2004-03-12

Autonomous Landing of a Rotary Unmanned Aerial Vehicle in a Non-cooperative Environment using Machine Vision

Joshua Martin Hintze
Brigham Young University - Provo

AUTONOMOUS LANDING OF A ROTARY UNMANNED AERIAL VEHICLE IN A NON-COOPERATIVE ENVIRONMENT USING MACHINE VISION

by

Joshua Hintze

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 Electrical and Computer Engineering

Brigham Young University

April 2004

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a thesis submitted by

Joshua Hintze

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

Date	Randal W. Beard, Chair
Date	Timothy W. McLain
Date	James K. Archibald

ABSTRACT

AUTONOMOUS LANDING OF A ROTARY UNMANNED AERIAL VEHICLE IN A NON-COOPERATIVE ENVIRONMENT USING MACHINE VISION

Joshua Hintze

Department of Electrical and Computer Engineering

Master of Science

Landing an Unmanned Aerial Vehicle (UAV) is a non-trivial problem. Removing the ability to cooperate with the landing site further increases the complexity. This thesis develops a multi-stage process that allows a UAV to locate the safest landing site, and then land without a georeference. Machine vision is the vehicle sensor used to locate potential landing hazards and generate an estimated UAV position. A description of the algorithms, along with validation results, are presented. The thesis shows that software-simulated landing performs adequately, and that future hardware integration looks promising.

ACKNOWLEDGMENTS

I wish to express my gratitude to my graduate committee for their comments and criticism that I used to help complete this thesis. I would also like to give thanks to the many employees located at the NASA Ames Research Center who have offered their help and support. Most of all, I owe a debt of gratitude to my wife Crystal and son Hunter, for their patience and love they have shown me over these many trying months.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Unmanned Aerial Vehicles (UAVs) have achieved considerable attention with the popularity and success of programs such as the Predator and Global Hawk, which have been used in recent conflicts[1]. There are many useful applications for UAVs including military, homeland security, crop dusting, traffic monitoring, glacial movement research, etc. There are many types of UAVs falling into two distinct categories: fixed-wing and rotary. Rotary UAVs (RUAVs) include helicopters, Micro Air Vehicles (MAVs), and Organic Air Vehicles (OAVs)[2]. Examples of fixed-wing aircraft are the Global Hawk and Predator. An important aspect of flight is landing, which is a complex problem when done autonomously.

1.1 Problem Statement

This thesis work was conducted in support of an on-going research program at NASA Ames Research Center called the Precision Autonomous Landing Adaptive Control Experiment (PALACE). The primary goal of the PALACE program is to develop enabling technologies for autonomous landing of rotorcraft in non-cooperative environments. Non-cooperative sites are those areas that have no pre-positioned objects, beacons, or landing area markings that the vehicle may use to guide its descent. These sites may also contain unknown terrain features and obstacles that must be identified and avoided during the landing task. Non-cooperative sites of interest include building rooftops in an urban setting or rugged terrain in remote locations.

When operating in urban environments where buildings may be closely spaced, it is common for Global Positioning Systems (GPS) to fail due to occlusion of a direct line of sight to the orbiting GPS satellites [3]. Therefore, this research assumes that GPS positioning is unavailable during the landing approach.

1.2 Complexity

The complexity in this problem comes from the fact that there is no *a priori* information of the landing site. Any number of objects could hinder and obstruct a successful landing. For example, consider the case of landing on a rooftop. Possible hazards to recognize include ventilation systems, antenna/satellite systems, slanted rooftops, edges of the building, and guy wires. The UAV must successfully identify these obstructions and choose a safe landing site, if one exists.

Another difficulty comes from the fact that inertial positioning is not available. One could integrate the outputs of the inertial measurement units, such as accelerometers and rate gyros, but doing so does not provide accurate information due to sensor noise. Therefore, alternative means of referencing the position of the vehicle during autonomous landing needs to be invented.

1.3 Objectives

There are three segments to a complete autonomous landing of a UAV which are illustrated in Figures 1.1, 1.2, and 1.3. First the UAV needs to navigate to the general area in which the landing is to take place. At higher altitudes we assume the UAV is safe from obstacles and that GPS positioning is available. Thus, the UAV may be controlled by a standard waypoint following routine. At the final waypoint the UAV holds its position and hovers at a predefined altitude and heading. The altitude is chosen so that the machine vision cameras maximize their searching area and minimize stereo ranging errors. A fuzzy logic controller determines the optimal approach heading by using information such as sun angles, time of day, and desired landing orientation.

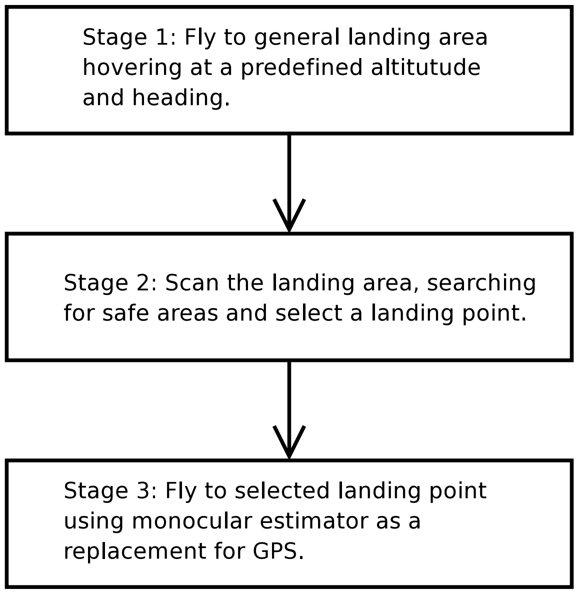

Stage 1: Fly to general landing area hovering at a predefined altitutude and heading.
Stage 2: Scan the landing area, searching for safe areas and select a landing point.
Stage 3: Fly to selected landing point using monocular estimator as a replacement for GPS.

Figure 1.1: Three stage flow chart

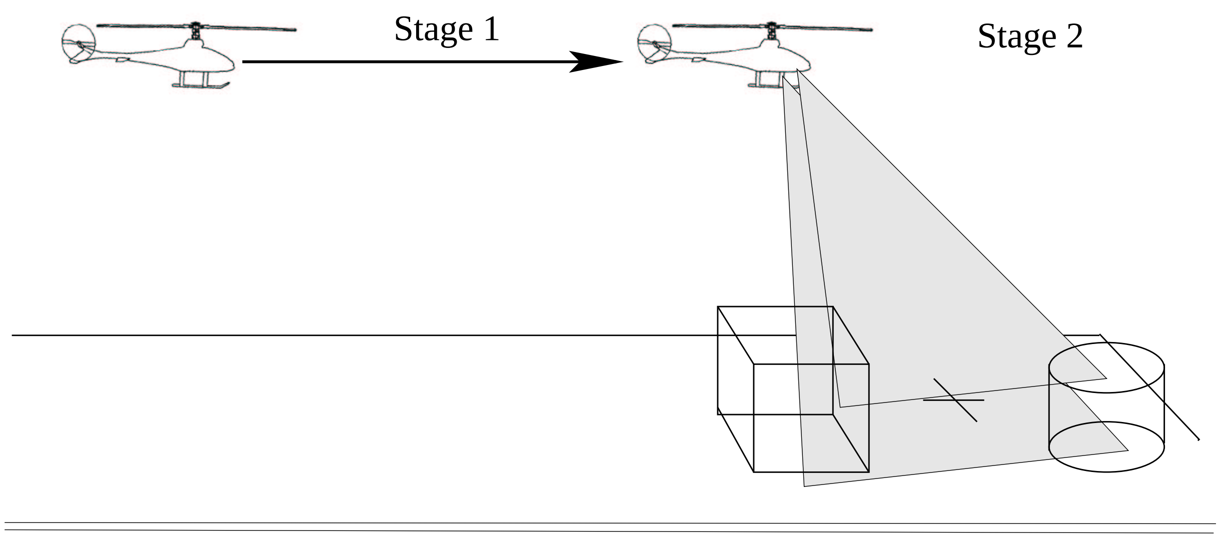


Figure 1.2: Stages 1 and 2 of the landing approach

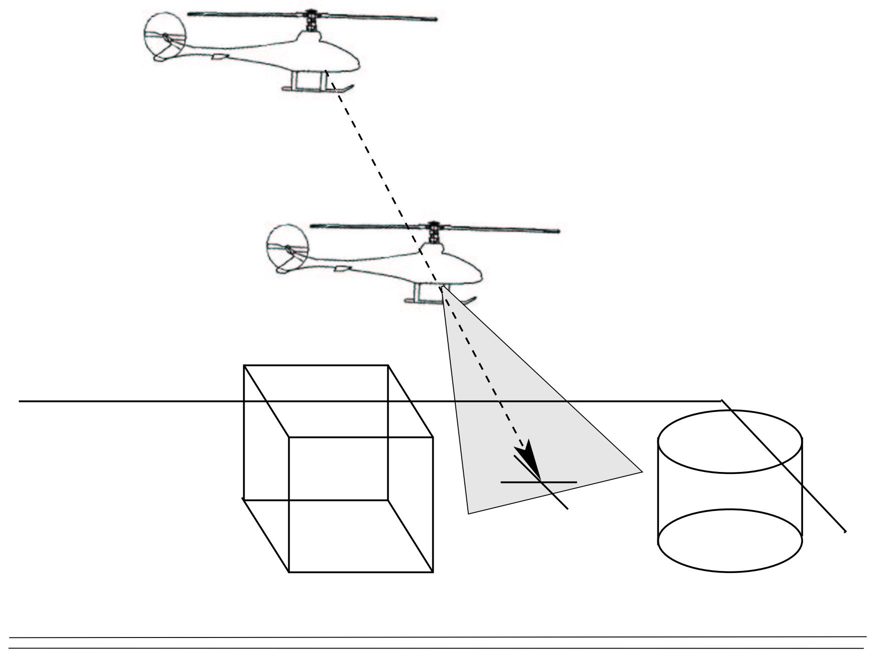


Figure 1.3: Stage 3 of the landing approach

In the second phase of the landing process, a safe landing point must be chosen. Criteria for a safe landing site may include landing area slope and distance from hazards. A hazard is any obstacle that may interfere with the UAV's glide slope during descent. If there are no safe landing sites within the general area, the UAV should carry out the procedure in another area.

Once a landing location has been chosen, the UAV can reference the location to itself and extract a position. Monocular tracking, along with distance to the tracking point and attitude information, may then be used to calculate the UAV's relative position. The control objective is then to decrease the tracking error between the UAV's calculated position and the ground location that is being tracked.

The objective of this thesis is to develop algorithms that execute the second and third phases of a complete autonomous landing. The algorithms must be able to identify safe landing areas and then produce estimated vehicle positions for use in the landing controllers.

1.4 Related Work

UAV flight is a broad area of research with many different areas of study. There has not been any direct research for landing UAVs autonomously in non-cooperative environments. However, related work exists and was used as a starting point to this thesis. A summary of approaches that helped guide the direction for the completion of this thesis follows.

1.4.1 Safe Landing Methods

A majority of autonomous landing research falls under the category of cooperative environments. There are a number of different ways to create a cooperative environment. Some of the more popular forms are millimeter wavelength radar transmitted from a base station, visual landmarks, and glide slope beacons used in commercial airliners. Two examples of using visual landmarks in a cooperative landing are shown in [4, 5] . In [4], Montgomery presents a vision-based helicopter landing system that navigates a UAV to an area which contains a landing pad. The landing pad has a painted “H” which guides the UAV to the predefined landing point. Once the UAV’s vision system has identified the landing spot, the vision controller then sends orientation and velocity commands to the navigation controller.

The Sierra Nevada Corporation developed a millimeter wavelength radar for use in autonomous landing [6]. Tactical Automated Landing System (TALS) is an all-weather ground station that directs UAVs where to fly to avoid unsafe regions. This system requires a person to setup a ground station, but has had over 400 successful Tactical Unmanned Aerial Vehicle (TUAV) recoveries since 2001.

Both of the above methods work well in certain circumstances, but this thesis assumes no prior knowledge of the landing area and no ground support. Therefore, the aforementioned research is not well suited for solving the current problem.

Navigation in non-cooperative environments requires a sensing mechanism for obstacle avoidance. Mobile robot researchers have been investigating this problem for many years. There are many different sensing packages that have been tested, including laser range finders, machine vision, IR sensors, and temperature sensors.

The two technologies that are best suited for the helicopter platform are 2D scanning laser range finders and machine vision.

A 2D scanning laser range finder works by sending out a laser pulse over a grid. By measuring the time of flight and knowing the propagation speed of the laser, a 3D point cloud may be constructed. In [7], Thrun, Diel, and Hänel demonstrate an aerial system flown under manual control for acquiring high-resolution 3D models of urban structures. With the 3D models acquired, a path may then be plotted to avoid obstacles that were found. Their article describes that small errors in estimation of helicopter pose can drastically affect the resulting 3D model. A problem that plagues a laser range finder is its weight and power consumption. With a UAV, payload weight comes at a premium cost; therefore every effort is made to minimize it.

Machine vision can generate 3D world coordinates of a surface from two 2D stereo images by triangulating matching features. It has been shown that accurate representations of 3D objects can be generated by stereo systems, which enable terrain navigation in mobile robots [8, 9, 10]. These methods demonstrate the ability to use stereo vision to detect safe spots for traveling. Xiong presented work for computing surface plots from descent images during a lunar landing [11]. This work uses a single camera to gather multiple images taken during descent. A stereo camera system is ideal for producing 3D data because they are lightweight, and the power consumption is low when compared to laser range finders. A stereo vision system's drawback is in the software processing required to produce the 3D information. However, with hardware implementations becoming available, vision algorithms can provide sensor data at acceptable rates.

1.4.2 Position Estimation Methods

Vehicle state estimation with vision augmentation is another topic that has been investigated extensively. Papers such as [12, 13] describe how using vision-based motion estimation, combined with inertial measurements, produces a more robust and less noisy estimate of position. Such research is used for lunar descents where GPS is not available. A single camera is used (along with a laser range finder to remove length ambiguities) to track multiple features from image to image during descent. Using the changes in feature locations, the algorithm is able to estimate the rigid body motion. These algorithms are complex and require a large amount of software processing which is not available on a small aerial vehicle.

In [14, 15], Amidi demonstrates a purely vision-based solution for position estimation. Amidi uses a helicopter platform and two cameras to track features from frame to frame. The first camera identifies a good feature to track from the center of the image, while the second camera looks for this same feature. Knowing the distance between the two cameras, and their respective camera models, the distance from the helicopter to the tracked feature can be estimated. Once the distance is known and a location is found for the feature in the camera's image, a vector can be made from the helicopter to the feature. This vector is then rotated using the helicopter's measured attitudes to place the vector into world coordinates. Using this newly found vector, and a vector from the previous location of the vehicle, the algorithm can then triangulate the new position of the helicopter. Amidi shows that this method is robust enough to use in the helicopter's flight control, allowing for autonomous navigation without GPS. A problem with this method comes from the fact that distances measured from stereo cameras are imprecise, and the degree of inaccuracy grows with the distance squared from the viewed object [16]. Amidi also assumes that if a feature is lost another feature can be easily acquired. Since the UAV is landing at a precise location, the landing site must remain tracked during the entire descent.

This thesis will use stereo vision to generate a surface plot of the landing area. The 3D surface plot will then be searched for a safe landing region. Once a safe location is chosen, monocular tracking will provide position data to the navigation controller during descent.

1.5 Contributions

This thesis contributes to the broad base of current autonomous landing methodologies involving UAVs while also accomplishing PALACE mission milestones.

The thesis demonstrates the following technology that was developed, tested, and merged to solve the overall problem:

- Monocular position generation for use in absence of a GPS signal,
- Stereo range mapping to gather a 3D database of a specified landing area,
- Safe Landing Area Determination to search over the 3D database for a safe landing spot.

Another contribution is evidence that stereo range mapping and monocular tracking can be simulated using 3D scene generators, thus allowing rapid testing and verification of algorithms without the initial need for complicated hardware testing.

1.6 Outline

This thesis proceeds as follows. Chapter 2 introduces the simulation and hardware setup that was used to test and validate the algorithms used in the thesis. Chapter 3 describes the monocular position estimator that was developed to help control the UAV in its landing phase. All equations, simulations, and hardware results pertaining to the estimator are presented in this chapter. Chapter 4 documents the stereo machine vision used to select a safe landing site. Camera calibration, stereo ranging, and safe landing area determination are detailed. An overall simulation of the PALACE mission and a discussion of the results are shown in Chapter 5. Chapter 6 presents conclusions and potential directions for future research.

Chapter 2

Simulation Environment and Experimental Apparatus

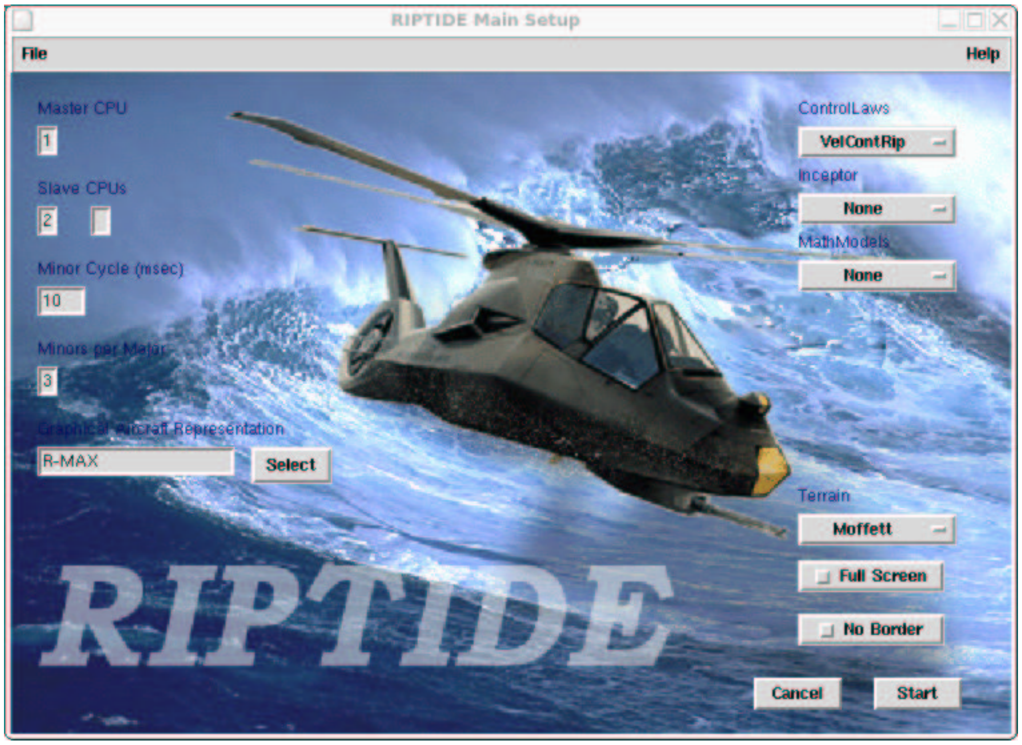
This chapter contains information on the setup of the simulation environment and the hardware equipment used to test the landing algorithms.

2.1 RIPTIDE

Testing vision algorithms that require camera position and orientation in hardware costs a great deal of money and time. Some researchers have invested vast amounts of money and time into large machine gantries that move in a 3-axis coordinate system to simulate the movement of a camera attached to a UAV. Although this is ideal because position and rotation sensors from the gantry contain accurate readings, the costs and setup time is extensive.

Another option is to first simulate what a camera would view as it is moving through space. Once software simulation has verified the landing algorithms, hardware testing can commence and validate the approach. This reduces uncertainty because the algorithms have already proven themselves in a simulated environment, saving research dollars and time.

A model of a UAV can be built into a software simulation along with noise parameters, wind turbulence, and aerodynamic constraints. To aid in viewing the output of a UAV simulation, 3D display programs have been developed. 3D display simulators have become extremely popular over the past decade with advancements in hardware video cards that allow graphics engines to portray life-like images. Photographs can be taken of real objects and applied to 3D models, allowing for the creation of databases to be used in building up a virtual world that can be viewed



Master CPU: 1
Slave CPUs: 2
Minor Cycle (msec): 10
Minors per Major: 3
Graphical Aircraft Representation: R-MAX
ControlLaws: VelContRip
Inceptor: None
MathModels: None
Terrain: Moffett

Figure 2.1: RIPTIDE start up screen

from any angle and position. This virtual world can then be used to quickly test vision and controller algorithms.

A requirement of the PALACE mission is to develop a complete landing simulation using the **R**eal-Time **I**nteractive **P**rototype **T**echnology **I**ntegration / **D**evelopment **E**nvironment (RIPTIDE) simulator, along with a Yamaha R-50 UAV software simulation math model developed in Matlab and C code. The RIPTIDE simulator is a workstation-based simulation environment that makes real-time, visual, full-flight-envelope, pilot or operator-in-the-loop simulation readily available throughout the design cycle [17]. The flexibility of the RIPTIDE simulator allows for simulated sensors to be easily added to the 3D environment, allowing for software testing before hardware integration.

To support this thesis, a pair of stereo cameras were simulated in RIPTIDE. Vision algorithms typically require images that have a large histogram of colors so that features may be easily distinguished. A majority of the surfaces in the RIPTIDE database were created using a small color palette which caused some initial vision problems. The solution to this problem was to texturize the required surfaces with photo images that have a high contrast of colors.

The simulated cameras allow streaming images to be sent out from RIPTIDE's drawing windows which are then received and used by the vision algorithms. The emulated cameras are attached to the simulated helicopter and separated by a fixed distance. A number of adjustable parameters for the virtual cameras allow for testing different setups. These parameters include

- Horizontal field of view,
- Camera separation distance,
- Camera pointing angles referenced from the UAV, [1]
- The rate of capture used by the 3D display for retrieving camera images.

Another sensor that was built into RIPTIDE for use in this thesis was a laser range finder. This sensor measured the distance from the focal point of the left stereo camera to the center of the image seen through the camera. The use of the laser range finder is described in Chapter 3.

Figure 2.2 shows a screen shot of RIPTIDE in action. RIPTIDE is made up of a number of different views which are selectable through the graphical user interface shown in the top left corner. The left camera view can be seen in the main top right section. This camera is looking down onto a 3D scene at a 45 degree down angle from the UAV's body axis. This scene shows a number of obstacles represented by tables and cones. These obstacles were used to help validate safe landing site selection algorithms and general-purpose tracking algorithms.

[1] The cameras are assumed to be attached to a tilting mechanism which allows the cameras to be pitched at any angle from pointing directly down from the UAV, to pointing out the nose towards the horizon. This is an important parameter since it allows for debugging vision algorithms which rely on camera angles.

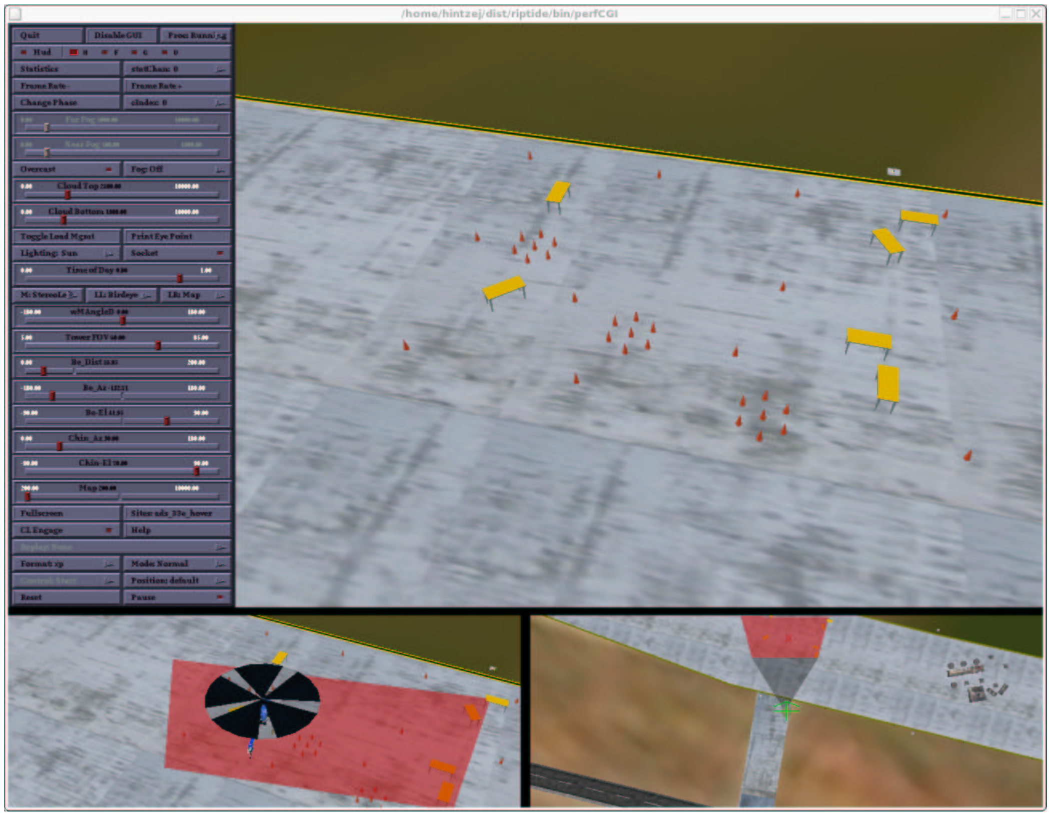


Figure 2.2: RIPTIDE vision system

To help visualize the camera's viewing area, a viewing frustum was built into RIPTIDE. The lower left view of Figure 2.2 shows a birds-eye view of the UAV along with a gray transparent frustum and a red intersecting plane. The same view is shown again, but from above the UAV, in the bottom right hand corner of the figure.

Matlab provides the simulation calculations for the UAV's movement and pose. To aid Matlab in producing correct simulation values, a mathematical model of a Yamaha R-50 was used. The R-50 UAV model was identified using system identification methods described in [18]. Wrapped around the UAV model is a series of control loops that are used to control the UAV's velocities and position.

2.2 Hardware Setup

Once software testing verifies the feasibility of the landing algorithms, then hardware testing can begin. The hardware apparatus that was used for testing was developed by a group located at NASA Ames Research Center called the Autonomous Rotorcraft Project. The project brings together several technologies to address NASA and US Army autonomous vehicle needs. A report of the hardware and software development is provided in [19]. This thesis worked primarily with the vision sensors which include a pair of Dragonfly cameras developed by Point Grey, and a compact PCI computer for software implementation of the vision algorithms.

The Dragonfly camera is an IEEE-1394 camera capable of grabbing and transmitting 640 by 480 pixel images over a firewire link at 30 frames per second. Traditional camera systems require the image to be represented by an analog video signal so that it can be received and reconstructed by a video frame grabber card. An analog signal can pick up noise while being transmitted, resulting in image imperfections and affecting the overall image quality. The Dragonfly allows for the original image that was captured by the camera to be sent to the vision computer and be bitwise exact for processing.

Since the Dragonfly cameras are attached to the UAV, quick testing of vision algorithms can be a laborious process. To alleviate this problem, a pair of iBOT web cameras were purchased. The web cameras provide an easy way to test the effects of turbulence on the vision tracking algorithms by manually shaking them.

The compact PCI computer runs Linux version 2.4.22 with Redhat 7.3. The latest versions of software libraries dc1394 and raw1394 were used to capture images from the firewire cameras.

Chapter 3

Monocular Position Estimation

This chapter contains the implementation details of the monocular position estimator. All mathematical equations, software and hardware implementations, and experimentation results are presented here.

3.1 Feature Tracking

Two different methods were discussed in Section 1.4.2 for estimating the vehicle's position using machine vision. Both require feature tracking. Feature tracking algorithms are designed to locate matching features between two images. A feature could be classified as a corner, edge, or a pixel region with a large histogram of different colors. Identification of corresponding points in two different images is the most important task for monocular position estimation algorithms. This thesis uses a feature tracker to reference the vehicle's position to a known location.

A feature tracker starts by saving a template of a selected image feature in memory. When using a camera, the template would consist of a pixel region, typically square, called the kernel that is then convolved across a windowed region in another image frame. At each convolution, a score is recorded for how well the template matched the portion of the window. After all possible convolutions have been evaluated, the region with the best matching score is the newly-found feature. The previous template is then updated with the new matching region, and stored in memory for the next search pass in the next image.

To write a highly-optimized feature tracker with sub-pixel accuracy is a complex problem, and has been the subject of past doctoral dissertations. To support this thesis, we were able to obtain code provided by Jet Propulsion Laboratories' Machine Vision Group. The code provides two main functions: one for selecting the best trackable features in a region, and another function for finding the sub-pixel correspondence between two image regions.

The first function requires as input

1) an 8-bit gray-scale image,

2) the dimensions of the image in pixels,

3) the number of best trackable features it should find.

The function returns a list of locations, in x and y pixel coordinates, to be used in the feature tracking function.

The feature tracking function takes as input the feature template, along with a search window to compare the template against. It returns the coordinates of the best match in x and y pixels and correlation coherence value ranging from 0 to 1.0, with 1.0 representing a perfect match.

To test the accuracy of the feature tracking code, an experiment was conducted. The experiment consisted of recording video from a Yamaha R-MAX as it made repeated landing approaches. A cluttered environment was created on a runway to simulate urban environment obstacles, allowing only a single feasible landing position. Turbulence was induced into the helicopter by having the RC pilot produce noisy control stick inputs. After gathering the video, post-processing commenced using the provided feature tracker. Figure 3.1 shows three sample images of the video footage taken during a landing approach. The black "X" indicates the feature that was being tracked in that frame.

Repeated attempts showed that the feature tracker successfully identified and tracked features of high contrast during the landing approach. It was also found that the feature tracker did a satisfactory job of tracking features that had low color variance.

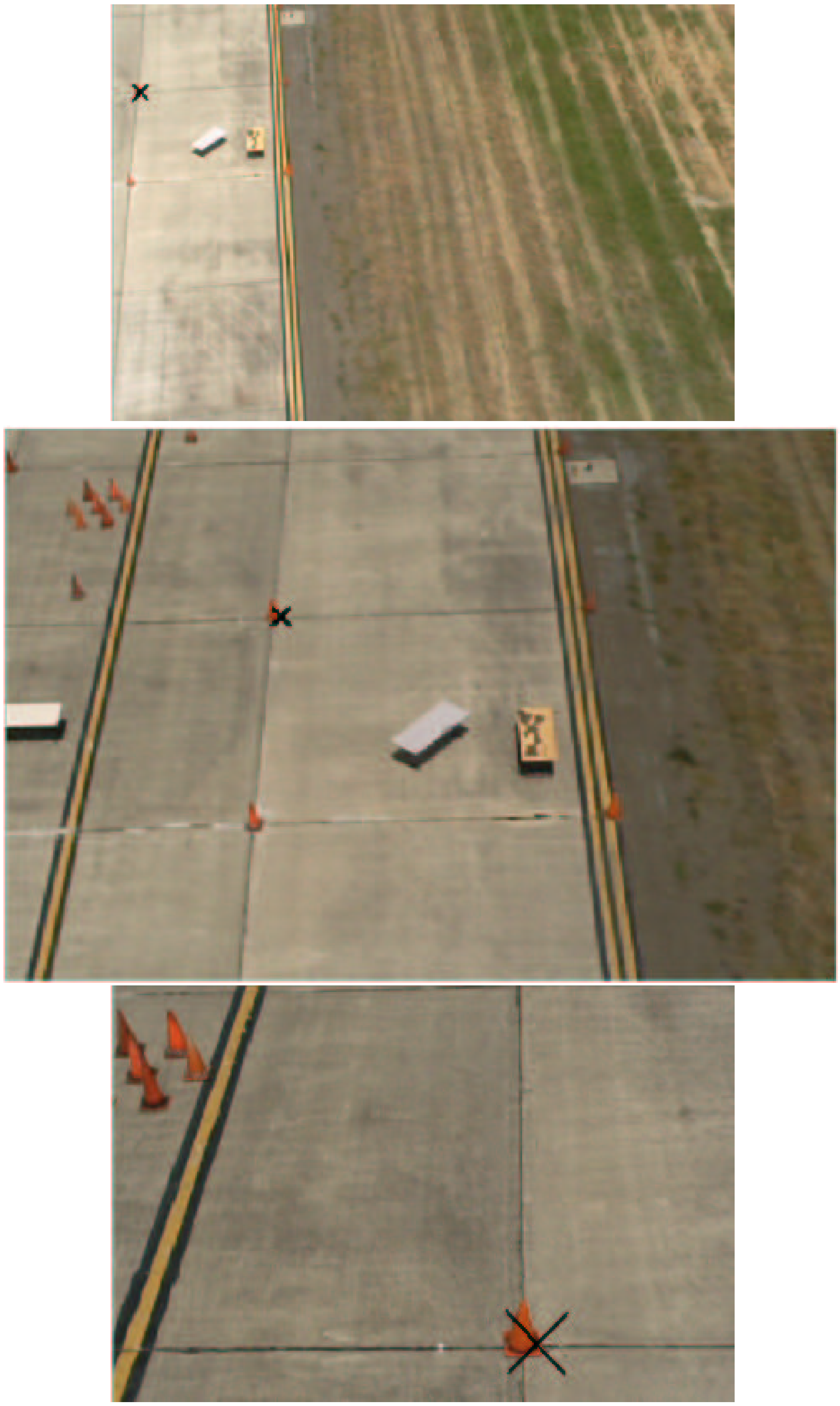


Figure 3.1: Consecutive descent images running feature tracking

3.2 Position Estimation

Position estimation is needed in the absence of GPS in order for the control laws to accurately navigate the UAV. A feature tracker only provides the vehicle with the pixel location of the landing site. Therefore, there needs to be a way to use the pixel location along with other sensor information to generate a pseudo position estimate. The estimate is a pseudo estimate because the vehicle does not know the exact location of the landing site that it uses to reference its position. Rather the UAV has an idea of the location through stereo ranging and the known GPS location of the vehicle before it started the landing phase.

An example of transforming tracking positions and vehicle orientations into a position estimate is given in [14]. A brief discussion on this method was presented in 1.4.2. Amidi's work generated vehicle positions by tracking features while the UAV was moving. If the features were occluded, or left the camera's view, different features would need to be found. This method does not work for landing because the UAV requires constant monitoring of the landing site features. Also, if the landing algorithms estimate the vehicle's position and the control objective is to minimize the distance between the vehicle and the landing site, the camera will eventually lose the feature because of the camera offset from the UAV center.

Thus, the control objective is to keep the landing feature in the camera's view, at all times, while flying toward the landing site. Ideally the landing feature would be located in the center of the camera's image to reduce the chance of it leaving the view. New algorithms have been devised in this thesis to resolve these problems.

3.2.1 Definition of Coordinate Frames

This section defines the coordinate frames that are used throughout the rest of this thesis. Each coordinate frame has an associated transformation allowing multiple representations of a point in space. To estimate UAV position, there needs to be defined an image, camera, helicopter, and ground coordinate system.

The inertial ground coordinate system is aligned with the earth's magnetic coordinate system as shown in Figure 3.2. The helicopter coordinate system's origin

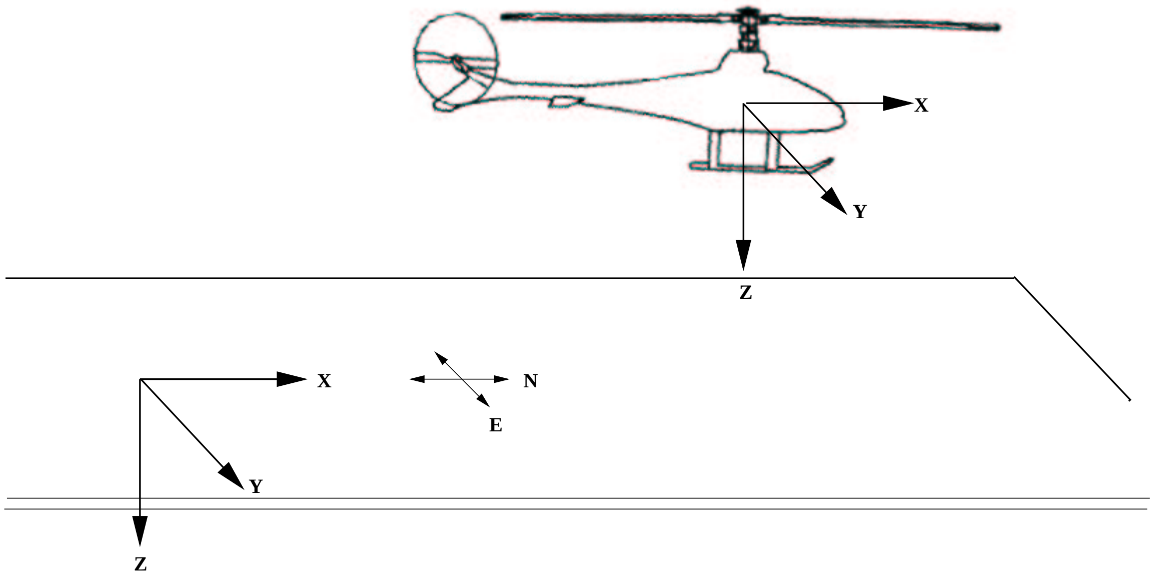


Figure 3.2: Inertial and helicopter coordinate systems

is located at the center of mass of the UAV, with positive attitude angles adhering to a right-handed coordinate system. The camera frame has its origin located at the camera's focal point, and is partially aligned with the helicopter's coordinate system as shown in Figure 3.3.

The cameras are fixed to the UAV on a tilt-able platform creating a pitch offset angle Θ_C between the two axis systems. Figure 3.4 displays the image coordinate system defining positive X and Y directions and the origin of the image.

3.2.2 Notation

Two different types of points that are of interest: a feature point, and a center-of-image point, both illustrated in Figure 3.5. A feature point is a position in the landing site that the UAV is tracking. The center-of-image point is the intersection of a vector originating from the camera's focal point, through the center of the image plane, and intersecting with the ground. To measure the length of this vector, a laser

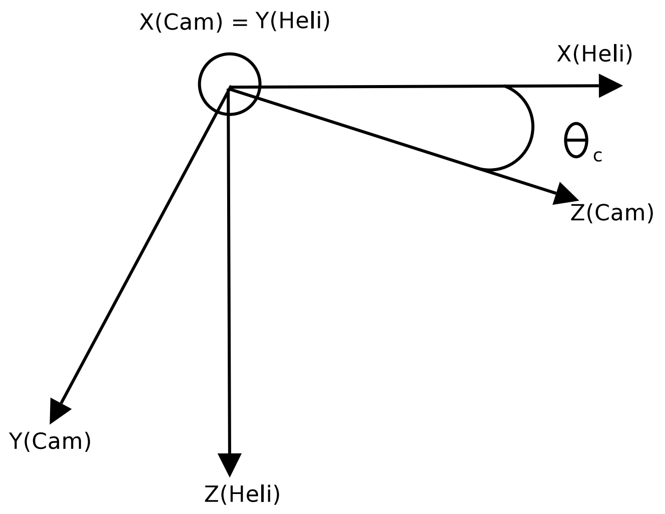


Figure 3.3: Coordinate system alignment for the helicopter and camera

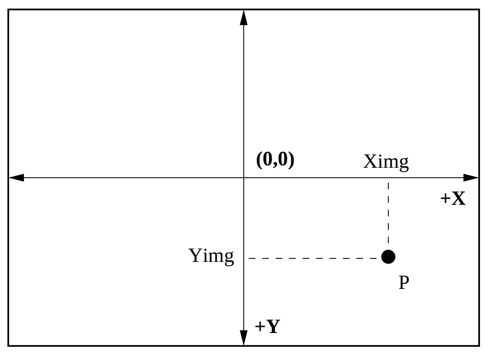


Figure 3.4: Camera image plane coordinate system

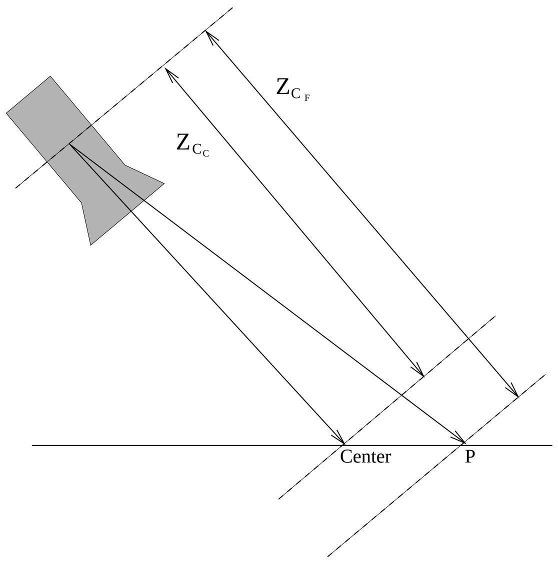


Figure 3.5: Points of interest and their distances

range finder is mounted on the UAV. The range finder is aligned with the camera so that it returns the approximate distance to the center-of-image point.

The variable notation is defined as follows. Let $P_{I_F} = [X_{I_F}, Y_{I_F}, Z_{I_F}]^\top$ be a feature point tracked in the inertial coordinate system. The first subscript identifies which coordinate system the vector belongs to. Subscripts I, H, C represent the inertial, helicopter, and camera coordinate systems respectively. The second subscript classifies whether the point is a feature (marked with an F), or the center-of-image point (marked with a C).

3.2.3 Derivation

The position estimator operates by tracking a feature point on the landing site, and then updating the vehicle's position from the movement of the feature in the subsequent camera images. It is assumed that the UAV has already determined the feature point's inertial location, by using stereo vision. This is described in Chapter 4.

The position estimation algorithm generates a vector in the inertial coordinate system, starting at the camera's focal point and ending at the feature point. This vector is then translated to the known feature location to produce the vehicle's estimated position. A problem arises from the fact that the vehicle does not know the distance from the camera to the feature, but rather it knows the distance to the camera's center of image point. Therefore, there needs to be a way to determine this length from known information. The following assumptions are used to solve this problem.

1) The feature point is assumed to be at the same height above ground as the center-of-image point. This assumption allows the distance to the feature point to be calculated using the distance measured from the laser range finder.

2) The only rotation of the camera relative to the UAV is in the pitch angle. This allows alignment of the helicopter and camera axes in at least two directions, eliminating the need to rotate helicopter attitudes other than adding a pitch offset angle.

3) Helicopter position will not be calculated about its center of mass, but will be calculated about the camera's focal point. This simplifies the equations since the camera is fixed relative to the helicopter's center of mass. This assumption also avoids the control problem of calculating trajectories to move the center of mass (instead of the camera) to the landing site.

The transformation from the image coordinate system to the camera coordinate system is calculated by similar triangles, using the pinhole camera model shown in Figure 3.6. To transform a point, the camera's focal length must be known. The focal length for a perfect (no distortion) camera given the image plane width, and the field of view (FoV) is

$$Focal = \frac{\frac{W}{2}}{\arctan \frac{FoV}{2}}. \tag{3.1}$$

If the cameras are not assumed to be perfect then camera calibration is required to determine the focal length.

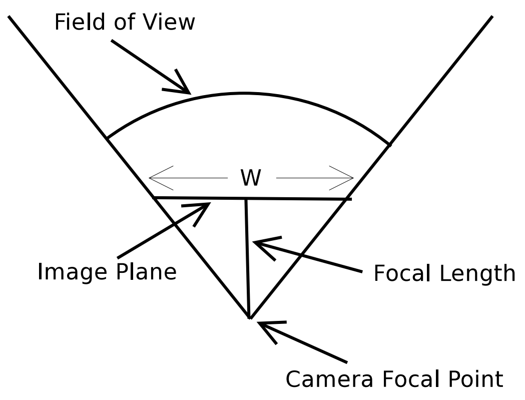


Figure 3.6: Pinhole camera model from a top-down view

To transform a point from a pixel location to the camera coordinate system the following equations are used:

$$\begin{aligned} X_{C_F} &= \frac{X_{img}}{Focal} Z_{C_F}, \\ Y_{C_F} &= \frac{Y_{img}}{Focal} Z_{C_F}. \end{aligned} \tag{3.2}$$

Notice that this transformation assumes the distance is known from the camera's focal point to the point of interest.

Figure 3.4 shows the setup for rotating a point from the camera coordinate system to the helicopter's body axis. Given the feature point $P_{C_F} = [X_{C_F}, Y_{C_F}, Z_{C_F}]^\top$ and camera offset angle Θ_C

$$P_{H_F} = \begin{bmatrix} X_{H_F} \\ Y_{H_F} \\ Z_{H_F} \end{bmatrix} = \begin{bmatrix} 0 & \sin\Theta_C & \cos\Theta_C \\ 1 & 0 & 0 \\ 0 & \cos\Theta_C & -\sin\Theta_C \end{bmatrix} \begin{bmatrix} X_{C_F} \\ Y_{C_F} \\ Z_{C_F} \end{bmatrix}, \tag{3.3}$$

where P_{H_F} is the feature point in the helicopter's body axis.

A standard (3x3) Euler rotation matrix [20] may be used to rotate the point P_{H_F} to the inertial coordinate system. The Euler rotation matrix takes the form

$$\Sigma(\Phi,\Theta,\Psi) = \begin{bmatrix} c\Theta c\Psi & s\Phi s\Theta c\Psi - c\Phi s\Psi & c\Phi s\Theta c\Psi + s\Phi s\Psi \\ c\Theta s\Psi & s\Phi s\Theta s\Psi + c\Phi c\Psi & c\Phi s\Theta s\Psi - s\Phi c\Psi \\ -s\Theta & s\Phi c\Theta & c\Phi c\Theta \end{bmatrix}, \tag{3.4}$$

where $c\Psi = \cos(\Psi)$ and $s\Psi = \sin(\Psi)$.

Therefore, the transformation from the helicopter body axis to inertial coordinate system is

$$P_{I_F} = \Sigma(\Phi,\Theta,\Psi) P_{H_F}. \tag{3.5}$$

The next step is to determine the distance from the camera's focal point to the feature location. Assumption 1, stated at the beginning of this section, allows us to conclude that the Z values of the center of image point and the feature point are equal in the inertial coordinate system, i.e.

$$Z_{I_F} = Z_{I_C}. \tag{3.6}$$

The laser range finder returns Z_{C_C}, which is the Z value of the center-of-image point in the camera coordinate system. This point is represented mathematically as $P_{C_C} = [0, 0, Z_{C_C}]^\top$. Transforming P_{C_C} into the inertial coordinate system yields

$$P_{I_C} = \begin{bmatrix} X_{I_C} \\ Y_{I_C} \\ Z_{I_C} \end{bmatrix} = \Sigma(\Phi,\Theta,\Psi) \begin{bmatrix} 0 & \sin\Theta_C & \cos\Theta_C \\ 1 & 0 & 0 \\ 0 & \cos\Theta_C & -\sin\Theta_C \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ Z_{C_C} \end{bmatrix}. \tag{3.7}$$

Solving for Z_{I_C} gives,

$$Z_{I_C} = Z_{I_F} = -\sin\Theta\cos\Theta_C Z_{C_C} - \cos\Phi\cos\Theta\sin\Theta_C Z_{C_C}. \tag{3.8}$$

Notice if all UAV attitudes equal zero, Equation 3.8 reduces to $Z_{I_C} = -sin\Theta_C Z_{C_C}$, which is the law-of-sines equation for a triangle. The reduced equation is negative because Θ_C is defined as a negative angle when rotated down from the helicopter body axis.

A similar method can be used to determine Z_{C_F}. Using Equation 3.2 and transforming into the helicopter coordinate system gives

$$P_{H_F} = \begin{bmatrix} X_{H_F} \\ Y_{H_F} \\ Z_{H_F} \end{bmatrix} = \begin{bmatrix} 0 & \sin\Theta_C & \cos\Theta_C \\ 1 & 0 & 0 \\ 0 & \cos\Theta_C & -\sin\Theta_C \end{bmatrix} \begin{bmatrix} \frac{X_{img}}{Focal} Z_{C_F} \\ \frac{Y_{img}}{Focal} Z_{C_F} \\ Z_{C_F} \end{bmatrix}, \tag{3.9}$$

or

$$\begin{aligned} X_{H_F} &= Z_{C_F} D_1, \\ Y_{H_F} &= Z_{C_F} D_2, \\ Z_{H_F} &= Z_{C_F} D_3, \end{aligned} \tag{3.10}$$

where

$$\begin{aligned} D_1 &= \cos\Theta_C + \tfrac{Y_{img}}{Focal} \sin\Theta_C, \\ D_2 &= \tfrac{X_{img}}{Focal}, \\ D_3 &= -\sin\Theta_C + \tfrac{Y_{img}}{Focal} \cos\Theta_C. \end{aligned}$$

Transforming the feature from the helicopter coordinate system to the inertial frame produces,

$$\begin{aligned} X_{I_F} &= Z_{C_F} F_1, \\ Y_{I_F} &= Z_{C_F} F_2, \\ Z_{I_F} &= Z_{C_F} F_3, \end{aligned} \tag{3.11}$$

where $F_3 = \Sigma(3,1)D_1 + \Sigma(3,2)D_2 + \Sigma(3,3)D_3$. Here $\Sigma(i,j)$ is the ith and jth index into the standard Euler matrix given in Equation 3.4.

Solving for Z_{C_F} gives the final equation

$$Z_{C_F} = \frac{Z_{I_C}}{-s(\Theta)\left(c(\Theta_C) + \frac{Y_{img}}{Focal} s(\Theta_C)\right) + c(\Theta)s(\Phi)\frac{X_{img}}{Focal} + c(\Theta)c(\Phi) * \Gamma}, \tag{3.12}$$

where $s(\Theta) = \sin\Theta$, $c(\Theta) = \cos\Theta$, and $\Gamma = \left(-\sin(\Theta_C) + \frac{Y_{img}}{Focal}\cos(\Theta_C)\right)$.

We can determine the vehicle's position using the following steps:

1) Select the initial landing site feature and track it during each image update.

2) Calculate Z_{I_C} using Equation 3.8.

3) Using the feature location, attitude angles, and Z_{I_C}, calculate Z_{C_F} as shown in Equation 3.12.

4) Calculate P_{C_F} with Equation 3.2.

5) Transform P_{C_F} to the helicopter coordinate system.

6) Transform P_{H_F} to the inertial coordinate system using the standard Euler rotation matrix.

7) Add P_{I_F} to the landing sites identified inertial position to determine the final vehicle position.

3.3 Monocular Experimentation Results

The algorithm in the previous section was implemented in ANSI C/C++ code and optimized for real-time use. Simulation and debugging was performed using the RIPTIDE environment. The simulated cameras in RIPTIDE were placed one meter apart in the Y axis of the helicopter. Each camera was tilted downward at a 45 degree angle and used a 45 degree field of view. The image width and height were set at 640 by 240 pixels. This size shares the same width as the hardware cameras but is only half the height, since RIPTIDE had problems producing images that large.

The experimentation setup called for the helicopter to hover 50 feet above ground level and positioned in such a way that a road cone was centered in the left cameras image. The objective of the experiment was to estimate the vehicle position while it flies to the road cone. Data recorded from the experiment included the actual vehicle position calculated by the math model, and the estimated vehicle position in the inertial frame. Figure 3.7 shows an overlapping plot of the X, Y, and altitude positions. As can be seen, the monocular estimate closely follows the actual

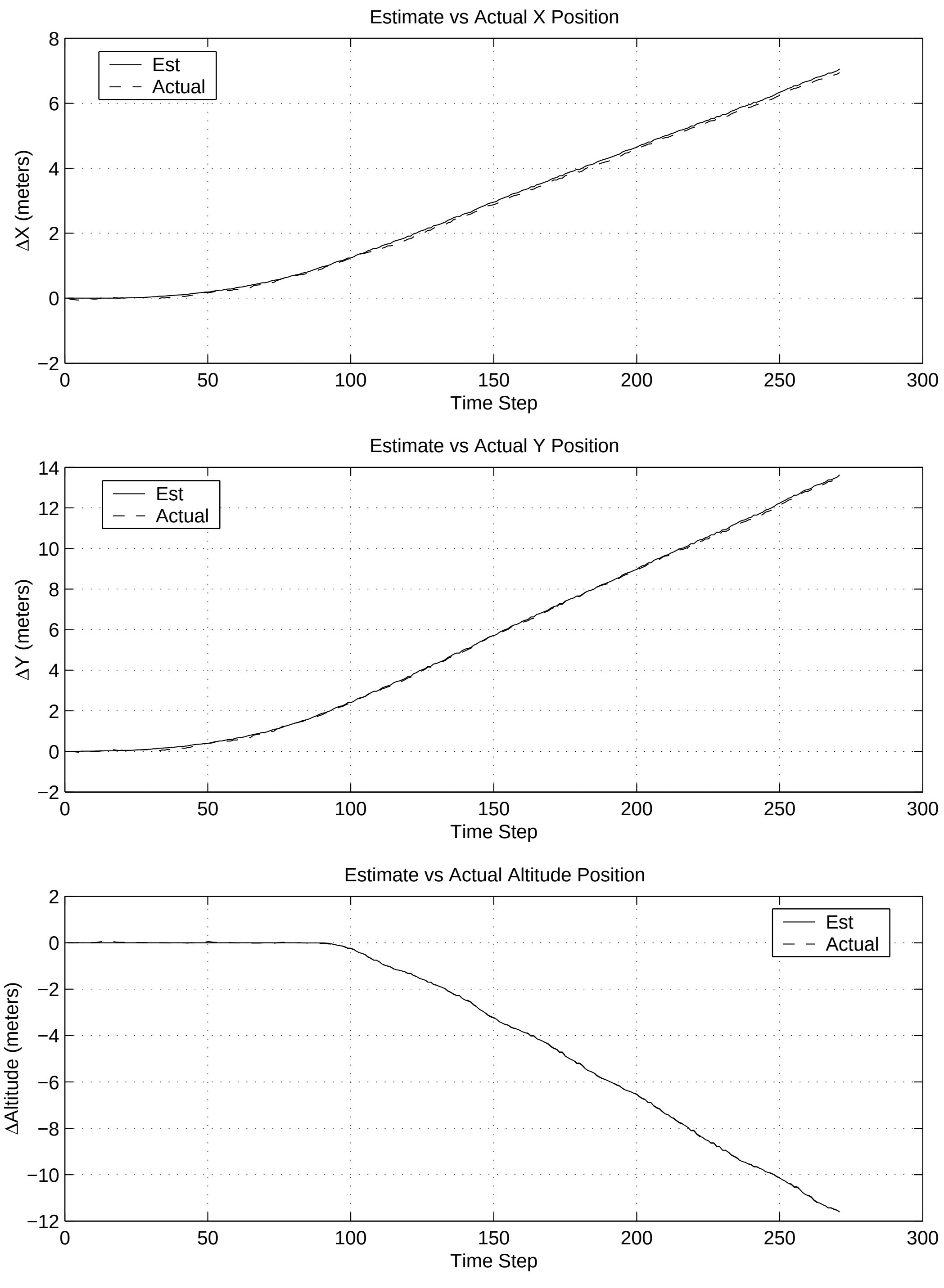


Figure 3.7: Estimated vs actual positions

values. The fact that the estimated position never deviates more than a quarter of a meter suggests that the monocular estimation could be used in place of GPS position coordinates during closed-loop flight.

Although the previous experiment took place in simulation, some hardware experimentation has also been conducted. The first hardware experiment tested the feature tracking algorithms using the iBot web cameras. This allowed the algorithms to run on real images that were captured and processed in real-time. The optimized feature tracking code ran at 30 frames per second, which is the frame rate for the iBot camera system. It successfully tracked features while the camera was moved by hand. Since the iBot camera's position and attitude cannot be determined, monocular position estimation was not tested.

The second hardware experiment took place on the Yamaha RMAX. The RMAX was fitted with two cameras that point downward at 35 degrees and are separated by 1.056 meters along the helicopter Y axis. All monocular position estimation code was transferred over to the RMAXs on-board vision computer. A controlled experiment was set up that positioned the RMAX over a runway, at NASA Ames Research Center, hovering autonomously at 150 ft above ground level. A glide-slope path was preplanned to allow the RMAX to travel from its first hovering point down toward the ground at the same angle that the stereo cameras pointed. This allowed objects in the center of the cameras image to remain in the image during descent. Since a laser range finder has not yet been integrated into the RMAXs sensor suite, a software simulated range finder was created.

The UAV autonomously traversed the glide slope numerous times, each time tracking a different feature and generating position estimates. Test results showed that the feature tracker successfully tracked the selected features in real-time. However, the tracking could be further improved by adjusting the template size and the search window size. Increasing the template and window size can improve the accuracy of the matching features, but doing so rapidly increases computer computation.

The results of the position estimation did not match with the actual positions. Upon post-processing it was discovered that the focal length used for the position estimation equations was incorrect.

Chapter 4

Stereo Machine Vision

This chapter contains the implementation details of the stereo vision algorithms. Described first is the camera calibration used to create accurate camera models that are required by the stereo machine vision algorithms. Following this are the methods used for choosing a safe landing location with their respective results.

4.1 Camera Calibration

Accurate camera calibration is one of the most important steps in using stereo vision algorithms [21]. Camera calibration is the method for determining camera parameters for use in sensor models. The sensor models are then used to extract information from corresponding stereo images.

There are multiple camera models that have been developed over the years and many different methods for finding values for the parameters of those models. Two camera models which have proved popular include the Tsai camera model, developed by Roger Tsai [22], and the CAHV or CAHVOR camera model, initially developed by Yakimovsky and Cunningham [23]. This section will compare the two camera models and explain the procedure for producing a model for use in the stereo vision algorithms.

4.1.1 Tsai Camera Model

The Tsai camera model is based on the pin hole camera model shown in Figure 3.6. This camera model was developed to estimate the projection of a 3D point in space onto a 2D image plane. There are 11 different parameters in the Tsai camera

Table 4.1: Tsai Model Intrinsic Parameters

f	camera focal length
κ	1st-order radial lens distortion
C_X and C_Y	x and y coordinates of the center of radial lens distortion and the exit point of the camera's Z axis in the camera coordinate system
S_X	the skew factor of a pixel

Table 4.2: Tsai Model Extrinsic Parameters

R_X, R_Y, R_Z	3 rotation angles for the transformation from inertial to camera coordinate system
T_X, T_Y, T_Z	3 translation values for the transformation from inertial to camera coordinate system

model which are divided into two subcategories: intrinsic (interior) and extrinsic (exterior). The intrinsic parameters of the Tsai camera model are shown in Table 4.1, along with the extrinsic parameters in Table 4.2.

Projecting a 3D homogeneous point onto a 2D image plane requires an de-warping procedure to remove radial distortion common in commercial cameras. This uses the intrinsic parameter κ along with a 3rd-order polynomial equation. Once the 3D point has had the effects of distortion removed from its position, a simple linear mapping can produce the projected 2D point. The linear mapping is

$$\begin{bmatrix} X_{img} \\ Y_{img} \\ 1 \end{bmatrix} = \begin{bmatrix} f & S_X & C_X \\ 0 & f & C_Y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ & \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{4.1}$$

Table 4.3: CAHVOR Model Parameters

C	Center of focus, the 3D coordinates of the pinhole focus point
A	The normal to the image plane which is not always forward
H	Horizontal information vector, not normalized
V	Vertical information vector, not normalized
O	Optical axis used for lens-distortion correction
R	Radial lens-distortion coefficients

where R is a 3×3 rotation matrix created by using rotation angles R_X, R_Y, R_Z, and $T = [T_X, T_Y, T_Z]^\top$ is a 3×1 translation vector.

4.1.2 CAHVOR Camera Model

The CAHV (no distortion information) or CAHVOR camera model is based on vectors that embed the camera model information. CAHVOR is an acronym where each letter of CAHVOR stands for a distinct vector used in the camera model. The different vectors and a short description are displayed in Table 4.3.

To transform a point $P = [X, Y, Z]^\top$ to 2D image coordinates the following equations are used:

$$X_{img} = \frac{(P - C) \cdot H}{P - C) \cdot A}, \tag{4.2}$$

$$Y_{img} = \frac{(P - C) \cdot V}{P - C) \cdot A}. \tag{4.3}$$

A detailed description of the CAHVOR camera models and a derivation of these equations can be found in [24].

This thesis acquired stereo vision algorithms that generate range maps for use in safe landing area detection algorithms from the Jet Propulsion Laboratory's Machine Vision Group. The stereo vision algorithms use the CAHVOR camera model exclusively, which is the primary reason the CAHVOR camera model was selected.

4.1.3 Camera Model Generation

A number of tools were received from JPL to facilitate the creation of a CAHVOR camera model. This software inputs images acquired from the cameras, and uses post-processing techniques to determine the camera parameters. A camera calibration target was built to aid this process. The steps for creating a CAHVOR camera model using stereo images are shown below:

1) Capture multiple stereo images of the calibration target shown in Figure 4.1. The images are to be taken with the calibration target placed at increasing distances away from the cameras, and oriented at different angles relative to the camera's line of vision.

2) Use centroid finding software that extracts the centers of the black calibration dots in pixel coordinates. This generates a file that contains all the coordinates for the corresponding dots.

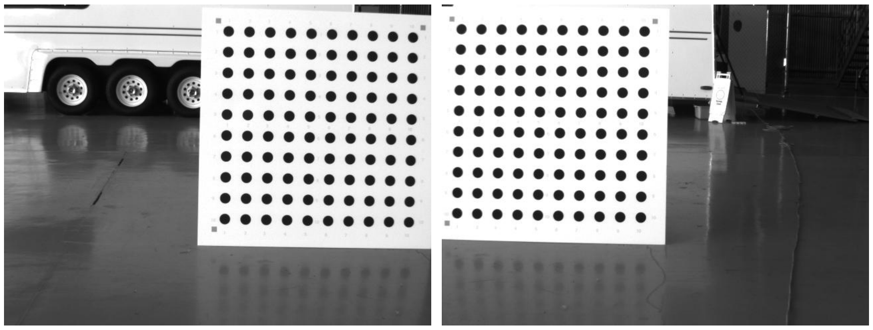
3) Generate a Tsai camera model using the extracted dot information and the spacing between dots, in meters.

4) Using the Tsai model, reverse project the 2D image coordinates to generate a list of 3D world coordinates that correspond to the placement of the actual dots in the inertial coordinate system.

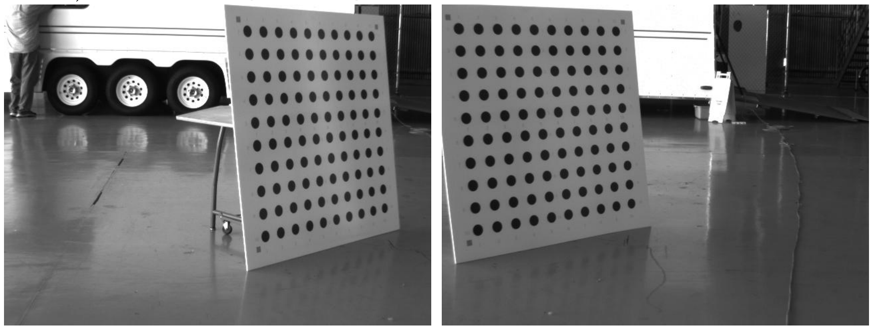
5) The 3D world coordinates and 2D image coordinates are used to identify the CAHVOR camera parameters.

These steps were followed to produce the camera models on the Yamaha RMAX. The tools produced a left and right CAHVOR camera model that may be used in the stereo vision software, and they also identified the individual camera focal lengths for use in monocular position estimation.

Although the tools were useful in identifying hardware camera parameters, the simulated cameras, used in the RIPTIDE environment, could not be calibrated by following the same method. The solution to finding the simulated camera CAHVOR



A) Left and right images acquired at a medium distance



B) Left and right images of the target placed at an offset angle

Figure 4.1: Calibration target images from stereo cameras.

model was quite simple. Because RIPTIDE simulates ideal cameras that are aligned vertically without orientation differences between the two, use of these two facts allowed the CAHVOR parameters to be calculated using the derived camera model equations.

4.2 Stereo Range Mapping

Stereo range mapping is the process of gathering two separate images of the same scene from different viewpoints, and then extracting 3D information from those images. Extracting the 3D information requires the two images to be modified using the known camera models. The reason for modifying the images is to assure that distinguishable features lie on the same horizontal pixel lines, making feature matching from the first image to the second image a 1D search problem. This step is called rectification.

After rectification, the stereo algorithms pass through the images and find the matching features. Since the images are at different viewpoints and they are already rectified, the only difference in feature locations are their horizontal positions. This difference is called a disparity. The stereo algorithms will pass through and create a disparity map that encodes the differences of the matched features. If a feature is not found, possibly because of occlusion, there will be a gap in the image. Smoothing techniques have been developed to remove these gaps.

Once a disparity map has been generated, the stereo algorithms use the camera model and a disparity map, to generate a range map. The range map holds the X, Y, and Z locations of all the matched features.

Although these steps appear straight forward, there are many problems that can arise when computing range maps. One problem occurs when one camera detects more light in the scene due to reflections. This can cause image saturation making similar features look completely different. Another problem comes from the fact that obstacles can occlude one camera's view. A detailed discussion on error analysis of stereo vision is presented in [25].

The algorithms provided by JPL follow the steps below.

1) **Capture the stereo images.** In the hardware setup, this involves sending a message to the firewire cameras requesting a black and white image to be captured synchronously. In RIPTIDE this causes the 3D graphics display to render two different scenes from different positions and angles, and then transfer the images, in memory, to the stereo algorithms.

2) **Read in the camera models.** This also sets up the de-warping tables for removing lens distortions.

3) **Create image pyramids.** This step creates an image pyramid where the lowest level of the pyramid is the original image and each level of the pyramid reduces the image size by a factor of 2. Doing so removes high frequency noise from the image and decreases computation time because there are less features to match in a smaller image.

4) **Use Laplacian or bilateral filtering on the stereo images.** This step is needed to reduce the image differences due to brightness seen in one image and not the other. It also reduces the amount of noise.

5) **Compute the disparity map.** This step involves de-warping the filtered images, then rectifying them to remove camera misalignment. It then runs a feature correspondence algorithm on the image to generate a dense disparity map.

6) **Determine the range map.** The range map is generated using the disparity map and the camera model. This holds the 3D information that is used in the safe landing area determination algorithms.

These algorithms were tested on images gathered in both hardware and software. On one of the RMAX's autonomous flights, a series of stereo images was gathered. The stereo images were post processed to generate the 3D scene viewed from the sky. Figure 4.2 shows a left camera view with the superimposed range map

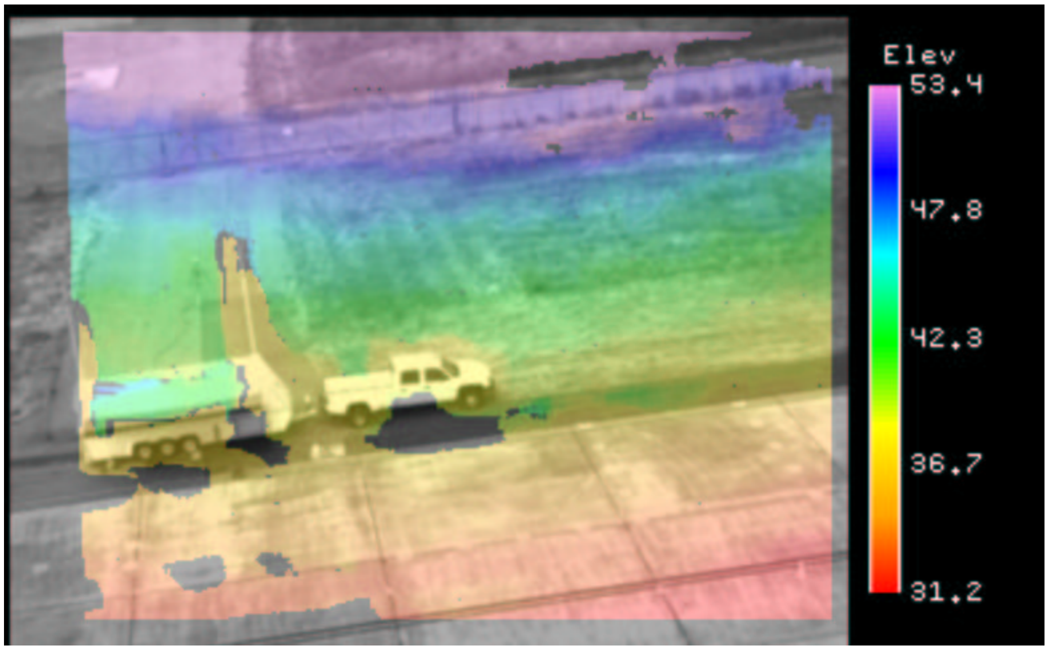


Figure 4.2: Range map information overlaid onto left camera image

information. Unfortunately, it is hard to test the accuracy of the hardware stereo vision in flight. A specialized obstacle would need to be created with known dimensions. However, in software simulation, obstacles can be constructed with exact dimensions. For this thesis, the verification of the stereo range mapping takes place in software with a known obstacle field.

Figure 4.3 shows the simulated obstacle field. It is created from a number of simple primitives: rectangles, squares, cylinders, and wedges used to test the accuracy of slope calculations. The obstacles were built into the RIPTIDE environment so that stereo ranging could take place. A picture of the obstacles in RIPTIDE is shown in Figure 4.4.

At first the stereo range mapping had troubles identifying the obstacles and making good feature matches. The source of the problem was that the obstacles were not textured with a sufficiently large gradient of pixels. To remedy this problem, a texture map of a city scape was applied to the primitives. With all the different build-

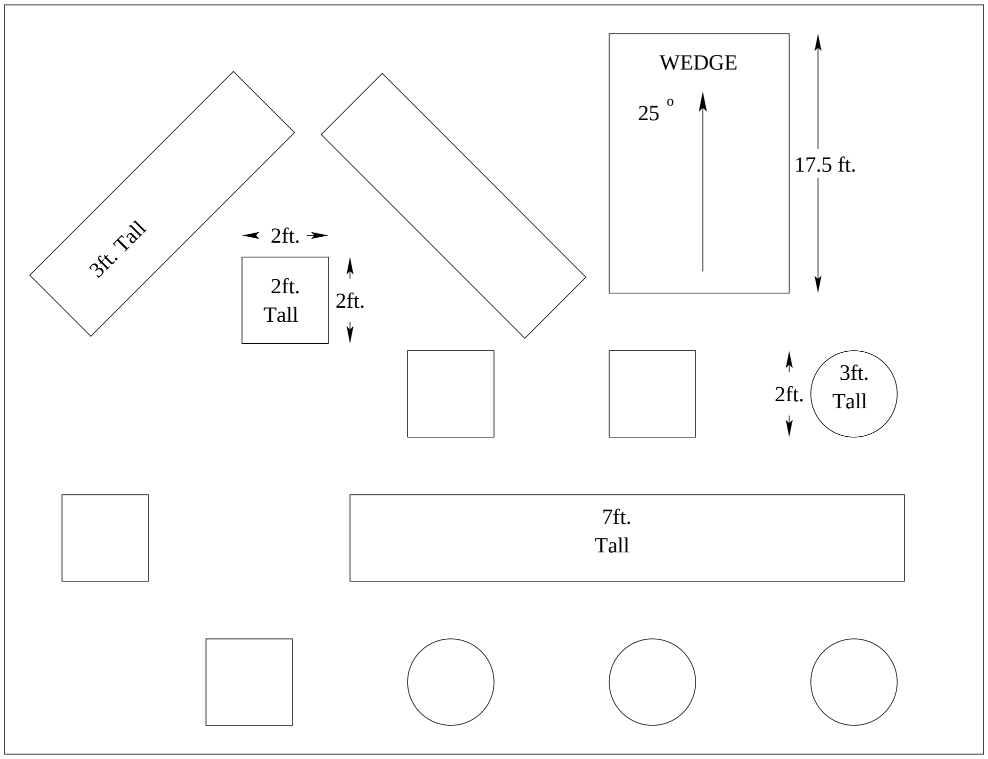


Figure 4.3: Obstacle field setup

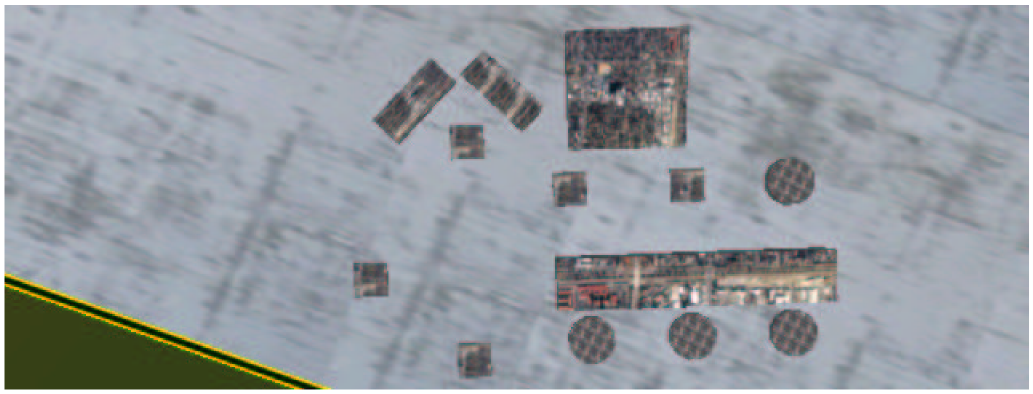


Figure 4.4: Obstacle field seen from inside RIPTIDE

ing shapes and colors, it made a random-like pattern with enough distinguishing figures for the matching algorithms to function properly.

The results of the range mapping were quite accurate. The range mapping algorithms were able to discern objects of six inches or higher at an altitude of 50 ft. The algorithms were also capable of calculating slopes within 5 degrees at the same altitude.

4.3 Safe Landing Area Determination

Once a 3D point cloud, shown in Figure 4.2, has been created over a potential landing site, Safe Landing Area Determination (SLAD) algorithms can process the 3D information. The SLAD algorithms look for a location that passes two key requirements: distance from obstacles, and landing area slope.

This thesis makes use of JPL algorithms for safe area selection, written by Andrew Johnson [26]. This section will briefly describe the different parts of Johnson's algorithm, including the inputs required to make it function properly, and the resulting outputs.

4.3.1 Elevation Map

The stereo vision algorithms, listed in Section 4.2, return a range map for all the corresponding features viewed from the camera. If the camera is looking at a flat surface, but viewing it from an angle, the surface will not look flat, but will have the same angle as the camera's incidence angle. Removing this artifact requires that the 3D range map be rotated by the camera Euler angles. Also, since a camera's view is a perspective and not an orthographic view, objects farther away will look smaller than objects nearby. To eliminate these factors, an elevation map must be created from the range data.

The elevation map generator operates by re-sampling the range data into an equally-spaced linear grid. The grid is a function of row and column indices, where $Z(r, c)$ represents the height at that index. When setting up the grid, the horizontal extent (width of the grid) must be determined along with the size of the individual grid cells. Johnson shows that if the field of view is known, along with the average range to the scene, the general formula for calculating horizontal extent (h) and grid size (s) is

$$h = 2R \tan(f/2) \tag{4.4}$$

$$s = h/n \tag{4.5}$$

where R is the average range to the scene, f is the camera focal length, and n is the number of samples across the scene.

This research tested these equations and concluded that the easiest way to determine grid size is to look at the minimum and maximum values of the range data, and use this information to set the bounds on the elevation map grid. All range data information would be accounted for, using this method. 100×100 grid cells were used to compromise between computation speed and accuracy. When a grid cell does not line up with any range map data, an interpolation process can determine the elevation at that cell. This interpolation process used all the surrounding grid cell heights to fill the void.

4.3.2 Hazard Maps

After the range data is re-sampled into a grid, the SLAD algorithms need to identify the hazardous areas. Once these areas have been found they are removed from the search space. This method creates a smoothness map which is then subtracted from the original elevation map to determine hazardous locations.

To generate the smoothness map, a 3D plane is fitted over the elevation data in the 3×3 grid surrounding each cell. Once a plane is fit to a grid, the normal to the plane can be compared to the UAV's landing vector. The angle between the landing vector and every grid cell can be calculated to create a new map $A_{map}(r, c)$. The algorithms search over this map to remove areas with dangerous landing slopes.

The next problem is to efficiently fit a plane to the elevation data. One possibility is to use Least Mean Squares (LMeanSQ) fitting. A problem occurs when using LMeanSQ if large outliers exist. The plane will be more shifted toward the outliers rather than the average of the inlier points. It is shown in [26] that Least Median Square (LMedSQ) fitting is more robust to outliers, but it does not fit the data as well as LMeanSQ. So the overall method is to first compute a LMedSQ plane to the data. If the distance away from the plane is above a certain threshold it is rejected. When all inlier points are found a LMeanSQ plane is fitted to the remaining data.

When the entire elevation map has been fitted by planes, the angle and roughness maps can be created. The $A_{map}(r,c)$ is computed by taking the dot product between the landing vector and the fitted plane's normal vector. The roughness map, $R_{map}(r,c)$, is the subtraction of the original elevation map and the smoothness map.

4.3.3 Cost Map

The cost map is the combined values of the angle and roughness map. If the landing grid angle is above the specified unsafe landing angle or

$$A_{map}(r,c) > A_{max} \tag{4.6}$$

then

$$C_{map}(r,c) = 1.0. \tag{4.7}$$

Likewise, if the roughness map height is above the maximum or

$$R_{map}(r,c) > R_{max} \tag{4.8}$$

then

$$C_{map}(r,c) = 1.0. \tag{4.9}$$

The remaining elements of the cost map are calculated by the normalized product of the angle and roughness maps using

$$C_{map}(r,c) = \frac{R_{map}(r,c)A_{map}(r,c)}{R_{max}A_{max}}. \tag{4.10}$$

Searching through the cost map for the minimum values yields the safest position to land. A smoothing filter can be applied to the cost map to assure that the chosen landing site is not near high-cost areas. The minimal value returned from the cost map is a 3D position (X, Y, Z). Since the monocular tracker needs to know where this landing site is on the image for tracking purposes, the point needs to be reverse projected. Equations 4.2 and 4.3 may be used for this procedure.

4.3.4 SLAD Inputs and Outputs

The SLAD algorithm has many inputs and outputs. Table 4.4 presents all the inputs and outputs along with a brief description of each. Tuning of the input parameters is an important task when trying to get SLAD to function properly. One problem that was noticed is that the input parameters need to be a function of the distance away from the landing site, especially when using a stereo vision system to acquire the range data. This comes from the problem associated with stereo ranging accuracy. At farther distances, the accuracy of the stereo vision system suffers, requiring the maximum deviation parameter to be increased. But if the maximum deviation parameter is increased, it is possible that the SLAD algorithms could choose an unsafe landing site.

One possible solution is to first find a safe landing site with an increased maximum deviation parameter. Then as the UAV is approaching the landing site, the SLAD algorithms can re-verify the feasibility of the landing location.

4.4 Stereo Experimentation Results

The entire stereo machine vision package was thoroughly tested in RIPTIDE. This involved using stereo vision for generating a range map, and SLAD for finding a safe landing site. The accuracy of the stereo range mapping was discussed in Section 4.2. This section will present the results of the SLAD algorithms which make use of the range map.

A simple obstacle field was set up in RIPTIDE including tables and cones scattered over a runway. One area of the cluttered environment was intentionally left void so that the UAV could land. The obstacle field was viewed by the UAV at different angles and positions, in each case running the stereo vision code. Figure 4.5 shows the left stereo camera's view of the landing site.

Table 4.4: Inputs and Outputs for the SLAD Algorithm

Inputs	
Grid Rows	The number of grid rows in the elevation map
Grid Columns	The number of grid columns in the elevation map
X_{min}, X_{max} Y_{min}, Y_{max}	The minimum and maximum values used for calculating the bounds of the elevation map
Border Size	The amount the border is interpolated allowing for a possible landing on the border of the image
Data Size	Stereo ranging holes are filled in by this size
Max Slope	The maximum allowable landing slope
Max Deviation	The maximum allowable deviation in the roughness map (used to classify hazardous areas)
Lander Base Size	The size of the landing vehicle
Max Rock Size	The sample spacing used for calculating the 3D plane fitting (increases computation speed)
Up Vector	The (X, Y, Z) vector corresponding to the vehicle's landing approach
Outputs	
Elevation Map	A matrix holding the height values of the newly-created elevation map
Roughness Map	A matrix containing the roughness values of the grid cells
Slope Map	A matrix containing the slope values for the grid cells
Cost Map	The cost calculated for all the grid cells
Safe Distance Map	A matrix containing the safe distance values from hazards
Safe Landing Map	The safe landing site displayed on the elevation map
Landing Site 3D Position	The (X, Y, Z) value of the landing site
Landing Site Pixel Position	The pixel corresponding to the (X, Y, Z) landing site in the image

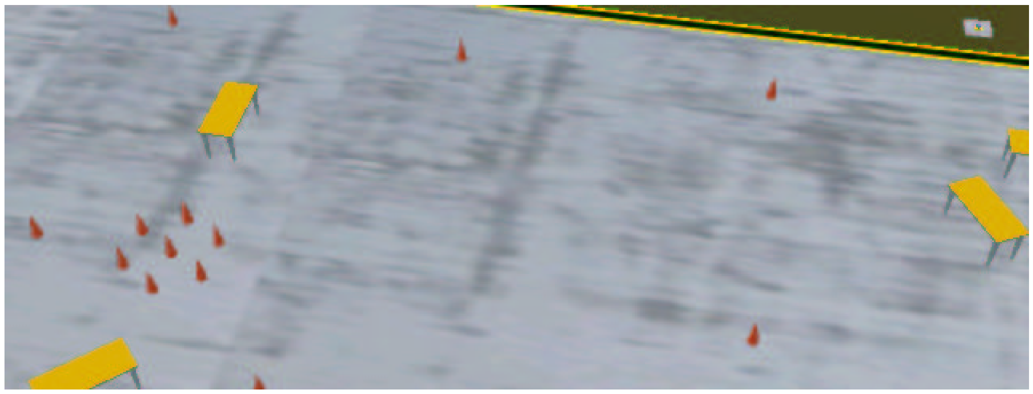


Figure 4.5: Cluttered landing area

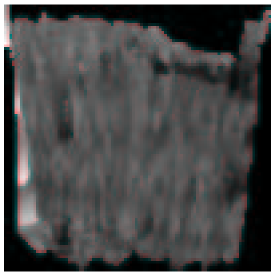


Figure 4.6: Elevation map

The output of the SLAD algorithms produced an elevation map. An image representation of the elevation map is shown in Figure 4.6. The elevation map is a top down view of the obstacle field. Regions with darker pixels correspond to lower elevations. The outputed maps from the SLAD algorithms are square. The reason for this is that we choose to calculate an elevation map which is 100×100. Doing so removes the perspective view inherent with cameras, allowing the SLAD algorithms to search over an equally spaced grided elevation map.

Figure 4.7 shows a slope map that was created by the SLAD algorithms. Each pixel records a slope value at that particular grid cell. When the pixel intensities change rapidly across scan lines it indicates that the slope has also changed rapidly. An example of a discontinuity in slope values is a table. When the SLAD algorithms are searching across the slope map, it will see the table as having too large of a slope

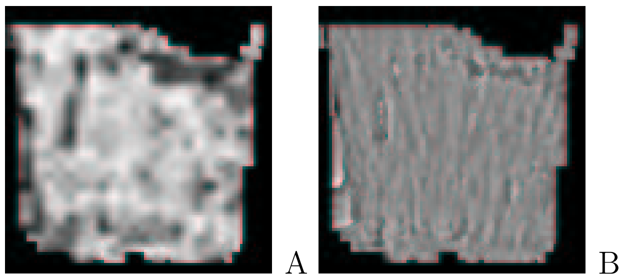


Figure 4.7: A) Slope map B) Roughness map

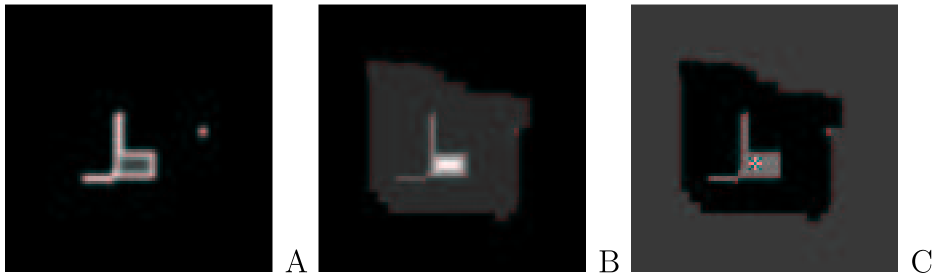


Figure 4.8: A) Cost map B) Safe distance from obstacles C) Safe landing site selection

and mark it as an unsafe place to land. It is important to note that at the center of the slope map the pixels do not change as rapidly. This indicates that the slope is relatively flat, which agrees with the obstacle field setup.

The roughness map is also displayed in Figure 4.7. The roughness map is the subtraction of the smooth planes from the original elevation. Areas that have darker pixels correspond to a rougher landing site. Obstacles can be more easily seen here.

After the slopes and obstacles have been filtered, the cost, safe distance, and landing site selection maps are created, as shown in Figure 4.8. In the cost map, only the white areas are acceptable landing positions. This takes into account the lander size as well as the slopes and obstacles. The safe distance map determines the location which is farthest away from any obstacles but still meets the slope criteria. Finding the farthest point away produces the safe landing location shown as a cross in Figure 4.8C.

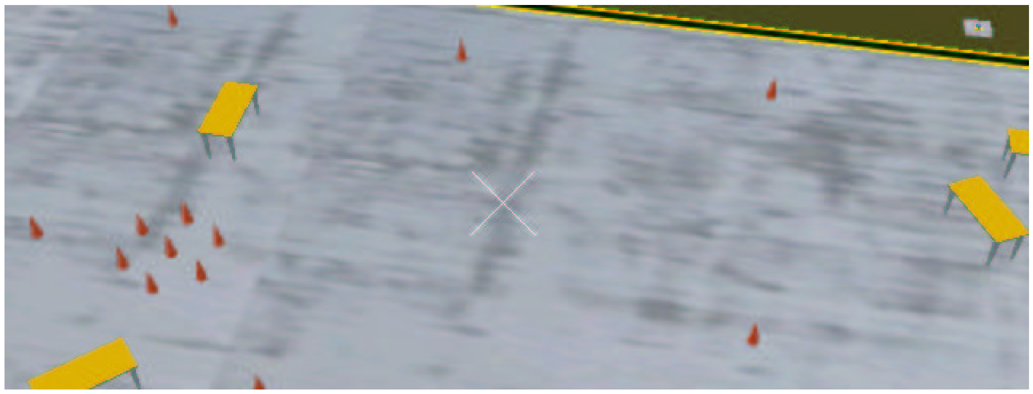


Figure 4.9: Safe landing position displayed on the cluttered landing area

Now that the vehicle has pinpointed a 3D location, this point is then projected onto the image. Figure 4.9 shows the chosen landing point overlaid onto the original clustered site image.

Repeated tests showed that the same landing site was selected in each case, regardless of the viewing position or angle. An additional series of tests were performed to identify the deficiencies of the vision algorithms. The key points that were examined were:

- Maximum distance away from the landing site,
- Minimum distance away from the landing site,
- Safe landing site selection accuracy, given a number of random obstacles.

The first two items were tested by creating a random cluttered environment using the road cones in RIPTIDE. The idea was that when the vehicle is positioned too close or too far away from the road cones, it will have a difficult time distinguishing the cones, thus making an incorrect landing site decision. This test was repeated fifty times at each different altitude. The findings are displayed in Table 4.5.

Higher altitudes were not tested because it was unclear whether the chosen landing point was correct or just a series of lucky guesses. Based on this test, it

Table 4.5: Safe Site Selection Range Accuracy

Altitude(ft)	**Accuracy(%)**
10	N\A
20	N\A
30	85
40	89
50	100
60	98
70	65

appears the optimal height above ground, given a 1 meter base line, is approximately 50 ft.

The final test determined how well the stereo vision algorithms reacted to different shapes, rather than just road cones, when selecting a safe landing site. This test used the primitives that were developed previously for use in checking the stereo ranging accuracy. A single shape was selected at random and positioned in the camera's field of view while the UAV was hovering 50 ft above ground level. The results showed that the vision algorithms were capable of correctly identifying the hazards and selecting safe landing sites. The vision algorithms would also choose to land on top of some of the obstacles if they were large enough and had a locally flat surface. An example of the flat surface was a rectangular box. The dimensions of the rectangle were larger than the RUAVs landing footprint. Therefore, the algorithms were correct in choosing that as a safe landing location because it passed the two SLAD landing requirements: distance away from hazarads and surface area slope.

Chapter 5

Mission Simulation

To execute an entire mission simulation, all of the different pieces of technology must work cooperatively with each other. This chapter discusses some of the difficulties that arose while building a complete mission demonstration. Final simulation results will also be discussed.

5.1 Integration Issues

Up to this point, all the previous sections discussed separate pieces of technology that have their own specific uses. When the technologies were combined, unforeseen problems appeared.

5.1.1 Monocular Tracking with Inner-loop Control

The monocular position estimator is meant to take the place of a GPS unit. In the current control laws, provided by the Autonomous Rotorcraft Project, a Kalman filter is used to combine sensor information from the inertial measurement unit and the GPS. It was decided that instead of replacing the GPSs signal in the Kalman filter with the vision estimate, the vehicle would run a separate Kalman filter. The reason for running two parallel filters is that the vision system could have different noise characteristics requiring unique Kalman filter gains. Also, when switching between the two Kalman filters, a discontinuity in the position could create a large velocity estimate.

When the control loops first started functioning using the monocular vision estimate, the UAV would crash almost instantly. Inspecting the Kalman filter velocities showed that they were extremely noisy and that the UAV's actuators did not have the dynamic range capable of sustaining flight. Figure 5.1 shows the instability that the Kalman filter would produce.

After inspecting the algorithms, it was found that the update rate from the vision system to the Kalman filter was not synchronized. The Kalman filter was expecting a 10 Hz update rate while the vision system would produce estimates as fast as possible, sometimes reaching up to 30 Hz. Correcting this problem cleaned up the velocities slightly, but it did not cause the UAV to behavior properly. The next step was to reduce the Kalman filter gains so that the high-frequency velocity estimate relied more on the on-board inertial measurement unit, and the low-frequency position estimate relied more on the vision. This drastically improved the velocity estimate as shown in Figure 5.2, but when attempting to fly the UAV in simulation the vehicle would still eventually crash.

The crashing problem was found when a number of different control loop gains were tested. It appeared that the integral gain was set at too high of a value for use with the monocular position estimate. Reasons for this could include the noisy nature of the vision estimate, or there may have been a bias introduced into the vision estimate. A negative effect of reducing the integral gains comes when the vehicle is flying in coordinated-turn mode. Since the integral gain is low, the vehicle will have a tendency to fly off course. This situation is not a concern for flying to a landing site since the path segments are typically straight.

5.1.2 Stereo Ranging for SLAD

One of the issues that was faced during the integration was to get a precise range map with correct units from the stereo vision. At first the range maps appeared to have a pitch rotation applied to them, even though the camera rotation had already been applied to correct the data. If an additional amount of rotation was applied it would fix the problem, but this was not an elegant solution. Also, the distance metrics

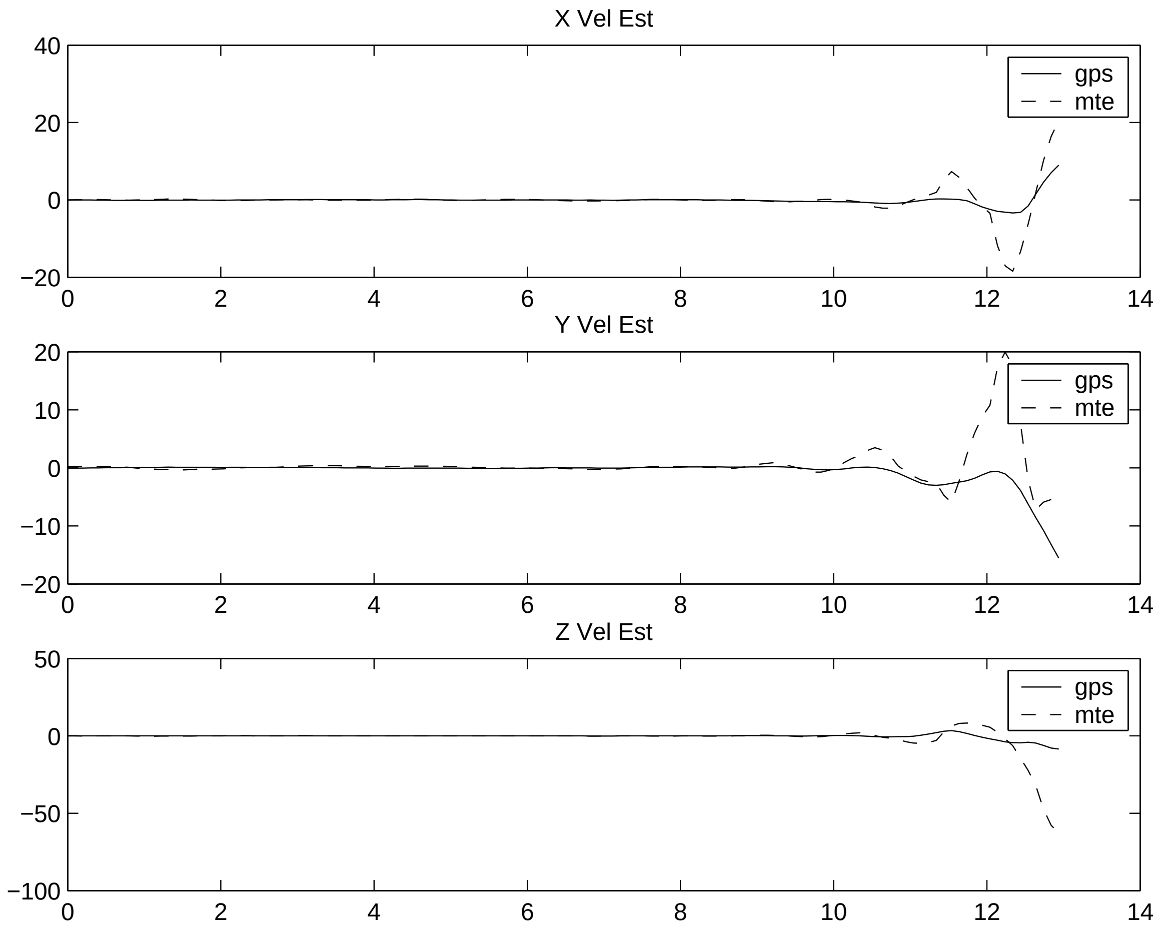


Figure 5.1: Unstable UAV velocity estimates

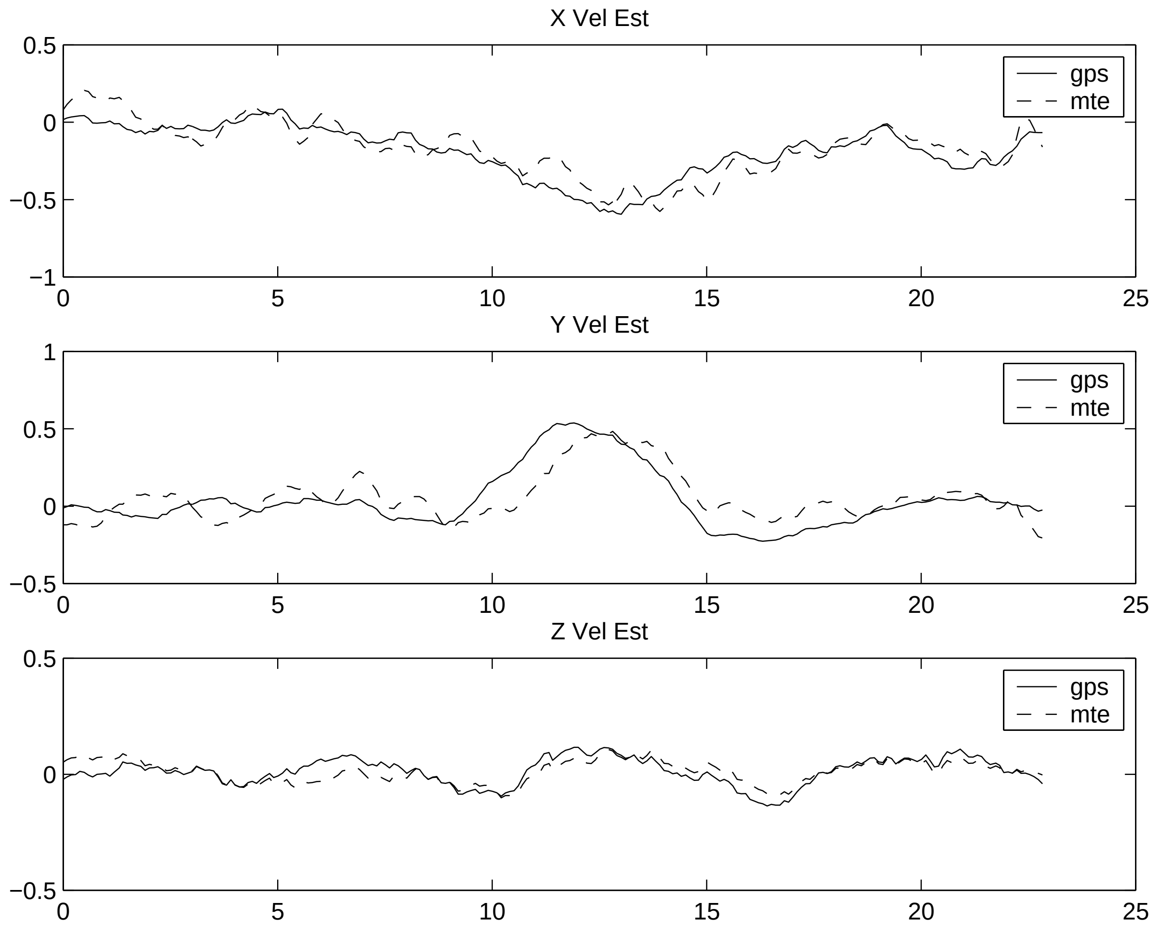


Figure 5.2: Improved UAV velocity estimates

appeared to be incorrect. If a measurement was taken from one side of the camera's image to the other side, the distances were off by an order of magnitude. This caused serious problems in the safe landing area determination algorithms because they were searching for a lander base size which was a great deal smaller than the resolution of the data.

The solution to the problem was to change the focal lengths of the cameras. It appears that the wrong calculation was used when the original focal lengths were determined. Fixing this problem resolved the issues with the stereo ranging.

5.2 Results

When the integration issues were resolved, a full working demonstration was presented. The UAV would first fly under GPS navigation to a final destination chosen by the user. Once it arrived, the vehicle would hover 50 feet above ground level, and use stereo range mapping and safe landing area determination to calculate the safest point to land. After the vehicle had its landing position locked in the feature tracker, the monocular position estimation would be enabled.

Because the velocity estimates, using machine vision, are not as precise as GPS velocity estimates, the vehicle appeared less stable. However, this behavior was not significant enough to cause it to lose sight of the landing point. Finally the vehicle descended to the landing site and touched down onto the ground. A graph of the landing approach using the full mission simulation, operating with closed-loop control laws around the monocular estimator, is shown in Figure 5.3. Errors in the estimate come from monocular tracking errors and not knowing the exact landing location. The end result was still the same. The UAV landed at the exact point chosen by the SLAD algorithms.

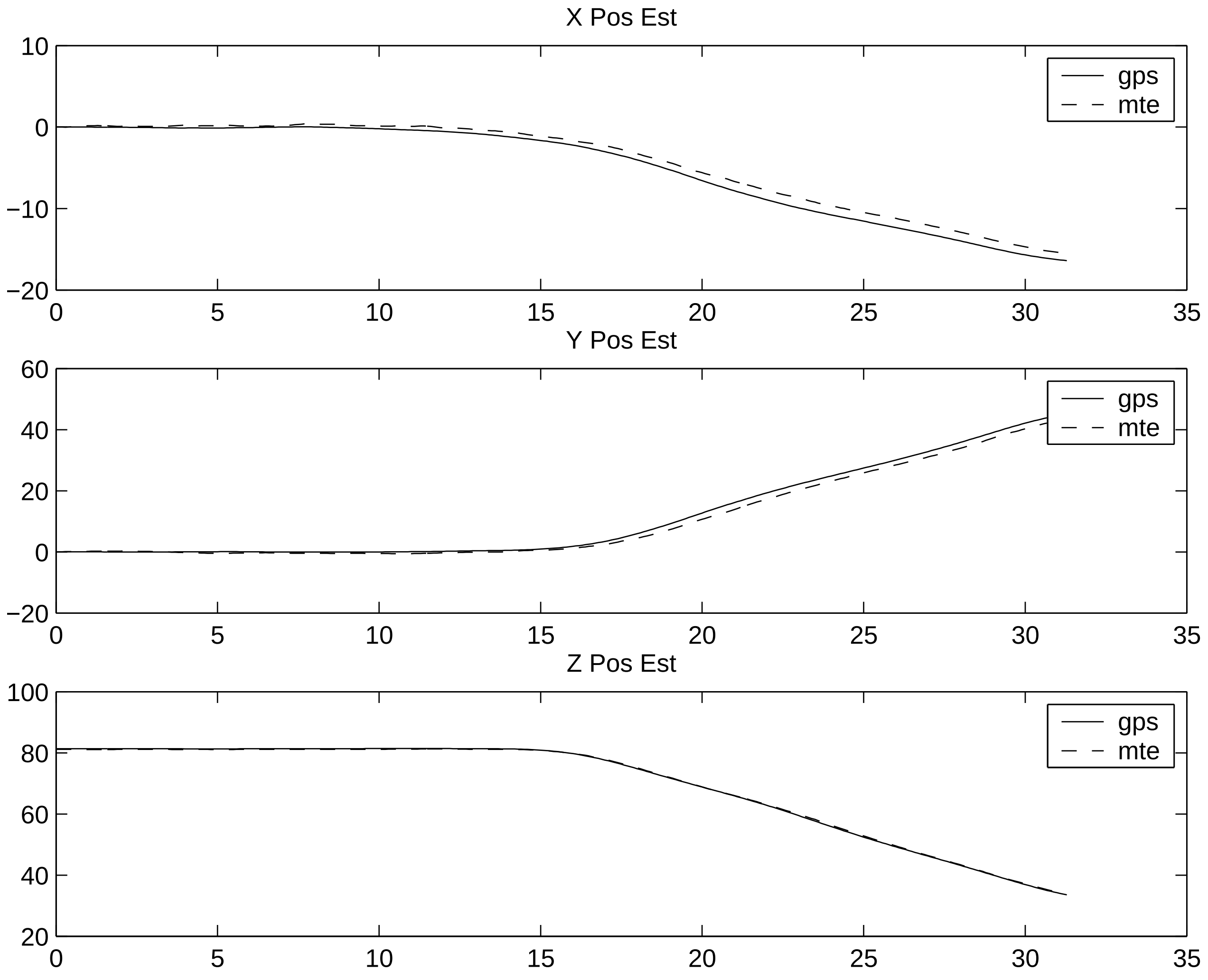


Figure 5.3: Mission landing results

Chapter 6

Conclusions and Recommendations

The PALACE mission was to demonstrate a complete autonomous landing of an unmanned aerial vehicle in a non-cooperative environment. In this thesis we created a demonstration that allows the vehicle to first fly to the remote landing site. We then used stereo vision algorithms to generate 3D information about the landing zone. Using the camera attitudes and known placement of the camera, the 3D data was rotated and translated creating an elevation map. The 3D data was searched by the safe landing area determination algorithms. A landing point was chosen and converted into 2D screen coordinates. The 2D pixel location was tracked by the monocular position estimation algorithms for use in a second Kalman filter. Velocities were estimated by the Kalman filter and fed into the control laws. The vehicle was then directed to fly a normal waypoint path to the landing site.

The scenario described above currently works in the RIPTIDE simulator and portions have been demonstrated in hardware. Although the algorithms work adequately in many situations, there are limitations with the current implementation.

One limitation that requires addressing is the safe landing area determination algorithms. When the vehicle is at a high altitudes, the stereo ranging algorithms do not produce accurate results for small objects on the ground. This could cause small objects to be excluded from the search space, when in actuality the zone should be marked as hazardous.

Another obstacle that is hard to detect using stereo vision, is power wires or antenna wires. Since they are small, vision algorithms have a hard time differentiating between the wires and the background. Perhaps a different sensor could be used to locate these types of obstacles.

There are also some limitations with the monocular position estimator. Since the feature tracker is updating its matching template after every camera frame, a sudden occlusion of the landing site would cause the new tracking template to be updated with incorrect information. For example, this could happen if a bird flies in between the camera and the landing point.

Several options for future research are listed below:

- **Multiple trackers.** If the monocular tracker was updated to search and track multiple features it would be less susceptible to lost features. Another option would be to run multiple monocular position estimators in parallel. It could use the feature tracker's coherence value to place a weight on the current position estimate.

- **Dynamic safe landing area determination parameters.** Allowing the landing parameters to change dynamically, based on the vehicles current distance away from the landing site would be a great improvement. This change would allow the vehicle to validate the landing site as it moves closer.

- **Full hardware integration.** Limited testing on a hardware platform was performed. This included feature tracking while the vehicle was under GPS waypoint navigation. The early results of the hardware tests look promising, but the algorithms will need to be tested thoroughly before inner-loop control is used with the monocular position estimation.

Bibliography

[1] Ross Britton, "Unmanned Aerial Vehicles: Changing the Sense of Self for the U.S. Military", March 2002, http://www.sit.wisc.edu/~wrbritton/Index.htm.

[2] John Roos, "Organic Air Vehicle", *Intelligence Surveillance Reconnaissance*, 2002, http://www.afji.com/ISR/Mags/2002/Issue2/organic.html.

[3] Jay Farrell, "GPS/INS Based Lateral and Longitudinal Control Demonstration", Tech. Rep. FINAL, University of California, Riverside, 1998.

[4] Srikanth Saripalli, James Montgomery, and Gaurav Sukhatme, "Vision-based Autonomous Landing of an Unmanned Aerial Vehicle", in *Proceedings of the IEEE International Conference on Robotics and Auotomation*, May 2002.

[5] Srikanth Saripalli and Gaurav Sukhatme, "Landing on a Moving Target using an Autonomous Helicopter", in *Proceedings of the International Conference on Field and Service Robotics*, July 2003.

[6] Sierra Nevada Corporation, "Tactical Automatic Landing System", http://www.sncorp.com/uav2.html, January 2004.

[7] Sebastian Thrun and Mark Diel, "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.

[8] Paul Bellutta and Larry Matthies, "Terrain Perception for DEMO III", *Proceedings of the Intelligent Vehicle Conference*, October 2000.

[9] Larry Matthies and Hebert Maclachan, “A Portable, Autonomous, Urban Reconnaissance Robot”, *International Conference on Intelligent Autonomous Systems*, July 2000.

[10] Patrik Nilsson, “Stereo Vision for Mobile Robots”, Tech. Rep. FINAL, Department of Engineering and Technology, 2003.

[11] Larry Matthies, Yalin Xiong, and Clark Olson, “Computing Depth Maps from Descent Imagery”, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 392–397, 2001.

[12] Srikanth Saripalli, James Montgomery, and Gaurav Sukhatme, “Augmenting Inertial Navigation with Image-Based Motion Estimation”, in *IEEE International Conference on Robotics and Automation*, May 11-15 2002, pp. 4326–33.

[13] Larry Matthies and Andrew Johnson, “Precise Image-Based Motion Estimation for Autonomous Small Body Exploration”, *5th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, June 1999.

[14] Omead Amidi, *An Autonomous Vision-Guided Helicopter*, PhD thesis, Carnegie Mellon University, 1996.

[15] Omead Amidi and Takeo Kanade, “A Visual Odometer for Autonomous Helicopter Fight”, in *Proceedings of the Fifth International Conference on Intelligent Autonomous Systems (IAS-5)*, June 1998.

[16] Qurban Memon and Sohaib Khan, “Camera calibration and three-dimensional world reconstruction of stereo-vision using neural networks”, *Internation Journal of Systems Science*, vol. 32, no. 9, pp. 1155–1159, 2001.

[17] Hossein Mansur and Weiliang Dai, *RIPTIDE Installation and User’s Guide*, NASA Ames Research Center, 2003.

[18] Bernard Mettler, Mark Tischler, and Takeo Kanade, “System Identification Modeling of a Small-Scale Unmanned Rotorcraft for Flight Control Design”, *J. American Helicopter Society*, vol. 47, no. 1, pp. 50–63, January 2002.

[19] Matthew Whalley, Mark Takahashi, Greg Schulein, Michael Freed, Daniel Christian, and Robert Harris, "The NASA/Army Autonomous Rotorcraft Project", in *American Helicopter Society Annual Forum*, 2003.

[20] Jan Roskam, *Airplane Flight Dynamics and Automatic Flight Controls*, Design,Analysis and Research Corporation, 3rd edition, 2001.

[21] Ginés Garcia Mateos, "A Camera Calibration Technique using Targets of Circular Features", Department of Computer Science and Systems, University of Murcia.

[22] 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, 1986.

[23] 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.

[24] Mark Maimone, "CAHVOR Camera Model Information", http://telerobotics.jpl.nasa.gov/people/mwm/cahvor.html.

[25] Larry Matthies and Yalin Xiong, "Error Analysis of a Real-Time Stereo System", *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997.

[26] Andrew Johnson, Allan Klump, James Collier, and Aron Wolf, "LIDAR-base Hazard Avoidance for Safe Landing on Mars", in *AAS/AIAA Space Flight Mechanics Meeting*, Santa Barbara, CA, February 2001.