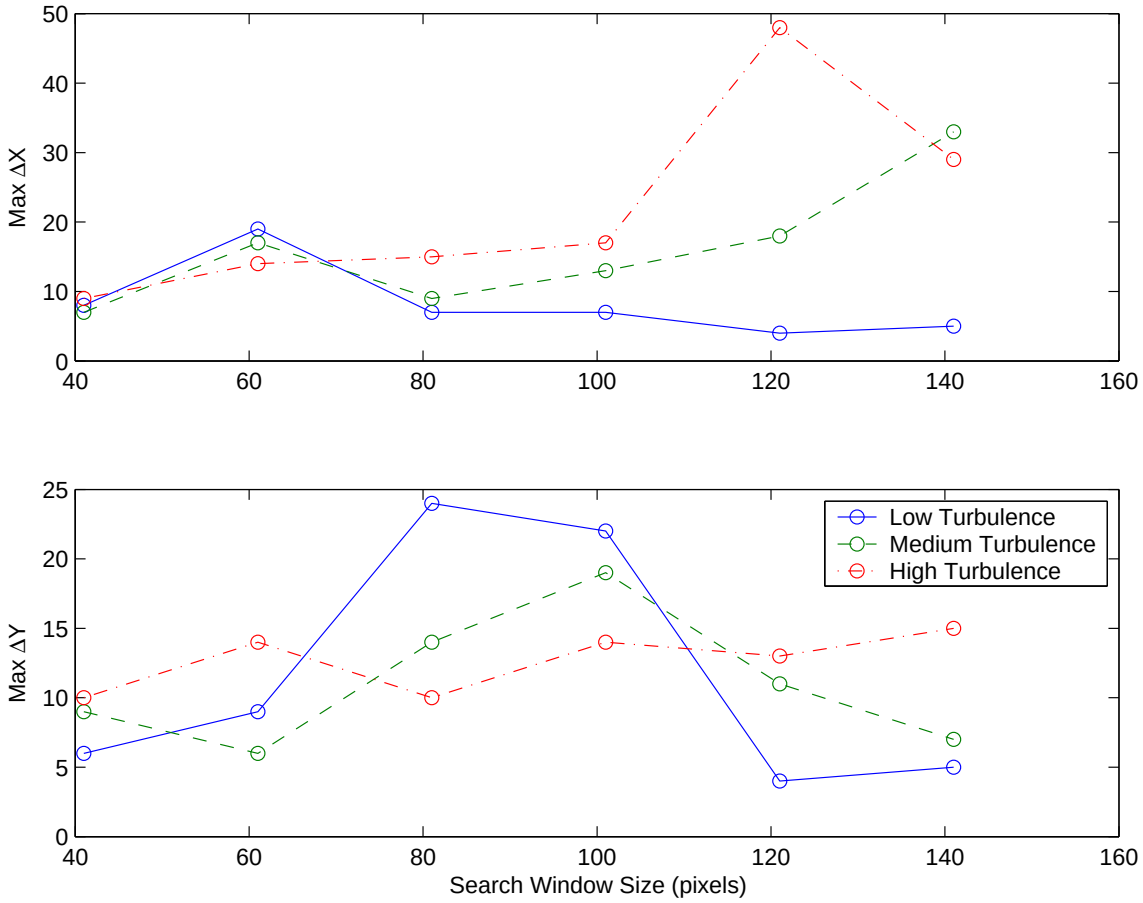



Figure 4.31: Maximum frame-to-frame pixel movement for tests shown in Figure 4.30.

beginning specified a search window size of 81 pixels. The reason for this was that processing time was not critical in these tests, and a window size of 81 pixels allows extra room to handle occasional, unusually-large variations in the pixel coordinates.

The results for the template-window-size tests are displayed in Figure 4.32. The coherence-count and max-MPE-error plots in this figure make it clear that tracking performance degrades with large template windows. This is contrary to what would be expected, since a larger template window should make it easier to uniquely identify a feature in subsequent camera images. However, referring back to Figure 4.29, we see that if the search window size remains constant, then increasing the template window size will actually shrink the effective search window. For a template window size of 61 pixels, the effective search window is reduced to 21×21 pixels,

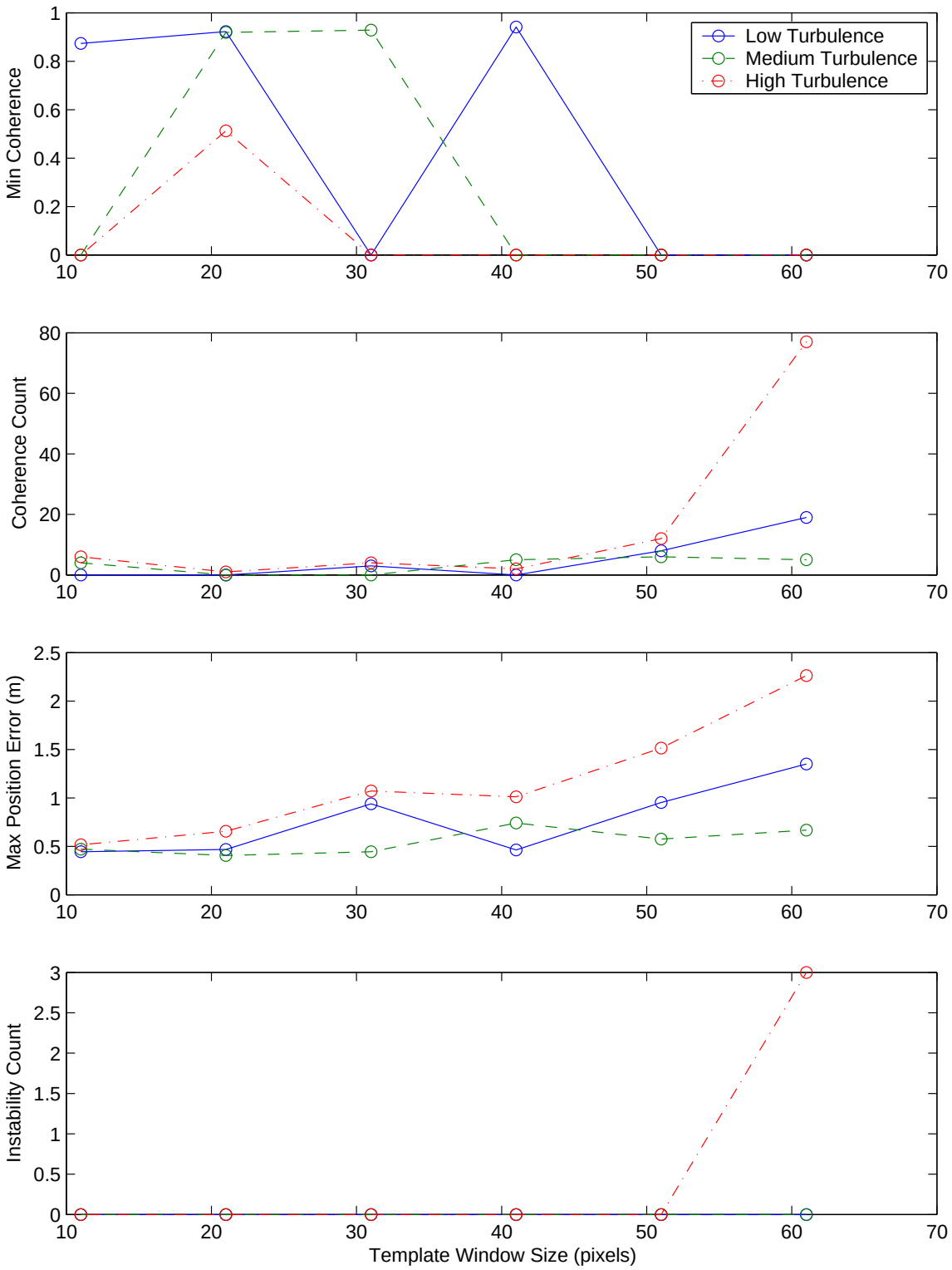



Figure 4.32: Test results of tracking performance vs template window size.

which will only accommodate a pixel-coordinate change of at most 10 pixels from frame-to-frame.

However, reasoning still stands that on the other extreme, a small template window will contain too few unique features to permit matching with a high level of confidence. In the minimum-coherence plot, it appears that a template window size of 11 pixels is sufficient for tracking in low-turbulence conditions, but it suggests that performance may be suffering in more extreme wind conditions. The best performance was achieved at a template window size of 21 pixels, which is suggested by the general local minima in the coherence-count plot, and the general local maxima in the minimum-coherence plot.

With regard to the optimum template window size, if processing time is not an important factor, then tracking performance might be improved by using a larger template window, provided that the search window size is also increased to compensate. However, increasing both window sizes simultaneously will quickly exhaust computational resources. Furthermore, since a template window size of 21 pixels has been observed to work sufficiently well, no additional tests were conducted to evaluate larger template window sizes.

### 4.2.5 Tracking During Descent

The last test that will be presented here was conducted to validate the effectiveness of the tracking algorithm during descent. This test used a procedure as follows:

- Hover at 12 meters above the ground facing into the wind, select the best feature to track from a region in the center of the image, and then switch over to navigation based on position estimation. The tracking window and search window sizes are set to 21 and 81 pixels respectively.

- Hold the hover for 30 seconds, then descend to 3 meters along a 30-degree glide slope and continue to hover. After two minutes from the start of the test, switch back to GPS navigation.

This test procedure was only executed under conditions of high turbulence to obtain data representative of a worst-case scenario. Figure 4.33 displays the first and last camera images from a sample test run, with the initial feature (from the first frame at 12 meters) marked by a large "X", and the final feature (from the last frame at 3 meters) marked by a small "X". The marks were manually added to both frames for reference. These camera images exhibit some drift similar to previous tracking tests, but they also illustrate that it was possible to track a feature with good results even when the images are changing significantly over time.

Figure 4.34 displays a plot of the true (GPS) coordinates along side the estimated position in each of the three world axes throughout the descent test procedure. These values are defined relative to the starting coordinates. The descent path is evident in the northing and height data at about 40 seconds into the test as the vehicle moved forward and down along a 30-degree glide slope to keep the tracked feature within the image. The easting coordinate remains relatively constant since the descent path did not require motion in the lateral direction.

Figure 4.35 shows the difference between the GPS and MPE signals. This plot demonstrates that the error in the position estimate for this test was generally less than 0.4 meters. This means that in a real landing scenario, the aircraft would have landed less than 0.4 meters from the original safe landing point output by the SLAD analysis. The error in the height estimate throughout this test is seen to be nearly constant because it is mostly dependent on the rangefinder readings, which are perfect in simulation. Finally, Figure 4.36 plots the tracking coherence values recorded during the test, and indicates that the algorithm processed every frame with a very high level of confidence. A very slight drop in the coherence value is evident at one point just after the start of the descent, but it is much too small to be cause for concern.

### 4.2.6 Tracking / Position-Estimation Testing Summary

Test results for tracking and position estimation have been presented to show that tracking is robust, and is capable of providing adequate data for position estimation calculations. It was shown that good tracking performance is possible under

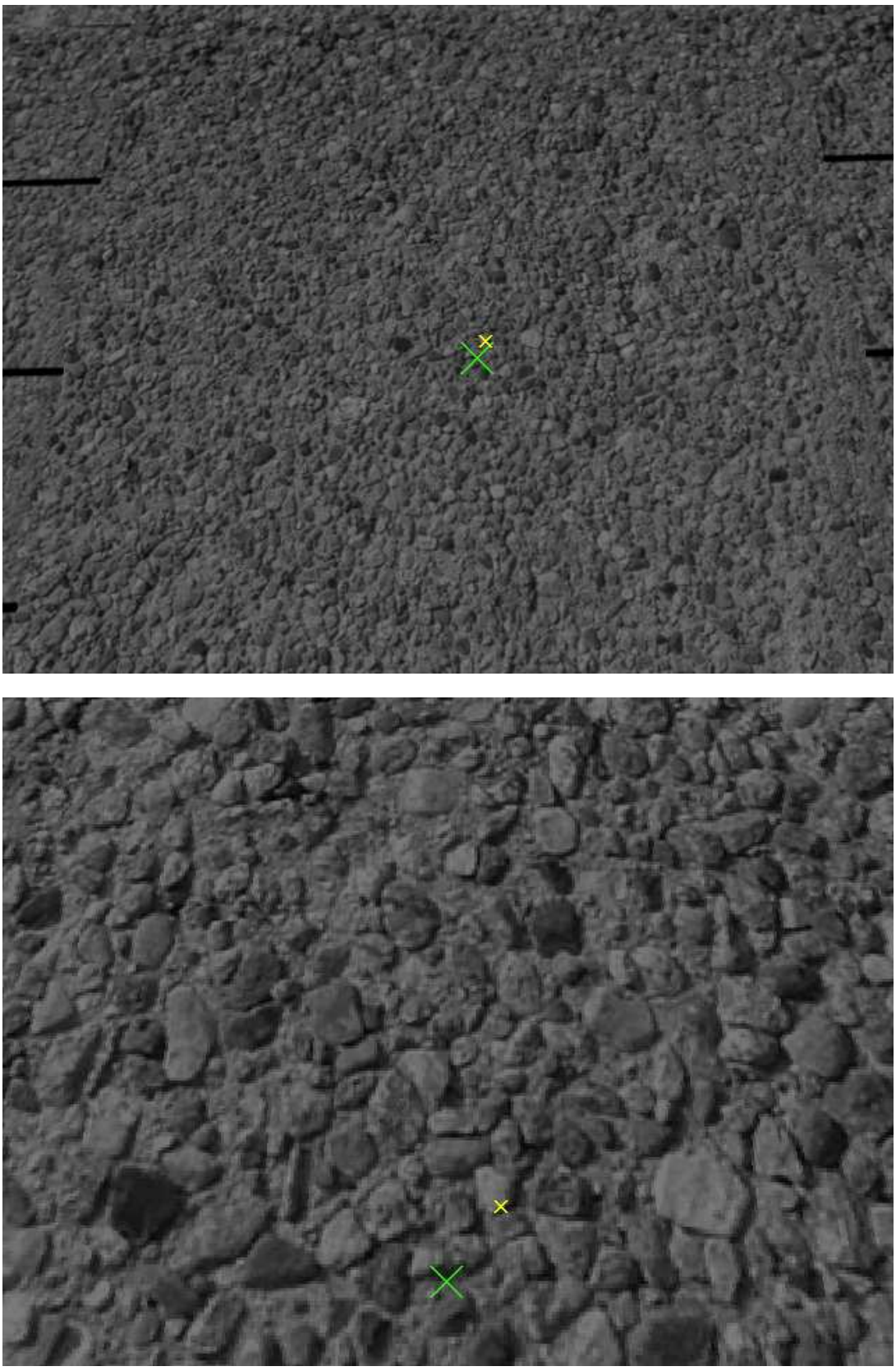


Figure 4.33: First and last frames from a tracking descent test.

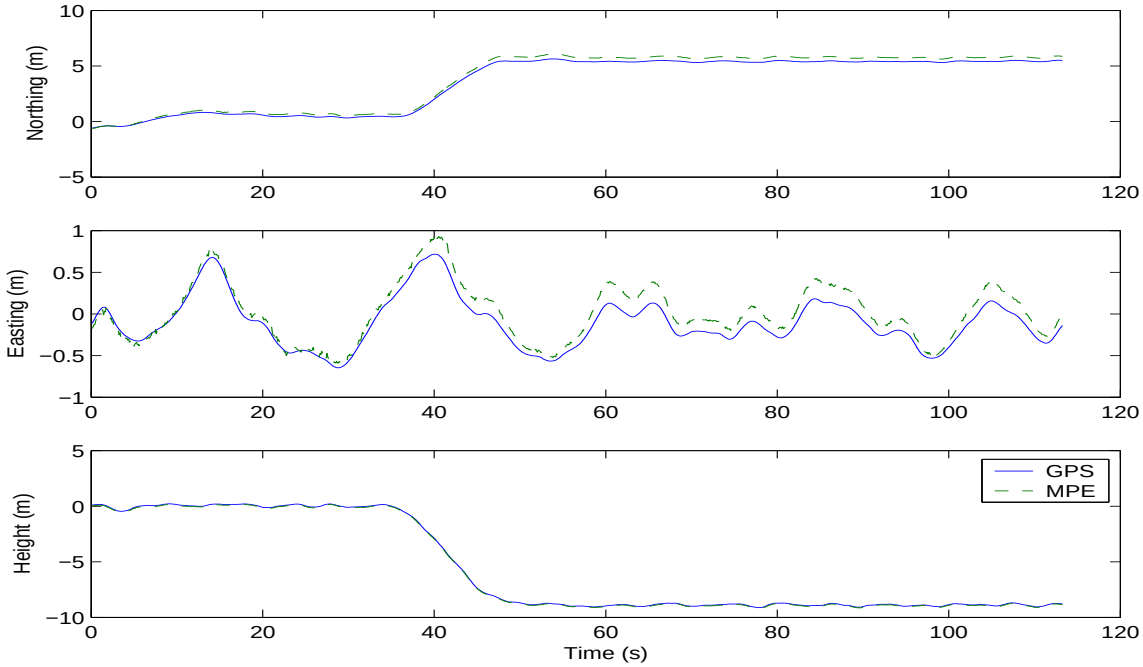



Figure 4.34: Estimated position and true GPS position recorded throughout a tracking test involving a slow descent.

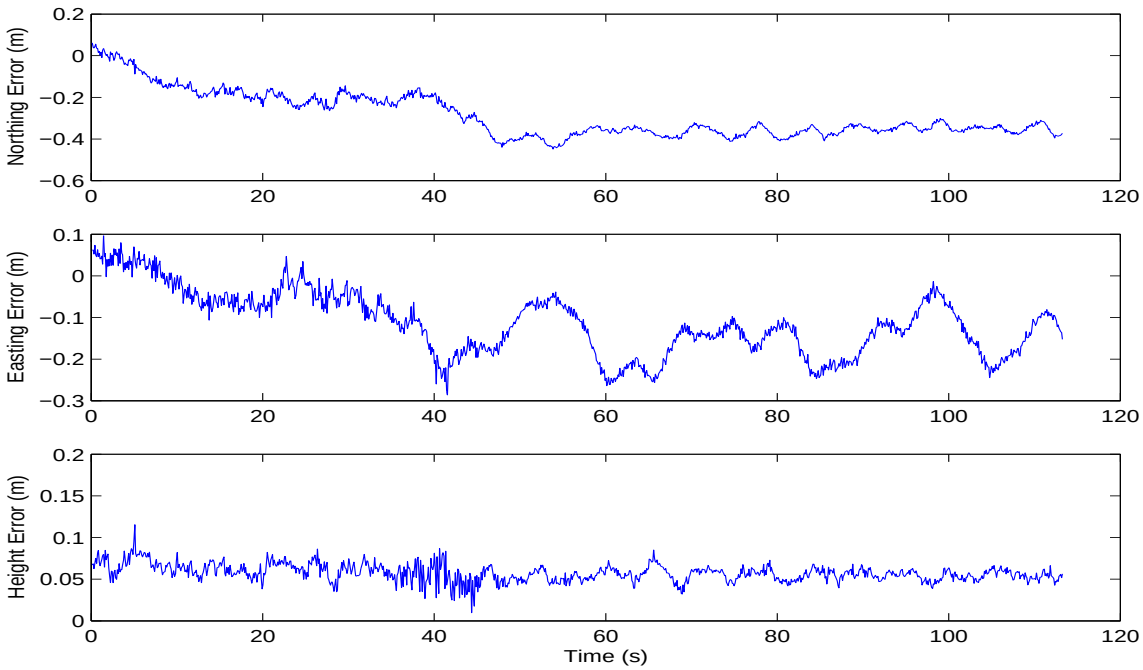



Figure 4.35: Position estimate error observed in Figure 4.34.

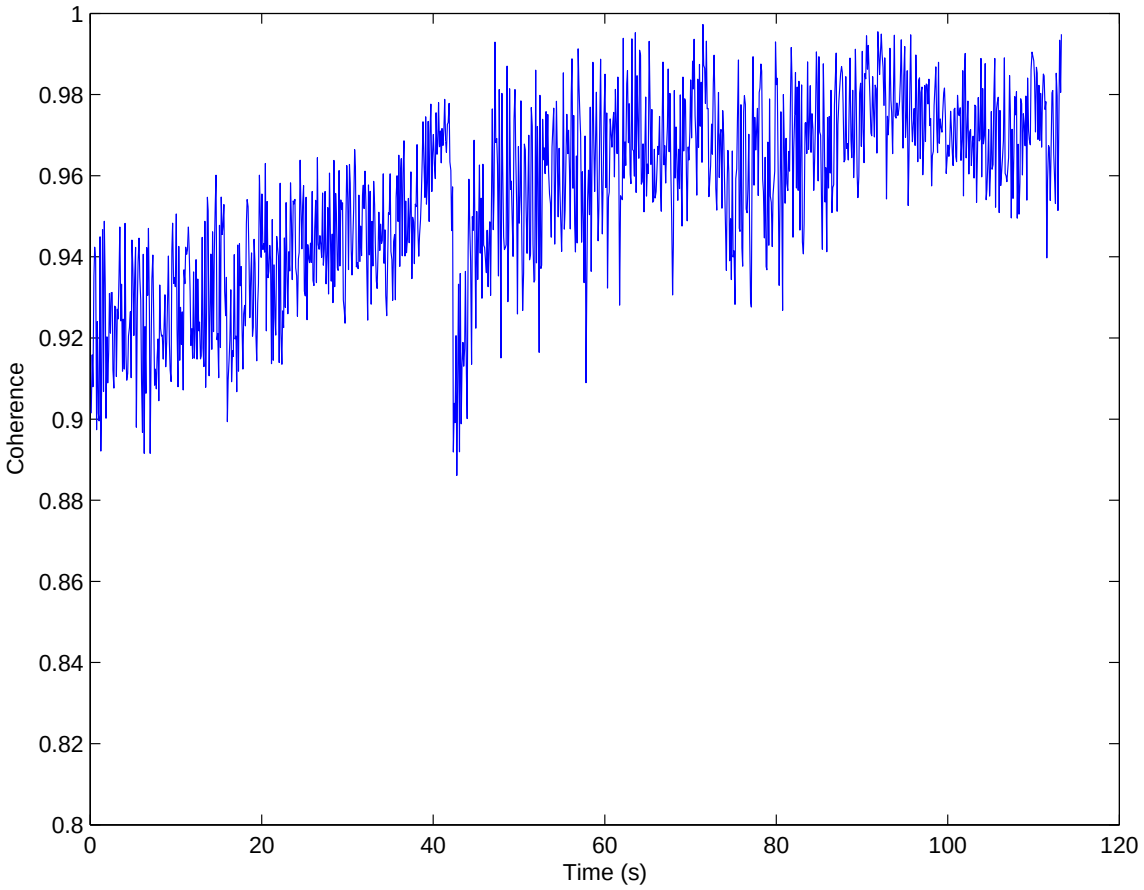



Figure 4.36: Tracking coherence values recorded during the test represented in Figure 4.34.

extreme wind conditions when the helicopter is facing into the wind. Data was also provided to show that minimum template and search window sizes of 21 and 61 pixels respectively are required for most conditions. The lower altitude limit on safe tracking and MPE navigation was established to be between 2.5 and 3.0 meters. Finally, the maximum error in the position estimates throughout all of the tracking tests was generally observed to be less than 1 meter. These conclusions indicate that the tracking and position-estimation algorithms will successfully accomplish the tasks necessary to support an autonomous landing in a PALACE mission.

# Chapter 5

# RMAX Helicopter Implementation

The focus of this thesis has largely been on the development of a simulation environment, followed by an evaluation of the vision algorithms' performance in simulation. However, this work was only a means to an end as the ultimate goal of the PALACE project is to demonstrate autonomous landing technologies using actual flight hardware.

Although Hintze conducted some tests of the vision algorithms using data collected in flight, the data was processed offline. As the ultimate goal of the PALACE program is to demonstrate autonomous landings in flight, the next step was to adapt the vision algorithms to the flight hardware so that real-time processing of sensor data could be performed. Therefore, a secondary goal of this thesis was to demonstrate individual real-time execution of the stereo/SLAD, tracking, and position-estimation algorithms in flight. This chapter describes the integration of the machine vision algorithms with flight hardware and presents some preliminary results with the flight hardware.

## 5.1 Flight Hardware

The hardware that is to be used for the PALACE demonstrations is the Yamaha RMAX helicopter (see Figure 5.1). The RMAX has a 3.63 meter rotor diameter, an empty mass of 66 kg, and a maximum payload of 28 kg. It has a hover endurance of about 1 hour, and includes an oversized generator that provides 100 watts of power for research hardware. The Autonomous Rotorcraft Project (ARP)

[24] at NASA Ames Research Center has fitted this helicopter with a variety of sensors, as well a compact PCI computer to process sensor signals, and radio modems for transmitting the data to a ground station. The sensors specifically of interest to this thesis include a Riegl LD90-3 laser rangefinder, and two grayscale Point Grey Flea cameras capable of capturing 640×480 images at 60 frames per second. The cameras are rigidly mounted on the ends of a 1 meter rod that can be rotated downward to 90 degrees to provide stereo images of the ground directly underneath the aircraft.

## 5.2 Stereo Ranging / SLAD Integration and Evaluation

The installation of the stereo ranging and SLAD code on the flight vehicle required only a couple of small changes from the code used in simulation. The most significant of these changes was that the interface had to be adapted to the camera images. Since it was anticipated that the stereo algorithm would not be the only process needing access to the camera images, a separate program was written to capture camera images at a user-specified rate and store them in shared memory. This program is referred to as JPEGTX. A common locking mechanism was also put in place to prevent other processes from trying to access the images in shared memory while they were being updated by the JPEGTX process.

With this simple arrangement in place, it was possible to execute a stereo analysis of current camera images on command. Testing was not conducted in flight, but real-time tests with the RMAX hardware were performed with the vehicle on the ground.

As described in Section 4.1.2, a test was performed in simulation with the cameras pointing straight down at a box on the ground, and the stereo estimate of the range was compared to the actual range. This test was repeated using the RMAX stereo hardware, except that the RMAX was manually placed on the ground at various distances from hangar doors (see Figure 5.2). A measuring tape was used to measure the distance from the left camera to two points on the hangar doors. The program described in Section 4.1.2 was also used here to record the stereo estimate of the minimum, maximum, and average range to the two points on the hangar doors.


Figure 5.1: Yamaha RMAX helicopter.

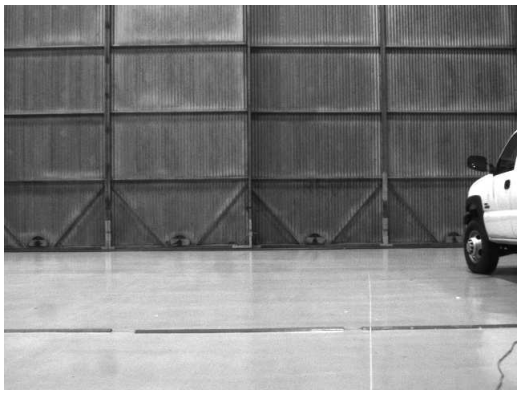


Figure 5.2: Stereo images from the RMAX cameras for range accuracy evaluation.

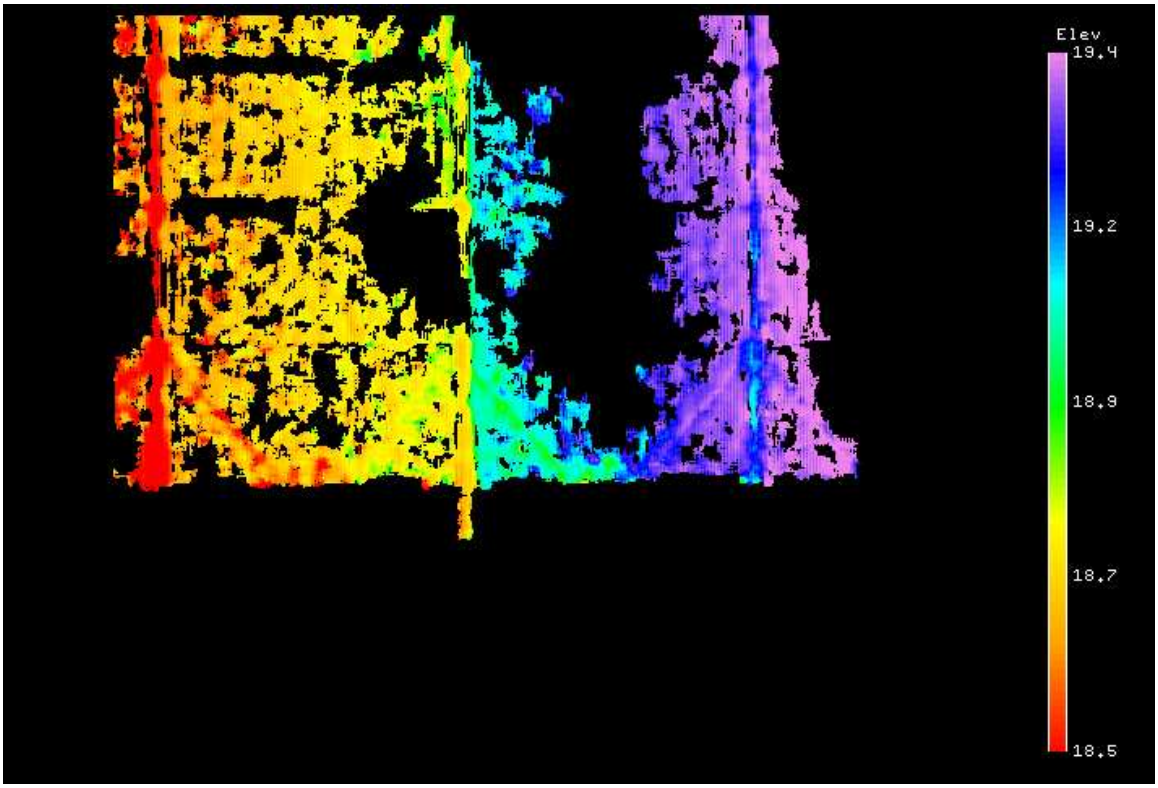



Figure 5.3: Representation of stereo ranges calculated based on the images in Figure 5.2. Legend displays a scale in units of meters.

Figure 5.3 displays a representation of the stereo ranges calculated for the images in Figure 5.2. The abundance of black areas in the figure suggest that the image texture is marginal for purposes of stereo processing, but there is sufficient data to collect range estimates corresponding to a few points on the hangar doors. The range values in the figure correctly portray the fact that the hangar door in the left half of the image stands in front of the hangar door in the right half of the image. It is also apparent that the optical axis of the camera is not perfectly perpendicular to the hangar door because the left edges of the doors are reported to be about 20 cm closer to the camera than the right edges.

The comparison of the estimated ranges and the actual ranges are shown in Figure 5.4. The hangar did not allow for testing out to 30 meters as was done in simulation, but this data shows that the range is estimated within 2.3% for a range up to 23 meters, which is comparable to the 1.3% error observed in simulation. The slightly less-accurate results are to be expected since the stereo setup involves imperfect cameras and a pin-hole camera model. Nevertheless, the accuracy is sufficient to establish confidence that the results of the simulation experiments can be used to approximate real-time performance.

## 5.3 Tracking / Position-Estimation Integration

The monocular tracking and position-estimation algorithms are more simple than the stereo methods, which allowed for some preliminary real-time tests in flight. As with the installation of the stereo code, the tracking algorithm that was used in simulation required only slight changes to be adapted to the RMAX hardware. The most significant code change was to read the camera images from the JPEGTX shared memory. Each cycle of the tracking algorithm consumed 60-70 milliseconds on the RMAX processor (compared to 5 milliseconds on the simulation desktop computer), but was still fast enough to do tracking at 10 Hz with template window and search window sizes of 21 and 81 pixels respectively.

To validate the algorithms, there is a requirement to store the images for post-processing. Besides storing camera images in shared memory, the JPEGTX program

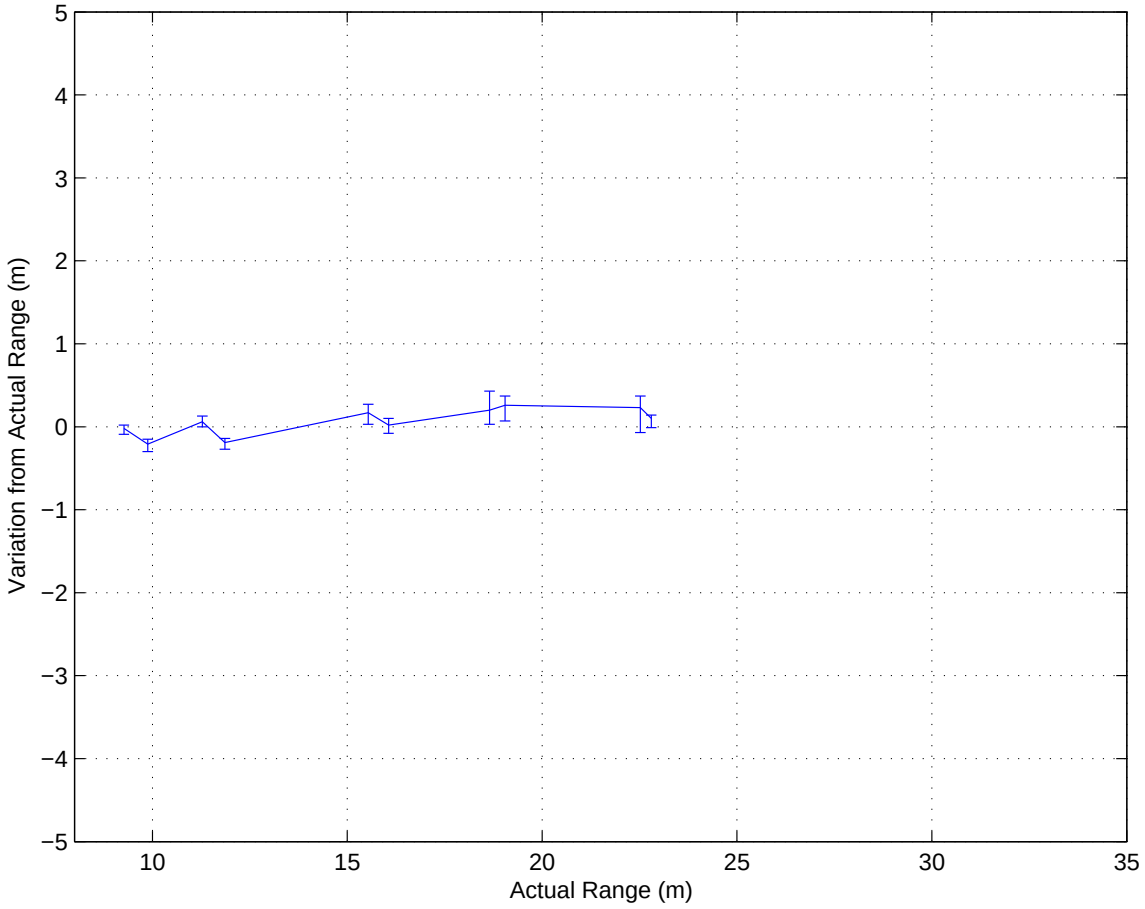



Figure 5.4: Results from stereo range accuracy tests using the RMAX cameras.

was responsible for compressing the images and transmitting them to the ground station. The ground station then displayed the images on a computer screen to show real-time video from the helicopter's camera. The JPEG standard image format was chosen since the compression utilities required very little processing time and the resulting images were small enough (20:1 compression) to be transmitted over the existing wireless link at 10 Hz.

For the flight tests of the tracker, the first step was for the ground-station operator to select the feature to be tracked. Figure 5.5 shows one frame from the ground-station where a user was using the mouse to select a general region of the image for the tracker to start tracking. In order to visualize the activity of the

Figure 5.5: Ground station selection of a general tracking region in an RMAX camera image.

tracking algorithm while it was running on the RMAX computer in flight, some code was added to the tracking program to send status information to the ground station via DOMS. The DOMS message included the $x, y$ pixel coordinates of the feature it was tracking and the corresponding coherence value. As the ground station displayed the JPEG images, it used this extra data to paint an "X" on the pixel that was being tracked. Figure 5.6 shows a screenshot from the ground station with an "X" painted on the runway light that the tracker chose to begin tracking. Figure 5.7 shows another frame some time later after the RMAX had descended from about 30 meters to about 10 meters while tracking the feature from the first frame.

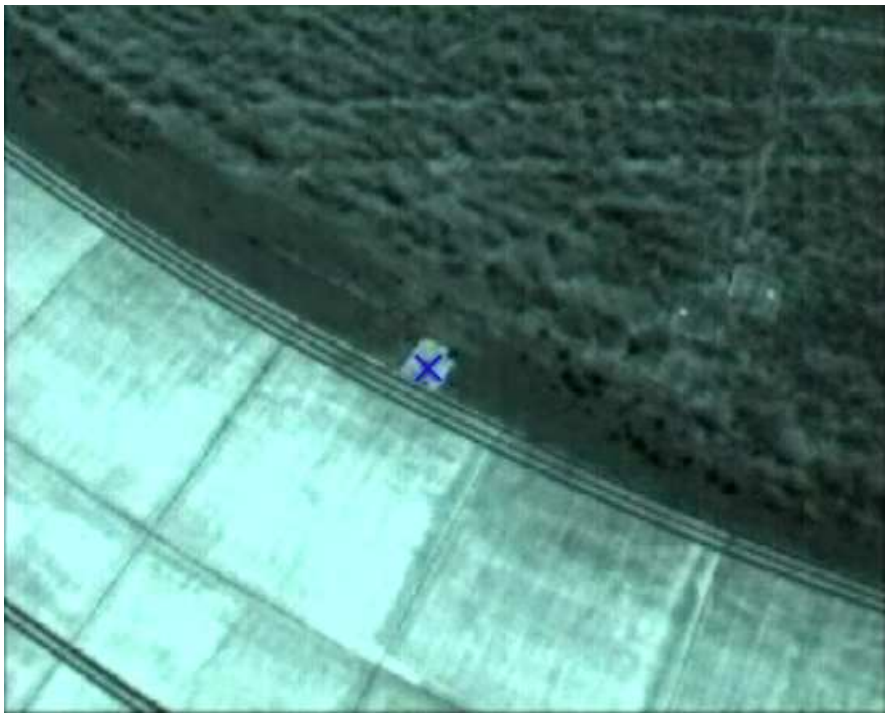


Figure 5.6: RMAX tracking frame from about 30 meters above the ground.

Figure 5.7: RMAX tracking frame from about 10 meters above the ground.

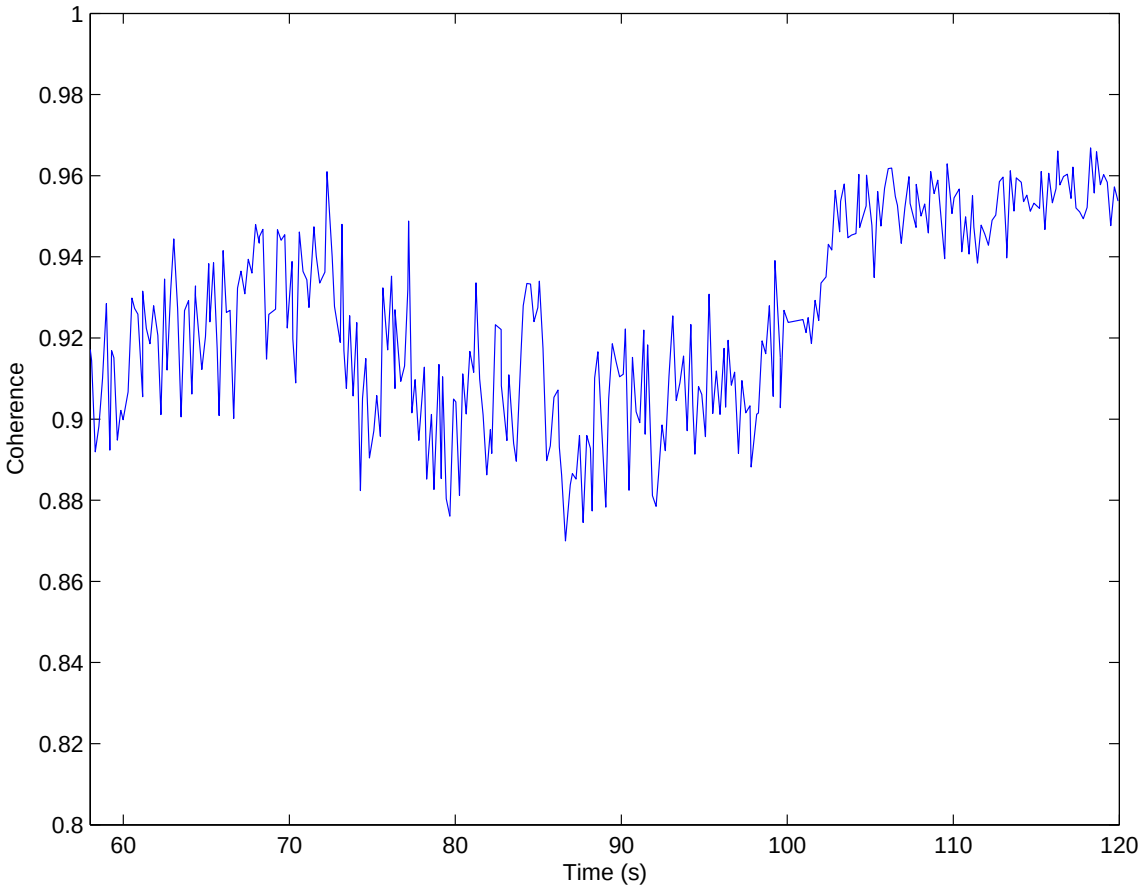



Figure 5.8: Coherence values for RMAX tracking during a descent.

One issue that was observed in these preliminary tracking tests was that the images coming from the vehicle were not time synchronized with the DOMS data messages coming from the tracker. This meant that the "X" was not painted on the same feature in every frame, and it initially appeared that the tracker was not working well. In contrast to this, the tracking coherence values for this test (plotted in Figure 5.8) indicated that the tracker had processed every frame with a high degree of confidence. The solution to this time synchronization issue was to add a delay to the camera images (which arrived at the ground station before the tracking pixel coordinates) in order to synchronize the DOMS data and the images. This test illustrated the importance of ensuring that data from different sources is time synchronized.

Table 5.1: RMAX tracking performance data.

| **Parameter** | **Value** |
|---|---|
| Minimum Coherence Value | 0.87 |
| Maximum $\Delta$X | 29 pixels |
| Maximum $\Delta$Y | 29 pixels |
| Average $\Delta$X | 5.66 pixels |
| Average $\Delta$Y | 5.84 pixels |
| $\Delta$X Standard Deviation | 5.52 pixels |
| $\Delta$Y Standard Deviation | 5.62 pixels |
| Number of Frames Processed | 595 frames |
| Number of $\Delta$X or $\Delta$Y $\geq$ 20 pixels | 15 |

Table 5.1 shows some measured values for the flight tracking test involving hovering and descent over the period of about 60 seconds. These results indicate that the $\Delta$X and $\Delta$Y changes in the pixel coordinates are usually much smaller than the search window size. This confirms the observation from the coherence values in Figure 5.8 that the feature did not move out of the search window, and that the tracking algorithm successfully identified the feature in each frame.

## 5.4 Laser Rangefinder

One assumption of the position-estimation routine is that the distance is known to the feature in the center of the camera image. Because a camera image only provides 2D information, a laser range finder is used to provide information in the third dimension and enable monocular position estimation. Due to time constraints, the work with this algorithm in hardware was limited to preliminary ground testing of the Riegl laser rangefinder.

Before installing the rangefinder on the RMAX, it was decided that some tests should be done to verify the manufacturer's claim of $\pm$5 cm accuracy with a measurement uncertainty of $\pm$3 cm. Software was written to interface with the rangefinder and broadcast a DOMS message containing sensor readings. The rangefinder was placed on a cart pointing parallel to the ground, and a black poster board was placed

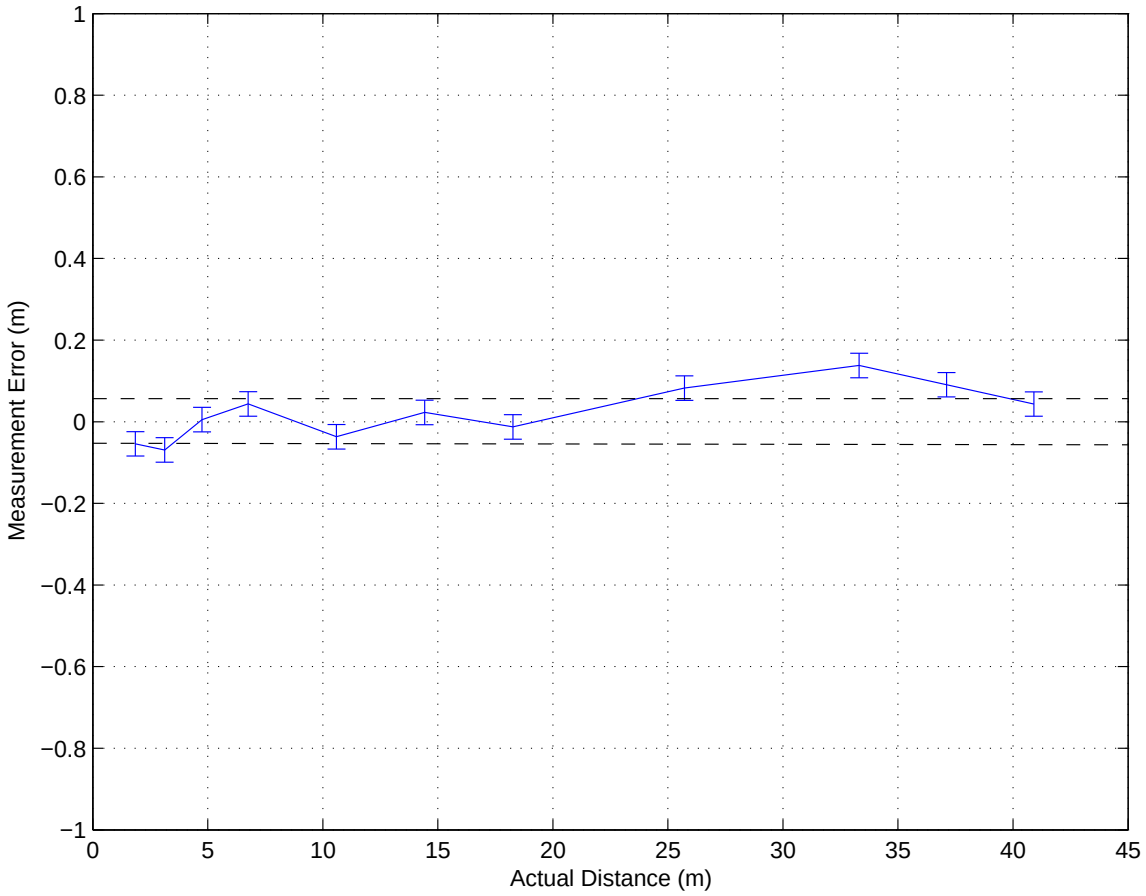



Figure 5.9: Results from tests of rangefinder accuracy.

at various distances ranging from 2 to 40 meters. The sensor readings are compared to the actual values by plotting the error (see Figure 5.9).

In this plot, the error bars represent the ±3 cm uncertainty, and the horizontal lines represent the ±5 cm claimed accuracy. It can be seen that most of the points' error bars fall within the ±5 cm margin, with the exception of the points at larger distances. However, the error for these points is still under 0.4%. These results indicate that the data from the Riegl laser will be more than adequate for the purpose of position estimation.

# Chapter 6

## Summary and Conclusions

An integrated simulation environment was constructed for the development and performance evaluation of machine vision autonomous landing technologies that are being investigated as part of the PALACE project. The primary goal of the PALACE project is the fully-autonomous landing demonstration of an RUAV in non-cooperative environments without the use of GPS. In addition to serving as a development and evaluation tool, the integrated simulation also provides a level of risk reduction for taking the machine vision technologies to flight.

A mission manager was developed that handles the entire PALACE mission from take-off to landing, although the focus of the mission manager is on the landing phase of the mission. Landing procedure logic was developed to manage the vision algorithms for the purpose of identifying a safe landing site and maintaining a current estimate of the aircraft position with respect to the landing site until the vehicle was safely on the ground. The feedback of the vision-based position estimate into the control system required special attention to minimize the system time delays and to time synchronize the signals from different sources.

Results were presented for the vision algorithms under a variety of conditions to evaluate performance, quantify effectiveness, and identify limitations. The stereo/SLAD algorithm was shown to be able to detect and choose landing sites with greater than 70% accuracy for most scenarios. Tests with the tracker/position-estimation algorithm showed a maximum error of about 1 meter during a two-minute

hover. Similar results were seen in descending flight and in adverse weather conditions. Optimum configuration parameters were identified for each of the vision algorithms.

Based on the results presented in this thesis, the following conclusions can be drawn:

1. Machine vision algorithms can be integrated and successfully applied to the autonomous landing task. Artificial imagery from simulation is an effective tool for the initial integration and testing of the vision algorithms.

2. The time synchronization of the various signals, including sensor measurements and camera images, proved to be important in providing accurate position estimates. Even small delays can have considerable influence on the system.

3. There is a trade-off between the required accuracy of the vision algorithms and the amount of processing time available. Higher resolutions provide more data on which to base judgements, but also significantly increases resources needed to process the data. Careful selection of the configuration parameters is required to produce acceptable levels of performance of the vision algorithms when the processing power is limited.

4. Stereo ranging accuracy on the hardware (2.3%) was comparable to the accuracy observed in simulation (1.3%). Tracking coherence values recorded in flight were seen to be comparable to those observed in simulation. This comparison between simulation and flight results suggests that the overall performance of the algorithms in flight would be similar to the performance in simulation.

## 6.1 Future Work

Some work has yet to be done before the final PALACE demonstration will be possible. This work includes the RMAX integration and testing of the laser rangefinder with the position-estimation algorithm. The results will need to be validated before the loop may be closed on the position-estimation output. The Mission

Manager and ground station software must be integrated with the RMAX hardware. These steps will facilitate a more thorough validation of the vision algorithms' performance that was documented from the tests in simulation.

In addition, Chapter 4 suggested the need for some fine-tuning of the vision algorithms. The test results made it clear that the SLAD algorithm must be examined more closely to correct or explain certain patterns of behavior in some scenarios. The current implementation of the algorithm will require more flexibility to dynamically define the maximum roughness constraint. Some additional testing is recommended to assess other aspects of stereo-ranging / SLAD performance, such as the influence of shadows, reflections, smoke, glass surfaces, thin wires, or the robustness to camera calibration errors and camera misalignment.

Several improvements are also necessary for the tracking algorithm. Due to the lower limit on the altitude at which the tracker is useful, a dead-reckoning approach must be developed and integrated to support the last few feet of descent in the landing phase of an autonomous RUAV mission. The tracking or stereo algorithm must be made to run in a separate thread so that tracking (and position-estimation) is not interrupted while stereo-ranging / SLAD processing is being done.

Finally, there are many possibilities for improving the robustness of the tracking algorithm. There is currently no support for handling dangerously-low coherence values. Since this often indicates that the feature has been lost, more appropriate action could be taken other than to simply press forward. If there is additional processing power available, tracking does not have to be limited to 10 Hz. Increasing the tracking frequency might significantly improve tracking performance by reducing the probability of the feature moving outside of the search window. Similar improvements might be realized by dynamically sizing the search window based on inertial measurements. If it is known that the aircraft will be hovering for some time, another improvement might involve holding onto the template window from the first image instead of replacing it with a template from every new image. This would reduce drift by ensuring that features are matched to the original image instead of the feature that happened to be selected in the previous frame. Finally, robustness might

be increased by tracking multiple points around the original feature and using these points as references to help locate the original feature if there is a sudden occlusion or other similar problem.

## Bibliography


[1] Joshua Hintze, "Autonomous landing of a rotary unmanned aerial vehicle in a non-cooperative environment using machine vision", Master's thesis, Brigham Young University, Provo, Utah, 2004.

[2] Joshua Hintze, Mark Tischler, Daniel Christian, Tim McLain, Colin Theodore, and James Montgomery, "Simulated autonomous landing of a rotorcraft unmanned aerial vehicle in a non-cooperative environment", American Helicopter Society Annual Forum, June 2004.

[3] Srikanth Saripalli, James F. Montgomery, and Gaurav S. Sukhatme, "Vision-based autonomous landing of an unmanned aerial vehicle", in *Proceedings of IEEE International Conference on Robotics and Automation*, May 2002.

[4] Sierra Nevada Corporation, "Tactical automatic landing system", `http://www.sncorp.com/uav2.html`.

[5] Sebastian Thrun, Mark Diel, and Dirk Hähnel, "Scan alignment and 3D surface modeling with a helicopter platform", in *Proceedings of the International Conference on Field and Service Robotics*, Lake Yamanaka, Japan, 2003.

[6] Yalin Xiong, Clark F. Olson, and Larry Matthies, "Computing depth maps from descent imagery", in *IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. 392–397.

[7] Bir Bahnu, Subhodev Das, Barry Roberts, and Dave Duncan, "A system for obstacle detection during rotorcraft low altitude flight", *IEEE Transactions on Aerospace and Electronic Systems*, vol. 32, no. 3, pp. 875–897, 1996.

[8] Andrew Joshnson, Allan Klump, James Collier, and Aron Wolf, "LIDAR-based hazard avoidance for safe landing on mars", in *AAS/AIAA Space Flight Mechanics Meeting*, Santa Barbara, CA, Feb. 2001.

[9] Jianbo Shi and Carlo Tomasi, "Good features to track", in *IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, June 1994.

[10] Omead Amidi, *An Autonomous Vision-Guided Helicopter*, PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 1996.

[11] Omead Amidi, Takeo Kanade, and K. Fujita, "A visual odometer for autonomous helicopter flight", in *Proceedings of the Fifth International Conference on Intelligent Autonomous Systems (IAS-5)*, June 1998.

[12] Berthold Horn and Brian G. Schunck, "Determining optical flow", *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[13] Emanuele Trucco and Alessandro Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, 1998.

[14] Stergios I. Roumeliotis, Andrew E. Johnson, and James F. Montgomery, "Augmenting inertial navigation with image-based motion estimation.", in *ICRA*, 2002, pp. 4326–4333.

[15] Andrew Johnson and Larry Matthies, "Precise image-based motion estimation for autonomous small body exploration", Proceedings of the Fifth International Symposium On Artificial Intelligence, Robotics and Automation in Space, June 1999.

[16] Ginés García Mateos, "A camera calibration technique using targets of circular features", V Ibero-American Simposium on Pattern Recognition, Sept. 2000.

[17] Yoram Yakimovsky and Roger Cunningham, "A system for extracting three-dimensional measurements from a stereo pair of tv cameras", *Computer Graphics and Image Processing*, vol. 7, pp. 195–210, 1978.

[18] Roger Tsai, "An efficient and accurate camera calibration technique for 3D machine vision", in *IEEE Conference on Computer Vision and Pattern Recognition*, Miami Beach, Florida, June 1986.

[19] Mark Maimone, "CAHVOR camera model information", `http://telerobotics.jpl.nasa.gov/people/mwm/cahvor.html`.

[20] Jan Roskam, *Airplane Flight Dynamics and Automatic Flight Controls*, Design, Analysis, and Research Corporation, third edition, 2001.

[21] Mohammadreza H. Mansur, Michael Frye, Bernard Mettler, and Michael Montegut, "Rapid prototyping and evaluation of control system design for manned and unmanned applications", Proceedings of the 56th Annual Forum of the American Helicopter Society, May 2000.

[22] Bernard Mettler, Mark Tischler, and Takeo Kanade, "System identification modeling of a small-scale unmanned rotorcraft for flight control design", *American Helicopter Society Journal*, vol. 47, no. 1, pp. 50–63, 2002.

[23] Matthew Whalley, Mark Takahashi, Greg Schulein, Michael Freed, Robert Harris, and Jason Howlett, "Design, integration, and flight test results for an autonomous surveillance helicopter", American Helicopter Society Annual Forum, Jan. 2005.

[24] Matthew Whalley, Mark Takahashi, Greg Schulein, Michael Freed, Daniel Christian, A. Patterson-Hine, and Robert Harris, "The NASA/Army autonomous rotorcraft project", American Helicopter Society Annual Forum, May 2003.