

Analysis of LIDAR-derived bare ground model accuracy in southern California chaparral

Andrew G. Cooke

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2008

Program Authorized to Offer Degree:
Forest Resources

University of Washington
Graduate School

This is to certify that I have examined this master's thesis by

Andrew G. Cooke

and have found it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Gerard F. Schreuder

Stephen E. Reutebuch

Hans-Erik Andersen

L. Monika Moskal

Date: _____

In presenting this thesis in partial fulfillment of the requirements for a master's degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of the thesis is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Any other reproduction for any purpose or by any means shall not be allowed without my written permission.

Signature _____

Date _____

University of Washington

Abstract

Analysis of LIDAR-derived bare ground model accuracy in Southern California chaparral

Andrew G. Cooke

Chair of the Supervisory Committee:
Professor Gerard F. Schreuder
Forest Resources

LIDAR is developing as a useful tool for measuring biomass. Accurate measurements of biomass may be useful in predicting both the behavior and intensity of wildfire. A better understanding of wildfire behavior in the fire-prone and densely populated areas around Los Angeles and San Diego would be of great benefit. The chaparral ecosystem of Southern California provides a unique operational setting for LIDAR systems in which assumptions of laser behavior may not hold true, calling into question the accuracy of LIDAR-based biomass estimates. A study was undertaken to determine the accuracy of LIDAR-derived bare ground models as a precursor to using LIDAR above-ground returns for biomass estimation. A real-time kinematic GPS survey of five research sites in the San Bernardino and Los Padres National Forests with vegetation cover ranging from low shrubs with grass to dense, mature chaparral was undertaken in June 2007. Measurements of ground elevation, vegetation height, and vegetation density were taken at 1709 points, and these measurements were compared to LIDAR data collected in August of 2005. Methods were used to account for vegetation growth and systematic differences between the LIDAR and GPS data. Overall it was shown that the LIDAR ground elevations were 0.04 ± 0.61 meters above the GPS ground elevations. In the highest density vegetation, the LIDAR ground elevations were 0.10 ± 0.69 meters above the GPS ground elevations, and this increased to 0.13 ± 0.70 meters in the tallest vegetation category.

TABLE OF CONTENTS

	Page
List of Figures	iii
List of Tables	v
1. Introduction.....	1
1.1 Introduction.....	1
1.2 Objectives	3
2. Background.....	4
2.1 LIDAR	4
2.2 Global Navigation Satellite System.....	10
2.2.1 How Position is Determined	10
2.2.2 GPS / GLONASS Constellations.....	11
2.2.3 Signals.....	12
2.2.4 Types of Solutions	13
2.2.5 Positional Dilution of Precision (PDOP).....	14
2.2.6 Types of Receivers.....	14
2.2.7 Types of Errors	15
2.2.8 Differential Correction / RTK.....	17
2.3 Chaparral.....	19
3. Methods.....	20
3.1 Field Sites.....	20
3.2 GPS Equipment.....	21
3.3 LIDAR Data.....	23
3.4 LIDAR Quality Assessment	24
3.5 Determining Field Data Collection Sites	26
3.5.1 Vegetation Transects.....	26
3.5.2 Control Points	37
3.6 Field Data Collection	39
3.6.1 Base GPS Points	39
3.6.2 Control GPS Points.....	40
3.6.3 RTK Vegetation GPS Points.....	41
3.6.4 Non-RTK Vegetation GPS Points	44
3.7 Vegetation Growth.....	44
3.8 LIDAR Ground Model.....	46
4. Results.....	52
4.1 Collected GPS Data	52
4.1.1 Overview of GPS Data.....	52
4.1.2 GPS Data Quality.....	54
4.2 Vegetation Growth.....	62
4.3 LIDAR / GPS comparison	63
4.3.1 Elevation Difference Formula.....	63
4.3.2 Local Offset	63
4.3.3 Elevation Differences.....	63

4.3.4 Predicting DTM Accuracy	68
5. Discussion	74
5.1 Error Budget.....	74
5.2 Triangle Method Accuracy	77
5.3 Conclusions about the Objectives.....	80
Bibliography	85
Appendices.....	89
Appendix A: Triangle Method Code 1	89
Appendix B: Triangle Method Code 2.....	113

LIST OF FIGURES

Figure Number	Page
1 – Simulated multi-modal LIDAR reflected energy waveform for a single pulse.....	5
2 – CFD method of determining a return from the LIDAR waveform, full-sized (top) and zoomed-in (bottom).....	6
3 – Determining position using GPS satellites	11
4 – PDOP resulting from different satellite configurations	14
5 – GPS choke-ring antenna (source Leica Geosystems)	16
6 – Map of fieldwork locations near Los Angeles, CA	20
7 – Javad Navigation Systems Maxor-GGDT GNSS receiver (source Javad Navigation Systems, Inc.).....	22
8 – Map of LIDAR point density for site Old 1.....	26
9 – Example of determining slope for site Old 1	29
10 – Example of determining slope for site Old 1	30
11 – Example of determining vegetation height for site Old 1.....	31
12 – Example of determining vegetation height for site Old 1.....	32
13 – Example of areas with first returns only (light green) and other returns (dark green) for Old 1	33
14 – Example of areas fitting collection criteria (green) for site Old 1	34
15 – Example of areas fitting collection criteria colored by vegetation density for site Old 1.....	35
16 – Example of data collection area polygons for site Old 1.....	36
17 – Example of data collection area polygons overlaid on orthophoto for site Old 1	37
18 – Example of flat areas (blue), bare areas (green), and control points (red) at site Old 1	38
19 – Example of local base station GPS, with receiver and RTK radio antenna.....	40
20 – Example control GPS point	41
21 – RTK GPS rover (left) and non-RTK GPS rover (right).....	42
22 – Height markings on GPS pole.....	43
23 – One of three plots used for measuring growth, 2005 vegetation height	45
24 – 2007 GPS points with interpolated 2005 vegetation heights	46
25 – Columnar minimum selection result possibilities. GPS point (red) LIDAR points (blue), lowest LIDAR point in column (yellow)	47
26 – LIDAR returns within two meters of a GPS point sorted into bins	48
27 – Example LIDAR ground triangle not containing the GPS point	49
28 – Example LIDAR ground triangle containing the GPS point	50
29 – Map of local GPS base stations and the CORS sites used to correct them	53
30 – All vegetation GPS points, PDOP vs. number of satellites, RTK (blue), non-RTK (red)	55

31 – Vegetation GPS points RMS vs. PDOP, RTK (blue) non-RTK (red)	57
32 – Cumulative distribution function of the RMS field data values	58
33 – Histogram of the RMS field data values with a normal distribution overlaid	59
34 – Histogram of the RMS field data values with an exponential distribution overlaid	60
35 – Field data RMS values (meters) vs. calculated LIDAR-GPS height Differences (meters).....	61
36 – Simulated RMS values (meters) vs. simulated LIDAR-GPS height differences (meters)	62
37 – Average LIDAR-GPS height differences (meters) and Standard Deviations (meters) for all Triangle Method selection radii (meters).	64
38 – Box plot of LIDAR-GPS height differences for vegetation density categories.....	66
39 – Box plot of LIDAR-GPS height differences for 2007 vegetation height categories	67
40 – Box plot of LIDAR-GPS height differences for 2005 vegetation height categories	68
41 – Plot of LIDAR-GPS height difference results against 2007 vegetation heights. RTK vegetation points (green), non-RTK vegetation points (blue), regression line (red)	70
42– Plot of LIDAR-GPS height difference results against categorized 2007 vegetation heights. RTK vegetation points (green), non-RTK vegetation points (blue), regression line (red)	71
43 – Plot of LIDAR-GPS height difference results against density categories. RTK vegetation points (green), non-RTK vegetation points (blue), regression line (red).....	72
44 – Histogram of simulated vertical errors caused by ground GPS point accuracy limitations on ground slopes of ± 50 degrees	75
45 – Histogram of simulated vertical errors caused by ground GPS point and LIDAR system accuracy limitations on ground slopes of ± 50 degrees	76
46 – Map of DTM height differences (Triangle Method – Fusion) in meters for site Old 1	78
47 – Histogram of DTM height differences (Triangle Method-Fusion) in meters for site Old 1.....	79
48 – Average LIDAR-GPS height differences (meters) and Standard Deviations (meters) for Triangle Method selection radii (meters) at sites: Old 1, Grand Prix, Piru 1, and Piru 2	80
49 – Average LIDAR-GPS height differences (meters) and Standard Deviations (meters) for Triangle Method selection radii (meters) at site: Old 2.....	81
50 – Overhead views (both top) and side views (both bottom) of Grand Prix (both left) and Old 2 (both right) sites.....	82

LIST OF TABLES

Table Number	Page
1 – Vegetation species encountered at sites	21
2 – Javad Accuracy Specifications (source Javad Navigation Systems, Inc.)	23
3 – LIDAR collection parameters (source Watershed Sciences, Inc.)	23
4 – LIDAR system accuracy specifications (source Optech, Inc.)	24
5 – Calculated LIDAR point densities	25
6 – Area of the sites in hectares	26
7 – Parameters used in filtering LIDAR ground returns	28
8 – Vegetation point density criteria	43
9 – Base GPS OPUS processing results	52
10 – Number and type of GPS points at each site	54
11 – Number, height, and density of vegetation GPS points at each site	54
12 – The number and percentage of GPS points in three RMS categories, RMS in meters	56
13 – Average control GPS point ground elevation offset (LIDAR-GPS) for each site in meters	63
14 – Ground elevation differences (LIDAR-GPS) for vegetation GPS points, in meters	65
15 – ANOVA results	69
16 – R^2 values for various linear regression models	73
17 – LIDAR DTM-GPS ground elevation differences for Fusion and Triangle Method DTMs	77
18 – Percentage of area in Old 1 site in each height difference category	79
19 – Percentage of vegetation height underestimation due to LIDAR DTM error	83

ACKNOWLEDGEMENTS

I would like to thank the Precision Forestry Cooperative, the University of Washington College of Forest Resources, the USDA Forest Service PNW Research Station, the Joint Fire Science Program*, and the Corkery Family for their generous support of my work. This research project could not have been completed without the knowledge, resources, and funding provided by these groups.

In particular I would like to thank my committee chair Gerard Schreuder for giving me the opportunity to pursue this project. I would also like to thank, Steve Reutebuch, Hans-Erik Andersen, and Robert McGaughey, of the Forest Service, and Monika Moskal and Megan O'Shea of the College of Forest Resources for their guidance, scientific, technical, and otherwise.

The arduous field work for this project could not have been completed without the immense effort of Tobey Clarkin and Jacob Strunk. I give you both special thanks for your help and friendship.

The friendship and support of my fellow Precision Forestry students Akira Kato, Sooyoung Kim, Yuzhen Li, and Alicia Sullivan, as well as Rural Technology Initiative staff Ara Erickson, Matthew McLaughlin, and Luke Rogers, have made this experience both enjoyable and greatly rewarding.

Finally, I'd like to thank my family, and in particular, my wife, for all of their unending encouragement, love, guidance, and support.

* Joint Fire Science Program Project Number: 04-1-2-02

Mapping and analysis of pre-fire fuels loading and burn intensity using pre-fire interferometric synthetic aperture radar data combined with burn intensity derived from post-fire multispectral imagery for the 2003 southern California fires.

INTRODUCTION

1.1 Introduction

Light Detection and Ranging (LIDAR) systems are emerging as useful tools for measuring biomass. Accurate measurements of biomass may be useful in predicting both the behavior and intensity of wildfire. A better understanding of wildfire behavior, especially in the wildland-urban interface, would be of great benefit. Accurate biomass estimates are dependent on accurate ground elevation models, and LIDAR is unique in its ability to provide these models.

Many methods have been used to create LIDAR-derived digital terrain models (DTMs) in a wide variety of vegetation types. The companies collecting the LIDAR data can provide filtered ground data or ground models developed with proprietary software, or raw data to be processed into DTMs using methods determined by the customer. Haugerud and Harding [20] created triangulated irregular networks (TINs) from a filtered last return dataset in a Pacific Northwest forest site. Hodgson et al. [21] used TINs derived from filtered ground returns in multiple vegetation types in North Carolina. Riaño et al. [32] used first percentile LIDAR elevations determined from raw data to calculate the elevations of grid cells in a forest in Germany. Riaño also used vendor provided DTMs for work in Spain [33]. Kraus and Pfeifer [23] and Andersen et al. [2] used a recursive method (described in Section 3.5.1) to develop DTMs in forests in Vienna and Washington State from raw data. Töyrä et al. [39] and Hopkinson et al. [22] interpolated vendor provided ground returns into DTMs for wetlands. Goodwin et al. [17] used a similar method for a eucalyptus forest in New South Wales, Australia. Mundt et al. [29] and Streutker et al. [38] used another recursive method on raw LIDAR returns in sagebrush vegetation in Idaho and Wyoming.

Despite the many methods of creating DTMs and the wide conditions in which they are used, developing an accurate terrain model from LIDAR data is dependent upon having returns from the ground, and on determining which returns in the “point cloud”

are from the ground. If there are no true ground returns, there is no way to build an accurate ground model.

In a 2003 study [34], Reutebuch et al. determined that LIDAR-derived DTMs achieved elevation accuracies of less than 20 centimeters in clear cuts and thinned forests, and 31 centimeters in mature, closed-canopy, Douglas-fir forest.

The chaparral ecosystem in southern California, provides a uniquely challenging operating environment for LIDAR systems. The dense, short vegetation greatly reduces the likelihood that laser pulses are hitting the ground. It is expected that in these vegetation conditions that LIDAR ground models will overestimate true ground height, resulting in underestimated biomass.

Dense vegetation prevents laser pulses from penetrating to the ground by reflecting, scattering, or absorbing all of a pulse's energy in the upper portions of the canopy. As a result, in very dense vegetation, most returns will be from on top of or just inside of the vegetation canopy.

Multiple laser reflections from short vegetation may not be distinguishable in current LIDAR systems. As a result, a LIDAR sensor will use some method to choose whether one of the reflections or some combination of the multiple reflections is a return. Further details of this constraint are provided in Section 2.1. Streutker [38] reported in May of 2006, that the resolving ability of common LIDAR sensors was not discussed in the literature. In a journal article in July of 2006, it is mentioned that the Optech ALTM 3025 sensor cannot distinguish between first and last returns from objects within 4.9 meters of one another due to the circuitry used in the system [17]. The effect this has on returns between the first and the last is unknown.

Large areas of the LIDAR data used in this project have only single returns, raising the question of whether or not laser pulses are penetrating the vegetation. If they are not, modeled ground surfaces in the conditions experienced in chaparral have a high likelihood of being inaccurate, which in turn creates errors in derivative work such as biomass estimates. The regular occurrence of fire in chaparral, and the high population density and large spatial extent of the wildland-urban interface in areas where chaparral

occurs, make the development of accurate biomass estimates and wildfire behavior models very important.

1.2 Objectives

The primary goal of this study is to determine the accuracy of LIDAR-derived ground models in chaparral. If they are accurate, then they can be used for other work without concern. If laser pulses are penetrating to the ground in chaparral, then ground models created from these pulses using existing methods should be very accurate. To this end, a survey of field sites in southern California was undertaken to measure true ground elevations using Global Navigation Satellite System (GNSS) equipment. Differences between GNSS and LIDAR ground elevations will provide a metric of the accuracy of LIDAR-derived ground models.

It is also of interest to determine whether or not the height and density of chaparral have an effect on the accuracy of LIDAR-derived ground models. The likelihood that LIDAR will reach and reflect from the ground is directly related to the height and density of vegetation. It is therefore assumed that increasing density and height (between zero and three meters) in chaparral will increase the error in LIDAR ground models.

Finally, it is of interest to determine if any inaccuracies in LIDAR-derived DTMs are significant. In other words, are the inaccuracies within ranges of LIDAR and GNSS positional accuracy, and are they small enough to have minimal effect on biomass estimates?

Chapter 2

Background

2.1 LIDAR

While it may take other forms, including large-footprint, continuous-waveform, digitized waveform, and terrestrial, for this thesis, LIDAR is a discrete-return, small-footprint, airborne laser scanner surveying system, based on a pulsed laser range finder. The laser mounted on the aircraft sends out a pulse of energy which reflects off of objects below the aircraft to a receiving sensor onboard. The time between when the pulse was sent and when the reflection was received can be translated into distance from the aircraft using the formula:

$$R = c \frac{t}{2}$$

where R is the distance, t is the time, and c is the speed of light [14]. The exact position and orientation of the laser at the time each pulse is fired is determined through the use of positioning and inertial measurement units onboard the aircraft, allowing a location for each reflection to be determined.

A laser pulse can reflect off of multiple objects, with many LIDAR systems configured to collect four or five reflections (returns) for each pulse. The receiving sensor on the aircraft measures incoming laser energy in discrete time increments.

The exact method used to determine which time interval along the reflected energy waveform corresponds to a return is determined by the sensor manufacturer, and is proprietary knowledge. However, most commercial LIDAR systems currently in operation use an avalanche photodiode (APD), and a constant fraction discriminator (CFD) to determine returns for a pulse [36]. Reflected energy is measured using the APD, which converts incoming photons into electrical current. The changes in the electrical current over time, correspond to the amount of energy reflected from objects below the aircraft, and can be represented as a waveform (Figure 1). If there is a single return (peak) from the pulse in the waveform, it is referred to as a unimodal waveform. If the pulse has multiple returns, the waveform is multi-modal.

The CFD is used to process the analog signal from the APD to determine returns. This is done by adding an inverted, attenuated version of the waveform to a delayed version of the waveform to create a CFD waveform. The time interval at which this CFD waveform value switches from negative to positive corresponds to the full-width-half-maximum (FWHM) point on the leading edge (front half) of each waveform peak (Figure 2). The time interval for each of these FWHM points becomes a return [36].

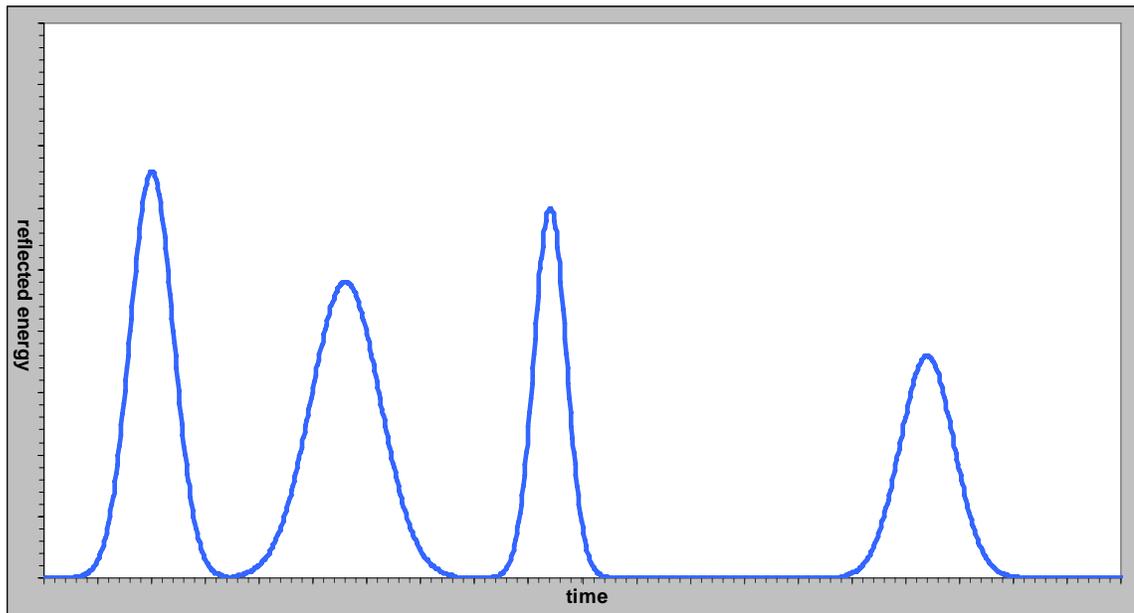


Figure 1 – Simulated multi-modal LIDAR reflected energy waveform for a single pulse.

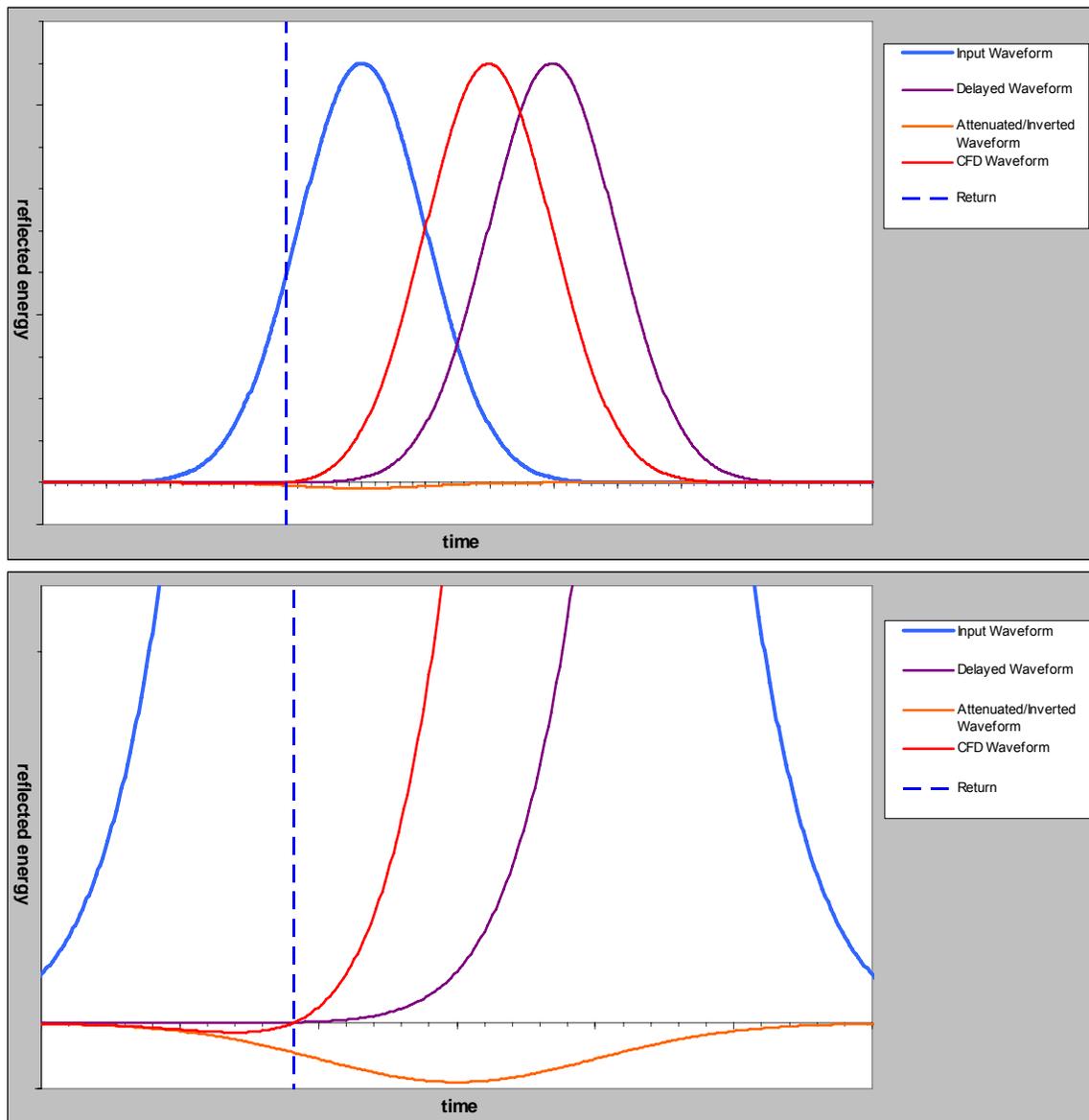


Figure 2 – CFD method of determining a return from the LIDAR waveform, full-sized (top) and zoomed-in (bottom).

A major limiting factor in LIDAR systems is what is known as “dead time”. This is the amount of time it takes for the circuitry in the sensor to return to steady state and become ready to measure a second return within a multi-modal waveform. If two returns arrive at the sensor within the “dead time” interval of one another, the circuitry cannot detect the second return. The major factor determining the length of the “dead time” interval is the length of the transmitted laser pulse, but the details of the circuitry in the

APD and CFD also play a role [36]. The length of the laser pulses in the LIDAR data used for this thesis are not known, but common pulse lengths range from six to 10 nanoseconds. This means that if two returns from a pulse arrive at the sensor within six to 10 nanoseconds of one another the second will not be detected. A six nanosecond laser pulse is approximately 1.8 meters long, while a 10 nanosecond laser pulse is approximately three meters long. This means that if two objects within 1.8 to three meters of one another both reflect a pulse's energy, only the first object will be recorded as a return by the system.

The research sites examined in this thesis had only single returns per pulse in the majority of the chaparral areas. The height of the chaparral was typically shorter than or equal to the maximum "dead time" distance of three meters. It seems likely that even if multiple reflections occurred for each laser pulse, the system was only able to record one return. This would result in a lower percentage of returns from the ground than would be expected in other operating conditions. This problem would be exaggerated further if the high density of chaparral is also considered.

The shorter the vegetation, the more likely it is that a second return will not be recorded, due to the "dead time" limitation. But as the vegetation height decreases, it is also more likely that any single return will be near the ground surface. Furthermore, as the vegetation height decreases, there are fewer reflective surfaces between the top of the vegetation and the ground, increasing the likelihood that a laser pulse could find a hole in the vegetation and reflect from near or on the ground. Once the vegetation reaches a height of three meters or greater, it becomes possible for a second return to be reflected from near the ground.

It is possible that LIDAR system manufacturers developed a method other than CFD to determine returns from the reflected energy waveform, such as identifying the FWHM point on the trailing edge (back half of a peak in the waveform). However, this method still requires that the circuitry can reset to detect later returns. "Dead time" remains a limiting factor in the detection of multiple returns for a pulse even if different methods of interpreting the waveform are used.

Lasers used in LIDAR systems are most often in the near infrared frequency range of 900-1100 nanometers (nm), with the most frequently used laser being a 1064 nm diode pumped Nd:YAG (neodymium-doped yttrium aluminum garnet; Nd:Y₃Al₅O₁₂) laser [36].

The size of the laser pulse when it hits an object (the laser footprint) is a function of the distance between the aircraft and the object and the divergence of the laser beam. The divergence of the laser can vary from system to system, but should be provided by the manufacturer of the LIDAR system. Footprint size for a laser pulse directly nadir to the aircraft can be determined using this formula:

$$D_L = a + 2h \tan\left(\frac{\gamma}{2}\right)$$

where D_L is the beam diameter, a is the laser aperture, h is the distance from the laser to the object and γ is the beam divergence [14]. For an aircraft flying 1000 meters above bare ground and using a laser with a beam divergence of 0.3 milliradians (0.0003 radians) and an aperture of 0.01 meters, a laser pulse directly below the aircraft will have a diameter of approximately 0.31 meters.

Laser power is a limiting factor in both aircraft flying height and laser firing speed (pulse rate). For a system with an average power of P_{av} the system power available for each pulse P_{Peak} can be determined with this formula:

$$P_{Peak} = \frac{P_{av}}{t_p * F}$$

where t_p is the length of the pulse in seconds and F is the frequency of the system (pulse rate) [14]. For a system with average power of 0.2 watts, a pulse length of 10 nanoseconds, and a frequency of 10 kilohertz (kHz), P_{Peak} is 2 kilowatts (kW). If the frequency is increased to 71 kHz (the firing rate for the data used in this thesis), P_{Peak} drops to 0.3 kW. Higher frequencies result in less power available for each pulse. With reduced pulse power there is less energy available to reflect to the aircraft requiring the aircraft to reduce altitude in order to collect data. The reverse is also true; higher altitude flying requires slower pulse rates in order to produce reflections strong enough to be detected.

Other factors also influence flying height, including beam divergence and atmospheric moisture and pollution. Higher altitudes result in more atmospheric moisture and pollution between the laser and objects being measured increasing absorption and scattering of the laser pulse. The further a laser beam diverges, the less energy available per unit area of footprint. At great enough altitudes, there is not enough energy available in the pulse to reflect to the aircraft.

The density of LIDAR returns underneath the aircraft (the number of returns per unit area) is affected by the altitude and speed of the aircraft, as well as the pulse rate, scan angle, and scan frequency of the LIDAR system. Overlapping the flight lines increases return density.

Scan angle is the measurement of the laser mirror oscillation from nadir perpendicular to the flight line of the aircraft, which is an adjustable parameter in LIDAR systems. For a given flying height, the larger the scan angle, the wider the area of ground covered by each flight line (swath width) as determined by this formula:

$$SW = 2h * \tan\left(\frac{\theta}{2}\right)$$

where SW is the swath width, h is flying height, and θ is the scan angle in degrees [14]. For a given scan angle, increasing aircraft altitude will also increase swath width. For an aircraft flying at 1000 meters, with a scan angle of 18 degrees the swath width over flat ground would be approximately 317 meters.

Scan frequency is the number of scan lines (oscillation of the laser mirror across the full scan angle) a system can perform in a second. This parameter is adjustable in LIDAR systems as well. For a given pulse rate and scan frequency, a larger scan angle or higher aircraft altitude results in greater spacing between laser reflections.

$$dx_{across} = \frac{SW}{N}$$

SW is defined above and N is the number of points per scan line as defined by:

$$N = \frac{F}{f_{sc}}$$

where F is the system pulse rate and f_{sc} is scan frequency [14]. A given pulse rate and scan frequency will make N a constant. Increasing the flying height or scan angle will

increase SW resulting in increased spacing between laser reflections in a scan line. Similarly, for a given altitude and scan angle, increasing pulse rate or reducing scan frequency will result in smaller spacing between laser reflections within a scan line.

Average spacing between scan lines is a factor of aircraft speed and scan frequency.

$$dx_{along} = \frac{v}{f_{sc}}$$

where v is aircraft speed and f_{sc} is scan frequency [14]. For a given scan frequency, increased aircraft speed results in greater spacing between lines.

2.2 Global Navigation Satellite System

A Global Navigation Satellite System (GNSS) is a way to determine a position on Earth using satellites. As of this writing, there are only two such systems in operation, the United States' Global Position System (GPS), which is fully operational, and the Russian Federation's GLONASS, which is not yet fully implemented. For the remainder of this thesis, except when talking about system specifications below, the term GPS will be used in place of GNSS, and refers to both the GPS and GLONASS systems used to collect data.

2.2.1 How Position is Determined

At its simplest, satellite-based positioning relies on determining the distance between a receiver and three or more satellites. The position of each satellite is known, so the receiver position on earth can be triangulated using the distance to various satellites. Each satellite in the GPS or GLONASS constellation sends out a radio signal at a specific time. The receiver has an accurate clock and produces an exact copy of the satellite's signal at the same time as the satellite. The signal from the satellite takes time to reach the receiver, resulting in a time offset between the satellite and receiver signals. This difference is the amount of time the signal travelled. Multiplying this length of time by the velocity at which the signal travelled (the speed of light) results in the distance between the receiver and the satellite. There are four unknowns for which a solution

needs to be determined, the receiver's latitude, longitude, elevation, and clock time. Having four unknowns requires having four distance measurements. Knowing the receiver distance from a single satellite places it on a sphere (Figure 3). Knowing the distance from two satellites limits the receiver position to a circle. Knowing the distance to three satellites limits the position to two points (usually only one of these two points makes sense). Knowing the distance to four satellites limits the position to a single point and makes possible the solving of clock accuracy errors.

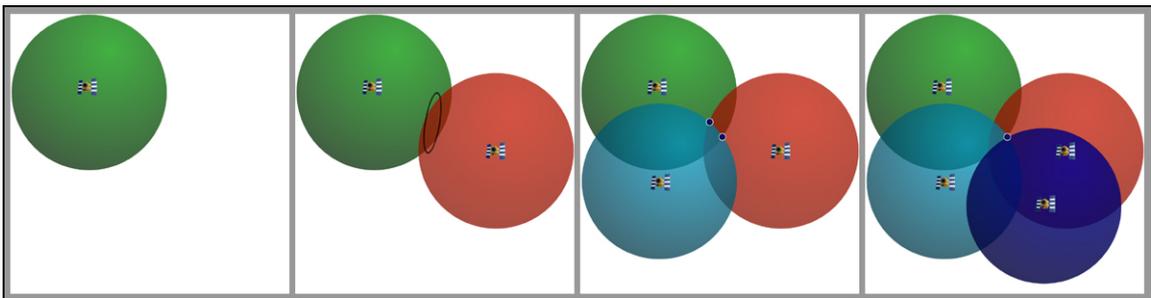


Figure 3 - Determining position using GPS satellites.

2.2.2 GPS / GLONASS Constellations

The United States' GPS constellation is made up of a minimum of 24 satellites and has had up to 32 at any one time. These satellites are organized in six circular orbital planes equally spaced 60 degrees from one another around the equator and inclined 55 degrees from the equatorial plane. There are at least four functional satellites per plane, each of which operates at an altitude of 20,200 km and has an orbital period of approximately 12 hours. As new satellites are launched to replace older satellites, there are often extra satellites in each plane providing redundancy until the older satellites are decommissioned. During field work for this thesis, there were 30 operational GPS satellites.

The GLONASS satellite constellation of the Russian Federation is a Global Navigation Satellite System intended to have 24 satellites. Of these satellites 21 are intended to actively broadcast signals, while three are intended as reserve satellites. The satellites are organized into three planes equally spaced 120 degrees apart around the equator and inclined 64.8 degrees from the equatorial plane. In a fully operational

system, there would be eight satellites per plane, each in a circular orbit with a radius of 19,100 km, and an orbital period of approximately 11 hours, 15 minutes. The GLONASS constellation has never been completed. During field work for this thesis, there were 12 operational GLONASS satellites.

2.2.3 Signals

GPS satellites transmit two microwave carrier signals. The L1 signal is transmitted at a frequency of 1575.42 MHz (\approx 19cm wavelength), and the L2 signal is transmitted at 1227.60 MHz (\approx 24cm wavelength). Each satellite modulates these carrier signals in three ways.

1. The Course-Acquisition (C/A) code modulates the L1 frequency. It is a 1,023 bit pseudo-random noise (PRN) code repeated every millisecond. Each satellite has its own PRN code, making it uniquely identifiable from the other satellites broadcasting on the same frequency. The Department of Defense can turn on Selective Availability (SA) for the C/A code, reducing positional accuracy by manipulating the orbit data and satellite clock frequency in the navigation message. Selective Availability was turned off permanently May 1, 2000.
2. The Precise (P) code modulates both the L1 and L2 frequencies. The P code is a 10.23 MHz PRN code that repeats once every seven days. The military has the ability to encrypt this signal in what is known as anti-spoofing (A-S) mode. When the P code is encrypted, it is known as Y code. Civilian receivers currently on the market cannot use the P code, but some can use the L2 carrier signal.
3. The Navigation Message also modulates the L1 frequency. The navigation message includes the ephemeris of the transmitting satellite, an almanac of all GPS satellite positions, approximate corrections for ionospheric delay, and the offset between the satellite's clock time and true GPS time.

GLONASS satellites transmit in two microwave spectrum ranges, also referred to as the L1 and L2 bands. The L1 band is used to transmit both the C/A and P signals, and the L2 band is used to transmit only the P code. Each satellite broadcasts its C/A and P

signal at a unique frequency in the assigned frequency ranges. The L1 band spans the frequency range from 1602.5625 MHz to 1615.5 MHz, with the exact frequency center for each satellite calculated using the equation: $1602 \text{ MHz} + n \times 0.5625 \text{ MHz}$ where n is the satellite's channel number (0-23). The L2 band spans the frequency range from 1240 MHz to 1260 MHz, with the exact frequency center for each satellite calculated using the equation: $1246 \text{ MHz} + n \times 0.4375 \text{ MHz}$. GLONASS satellites do not use selective availability.

2.2.4 Types of Solutions

A position calculated using GNSS is called a solution. The primary solution type is a Course-Acquisition (C/A) or Code solution, which is based on using the C/A code to calculate the pseudorange to multiple satellites and triangulate position on Earth. The pseudorange becomes the range to a satellite when errors have been corrected. Pseudorange is calculated when a receiver detects and identifies a satellite's PRN code, generates a replica of the code, measures the phase difference between the satellite and replica codes, and multiplies this by the speed of light. The C/A code is one microsecond in length. At the speed of light, a one microsecond signal is approximately 300 meters in length, so if the PRN code and replica code are unaligned by the full signal phase, the result would be a 300 meter position error. While modern receivers can get much closer than this, even a one percent misalignment will result in a 3m position error.

Increased accuracy can be achieved by calculating a second solution type called a Carrier solution. The wavelength of the L1 carrier signal is approximately 19cm, which is over 1000 times shorter than the length of the C/A code. If the receiver can count the exact number of L1 frequency cycles between the satellite and the receiver, a solution could be calculated with millimeter accuracy. However the carrier signal is uniform, making counting cycles very difficult. If the receiver is able to determine an integer number of cycles, (ambiguity resolution) a fixed carrier solution is obtained. If an integer number of cycles cannot be resolved, a float carrier solution is obtained. The pseudorange can be used to estimate the approximate number of cycles, and reduce processing time to reach a fixed solution. A receiver using both the L1 and L2 carrier

frequencies can create a “virtual carrier frequency” [18] by calculating the difference between the L1 and L2 frequencies ($1575.42\text{MHz} - 1227.60\text{MHz} = 347.82\text{ MHz}$) further reducing processing time. It is not possible for a single receiver to accurately account for signal and timing errors when resolving cycle count. Carrier solutions require a second GPS operating as a base station collecting data simultaneously.

2.2.5 Positional Dilution of Precision (PDOP)

GPS position accuracy is directly affected by the arrangement of the satellites used to calculate the position. PDOP is a quality measurement of the satellite geometry. A high PDOP value for a position means that the satellites were grouped together during the data collection, while a low PDOP value means that the satellites were spread apart. A receiver will have nearly identical range measurements to all of the satellites in a tight group (Figure 4). This will magnify small ranging errors including those from other sources, such as clock errors [18]. Having a low PDOP value reduces the likelihood of having significant ranging errors.

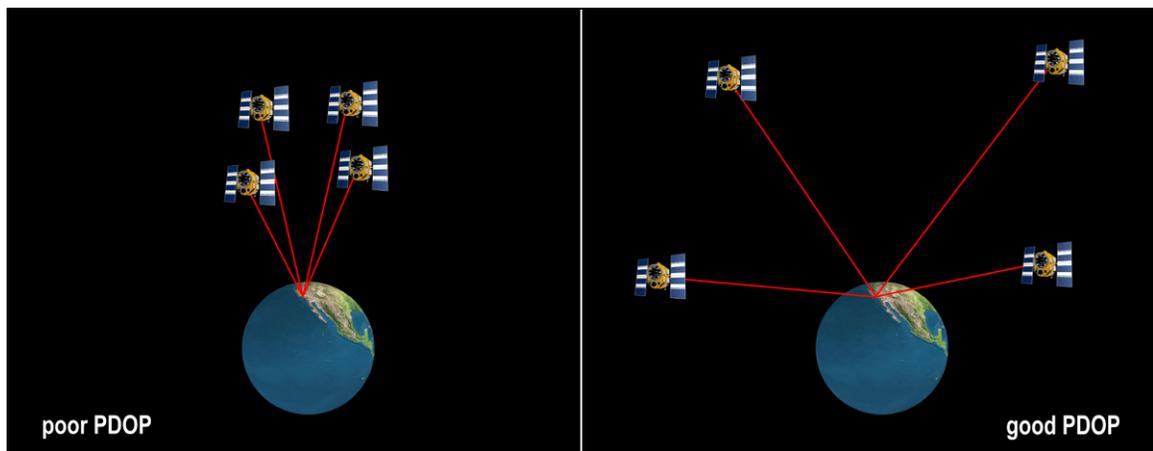


Figure 4 – PDOP resulting from different satellite configurations.

2.2.6 Types of Receivers

There are many types of GPS receivers with features that vary from model to model. The number of channels a receiver has determines the number of satellites with which it can communicate at a given time. Increasing the number of channels increases

number of satellites that can be used to determine a solution. A single frequency receiver can only use the L1 frequency, while a dual frequency receiver also has the ability to use the L2 carrier. Some receivers will only be able to calculate code solutions, while others will be able to calculate carrier solutions. Most receivers on the market today only use GPS satellites, while a few manufacturers make models able to use GLONASS and Galileo (when and if it becomes operational) satellites. Finally, some receivers are able to use real time correction information to improve their solutions.

Receivers fall into several grade categories, based on their functionality. Two grades are typically used for resource management research. Mapping or resource grade receivers are typically single frequency receivers, able to use C/A code and the L1 carrier to calculate solutions with accuracy in the open of around 0.5 meters. Survey grade receivers are dual frequency receivers, able to use C/A code and both L1 and L2 carriers to calculate solutions with accuracy in the open of one centimeter or less.

2.2.7 Types of Errors

In Section 2.2.4, it was mentioned that in order to get true range to a satellite, there are signal and timing errors which must be corrected. These errors include: multipath, ionospheric delay, tropospheric delay, and timing errors.

Multipath is caused when the ground or objects near the receiver reflect the GNSS signals, resulting in more than one signal from each satellite. In some operating environments, such as closed-canopy forest, the direct satellite signal may be blocked so that only multipath signals reach the receiver. In this case, there is no way to detect multipath errors, without extended occupation times. Reflected satellite signals have a longer path than the direct signal and therefore have a longer phase difference. These multiple signals result in altered amplitude and phase of the satellite signal, as well as increased difficulty resolving carrier signal ambiguity. Multipath is the dominant source of error in differential GPS, resulting in errors of up to 10 meters, and cannot be corrected using differential techniques [18]. There are several software methods for addressing multipath errors, but the simplest method to reduce multipath is through antenna design and placement. Groundplane antennas are shielded from reflected signals approaching

the antenna from below, but are affected by signals that arrive at the antenna edge. Choke-ring antennas are a specific type of groundplane antenna in which the shield is made up of a series of concentric circles spaced apart one quarter of a wavelength (Figure 5). These create high impedance at the GPS carrier frequency, effectively blocking multipath signals arriving from below or at the antenna edge. Choke-ring antennas tend to be impractical for much fieldwork due to their size and weight.



Figure 5 – GPS choke-ring antenna (source Leica Geosystems).

Shielded antennas are protected from multipath signals off of the ground, but not from multipath signals from above. Placing the antenna at a location with clear sky view and no surrounding objects is the most effective way to deal with this type of multipath error, but is not always feasible. Long observation times result in changing multipath errors due to satellite movement [16]. As the delays between the true satellite signals and multipath reflections change, multipath errors can be identified. Long observations at a single point are not always feasible, but this method works well for base stations.

Ionospheric delay is a second source of GPS solution error caused by solar radiation ionizing gases in the atmosphere, creating free electrons that affect the velocity of electromagnetic radiation. Code and navigation data are delayed, while the carrier signals at the same wavelength are sped up by the same amount. An overhead satellite could have a signal delay between one meter at night and five to 15 meters during the

day. A satellite closer to the horizon could have a signal delay of up to 50 meters during the day [18]. The side of the earth facing the sun has maximum ionization, while the side facing away has minimum ionization changing the amount of delay in a signal throughout the day. This diurnal pattern along with non-homogeneity of the atmospheric, make it difficult to predict what the signal delay will be at a given time. Because ionospheric delay is frequency dependent, single frequency (L1 only) receivers rely on modeled ionospheric conditions to remove this error. Dual frequency receivers (L1 and L2) used to collect differential data can remove this error using only the satellite signals, resulting in more accurate solutions.

Tropospheric delay is a third cause of positional error caused by atmospheric refraction of the satellite signals at lower altitudes. Error can vary from two and half meters to 15 meters depending on the elevation angle of the satellites. This type of delay is not frequency dependent, so it cannot be addressed by using a dual frequency receiver. Tropospheric delay must be modeled or corrected using differential GPS.

Timing errors fall into two categories, satellite clock errors and receiver clock errors. Each satellite has a very accurate atomic clock, however, it is not feasible to synchronize all satellite clocks to the exact same time. Instead, the drift of each satellite's clock is monitored and correction information is uploaded to the satellites. When this correction is applied, the time changes from satellite vehicle time to GPS time. GPS satellite signal transmissions must be made in GPS time. After the correction is applied, the satellite's clock should be no more than three or four nanoseconds off of GPS time. A three or four nanosecond timing error would result in a 0.9 to 1.2 meter ranging error. Receiver clock errors are much easier to deal with. There is a time term in the position calculation that allows for the resolution of receiver clock errors as long as at least four satellites are being used to determine the solution.

2.2.8 Differential Correction / RTK

Differential correction is a method to correct for systematic GPS position errors by measuring these errors with a second stationary GPS receiver at a known location

nearby. Correction information can be applied in real time or while post processing the data.

The Federal Aviation Administration developed a real time correction system known as WAAS, the Wide Area Augmentation System, as a means to give pilots position information at a higher level of accuracy at any time than is capable by GPS alone. WAAS is one of several satellite based GPS augmentation systems (SBAS). The WAAS system is made up of a series of GPS monitoring ground stations which determine position correction information for each GPS satellite. This information is then sent to geostationary communications satellites and re-broadcast on the same frequency as the GPS signal. WAAS equipped receivers decode this signal and apply the correction information to the solution as needed.

A second real time correction system provided by the U.S. Coast Guard is the Coast Guard Maritime Differential GPS Service. The Coast Guard is also developing the Nationwide DGPS Service. The Maritime Differential GPS Service is similar to WAAS in that ground stations determine position correction information for each GPS satellite, but rather than sending this correction information to satellites to be re-broadcast, it is sent to broadcasting radiobeacons, where it is broadcast using the Radio Technical Commission for Maritime Services RTCM SC-104 format. GPS receivers capable of using this correction information and within broadcast range of a radiobeacon can decode and apply it as necessary.

A third real time correction system is provided by establishing a real-time kinematic base station in the vicinity of the location data are being collected. This base station is set up on a known point and is equipped with a radio. It determines position correction information for each satellite it “sees” and broadcasts this to the local area. Any GPS receiver in broadcast range of the base station capable of using this correction information can apply it as needed.

2.3 Chaparral

Chaparral is an evergreen shrubland predominantly in the coastal mountains, west of the Sierra Nevada crest in the southern half of California, though its range extends from south-central Oregon to northwestern Mexico. Chaparral is the dominant vegetation type in this geographic area at elevations between 300m and 1500m. Chaparral occurs in areas with shallow, rocky soils, and a climate pattern with winter rain and summer drought. There are nearly 1200 plant species that occur in chaparral, and nearly 500 species that are endemic to this vegetation type [15], resulting in an immense variety of species occurring in any specific plant community. The variety in species makeup is greatly influenced by aspect, soil moisture, localized temperature and weather patterns, and fire history. Many classification systems have been developed to describe the species composition of chaparral communities, one system with more than 60 types. Chaparral is dominated by shrubs including: Ceanothus (*Ceanothus spp.*), Chamise (*Adenostoma fasciculatum*), Flannelbush (*Fremontodendron spp.*), Manzanita (*Arctostaphylos spp.*), Scrub Oak (*Quercus berberidifolia*), and Silktassle (*Garrya spp.*).

Fire plays a very significant ecological role in chaparral vegetation, and is one of its more well known attributes. High density vegetation combined with annual Santa Ana winds and summer drought, create a setting in which fast moving, destructive, crown fires can cover very large areas. Ground fuels appear to play a limited role in chaparral fire [15], and the likelihood of fire does not appear to increase with stand age [28]. Chaparral has a relatively short fire return interval of 70 years [27] or less [28], and has historically burned frequently.

Chapter 3

Methods

3.1 Field Sites

The five fieldwork sites are divided into three locations in the mountains north and east of Los Angeles, California, within the San Bernardino and Los Padres National Forests (Figure 6). The sites were established as part of a joint USDA Forest Service Pacific Northwest Research Station and University of Washington Precision Forestry Cooperative research project for the Joint Fire Science Program (JFSP), and as such were located close to the boundaries of three 2003 wildfires (the Old, Grand Prix, and Piru fires).

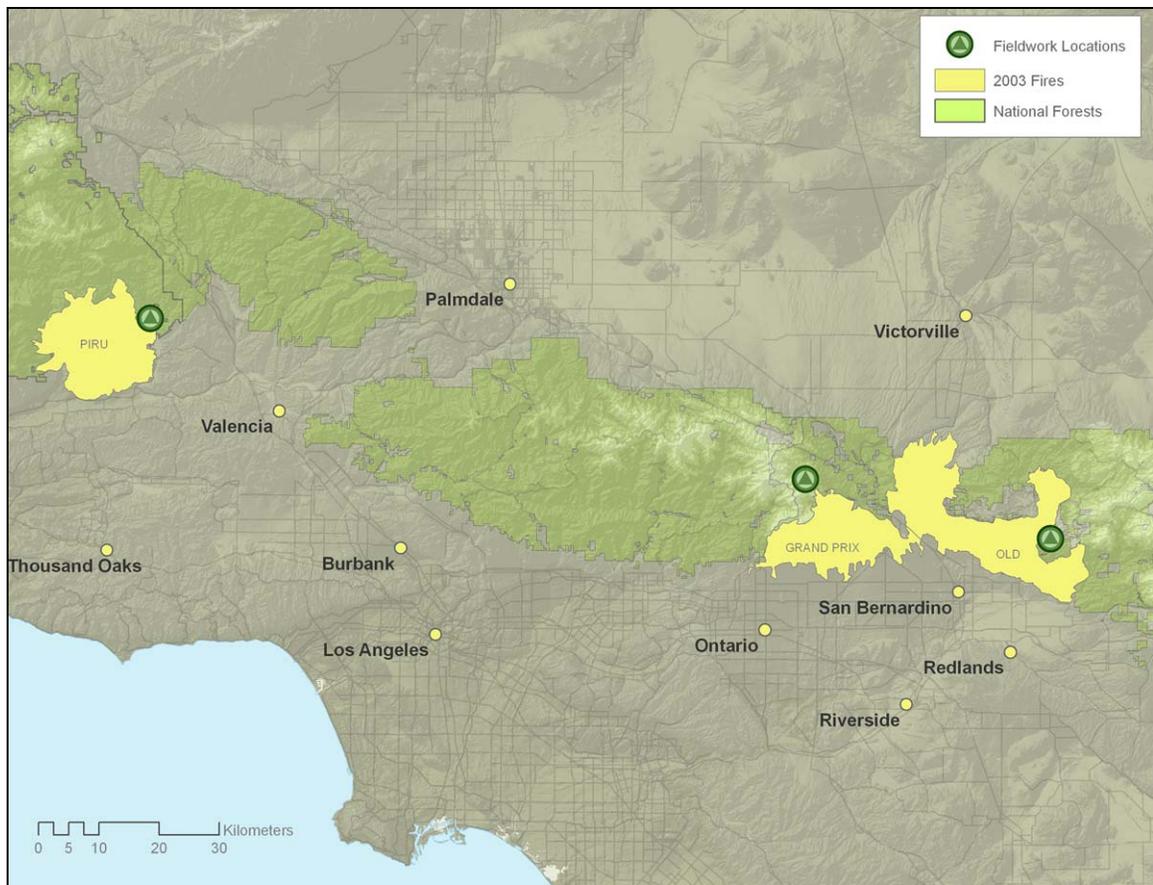


Figure 6 – Map of fieldwork locations near Los Angeles, CA.

The first two sites, Old1 and Old2, are located in the San Bernardino Mountains along California Highway 330 northeast of San Bernardino on generally south facing slopes at elevations between 1300 and 1700 meters. The Grand Prix site is located in the San Gabriel Mountains near Lytle Creek, CA, in a generally northeast facing river valley with elevations between 1500 and 1600 meters. The final two sites, Piru1 and Piru2 are located in the Topatopa Mountains north of Piru, CA, on generally east facing slopes with elevations between 300 and 500 meters. All sites are within the Southern California Mountains and Valleys eco-region as defined by the USDA Forest Service ECOMAP program [26]. The vegetation species encountered during fieldwork at the sites are shown in Table 1. Old 1 and 2 were dominated by Manzanita, Chamise, Canyon Live Oak and Scrub Oak. Grand Prix was dominated by Ceanothus and Flannel Bush. Piru 1 and 2 were dominated by Manzanita and Chamise, but also included a great deal of California Buckwheat, Golden Yarrow, and Oak.

Table 1 – Vegetation species encountered at sites.

Old 1 and 2	Chamise, Deer Brush, Dogwood, Parry Manzanita, White Leaf Manzanita, Yucca, Canyon Live Oak, California Scrub Oak, California Black Oak, Coulter Pine, Ponderosa Pine, White Fir
Grand Prix	California Buckwheat, Ceanothus, Flannel Bush, Sage Brush, Service Berry, Silktassel, Yucca, Canyon Live Oak, Coast Live Oak, Engelmann Oak, Cedar, Ponderosa Pine, Big Cone Douglas Fir, White Fir
Piru 1 and 2	Buckbrush, California Buckwheat, Chamise, Emory Baccharis, Grass, Lupine, Pink Bract Manzanita, Milkweed, Thistle, Black Sage, Purple Sage, Redberry, Redshank, Silktassel, Snowdrop Bush, Golden Yarrow, California Scrub Oak, Canyon Live Oak, Toyon, Hollyleaf Cherry

3.2 GPS Equipment

Four Javad Navigation Systems Maxor-GGDT GNSS integrated antenna/receivers were used for this thesis (Figure 7). They are 20 channel, dual frequency, GPS/GLONASS receivers using the MarAnt+ antenna and GGD-112T receiver board. The MarAnt+ is a groundplane antenna shielding it from multipath signals below the antenna [6]. The GGD-112T uses Javad's Advanced Multipath

Mitigation, a proprietary signal processing technique to reduce the effect of multipath errors from above the receiver [3]. The receivers were purchased as two real-time kinematic (RTK) pairs, and included internal Pacific Crest Positioning Data Link (PDL) UHF radio modem boards. An external Pacific Crest radio was used at the base station to broadcast correction information from the base receiver. One RTK base receiver was set up as a true base station and was used in cooperation with the two RTK rovers. The second RTK base receiver was used independently (neither broadcasting nor receiving real time correction information) to measure control GPS points and Non-RTK vegetation GPS points. These data types will be discussed below in Section 3.6, Field Data Collection.



Figure 7 – Javad Navigation Systems Maxor-GGDT GNSS receiver (source Javad Navigation Systems, Inc.).

Table 2 provides the specifications for the equipment used in this thesis [8]. The GPS antennas were above the top of vegetation and had a clear sky view while collecting data.

Table 2 – Javad Accuracy Specifications (source Javad Navigation Systems, Inc.).

Component	Details
Tracking Specifications	
Standard Channels	Euro-112 GGD - 20 channels (GG,GD,GGD), GPS/GLONASS L1, GPS L1/L2, GPS/GLONASS L1/L2, WAAS/EGNOS
Tracked Signals	GPS/GLONASS, L1/L2, C/A and P-Code and Carrier, WAAS/EGNOS
Survey Mode	
	Static Kinematic RTK (Real-time Kinematic) DGPS (Differential GPS)
Survey Accuracy	
Static, Fast Static	For L1 + L2 - H: 3mm + 1ppm x D V: 5mm + 1.4ppm x D
RTK	For L1 + L2 - H: 10mm + 1.5ppm x D V: 20mm + 1.5ppm x D
Cold Start	< 60 sec
Warm Start	< 10 sec
Reacquisition	< 1 sec

3.3 LIDAR Data

The LIDAR data used for this analysis were collected by Watershed Sciences, Inc. (<http://www.watershedsciences.com/>) of Corvallis, OR, on August 17th, 2005, using an Optech ALTM 3100 LIDAR system [9]. The system parameters for the data collection as reported by the data provider [11] are given in Table 3.

Table 3 – LIDAR collection parameters (source Watershed Sciences, Inc.).

LIDAR Collection Parameters	
scan angle:	18° (9° from Nadir)
altitude above ground:	1000m terrain following
pulse rate:	71 kHz
scan width:	≈ 316m
flight line overlap:	100% (50% side-lap)
point density:	at least 4 per m ²

Accuracy specifications for the LIDAR system, as determined by Optech, Inc., are provided in Table 4. At a flying altitude of 1000 meters, horizontal accuracy is ± 0.5 meters and vertical accuracy is approximately ± 0.15 meters for open hard surfaces.

Table 4 – LIDAR system accuracy specifications (source Optech, Inc.).

ALTM 3100 Specifications	
operating altitude:	80 - 3,500 m nominal
horizontal accuracy:	1/2,000 x altitude; 1 sigma
elevation accuracy:	<15 cm at 1.2 km; 1 sigma <25 cm at 2.0 km; 1 sigma <35 cm at 3.0 km; 1 sigma

3.4 LIDAR Quality Assessment

In an initial examination of the LIDAR data, there appeared to be problems with vertical data alignment between flight lines for each site. For the same location, where data from overlapping flight lines should have the same elevation, vertical differences were detected. In a worst case scenario, this difference could result in the data from one of the flight lines being removed when creating a digital terrain model. The practical effect is the reduction by half of the data density. To quantify this difference error, a method of quickly comparing data from separate flight lines, developed by Damir Latypov [24], was implemented for one of the five test sites, Piru 2. It was found that there was a vertical difference of seven to 13 centimeters between the flight lines. Watershed Sciences reprocessed the dataset, and the vertical difference was reduced to three to five centimeters.

The swath width of the data was also examined in the quality assessment. According to the formula presented in Section 2.1 the swath width should be approximately 317 meters, while the specifications provided by Watershed Sciences, claim that it is approximately 316 meters. Visual inspection of the data showed that the swath widths were approximately 320 meters.

Finally, LIDAR point density for the data was of interest. It was specified in the contract with Watershed Sciences that the data should have at least four LIDAR returns per square meter. Table 5 was calculated from the data. While the average point densities were at specification, the sites had between five and 24 percent of their area below specified density.

Table 5 – Calculated LIDAR point densities.

Site	Avg. Dens.	Min. Pt. Dens.	Max. Pt. Dens.	% Area LT 4	% Area GT 4
Old1	7.12	0	43	10.60	89.40
Old2	8.38	0	54	10.12	89.88
GP1	9.11	0	337	10.04	89.96
Piru1	7.55	0	71	5.08	94.92
Piru2	5.33	0	30	24.47	75.53

The map below (Figure 8) shows the high variability of point density within one of the five sites, Old 1. Point density was similarly variable for the other four sites. Orange and green areas are below specified density. The aircraft flight lines were horizontal across the image. The horizontal lines visible in the data are flight line edges, while the vertical lines are changes in point density caused by changes in aircraft speed or aircraft altitude.

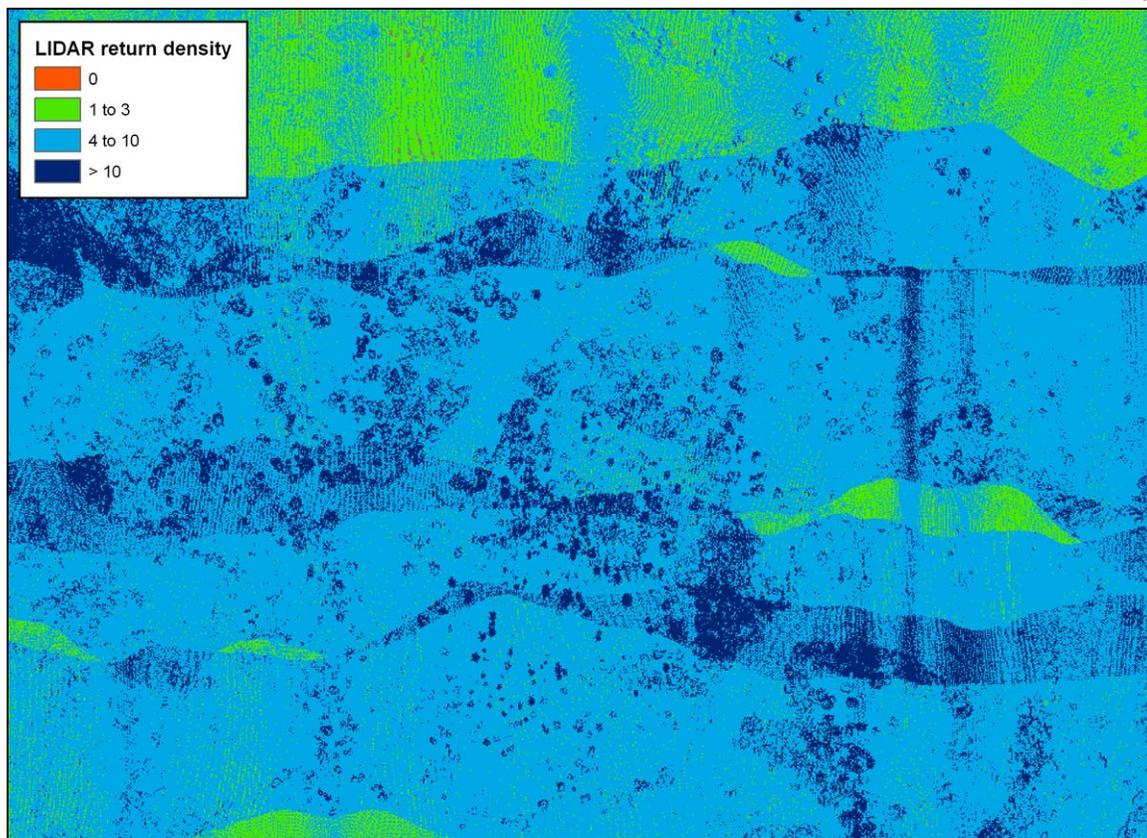


Figure 8 – Map of LIDAR point density for site Old 1.

3.5 Determining Field Data Collection Sites

3.5.1 Vegetation Transects

The five sites contained a total area of more than 500 hectares (Table 6), with a variety of vegetation, including grassland, forest, and various ages and mixes of chaparral.

Table 6 – Area of the sites in hectares.

Site	Area (Hectares)
Old 1	131
Old 2	43
Grand Prix	188
Piru 1	71
Piru 2	68

It was important to determine areas within the site that were appropriate for the scope of this project, and limit data collection to those areas. LIDAR data were available before the field work, and although, the purpose of this thesis is to determine the accuracy of the LIDAR data, they could provide an estimate of site conditions, useful for determining where to collect field data.

Characteristics of the sites that were identified as having an effect on LIDAR were slope, vegetation height, and vegetation density. It was also decided that fieldwork should be limited to areas having only first returns. It is believed that areas having vegetation cover and only first returns are the least likely to have ground returns. In these areas, this single return is hypothesized to be the energy reflected from both low vegetation and the ground.

Geographic Information Systems (GIS) techniques were used to select field sites. The overall approach was to develop slope, vegetation height, and vegetation density rasters from the LIDAR data, then combine these to locate field sites.

The first step was to create a digital terrain model (DTM) from the LIDAR data. This was done in Fusion LIDAR processing software (version 2.51) developed by Robert McGaughey of the USDA Forest Service Pacific Northwest Research Station. Specifically, two command line utilities, `groundfilter.exe` and `GridSurfaceCreate.exe`, were used to first filter the data, and second build a DTM from the filtered points.

The ground filtering methodology is based on work done by Kraus and Pfeifer [23] and described in a paper by Andersen et al. [2]. Using the average height of all returns in a cell, an initial ground surface is created. For each LIDAR return, the distance between the return's elevation and the elevation of the ground model at the return's coordinates is calculated. Ground returns will likely be below the surface and so have large negative distances, while vegetation returns will likely be close to or above the surface and so will have small negative or positive distances. Kraus and Pfeifer developed a weighting method involving four parameters: a , b , g , and w . The parameters a and b were determined to be 1 and 4 respectively by Kraus and Pfeifer, and parameters g and w are determined by the user. The parameter g is a distance threshold below which returns get the maximum weight of one. The parameter w , is the width of the envelope of

possible distances that will be considered ground. Returns with distances in this envelope will get a weight determined by this formula:

$$p_i = \frac{1}{1 + [a(v_i - g)^b]}$$

where p_i is the weight, and v_i is the distance. Returns with distances above the ground envelope, will be given a weight of zero. For example if g is set to zero, and w is set to 0.5, then a return with a distance less than zero will get a weight of one, a return with a distance between zero and 0.5 will get a weight using the formula above, and a return with a distance greater than 0.5 will get a weight of zero. Using these weights a new ground surface is created. The weights “pull” the new ground surface towards returns that are below the initial surface. This process is repeated as many times as the user chooses. After all repetitions are completed, the returns below or within 15 centimeters of the final ground surface are considered ground. Table 7 provides the parameters used to filter the LIDAR for this thesis work. Visual inspection determined that a smaller cell size produced better results at the Piru 2 site.

Table 7 – Parameters used in filtering LIDAR ground returns.

Site	g parameter	a parameter	cell size	num. iterations
other sites	-0.1	0.25	3	5
Piru2	-0.1	0.25	1.5	5

Once filtered ground returns were created, they were used to create 1m DTMs. These DTMs are the basis for the rest of the field site determination work.

Slope (Figure 9) was calculated using Spatial Analyst in ArcGIS 9.2 [13]. Because slopes above 50 degrees, were inaccessible, sites were limited to locations with a slope less than 50 degrees (Figure 10).

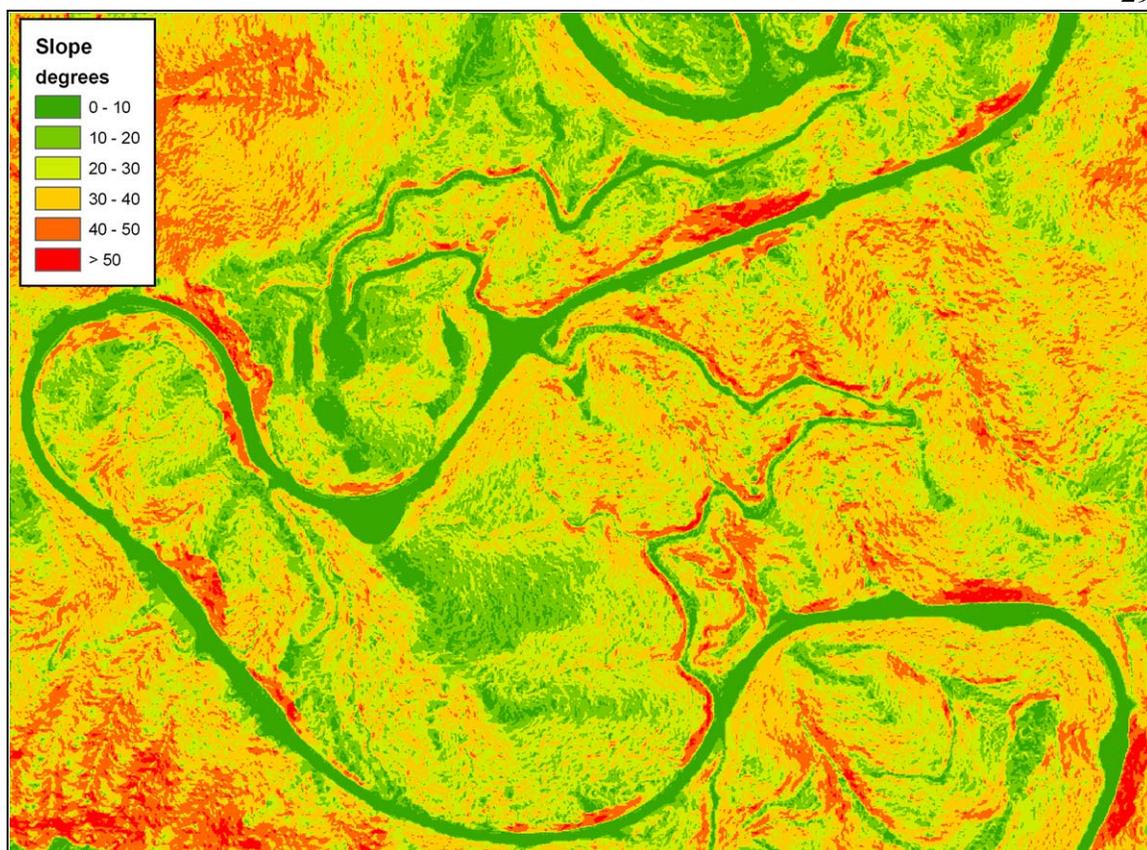


Figure 9 – Example of determining slope for site Old 1.

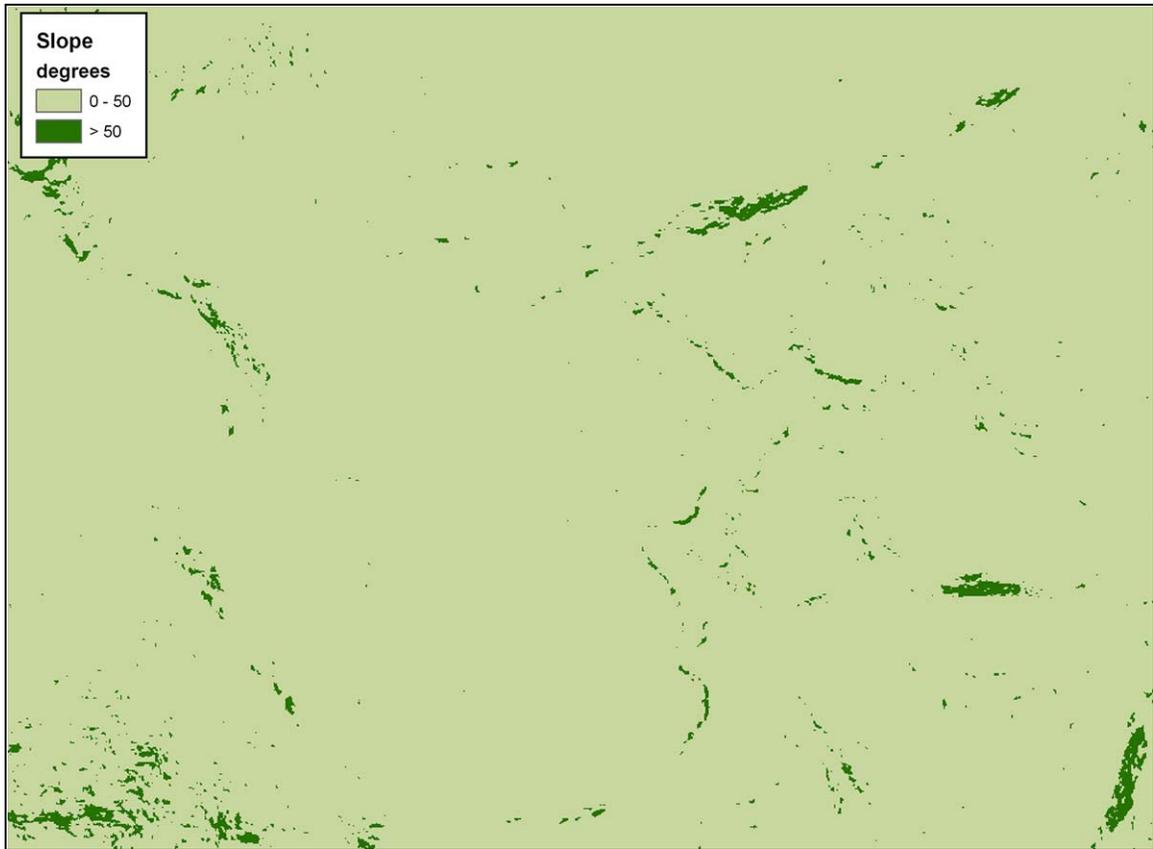


Figure 10 – Example of determining slope for site Old 1.

Second, the height of the vegetation needed to be estimated. The first step is to create a surface representing the top of the vegetation, a canopy surface model. The second step is to subtract the DTM from the canopy surface model to produce vegetation height. CanopyModel.exe is another command line utility included with Fusion. As the name implies, it is used to create canopy surface models from LIDAR data. This is done by dividing the area covered by LIDAR into a grid of cells and setting each cell elevation to the elevation of the highest LIDAR return in each cell. The best vegetation height results will be produced if the DTM and canopy surface model have the same spatial extent and cell size. A canopy surface model with one meter cells was created for each site, and the DTM elevation subtracted from each cell (Figure 11). Areas with vegetation height between zero and three meters were selected (Figure 12).

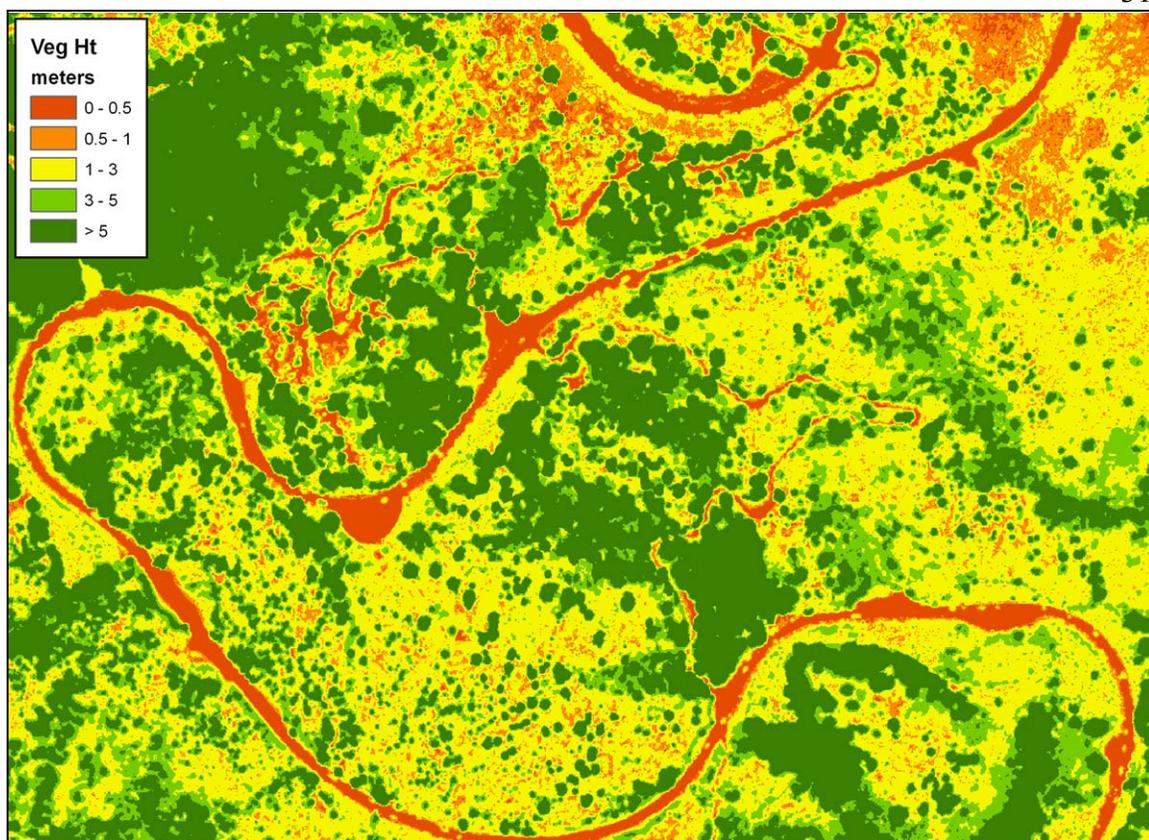


Figure 11 – Example of determining vegetation height for site Old 1.

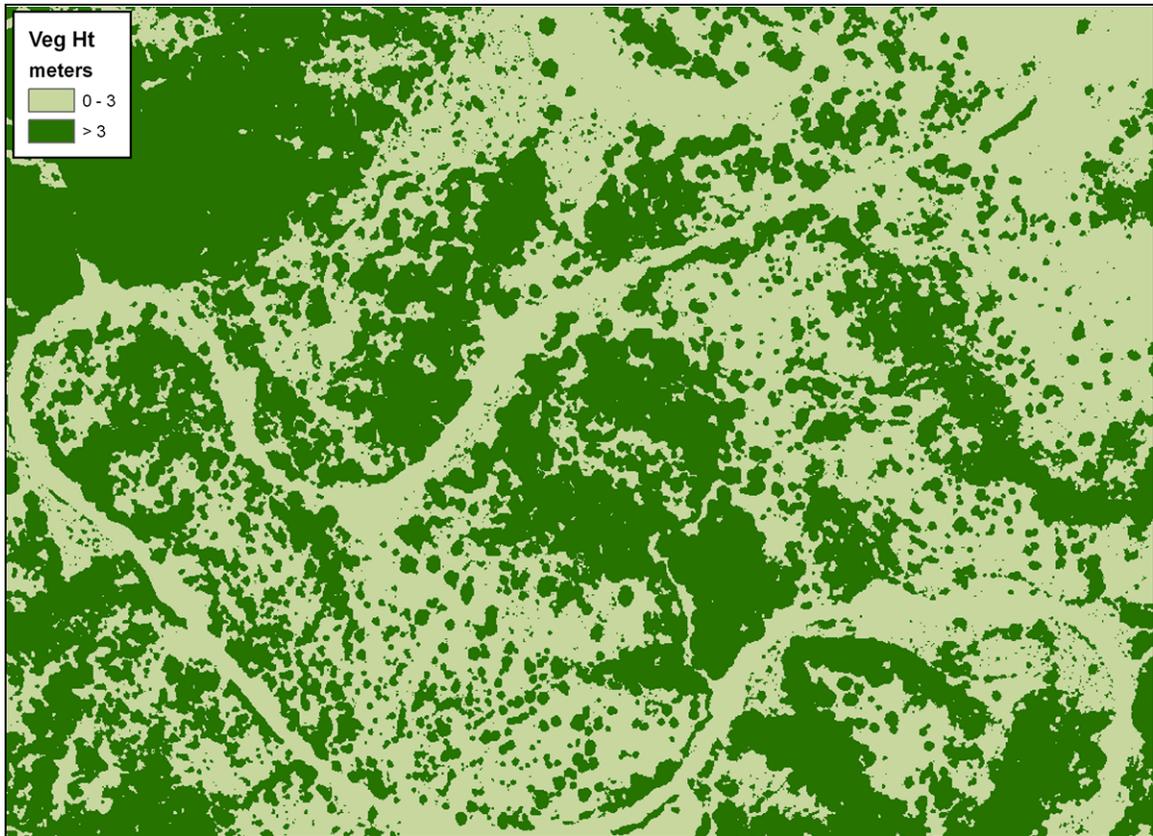


Figure 12 – Example of determining vegetation height for site Old 1.

Third, the collection areas needed to be limited to locations having only a first return for a pulse. The LIDAR data were provided by the vendor in the LAS format [10]. This format requires that a return number be provided for each LIDAR return in the file. A program was written in IDL [7] to read the LAS formatted LIDAR data for each site and separate the returns by return number. These data contained up to four returns per pulse. These separated returns were then gridded with one meter cells, and cells containing only first returns were determined (Figure 13).

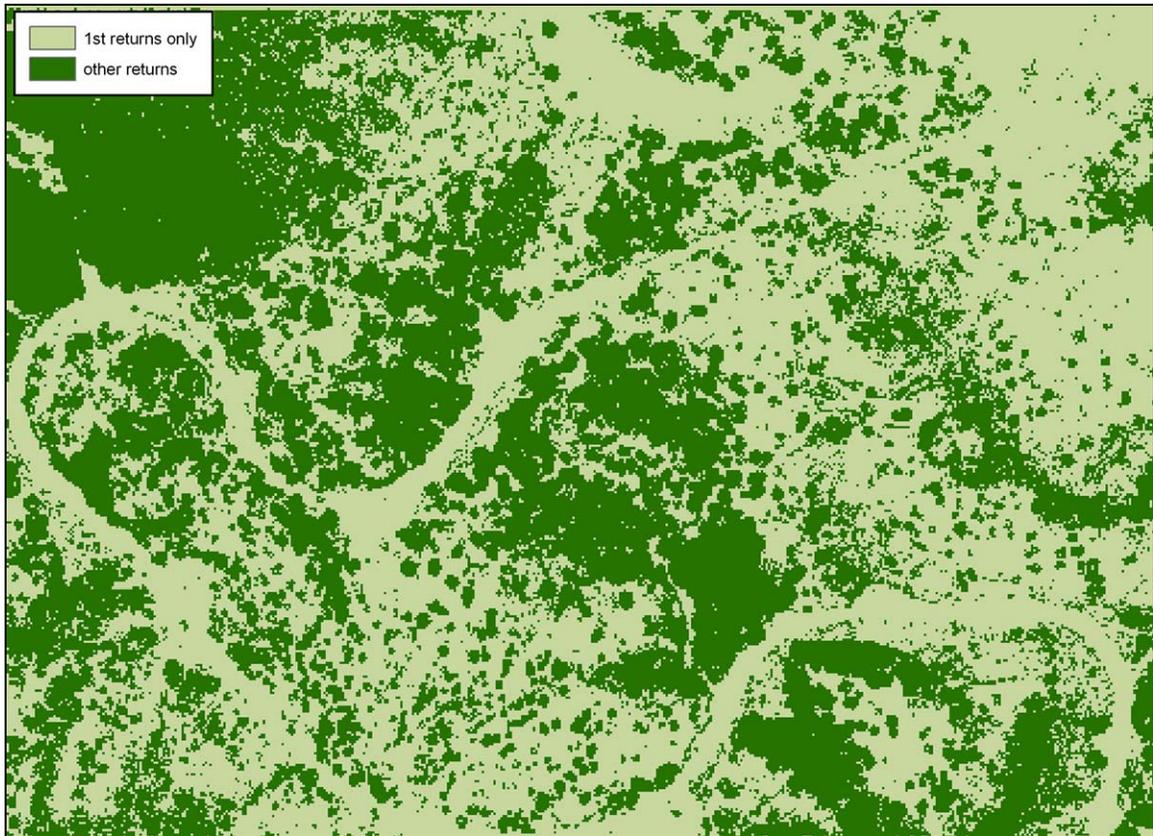


Figure 13 – Example of areas with first returns only (light green) and other returns (dark green) for Old 1.

Using the raster calculator in Spatial Analyst [13], the above three datasets were combined to determine areas in which data should be collected (Figure 14). These collection areas were divided into three subcategories of density: low, medium, and high, as estimated using a percent cover statistic (Figure 15).

In previous work with LIDAR in forests, percent cover has been called mean canopy cover density [30] [31], tree cover [32] (both are the number of canopy returns in a cell divided by the total number of returns in a cell), and canopy density metric (percentage of first returns in the canopy in each cell) [1]. In work with LIDAR in non-forest vegetation, this metric has been called the percentage of vegetative cover [38] (the number of non-ground returns in a cell divided by the total number of returns in that cell), and a related metric called percent bare soil (the number of ground returns divided by the total number of returns in a cell) was used in sagebrush [29].

For the purposes of this thesis, because the areas of interest are limited to areas with single returns, and because in short vegetation it can be difficult to separate ground returns from vegetation returns, percent cover is defined as the number of returns in a cell with height greater than or equal to one meter divided by the total number of returns in that cell. Again, a program was written in IDL to place all returns in a cell and determine their height above the DTM, and from this, calculate percent cover. A low density cell had a percent cover of 10 percent or less, a medium density cell had a percent cover between 10 and 50 percent, and high density cell had a percent cover greater than 50 percent. Attempts were made to sample in all three densities.

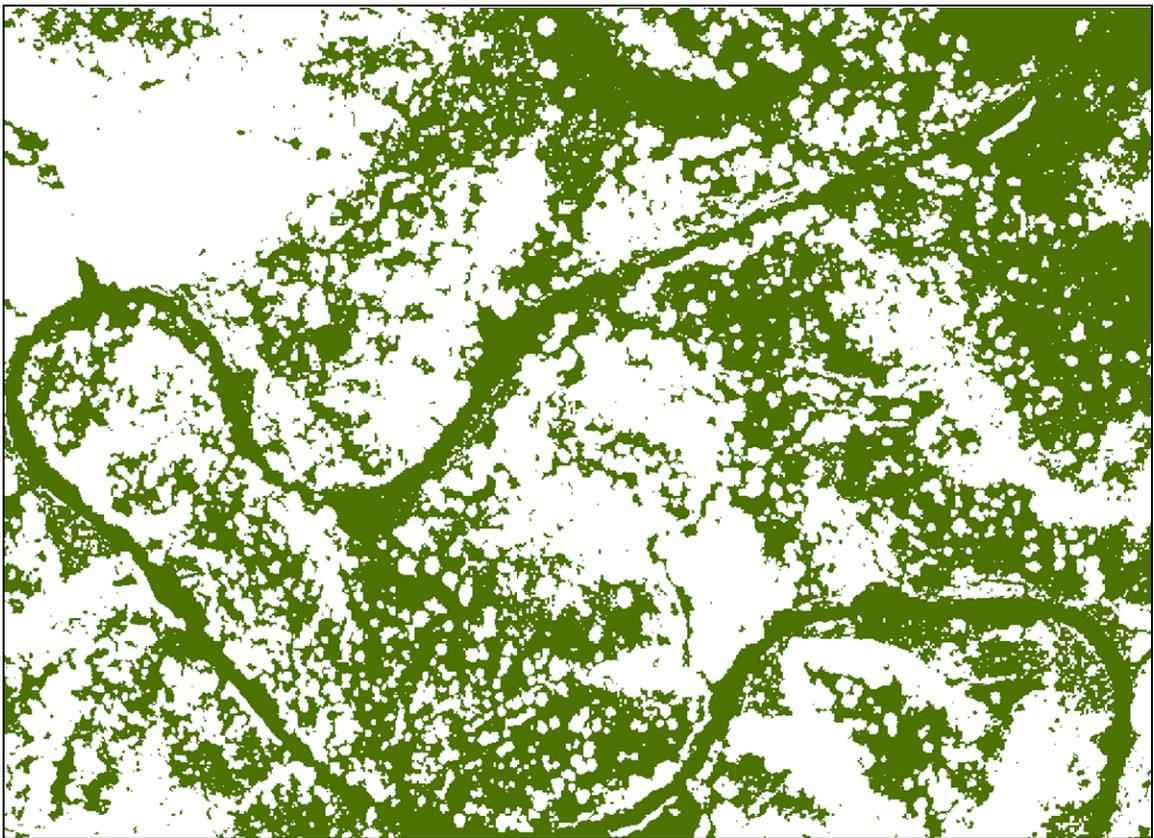


Figure 14 – Example of areas fitting collection criteria (green) for site Old 1.

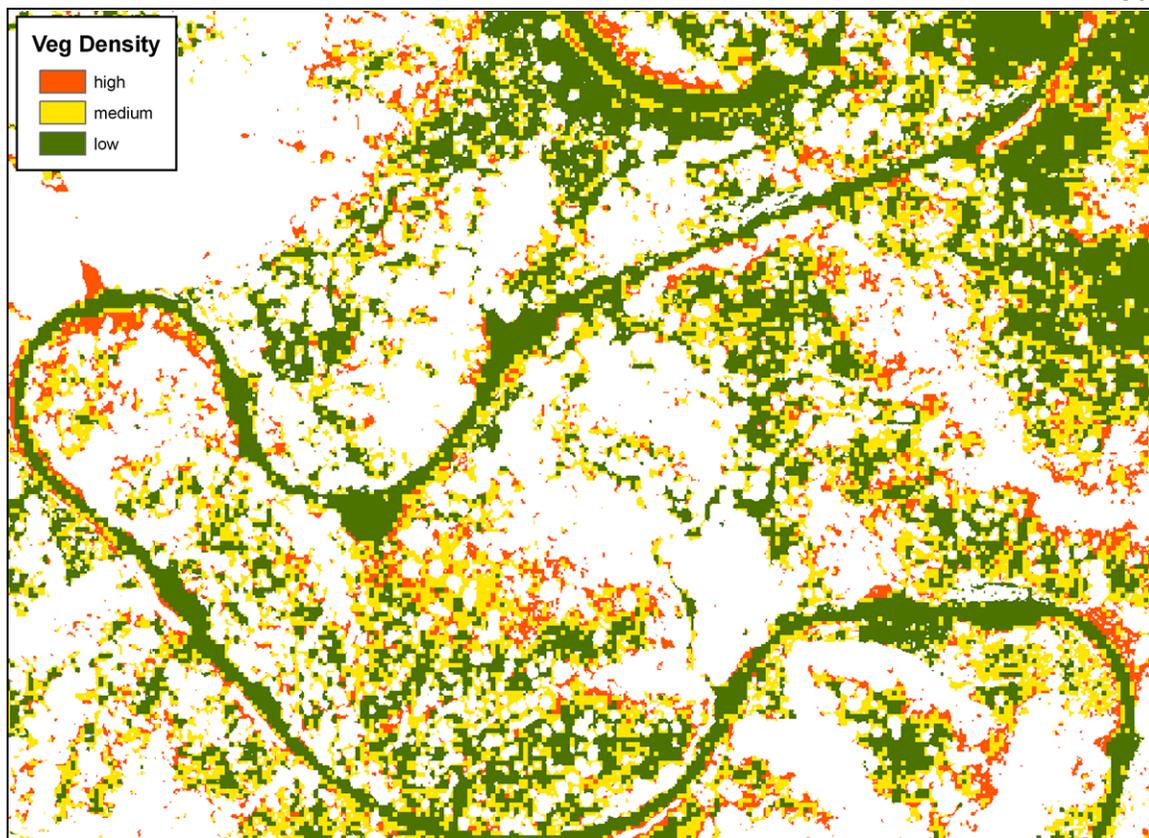


Figure 15 – Example of areas fitting collection criteria colored by vegetation density for site Old 1.

Finally, polygons were manually drawn around areas deemed suitable for field data collection based on visual inspection of the results above (Figure 16). Three band red, green, blue (RGB) orthophotos were available for all sites (Figure 17), and a visual comparison of the data collection polygons with these orthophotos confirmed vegetation cover. These data collection polygons were loaded into the data loggers used to collect the field data, and GPS transects were planned to fall within these areas.

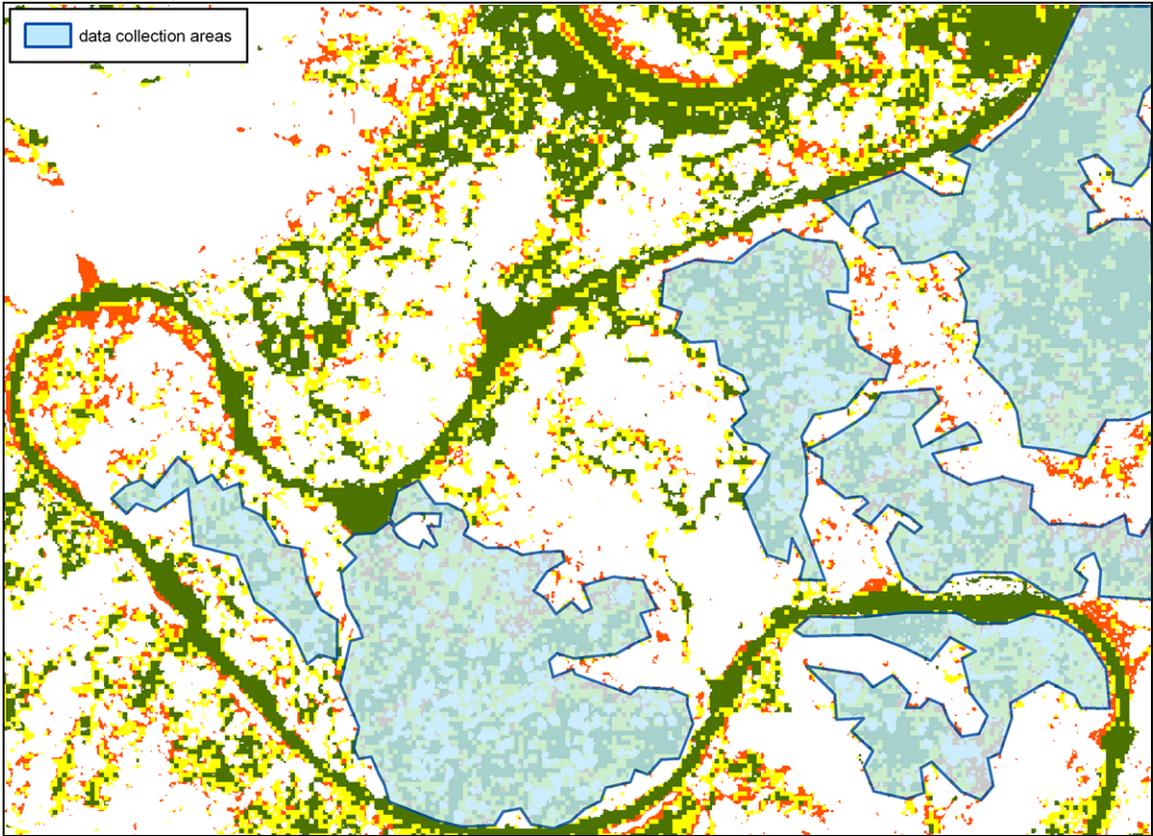


Figure 16 – Example of data collection area polygons for site Old 1.

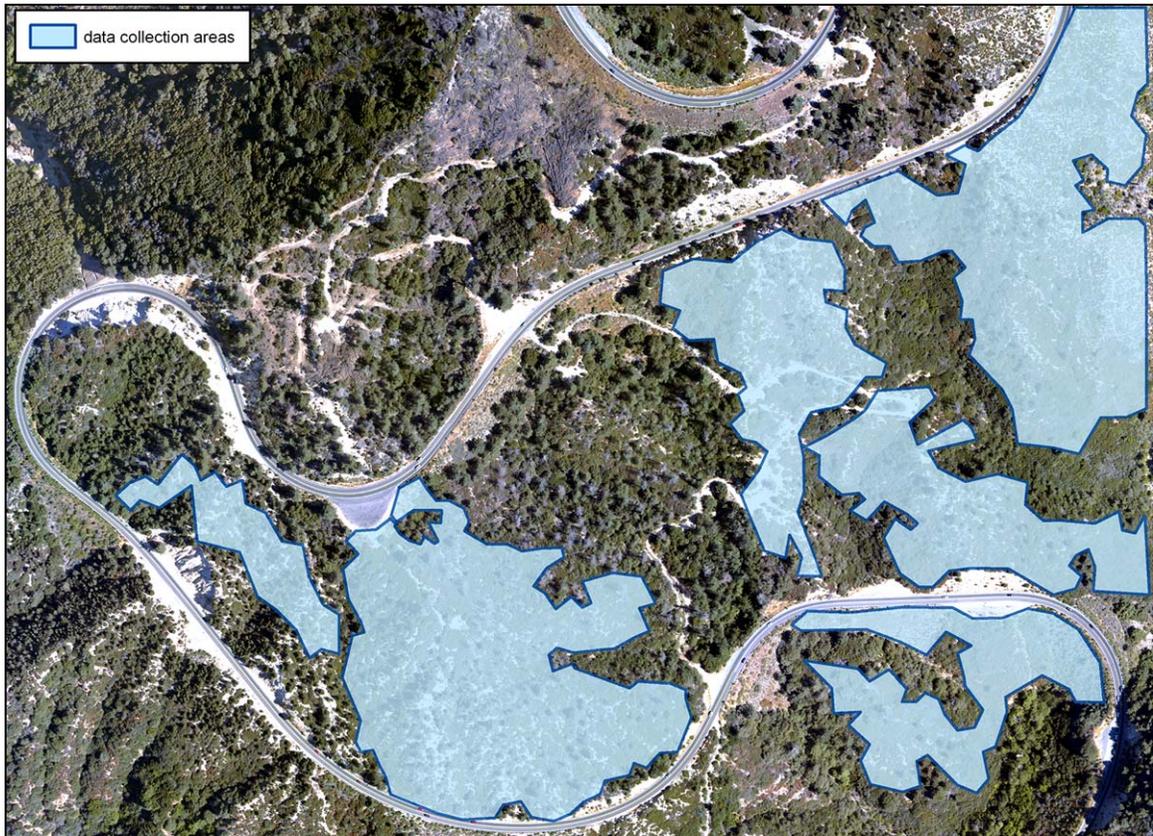


Figure 17 – Example of data collection area polygons overlaid on orthophoto for site Old 1.

3.5.2 Control Points

The LIDAR data were collected in 2005 and corrected by Watershed Sciences using proprietary software. The GPS data were collected in 2007, using different equipment, different satellite constellations, and processed using different proprietary software. It was therefore possible, even likely, that there would be systematic elevation differences between the two datasets. If these existed and were not detected and addressed, all results would be biased.

It is easiest to build an accurate ground surface model from LIDAR in areas that are flat. Ground returns in a flat location will have consistent elevation over a large area. Areas with bare ground (no vegetation) are the most likely to have laser reflections from the ground. Therefore, in bare, flat areas, there will be many ground returns at a consistent elevation, which will result in the best possible ground surface from the

LIDAR dataset. If GPS data could be collected at bare, flat points, it would be possible to detect systematic differences between the LIDAR and GPS data, and remove them.

To this end, it was important to pick locations in the LIDAR data that were both flat and free of vegetation, and to plan to collect control GPS points at these locations. The DTM was again used, this time to find areas with a slope less than seven degrees. The percent cover vegetation density metric was used to find area with low (less than 10 percent) percent cover. Points that met both of these criteria were selected for control point data collection. Control points were spread throughout each site (Figure 18).

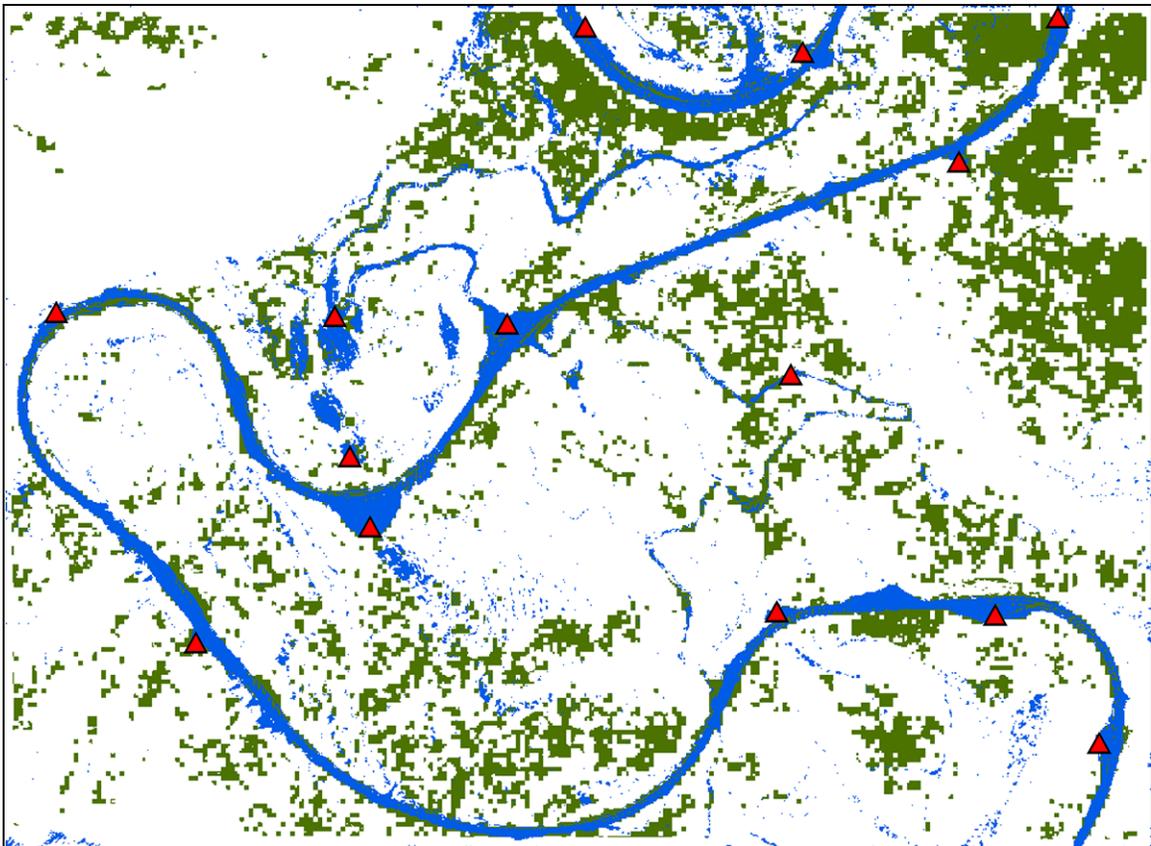


Figure 18 – Example of flat areas (blue), bare areas (green), control points (red) at site Old 1.

3.6 Field Data Collection

There were four types of GPS data collected while working in the field: local base station GPS points, control GPS points, RTK vegetation GPS points, and non-RTK vegetation GPS points.

3.6.1 Local Base Station GPS Points

Local base station GPS points were collected to post-process the rover positions, and provide real time correction information. Real time correction information was central to the data collection methods used in this thesis, guaranteeing that each position had a fixed carrier solution as it was collected. Typically, GPS base stations would be set up on a survey monument with known coordinates. This was not an option at these field sites, so the local GPS base stations were set up where site conditions permitted: at locations with open sky view, at locations with a sky view similar to that “seen” by the rovers, and at locations with few line-of-sight obstructions between the base and the rovers (Figure 19). One of the RTK receivers, connected to the external Pacific Crest radio, was used to collect each local base station GPS point. Real time correction data were transmitted throughout the data collection period. Data collection at local base station GPS points was begun before any rover data collection started, and was not stopped until after all rover data collection finished. The receiver was placed at the top of a plumb tripod at a height greater than three meters, with the radio antenna attached to the side of the tripod. Position data were collected at one second epochs.

Local base station GPS positions were post-processed using permanent Continuously Operating Reference Stations (CORS), to produce their final position, as described in Section 4.1.1.



Figure 19 – Example local base station GPS, with receiver and RTK radio antenna.

3.6.2 Control GPS Points

Control GPS points were collected in order to determine if there was a systematic offset between the LIDAR data flown in 2005 and the GPS data collected for this project nearly two years later. The locations for control GPS points at each site were predetermined using the methods in Section 3.5.2, and were loaded into the data loggers used to collect the field data. The operator collecting control points navigated to these locations using the coordinates in the data logger (Figure 20) and found an open flat location as close to the coordinates as possible.



Figure 20 – Example control GPS point.

For each of the five sites, the non-RTK base receiver was set up at each control point on a plumb height pole, and position data were collected for 10 minutes, with 1 second epochs.

3.6.3 RTK Vegetation GPS Points

Vegetation GPS points were collected in each of the five sites in collection areas selected using the methodology in Section 3.5.1.



Figure 21 – RTK GPS rover (left) and non-RTK GPS rover (right).

Using an RTK enabled rover on top of a three meter height pole, a data point was collected approximately every three meters along each transect (Figure 21). Each data point was occupied for 15 one second epochs, with the rover set to accept only fixed solution epochs. Transects were established by picking a compass bearing from a starting point, and maintaining that bearing through the collection area. Field crews attempted to navigate each transect regardless of ground conditions (not avoiding dense vegetation or other obstacles). Vegetation height was measured to the nearest five centimeters by reading where the tallest vegetation contacted the side of the height pole (Figure 22). If vegetation was over three meters, a position was taken with the height recorded as greater than three meters (gt3). A bubble level on the side of each height pole was used to keep the pole plumb for the duration of the occupation.



Figure 22 – Height markings on GPS pole.

Vegetation density was measured by observing the one meter diameter circle centered on the height pole, and deciding how much vegetation was within the circle. The following criteria were used for the density metric.

Table 8 – Vegetation point density criteria.

Density	Amount of Observed Vegetation
0	none or nearly none
1	some
2	a lot

Data were recorded into the description field for each point in the GPS field data logger software, Fieldview, using this format: height, density. For example, if a GPS point had a vegetation height of 1.75 meters and some vegetation, the data would be entered in the comment field for that point as: 1.75,1.

Occupation time for RTK vegetation points was intentionally short, as it allowed a much greater number of data points to be collected. It is reasonable to trust the accuracy of the GPS position data even with short occupations because: the GPS receivers have ground plane antennas and were above the vegetation reducing the likelihood of multipath errors, there were few terrain obstacles blocking the sky guaranteeing a large number of satellites available for calculating solutions, and most importantly, the equipment collected only fixed solution data. A discussion of the number

of satellites used, PDOP values, and RMS errors for each GPS point follows below in Section 4.1.2.

3.6.4 Non-RTK Vegetation GPS Points

When the non-RTK base receiver was not collecting control points, it was used to collect vegetation points. Because it could not receive real time correction information, it was left on each position for 60 one second epochs. Vegetation height and density were recorded for these points in the same manner as for the RTK Vegetation GPS points. A discussion of the number of satellites used, PDOP values, and RMS errors for each GPS point follows below in Section 4.1.2.

3.7 Vegetation Growth

One of the questions of this thesis is to determine if vegetation height in chaparral has an effect on laser penetration to the ground, and in turn on DTM accuracy. To this end, vegetation heights were measured at each vegetation GPS point. These will be referred to as 2007 vegetation heights. In order to perform a comparison between the LIDAR vegetation heights and the GPS vegetation heights, an estimate of vegetation growth over the two year period between these datasets needed to be developed.

In the summer of 2005, the Precision Forestry Cooperative sent a field crew to the Piru 1 and Piru 2 sites to measure vegetation as part of the Joint Fire Science Program project. There were 17 plots measured that summer, of which seven were in either forest or grassland vegetation types, and ten were chaparral. Each of these plots was a 20 meter square divided into two meter grid cells. At each of the 121 cell intersections, vegetation height was measured (Figure 23). Using these points, a 2005, field-measured, canopy surface model was created.

As part of the field work for this thesis, 25 RTK vegetation GPS points were measured inside three of the ten chaparral plots, for a total of 75 points. The elevation of the 2005 canopy surface model at each of these points was calculated using bilinear interpolation. The difference between these interpolated elevations and the 2007

vegetation heights, provided an estimate of growth (Figure 24). This growth estimate was subtracted from the 2007 heights to obtain an estimate of vegetation height at each point in 2005, which will be referred to as 2005 vegetation heights.

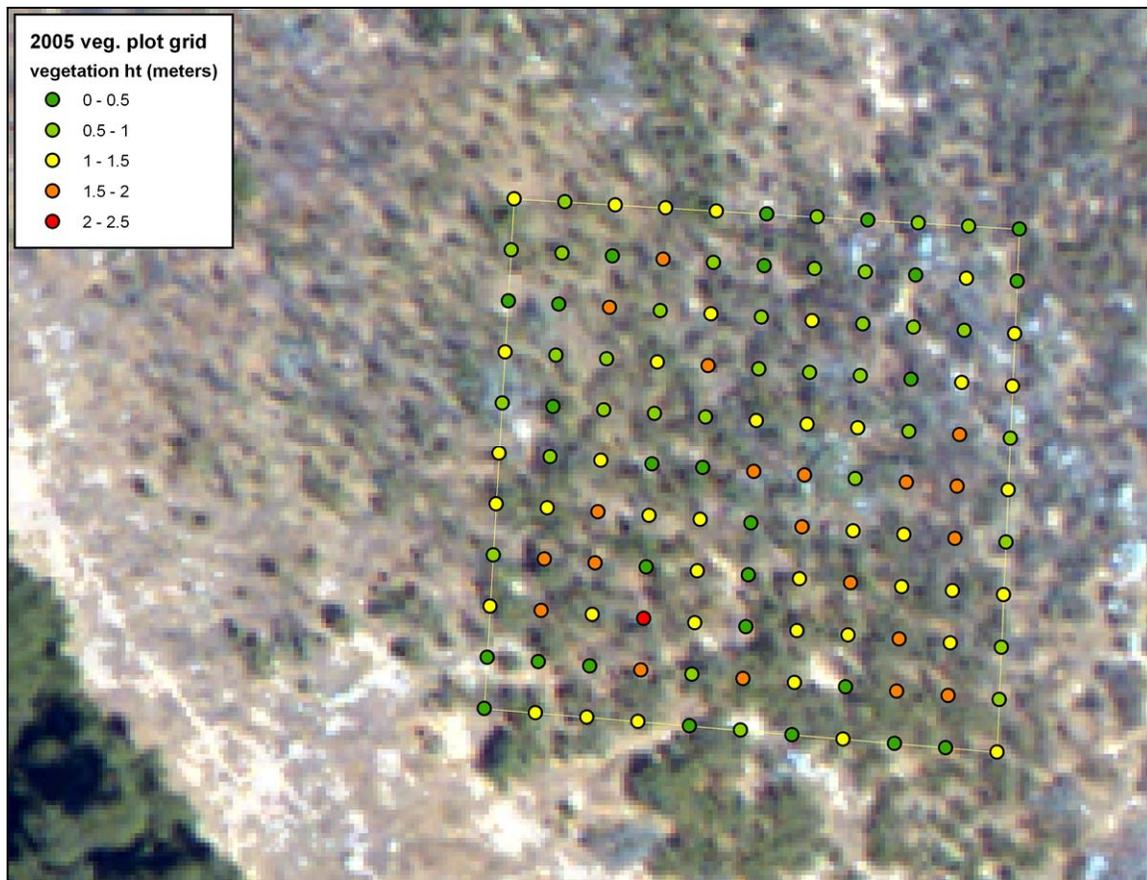


Figure 23 – One of three plots used for measuring growth, 2005 vegetation height.

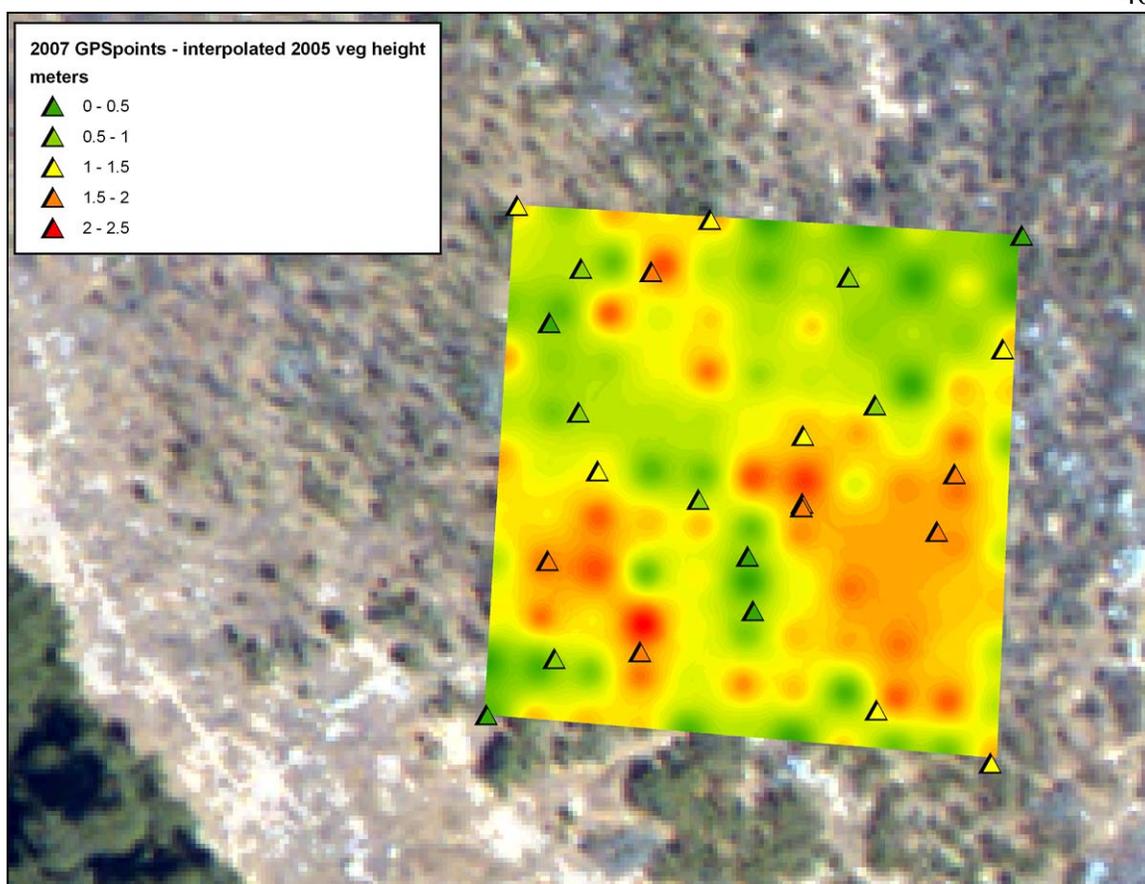


Figure 24 – 2007 GPS points with interpolated 2005 vegetation heights.

3.8 LIDAR Ground Model

The central point of this thesis is to compare LIDAR-derived ground elevations at each GPS point to the GPS measured ground elevations. In order to do this, ground elevations must be calculated from the LIDAR point cloud.

One method to achieve this goal is to create a DTM, as was described above, in Section 3.5.1. There is a great deal of filtering and processing done to the LIDAR data to create a DTM, introducing many unknowns into the final elevation for any single point. To avoid these unknowns, this method was not chosen. A second method, sometimes referred to as the columnar minimum method, involves finding the lowest LIDAR point within a specified radius of each GPS point, and setting its elevation as the ground elevation. On slopes this becomes problematic as it is difficult to determine an

appropriate search radius. A large search radius will likely include LIDAR points down slope (Figure 25 a), while a small search radius could result in the column containing no true ground returns (Figure 25 d). A variety of other possibilities could occur as well (Figure 25 b and c). Deciding which LIDAR return is the most suitable representation of the ground is very difficult.

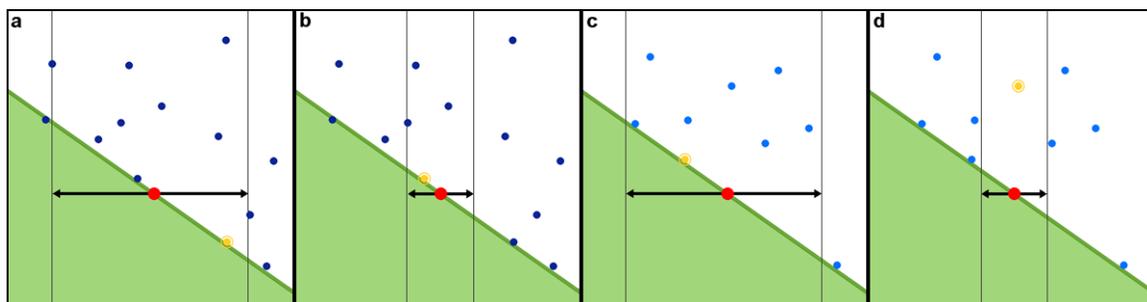


Figure 25 – Columnar minimum selection result possibilities. GPS point (red), LIDAR points (blue), lowest LIDAR point in column (yellow).

To avoid having to select a single LIDAR return from the LIDAR data cloud, and to avoid using a DTM, an automated method was developed for this thesis to build a localized Triangular Irregular Network (TIN) ground model around each GPS point. The minimum number of points required to build a simple surface is three. Dividing up the LIDAR returns around a GPS point into three equal bins and using the lowest point in each bin would make possible the building of surface from the localized LIDAR points using the lowest possible data. This is called the Triangle Method. If there are no LIDAR points on the ground around the GPS, then this method will produce a ground model above the true ground height, which is the desired behavior.

All LIDAR returns within a specified distance of the GPS point must first be selected. Multiple selection radii were tested for this thesis. The process of choosing the best selection radius is discussed in Section 4.3.3. These selected returns were divided up geographically, using their azimuth from the GPS ground point, into three bins (Figure 26). There are 360 degrees of rotation around the GPS point, so each bin was 120 degrees wide. If the azimuth to the LIDAR return was within the degree range of a bin, it was placed in that bin.

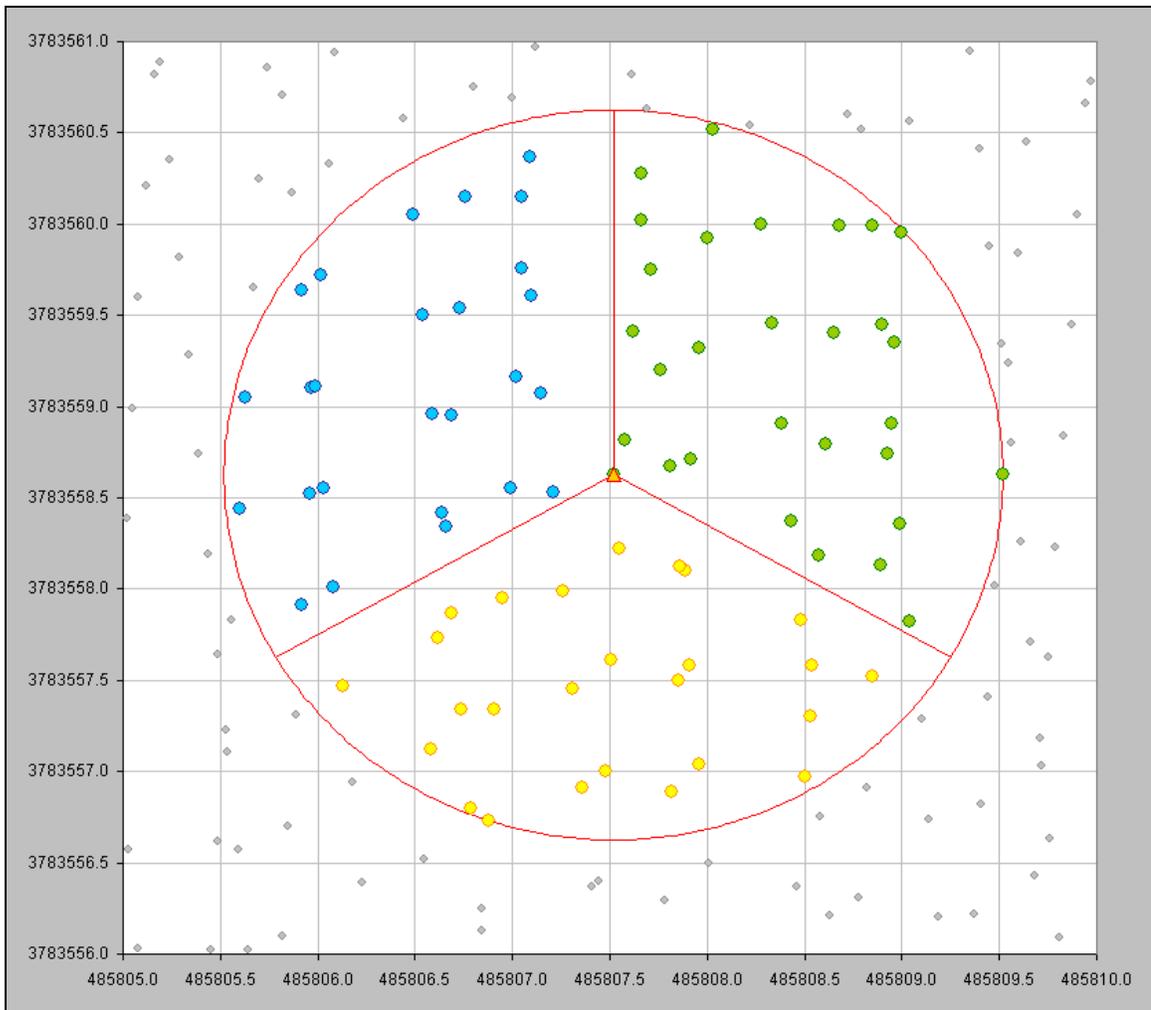


Figure 26 – LIDAR returns within two meters of a GPS point sorted into bins.

The returns within each bin were sorted from lowest elevation to highest. If two returns had the same elevation, the return farther from the GPS point was placed before the nearer return. A triangle was made using the first (lowest) return in each bin. This represents the lowest ground surface for the LIDAR points within two meters of the GPS point.

Tests with this lowest triangle determined that it often did not contain the GPS point. Most GPS points were at locations with moderate to steep slope. If for example, the slope in the images below (Figures 27 and 28) is toward the right edge, the lowest LIDAR return in the upper-left bin will be near the downhill edge, while the lowest LIDAR returns in the upper-right and bottom bins will be near the outside of the circle.

In other words the triangle is down slope from the GPS point. It does not take into account the LIDAR or ground conditions at or above the GPS point. It was also determined in testing that the slope of a triangle not containing the GPS point is often inaccurate (too high or low) when it is extrapolated to the GPS location.

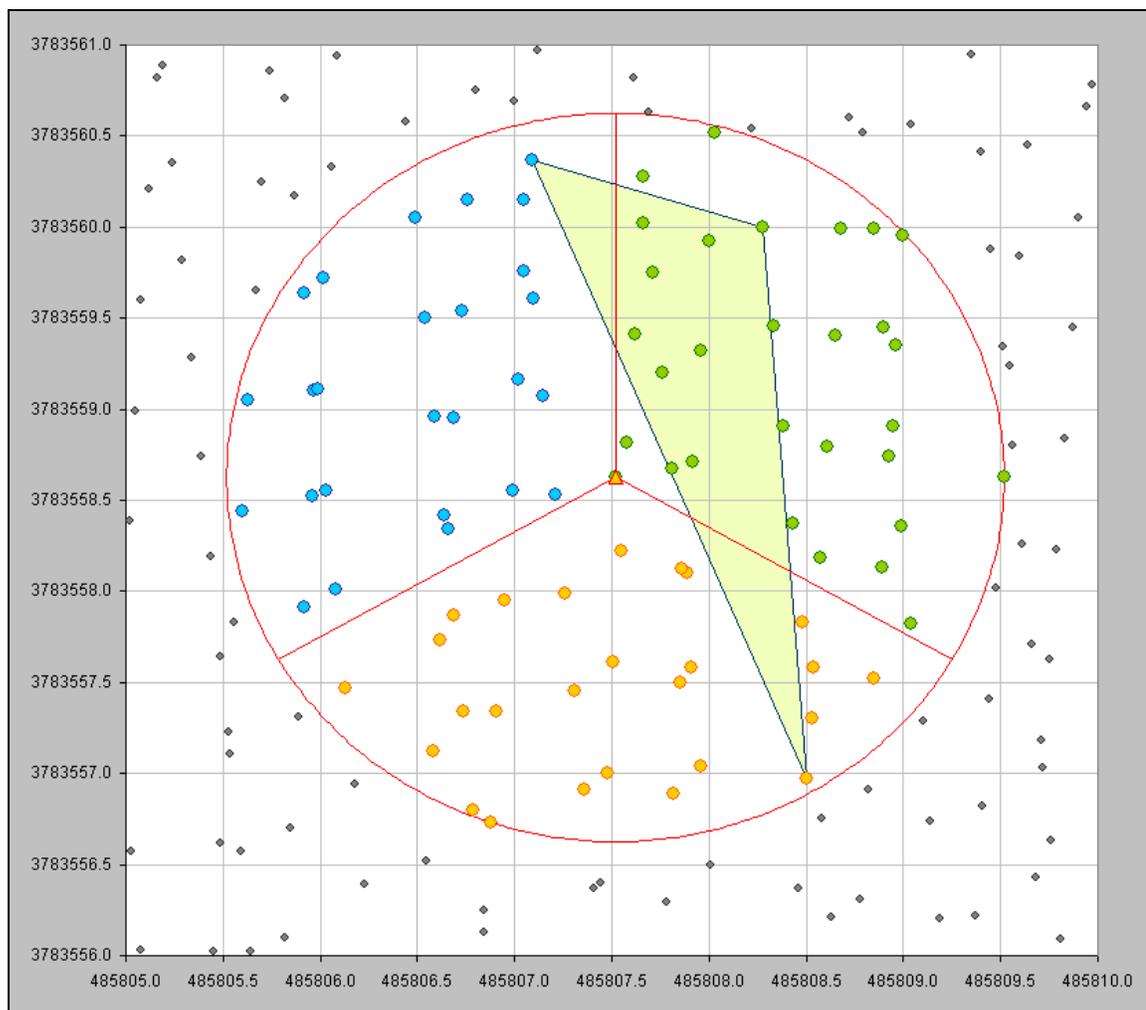


Figure 27 – Example LIDAR ground triangle not containing the GPS point.

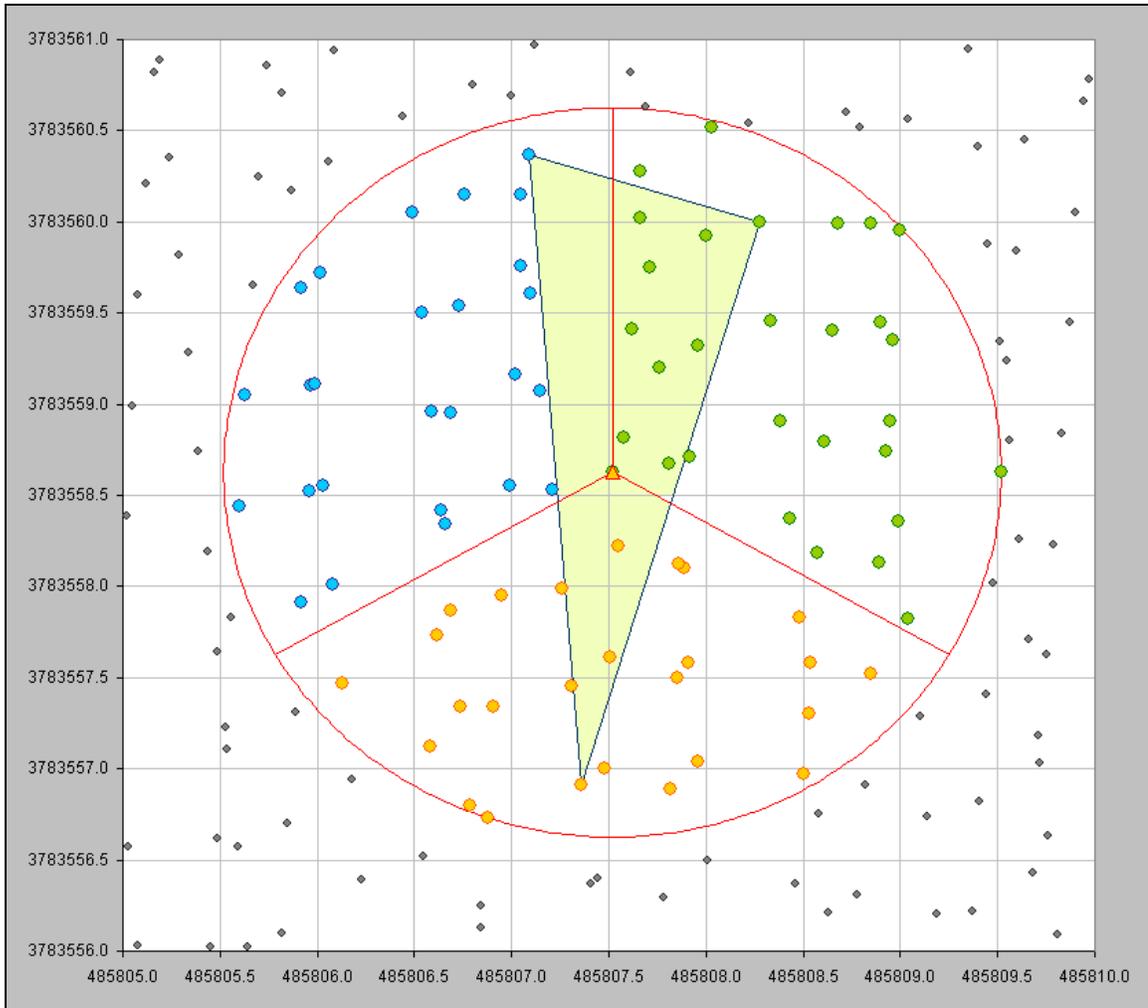


Figure 28 – Example LIDAR ground triangle containing the GPS point.

It was therefore required that the triangle contain the GPS point. To do this, each time a triangle was created it was tested to see if it contained the GPS point or not. If it did not, the return in any of the three bins with the next lowest elevation was selected, replacing the original return in its bin, and a new triangle was created and tested. This was repeated until the triangle contained the GPS point.

During testing, it was noted that the orientation of the bins had an effect on which LIDAR returns were used to create the triangle and therefore on the LIDAR ground elevation at the GPS point. To address this issue, the bins were rotated six times for each GPS point, 20 degrees each time, for a total rotation of 120 degrees. A triangle ground model containing the GPS point was created for each of the rotations, and the LIDAR

ground elevation at the GPS point was determined. The lowest elevation of the six rotations was used as the LIDAR ground. Height differences between the LIDAR and GPS ground elevations were then calculated by subtracting the GPS elevation from the LIDAR elevation. Positive differences mean the LIDAR ground is above the GPS ground, while negative differences mean the GPS ground is above the LIDAR ground.

Chapter 4

Results

4.1 Collected GPS Data

4.1.1 Overview of GPS Data

The GPS data were post-processed to get final position information using the following steps. All base station files were uploaded to the National Geodetic Survey (NGS) Online Position User Service (OPUS) website [4], where they were each corrected using three Continuously Operating Reference Stations (CORS) sites. Table 9 provides the results of the OPUS processing. Figure 29 shows the arrangement of the CORS sites used to correct the data in relation to the local base GPS points.

Table 9 – Base GPS OPUS processing results.

Base Station Name	Observations Used	Number of Fixed Ambiguities	Overall RMS (meters)	CORS Sites Used		
June 11	13741 / 14132 - 97%	51 / 54 - 94%	0.017	P470	CRFP	P584
June 12	12181 / 12319 - 99%	49 / 51 - 96%	0.013	P470	CRFP	P584
June 13	10557 / 10927 - 97%	81 / 85 - 95%	0.019	P470	CRFP	P584
June 14	8291 / 8447 - 98%	43 / 44 - 98%	0.016	P470	CRFP	P584
June 15	12983 / 13204 - 98%	45 / 47 - 96%	0.016	CLAR	TABL	P470
June 16	6785 / 6877 - 99%	30 / 30 - 100%	0.012	CLAR	TABL	P470
June 17	11320 / 11405 - 99%	44 / 45 - 98%	0.018	SFDM	ROCK	CMP9
June 18	11678 / 11886 - 98%	57 / 57 - 100%	0.014	SFDM	ROCK	CMP9
June 19	11691 / 11746 - 100%	44 / 45 - 98%	0.018	SFDM	ROCK	CMP9
June 20	6694 / 6735 - 99%	32 / 32 - 100%	0.015	SFDM	ROCK	CMP9
June 20 veght	3861 / 3942 - 98%	24 / 25 - 96%	0.013	SFDM	ROCK	CMP9

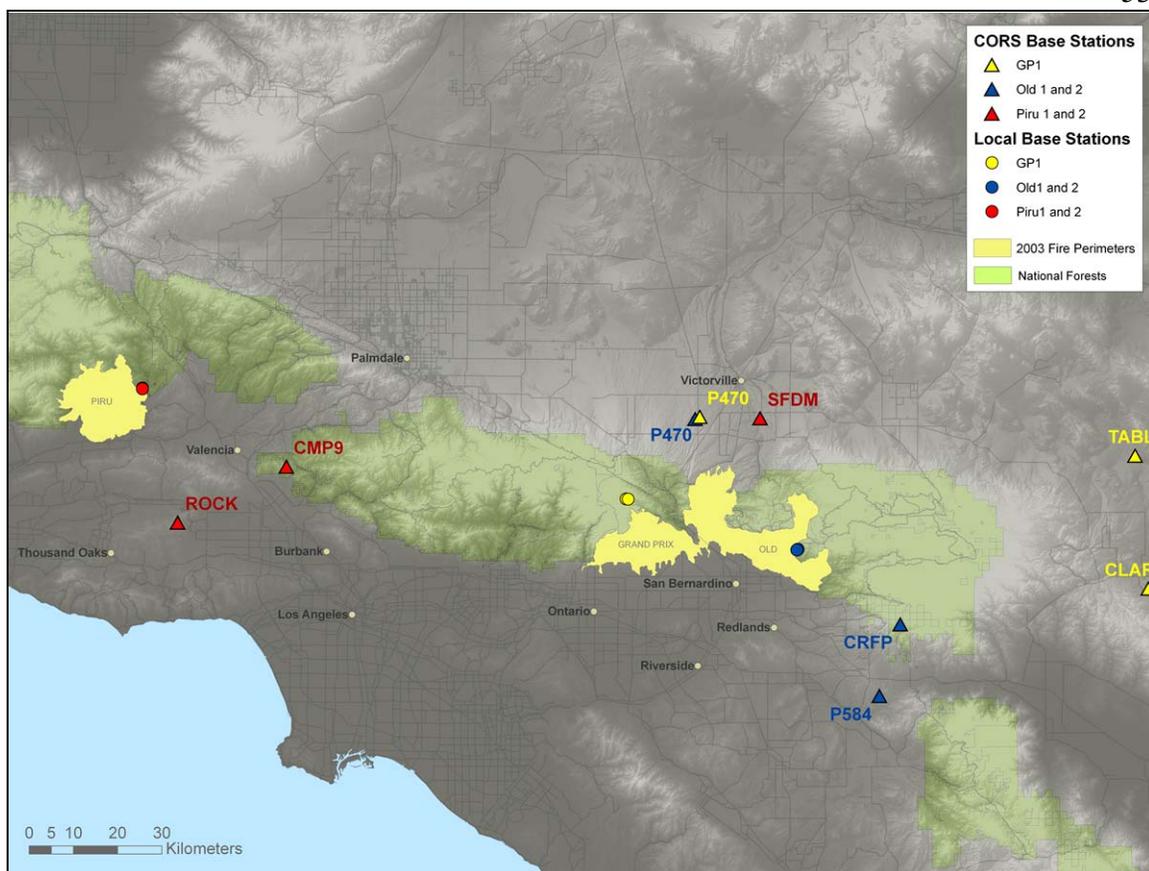


Figure 29 – Map local GPS base stations and the CORS sites used to correct them.

The rest of the GPS data were processed using the software Pinnacle [5]. The coordinates of the base GPS points provided by OPUS were used as the base station positions in Pinnacle, assigning the OPUS-derived coordinates to each local base GPS point. Vegetation and control GPS points were then corrected using these local base station files.

Over the course of 10 days, 1709 vegetation GPS points were collected. Tables 10 and 11 provide a breakdown of their types, vegetation heights, and vegetation densities.

Table 10 – Number and type of GPS points at each site.

Total Points by Site and Type				
Site	Veg non-RTK	Veg RTK	Control	Base
Old1	18	354	14	2
Old2	32	127	7	2
Grand Prix	0	263	14	2
Piru2	84	366	16	3
Piru1	19	446	15	2

Table 11 – Number, height, and density of vegetation GPS points at each site.

Site	Point Type	Categorized 2007 Height				Density		
		0-1m	1-2m	2-3m	>3m	0	1	2
Old1	Veg RTK	82	195	62	15	27	164	163
	Veg non-RTK	8	9	1	0	2	12	4
Old2	Veg RTK	2	28	75	22	0	51	76
	Veg non-RTK	1	26	5	0	0	0	32
Grand Prix	Veg RTK	42	125	74	22	10	72	181
	Veg non-RTK	0	0	0	0	0	0	0
Piru2	Veg RTK	59	159	131	17	14	202	150
	Veg non-RTK	14	59	5	6	0	31	53
Piru1	Veg RTK	103	271	57	15	30	300	116
	Veg non-RTK	1	18	0	0	0	10	9

Site Old 2 had significantly more difficult working conditions than the other sites. Most sites had the largest percentage of their GPS points in the one to two meter vegetation height category. Old 2 had the highest percentage of points in the two to three meter category, a higher percentage of points in the greater than three meter category than the other sites, and almost no points in the zero to one meter category. Old 2 and Grand Prix had approximately two-thirds of their points in the highest density vegetation, Old 1 and Piru 2 had nearly half, and Piru 1 had around one quarter. In no site did points with zero vegetation density account for more than eight percent of the points.

4.1.2 GPS Data Quality

The GPS data provides “true” ground elevation data for this thesis, so with the short occupation times, and multiple types of data collection, it is important to show that the GPS positions are trustworthy. Figure 30 displays the average PDOP for each

vegetation GPS position, against the number of satellites used for each vegetation GPS position. It is important to notice that all positions were collected using between six and 14 satellites, and that the vast majority were collected with PDOP values below four.

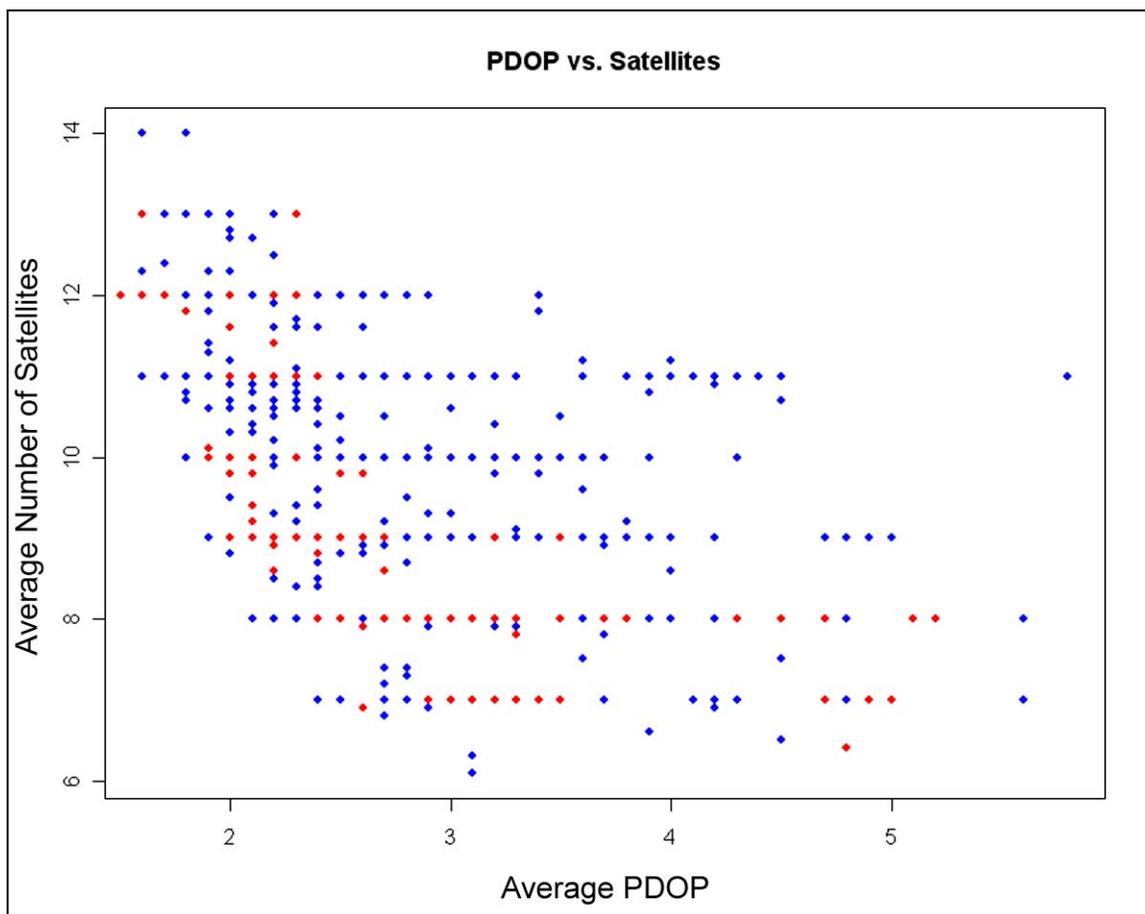


Figure 30 – All vegetation GPS points, PDOP vs. number of satellites, RTK (blue), non-RTK (red).

RMS (Root Mean Square Error) is a measure of the precision of the individual measurements used to calculate a GPS position. Each RTK vegetation point used 15 individual position measurements (15 one second epochs) to calculate the final point position. The smaller the RMS value, the more tightly grouped the individual measurements were. If there is a great deal of variability in the individual measurements (high RMS), there is an increased likelihood that the accuracy of the final position is questionable. RMS is calculated using this formula:

$$RMS = \sqrt{\sigma_X^2 + \sigma_Y^2 + \sigma_Z^2}$$

where the σ^2 values are calculated using this formula (substituting X, Y, Z as needed):

$$\sigma_X = \sqrt{\frac{\sum_{i=1}^N (X_{i,obs} - X_{calculated})^2}{N}}$$

where $X_{i,obs}$ is an easting value for an individual epoch at a point, $X_{calculated}$ is the final easting position for a point, and N is the number of epochs used to calculate the position. $Y_{i,obs}$ and $Y_{calculated}$ are the northing values for a point, and $Z_{i,obs}$ and $Z_{calculated}$ are the elevation values for a point.

Table 12 provides a break down of the number and percentage of GPS points in each of three RMS categories: less than two centimeters, two to 10 centimeters, and greater than 10 centimeters. The majority of vegetation GPS points had very low RMS values, demonstrating that the data used to calculate the positions, were very precise. It is therefore likely that the position data for these points are trustworthy.

Table 12 – The number and percentage of GPS points in three RMS categories, RMS in meters.

Point Type	Total Points	# RMS < 0.02	%	#RMS 0.02 - 0.1	%	# RMS > 0.1	%
All Veg	1709	1118	65.4	494	28.9	97	5.7
Veg RTK	1556	1002	64.4	468	30.1	86	5.5
Veg non-RTK	153	116	75.8	26	17.0	11	7.2

Figure 31 is a plot of RMS and PDOP values for the GPS points, again showing that the majority of GPS points had RMS values below 10 centimeters.

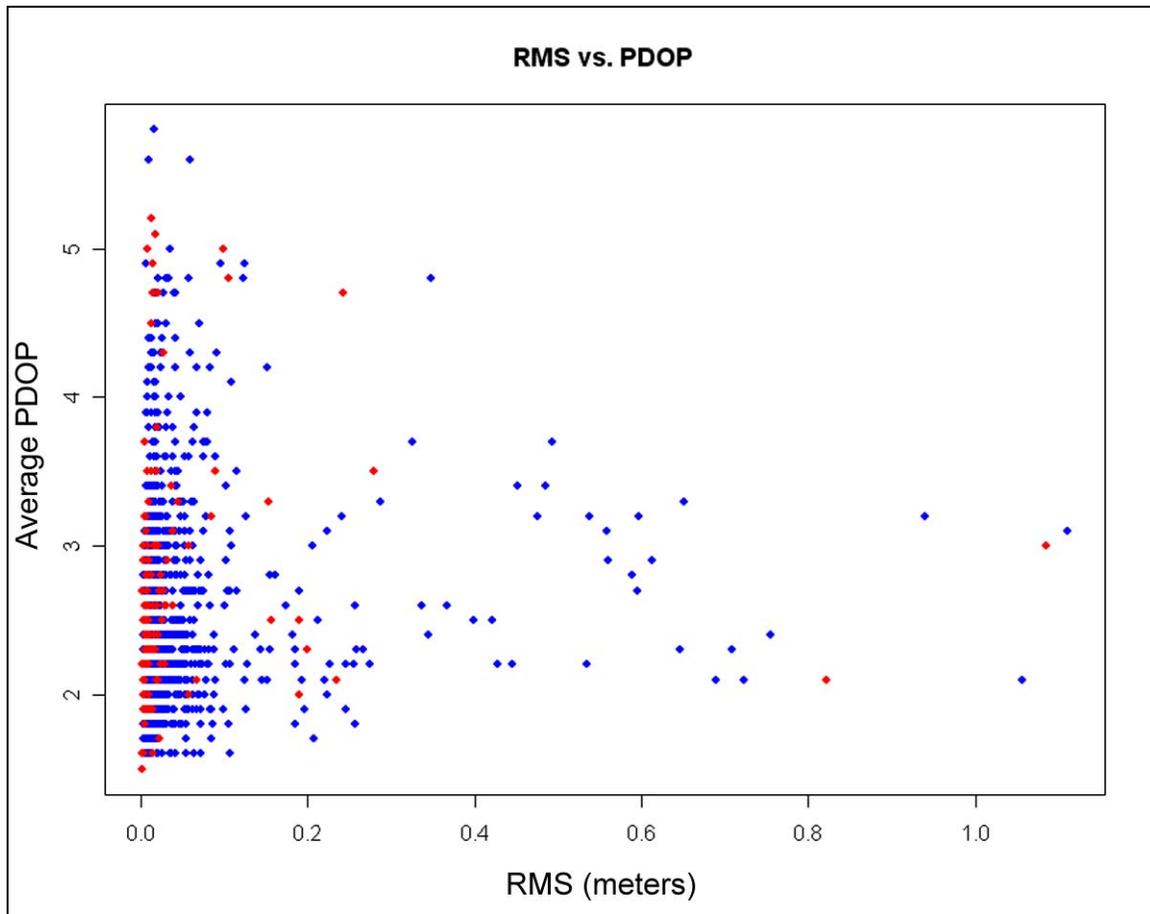


Figure 31 – Vegetation GPS points RMS vs. PDOP, RTK (blue) non-RTK (red).

It was of interest to come up with an estimate of the range of RMS values that could reasonably be expected for the GPS data, and to see if any of the field data fell outside of this range. It may be worth questioning the accuracy of GPS points with RMS values outside of this range. A simulation was performed to show what RMS data should look like.

The distribution that best fits the RMS data is important for an accurate simulation. RMS values cannot be less than zero, and low RMS values tend to occur much more frequently than high RMS values. Assuming that RMS values are best described by a normal distribution is likely not accurate. Figure 32 is a cumulative distribution function (CDF) of the RMS field data (black line), the CDF line for a normal distribution with the same mean and standard deviation as the RMS field data (blue), and the CDF line for an exponential distribution with the same rate as the RMS field data

(red) created in the statistical software R [12]. The exponential distribution is a better fit to the data.

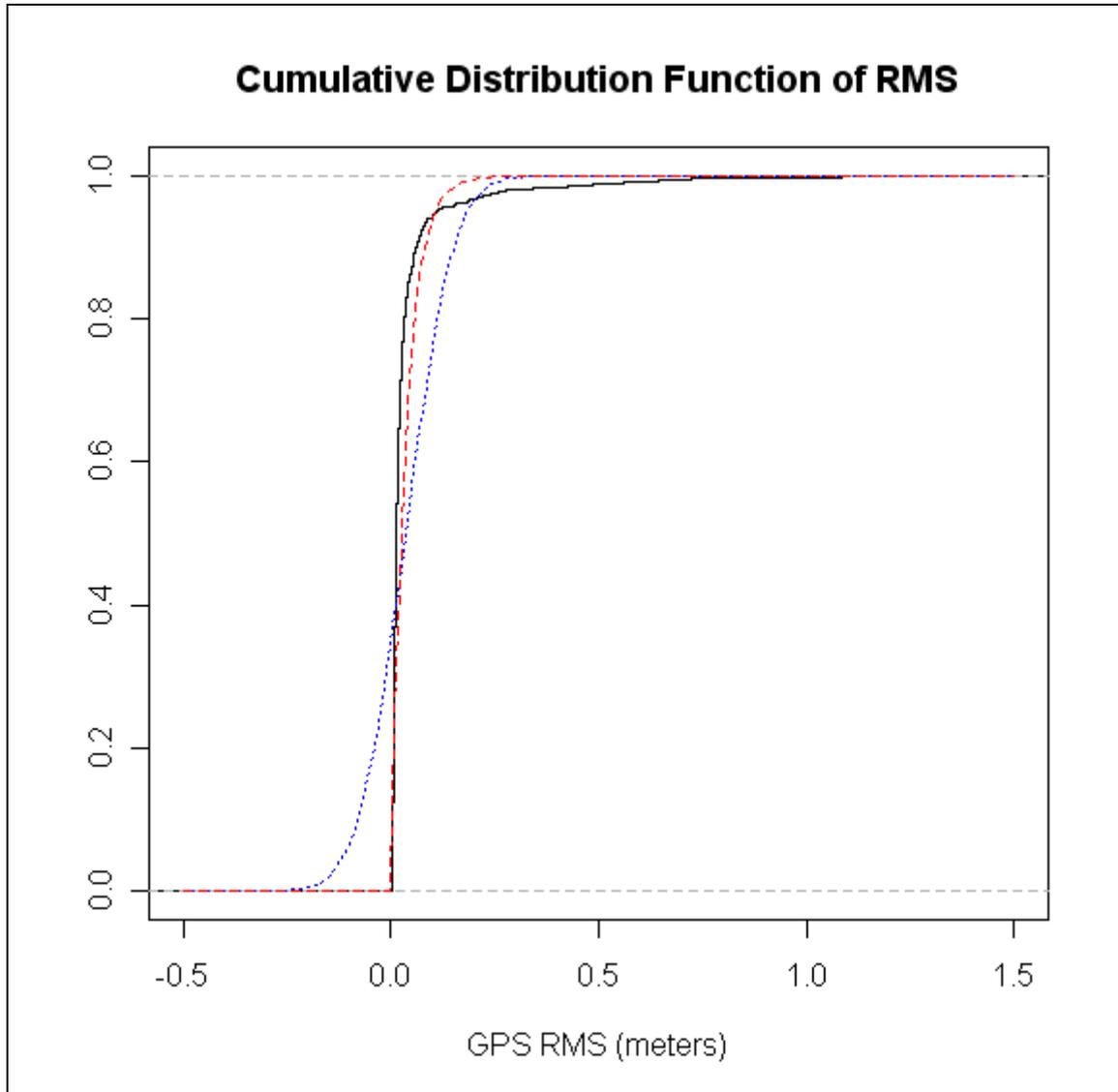


Figure 32 – Cumulative distribution function of the RMS field data values.

To further confirm that RMS data may be better described by an exponential distribution than a normal distribution, histograms of the RMS field data were plotted with a normal distribution overlaid (Figure 33) and an exponential distribution overlaid (Figure 34) using R. Again the exponential distribution appears to better fit the data.

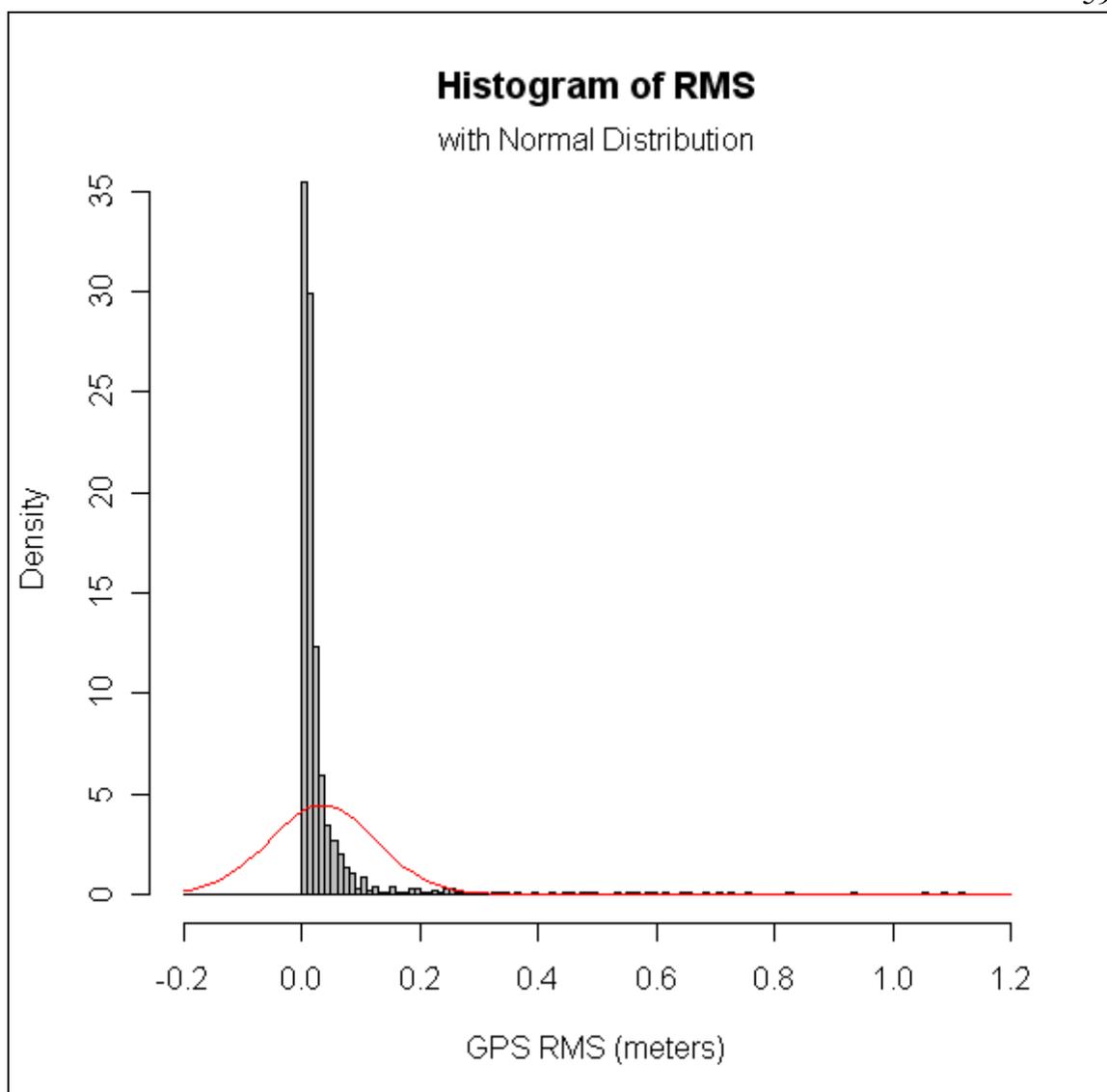


Figure 33 – Histogram of the RMS field data values with a normal distribution overlaid.

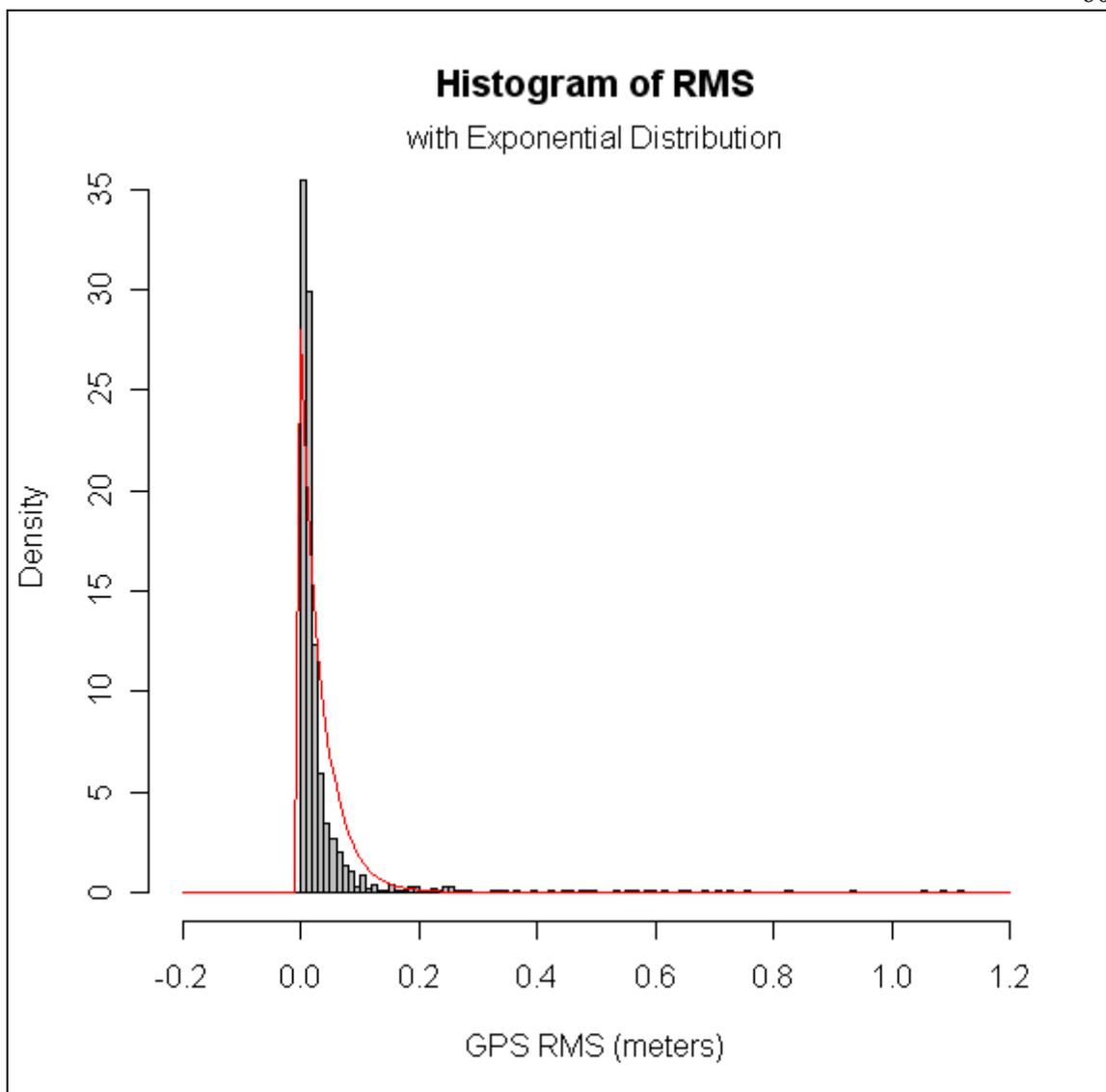


Figure 34 – Histogram of the RMS field data values with an exponential distribution overlaid.

Figure 35 is a plot of the field data RMS values and the LIDAR-GPS height differences (calculated as described in Section 4.3.1). The range of calculated height differences between the GPS ground elevations and the modeled LIDAR ground elevations did not increase as RMS values increased.

Figure 36 is a plot of simulated data. 100,000 height difference values were drawn at random from a normal distribution with the same mean and standard deviation as the GPS data from the field, and 100,000 RMS values were drawn from an exponential distribution with a rate of $1/\text{mean}(RMS)$, where $\text{mean}(RMS)$ is the mean of the RMS

values from the GPS field data. It is interesting to note that no RMS values were greater than 0.45 meters. This may call into question the 23 GPS points with RMS values greater than 0.45. However, because these 23 points are such a small percentage of the total data, they were not removed from the analysis.

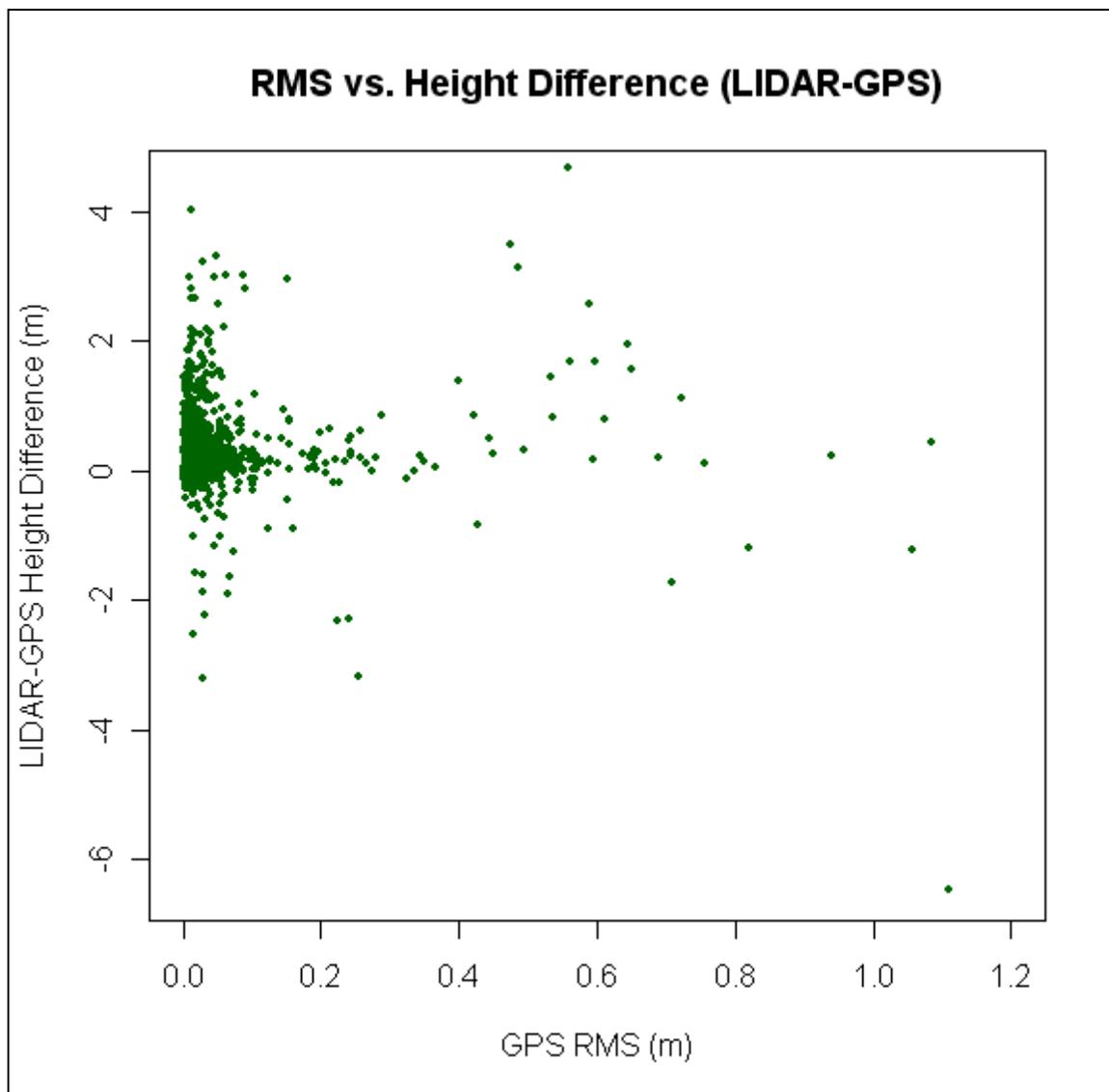


Figure 35 – Field data RMS values (meters) vs. calculated LIDAR-GPS height differences (meters).

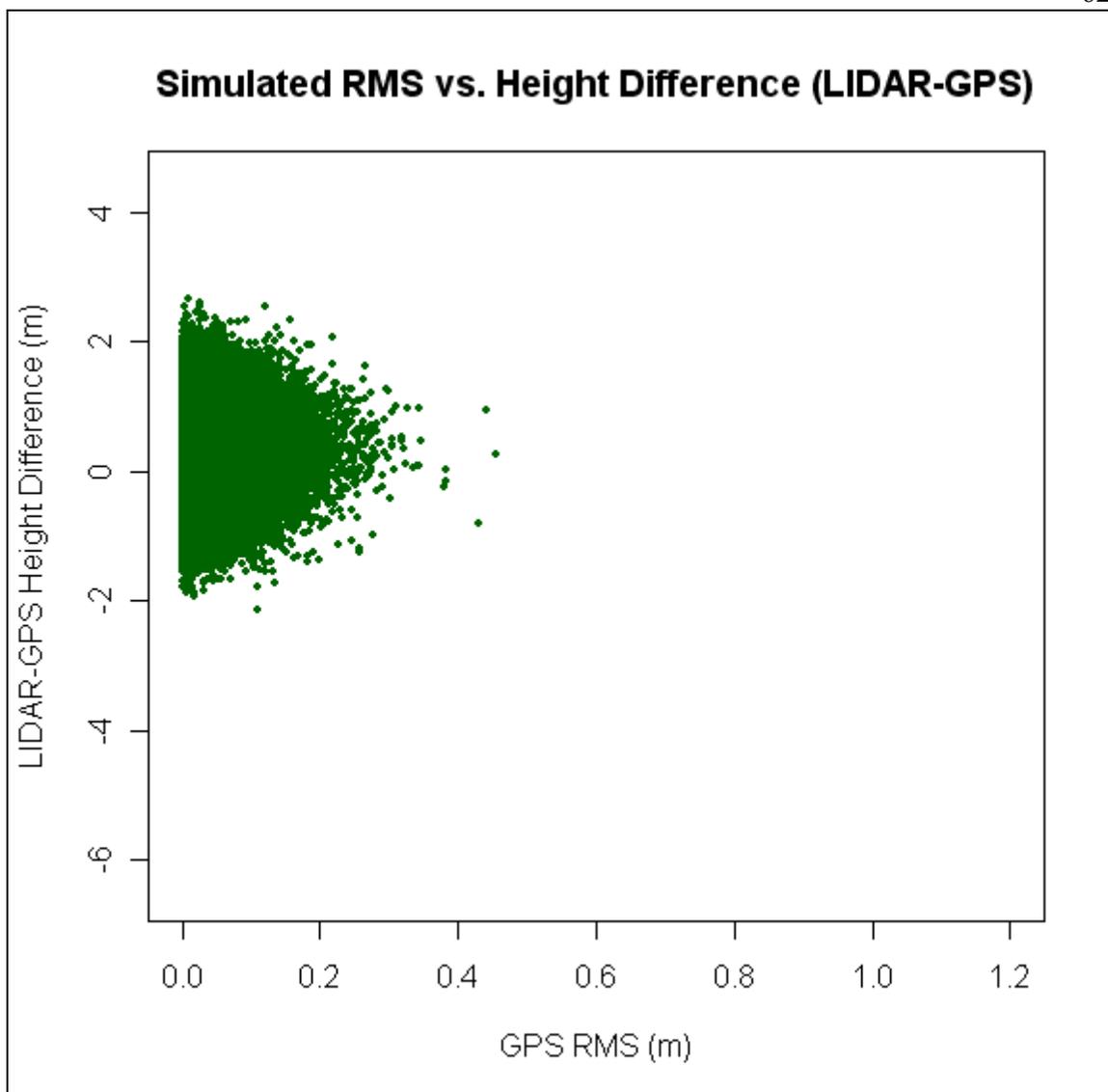


Figure 36 – Simulated RMS values (meters) vs. simulated LIDAR-GPS height differences (meters).

4.2 Vegetation Growth

For the 75 GPS points taken within the three 2005 field plots, it was determined, using the method described in Section 3.7, that the average vegetation growth between when the LIDAR was flown in 2005 and when the GPS data were collected in 2007, was 0.54 meters \pm 0.53 meters. 0.54 was subtracted from the 2007 vegetation heights to estimate 2005 vegetation heights.

4.3 LIDAR / GPS comparison

4.3.1 Elevation Difference Formula

LIDAR and GPS ground elevations were compared by calculating the differences in their elevation values at the 1709 vegetation GPS points, using this formula:

$$\text{elevation difference} = \text{LIDAR elevation} - (\text{GPS elevation} - \text{local offset})$$

where *LIDAR elevation* was calculated using the Triangle Method as described in Section 3.8, *GPS elevation* was collected in the field, and *local offset* was calculated as described in Section 4.3.2 below.

4.3.2 Local Offset

The first step in determining if there was a difference between the LIDAR ground elevations and GPS ground elevations at each GPS point was removing any systematic offset between the two datasets. This systematic offset, the local offset, was determined by finding the average difference between the LIDAR and GPS elevations at the control GPS points for a site, all located in flat, open, non-vegetated areas. The local offset for each of the five sites was determined individually. This average value for each site was subtracted from the GPS elevations at that site to correct for the systematic bias. The Triangle Method with a two meter selection radius was used to calculate the LIDAR elevations for each control GPS point. Table 13 provides the offsets for each site.

Table 13 – Average control GPS point ground elevation offset (LIDAR-GPS) for each site in meters.

Site	Average Height Offset	Std. Dev. Of Height Offsets	# of Points
Old 1	0.239	0.042	11
Old 2	0.246	0.116	5
Grand Prix	0.184	0.060	12
Piru 2	0.283	0.073	13
Piru 1	0.207	0.060	15

4.3.3 Elevation Differences

The second step in determining the elevation differences was to use the Triangle Method to calculate the LIDAR ground elevation at all vegetation GPS points. Multiple selection radii were tested, from one to 10 meters (Figure 37). When all 1709 vegetation

GPS points were considered, the five meter selection radius had the average height difference closest to zero. The following results were calculated using this five meter selection radius in the Triangle Method.

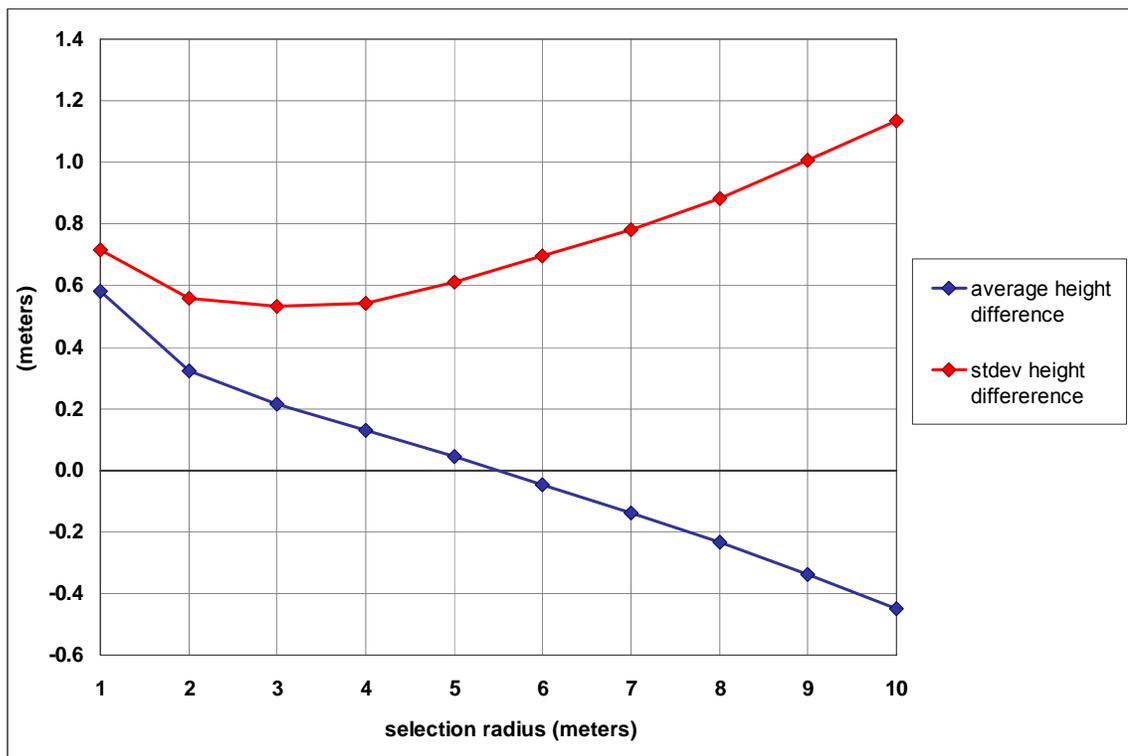


Figure 37 – Average LIDAR-GPS height differences (meters) and Standard Deviations (meters) for all Triangle Method selection radii (meters).

The formula in Section 4.3.1 was used to calculate the elevation differences between the LIDAR and GPS datasets (Table 14).

Table 14 – Ground elevation differences (LIDAR-GPS) for vegetation GPS points, in meters.

		Category	Mean	Std. Dev.	# of Points
		All Points	0.043	0.613	1709
		RTK Points	0.042	0.601	1556
		Non-RTK Points	0.058	0.726	153
		Density 0 Points	-0.060	0.374	83
		Density 1 Points	-0.003	0.549	842
		Density 2 Points	0.104	0.688	784
Vegetation Height	2007	Height 0-1m Points	-0.032	0.477	312
		Height 1-2m Points	0.030	0.581	890
		Height 2-3m Points	0.133	0.699	410
		Height gt3m Points	0.031	0.822	97
	2005	Height 0-1m Points	-0.002	0.493	862
		Height 1-2m Points	0.091	0.688	586
		Height 2-3m Points	0.118	0.725	164
		Height gt3m Points	n/a	n/a	97

The GPS points were separated into categories to see if the receiver type, vegetation height, and vegetation density had an effect on the LIDAR-GPS height differences. Table 14 displays the results for RTK vegetation GPS points, non-RTK vegetation GPS points, and the GPS points categorized by their field-recorded density. It also has results for points broken down into one meter vegetation height categories. The green rows are the 2007 vegetation heights, while the yellow rows are the 2005 vegetation heights. The greater than three meter height category could not be adjusted by subtracting the vegetation growth, because true heights at those points could not be measured in the field. Therefore the results are the same in this height category for both 2007 and 2005 heights.

Figures 38 through 40 are box plots of the LIDAR-GPS height differences separated into categories. Figure 38 displays the three density groups, Figure 39 displays the four 2007 vegetation height categories, and Figure 40 displays the four 2005 estimated vegetation height categories. In all three box plots, the means increase as vegetation densities or vegetation heights increase. The standard deviations also increase as the vegetation densities or heights increase.

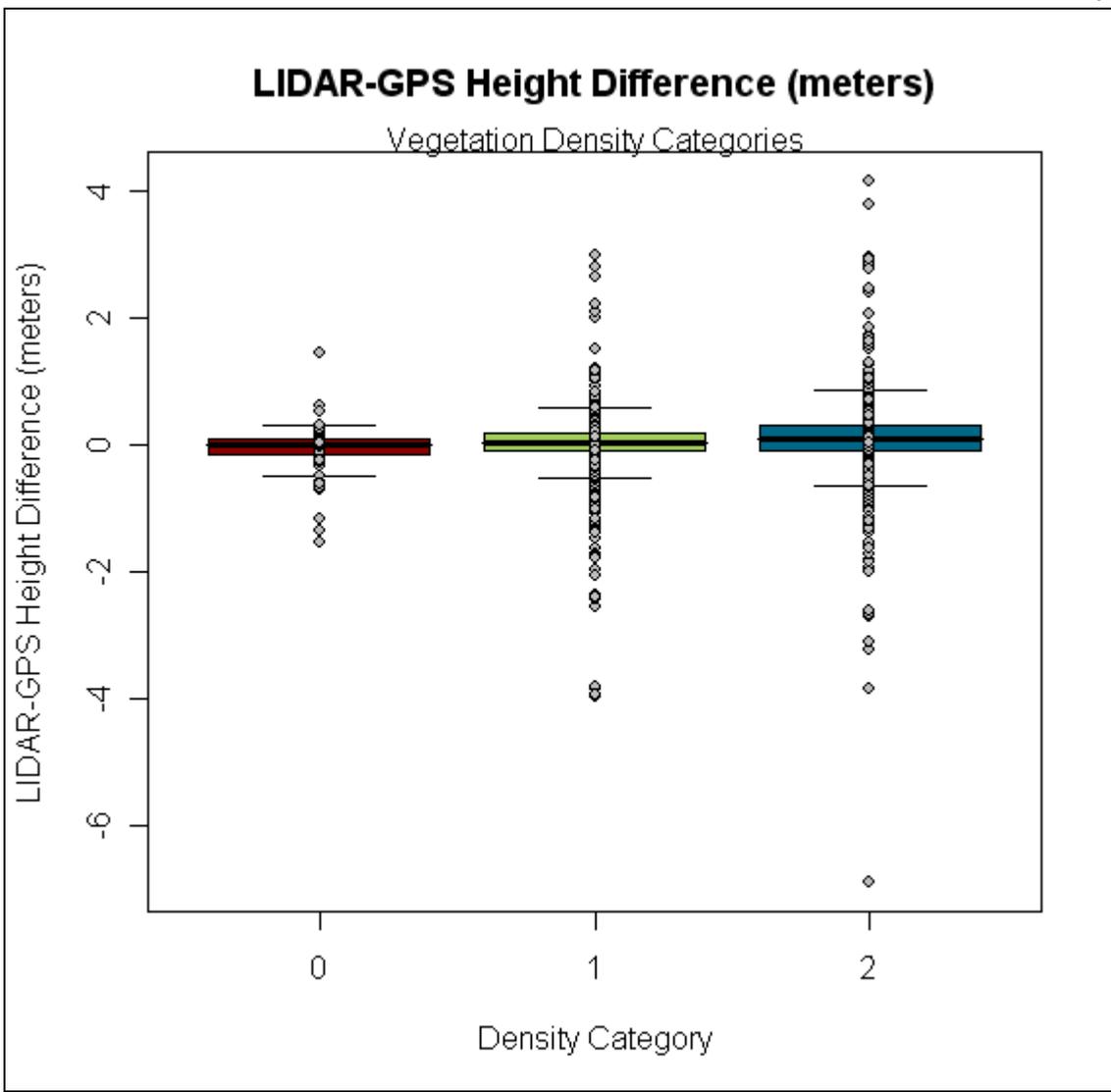


Figure 38 – Box plot of LIDAR-GPS height differences for vegetation density categories.

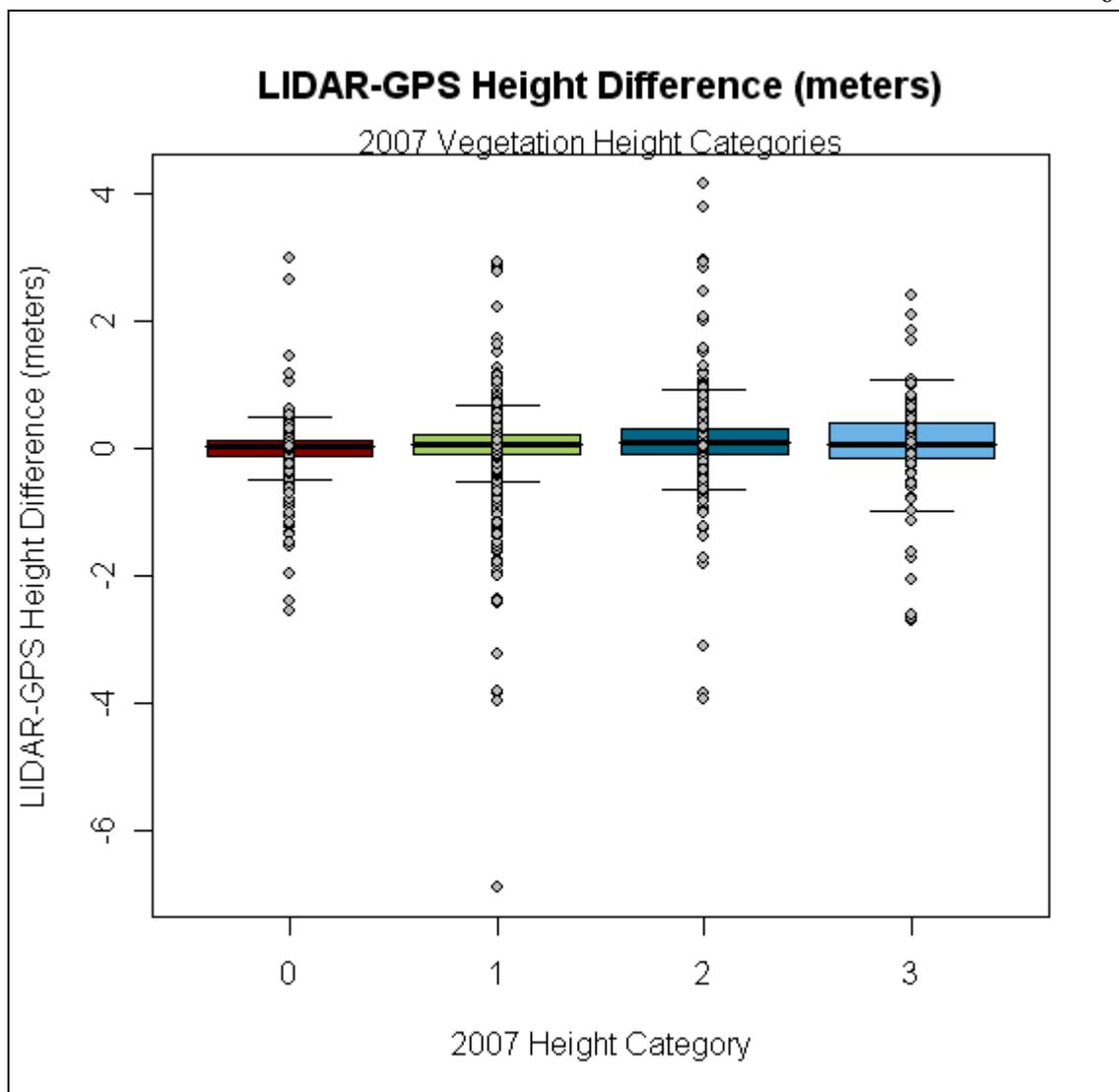


Figure 39 – Box plot of LIDAR-GPS height differences for 2007 vegetation height categories.

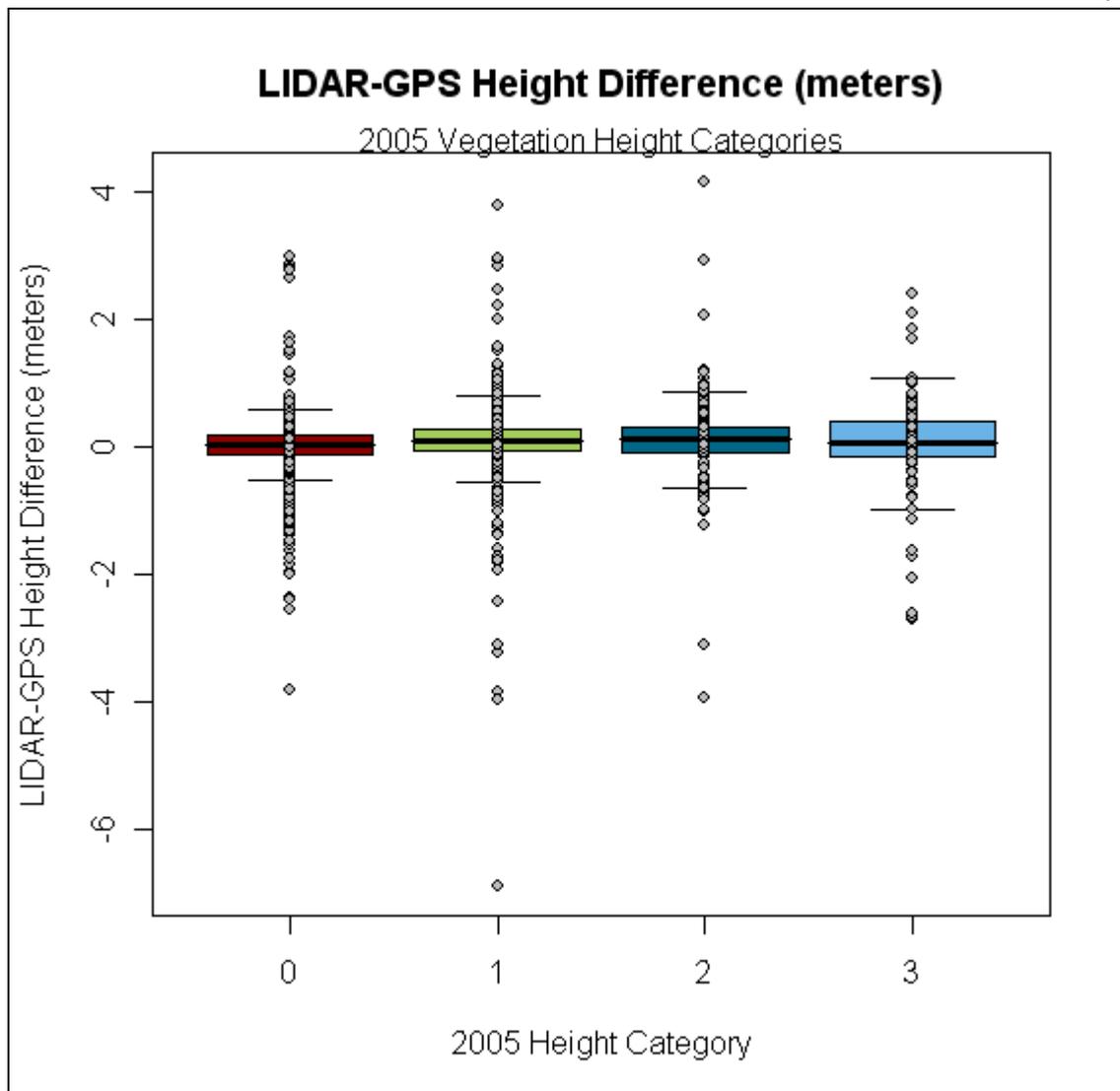


Figure 40 – Box plot of LIDAR-GPS height differences for 2005 vegetation height categories.

4.3.4 Predicting DTM Accuracy

Noting that there was a trend of increasing DTM error (LIDAR-GPS elevation difference) as the vegetation height or density increased, led to the question of whether or not DTM error could be predicted if vegetation height or density were known. To answer this question linear regression models were developed in the statistical software R [12]. The categorized height and density data were treated as categorical variables in the regression models, using the R function “as.factor()”.

ANOVAs (using single independent variables) for the data show that the mean height differences for the density and height categories are not equal (Table 15).

Performing a Student-Newman-Keuls (SNK) test on the means showed that the mean of density category two is significantly different than the means of categories zero and one, and that the mean of density category one is not significantly different than the mean of category zero. Using the same test, the mean for height category two is significantly different than the means for height categories zero and one, using the 2007 vegetation height categorization. Using the 2005 vegetation height categorization, the mean for height category two is significantly different than the mean for height category zero, and the mean for height category one is significantly different than the mean for height category zero. All other height categories for both categorizations have means that are not significantly different.

Table 15 – ANOVA results.

Independent Variable	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Density (categorical)	1	5.51	5.51	14.79	1.24E-04
Error	1707	635.33	0.37		
Categorized 2007 Heights (categorical)	1	3.05	3.05	8.16	4.33E-03
Error	1707	637.78	0.37		
Categorized 2005 Heights (categorical)	1	1.87	1.87	5.01	2.54E-02
Error	1707	638.96	0.37		

Figures 41 through 43 present the calculated LIDAR-GPS height differences plotted against 2007 vegetation height (Figure 41), 2007 categorized vegetation height (Figure 42), and vegetation density (Figure 43). The spread of the height differences (the variance) is wide for all levels of vegetation height and density, but increases slightly as the height and density levels increase. The slopes of the regression lines are also all positive, showing that as vegetation height and density increase, so will the LIDAR-GPS height difference values. For the continuous vegetation heights (Figure 41), the points with a height greater than three meters were not included, because their true heights were not known.

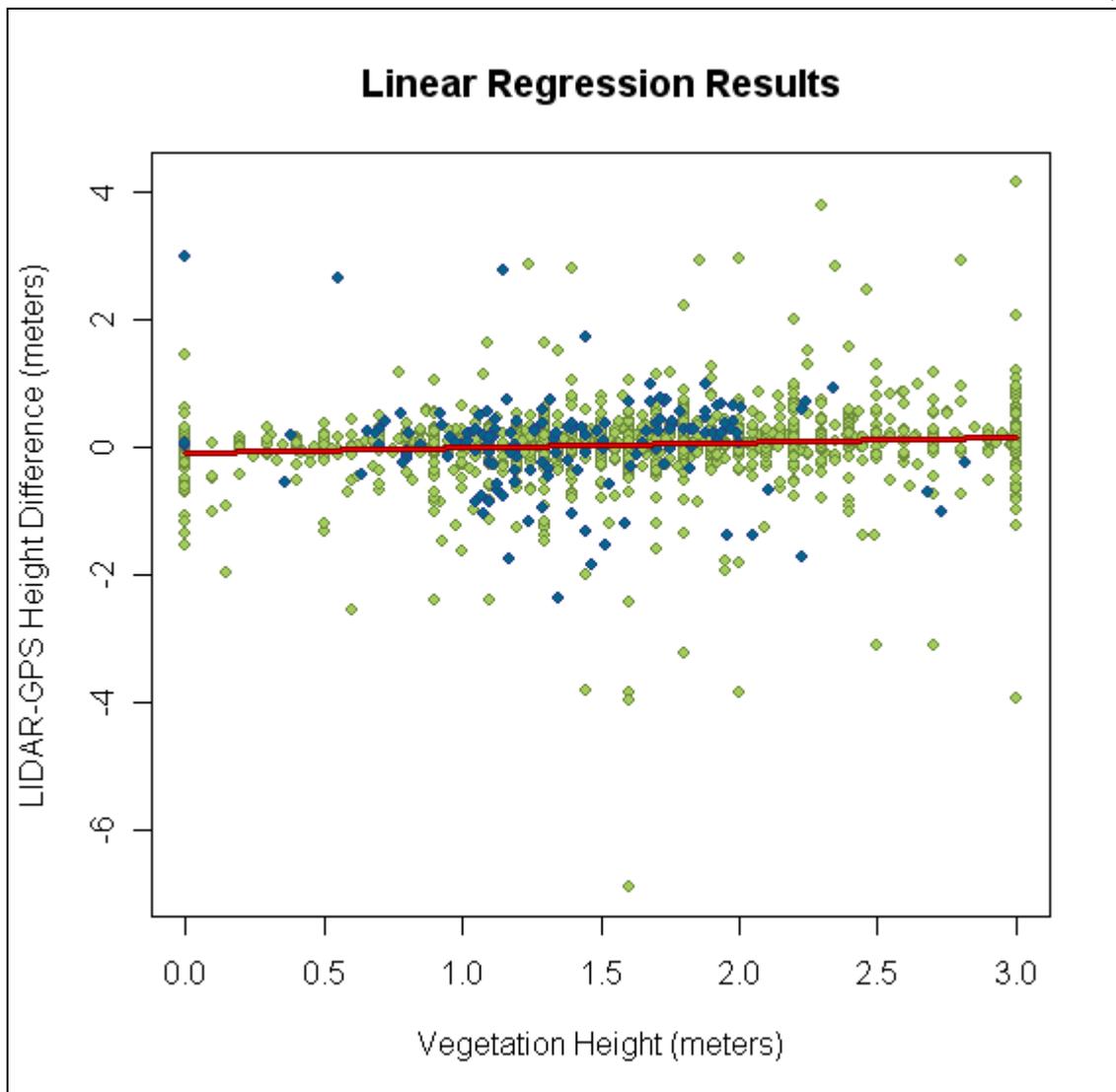


Figure 41 – Plot of LIDAR-GPS height difference results against 2007 vegetation heights. RTK vegetation points (green), non-RTK vegetation points (blue), regression line (red).

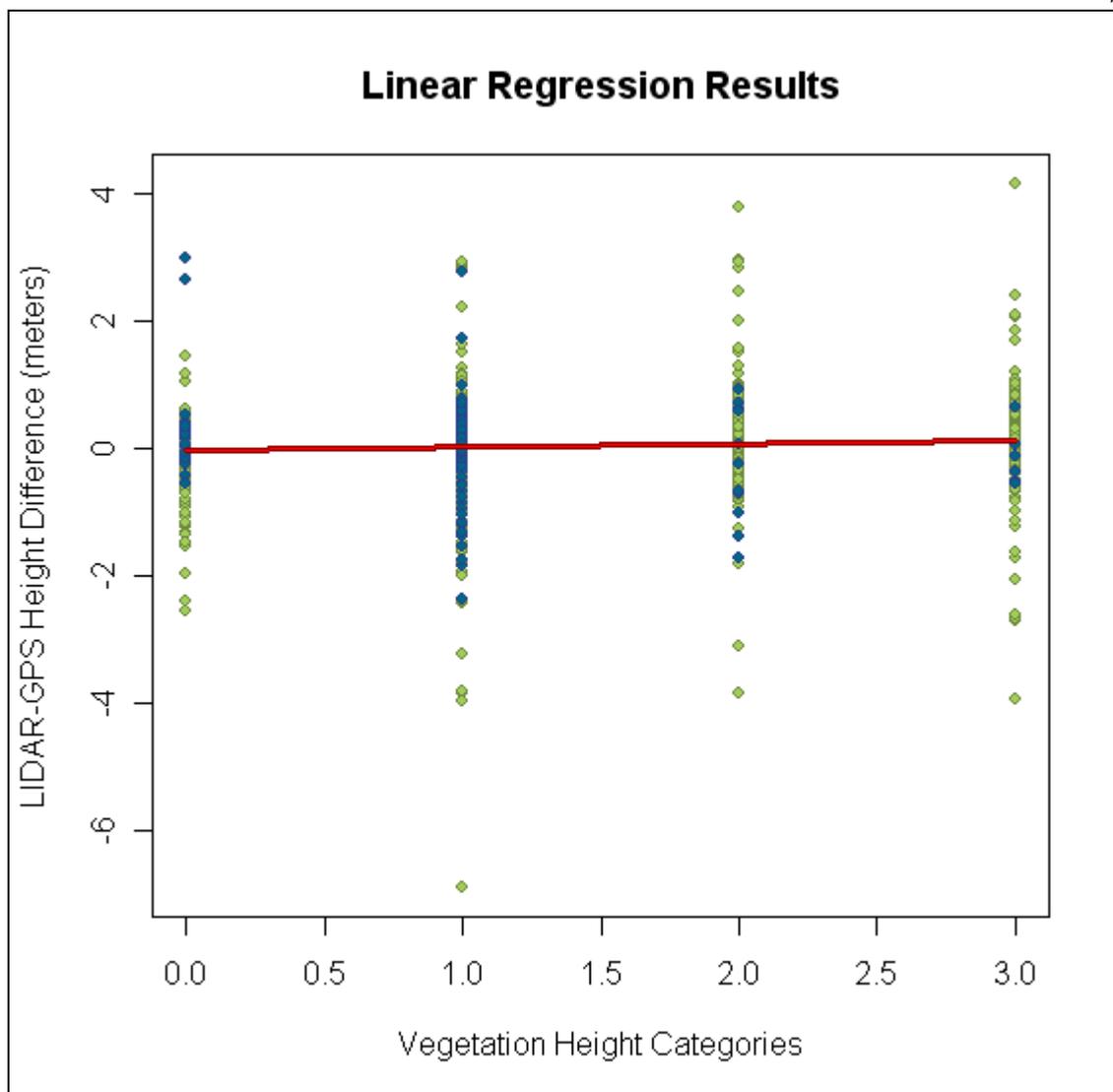


Figure 42 – Plot of LIDAR-GPS height difference results against categorized 2007 vegetation heights. RTK vegetation points (green), non-RTK vegetation points (blue), regression line (red).

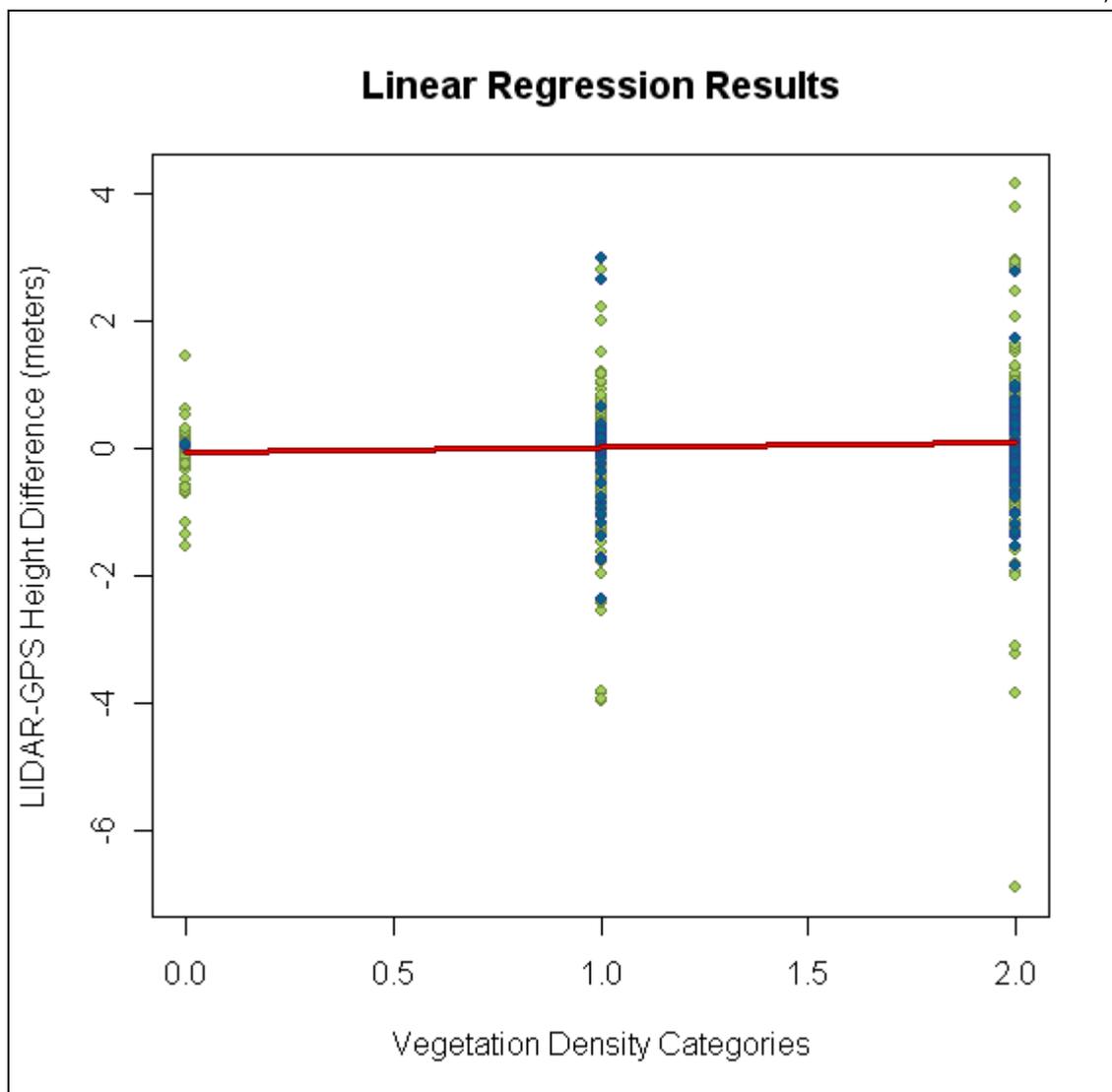


Figure 43 – Plot of LIDAR-GPS height difference results against density categories, RTK vegetation points (green), Non-RTK vegetation points (blue), regression line (red).

ANOVAs (using multiple independent variables) for the data showed that density, vegetation height, and their interaction were significant in determining the LIDAR-GPS height differences.

No linear regression model using vegetation density, 2007 or 2005 vegetation heights, or categorized 2007 or 2005 vegetation heights, was able to produce an R^2 value greater than 4 percent (Table 16). The large variance of the LIDAR-GPS height

differences is a major factor in the low R^2 values. It appears that predicted DTM error from chaparral height and density data will have low accuracy.

Table 16 – R^2 values for various linear regression models.

Dependent Variable	Independent Variable(s)	R^2
LIDAR-GPS height difference	One Independent Variable	
	Density	0.01
	2007 Heights	0.01
	2005 Heights	0.00
	Categorized 2007 Heights	0.00
	Categorized 2005 Heights	0.00
	Two Independent Variables	
	Categorized 2007 Heights * Density	0.01
	2007 Heights * Density	0.01
	Categorized 2005 Heights * Density	0.02
	2005 Heights * Density	0.01
	Five Independent Variables	
	Density * 2007 Heights * 2005 Heights * Categorized 2007 Heights * Categorized 2005 Heights	0.04

Discussion

5.1 Error Budget

It is important to discuss the error budget, the error that could be expected in a solution due to the error in each of the components. If the error detected between the LIDAR and GPS ground elevations is smaller than the error produced by the equipment used to measure the elevations, it cannot be called significant.

The simplest method of approaching the error budget is to examine the error detected in the control point elevations, because these points were collected under the most ideal data conditions: flat, open, non-vegetated areas. The control point offset must be subtracted from these points in order to examine error. Because the offsets, or biases, are the average of the height differences for the control points, the average error for the control points is 0.00 meters \pm 0.06 meters. So under ideal conditions, 68 percent of the GPS elevations should be within six centimeters of the LIDAR elevations.

It is also useful to examine the vegetation points with bare ground, in other words vegetation points with a field measured height and density of zero. These points are not limited to locations on flat ground, so they demonstrate expected errors in conditions slightly less ideal than the control points. There were 79 vegetation GPS points in this category with an average LIDAR-GPS height difference of 0.02 meters \pm 0.29 meters.

To further explore what to expect for the differences between LIDAR and GPS ground models, simulations were run for 10,000 points selecting vertical and horizontal errors for the GPS and LIDAR equipment from normal distributions based on the equipment manufacturer's accuracy specifications. A random slope from a normal distribution between -50 and 50 degrees was selected for each point.

The initial simulation assumed that there is no positional error in the LIDAR data, that the GPS is sitting on the LIDAR surface, and that the GPS positional error follows the specifications provided by Javad Navigation Systems, Inc., in Table 2 (horizontal accuracy of \pm 0.01 meters plus 1.5 parts per million (ppm), and vertical accuracy of \pm

0.02 meters plus 1.5 ppm). On perfectly flat ground, where horizontal errors cannot contribute to vertical error, LIDAR elevation errors ranged from -0.08 meters to 0.07 meters. Adding a slope term into the simulation made it possible for horizontal errors to contribute to vertical errors. Slopes were limited to ± 50 degrees. The vertical errors were 0.00 meters \pm 0.02 meters with errors ranging from -0.08 meters to 0.07 meters (Figure 44). These results suggest that the very small horizontal errors result in very little slope effect on the vertical errors.

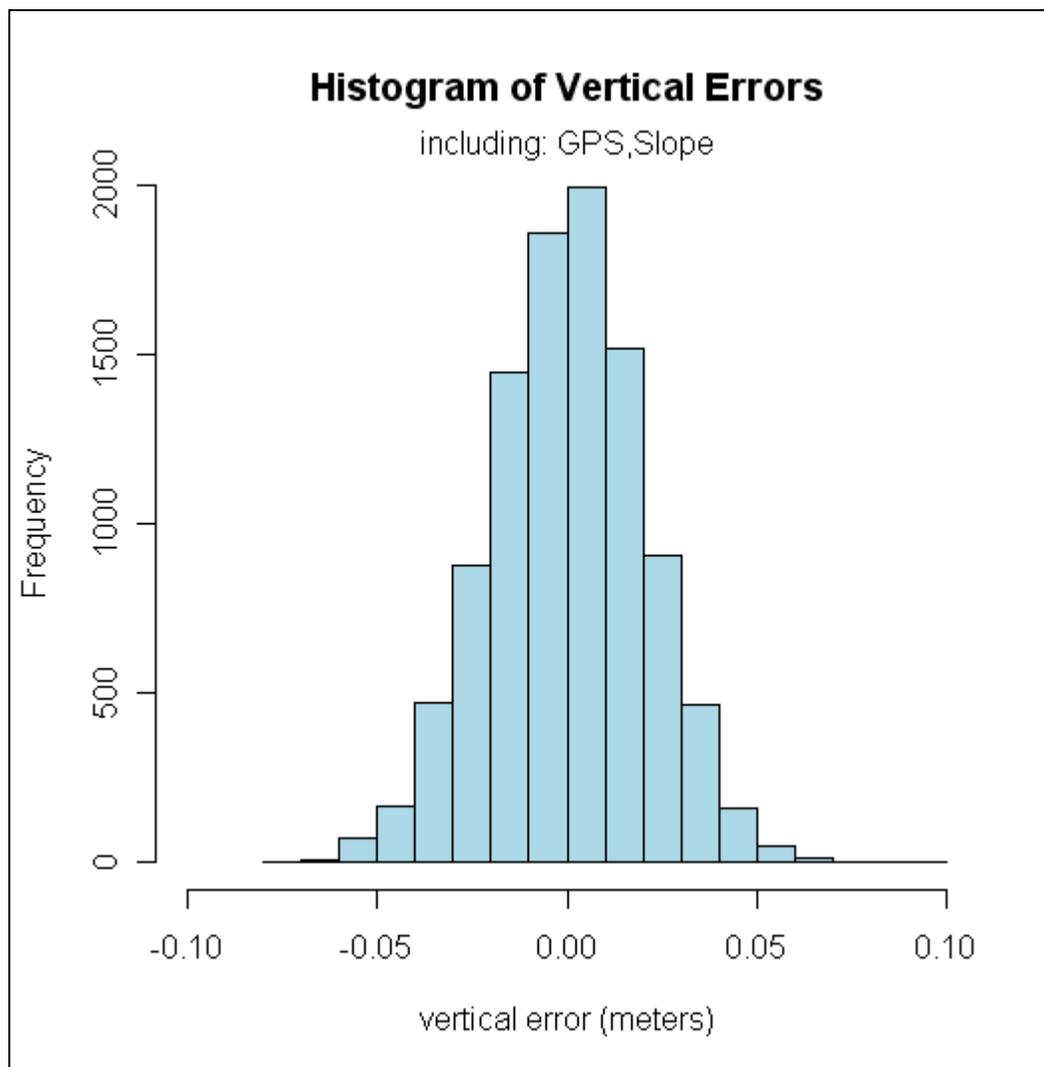


Figure 44 – Histogram of simulated vertical errors caused by ground GPS point accuracy limitations on ground slopes of ± 50 degrees.

A second simulation was run, this time including LIDAR positional error. As described in Table 4, the LIDAR data horizontal accuracy at a flying height of 1000 meters is ± 0.5 meters and the LIDAR data vertical accuracy is ± 0.15 meters. On flat ground, where only vertical equipment accuracy will affect vertical errors, LIDAR elevation errors now ranged from -0.58 meters to 0.69 meters. When slope was included, LIDAR elevation errors were 0.00 meters ± 0.20 meters and ranged from -1.13 meters to 0.93 meters (Figure 45). Because the LIDAR system has much larger possible horizontal and vertical error, slope has a larger effect on vertical errors.

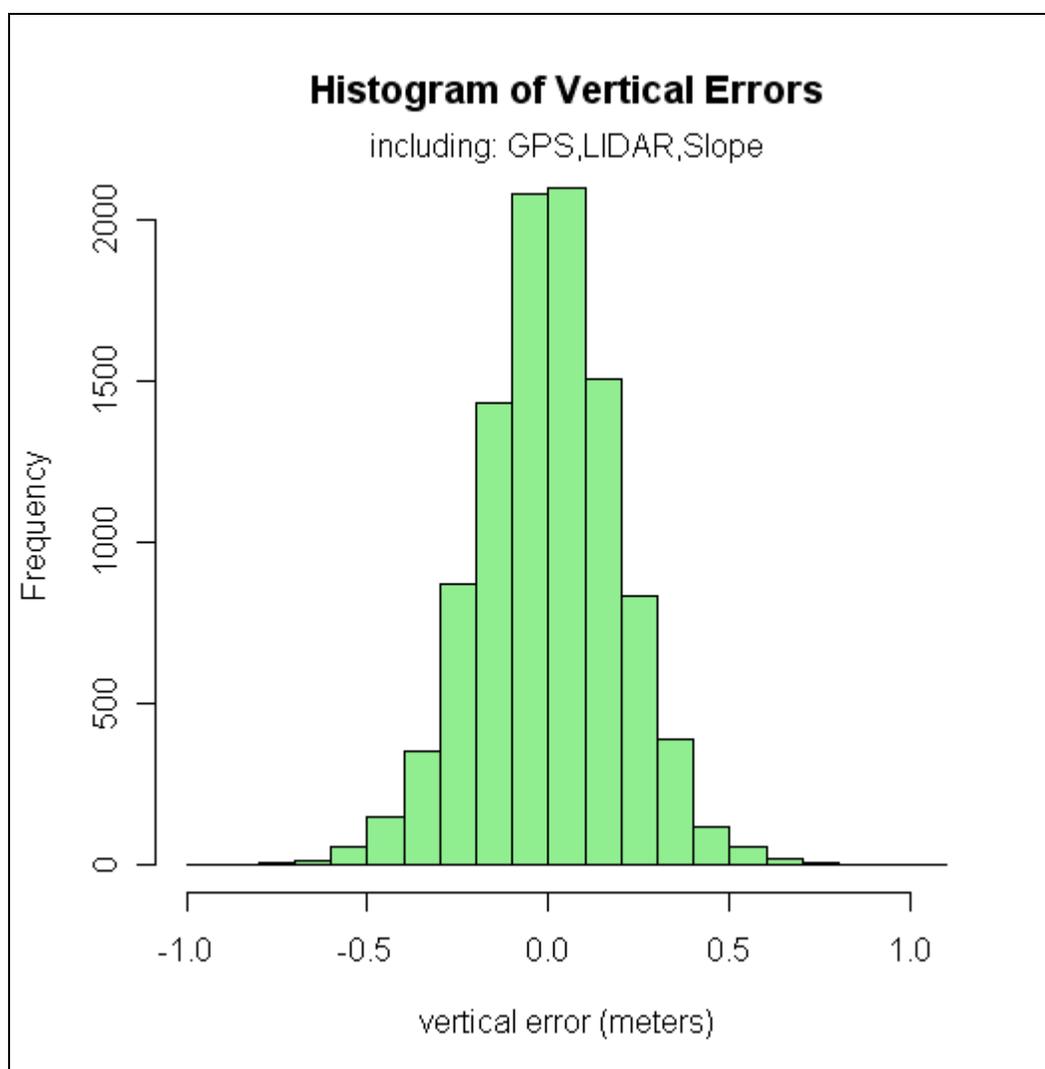


Figure 45 – Histogram of simulated vertical errors caused by ground GPS point and LIDAR system accuracy limitations on ground slopes of ± 50 degrees.

5.2 Triangle Method Accuracy

The Triangle Method for determining LIDAR ground elevation around each GPS point (Section 3.8), was specifically developed for this thesis, and is central to the conclusions. If the Triangle Method is not trustworthy, the conclusions cannot be trusted either. It is therefore important to demonstrate that the Triangle Method can determine ground elevations accurately. There is no way to know what the true ground elevation is, as all DTMs created from the LIDAR will have their own biases, however a useful comparison can be made to the LIDAR DTM created using Fusion for field site selection (Section 3.5.1). If the Triangle Method produced dramatically different ground elevations than Fusion, there may be reason to question its accuracy. Table 17 provides the differences for the two DTMs for the 1709 vegetation GPS points.

Table 17 – LIDAR DTM-GPS ground elevation differences for Fusion and Triangle Method DTMs.

DTM Height Difference Comparison (meters)				
LIDAR DTM-GPS	Height Difference	Fusion DTM-GPS	Triangle Method-GPS	Triangle-FusionDTM
	Average	0.228	0.043	-0.185
	Standard Deviation	0.547	0.613	0.365
	Absolute Value Average	0.360	0.339	0.254
	Maximum	4.726	4.179	0.983
	Minimum	-6.604	-6.889	-3.928

To further explore the accuracy of the Triangle Method, a comparison for an entire site was also undertaken. A one meter resolution DTM was created for the Old 1 site using the Triangle Method, and the Fusion DTM was subtracted to create an elevation difference dataset (Figure 46). The blue-green areas in the Figure (areas where the Triangle Method produced a higher ground elevation) occurred in two situations:

- At the top of embankments along road cuts, where the parameters used in Fusion filtered out LIDAR points along the tops of the embankments resulting in the embankments being rounded-off
- In tall vegetation with large radii where the five meter search radius used in the Triangle Method was smaller than the vegetation, and was not able to find LIDAR returns near the ground

Table 18 displays what percentage of the Old 1 site falls into the various height difference categories. Negative height differences mean that the Triangle Method DTM had lower elevation than Fusion DTM, while positive height differences mean that the Triangle Method DTM had a higher elevation. Approximately 97% of the area in this site had height differences between the two DTM methods of \pm one meter. The height differences are not normally distributed, but rather, are slightly skewed negative (Figure 47). The largest percentage of height differences falls into the negative one to 0.1 meter category, while the second largest percentage is in the -0.1 to 0.1 meter height difference category.

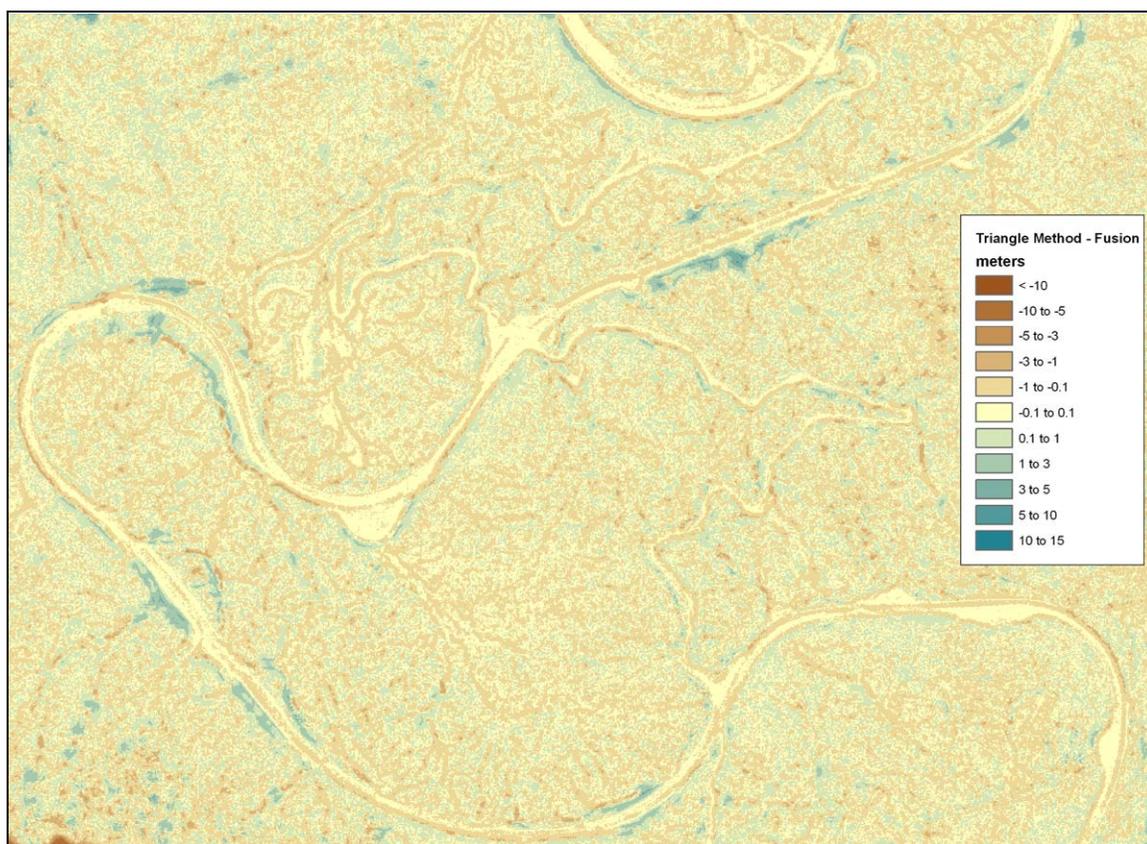
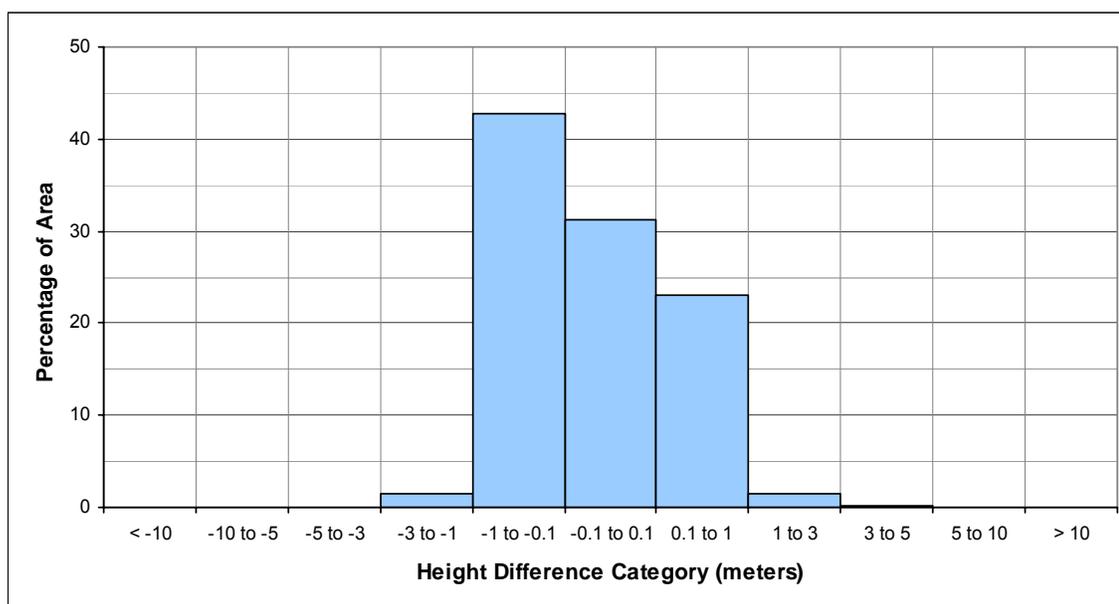


Figure 46 – Map of DTM height differences (Triangle Method – Fusion) in meters for site Old 1.

Table 18 – Percentage of area in Old 1 site in each height difference category.

Percentage of Area in Each Height Difference Category											
Height Difference Category (meters)	< -10	-10 to -5	-5 to -3	-3 to -1	-1 to -0.1	-0.1 to 0.1	0.1 to 1	1 to 3	3 to 5	5 to 10	> 10
Area (m ²)	52	184	148	10932	309814	226660	166867	10433	695	31	0
Percentage of Area	0.01	0.03	0.02	1.51	42.68	31.23	22.99	1.44	0.10	0.00	0.00

**Figure 47** – Histogram of DTM height differences (Triangle Method – Fusion) in meters for site Old 1.

These results suggest, that on average, both DTM methods are very similar in their performance, but that the Triangle method DTM tends to be slightly lower in some conditions. This does not mean that the Triangle Method DTM is more accurate. Changing the parameters in Fusion to filter LIDAR points more or less aggressively, or changing the search radius for the Triangle Method, would likely alter the results.

Overall, this confirms that the Triangle Method LIDAR ground elevations are within believable ranges, and that the results are likely as good as any method could produce.

5.3 Conclusions about the Objectives

In four out of the five field sites used for this work (Old 1, Grand Prix, Piru 1, and Piru 2), the five meter search radius produced the best average Triangle Method results (Figure 48). In the fifth site (Old 2), it did not (Figure 49).

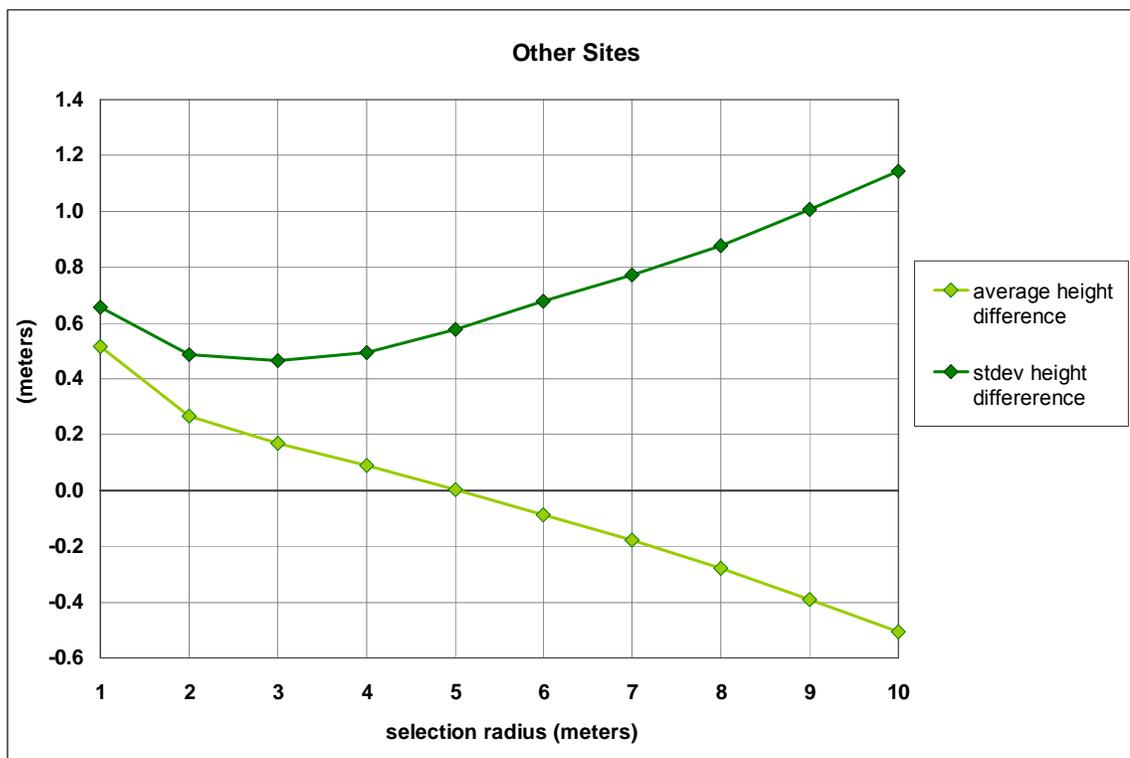


Figure 48 – Average LIDAR-GPS height differences (meters) and Standard Deviations (meters) for Triangle Method selection radii (meters) at sites: Old 1, Grand Prix, Piru 1, and Piru 2.

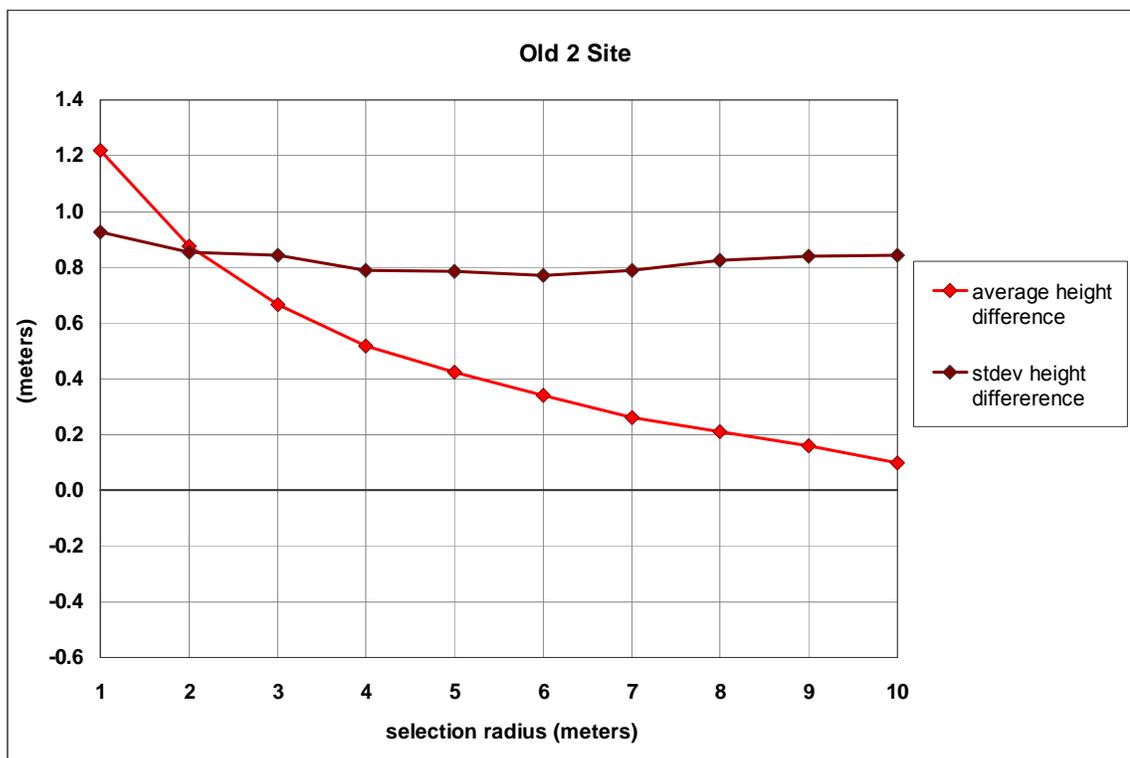


Figure 49 – Average LIDAR-GPS height differences (meters) and Standard Deviations (meters) for Triangle Method selection radii (meters) at site: Old 2.

One possible explanation for this difference is based on the characteristics of the chaparral itself. Search radii smaller than five meters produced larger height differences for all five sites than the five meter radius. The chaparral at site Old 2 was the most continuous and dense of the five sites. The chaparral at the other sites had more gaps and canopy variation (Figure 50). It appears that for the chaparral with gaps, that the diameter of individual plants or plant groupings is around five meters or less. With a search radius smaller than the size of an individual plant or group of plants, only returns from within a plant are selected. It would appear that within a plant LIDAR pulses are impeded from reaching the ground. With a five meter search radius in chaparral with gaps, it is possible that returns from the ground are occurring around the edges of plants. With search radii larger than five meters, the effects of slope and surface variability appear to pull the LIDAR triangle surface below the true ground elevation. For the more continuous chaparral without gaps at site Old 2, the five meter search radius still

produced LIDAR ground models 43 centimeters above the true ground elevation on average.

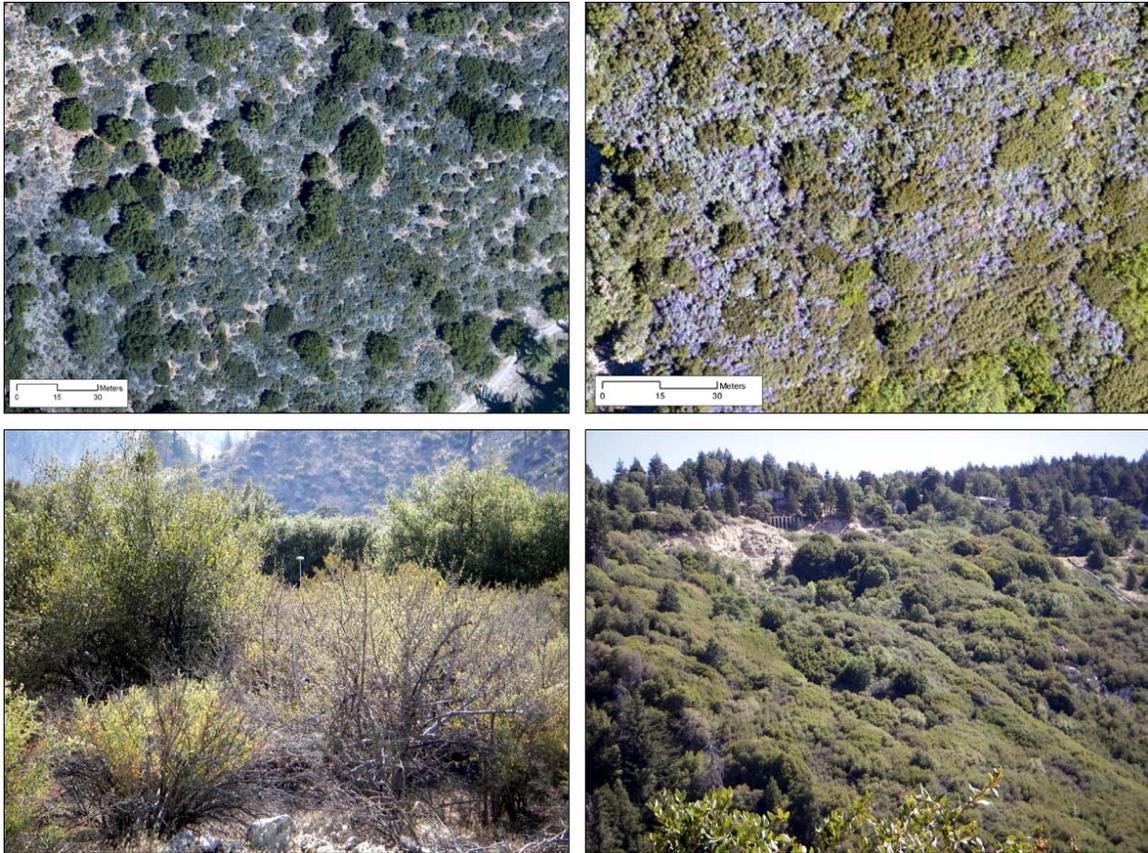


Figure 50 – Overhead views (both top) and side views (both bottom) of Grand Prix (both left) and Old 2 (both right) sites.

Means for all vegetation categories were slightly larger than the average height differences from the simulations in Section 5.1, and the standard deviations of the height differences were much larger than those from the simulations. This suggests that the differences between LIDAR and GPS ground elevations may be due to more than positional accuracy of the equipment. It therefore seems likely that chaparral does prevent LIDAR from measuring true ground positions.

ANOVAs demonstrated that the LIDAR-GPS height difference means have significant differences between some of the height and density categories, both when height and density are considered individually and when they are considered in

combination. This suggests that as vegetation height and density increase, LIDAR-GPS height differences also increase. It therefore seems likely that in all but the shortest and least dense chaparral, laser pulses are often not reaching the ground, but get close. If the purpose of the LIDAR-derived bare ground model is to estimate biomass in chaparral, vegetation heights could be underestimated.

Based on the average LIDAR-GPS height differences for the 2007 and 2005 vegetation height categories in Table 14, the percentages that vegetation heights will be underestimated due to LIDAR DTM errors were calculated (Table 19). The height underestimations range from 0% to 9%. Overall LIDAR DTM accuracy is still high and likely better than other sources of elevation data.

Table 19 – Percentage of vegetation height underestimation due to LIDAR DTM error.

Vegetation Height Category	Vegetation Height (meters)	Average 2007 Height Difference	2007 Height Underestimation (%)	Average 2005 Height Difference	2005 Height Underestimation(%)
0 to 1 m	0.500	-0.032	0.00	-0.002	0.00
	0.750		0.00		0.00
1 to 2 m	1.000	0.030	3.00	0.091	9.10
	1.250		2.40		7.28
	1.500		2.00		6.07
	1.750		1.71		5.20
2 to 3 m	2.000	0.133	6.65	0.118	5.90
	2.250		5.91		5.24
	2.500		5.32		4.72
	2.750		4.84		4.29
	3.000		4.43		3.93
> 3 m	3.250	0.031	0.95	n/a	n/a
	3.500		0.86		n/a
	3.750		0.83		n/a
	4.000		0.78		n/a

Attempts to develop linear regression models to predict LIDAR-GPS height difference based on vegetation height and density were unable to produce an R^2 greater than four percent. This is in large part due to the variability in the LIDAR-GPS height differences for individual points within the categories.

All density and height categories of the vegetation points had mean elevation differences less than or equal to that seen in the clear cut and thinned forest points in the work done by Reutebuch et al. [34]. However, the standard deviations of the height

differences for all categories were larger than those seen for the mature forest points in Reutebuch's work. It appears therefore, that chaparral provides a more difficult operating environment for LIDAR systems than dense, closed-canopy forest.

LIDAR-derived bare ground models have been shown to be more accurate than other sources of elevation data, even in chaparral. However, it may be possible to improve their accuracy further with the use of full waveform LIDAR data, such as that provided by a waveform digitizer. While waveform digitizers still have "dead time" it tends to be on the order of one nanosecond rather than six to 10 nanoseconds [36], which increases the likelihood that a more accurate waveform can be recorded. However, waveform digitizers have typically required the LIDAR systems to operate with lower pulse rates in order to process the waveform. The increased time resolution gained by using a waveform digitizer may be outweighed by the reduction in return density. Work by Gutierrez et al. [19] with a waveform digitizer and an Optech ALTM 1225 also ran into issues with hardware delays between the LIDAR system and the digitizer resulting in errors between the LIDAR data elevations and the digitized waveform elevations of 42 and 76 centimeters. It is possible that these errors could be reduced with further work.

BIBLIOGRAPHY

- [1] Andersen, Hans-Erik, R.J. McGaughey, S.E. Reutebuch. 2005. Estimating forest canopy fuel parameters using LIDAR data. *Remote Sensing of Environment*, 94:441-449.
- [2] Andersen, Hans-Erik, S.E. Reutebuch, R.J. McGaughey. 2006. A rigorous assessment of tree height measurements obtained using airborne lidar and conventional field methods. *Canadian Journal of Remote Sensing*, 32(5): 355-366.
- [3] Anonymous. GGD-112T. Javad Navigation Systems, Inc. From: <http://www.javad.com/jns/index.html?/jns/products/GGD-112T.html> on December 11, 2007.
- [4] Anonymous. Online Positioning User Service. National Geodetic Survey. <http://www.ngs.noaa.gov/OPUS/>.
- [5] Anonymous. 2002. Pinnacle Version 1.00. Javad Navigation Systems, Inc.
- [6] Anonymous. 2003. PG-A1 PG-A2. Topcon Positioning Systems, Inc. From: http://www.topconpositioning.com/uploads/tx_ttopconproducts/PGAsheetREVC.pdf on December 11, 2007.
- [7] Anonymous. 2004. Interactive Data Language (IDL) 6.1. Research Systems, Inc. / ITT Visualization Solutions, Inc.
- [8] Anonymous. 2004. Maxor GNSS Receiver: User's Manual. Javad Navigation Systems, Inc.
- [9] Anonymous. 2004. ALTM 3100 System Specifications. Optech, Inc. From: http://www.optech.ca/pdf/Specs/specs_altm_3100.pdf on December 12, 2007.
- [10] Anonymous. 2005. LAS Specification: Version 1.1. American Society of Photogrammetry and Remote Sensing. From: http://www.asprs.org/society/divisions/ppd/standards/asprs_las_format_v11.pdf on December 12, 2007.
- [11] Anonymous. 2005. LiDAR Data Description: Near Piru Lake, Cajon Junction, Fredalba, California. Watershed Sciences, Inc.
- [12] Anonymous. 2005. R 2.1.0. The R Project for Statistical Computing. <http://www.r-project.org/>.
- [13] Anonymous. 2006. ArcGIS Desktop 9.2. Environmental Systems Research Institute, Inc.

- [14] Baltsavias, E.P. 1999. Airborne laser scanning: basic relations and formulas. *ISPRS Journal of Photogrammetry & Remote Sensing*, 54:199-214.
- [15] Barbour, M.G., T. Keeler-Wolf, A.A. Schoernherr. 2007. *Terrestrial Vegetation of California*. 3rd ed. The University of California Press, Berkeley.
- [16] Clarkin, Tobey. 2007. Modeling global navigation satellite system positional error under forest canopy based on LIDAR-derived canopy densities. MS Thesis, University of Washington, Seattle, WA.
- [17] Goodwin, Nicholas R., N.C. Coops, D.S. Culvenor. 2006. Assessment of forest structure with airborne LiDAR and the effects of platform altitude. *Remote Sensing of Environment*, 103:140-152.
- [18] Grewal, M. S., L. R. Weill, et al. 2001. *Global Positioning Systems, Inertial Navigation, and Integration*, John Wiley & Sons, Inc.
- [19] Gutierrez, Roberto, A. Neuenschwander, M.M. Crawford. 2005. Development of Laser Waveform Digitization for Airborne LIDAR Topographic Mapping Instrumentation. *Proceedings to the IEEE International Geoscience and Remote Sensing Symposium*, 2:1154-1157.
- [20] Haugerud, R.A., D.J. Harding. 2001. Some Algorithms for Virtual Deforestation (VDF) of LIDAR Topographic Survey Data. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XXXIV-3/W4, 211-217.
- [21] Hodgson, Michael E., J.R. Jensen, L. Schmidt, S. Schill, B. Davis. 2003. An evaluation of LIDAR- and IFSAR-derived digital elevation models in leaf-on conditions with USGS Level 1 and Level 2 DEMs. *Remote Sensing of Environment*, 84:295-308.
- [22] Hopkinson, C., L.E. Chasmer, G. Zsigovics, I.F. Creed, M. Sitar, P. Treitz, R.V. Maher. 2004. Errors in LiDAR ground elevation and wetland vegetation height estimates. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XXXVI Part 8/W2, 108-113.
- [23] Kraus, K. and N. Pfeifer. 1998. Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS Journal of Photogrammetry & Remote Sensing*, 53: 193-203.
- [24] Latypov, Damir. 2002. Estimating relative lidar accuracy information from overlapping flight lines. *ISPRS Journal of Photogrammetry & Remote Sensing*, 56: 236-245.

- [25] McGaughey, Robert J. 2007. FUSION/LDV: Software for LIDAR Data Analysis and Visualization. From: http://forsys.cfr.washington.edu/fusion/FUSION_manual.pdf on December 12, 2007.
- [26] Miles, S. R. and C. R. Goudey. 1997. Ecological subregions of California: Section and Subsection Descriptions. *USDA, Forest Service, Pacific Southwest Region, R5-EM-TP-005*. Also: <http://www.fs.fed.us/r5/projects/ecoregions/>
- [27] Minnich, R.A., and Y.H. Chou. 1997. Wildland Fire Patch Dynamics in the Chaparral of Southern California and Northern Baja California. *International Journal of Wildland Fire*, 7(3):221-248.
- [28] Moritz, M.A., J.E. Keeley, E.A. Johnson, and A.A. Schaffner. 2004. Testing a basic assumption of shrubland fire management: how important is fuel age? *Frontiers in Ecology and the Environment*, 2(2): 67-77.
- [29] Mundt, Jacob T., D.R. Streutker, N.F. Glenn. 2006. Mapping Sagebrush Distribution Using Fusion of Hyperspectral and Lidar Classifications. *Photogrammetric Engineering & Remote Sensing*, 72(1):47-54.
- [30] Næsset, Erik. 1997. Estimating Timber Volume of Forest Stands Using Airborne Laser Scanner Data. *Remote Sensing of Environment*, 61:246-253.
- [31] Næsset, Erik, Kjell-Olav Bjercknes. 2001. Estimating tree heights and number of stems in young forest stands using airborne laser scanner data. *Remote Sensing of Environment*, 78:328-340.
- [32] Riaño, David, E. Meier, B. Allgöwer, E. Chuvieco, S.L. Ustin. 2003. Modeling airborne laser scanning data for the spatial generation of critical forest parameters in fire behavior modeling. *Remote Sensing of Environment*, 86:177-186.
- [33] Riaño, David, E. Chuvieco, S. Condés, J. González-Matesanz, S.L. Ustin. 2004. Generation of crown bulk density for *Pinus sylvestris* L. from lidar. *Remote Sensing of Environment*, 92:345-352.
- [34] Reutebuch, Stephen E., R.J. McGaughey, H.E. Andersen, W.W. Carson. 2003. Accuracy of a high-resolution lidar terrain model under a conifer forest canopy. *Canadian Journal of Remote Sensing*, 29(5): 527-535.
- [35] Shreshta, R., W. Carter, C. Slatton, W. Dietrich. 2007. "Research Quality" Airborne Laser Swath Mapping: The Defining Factors, Version 1.2. National Center for Airborne Laser Mapping, Gainesville, FL. From: http://www.ncalm.ufl.edu/publication_pdf/NCALM_WhitePaper_v1.2.pdf on March 12, 2008.

- [36] Slatton, Clint, and W.Carter. 2008. A Lidar Primer. National Center for Airborne Laser Mapping, Gainesville, FL. From: http://www.ncalm.ufl.edu/publication_pdf/A_Lidar_Primer.pdf on March 12, 2008.
- [37] Slatton, Clint K., W.E. Carter, R.L. Shreshta, W. Dietrich. 2007. Airborne Laser Swath Mapping: Achieving the resolution and accuracy required for geosurficial research. *Geophysical Research Letters*, 34(L23S10).
- [38] Streutker, David R., N.F. Glenn. 2006. LiDAR measurement of sagebrush steppe vegetation heights. *Remote Sensing of Environment*, 102:135-145.
- [39] Töyrä, Jessika, A. Pietroniro, C. Hopkinson, W. Kalbfleisch. 2003. Assessment of airborne scanning laser altimetry (lidar) in a deltaic wetland environment. *Canadian Journal of Remote Sensing*, 29(6): 718-728.

Appendices

Appendix A: Triangle Method Code 1

```

#!/usr/bin/perl

my $total_time_start=time(); #Start time for entire program

#-----NOTES/ABOUT-----
#
# Written by: Andrew Cooke -- agcooke@u.washington.edu
#           of the Precision Forestry Cooperative
#           College of Forest Resources, University of Washington
#
# as part of my Master's Thesis work. Thesis title:
# Analysis of LIDAR-derived bare ground model accuracy in southern
# California chaparral. Available at the UW Libraries.
# http://www.lib.washington.edu/
#
# Also part of work for the USDA Forest Service PNW Research Station
# funded by the Joint Fire Science Program, http://www.firescience.gov/
# JFSP Project Number: 04-1-2-02
#
# Started: September 21, 2007
# LAST REVISION: APRIL 7, 2008
#
# Program written in Perl version 5.8.7 for MSWin32-x86-multi-thread
# from ActiveState, http://www.activestate.com/Products/activeperl/
# run on Windows XP Service Pack 2
# System is Little-Endian.
#
#-----INPUTS-----
#
# 1. File with GPS coordinates. All that is really needed are the X,Y,Z
# coordinates for each GPS point. However, for this project other
# fields were included. The code expects these other fields and must
# be modified if they are not included. You are responsible for this.
# GPS points must be within the area of LIDAR coverage, or you will
# get no results.
#
# 2. LIDAR file. LAS Versions 1.0,1.1 should both work. ASCII data,
# should be X,Y,Z or X,Y,Z,I comma delimited, one point per line,
# no header line, file extensions: asc, csv, xyz, txt.
#
#-----OUTPUTS-----
#
# 1. Method4results_20degRotations.asc
# Poorly named triangle method results with header line.
# LIDAR_Z = the triangle method ground elevation. Bins = the 3
# azimuths of the 3 bin lines used to get the LIDAR_Z. The other fields
# should be self-explanatory.
#
# 2. CircleMethod.asc
# Poorly named columnar minimum results. This method finds the lowest
# LIDAR return within 1 meter of the GPS coordinates and uses that as
# the LIDAR_Z. Number of points = the number of LIDAR returns in the
# 1 meter circle.
#
#-----PROCESS-----
#
# 1. Read in the GPS points, find their X and Y extent. Buffer this extent

```

```

#   by 20 meters. Buffer should be larger if search radius is 20m or
#   larger.
#
# 2. Check if an indexed.asc file exists.
#   A. If it does load it into %HoA.
#   B. If not ask user for indexing cellsize and load LIDAR file in
#       $lidarPath. As file is read in only load LIDAR points inside of
#       GPS extent into @lidar ignore others. Index @lidar into %HoA.
#       Ask user if they want to save %HoA to indexed.asc
#
# 3. Calculate LIDAR-GPS height differences for each GPS point, and save to
#   files.
#
#
#-----IMPORTANT-----
#
# 1. GPS and LIDAR points must both have the same type of elevation...
#   ellipsoid height, orthometric height, geoid height, etc. for a
#   meaningful comparison.
#
# 2. All data used in this project was in UTM meters. What will happen
#   if other units or coordinate systems are used is unknown. Good Luck.
#
# 3. The search radius for the triangle method was in meters for my work,
#   the same units as the data. It is not hard-coded as meters, but again
#   what will happen if other units are used is unknown. Good Luck.
#
# 4. The triangle method currently uses the LIDAR data in the same cell as
#   the GPS point, and the 8 surrounding cells, if they exist. If the
#   search radius for the triangle method is larger than the cellsize,
#   the circle will extend outside of the LIDAR passed to the triangle
#   method. Not all of the LIDAR data that should be used will be used.
#   It is important that the search radius be the same size or smaller
#   than the indexed cell size.
#
# 5. The GPS_Z elevation in the output is the original from the input. The
#   LIDAR_Z is as calculated. The local offset correction term is not in
#   the output file. The LIDAR-GPS elevation difference != LIDAR_Z - GPS_Z
#   in the output file. HTDIF = LIDAR_Z - (GPS_Z - local offset).
#   The local offset values are for each site and are in an array:
#       @mhd_cpt_hdif
#   in method4_FAST_Rotate_azimuth_by_x_degrees ('mainHtDif')
#
#-----FUTURE IMPROVEMENTS TO BE MADE IF TIME ALLOWS-----
#
# 1. There is only 1 indexed.asc file currently. Every time you index new
#   data, this file is overwritten. Fix that.
#
# 2. The output files are always named the same thing and are overwritten.
#   You currently must rename them if you want them saved when you re-run
#   the program.
#
# 3. The indexed.asc file is ASCII. Saving to a binary could save some
#   space, but introduces a new custom file format. Not sure if I want
#   to get into this.
#
# 4. Modify the 'which cells needed' subroutine to adaptively select
#   needed index cells depending on search radius. If cellsize <= radius
#   select more surrounding index cells.
#
# 5. Remove the 'remove duplicates' subroutine from:
#   method4_FAST_Rotate_azimuth_by_x_degrees ('mainHtDif')
#
#-----
#---REQUIRED MODULES/ETC.---

```

```

use strict;
use POSIX qw(ceil floor);
use Math::Trig;
use htDif('mainHtDif'); #the subroutine that will build the LIDAR ground, and
                        #calculate the height differences.
use Cwd; #To get current working directory
$| =1; #FOR OVERWRITING PROGRESS

#---VARIABLES THAT NEED GLOBAL SCOPE
my %HoA = (); #Hash of Arrays. Each index cell gets an array in
              #this hash. There are numXCells*numYCells index
              #cells, each with a unique ID used as the hash
              #key.
my (@lidarraw)=(); #Array to hold LIDAR data read in from file.
my (@gps)=(); #Array to hold GPS data read in from file.
my $cellsize; #the size of the index cells (they are square, so
              #this is both length and width)
my $ls=0; #the number of LIDAR points loaded from all LIDAR
          #files.
my $stx; #
my $sty; #
my $numXCells; #The number of index cells in the X (easting)
              #direction for the LIDAR data and cellsize.
my $numYCells; #The number of index cells in the Y (northing)
              #direction for the LIDAR data and cellsize.
my $numTotCells; #The total number of index cells needed for the
                 #LIDAR data and cellsize.
my (@indexCellChoices)=(); #Array to hold the numbers of the index cells
                           #surrounding each GPS point. For each GPS point,
                           #get the LIDAR in these cells for further
                           #analysis.
my (@LIDARSelectedCells)=(); #Array to hold the LIDAR for the cells in
                              #indexCellChoices.
my $calcElevDif; #do you want to just index new data, or do you
                 #want to run the LIDAR-GPS calculations?

#-----
# SET USER PARAMETERS START

#---FILES WITH DATA---
#IF you're using indexed data in the indexed.asc file, you don't need a proper
#path to the LIDAR FILE here. If you're reading in new ASCII or LAS data you do

my $gpsPath="path/to/gps/directory ";
my $lidarPath="path/to/lidar/directory";

#---TRIANGLE METHOD SEARCH RADIUS---
my $radius = 5; #the search radius the triangle method uses. LIDAR pts.
               #this far from GPS should be included.

#---IF YOU WANT TO FILTER OUT SOME LIDAR POINTS FROM AN LAS FILE
#THAT ARE EITHER TOO HIGH OR LOW, SET THESE, OTHERWISE LEAVE EMPTY.
#THIS ONLY APPLIES TO READING IN LAS FILES. IF LEFT EMPTY, THEY WILL
#BE SET TO THE MIN/MAX Z VALUES FROM THE LAS HEADER
#these are modified in the subroutine readLASHeader in the subroutine
#loadNewDataLas.
my $lowerlimit; # = 1157.9; #TO REMOVE WEIRD LOW POINTS FROM OLD 1 DATA
my $upperlimit;

# SET USER PARAMETERS END
#-----

```

```

#---PRINT INTRO STUFF-----
print "\n";
print "\n";
print "_____\n";
print "_____\n";
print "CALCULATE LIDAR-GPS ELEVATION DIFFERENCES\n";
print "\n";

my(@lp)=split(/\/\/,$lidarPath);
print "LIDAR File\: $lp[$#lp]\n";
print "    In DIR\: ";
for my $lpi (0 .. $#lp-1){
    print "$lp[$lpi]\n";
}
print "\n";
my(@gp)=split(/\/\/,$gpsPath);
print "    GPS File\: $gp[$#gp]\n";
print "    In DIR\: ";
for my $gpi (0 .. $#gp-1){
    print "$gp[$gpi]\n";
}
print "\n";

#---TEST THAT INPUT DATA FILES EXIST---
if(! -e $gpsPath){
    print "GPS file does not exist\n";
    exit;
}
if(! -e $lidarPath){
    print "LIDAR file does not exist\n";
    exit;
}
print "\n";
print "Triangle Method Search Radius: $radius\n";
print "_____\n";
print "\n";
print "\n";

#-----
#           THE PROGRAM START
#-----

# 1 ---LOAD GPS DATA---
my(@gpsExtent)=loadGPS($gpsPath);

#-----
# 2 ---NEW(N) OR SAVED-INDEXED(S) FILE---
my(@ldrpth)=split(/\/\/,$lidarPath);
#test to see if there is an 'indexed.asc' file already.
my $indexedascExistsYN=indexedExists($ldrpth[$#ldrpth]);

my $wfns;
if($indexedascExistsYN){
    $wfns = whichFileNS(); #'indexed.asc' exist, ask user if they want to use it.
}else{
    print "Saved indexed data (s) does not yet exist, use new non-indexed data (n)\n";
    #'indexed.asc' does not exist, tell user, then go
    #straight to New Data.
    $wfns = "n";
}

#-----

```

```

# 3 ---READ IN NEW DATA OR USE INDEXED.ASC---
if($wfns eq "n"){
  #READ IN NEW DATA

  #ASK USER WHAT SIZE INDEX CELLS TO USE
  getCellsize();

  #STOP THE PROGRAM UNLESS THE CELLSIZE IS >= SEARCH RADIUS
  if($radius > $cellsize){
    print "The triangle method search radius is larger than the cellsize used for indexing
the LIDAR data. It must be less than or equal to. Re-index the LIDAR data, or reduce the
search radius\n";
    print "    search radius: $radius\n";
    print "indexing cellsize: $cellsize\n";
    exit;
  }

  #LOAD AND FILTER NEW LIDAR DATA FILE
  my($junk,$loadType) = split(/\.\/,$lp[$#lp]);
  if(($loadType eq "asc") || ($loadType eq "txt") || ($loadType eq "xyz") || ($loadType
eq "csv")){
    loadNewFileAsc(@gpsExtent);          #Load ASCII file in $lidarPath into @lidarraw
  }elseif($loadType eq "las"){
    loadNewFileLas(@gpsExtent);          #Load LAS file in $lidarPath into @lidarraw
  }else{
    print "Unknown file type: $loadType\n";
    print "File needs to be ASCII comma delimited X,Y,Z -- or -- LAS format\n";
    print "File extension should be: asc, txt, xyz, csv, las\n";
    exit;
  }

  if($ls>0){
    #INDEX DATA IN @LIDARRAW, LOADING %HoA
    indexNewData(@gpsExtent);
  }else{
    print "no data to index\n";
    exit;
  }

  my $s = saveYN();
  if($s){
    saveIndexedHashtoASCII(); #Save data to an indexed ASCII?
  }

  $calcElevDif=calcYN();
}
if($wfns eq "s"){
  #USE INDEXED.ASC
  loadIndexedASCIIToHash(); #load LIDAR data already sorted and saved into
                             #ASCII file

  if(!$cellsize){
    getCellsize();          #for some reason, indexed data doesn't have
                             #cellsize associated...ask user what size index
                             #cells to use. Won't work unless cellsize here is
                             #same as used to index.
  }else{
    #have the cellsize used to index the data.
    print "cellsize from indexed data to be used\: $cellsize\n";
  }

  #STOP THE PROGRAM UNLESS THE CELLSIZE IS >= SEARCH RADIUS
  if($radius > $cellsize){
    print "\n";
    print "The triangle method search radius is larger than the cellsize used for indexing
the LIDAR data. It must be less than or equal to. Re-index the LIDAR data, or reduce the
search radius\n";
    print "    search radius: $radius\n";
  }
}

```

```

    print "indexing cellsize: $cellsize\n";
    exit;
}

if(!$stx && !$sty && !$numXCells && !$numYCells && !$numTotCells){
    indexExistingHOA ();          #FOR SOME REASON, EXTENT DATA NOT SAVED, take
                                #data and figure out extents etc.
}
else{
    print "STX:\:$stx, STY:\:$sty, NUMX:\:$numXCells, NUMY:\:$numYCells,
TOTCELLS:\:$numTotCells\n";
}

$calcelevDif = 1;                #CALCULATE LIDAR-GPS ELEVATION DIFFERENCES
}

#-----
# 4 CALCULATE THE ELEVATION DIFFERENCES
if($calcelevDif){
    #---CALULATE---
    my $main_time_start=time();
    print "\n";
    print "_____ \n";
    print "\n";
    print "Calculate Height Differences \.\.\.\n";
    print "\% Complete -- Est. Total Time -- Time Elapsed -- Time Remaining\n";

    my $outfile = "Method4results_20degRotations.asc";
    my $outcfile = "CircleMethod.asc";
    open(OUT, "> $outfile");
    print OUT
("SITE,GPS_NAME,GPS_X,GPS_Y,GPS_Z,LIDAR_Z,HTDIF,BINS,VEGHT_(m),VEGDENS,NUMEPOCH,AVGSAT,AV
GPDOP,TYPE,RMS_(m),NumLidarInRadius\n");

    #---FOREACH GPS POINT START---
    my($tothr,$totmn,$totsc);
    my $calcprogdenom=@gps;
    for my $calccount(0..$#gps){

        #---PROGRESS START---
        #PERCENT COMPLETED
        my $prog= (100*($calccount/$calcprogdenom));
        $prog = sprintf("%.2f", (floor($prog*100))/100 );
        $prog=" " . $prog if($prog<10);
        #TIME
        my $ratedenom=$calccount;
        $ratedenom=1 if($ratedenom<1);
        my $ti=(time()-$main_time_start);
        my $rate = $ti/$ratedenom;
        my $totaltime=sprintf("%.0f", ($calcprogdenom*$rate));
        #TOTAL
        $tothr = ($totaltime/3600)%3600;
        $totmn = ($totaltime/60)%60;
        $totsc = $totaltime-(3600*$tothr)-(60*$totmn);
        $tothr="0".$tothr if($tothr<10);
        $totmn="0".$totmn if($totmn<10);
        $totsc="0".$totsc if($totsc<10);
        #ELAPSED
        my $elaphr=($ti/3600)%3600;
        my $elapmn=($ti/60)%60;
        my $elapsc=$ti-(3600*$elaphr)-(60*$elapmn);
        $elaphr="0".$elaphr if($elaphr<10);
        $elapmn="0".$elapmn if($elapmn<10);
        $elapsc="0".$elapsc if($elapsc<10);
        #REMAINING
        my $remainingtime=$totaltime-$ti;
        my $remhr = ($remainingtime/3600)%3600;
        my $remmn = ($remainingtime/60)%60;
    }
}

```

```

my $remsc = $remainingtime-(3600*$remhr)-(60*$remmn);
$remhr="0".$remhr if($remhr<10);
$remmn="0".$remmn if($remmn<10);
$remsc="0".$remsc if($remsc<10);
print "$prog \%      -- $tothr\:$totmn\:$totsc      -- $elaphr\:$elapmn\:$elapsc
-- $remhr\:$remmn\:$remsc\r";
#---PROGRESS END-----

#---GET LIDAR FOR EACH GPS POINT---

my($sigpoint,$sigs,$sigy,$sigx,$sigz_ell,$sigz_orth,$sigvght,$sigvgdns,$signumep,$sigavgst,$sigavgpd
p,$sigtype,$sigrms) = split(/\,/,$gps[$scalccount]);
whichCellsNeeded($sigx,$sigy);

#---GET LIDAR GROUND ELEVATION ETC., PRINT TO FILES---

my(@res)=mainHtDif($radius,$sigpoint,$sigs,$sigy,$sigx,$sigz_ell,$sigz_orth,$sigvght,$sigvgdns,$i
gnumep,$sigavgst,$sigavgpd,$sigtype,$sigrms,@LIDARSelectedCells);
print OUT
("$res[0],$res[1],$res[2],$res[3],$res[4],$res[5],$res[6],$res[7],$res[8],$res[9],$res[10]
,$res[11],$res[12],$res[13],$res[14],$res[15]\n");
}
#---FOREACH GPS POINT END-----

close (OUT);

my $ti=(time()-$main_time_start);
my $hr = ($ti/3600)%3600;
my $min = ($ti/60)%60;
my $sec = $ti-(3600*$hr)-(60*$min);
$hr="0".$hr if($hr<10);
$min="0".$min if($min<10);
$sec="0".$sec if($sec<10);
print "\r100.00 \%      -- $tothr\:$totmn\:$totsc      -- $hr\:$min\:$sec      --
00\:00\:00";
print "\n";
}

my $ti=(time()-$total_time_start);
my $hr = ($ti/3600)%3600;
my $min = ($ti/60)%60;
my $sec = $ti-(3600*$hr)-(60*$min);
$hr="0".$hr if($hr<10);
$min="0".$min if($min<10);
$sec="0".$sec if($sec<10);
print "\n";
print "Total Runtime\: $hr\:$min\:$sec\n";

exit;
#-----
#           THE PROGRAM END
#-----

```

```

#-----
#                SUBROUTINES
#-----

#\-----
#---ASK USER WHETHER OR NOT TO RUN CALCULTIONS---
sub calcYN {
  print "Do you want to calculate LIDAR-GPS height differences (y or n): ";
 .chomp(my $save = <>);

  print "\n";
  if(($save ne "n") && ($save ne "y")){
    calcYN();
  }else{
    if($save eq "n"){
      return 0;
    }else{
      return 1;
    }
  }
}
#\-----

```

```

#\-----
#---ASK USER WHAT SIZE INDEX CELL TO USE---
sub getCellsize {
  print "\n";
  #(in units of data, meters/feet. etc.)

  if($cellsize){
    print "Current cell size\: $cellsize\n";
  }else{
    print "indexing cell size is undefined\n";
  }
  print "What size cells for indexing: ";
 .chomp(my $whichsize = <>);
  print "You entered: $whichsize\n";
  my $res=getnum($whichsize);
  if($res){
    if($res<0){
      print "Error: cellsize less than 0\n";
      print "\n";
      getCellsize();
    }else{
      $cellsize = $res;
      print "cellsize now\: $cellsize\n";
    }
  }else{
    print "Error: not a number\n";
    print "\n";
    getCellsize();
  }
  print "_____ \n";
  print "\n";
}
#\-----

```

```

#\-----
#---MAKE SURE WHAT USER TYPED IS A NUMBER---

```

```

sub getnum {
    use POSIX qw(strtod);
    my $str = shift;
    $str =~ s/^\s+//;
    $str =~ s/\s+$//;
    $! = 0;
    my($num, $unparsed) = strtod($str);
    if (($str eq '') || ($unparsed != 0) || $!) {
        return;
    } else {
        return $num;
    }
}
#\-----

#\-----
#---IS THERE AN indexed.asc FILE IN THE DIRECTORY?
#---If not, there's no reason to offer choice (s)
#---GO straight to (n)
sub indexedExists{
    my($lidarsite) = @_;

    #my $datadir = getcwd; #get current working directory
    #OR
    my $datadir = "path/to/lidar/directory";

    opendir(FILE, "$datadir");
    my(@filenames) = readdir(FILE);
    closedir(FILE);
    my(@alpha) = sort(@filenames);

    #---remove the 1st 2 elements---
    shift(@alpha);
    shift(@alpha);

    #my $ck="indexed.asc";
    my(@ckresults)=grep(/^$lidarsite$/,@alpha);
    my $ckressize=@ckresults;
    if($ckressize>0){
        print "$lidarsite exists\n";
        return 1;
    }else{
        return 0;
    }
}
#\-----

#\-----
#---ONCE EXISTING, INDEXED ASCII DATA IS LOADED INTO %HOA, FIND
#---EXTENTS/EDGES/ETC. FOR FURTHER PROCESSING---
sub indexExistingHoA {
    my $sub_time_start=time();

    print "Get Data extents\n";

    #---FIND GRID EDGES, Xmin/max, Ymin/max, ETC.---
    my(@keys) = (keys %HoA);

    my $kmax=$keys[0];
    my $kmin=$keys[0];
    foreach(@keys){
        $kmax=$_ if($_>$kmax);

```

```

    $kmin=$_ if($_<$kmin);
}

my(@tmp) = @{ $HoA{$kmin} };
my($x,$y,$z)=split(/\./,$tmp[0]);
my $xmin=$x;
my $xmax=$x;
my $ymin=$y;
my $ymax=$y;

for my $q ($kmin .. $kmax){
    if(defined $HoA{$q}){
        my(@tmp) = @{ $HoA{$q} };
        foreach(@tmp){
            my($x,$y,$z)=split(/\./,$_);
            $xmin=$x if($x<$xmin);
            $ymin=$y if($y<$ymin);
            $xmax=$x if($x>$xmax);
            $ymax=$y if($y>$ymax);
        }
    }
}

my $xdist = $xmax - $xmin;
my $ydist = $ymax - $ymin;

$stx=floor($xmin);
$xdist=$xmax-$stx;
$numXCells=ceil($xdist/$cellsize);
my $enx=$stx+($numXCells*$cellsize);
$sty=floor($ymin);
$ydist=$ymax-$sty;
$numYCells=ceil($ydist/$cellsize);
my $eny=$sty+($numYCells*$cellsize);

$numTotCells=$numXCells*$numYCells;

print "EXTENTDATA: $stx,$sty,$numXCells,$numYCells,$numTotCells\n";

my $ti=(time()-$sub_time_start);
my $hr = ($ti/3600)%3600;
my $min = ($ti/60)%60;
my $sec = $ti-(3600*$hr)-(60*$min);
$hr="0".$hr if($hr<10);
$min="0".$min if($min<10);
$sec="0".$sec if($sec<10);
print "$hr\":$min\":$sec\n";

print "\n";
}
#/\-----

#/\-----
#---FIND EXTENTS OF LIDAR DATA, CREATE A GRID USING EXTENTS/CELLSIZE, AND FIND
#---THE INDEX NUMBER IN THE GRID FOREACH LIDAR POINT. LOAD LIDAR INTO %HOA
#---USING INDEX NUMBER AS KEY---
sub indexNewData {
    my $sub_time_start=time();

    print "\n";
    print "indexing new data and loading HoA\n";

    #USE GPS BASED FILTERING EXTENTS FROM loadNewDataAsc/Las
    my($xmin,$xmax,$ymin,$ymax)=@_;

    $stx=floor($xmin);

```

```

my $xdist=$xmax-$stx;
$numXCells=ceil($xdist/$cellsize);
my $enx=$stx+($numXCells*$cellsize);
$sty=floor($ymin);
my $ydist=$ymax-$sty;
$numYCells=ceil($ydist/$cellsize);
my $eny=$sty+($numYCells*$cellsize);

$numTotCells=$numXCells*$numYCells;

#---LOAD HoA WITH NEW LIDAR DATA-----
#---FIND INDEX NUMBER in index grid foreach LIDAR point---
#---Lower left corner is cell 1, upper right cell is numXCells*numYCells---
for my $i (0 .. $#lidarraw){
  my ($x,$y,$z)=split(/\./,$lidarraw[$i]);
  chomp($z); # $z is last term in line, has a \n. remove it.

  my $xdist = (floor(($x-$stx)/$cellsize))+1;
  my $ydist = (floor(($y-$sty)/$cellsize))+1;

  my $base=($ydist-1)*$numXCells;
  my $index=($xdist+$base);

  #TEST IF INDEX EXISTS IN HASH ALREADY - YES:PUSH,NO:ADD
  #EACH ITEM OF HASH IS AN ARRAY
  #EACH ITEM IN ARRAY IS A COMMA DELIMITED X,Y,Z STRING FOR LIDAR POINT
  my $addstring = $x.", ".$y.", ".$z;
  if (exists $HoA{$index}){
    push @{$HoA{$index}}, $addstring; #ADD ANOTHER ITEM TO ARRAY
  }else{
    $HoA{$index} = [ $addstring ]; #CREATE AN ARRAY FOR INDEX, MAKE
    #FIRST ELEMENT
  }
}

my $ti=(time()-$sub_time_start);
my $hr = ($ti/3600)%3600;
my $min = ($ti/60)%60;
my $sec = $ti-(3600*$hr)-(60*$min);
$hr="0".$hr if($hr<10);
$min="0".$min if($min<10);
$sec="0".$sec if($sec<10);
print "$hr\":$min\":$sec\n";

print "\n";
}
#/\-----

#\-----
#---OPEN AND LOAD GPS POINTS INTO MEMORY---
sub loadGPS {
  print "Load GPS points\n";

  my($gfile) = @_ ;
  print "From: $gfile \.\.\.";
  print "\n";
  print "\n";

  #---GPS DATA FORMAT---
  #($gpoint,$gs,$gy,$gx,$gz_ell,$gz_orth,$gvght,$gvgdns,$gnumep,$gavgst,$gavgpdp,
  # $gtype,$grms)
  #point,site,Y,X,Z_ellipsoids,Z_orthometric,VegHt,VegDense,NumEpochs,
  #AvgSatellites,AvgPDOP,dataType,RMS

```

```

#GET GPS EXTENTS
#print "GPS Extents for: $gfile \.\.\.\n";
#GET X/Y VALUES OF 1ST GPS POINT
my($minY,$minX,$maxY,$maxX);
open(ING, "< $gfile");
my $tmp = (<ING>);
my
($sigpoint,$sigs,$igy,$igx,$igz_ell,$igz_orth,$igvght,$igvgdns,$ignumep,$igavgst,$igavgpdp,
$igtype,$igrms) = split(/\./,$tmp);
    $minX=$igx;
    $maxX=$igx;
    $minY=$igy;
    $maxY=$igy;
close (ING);
#GO THROUGH ALL GPS POINTS AND GET EXTENTS
open(ING, "< $gfile");
    #---GPS DATA FORMAT---
    #($gpoint,$gs,$gy,$gx,$gz_ell,$gz_orth,$gvght,$gvgdns,$gnumep,$gavgst,$gavgpdp,
#$gtype,$grms)
#print,site,Y,X,Z_ellipsoids,Z_orthometric,VegHt,VegDense,NumEpochs,
#AvgSatellites,AvgPDOP,dataType,RMS
    while(<ING>){

($sigpoint,$sigs,$igy,$igx,$igz_ell,$igz_orth,$igvght,$igvgdns,$ignumep,$igavgst,$igavgpdp,
$igtype,$igrms) = split(/\./,$_);
        $minX=$igx if($igx<$minX);
        $minY=$igy if($igy<$minY);
        $maxX=$igx if($igx>$maxX);
        $maxY=$igy if($igy>$maxY);
    }
close (ING);

#ADD BUFFER
my $bufferSize=20;
print "GPS positions buffered by $bufferSize meters\n";
print "\n";
$minX=(floor($minX)-$bufferSize);
$minY=(floor($minY)-$bufferSize);
$maxX=(ceil($maxX)+$bufferSize);
$maxY=(ceil($maxY)+$bufferSize);

#PRINT RESULTS
print "GPS minX: $minX\n";
print "GPS maxX: $maxX\n";
print "GPS minY: $minY\n";
print "GPS maxY: $maxY\n";

open(IN, "< $gfile");
@gps = <IN>;
close(IN);

my $gs = @gps;
print "number of GPS points loaded: $gs\n";
print "_____ \n";
print "\n";

my(@res)=($minX,$maxX,$minY,$maxY);
return(@res);
}
#/\-----

#\/-----
#---LOAD INDEXED ASCII INTO HASH-----
#---DATA IN ASCII SHOULD BE IN THIS FORMAT index:x,y,z ---
sub loadIndexedASCIIToHash {
my $sub_time_start=time();

```

```

print "\n";
print "Loading Indexed ASCII file to Hash\n";
my $infile = "indexed.asc";
open(IN, "< $infile") or die "Error\: Can\'t open sorted ascii file \'$infile\'.\n";
while(<IN>){
  my $line = $_;
  chomp $line;

  if($line =~ /^EXTENTDATA/){
    my ($junk,$keep) = split(/\:/,$line);
    ($cellsize,$stx,$sty,$numXCells,$numYCells,$numTotCells) = split(/\,/,$keep);
  }else{
    my ($index,$coords)=split(/\:/,$line);
    if (exists $HoA{$index}){
      push @{$HoA{$index}}, $coords; #ADD ANOTHER ITEM TO ARRAY
    }else{
      $HoA{$index} = [ $coords ]; #CREATE AN ARRAY FOR INDEX, MAKE
      #FIRST ELEMENT
    }
  }
}
close (IN);

my $ti=(time()-$sub_time_start);
my $hr = ($ti/3600)%3600;
my $min = ($ti/60)%60;
my $sec = $ti-(3600*$hr)-(60*$min);
$hr="0".$hr if($hr<10);
$min="0".$min if($min<10);
$sec="0".$sec if($sec<10);
print "$hr\: $min\: $sec\n";

print "_____ \n";
print "\n";
}
#/\-----

#\/-----
#--OPEN AND LOAD NEW LIDAR POINTS INTO MEMORY--
#--FROM ASCII FILE--
sub loadNewFileAsc {
  my $sub_time_start=time();

  my($minX,$maxX,$minY,$maxY)=@_;

  print "Loading new ASCII LIDAR file\n";
  print "From: $lidarPath \.\\.\\.";

  #IS ASCII STRAIGHT FROM FUSION, X,Y,Z,I
  #OR IS IT ONLY X,Y,Z
  open(IN, "< $lidarPath");
  my $tmp=<IN>;
  close(IN);
  my(@tmp)=split(/\,/,$tmp);
  my $tmpsize=@tmp;

  open(IN, "< $lidarPath");
  #--FILTER UNNECESSARY LIDAR--
  if($tmpsize==4){
    while(<IN>){
      my($x,$y,$z,$i)=split(/\,/,$_);
      if(($x>=$minX) && ($x<=$maxX) && ($y>=$minY) && ($y<=$maxY)){
        my $outtmp = $x.", ".$y.", ".$z;
        push(@lidarraw,$outtmp);
      }
    }
  }
}

```

```

    }
  }
}elseif($tmpsize==3){
  while(<IN>){
    my($x,$y,$z)=split(/\/,\/,$_);
    if(($x>=$minX) && ($x<=$maxX) && ($y>=$minY) && ($y<=$maxY)){
      chomp($z);
      my $outtmp = $x.", ".$y.", ".$z;
      push(@lidarraw,$outtmp);
    }
  }
}else{
  print "ASCII LIDAR data formatted incorrectly\n";
  print "Needs to be comma delimited X,Y,Z or X,Y,Z,I\n";
  exit;
}
close(IN);

$ls = @lidarraw;
print "number of LIDAR points: $ls\n";

my $ti=(time()-$sub_time_start);
my $hr = ($ti/3600)%3600;
my $min = ($ti/60)%60;
my $sec = $ti-(3600*$hr)-(60*$min);
$hr="0".$hr if($hr<10);
$min="0".$min if($min<10);
$sec="0".$sec if($sec<10);
print "$hr\":$min\":$sec\n";

print "_____ \n";
print "\n";
}
#/\-----

#\/-----
#---OPEN AND LOAD NEW LIDAR POINTS INTO MEMORY---
#---FROM LAS FILE---
#   loadNewFileLas
#   AT BOTTOM OF THIS FILE
#/\-----

#\/-----
#---SAVE INDEXED %HOA TO INDEXED ASCII FILE---
sub saveIndexedHashtoASCII {
  my $sub_time_start=time();
  print "Saving Indexed Hash to ASCII\n";

  my(@keys) = (keys %HoA);

  #----FIND BIGGEST KEY THEN GO FROM 1 to BIGGEST
  #----DID IT THIS WAY B/C SORT ARRAY OF KEYS, NOT TRULY SORTED
  #----EX:(1,100,101,2,20,21,22,200,3,etc.)
  #----LEARN HOW TO SORT, OR DON'T WORRY PERFECT ORDER DOESN'T MATTER ANYWAY,
  #----OR USE ABOVE
  my $kmax=$keys[0];
  foreach(@keys){
    $kmax=$_ if($_>$kmax);
  }
  my $outfile = "indexed.asc";
  open(OUT, "> $outfile");

  print OUT ("EXTENTDATA\":$cellsize,$stx,$sty,$numXCells,$numYCells,$numTotCells\n");
  #---SAVE INDEXING INFO TO INDEXED DATA---

```

```

for my $index ( 1 .. $kmax ) {
    #GO THROUGH KEYS IN INCREASING ORDER, MAKE SURE KEY IS DEFINED
    if(defined $HoA{$index}){
        my(@tmp)=@{ $HoA{$index} };
        chomp(@tmp);
        for my $i (0..$#tmp){
            print OUT (" $index:$tmp[$i]\n");
        }
    }
}
close (OUT);

my $ti=(time()-$sub_time_start);
my $hr = ($ti/3600)%3600;
my $min = ($ti/60)%60;
my $sec = $ti-(3600*$hr)-(60*$min);
$hr="0".$hr if($hr<10);
$min="0".$min if($min<10);
$sec="0".$sec if($sec<10);
print "$hr\":$min\":$sec\n";

print "\n";
}
#/\-----

#/\-----
# IN FUTURE ADD saveIndexedHashToBinary
#\/-----

#\/-----
#---ASK USER WHETHER OR NOT TO SAVE %HOA TO INDEXED ASCII FILE---
sub saveYN {
    print "Do you want to save indexed data to a file (y or n): ";
    chomp(my $save = <>);

    print "\n";
    if(($save ne "n") && ($save ne "y")){
        saveYN();
    }else{
        if($save eq "n"){
            return 0;
        }else{
            return 1;
        }
    }
}
}
#/\-----

#\/-----
#---FOR A GIVEN POINT, WHAT LIDAR DO I NEED? FIND INDEX CELL FOR GPS POINT,
#---AND SURROUNDING INDEX CELLS---
sub whichCellsNeeded {

    #---FIND WHICH CELL GPS POINT IS IN---
    my($x,$y)=@_; #GPS POINT COORDS
    my $xdist = (floor(($x-$stx)/$cellsize))+1;
    my $ydist = (floor(($y-$sty)/$cellsize))+1;
    my $base=($ydist-1)*$numXCells;
    my $index=($xdist+$base);

    if($index<1 || $index>$numTotCells){
        print "GPS point is not in LIDAR coverage\n";
    }else{

```

```

#---FIND SURROUNDING CELLS---
#---$xdist = 1 means in leftest column -- can't select any more left
#---$xdist=$numXCells means in rightest column, can't select any more right
#---if $ydist=1 in bottomest row -- can't select any more down
#---if $ydist=$numYCells in topest row -- can't select any more up

my(@xcellchoices,@ycellchoices);
if($xdist==1){
  @xcellchoices=($xdist,$xdist+1);
}elseif($xdist==$numXCells){
  @xcellchoices=($xdist-1,$xdist);
}else{
  @xcellchoices=($xdist-1,$xdist,$xdist+1);
}
if($ydist==1){
  @ycellchoices=($ydist,$ydist+1);
}elseif($ydist==$numYCells){
  @ycellchoices=($ydist-1,$ydist);
}else{
  @ycellchoices=($ydist-1,$ydist,$ydist+1);
}

my(@indexCellChoices)=();
for my $ycc(0..$#ycellchoices){
  for my $xcc(0..$#xcellchoices){
    my $ccbase=($ycellchoices[$ycc]-1)*$numXCells;
    my $ccindex=($xcellchoices[$xcc]+$ccbase);
    push(@indexCellChoices,$ccindex);
  }
}

#---LOAD LIDAR DATA FOR CELLS AROUND GPS POINT---
@LIDARSelectedCells=();

for my $icc (0..$#indexCellChoices){
  #MAKE SURE KEY IS DEFINED
  if(defined $HoA{$indexCellChoices[$icc]}){
    my(@tmp)=@{ $HoA{$indexCellChoices[$icc]} };
    chomp(@tmp);
    @LIDARSelectedCells = (@LIDARSelectedCells, @tmp);
  }
}
}
#/\-----

#\-----
#---ASK USER WHETHER TO USE NEW OR SAVED/INDEXED DATA---
sub whichFileNS {
  print "\n";
  print "Load LIDAR from which type of file?\n";
  print "new non-indexed(n), saved indexed(s): ";
  chomp(my $whichload = <>);

  if(($whichload ne "n") && ($whichload ne "s")){
    print "you entered an invalid file type\n";
    whichFileNS();
  }else{
    return $whichload;
  }
  print "\n";
}
#/\-----

```

```

#\-----
sub loadNewFileLas {
    my $sub_time_start=time();

    my($Xmin,$Xmax,$Ymin,$Ymax)=@_;

    print "Loading new LAS LIDAR file\n";
    print "From: $lidarPath \. \. \. \n";

#-----NEW-----
#---MAKE SURE FILE IS BINARY -- AND DISPLAY ITS INFORMATION---
if (-B $lidarPath){

    print "File is binary\n";

#---FILE SIZE---
    my $bytesize=(-s $lidarPath);
    my $kbsize=sprintf("%.2f", ($bytesize/1024));
    my $mbsize=sprintf("%.2f", ($kbsize/1024));
    print "This file is ", $bytesize, " bytes; $kbsize KB; $mbsize MB.\n";
    print "\n";

#SYSTEM'S BYTE ORDER PREFERENCE
#0x12 0x34 0x56 0x78 = big-endian
#0x78 0x56 0x34 0x12 = little-endian
    my $endianness = join(" ", map{ sprintf "%#02x", $_ }
unpack("C*",pack("L",0x12345678)));
    if ($endianness eq "0x78 0x56 0x34 0x12"){
        print "System is Little-Endian\n";
    }elseif($endianness eq "0x12 0x34 0x56 0x78"){
        print "System is Big-Endian\n";
        print "LAS files are little-endian.\n";
        exit;
    }else{
        print "System has unknown Endian-ness\n";
        print "LAS files are little-endian.\n";
        exit;
    }
    print "\n";

    readInBinaryLAS($bytesize);

}else{
    print "File is not binary\n";
    exit;
}

sub readInBinaryLAS{
    my($bytesize)=@_;

#OPEN THE BINARY LAS FILE FOR READING IN BINMODE
    open(FILE, "< $lidarPath") or die "\nCan't open $lidarPath for reading: $!\n";
    binmode(FILE);

#READ AND DISPLAY HEADER;
my($offsetToPtData,$VerMaj,$VerMin,$ptDataFormatId,$xScaleFactor,$yScaleFactor,$zScaleFactor,$xOffset,$yOffset,$zOffset,$numPtRec,$ptDataRecLen) = readLASHeader($bytesize);

#MAKE SURE THAT THE FILE IS LAS VERSION 1.0 or 1.1 AND HAS A POINT DATA RECORD FORMAT OF 0 or 1

```

```

        if(($VerMaj != 1) || (($VerMin != 0) && ($VerMin != 1)) || (($ptDataFormatId != 0) &&
($ptDataFormatId != 1))){
            print "File claims to have an invalid LAS version\: $VerMaj\.$VerMin\n";
            print "or Point Data Record Format\: $ptDataFormatId\n";
            close(FILE);
            exit;
        }else{

            #READ AND DISPLAY POINT DATA;
readLASPtData($bytesize,$offsetToPtData,$VerMaj,$VerMin,$ptDataFormatId,$xScaleFactor,$yScaleFactor,$zScaleFactor,$xOffset,$yOffset,$zOffset,$numPtRec,$ptDataRecLen);
        }

        #CLOSE BINARY LAS FILE AND EXIT
        close(FILE);
    }

sub readLASHeader {
    my ($bytesize)=@_;

    # UNPACK USED TO CONVERT BINARY BYTES TO REQUIRED PERL DATA TYPES
    # types needed to unpack binary to:
    # C = An unsigned char value. 1 byte.
    # v = An unsigned short in little-endian order. 2 bytes.
    # V = An unsigned long in little-endian order. 4 bytes.
    # d = A double-precision float in the native format. 8 bytes.

    print " _____\n";
    print "          LAS HEADER\n";
    print "\n";

    my $fileSig;
    read(FILE, $fileSig, 4);
    print "    File signature\: $fileSig\n";
    my $fileSourceID;
    read(FILE, $fileSourceID, 2);
    $fileSourceID = unpack('v',$fileSourceID);
    print "    File Source ID\: $fileSourceID\n";
    my $reserved;
    read(FILE, $reserved, 2);
    $reserved = unpack('v',$reserved);
    print "    Reserved\: $reserved\n";

    my $projId1;
    read(FILE, $projId1, 4);
    $projId1 = unpack('V',$projId1);
    print "    Project ID 1\: $projId1\n";
    my $projId2;
    read(FILE, $projId2, 2);
    $projId2 = unpack('v',$projId2);
    print "    Project ID 2\: $projId2\n";
    my $projId3;
    read(FILE, $projId3, 2);
    $projId3 = unpack('v',$projId3);
    print "    Project ID 3\: $projId3\n";
    my $projId4;
    read(FILE, $projId4, 8);
    $projId4 = unpack('d',$projId4);
    print "    Project ID 4\: $projId4\n";

    my $VerMaj;
    read(FILE, $VerMaj, 1);
    $VerMaj = unpack('C', $VerMaj);
    print "    Version Major\: $VerMaj\n";
    my $VerMin;
    read(FILE, $VerMin, 1);
    $VerMin = unpack('C', $VerMin);

```

```

print "      Version Minor\ : $VerMin\n";

my $sysId;
read(FILE, $sysId, 32);
print "      System ID\ : $sysId\n";

my $genSoft;
read(FILE, $genSoft, 32);
print "Generating Software\ : $genSoft\n";

my $fileCreaDoY;
read(FILE, $fileCreaDoY, 2);
$fileCreaDoY = unpack('v', $fileCreaDoY);
print "      File Create DoY\ : $fileCreaDoY\n";
my $fileCreaYear;
read(FILE, $fileCreaYear, 2);
$fileCreaYear = unpack('v', $fileCreaYear);
print "      File Create Year\ : $fileCreaYear\n";

my $HeaderSize;
read(FILE, $HeaderSize, 2);
$HeaderSize = unpack('v', $HeaderSize);
print "      Header Size\ : $HeaderSize\n";
my $offsetToPtData;
read(FILE, $offsetToPtData, 4);
$offsetToPtData = unpack('V', $offsetToPtData);
print "      Offset to Pt Data\ : $offsetToPtData\n";
my $numVarLenRec;
read(FILE, $numVarLenRec, 4);
$numVarLenRec = unpack('V', $numVarLenRec);
print "      Num Var Length Rec\ : $numVarLenRec\n";
my $ptDataFormatId;
read(FILE, $ptDataFormatId, 1);
$ptDataFormatId = unpack('C', $ptDataFormatId);
print "      Pt Data Format ID\ : $ptDataFormatId\n";
my $ptDataRecLen;
read(FILE, $ptDataRecLen, 2);
$ptDataRecLen = unpack('v', $ptDataRecLen);
print "      Pt Data Rec Len\ : $ptDataRecLen\n";
my $numPtRec;
read(FILE, $numPtRec, 4);
$numPtRec = unpack('V', $numPtRec);
print "      Number Pt Recs\ : $numPtRec\n";

my $numPtsByRet1;
read(FILE, $numPtsByRet1, 4);
$numPtsByRet1 = unpack('V', $numPtsByRet1);
print "Num Pts by Return 1\ : $numPtsByRet1\n";
my $numPtsByRet2;
read(FILE, $numPtsByRet2, 4);
$numPtsByRet2 = unpack('V', $numPtsByRet2);
print "Num Pts by Return 2\ : $numPtsByRet2\n";
my $numPtsByRet3;
read(FILE, $numPtsByRet3, 4);
$numPtsByRet3 = unpack('V', $numPtsByRet3);
print "Num Pts by Return 3\ : $numPtsByRet3\n";
my $numPtsByRet4;
read(FILE, $numPtsByRet4, 4);
$numPtsByRet4 = unpack('V', $numPtsByRet4);
print "Num Pts by Return 4\ : $numPtsByRet4\n";
my $numPtsByRet5;
read(FILE, $numPtsByRet5, 4);
$numPtsByRet5 = unpack('V', $numPtsByRet5);
print "Num Pts by Return 5\ : $numPtsByRet5\n";

my $xScaleFactor;
read(FILE, $xScaleFactor, 8);
$xScaleFactor = unpack('d', $xScaleFactor);

```

```

print "      X Scale Factor\: $xScaleFactor\n";
my $yScaleFactor;
read(FILE, $yScaleFactor, 8);
$yScaleFactor = unpack('d', $yScaleFactor);
print "      Y Scale Factor\: $yScaleFactor\n";
my $zScaleFactor;
read(FILE, $zScaleFactor, 8);
$zScaleFactor = unpack('d', $zScaleFactor);
print "      Z Scale Factor\: $zScaleFactor\n";
my $xOffset;
read(FILE, $xOffset, 8);
$xOffset = unpack('d', $xOffset);
print "      X Offset\: $xOffset\n";
my $yOffset;
read(FILE, $yOffset, 8);
$yOffset = unpack('d', $yOffset);
print "      Y Offset\: $yOffset\n";
my $zOffset;
read(FILE, $zOffset, 8);
$zOffset = unpack('d', $zOffset);
print "      Z Offset\: $zOffset\n";

my $maxX;
read(FILE, $maxX, 8);
$maxX = unpack('d', $maxX);
print "      Max X\: $maxX\n";
my $minX;
read(FILE, $minX, 8);
$minX = unpack('d', $minX);
print "      Min X\: $minX\n";
my $maxY;
read(FILE, $maxY, 8);
$maxY = unpack('d', $maxY);
print "      Max Y\: $maxY\n";
my $minY;
read(FILE, $minY, 8);
$minY = unpack('d', $minY);
print "      Min Y\: $minY\n";
my $maxZ;
read(FILE, $maxZ, 8);
$maxZ = unpack('d', $maxZ);
print "      Max Z\: $maxZ\n";
my $minZ;
read(FILE, $minZ, 8);
$minZ = unpack('d', $minZ);
print "      Min Z\: $minZ\n";

#SET UPPER AND LOWER LIMITS TO $maxZ and $minZ if they are undefined
$upperlimit = $maxZ if(!$upperlimit);
$lowerlimit = $minZ if(!$lowerlimit);
print "\n";
print "      Elevation Filtering Limits\n";
print "      Upper: $upperlimit\n";
print "      Lower: $lowerlimit\n";

print "_____ \n";
print "\n";

#---GET THE NUMBER OF BYTES BETWEEN WHERE YOU NOW ARE IN THE FILE AND WHERE
#---THE POINT DATA STARTS---
#---THIS IS THE LENGTH OF THE VARIABLE LENGTH RECORDS---
my $pos = tell(FILE) unless eof(FILE);
my $posdif=$offsetToPtData-$pos;

#---MOVE THROUGH VARIABLE LENGTH RECORDS TO POINT DATA---
my $buffer;
read(FILE, $buffer, $posdif);

```

```

#---MAKE SURE YOU'RE NOW AT THE POINT DATA---
$pos = tell(FILE) unless eof(FILE);
$posdif=$offsetToPtData-$pos;

if($posdif != 0){
    print "Data Reading error in sub readHeader\(\)\:\n";
    print "You somehow managed to NOT reach the point data properly. The LAS header
says point data starts at byte\: $offsetToPtData. You're supposed to be there now, but
you're at byte\: $pos\n";
    close(FILE);
    exit;
}

my @res =
($offsetToPtData,$VerMaj,$VerMin,$ptDataFormatId,$xScaleFactor,$yScaleFactor,$zScaleFactor,
$xOffset,$yOffset,$zOffset,$numPtRec,$ptDataRecLen);
return @res;
}

sub readLASPtData{
    my
($bytesize,$offsetToPtData,$VerMaj,$VerMin,$ptDataFormatId,$xScaleFactor,$yScaleFactor,$z
ScaleFactor,$xOffset,$yOffset,$zOffset,$numPtRec,$ptDataRecLen)=@_;

    #---READ POINT DATA---
    # REQUIRED PARAMETERS
    # $bytesize,$offsetToPtData,$VerMaj,$VerMin,$ptDataFormatId <--TO READ IN
    # DATA PROPERLY
    # $xScaleFactor,$yScaleFactor,$zScaleFactor,$xOffset,$yOffset,$zOffset <--
    # TO GET PROPER PT COORDS
    # $numPtRec,$ptDataRecLen <--
    # TO MAKE SURE DATA LENGTH AND REPORTED NUMBER OF POINTS MATCH. SO READ IN
    # LOOP WORKS PROPERLY.
    #
    #1 DECIDE WHICH LAS / POINT DATA RECORD FORMATS DATA IN --> CALL CORRECT
    # PROGRAM PART
    #2 READ THROUGH...EACH LIDAR POINT IS TOTAL OF:
    #20 Bytes for POINT DATA RECORD FORMAT 0 LAS 1.0 and 1.1
    #28 Bytes for POINT DATA RECORD FORMAT 1 LAS 1.0 and 1.1
    #
    # LAS 1.0
    # POINT DATA RECORD FORMAT 0:
    # X,Y,Z,Int,RetNum,NumRets,ScnDirFlg,EoFL,Class,ScnAnglRnk,
    # File Marker,User Bit Field
    # POINT DATA RECORD FORMAT 1:
    # X,Y,Z,Int,RetNum,NumRets,ScnDirFlg,EoFL,Class,ScnAnglRnk,
    # File Marker,User Bit Field,GPS Time
    #
    # LAS 1.1
    # POINT DATA RECORD FORMAT 0:
    # X,Y,Z,Int,RetNum,NumRets,ScnDirFlg,EoFL,Class,ScnAnglRnk,
    # User Data,Point Source ID
    # POINT DATA RECORD FORMAT 1:
    # X,Y,Z,Int,RetNum,NumRets,ScnDirFlg,EoFL,Class,ScnAnglRnk,
    # User Data,Point Source ID,GPS Time

    my $iterDenom;
    if($ptDataFormatId==0){
        $iterDenom=20;
    }elsif($ptDataFormatId==1){
        $iterDenom=28;
    }else{
        print "Data Reading error in sub readPtData\(\)\:\n";
        print "An invalid Point Data Record Format\: $ptDataFormatId was passed to sub
readPtData\(\) from the LAS header. Should be 0 or 1.\n";
    }
}

```

```

        close(FILE);
        exit;
    }

    if($iterDenom != $ptDataRecLen){
        print "Data Reading error in sub readPtData\(\)\:\n";
        print "LAS file claims a data record length\:\ $ptDataRecLen and a Point Data Record
Format\:\ $ptDataFormatId. These do not match.\n";
        close(FILE);
        exit;
    }

    my $iter=$(( $bytesize-$offsetToPtData) / $iterDenom);
    if($iter != $numPtRec){
        print "Data Reading error in sub readPtData\(\)\:\n";
        print "LAS file claims to have\:\ $numPtRec LIDAR points. Math on file size says\:\
$iter LIDAR points.\n";
        close(FILE);
        exit;
    }else{
        print "There are $iter LIDAR points. Begin reading in points.\n";
    }

    # $iter=2; #<--REMOVE TO RUN FOR REAL---
    my $progchange=0;
    while ($iter>0){

        my
($ptx,$pty,$ptz,$pti,$ptRnNrSdfEofl,$ptClass,$ptSar,$ptFm,$ptUd,$ptUbf,$ptPsid,$ptGpsTime
);

        #---X,Y,Z COORDS NEED TO BE MULTIPLIED BY X,Y,Z SCALEFACTORS FROM HEADER
        #---AND HAVE X,Y,Z OFFSETS ADDED--
        read(FILE, $ptx, 4);
        $ptx = ((unpack('V', $ptx))*$xScaleFactor)+$xOffset;
        #print "Point X\:\ $ptx\n";
        read(FILE, $pty, 4);
        $pty = ((unpack('V', $pty))*$yScaleFactor)+$yOffset;
        #print "Point Y\:\ $pty\n";
        read(FILE, $ptz, 4);
        $ptz = ((unpack('V', $ptz))*$zScaleFactor)+$zOffset;
        #print "Point Z\:\ $ptz\n";
        read(FILE, $pti, 2);
        # $pti = unpack('v', $pti);
        #print "Point I\:\ $pti\n";

        #---FOUR FIELDS IN 1 BYTE-----
        # 1.Return Number (1st three bits)
        #   The number of the return in its pulse.
        # 2.Number of Returns (2nd three bits)
        #   Total number of returns for a given pulse. Max is 5.
        # 3.Scan Direction Flag (7th bit)
        #   0=negative,1=positive
        # 4.Edge of Flight Line (8th bit)
        #   0=not at edge of flight line,1=at flight line edge
        #
        # LAS format is Little Endian

        #---READ 1 BYTE FROM THE LAS FILE AND UNPACK IT INTO
        #   A bit string (ascending bit order inside each byte)
        read(FILE, $ptRnNrSdfEofl, 1);

        # $ptRnNrSdfEofl=unpack('b32', $ptRnNrSdfEofl);
        #---USE BELOW IF YOU WANT THESE THINGS---
        #---THE 1 BYTE IS NOW AN 8 NUMBER STRING OF 0s AND 1s.
        #---USE SUBSTRING TO GET VALUES---

```

```

        #my $ptRn =
(substr($ptRnNrSdfEofl,0,1)*1)+(substr($ptRnNrSdfEofl,1,1)*2)+(substr($ptRnNrSdfEofl,2,1)
*4);
        ##print "Return Number\ : $ptRn\n";
        #my $ptNr =
(substr($ptRnNrSdfEofl,3,1)*1)+(substr($ptRnNrSdfEofl,4,1)*2)+(substr($ptRnNrSdfEofl,5,1)
*4);
        ##print "Number of Returns\ : $ptNr\n";
        #my $ptSdf = substr($ptRnNrSdfEofl,6,1);
        ##print "Scan Direction Flag\ : $ptSdf\n";
        #my $ptEofl = substr($ptRnNrSdfEofl,7,1);
        ##print "Edge of Flight Line\ : $ptEofl\n";

read(FILE, $ptClass, 1);
#$ptClass = unpack('C', $ptClass);
#print "Classification\ : $ptClass\n";
read(FILE, $ptSar, 1);
#$ptSar = unpack('c', $ptSar);
#print "Scan Angle Rank\ : $ptSar\n";

if($VerMin==0){
    read(FILE, $ptFm, 1);
    # $ptFm = unpack('C', $ptFm);
    #print "File Marker\ : $ptFm\n";
    read(FILE, $ptUbf, 2);
    # $ptUbf = unpack('v', $ptUbf);
    #print "User Bit Field\ : $ptUbf\n";
}elsif($VerMin==1){
    read(FILE, $ptUd, 1);
    # $ptUd = unpack('C', $ptUd);
    #print "User Data\ : $ptUd\n";
    read(FILE, $ptPsid, 2);
    # $ptPsid = unpack('v', $ptPsid);
    #print "Point Source ID\ : $ptPsid\n";
}else{
    print "You somehow got this far with an invalid LAS Minor Version\ : $VerMin. You
shouldn't have done that.\n";
    close(FILE);
    exit;
}
#---LAS POINT DATA FORMAT 1 (for both LAS 1.0 and 1.1)
#---HAS ANOTHER 8 BYTE GPS TIME FIELD---
if($ptDataFormatId==1){
    read(FILE, $ptGpsTime, 8);
    # $ptGpsTime = unpack('d', $ptGpsTime);
    #print "GPS Time\ : $ptGpsTime\n";
}

if(($ptx>=$Xmin) && ($ptx<=$Xmax) && ($pty>=$Ymin) && ($pty<=$Ymax)){
    if(($ptz>=$lowerlimit) && ($ptz<=$upperlimit)){
        my $outtmp = $ptx.", ".$pty.", ".$ptz;
        push(@lidarraw,$outtmp);
    }
}

#PERCENT COMPLETED
my $scurPtNum = $numPtRec-$iter;
my $prog= (100*($scurPtNum/$numPtRec));
$prog = sprintf("%.0f", (floor($prog*100))/100 );
if(($prog%100==0)&& ($prog != $progchange)){
    print "$progchange\% completed\r";
    $progchange=$prog;
}

$iter--;
}
}

```

```
$ls = @lidarraw;
print "100\% completed\n";
print "number of LIDAR points in GPS extent: $ls\n";

my $ti=(time()-$sub_time_start);
my $hr = ($ti/3600)%3600;
my $min = ($ti/60)%60;
my $sec = $ti-(3600*$hr)-(60*$min);
$hr="0".$hr if($hr<10);
$min="0".$min if($min<10);
$sec="0".$sec if($sec<10);
print "$hr\":$min\":$sec\n";

print "_____ \n";
print "\n";
}
#/\-----
```

Appendix B: Triangle Method Code 2

```

sub mainHtDif { #----START OF SUB MAINHTDIF---
#-----NOTES-----#
#THIS SUBROUTINE:
# CREATES three bins for the LIDAR data, by Azimuth from the GPS.
# Initial bins are 0,120,240 degrees.
# Sort the LIDAR points by elevation.
# Take the lowest point in each bin and create a surface.
# Test the surface to make sure it contains the GPS point.
# If it doesn't, select the next lowest point in any bin.
# Keep testing and creating surfaces until one contains the GPS point.
# Find the elevation of this surface at the GPS point, and find the
# difference from the GPS elevation.
# ROTATES THE BINS 20degrees (to 20,140,260 degrees) and repeat the entire
# process.
# ROTATE 20degrees, four more times (for a total of six rotations) repeating
# entire process.
# KEEP THE LOWEST PLANE ELEVATION FROM ALL SIX ROTATIONS, THEN USE THAT AS
# THE FINAL HEIGHT.
#-----#

#-----#
#---PASS IN DATA FROM INDEXING---
$intArraySize=@_;
$sliceEnd=$intArraySize-1;

my $radius = @_[0];

my($point,$s,$y,$x,$z_ell,$z_orth,$vght,$vgdns,$numep,$avgst,$avgpdp,$type,$rms)=@_[1..13
];
chomp($rms);
my(@lidar)=@_[14..$sliceEnd];
#-----#

#-----#
#---SETUP GPS DATA---
@mhd_gps=();
@mhd_t = ($x,$y,$z_ell,$point,$s,$vght,$vgdns,$numep,$avgst,$avgpdp,$type,$rms);
push @mhd_gps, [@mhd_t];
#-----#

#-----#
# There are HEIGHT DIFFERENCES BETWEEN THE CONTROL POINTS AND
# THE LIDAR at the sites. It is different for each site.
#
# The values here are the AVERAGE difference FOR ALL CONTROL
# POINTS AT A SITE, EXCLUDING OUTLIERS.
#
# THE GPS ELEVATIONS ARE THIS HEIGHT ABOVE THE LIDAR.
# IN THIS ORDER: Old1,Old2,Gp1,Piru2,Piru1.
#
# THIS VALUE NEEDS TO BE SUBTRACTED FROM THE GPS HT IN ORDER
# TO PROPERLY LINE UP WITH THE LIDAR and remove the systematic
# error.
@mhd_cpt_hdif=(0.238754,0.245841,0.184279,0.283400,0.206734);
#---FROM TRIANGLE METHOD 2M RADIUS---
#-----#

```

```

#----THE MAIN PROGRAM---
$first=0;
$rotation=20;          #ROTATE THE BINS THIS MANY DEGREES EACH TIME
$num_rot=(360/($rotation*3))-1;

#-----FOREACH GPS POINT START-----
foreach $i(0 .. $#mhd_gps){

    @resultos=();

    #---SET UP GPS DATA START-----
    $tgps=$mhd_gps[$i][4];          #Site Name
    $gpsdif=$mhd_cpt_hdif[0] if($tgps eq "Old1");
    $gpsdif=$mhd_cpt_hdif[1] if($tgps eq "Old2");
    $gpsdif=$mhd_cpt_hdif[2] if($tgps eq "GP1");
    $gpsdif=$mhd_cpt_hdif[3] if($tgps eq "Piru2");
    $gpsdif=$mhd_cpt_hdif[4] if($tgps eq "Piru1");

    $gpsx=$mhd_gps[$i][0];          #GPS X Coordinate
    $gpsy=$mhd_gps[$i][1];          #GPS Y Coordinate
    $gpsz=( $mhd_gps[$i][2] )-$gpsdif; #GPS Z Coordinate - Avg. Control Point HD
    #---SET UP GPS DATA END-----

    #---GET THE POINTS WITHIN DISTANCE OF GPS START---
    #---SAVE TO @lines---
    @lines=();
    for $l1(0 .. $#lidar){
        chomp($lidar[$l1]);
        ($lx,$ly,$lz) = split(/\./,$lidar[$l1]);

        #---deltaX / deltaY---
        $dX=($lx-$gpsx);
        $dY=($ly-$gpsy);
        $dist = sqrt($dX**2+$dY**2); #---Use horizontal distance. Steep slopes,
                                     #may shorten up/down slope distances in
                                     #unacceptable and unpredictable ways.

        if($dist<=$radius){
            $ldr=$lx.", ".$ly.", ".$lz.", ".$dX.", ".$dY.", ".$dist;
            push(@lines,$ldr);
        }
    }
    $numLidarInRadius=@lines;
    #---GET THE POINTS WITHIN DISTANCE OF GPS END---

    #---SETUP OUTPUT VALUES FOR EACH GPS POINT---
    @finalp1=(0,0,0);
    @finalp2=(0,0,0);
    @finalp3=(0,0,0);
    $lowest_plane=100000;
    $final_htdif=100000;
    $final_rot=100000;

    #-----REPEAT $num_rot TIMES START-----
    for $q (0..$num_rot){

```

```

#---SET UP BINS START-----
@bin1=();
@bin2=();
@bin3=();

if(($first==360) || ($first==0) || ($first==120) || ($first==240)){
    $first=0;
}
$binline1=$first+($q*$rotation);
$binline2=$binline1+120;
$binline2-=360 if($binline2>360);
$binline3=$binline2+120;
$binline3-=360 if($binline3>360);
#---SET UP BINS END-----

#----PUT POINTS IN BINS START-----
for $j (0 .. $#lines){
    ($lx,$ly,$lz,$dX,$dY,$dist) = split(/\./,$lines[$j]);
    #-----
    #   IN RARE CASES, THE LIDAR POINT HAS EXACTLY THE SAME Y COORDINATE
    #   AS THE GPS---
    #   THIS CAUSES THE AZIMUTH CALCULATION BELOW TO DIVIDE BY ZERO.
    #   IF THE Y COORDS ARE THE SAME, THE AZ IS EITHER 90 or 270, or THE
    #   LIDAR POINT HAS EXACTLY THE SAME COORDS AS THE GPS
    #-----
    if($dY==0){
        if($dX>0){
            $az=90; #LIDAR IS STRAIGHT EAST OF GPS
        }elseif($dX<0){
            $az=270; #LIDAR IS STRAIGHT WEST OF GPS
        }else{
            $az=0; #LIDAR POINT HAS EXACTLY THE SAME COORDS AS GPS, bin
                #doesn't matter. This did not happen with the 1709 GPS
                #points from SOCAL.
            print "$gps[$i][3] has a lidar point with exactly the same coordinates.
$lx,$ly,$lz\n";
            print "may want to just use LIDAR z, rather than the plane height.\n";
        }
    }else{
        #---AZIMUTH---
        $az=rad2deg(atan($dX/$dY));
        if($dY<0){
            $az+=180;
        }
        if(($dY>=0) && ($dX<0)){
            $az+=360;
        }
    }
    @t=($lx,$ly,$lz,$dist);

    #----BIN1-----
    if($binline1==0){
        #special case where binline is on 0/360.
        #Azimuths of 0 OR 360 should be included
        if($az==360){
            push @bin1, [@t];
        }
    }
    if($binline1>=240){
        #---IF the upper limit of this bin is over 360/0, then && logic won't
        #work, need to use || ---
        if(($az>=$binline1) || ($az<$binline2)){ #---So need to use OR---
            push @bin1, [@t];
        }
    }
    }else{
        if(($az>=$binline1) && ($az<$binline2)){ #--- can use AND---

```



```

@bin3 = hashSort(@bin3);

#-----SELECT LOWEST POINTS START-----
#---Starting at the beginning of each bin, select 1st points.
$bin1_index=0;
$bin2_index=0;
$bin3_index=0;
@p1=($bin1[$bin1_index][0],$bin1[$bin1_index][1],$bin1[$bin1_index][2]);
@p2=($bin2[$bin2_index][0],$bin2[$bin2_index][1],$bin2[$bin2_index][2]);
@p3=($bin3[$bin3_index][0],$bin3[$bin3_index][1],$bin3[$bin3_index][2]);

#---call inOut() to see if GPS is in this triangle.
#---if it's not ($goodbad==1), run htDif() to find next lowest point.
#---repeat until GPS is in triangle ($goodbad==0), then go on.
$goodbad = inOut();

$numloops=0;
while($goodbad){
  if( ($bin1_index >= $stop1) && ($bin2_index >= $stop2) && ($bin3_index >= $stop3)
){
  #---REACHED THE END OF ALL INDEXES---NO TRIANGLE CONTAINS GPS
  #---USE ORIGINAL 3 LOW POINTS---
  print "$mhd_gps[$i][3] -- no triangle contains GPS\n";
  @p1=($bin1[0][0],$bin1[0][1],$bin1[0][2]);
  @p2=($bin2[0][0],$bin2[0][1],$bin2[0][2]);
  @p3=($bin3[0][0],$bin3[0][1],$bin3[0][2]);
  $goodbad=0;
  }else{
  #print "$goodbad -- $bin1_index -- $bin2_index -- $bin3_index\n";
  hiteDif();
  $goodbad = inOut();
  $numloops++;
  }
}
#-----SELECT LOWEST POINTS END-----

#-----
#
#   USE THREE POINTS TO FIND PLANE ELEVATION UNDER GPS POINT
#
#---equation of plane -- using cross product---
#---EQUATION OF A PLANE---
#   Ax + By + Cz + D = 0
#   $t1*x + $t2*y + $t3*z + $t4 = 0
#
#---2 vectors from 3 points---
#---vector1 (@v1) p2 -> p1---
$vx = $p2[0]-$p1[0];
$vy = $p2[1]-$p1[1];
$vz = $p2[2]-$p1[2];
@v1 = ($vx,$vy,$vz);
#---vector2 (@v2) p3 -> p1---
$vx = $p3[0]-$p1[0];
$vy = $p3[1]-$p1[1];
$vz = $p3[2]-$p1[2];
@v2 = ($vx,$vy,$vz);

#---orthogonal vector---
$t1 = (($v1[1]*$v2[2])-( $v1[2]*$v2[1]));
$t2 = (($v1[2]*$v2[0])-( $v1[0]*$v2[2]));
$t3 = (($v1[0]*$v2[1])-( $v1[1]*$v2[0]));
#---D term---
$t4 = ($t1*$p1[0])+($t2*$p1[1])+($t3*$p1[2]);
#-----

```

```

#---height of plane at gps point start-----
#LIDAR ground elevation at GPS point.
$ngpsz = ($t4 - (($t1*$gpsx)+($t2*$gpsy)))/$t3;
$gps_htdif = $ngpsz-$gpsz;
#LIDAR ground elevation - GPS ground elevation. Positive difference is
#LIDAR above GPS, negative LIDAR below GPS.
#---height of plane at gps point end-----

#---KEEP DATA FOR LOWEST OF THE ROTATIONS.---
if($ngpsz<$lowest_plane){
  $lowest_plane=$ngpsz;
  $final_htdif=$gps_htdif;
  $final_rot=$binline1."-".$binline2."-".$binline3;
  @finalp1=($p1[0],$p1[1],$p1[2]);
  @finalp2=($p2[0],$p2[1],$p2[2]);
  @finalp3=($p3[0],$p3[1],$p3[2]);
}
}
#-----REPEAT $num_rot TIMES END-----

@triout =
($mhd_gps[$i][4],$mhd_gps[$i][3],$mhd_gps[$i][0],$mhd_gps[$i][1],$mhd_gps[$i][2],$lowest_
plane,$final_htdif,$final_rot,$mhd_gps[$i][5],$mhd_gps[$i][6],$mhd_gps[$i][7],$mhd_gps[$i
][8],$mhd_gps[$i][9],$mhd_gps[$i][10],$mhd_gps[$i][11],$numLidarInRadius);

  return(@triout);
}
#-----FOREACH GPS POINT END-----

#-----SUBROUTINES-----
#\-----
sub hiteDif{
#-----NOTES-----#
# This finds the next lowest point.
# Pick the next point in all three bins and return the lowest.
# IF YOU GET TO THE END OF ONE BIN, AND STILL HAVE POINTS IN THE OTHER
# BINS YOU WANT TO STOP USING THE BIN THAT'S ENDED, AND ONLY USE THE OTHER
# TWO.
# So, there needs to be an upper limit.
# IF you get to the last point in all three bins and you still don't have
# a triangle with the GPS point - EXIT
# THIS SHOULD HAVE BEEN DONE ABOVE. THIS SUBROUTINE SHOULD NOT HAVE BEEN
# CALLED IF THIS IS THE CASE.
#
# OTHERWISE, IF you get to the last point in one bin, and haven't found a
# triangle, BUT there are more points in the other bins still, start
# incrementing ONLY the other bins, and stop using the limited bin.
#
# This can be done by making the elevation in the limited bin ridiculously
# high.
# That way the smallest elevation increase will always happen in the bins
# with more points.
#-----#

if(($binl_index+1)>$stop1){
  #---IF AT THE END OF BIN 1, SET HEIGHT TO 10000+ELEVATION OF LAST POINT
  $hdf1=$binl[$#binl][2]+10000;
}else{
  #---OTHERWISE SET HEIGHT TO ELEVATION OF NEXT POINT
  $hdf1=$binl[$binl_index+1][2];
}
}

```

```

if(($bin2_index+1)>$stop2){
  #---IF AT THE END OF BIN 2, SET HEIGHT TO 10000+ELEVATION OF LAST POINT
  $hdf2=$bin2[$#bin2][2]+10000;
}else{
  #---OTHERWISE SET HEIGHT TO ELEVATION OF NEXT POINT
  $hdf2=$bin2[$bin2_index+1][2];
}
if(($bin3_index+1)>$stop3){
  #---IF AT THE END OF BIN 3, SET HEIGHT TO 10000+ELEVATION OF LAST POINT
  $hdf3=$bin3[$#bin3][2]+10000;
}else{
  #---OTHERWISE SET HEIGHT TO ELEVATION OF NEXT POINT
  $hdf3=$bin3[$bin3_index+1][2];
}

if($hdf1<$hdf2){
  if($hdf1<$hdf3){
    #---HDF1 LOWEST---
    $bin1_index+=1;
    @p1=($bin1[$bin1_index][0],$bin1[$bin1_index][1],$bin1[$bin1_index][2]);
#---CHANGE BIN 1 POINT TO NEXT POINT---
  }else{
    #---HDF3 LOWEST---
    $bin3_index+=1;
    @p3=($bin3[$bin3_index][0],$bin3[$bin3_index][1],$bin3[$bin3_index][2]);
#---CHANGE BIN 3 POINT TO NEXT POINT---
  }
}else{
  if($hdf2<$hdf3){
    #---HDF2 LOWEST---
    $bin2_index+=1;
    @p2=($bin2[$bin2_index][0],$bin2[$bin2_index][1],$bin2[$bin2_index][2]);
#---CHANGE BIN 2 POINT TO NEXT POINT---
  }else{
    #---HDF3 LOWEST---
    $bin3_index+=1;
    @p3=($bin3[$bin3_index][0],$bin3[$bin3_index][1],$bin3[$bin3_index][2]);
#---CHANGE BIN 3 POINT TO NEXT POINT---
  }
}
}
#/\-----

#\-----
# SORTING
#
sub hashSort{
  my(@bin)=@_;
  my(@out);
  my %ElevHash = ();

  #---ADD EVERY RETURN TO HASH WITH ELEVATION AS KEY---
  #---THIS WOULD BE IMPROVED IF YOU COULD MAKE A HASH OF 2D ARRAYS
  #---EACH RETURN IS AN ARRAY, EACH ELEVATION IN HASH IS AN ARRAY OF
  #---ARRAYS INDEXED BY ELEVATION
  #---COULD GET RID OF ALL THE ARRAY-->STRING-->ARRAY CONVERSIONS
  for my $i(0 .. $#bin){
    my $index=$bin[$i][2];

    my $tp = $bin[$i][0].",".$bin[$i][1].",".$bin[$i][2].",".$bin[$i][3];

    if(exists $ElevHash{$index}){
      push @{$ElevHash{$index}} , $tp; #ADD ANOTHER ITEM TO ARRAY
    }else{
      $ElevHash{$index} = [ $tp ]; #CREATE AN ARRAY FOR INDEX, MAKE
      #FIRST ELEMENT
    }
  }
}

```

```

#---IF MORE THAN 1 RETURN AT EACH ELEVATION---QUICKSORT BY DISTANCE---
for my $key ( keys %ElevHash ) {
    #MAKE SURE KEY IS DEFINED
    if(defined $ElevHash{$key}){
        my(@temp) = @ { $ElevHash{$key} };

        if($#temp){
            #CONVERT ARRAYofSTRINGS TO ARRAYofARRAYS
            my(@sort);
            foreach(@temp){
                my(@t)=split(/\./,$_);
                push(@sort,@t);
            }

            #SORT ARRAYofARRAYS
            my $lb = 0;
            my $hb = $#sort;
            @sort = quicksort(@sort,$lb,$hb,3);
            @sort = reverseArray(@sort);

            #CONVERT ARRAYofARRAYS TO ARRAYofSTRINGS
            my(@aoaTaos);
            for my $o (0..$#sort){
                my $f = $sort[$o][0].",".$sort[$o][1].",".$sort[$o][2].",".$sort[$o][3];
                push(@aoaTaos,$f);
            }
            $ElevHash{$key} = [@aoaTaos];
        }
    }
}

#---WRITE OUT HASH IN ORDER TO NEW ARRAY---
my(@keys) = (sort keys %ElevHash);
for my $k (0..$#keys) {
    if(defined $ElevHash{$keys[$k]}){
        my(@tmp)=@ { $ElevHash{$keys[$k]} };
        for my $i (0..$#tmp){
            my(@t)=split(/\./,$tmp[$i]);
            push (@out,@t);
        }
    }
}

return @out;
}

sub quicksort {
    my(@arraytosort)=@_;

    my $sortindex=$arraytosort[$#arraytosort];
    my $hi=$arraytosort[$#arraytosort-1];
    my $lo=$arraytosort[$#arraytosort-2];
    pop(@arraytosort);
    pop(@arraytosort);
    pop(@arraytosort);

    #lo is the lower index, hi is the upper index
    #of the region of array a that is to be sorted
    my $i=$lo;
    my $j=$hi;
    my (@t,@u);
    my $x=$arraytosort[(($lo+$hi)/2)[$sortindex];

    #partition
    do {
        $i++ while($arraytosort[$i][$sortindex]<$x);
        $j-- while($arraytosort[$j][$sortindex]>$x);
    }
}

```

```

        if ($i<=$j) {
@t=($arraytosort[$i][0],$arraytosort[$i][1],$arraytosort[$i][2],$arraytosort[$i][3]);
@u=($arraytosort[$j][0],$arraytosort[$j][1],$arraytosort[$j][2],$arraytosort[$j][3]);
    $arraytosort[$i]=@u;
    $arraytosort[$j]=@t;
    $i++;
    $j--;
    }
} while ($i<=$j);

#recursion
@arraytosort = quicksort(@arraytosort,$lo,$j,$sortindex) if($lo<$j);
@arraytosort = quicksort(@arraytosort,$i,$hi,$sortindex) if($i<$hi);

return @arraytosort;
}

sub reverseArray{
    my(@in)=@_;

    my(@tmp);
    for(my $i=#in; $i>=0; $i--){
        $tmp[$#in-$i] = $in[$i];
    }
    return @tmp;
}
#/\-----

#\/-----
sub inOut{
    #---IS GPS INSIDE OR OUTSIDE OF TRIANGLE MADE BY THESE THREE LIDAR POINTS?
    #-----NOTES-----#
    #---POINT IN TRIANGLE FORMULAS-----#
    #fAB = (y-y1)*(x2-x1) - (x-x1)*(y2-y1) #
    #fCA = (y-y3)*(x1-x3) - (x-x3)*(y1-y3) #
    #fBC = (y-y2)*(x3-x2) - (x-x2)*(y3-y2) #
    # #
    #---VARIABLES-----#
    # x = $gps[$i][0] #
    # y = $gps[$i][1] #
    #x1 = $p1[0] #
    #y1 = $p1[1] #
    #x2 = $p2[0] #
    #y2 = $p2[1] #
    #x3 = $p3[0] #
    #y3 = $p3[1] #
    #-----#

    my(@iop1)=($p1[0],$p1[1]); #LIDAR POINT 1 X,Y
    my(@iop2)=($p2[0],$p2[1]); #LIDAR POINT 2 X,Y
    my(@iop3)=($p3[0],$p3[1]); #LIDAR POINT 3 X,Y
    my $iogpsx=$mhd_gps[$i][0]; #GPS X
    my $iogpsy=$mhd_gps[$i][1]; #GPS Y

    my $inout;
    my $fAB = ($iogpsy-$iop1[1])*($iop2[0]-$iop1[0]) - ($iogpsx-$iop1[0])*($iop2[1]-
    $iop1[1]);
    my $fCA = ($iogpsy-$iop3[1])*($iop1[0]-$iop3[0]) - ($iogpsx-$iop3[0])*($iop1[1]-
    $iop3[1]);
    my $fBC = ($iogpsy-$iop2[1])*($iop3[0]-$iop2[0]) - ($iogpsx-$iop2[0])*($iop3[1]-
    $iop2[1]);
    if(((($fAB*$fBC)>0) && (($fBC*$fCA)>0))){
        $inout = 0; #GPS point is inside the triangle
    }else{
        $inout = 1; #GPS point is OUTside the triangle
    }
}

```

```

return $inout;          #return $inout
}
#/\-----#

#\/-----#
sub removeDuplicatess{
#-----NOTES-----#
# SLOW SUBROUTINE, TO REMOVE LIDAR RETURNS, IF THERE'S MORE THAN ONE IN
# EACH BIN WITH EXACT SAME DATA.
# LOOPS THROUGH EACH POINT OF EACH BIN AND COMPARES IT TO A GROWING ARRAY
# OF NON-DUPLICATE POINTS.
# IF A POINT IS NOT IN THE NON-DUPLICATE ARRAY, ADD IT. IF IT'S IN THE
# ARRAY, DON'T ADD IT
#-----#

#---BIN1-----#
#---COMPARE EACH TO NEW ARRAY---ADD IF NOT IN, DON'T IF IN---
my(@tmp);
my($k,$l,$alreadyIn);

foreach $k (0 .. $#bin1) {
    $alreadyIn=1;
    foreach $l (0..$#tmp){
        if($bin1[$k][0]==$tmp[$l][0]){
            if(($bin1[$k][1]==$tmp[$l][1]) && ($bin1[$k][2]==$tmp[$l][2]) &&
($bin1[$k][3]==$tmp[$l][3]) && ($bin1[$k][4]==$tmp[$l][4])){
                $alreadyIn=0;
                last;
            }
        }
    }
    if($alreadyIn){
        @t=($bin1[$k][0], $bin1[$k][1], $bin1[$k][2], $bin1[$k][3], $bin1[$k][4]);
        push @tmp, [@t];
    }
}
@bin1 = @tmp;

#---BIN2-----#
#---COMPARE EACH TO NEW ARRAY---ADD IF NOT IN, DON'T IF IN---
@tmp=();
foreach $k (0 .. $#bin2) {
    $alreadyIn=1;
    foreach $l (0..$#tmp){
        if($bin2[$k][0]==$tmp[$l][0]){
            if(($bin2[$k][1]==$tmp[$l][1]) && ($bin2[$k][2]==$tmp[$l][2]) &&
($bin2[$k][3]==$tmp[$l][3]) && ($bin2[$k][4]==$tmp[$l][4])){
                $alreadyIn=0;
                last;
            }
        }
    }
    if($alreadyIn){
        @t=($bin2[$k][0], $bin2[$k][1], $bin2[$k][2], $bin2[$k][3], $bin2[$k][4]);
        push @tmp, [@t];
    }
}
@bin2 = @tmp;

#---BIN3-----#
#---COMPARE EACH TO NEW ARRAY---ADD IF NOT IN, DON'T IF IN---
@tmp=();
foreach $k (0 .. $#bin3) {
    $alreadyIn=1;
    foreach $l (0..$#tmp){
        if($bin3[$k][0]==$tmp[$l][0]){

```

```

        if(($bin3[$k][1]==$tmp[$l][1]) && ($bin3[$k][2]==$tmp[$l][2]) &&
($bin3[$k][3]==$tmp[$l][3]) && ($bin3[$k][4]==$tmp[$l][4])){
            $alreadyIn=0;
            last;
        }
    }
    if($alreadyIn){
        @t=($bin3[$k][0], $bin3[$k][1], $bin3[$k][2], $bin3[$k][3], $bin3[$k][4]);
        push @tmp, [@t];
    }
}
@bin3 = @tmp;
}
#/\-----
} #----END OF SUB MAINHTDIF---

1; #---NEEDS TO BE HERE FOR PERL TO WORK PROPERLY---

```