NAVAL
POSTGRADUATE
SCHOOL

MONTEREY, CALIFORNIA

THESIS

SIMULATING FULL-WAVEFORM LIDAR

by

Angela M. Kim

September 2009

Thesis Co-Advisors:	Carlos F. Borges Richard C. Olsen

Approved for public release; distribution is unlimited

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2009	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE Simulating Full-waveform LIDAR			**5. FUNDING NUMBERS**
6. AUTHOR(S) Angela M. Kim			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

LIDAR (LIght Detection And Ranging) is used to remotely measure the three-dimensional shapes and arrangements of objects with high efficiency and accuracy by making precise measurements of time-of-flight of pulses of light. Discrete return LIDAR systems provide a discrete series of elevation points corresponding to reflections from objects in the scene. Full-waveform LIDAR systems measure the intensity of light returned to the sensor continuously over a period of time. Relatively little research has been done on full-waveform LIDAR signals. This thesis presents a Monte Carlo model of laser propagation through a tree which allows simulation of full-waveform LIDAR signatures. The model incorporates a LIDAR system and a "natural" scene, including an atmosphere, tree and ground surface. Test cases are presented which enlighten various aspects of the model, and give insight into full-waveform LIDAR data collection and analysis. Changes in the scene such as varying ground reflectance, sloped versus flat ground, and comparisons of "leaf-on" and "leaf-off" conditions are analyzed. Changes in the LIDAR system are also studied, such as changing laser wavelength, shape and length of transmitted pulses, sensing geometry, etc. Results of the simulations and analysis of the effects of physical changes in the scene and sensor are presented.

14. SUBJECT TERMS LIDAR, Monte Carlo simulation, full-waveform, model			**15. NUMBER OF PAGES** 127
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UU

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

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

SIMULATING FULL-WAVEFORM LIDAR

Angela M. Kim
Civilian, United States Navy
B.S., Norwich University, 2000

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED MATHEMATICS

from the

NAVAL POSTGRADUATE SCHOOL
September 2009

Author:	Angela M. Kim
Approved by:	Carlos F. Borges Thesis Co-Advisor
	Richard C. Olsen Thesis Co-Advisor
	Carlos F. Borges Chairman, Department of Applied Mathematics

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

LIDAR (LIght Detection And Ranging) is used to remotely measure the three-dimensional shapes and arrangements of objects with high efficiency and accuracy by making precise measurements of time-of-flight of pulses of light. Discrete return LIDAR systems provide a discrete series of elevation points corresponding to reflections from objects in the scene. Full-waveform LIDAR systems measure the intensity of light returned to the sensor continuously over a period of time. Relatively little research has been done on full-waveform LIDAR signals. This thesis presents a Monte Carlo model of laser propagation through a tree which allows simulation of full-waveform LIDAR signatures. The model incorporates a LIDAR system and a “natural” scene, including an atmosphere, tree and ground surface. Test cases are presented which enlighten various aspects of the model, and give insight into full-waveform LIDAR data collection and analysis. Changes in the scene such as varying ground reflectance, sloped versus flat ground, and comparisons of “leaf-on” and “leaf-off” conditions are analyzed. Changes in the LIDAR system are also studied, such as changing laser wavelength, shape and length of transmitted pulses, sensing geometry, etc. Results of the simulations and analysis of the effects of physical changes in the scene and sensor are presented.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I am thankful for the many people who have enriched my life, and taught me a joy of learning.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A LIDAR sensor measures the time-of-flight for light to travel from the sensor, to an object, and back. This time-of-flight information directly correlates to distance. When a laser interacts with a complex object, such as a tree, some of the laser energy will be reflected from the very top of the object, some of the light will be reflected from the branches within the canopy, and some of the light will reach the ground and be reflected back through the canopy to the sensor. A LIDAR waveform is a continuous measurement of the amount of laser energy returned over time, and provides information about the 3-D structure of objects.

The model presented in this thesis allows simulation of a LIDAR waveform. The purpose of developing this simulation capability is to provide a way of studying the factors that influence the shape of a LIDAR waveform, and to assist in the development of LIDAR waveform processing algorithms. The model consists of a simulated LIDAR system and a "natural" scene, including an atmosphere, a tree with leaves, and a ground surface.

The simulated LIDAR system has variable transmission and receiver parameters, including an initial pulse energy, shape and length, minimum detection energy, angular beam spread, sensor aperture size, sensor flying height and location, timing accuracy, and laser wavelength. The atmosphere is dependent upon altitude, and is modeled according to the Beer-Lambert law (Measures, 1992). A simulated tree is included in the scene, and consists of two components: the physical shape and arrangement of branches, twigs and leaves, and the reflectance properties of these materials. The shape of the tree is modeled based on a method called Lindenmayer-systems, or L-systems for short, which were developed by Aristid Lindenmayer in 1968 to study the geometric growth patterns of plants (Prusinkiewicz, 2004).

Within the tree, a bud location is recorded for each leaf, and a defined shape around the bud is defined as containing leaves. The bark and leaf models are combined as a probability map, with each location assigned a likelihood of being bark, leaf, or air.

Leaf reflectance, transmission, and absorption values are determined according to the PROSPECT leaf reflectance model of Jacquemoud and Baret (Jacquemoud, 1990). Tree bark and the ground surface radiometric values are assigned based on spectra taken from the publicly-available U.S. Geological Survey (USGS) vegetation library.

The propagation of LIDAR is modeled using a Monte Carlo approach. The LIDAR pulse is divided into multiple small bins, simulating the photons that make up the laser pulse. The energy within each bin is propagated through the model, with material interactions being determined probabilistically, and according to the radiometric properties of the material. The distribution of energy is recorded, whether the photons are returned to the sensor, absorbed by materials in the scene, or reflected off into space. The time taken for energy to return to the sensor is recorded, and a simulated waveform is created. This process is repeated multiple times. The amount of energy returned to the sensor for each model iteration is tracked, and averaged after completion of all runs.

A very brief history of the development of LIDAR is presented in the Background section, along with basic principles of LIDAR data collection. Equations used to determine theoretical relationships between transmitted and received energy levels are included to illustrate the complex relationship between transmitted and returned LIDAR energy. A short survey of the different methods for collecting LIDAR, and a sampling of methods for modeling vegetation are also presented.

The Results and Analysis section of the thesis includes details on each of the model elements, the operation of the model, and the simulation results. Multiple case studies are presented to show the operation of the model, and its sensitivity to varying parameters.

II. BACKGROUND

In 1958, A. L. Schawlow and C. H. Townes published the paper “Infrared and Optical Masers” in the journal *Physical Review*. This paper explained the theory of producing “extremely monochromatic and coherent” light, which is now known as a laser (Schawlow, 1958). This publication sparked a race to build the instrument. In 1960, the first solid-state laser was built by the American physicist Theodore Maiman (Maiman, 1960). Development of the gas laser followed in 1961 (Javan, 1961), and the semiconductor laser in 1962 (Hall, 1962).

The use of the laser as a precision measurement tool was recognized immediately. NASA scientists were using laser-ranging technology to take precise measurements of satellites orbiting the Earth by 1964 (Abbot et al., 1973). By 1965, airborne platforms were being used to measure terrain profiles. By 1969, laser range observations to the moon accurate to within tens of centimeters were being taken at the McDonald Observatory, Fort Davis, Texas, using reflector arrays installed by Apollo astronauts on the moon (Abbot et al., 1973).

The first airborne LIDAR systems were profiling systems, which collect only a single transect of terrain profile at once. Scanning airborne systems, which collect a wide swath of terrain information at once, depend on accurate GPS and Inertial Navigation Systems (INS) geo-referencing capabilities. Scanning systems became viable with the deployment of the GPS constellation in the mid-1990s (Shan, 2009). Space-borne systems are currently limited to profiling due to the difficulties of collecting LIDAR information from space (Shan, 2009).

The applications of LIDAR data are innumerable. The amount of information available, and the accuracy of the data, are unprecedented. LIDAR data provides information not only about the overall heights of objects, but also information about the 3-dimensional distribution of branches, leaves, and the ground underneath as light filters through vegetation to the ground, and back to the sensor.

A. PROBLEM STATEMENT

The motivation for the development of the models presented in this thesis lies in the desire to understand how light transmitted from a LIDAR system interacts with objects and reflects back to the detector, and the sensitivity of the measurement to physical changes in the scene, differing sensor parameters, and sensing geometry. Understanding the influences on the signal recorded by a LIDAR system will allow further information to be extracted from the data.

The first step is to create a model of the system, including the geometric and radiometric properties of the tree, ground surface, and atmosphere. The model must include the LIDAR system transmission and receiving system properties. Finally, the interaction of light with each of the elements of the model must be understood, so that the propagation of light through the tree can be modeled on a step-by-step basis. Because the interaction of light with a tree is very complex in nature, it lends itself to a Monte Carlo simulation, where the results of each single interaction are random, and the overall result is an aggregate of many repeated iterations.

A living tree is a complex biological system. There are many methods for modeling vegetation, with varying degrees of biological realism and visual appeal. In this thesis, two methods are explored. The first is a biologically-realistic model based on methods outlined by de Reffye et al. (1988). The second is an algorithmic curve drawing process developed to study the geometric growth patterns of plants by Aristid Lindenmayer (Prusinkiewicz, 2004). These models provide the geometric framework for the tree model. In modeling the radiometric properties of the tree, the reflectance values of the bark and ground materials are assigned based on published spectral signatures. The PROSPECT leaf reflectance model of Jacquemoud and Baret (1990) was incorporated to model the reflectance, transmittance, and absorbance of the leaves.

B. LIDAR SYSTEM BASIC PRINCIPLES

A LIDAR system measures the time-of-flight between the firing of a laser pulse and the return of reflected energy from the scene. This time-of-flight information is processed into a series of points, which correspond to physical locations in space. Information about the intensity of the reflection is usually also recorded, but at present rarely used in analysis.

There are both pulsed and continuous wave LIDAR systems. A pulsed LIDAR system transmits short pulses of light, and measures the time-of-flight of each pulse to determine distances to objects. A continuous wave system transmits an uninterrupted beam of laser radiation, which has two frequencies—a carrier frequency and a modulation frequency. The carrier frequency of the transmitted laser radiation is modulated by means of the modulation frequency, and this allows very precise measurement of the phase difference between the transmitted and returned waves. Continuous wave systems are mostly used for land-based terrestrial scanning and surveying purposes, while almost all airborne systems are pulsed systems.

For a pulsed system, the travel time, *t*, of a pulse of light is measured quite precisely using the known speed of light, *c*, and the basic formula $R = \frac{tc}{2}$, where *R* is the range to an object, and *t* the time taken for the pulse of light to travel to and from the LIDAR sensor.

Discrete return LIDAR systems measure a series of discrete pulse return events. Full-waveform systems digitize the returned signal over pre-determined time frame, into a set number of samples. Usually, the full-waveform signal is then processed into a series of discrete returns. The advantage of full-waveform data over discrete return data is the higher point density achievable with full-waveform systems. More details on these two types of systems are presented in Section D – Strategies for Collecting LIDAR Data.

C. LIDAR EQUATIONS

The formula found most often in the literature is one that relates the power transmitted P_T to the power received P_R after reflectance. Baltsavias (1999b) provides a step-by-step derivation of the following equation:

$$P_r = \rho \frac{\eta^2 A_r}{\pi R^2} P_T \text{ ,}$$

where ρ, the reflectivity of the target, is the percentage of light reflected from the object, η is the atmospheric transmission factor, R is the range from sensor to target, and A_r is the area of the laser footprint on the ground. The target is assumed to fully fill the area of laser illumination.

While this idealized equation contains many assumptions and simplifications, it nonetheless shows the basic relationship between transmitted and received power. There are many factors which affect the energy returned to the LIDAR sensor.

As a practical example, the Velodyne HDL-64E has a peak transmission power of P_T = 60 Watts and a maximum range of 120 meters (Velodyne, 2007). Using typical values, let the atmospheric factor η = 0.8 and the target reflectance ρ = 0.7. For a receiver area 10 cm^2 (0.01 m^2) in size,

$$P_r = 0.7 \frac{(0.8^2)(0.01m^2)}{\pi(120m)^2} 60W = 5.94 \times 10^{-6} W$$

The amount of power received at the sensor is very much smaller than the amount of power transmitted.

The relationship between laser pulse power, P_{peak} (W), and energy per pulse, E (J), is given by

$$E = P_{peak} t_p \text{ ,}$$

where t_p is the pulse duration in nano-seconds (Baltsavias, 1999b).

Continuing with the example of the Velodyne HDL-64E laser, with a peak power of 60 W and pulse duration of 5 ns, the transmitted energy per pulse is

$$E = 60W * 5\times10^{-9}s = 3\times10^{-7}J$$

The received energy per pulse is

$$E_r = 5.94\times10^{-6}W * 5\times10^{-9}s = 2.97\times10^{-14}J\ .$$

This can be expressed as a number of photons received per pulse N_r, using the following equation:

$$N_r = \eta\frac{E_r}{h\nu},$$

where the Plank constant h = 6.626×10^{-34} J s , v is the laser frequency, and hv is then the energy per photon (Baltsavias, 1999b).

The Velodyne HDL-64E operates a laser at 905 nm wavelength (Velodyne, 2007). Assume a detector quantum efficiency of 0.3, and received energy $E_r = 2.97\times10^{-14}J$. The frequency of a laser is related to the wavelength via

$$v = \frac{c}{\lambda},$$

where c is the speed of light, v is the frequency, and λ is the wavelength. For a laser operating at 905 nm, the frequency v = $\frac{2.998\times10^{8}m/s}{905\times10^{-9}m}$ = 3.313×10^{14} Hz. Therefore, the number of photons received per pulse should be approximately:

$$N_r = 0.3\frac{2.97\times10^{-14}J}{6.626\times10^{-34}Js * 3.313\times10^{14}s^{-1}} = 135,308 \text{ photons.}$$

D. STRATEGIES FOR COLLECTING LIDAR DATA

The following section gives a brief introduction to some of the ways in which LIDAR data is collected. The paper by Mallet, "Full-waveform topographic lidar: State-of-the-art," contains a good overview of current systems, with particular emphasis on full-waveform systems (Mallet, 2009).

1. Discrete Return Systems

Discrete return LIDAR is the most traditional LIDAR system. In a discrete return system, a high-energy laser pulse is emitted, and the time at which energy is returned to the sensor is measured. The detection device is triggered by means of a threshold. When enough energy is entering the detection system, the detector is turned on. The threshold is set high enough to ignore the effects of background light, such as sunshine. The detector is switched off when the level of reflected light falls below the threshold value. Multiple returns, typically 4–5 at most, can be detected in this way. Since the energy must fall below the threshold level in order to reset the detector, returns from objects in the scene that are located close together may not be distinguishable.

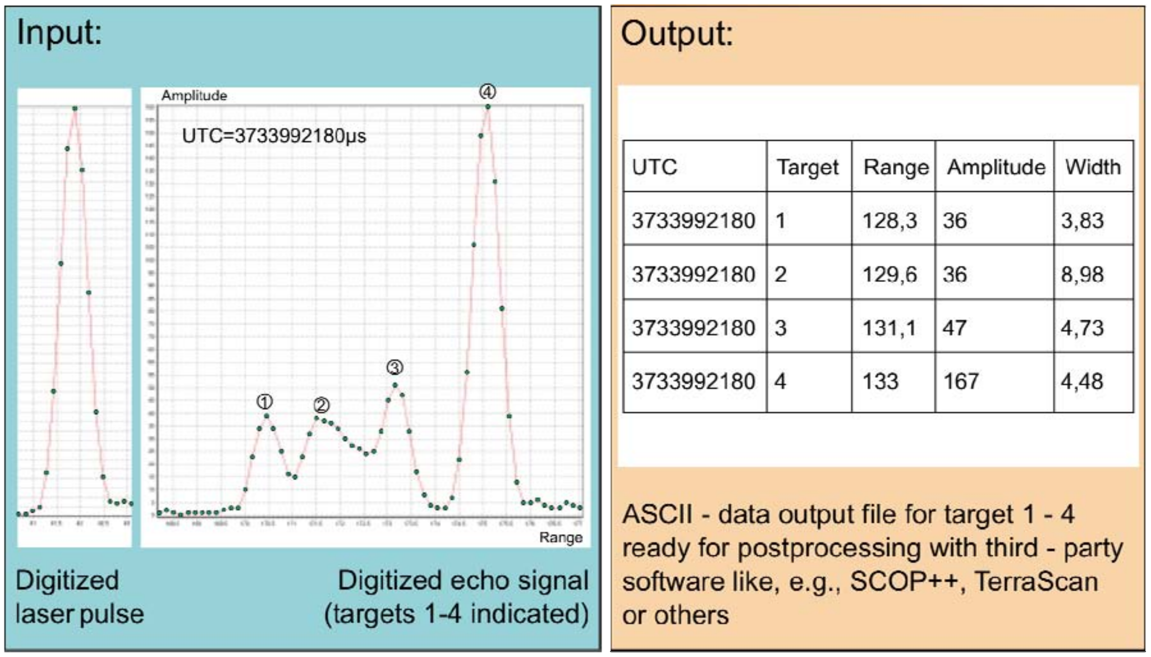
The travel time of a pulse of light emitted from a LIDAR system is $t = \frac{2R}{c}$, where R is the ranging distance of the system, and c is the speed of light. The maximum pulse frequency, assuming no transmit/receive overlap, is therefore $1/t = \frac{c}{2R}$. The range resolution is $\Delta R = \frac{1}{2} c \Delta t$ (Baltsavias, 1999b). For a typical LIDAR system with a pulse width of 10 ns, which equates to a distance of about 3 m, the range resolution is nominally 1.5 m.

While somewhat out of date, Baltsavias (1999a) presents a very complete survey of LIDAR providers, along with specification of the then-current systems. This paper is a good source of background information. Probably the largest change in a decade has been an increase in the pulse repetition frequency—systems with PRFs up to 250 kHz are now being sold.

2. Full-waveform Systems

In a full-waveform system, the recorder is triggered according to a threshold. A predefined maximum number of measurements are made after this threshold is crossed. For the current Optech ALTM systems, 440 samples per pulse can be recorded, which corresponds to a vertical distribution of at most 66 m (Mallet, 2009).

The entire full-waveform signal is recorded, but in most cases the data is post-processed by finding peaks within the waveform signal. These peaks correspond to object reflections, and are used to generate a 3D point cloud.



UTC	Target	Range	Amplitude	Width
3733992180	1	128,3	36	3,83
3733992180	2	129,6	36	8,98
3733992180	3	131,1	47	4,73
3733992180	4	133	167	4,48

ASCII - data output file for target 1 - 4 ready for postprocessing with third - party software like, e.g., SCOP++, TerraScan or others

Figure 1. Inputs and outputs to the RiAnalyze 560 software from Riegl Laser Management Systems (From Riegl, 2007)

Figure 1 shows the input and output for Riegl Laser Measurement Systems' RiAnalyze 560 software. The LIDAR pulse is digitized as the energy returns to the system at a predefined sampling frequency. The peaks of the digitized waveform are found using a waveform analysis algorithm. The RiAnalyze software implements three different waveform analysis algorithms, but details on the specifics of the algorithm are not provided (Riegl, 2007). Mallet (2009) provides an overview of the types of “waveform modeling” or “waveform fitting” algorithms. Typically, the output from waveform analysis algorithms includes a range, intensity (amplitude of peak), and width for each waveform peak.

The advantage of waveform LIDAR data over discrete return LIDAR is that a denser 3-D point cloud can be achieved. Unfortunately, according to Mallet (2009), this denser point cloud does not allow fine material classification, even when information about the shape of the modeled waveform is included. Examples of typical waveforms returned from the Shuttle Laser Altimeter II system are shown in Figure 2. Coarse classifications from large footprint full-waveform LIDAR are possible; an example is the ICESAT system (70 m diameter footprint), which can provide waveforms classified as land, ice sheet, sea ice or ocean (Mallet, 2009).

The Shuttle Laser Altimeter II (SLA-02) was flown on STS-85, which launched on August 7, and landed on August 18, 1997. The SLA-02 system had a 100 m footprint on the ground. With such a large footprint, the system is optimized for landscape level mapping. In Figure 2, some typical waveforms are shown, along with the material classification.

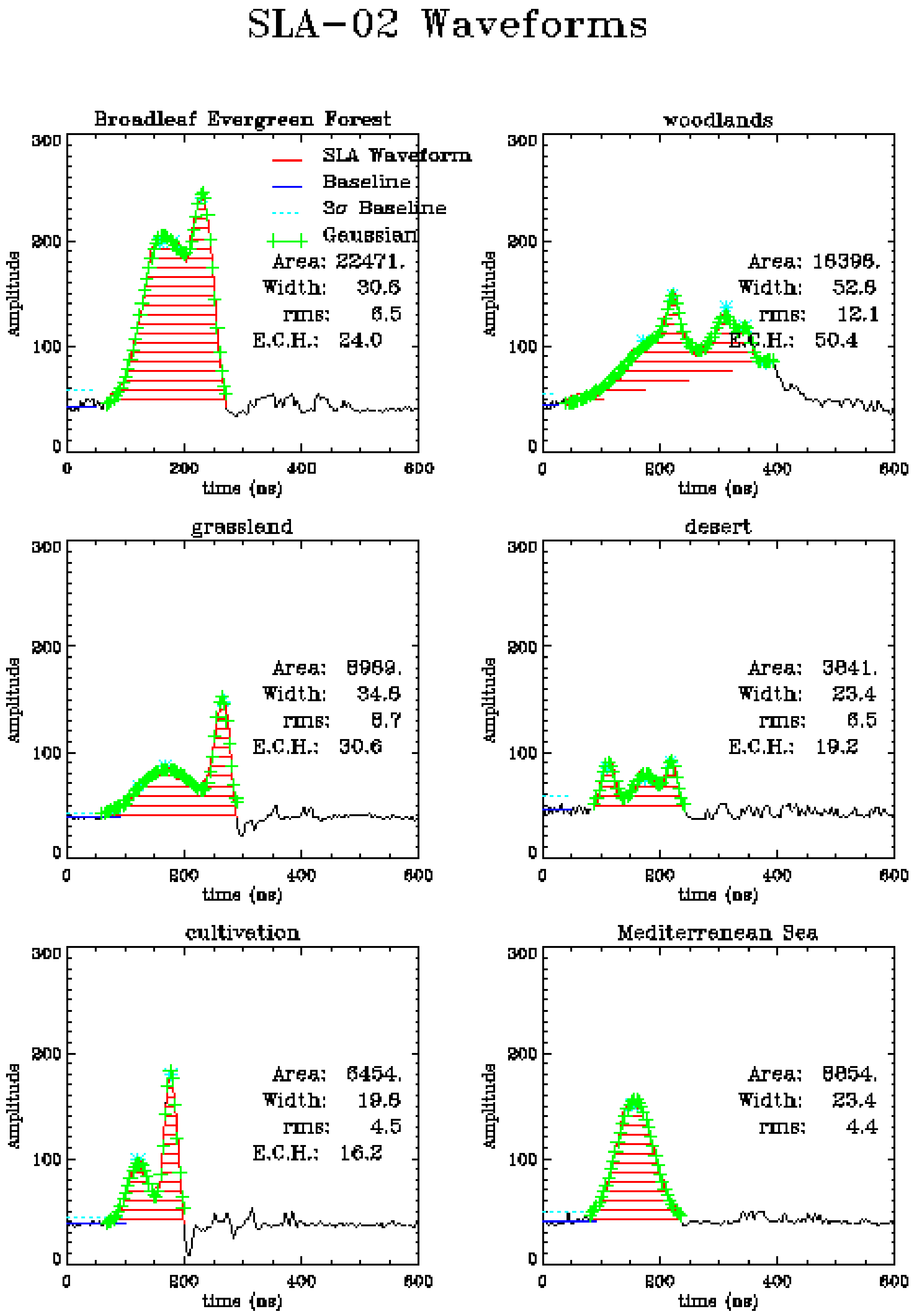


Figure 2. A graphic from NASA which shows “A sampler of 6 representative echoes, including rough Mediterrannean water, rugged desert, and various forms of vegetated landscapes” from the SLA-02 instrument (From NASA, 1997)

Relatively little research has been done on using the waveform directly, but Mallet (2009) provided a list of some current research efforts which focus on elements which could effect the shape of waveforms, including distance to sensor and emission angle, surface roughness, target geometry, target reflectivity, and atmospheric effects.

Some typical specifications of full-waveform LIDAR systems taken from Mallet (2009) are shown in Figure 3.

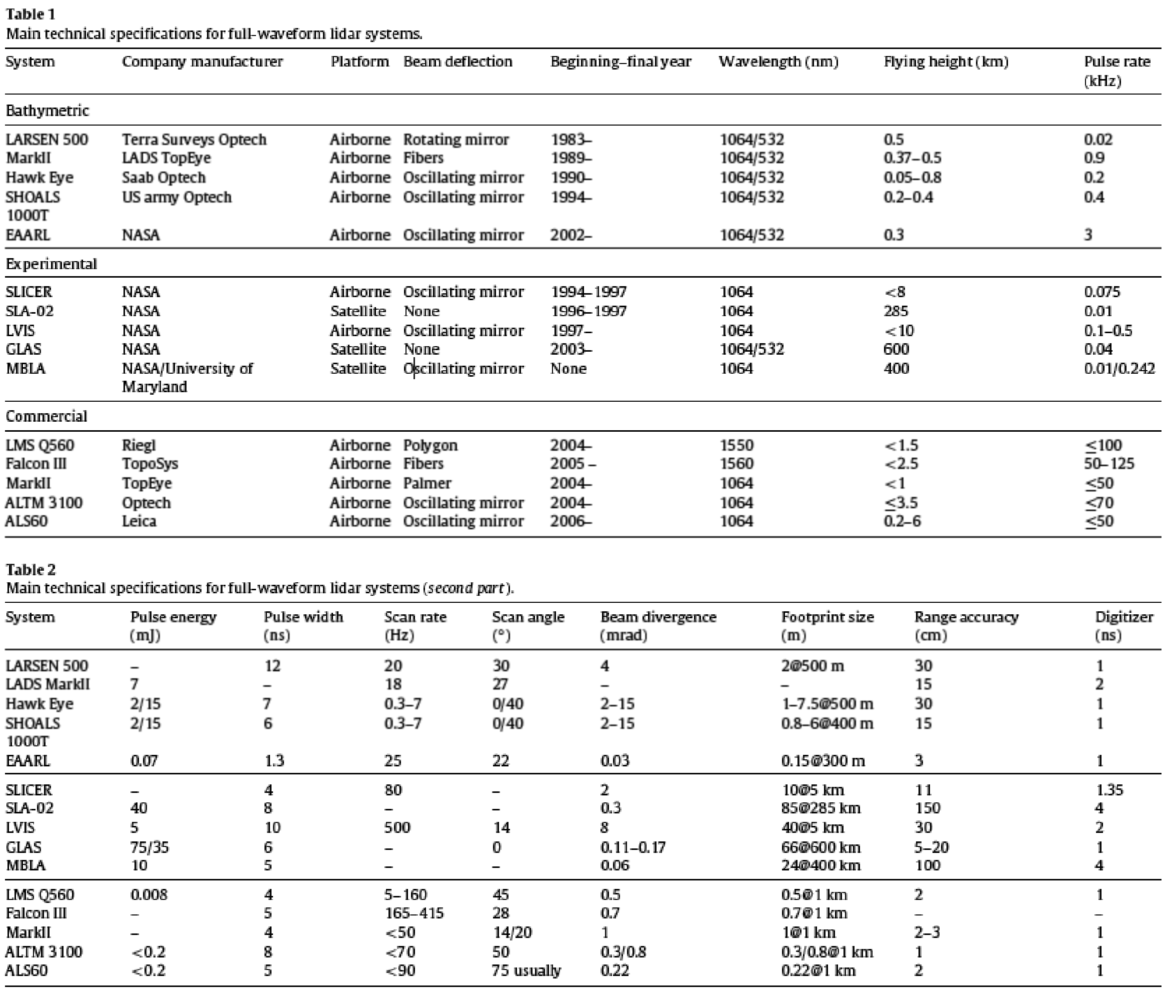


Table 1
Main technical specifications for full-waveform lidar systems.

System	Company manufacturer	Platform	Beam deflection	Beginning–final year	Wavelength (nm)	Flying height (km)	Pulse rate (kHz)
Bathymetric							
LARSEN 500	Terra Surveys Optech	Airborne	Rotating mirror	1983–	1064/532	0.5	0.02
MarkII	LADS TopEye	Airborne	Fibers	1989–	1064/532	0.37–0.5	0.9
Hawk Eye	Saab Optech	Airborne	Oscillating mirror	1990–	1064/532	0.05–0.8	0.2
SHOALS 1000T	US army Optech	Airborne	Oscillating mirror	1994–	1064/532	0.2–0.4	0.4
EAARL	NASA	Airborne	Oscillating mirror	2002–	1064/532	0.3	3
Experimental							
SLICER	NASA	Airborne	Oscillating mirror	1994–1997	1064	<8	0.075
SLA-02	NASA	Satellite	None	1996–1997	1064	285	0.01
LVIS	NASA	Airborne	Oscillating mirror	1997–	1064	<10	0.1–0.5
GLAS	NASA	Satellite	None	2003–	1064/532	600	0.04
MBLA	NASA/University of Maryland	Satellite	Oscillating mirror	None	1064	400	0.01/0.242
Commercial							
LMS Q560	Riegl	Airborne	Polygon	2004–	1550	<1.5	≤100
Falcon III	TopoSys	Airborne	Fibers	2005 –	1560	<2.5	50–125
MarkII	TopEye	Airborne	Palmer	2004–	1064	<1	≤50
ALTM 3100	Optech	Airborne	Oscillating mirror	2004–	1064	≤3.5	≤70
ALS60	Leica	Airborne	Oscillating mirror	2006–	1064	0.2–6	≤50

Table 2
Main technical specifications for full-waveform lidar systems (*second part*).

System	Pulse energy (mJ)	Pulse width (ns)	Scan rate (Hz)	Scan angle (°)	Beam divergence (mrad)	Footprint size (m)	Range accuracy (cm)	Digitizer (ns)
LARSEN 500	–	12	20	30	4	2@500 m	30	1
LADS MarkII	7	–	18	27	–	–	15	2
Hawk Eye	2/15	7	0.3–7	0/40	2–15	1–7.5@500 m	30	1
SHOALS 1000T	2/15	6	0.3–7	0/40	2–15	0.8–6@400 m	15	1
EAARL	0.07	1.3	25	22	0.03	0.15@300 m	3	1
SLICER	–	4	80	–	2	10@5 km	11	1.35
SLA-02	40	8	–	–	0.3	85@285 km	150	4
LVIS	5	10	500	14	8	40@5 km	30	2
GLAS	75/35	6	–	0	0.11–0.17	66@600 km	5–20	1
MBLA	10	5	–	–	0.06	24@400 km	100	4
LMS Q560	0.008	4	5–160	45	0.5	0.5@1 km	2	1
Falcon III	–	5	165–415	28	0.7	0.7@1 km	–	–
MarkII	–	4	<50	14/20	1	1@1 km	2–3	1
ALTM 3100	<0.2	8	<70	50	0.3/0.8	0.3/0.8@1 km	1	1
ALS60	<0.2	5	<90	75 usually	0.22	0.22@1 km	2	1

Figure 3. Full-waveform LIDAR systems main technical specifications (From Mallet, 2009)

3. Photon-counting Systems

Photon-counting systems rely on a multitude of low-intensity pulses, emitted either simultaneously, or at a very high frequency of repetition. This is different from traditional LIDAR systems, which emit a series of high-intensity pulses at a lower pulse repetition frequency.

A theoretical system described by Degnan (2002) and a system designed by Massachusetts Institute of Technology, Lincoln Laboratory (MIT/LL) use photon-counting focal planes, which record multiple observations simultaneously (Marino, 2009). The MIT/LL system emits a set of pulses at once using a “flash” system that does not require any moving parts. Degnan’s system emits a high-frequency train of pulses.

The main advantage of a photon-counting LIDAR system is that much less power is needed for the laser, both in transmission and detection. A single photon return can be detected. This allows the pulse repetition frequency (PRF) of the system to be increased greatly, which means a system can realistically be designed that has low power requirements, but still allows large areas of coverage with high resolution (Degnan, 2002).

Another advantage of a photon-counting system is the inherent accuracy of the system. A traditional LIDAR with a high-intensity pulse depends on recording the returned waveform over time, and then detecting a series of peaks in the signal that correspond to returns from objects. There will be ambiguity in the measurement of the location of this peak. Mallet (2009) cites several studies that show that this measurement is critically dependent on the algorithm used. There will also be ambiguity due to the larger spatial coverage of the laser footprint. The waveform is made up of all of the photons returning from within the laser footprint, which can contain multiple different materials. With a photon-counting system, the measurement is more like a point-to-point measurement, with each photon being measured one at a time.

The disadvantage, or main difficulty, of a photon-counting system is that it becomes harder to distinguish the returned signal from the background noise of solar illumination. This difficulty can be overcome using range-gating methods, in which a prediction about the time-of-flight of a LIDAR pulse is made based on the flying height of the sensor and time taken for preceding pulses. Returns occurring outside the expected time-range are ignored. Figure 4 is an illustration of the range-gating concept taken from Degnan (2002).

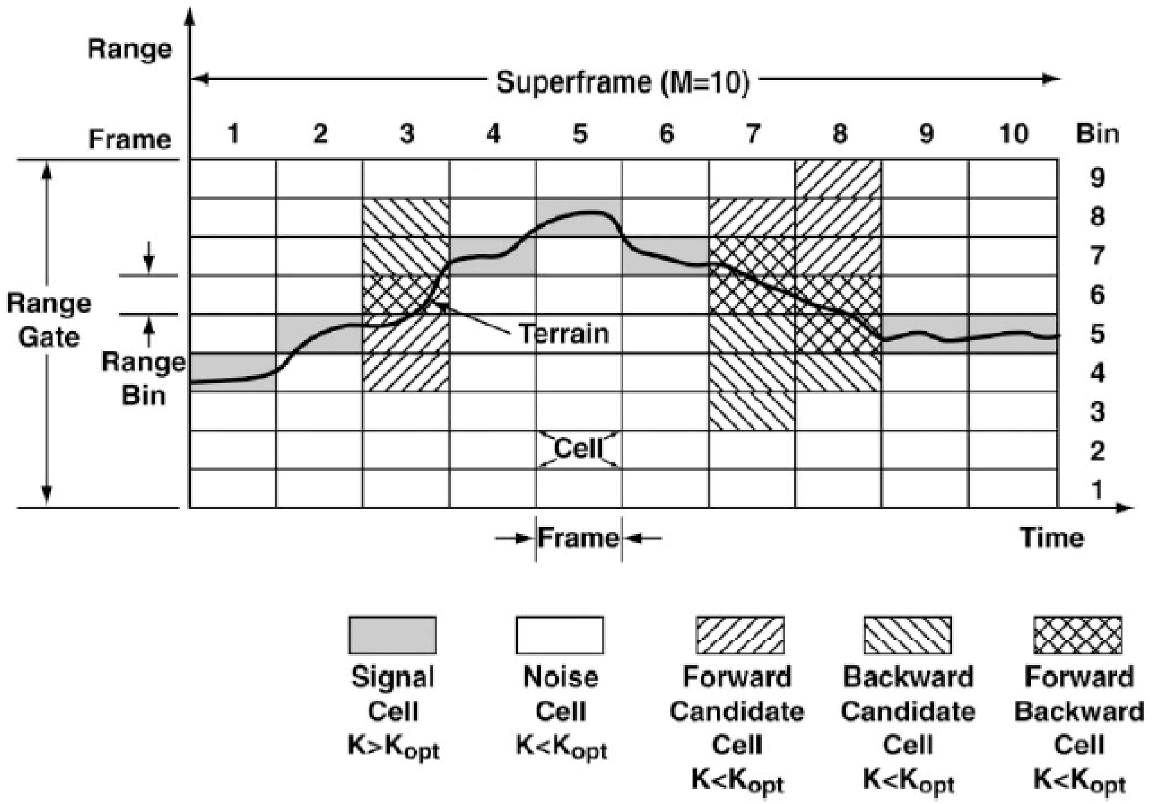


Figure 4. Illustration of range-gating concept (From Degnan, 2002)

The image caption in Degnan (2002) is as follows:

Principles behind a correlation range receiver. The range gate is divided into equal duration range bins, and several consecutive laser fires are combined to form a frame. The 2-D area bounded by the range bin and frame dividing lines is a cell. Multiple frames form a superframe. Photon counts are accumulated within each cell and, if the total count K exceeds the frame threshold Kopt, the cell is identified as signal; otherwise it is tentatively identified as noise. A valid trajectory is one in which the signal cell is not displaced by more than one range bin in consecutive frames. Applying this criteria in algorithms which look forward and backward in time and an N of M criteria can help recover lost or missing data in near real time. (p. 508)

For examples of this type of system, see results of an experimental NASA program presented in Degnan (2002) or the paper by Harding et al. (2007), which presented a prototype photon-counting LIDAR system called the Swath Imaging Multi-polarization Photon-counting LIDAR (SIMPL).

Parameters for a photon-counting LIDAR as proposed by Degnan (2002) are shown in Figure 5.

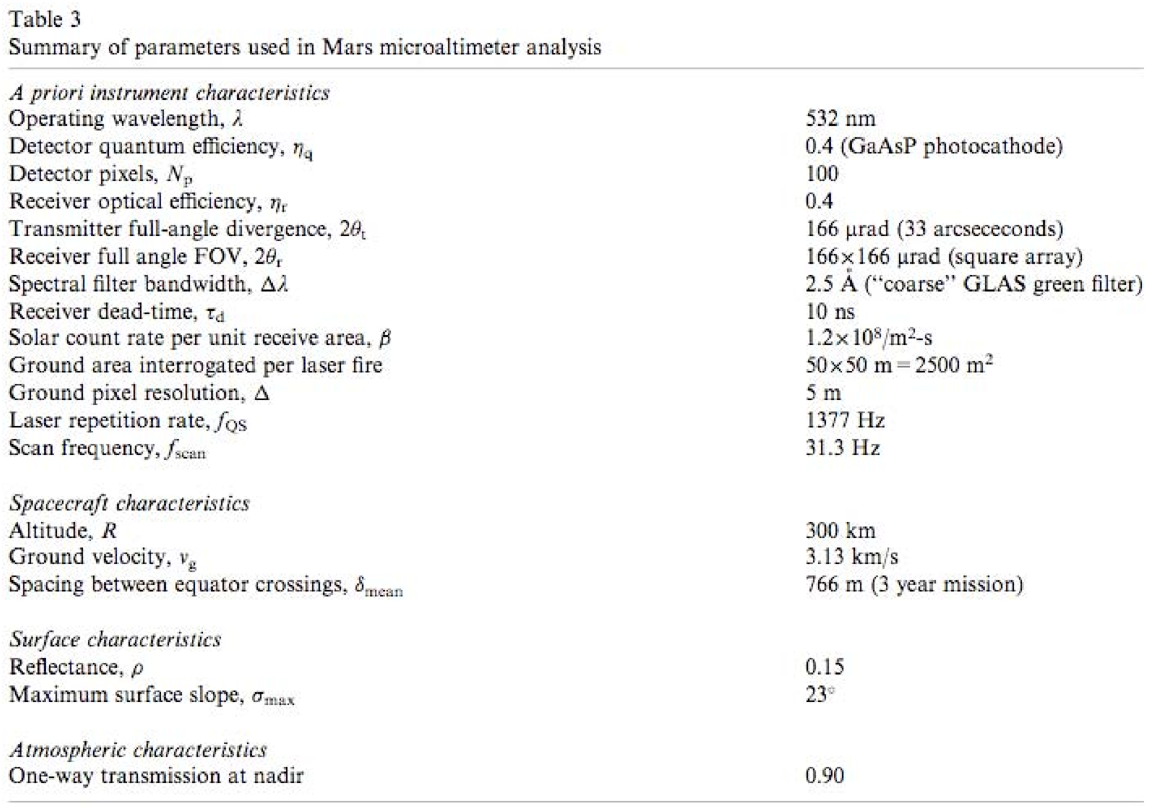


Table 3
Summary of parameters used in Mars microaltimeter analysis

A priori instrument characteristics	
Operating wavelength, λ	532 nm
Detector quantum efficiency, η_q	0.4 (GaAsP photocathode)
Detector pixels, N_p	100
Receiver optical efficiency, η_r	0.4
Transmitter full-angle divergence, $2\theta_t$	166 μrad (33 arcsececonds)
Receiver full angle FOV, $2\theta_r$	166×166 μrad (square array)
Spectral filter bandwidth, $\Delta\lambda$	2.5 Å (“coarse” GLAS green filter)
Receiver dead-time, τ_d	10 ns
Solar count rate per unit receive area, β	1.2×10^8/m^2-s
Ground area interrogated per laser fire	50×50 m $=2500$ m^2
Ground pixel resolution, Δ	5 m
Laser repetition rate, f_{QS}	1377 Hz
Scan frequency, f_{scan}	31.3 Hz
Spacecraft characteristics	
Altitude, R	300 km
Ground velocity, v_g	3.13 km/s
Spacing between equator crossings, δ_{mean}	766 m (3 year mission)
Surface characteristics	
Reflectance, ρ	0.15
Maximum surface slope, σ_{max}	23°
Atmospheric characteristics	
One-way transmission at nadir	0.90

Figure 5. Specifications used in design of a proposed photon-counting LIDAR system for use in mapping the Martian surface (From Degnan, 2002)

4. Synthetic Aperture LIDAR

Synthetic Aperture LIDAR is still very much an experimental technology, but builds on a long history of development of Synthetic Aperture RADAR (SAR). The principle of Synthetic Aperture LIDAR is the same as SAR, but uses much shorter optical wavelengths. This allows for the creation of a more detailed image, and a shorter collection time. Lucke and Rickard (2002) present the theoretical background of a Synthetic Aperture LIDAR system.

Recent research efforts have been sponsored to develop this technology. DARPA (Defense Advanced Research Projects Agency) sponsors the SALTI (Synthetic Aperture Ladar for Tactical Imaging) program, which seeks to foster this technology. According to a 2006 news release, airborne synthetic aperture LIDAR images were acquired by both Raytheon and Northrup Grumman experimental systems (DARPA, 2006). A February 2008 Broad Agency Announcement calls for proposals to "develop and demonstrate a Synthetic Aperture Laser RADAR (LADAR) sensor capable of long range, high resolution synthetic aperture imaging from a Contractor operated aircraft to demonstrate performance and validate readiness for transition to an operational customer" (DARPA, 2007).

E. STRATEGIES FOR MODELING VEGETATION

Many options exist for modeling vegetation with varying degrees of biological realism and visual appeal; two methods are presented here.

1. De Reffye Model

The de Reffye model is a probabilistic method that captures the continuous growth of a plant. To understand the algorithm, a brief introduction to biological terminology is required. The following set of figures introduces the biological terminology (de Reffye, 1988).

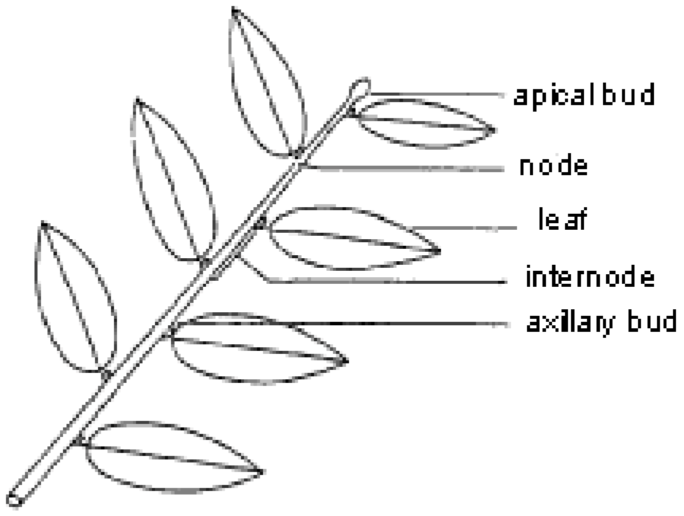


Figure 6. Introduction of biological terminology (From de Reffye, 1998, p. 152)

The apical bud is at the tip of the stem, and is the location where growth may occur. The axillary buds are locations where leaves may grow. The internode represents the length of stem between axillary buds.

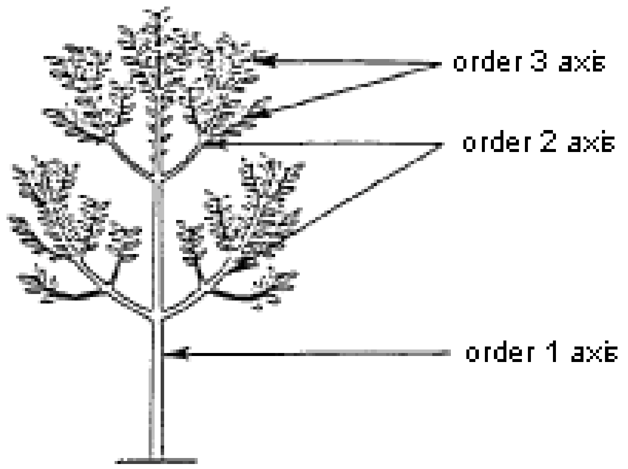


Figure 7. The order of axes (From de Reffye, 1998, p. 152)

The order of the axis is important biologically (and algorithmically) because it determines the behavior of the plant. The length of growth units and the thicknesses of branches tend to decrease with higher-order axes.

De Reffye's algorithm also allows incorporation of different leaf arrangements, branch arrangements, and branching angles, with the expression of each being determined probabilistically. While the probabilistic nature of the algorithm creates physically realistic tree models, it makes the algorithm difficult to use for the purposes of this thesis, in which the shape of the tree must be controlled carefully to determine the effect of minor geometric changes on the simulated LIDAR waveforms.

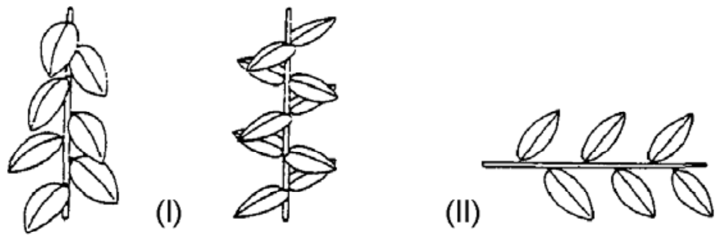


Figure 8. "Phyllotaxy: (I) spiraled, (II) dystic" (From de Reffye, 1998, p. 152)

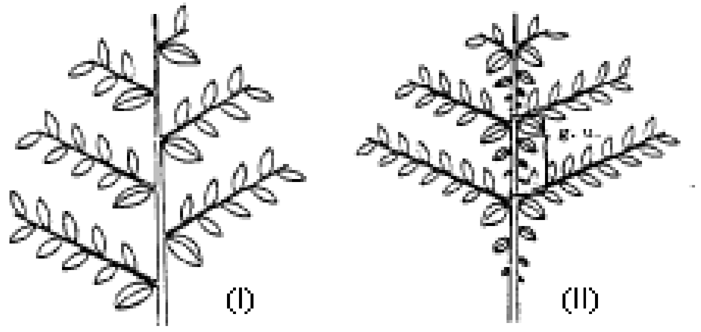


Figure 9. "Ramification (I) continuous, (II) rhythmic" (From de Reffye (1998, p. 152)

Pseudocode for the plant growth algorithm adapted from the original de Reffye paper is given in Computer Graphics Principles and Practice by Foley, van Dam, Feiner, and Hughes (Foley, 1990, p. 1031).

```
for each clock time do
    for each bud that is still alive do
        begin
            determine from order, age, etc., what happens to bud;
            if bud does not die then
                if bud does not sleep then
                    begin
                        create an internode
                            (with geometric information about its position,
                            direction, etc.);
                        create an apical bud;
                        for each possible bud at old bud location do
                            if ramification then create axillary buds
                        end
        end
```

De Reffye's model is a stochastic algorithm. A bud is assigned a probability of growing, dying, or "resting," based on the age of the tree overall, the age of the branch where the bud occurs, and the order of the branch. All facets of the growth of the plant, including the locations of leaves, the angle of branches, locations of cones or flowers, etc., can be controlled in a similar fashion by assigning probabilities of occurrence. The options are unlimited, and the models created using this method are very realistic looking. A few examples from de Reffye's paper are included in Figure 10.

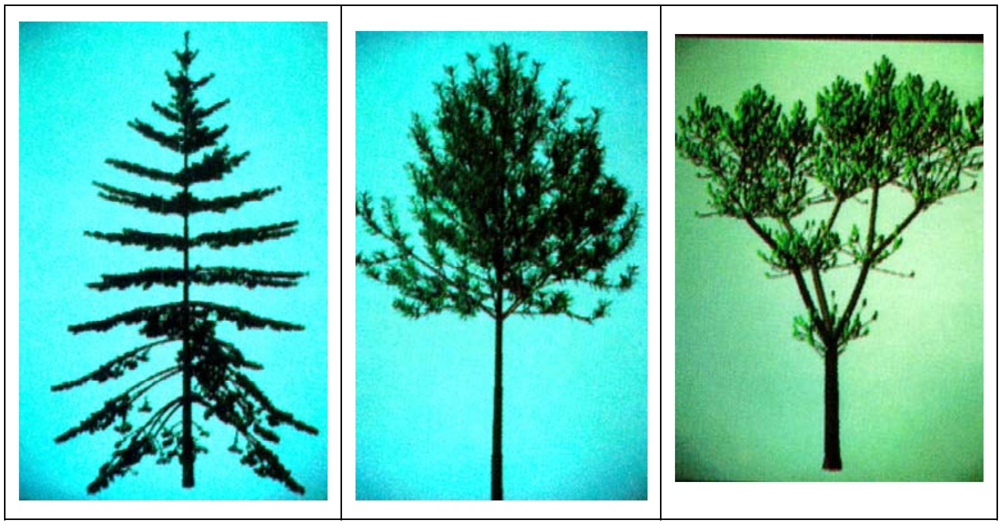


Figure 10. Fir tree, pine tree, and pruned tree (with traumatic reiterations) (From de Reffye, 1998, p. 156)

For the purposes of this thesis, de Reffye's algorithm was not ideal because of the probabilistic nature of the model tree generation. Each application of the algorithm creates a unique tree. While this is physically realistic, the purpose of this thesis is to study the influences affecting full-waveform LIDAR signatures. This requires the ability to control and change the shape of the tree, which becomes very difficult to do using de Reffye's algorithm. The L-system method proved to be more appropriate for this purpose.

2. L-systems

In 1956, Noam Chomsky wrote a paper titled "Three models for the description of Language" (Chomsky, 1956), in which he applied the "concept of rewriting to describe the syntactic features of natural languages" (Prusinkiewicz, 2004). This led to a widespread interest in methods for "generating, recognizing and transforming formal languages" (Prusinkiewicz, 2004).

In 1968, Aristid Lindenmayer developed the Lindenmayer-systems, or L-systems for short, to study the geometric growth patterns of plants (Prusinkiewicz, 2004). The L-system concept was initially designed to study the development of simple multi-cellular organisms, and was later expanded by other researchers to include the interpretive rendering of pictures (such as plants and trees) (Prusinkiewicz, 2004).

The L-system method is at its core a "string-rewriting mechanism," in which parts of a simple string are replaced using a set of rewriting rules known as "productions." The string is interpreted as a blueprint or schematic of the organism being modeled.

Prusinkiewicz (2004) gives the following formal definition of a "deterministic 0L-system":

> Let V denote an L-system alphabet, V^* the set of all words over V, and V^+ the set of all non-empty words over V.
>
> A 0L-system is an ordered triplet $G = \{V, \omega, P\}$, where V is the **alphabet** of the system, $\omega \in V^+$ is a nonempty word called the **axiom** and $P \subset V x V^+$ is a finite set of **productions**. If a pair (a, χ) is a production, we write $a \rightarrow \chi$. The letter 'a' and the word χ are called the **predecessor** and the **successor** of this production respectively. It is assumed that for any letter $a \in V$, there is at least one **word** $\chi \in V^*$ such that $a \rightarrow \chi$. If no production is explicitly specified for a given predecessor $a \in V$, the identity production $a \rightarrow a$ is assumed to belong to the set of productions P. A 0L-system is **deterministic** IFF for each $a \in V$ there is exactly one $\chi \in V^*$ such that $a \rightarrow \chi$ (p. 4).

The simplest of the L-systems is referred to as a "D0L-system"; the D for "deterministic," and the "0L" for 0-context, or context free. A system that is context free is one in which the production rules apply regardless of the context of the symbol in the string. A deterministic system refers to a system in which only one outcome is possible given the starting state and production rules (Prusinkiewicz, 2004).

De Reffye's algorithm is a grammar-based model, with each bud being treated as an "axiom." The major difference between de Reffye's model and L-systems is that the de Reffye method is not deterministic.

The following example from *The Algorithmic Beauty of Plants* (Prusinkiewicz, 2004) introduces the L-System concept. Let $V = \{a, b\}$, $\omega = b$, and $P = \{a \rightarrow ab, b \rightarrow a\}$. In the first step, the axiom "b" is replaced by "a" according to the production rule "$b \rightarrow a$." In the second step, "a" is replaced with "ab" according to the production rule "$a \rightarrow ab$." In the following step, "a" is replaced by "ab," and at the same time, "b" is replaced by "a." The production rules being applied in parallel, rather than in sequence, is the primary characteristic that distinguishes L-systems from Chomsky grammars (Prusinkiewicz, 2004).

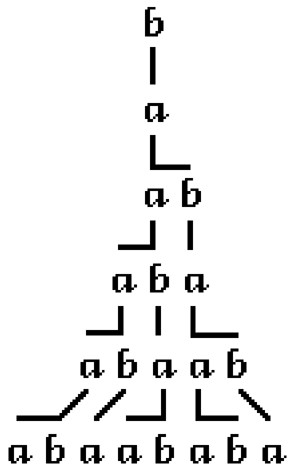


Figure 11. Example of an L-System string generation (From Prusinkiewicz (2004, p. 4)

The L-System model consists of two parts—the first being creation of the string using the grammar rules, and the second being translation of the string into a graphical representation or picture. This translation is accomplished by means of a “turtle,” which interprets the strings developed by an L-system.

A turtle is a way of defining the graphical interpretation of each symbol in the L-system string. The state of the turtle is defined by an ordered triplet (x, y, δ), where “x” and “y” refer to the location of the turtle, and “δ” refers to the heading. Each symbol in the alphabet V can be associated with graphical objects, having a length, color, dimension, etc.

For example, again from *The Algorithmic Beauty of Plants,* the following figure shows the graphical representation of an L-system with V = {F, +, −}, ω = F, and production rules P = {F → FFF-FF-F-F+F+FF-F-FFF, + → +, − → −} (Prusinkiewicz, 2004). The turtle interpretation of the string symbols F, +, and − is represented on the left. On the right is the graphical interpretation of first iteration of the L-system.

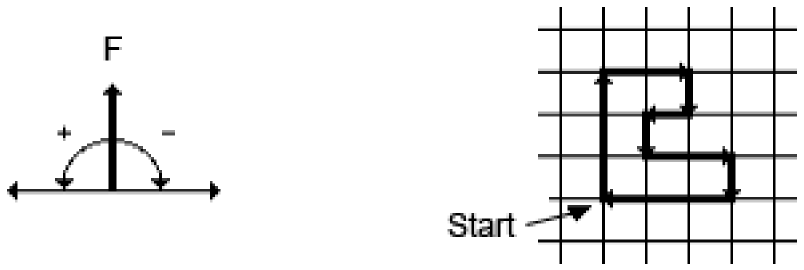


Figure 12. Turtle interpretation of L-system grammar (From Prusinkiewicz, 2004, p. 7)

In this case, the symbols are interpreted as follows:

F — draw a line of length *d* in direction of heading

+ — turn clockwise 90 degrees

− — turn counterclockwise 90 degrees

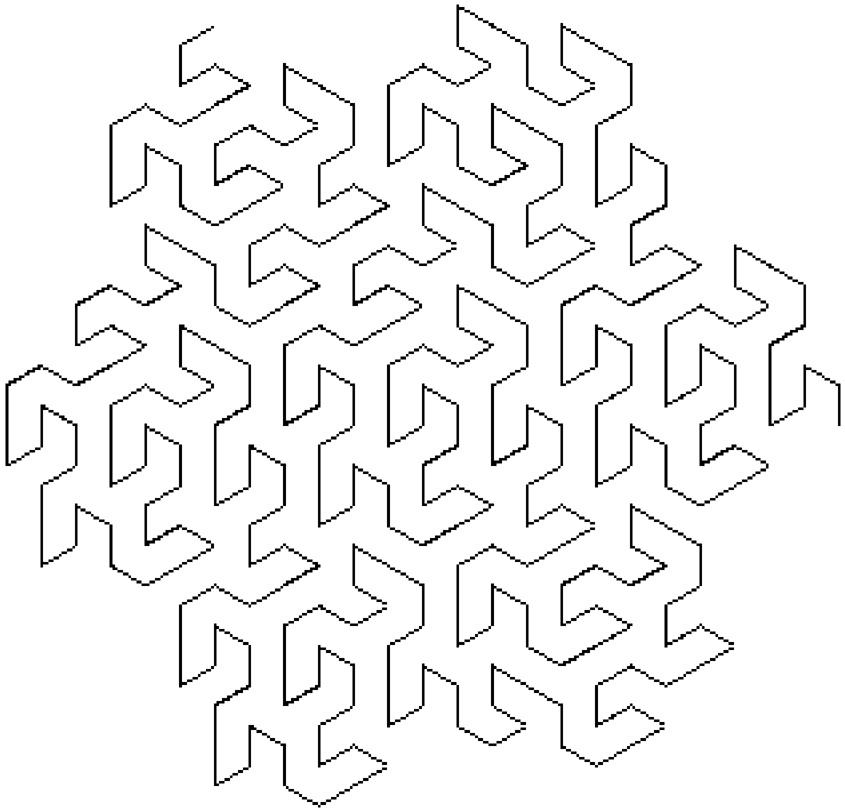
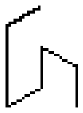
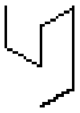
	
Gosper curve as defined by M. Gardner in “Mathematical games—in which ‘monster’ curves force redefinition of the word ‘curve’.” Scientific American, 235(6):124–134, December 1976 as referenced by Prusinkiewicz in the Algorithmic Beauty of Plants; rendered using the authors code for generating L-system trees.	
	
The “0” production statement turtle interpretation, where 0 → 0 + 1 + + 1 - 0 − − 0 0 − 1 +	The “1” production statement turtle interpretation, where 1 → − 0 + 1 1 + + 1 + 0 − − 0 − 1

Figure 13. Gosper curve

The L-System used to create the Gosper curve shown in Figure 13 is defined as V = {1, 0, +, −}, ω = 0, δ = 60°, and P = {0 → 0 + 1 + + 1 − 0 − − 0 0 − 1 +, 1 → − 0 + 1 1 + + 1 + 0 − − 0 – 1, + → +, and − → −}.

The “+” is interpreted by the turtle as a change in heading of δ = 60 degrees counter-clockwise, and the “-” symbol is an equal turn in the clockwise direction.

Further examples of the beautiful and complex shapes created using L-systems can be found in (Prusinkiewicz, 2004). Other applications of L-systems with turtle interpretation include “realistic modeling of herbaceous plants, description of kolam patterns (an art form from Southern India), synthesis of musical scores and automatic generation of spacefilling curves” (Prusinkiewicz, 2004). This thesis will focus on using L-systems to create models of trees.

To create trees, there must be a mechanism for creating branches. This is accomplished by means of brackets, “[ ]”. In the L-system, the brackets are treated as constants, so each symbol maps to itself; i.e., $\{[\rightarrow [\ ,\ ] \rightarrow]\} \subset P$. In the turtle interpretation process, when a “[” is encountered, the current state of the turtle (location and heading) must be recorded, so that when the “]” symbol is encountered, the branch will be completed, and the turtle can return to that state. The example illustrated in Figure 14 will help to clarify this point.

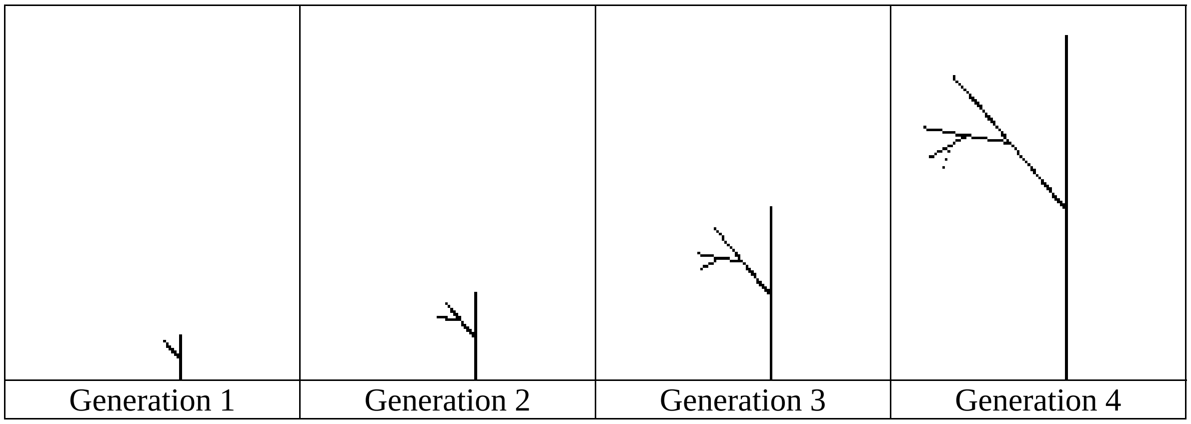


Figure 14. Four generations of an L-system tree. The L-System is defined as: $V = \{1, 0, [,]\}$, $\omega = 0$, $\delta = 40°$, $P = \{0 \rightarrow 1[0]1,\ 1 \rightarrow 11,\ [\rightarrow [\ ,\ ] \rightarrow]\}$

In the turtle interpretation of this tree, either 0 or 1 generates a straight segment of length *d*. The left-bracket, “[” tells the turtle to start a branch. When a new branch is created, the location and heading of the turtle is noted. The bracket symbol also serves to identify the angle at which the branch will be created. The left-bracket symbol “[” starts a branch that is δ degrees counter-clockwise from the current branch, and the right-bracket symbol “]” signifies the end of the branch. A second set of symbols, “(” and “),” can be used to create branches that turn the turtle in the clockwise direction.

The L-system tree shown in Figure 14 is created as follows:

Step 0: 0 The axiom to start the tree string is 0.

Step 1: 1[0]1 The 0 is replaced by 1[0]1.

Step 2: 11[1[0]1]11 Each 1 is replaced by 11, each 0 is replaced by 1[0]1, and brackets are treated as identities.

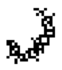
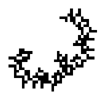
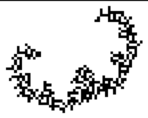
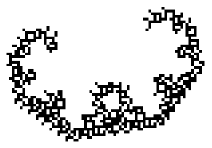
Step 3: 1111[11[1[0]1]11]1111

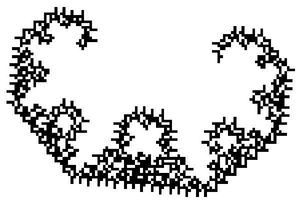
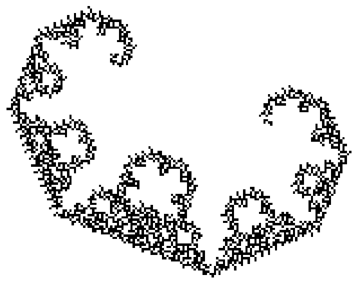
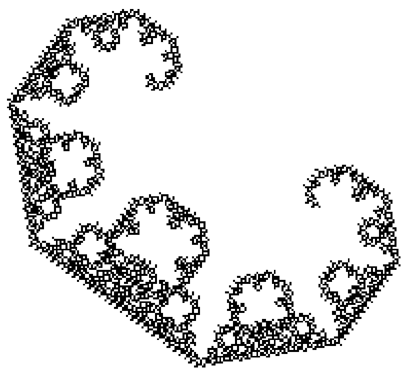
Step 4: 11111111 [1111[11[1[0]1]11]1111]11111111

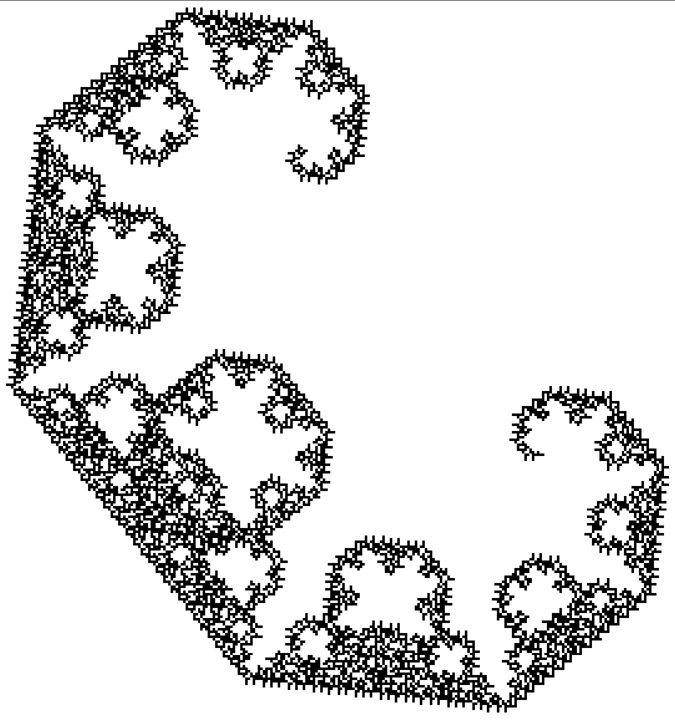
The IDL code used to create the L-system tree, and render the graphics in the figure above, is included in Appendix C.

In the example above, all of the branches point to the left. A second set of symbols, namely "(" and ")", can be used to create branches to the right. The turtle interprets "[" and "]" as branches with a counterclockwise change in heading direction, while "(" and ")" are interpreted as branches created with a clockwise change in heading direction.

An example of the complicated shapes achievable with this method is shown in Figure 15. This example is included as a neat illustration of the capabilities of the L-system concept.

							
Gen. 1	Gen. 2	Gen. 3	Gen 4.	Gen 5.	Gen. 6	Generation 7	Generation 8

		
Generation 9	Generation 10	Generation 11


Generation 12

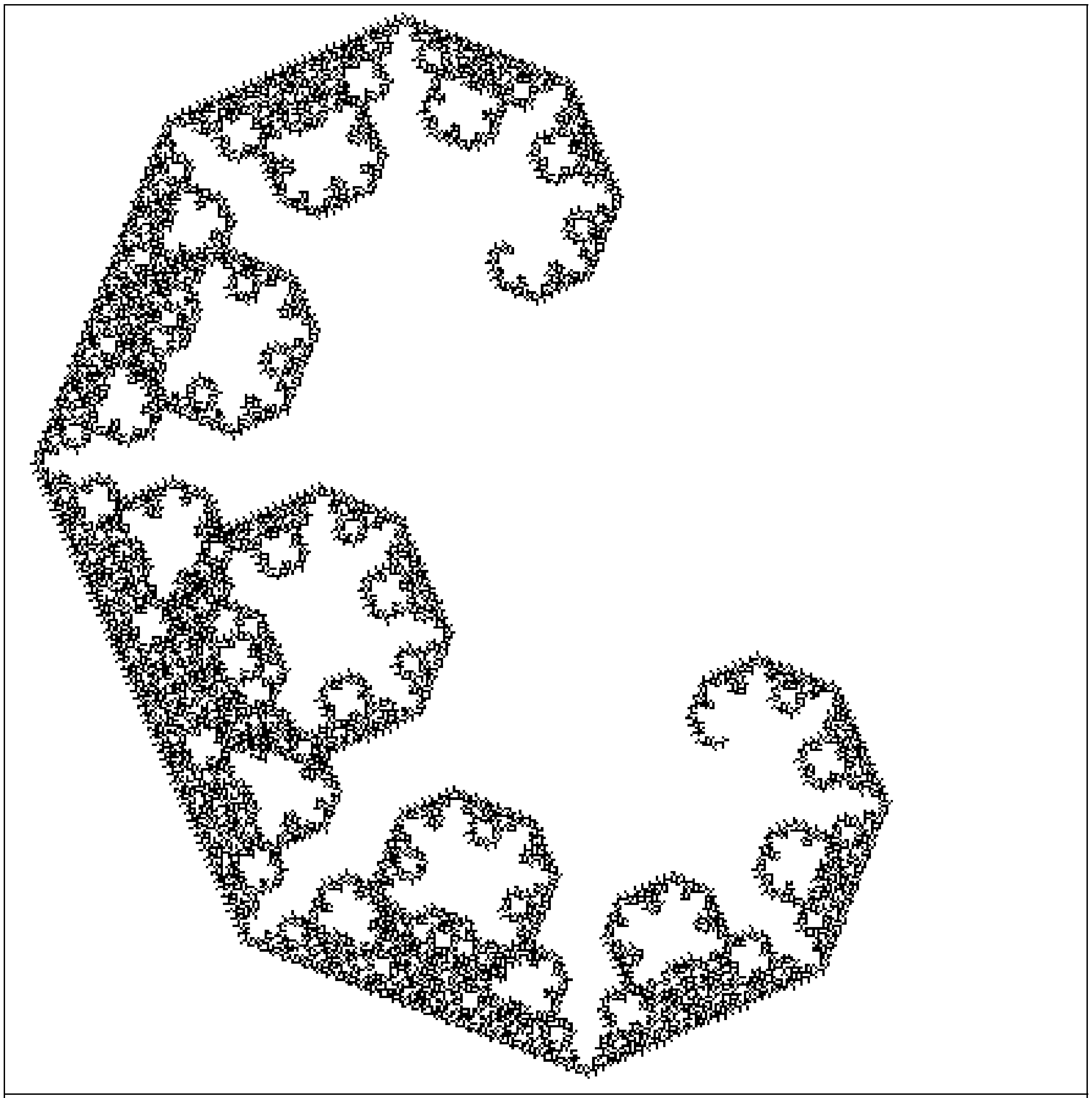


Figure 15. Shape generated using the L-System defined as V = {1, 0, [,]}, ω = 0, δ = 30°, P = { 0 → [1 ((1))] (1 (1 0) [[0]]), 1 → [1], [→ [,] →] }

Using the same IDL code, but with a new L-system generation, the shape of a tree is generated as shown in Figure 16.

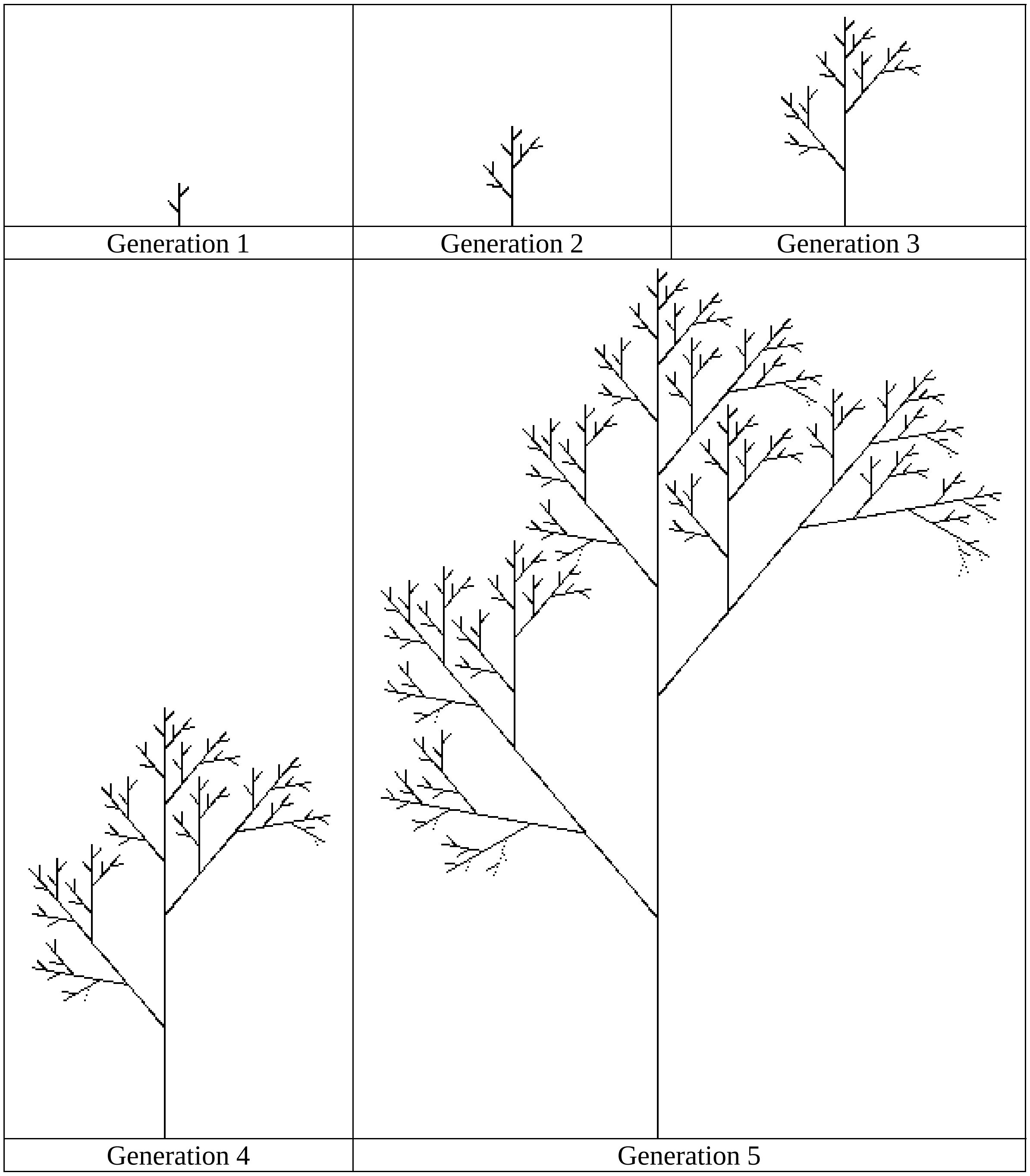


Figure 16. Five generations of tree growth using the L-system model defined as $V = \{1, 0, [,]\}$, $\omega = 0$, $\delta = 40°$, $P = \{ 0 \rightarrow 1 [0] 1 (0) 0, 1 \rightarrow 11, [\rightarrow [,] \rightarrow] \}$

In the example shown in Figure 17, each branch segment is assigned a width based on the age of the branch. The branching angle and length of segments can also be adjusted with age to represent a biologically realistic shape.

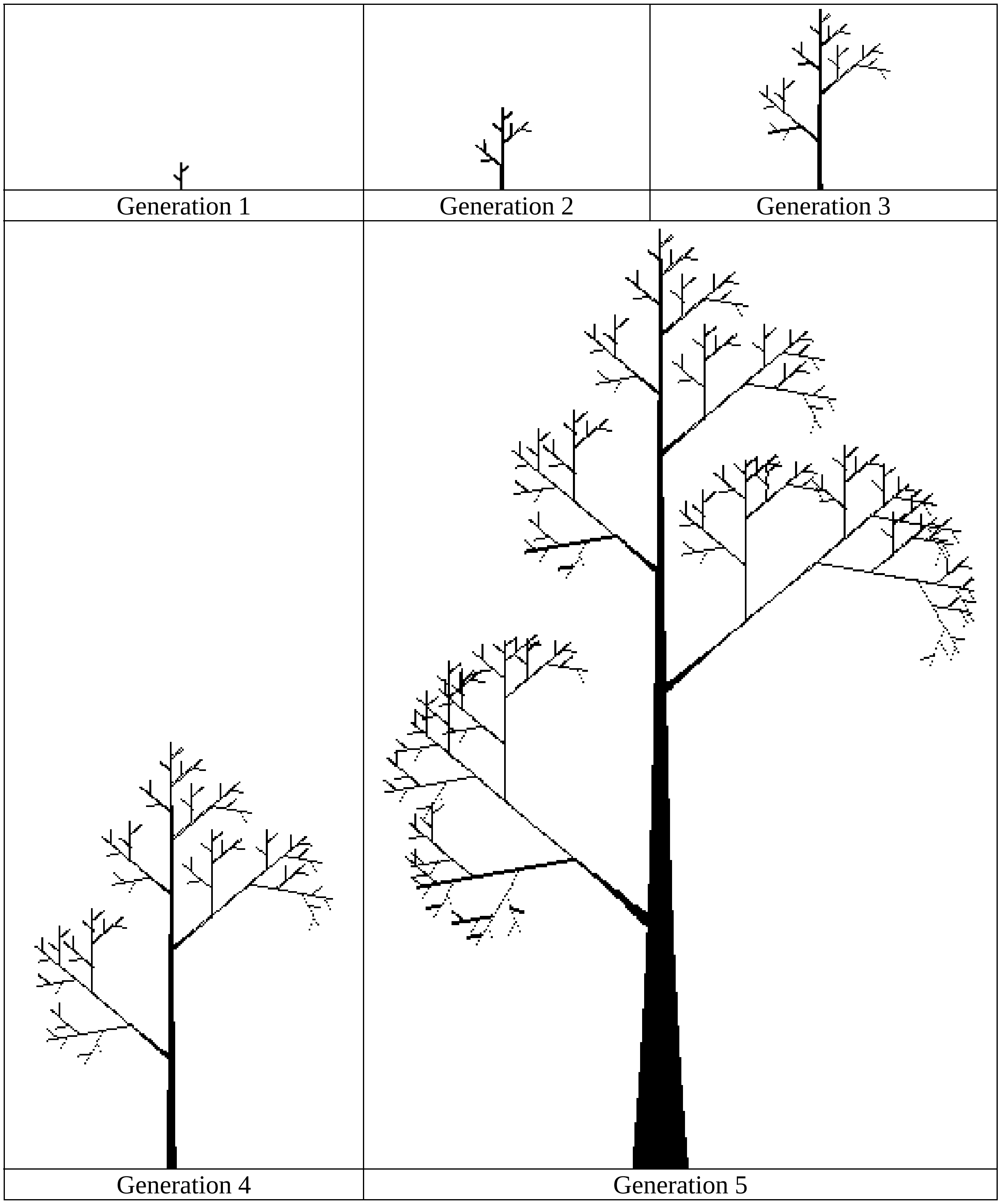


Figure 17. L-system tree with width associated with branch segments

These examples demonstrate some of the capabilities of modeling using L-systems. The deterministic L-systems are fairly easy to create and understand, and, because they are deterministic, the results are repeatable.

3. PROSPECT Leaf Reflectance Model

The PROSPECT leaf reflectance model was designed by Jacquemoud and Baret (1990) and is in widespread use throughout the remote sensing community. The model accurately simulates the reflectance and transmittance of plant leaves over the spectrum from 400 to 2500 nm. Several versions of the software have been developed and validated (publicly available at http://teledetection.ipgp.jussieu.fr/opticleaf/models.htm). An IDL version of the program, Version 2.01, of the PROSPECT model was modified by the author; the GUI interfaces were removed to allow incorporation in the LIDAR simulation program. The IDL code for the modified PROSPECT model is included in Appendix B.

From http://teledetection.ipgp.jussieu.fr/opticleaf/models.htm, “Version 2.01 of the model has five input parameters: leaf structure parameter (N), chlorophylls a+b concentration (Cab in µg/cm2), equivalent water thickness (Cw in cm), protein content (Cp in g/cm2), and cellulose + lignin content (Cc in g/cm2). IDL code (20 March 2001, Author: Glenn J. Newnham).”

An example of the reflectance and transmittance spectra as simulated by the PROSPECT model is given in Figure 18.

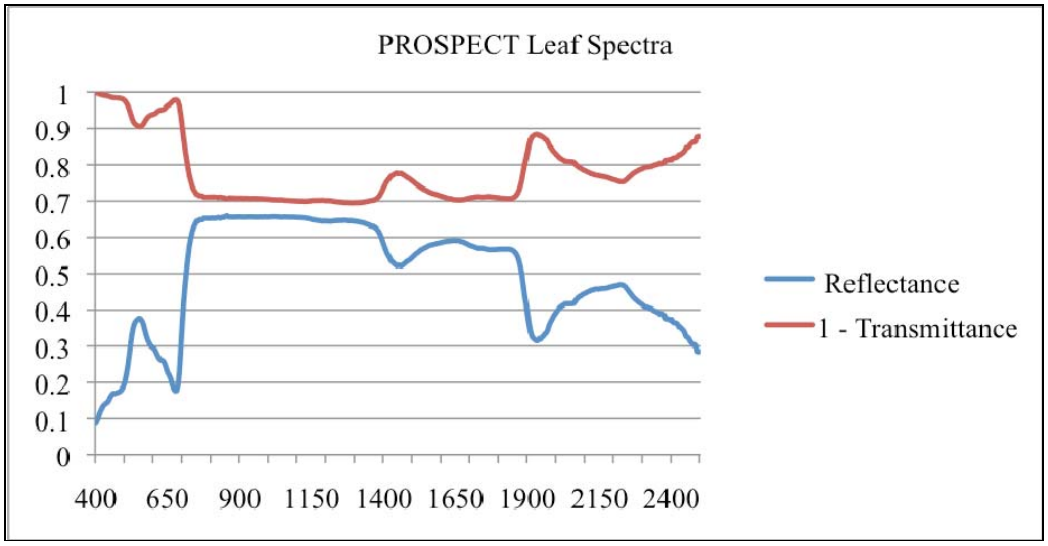


Figure 18. PROSPECT leaf reflectance model output. (N = 3.4, Cab = 20.0, Cw = 0.002, Cp = 0.001, Cc = 0.001, deg = 20)

The publicly-available USGS vegetation library contains spectra for dry grass, lawn grass, and maple leaf. Visually, the reflectance spectrum from the PROSPECT model is very similar to the USGS spectra.

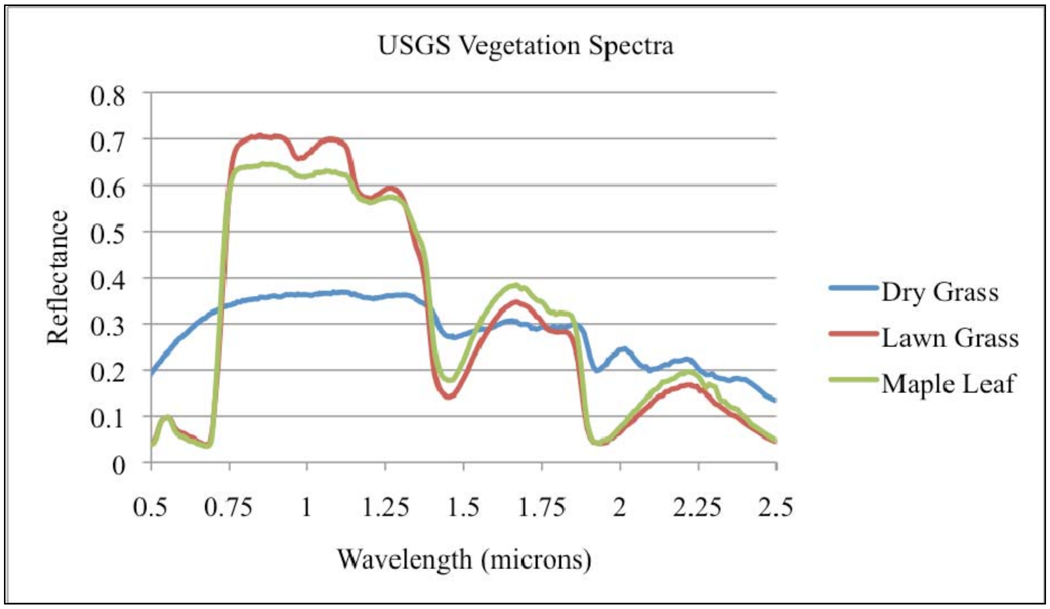


Figure 19. USGS Vegetation Spectra as included in the ENVI 4.5 software

The effect of the various input parameters is demonstrated in the following series of Figures 20 through 25.

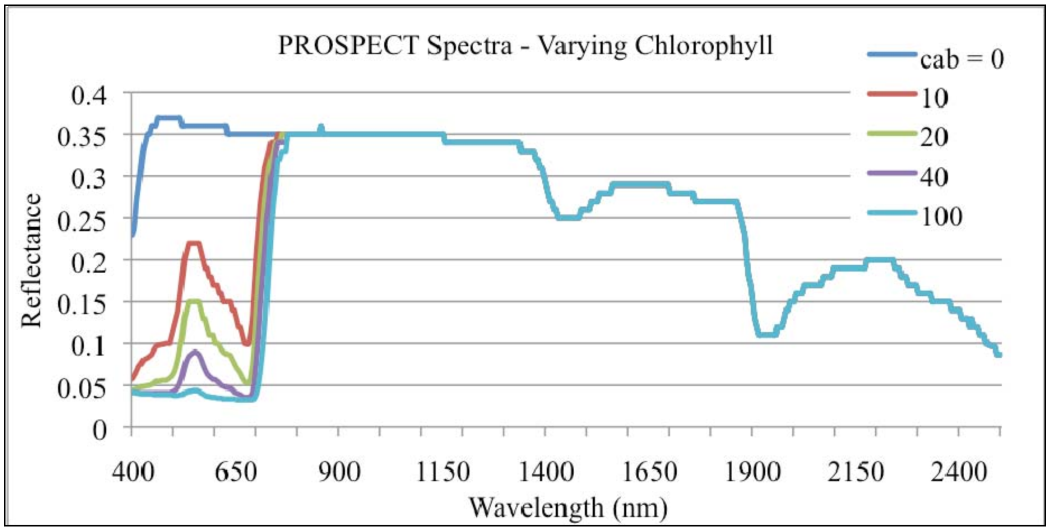


Figure 20. PROSPECT model output. (Cab = varying from 0 to 100, N = 3.4, Cw = 0.002, Cp = 0.001, Cc = 0.001, deg = 20)

The amount of chlorophyll affects the portion of the spectrum from 400 to 700 nm.

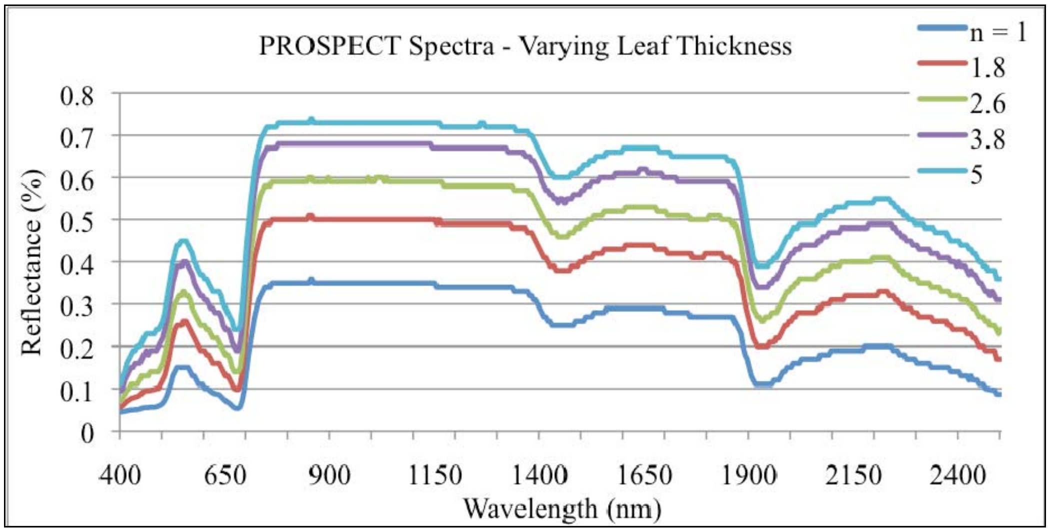


Figure 21. PROSPECT model output. (N = varying from 1 to 5, Cab = 20, Cw = 0.002, Cp = 0.001, Cc = 0.001, deg = 20)

Varying the leaf thickness has the greatest overall impact on the leaf reflectance, as compared to the other PROSPECT model parameters.

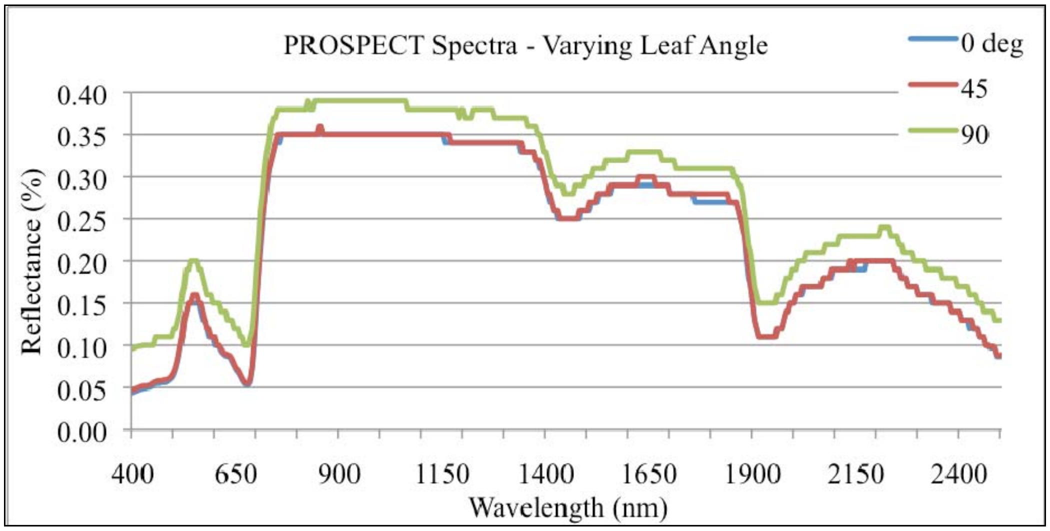


Figure 22. PROSPECT model output. (deg varying from 0 to 90, N = 3.4, Cab = 20, Cw = 0.002, Cp = 0.001, Cc = 0.001)

While the leaf angle does not have a very significant impact on the reflectance and transmittance of the leaves, this parameter is important within the LIDAR simulation as it impacts the direction of reflection.

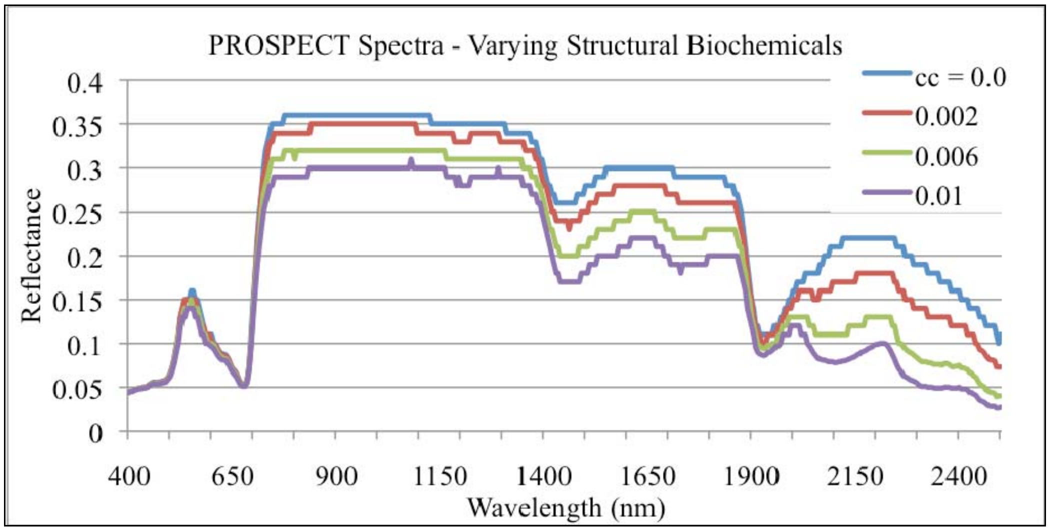


Figure 23. PROSPECT model output. (Cc varying from 0.0 to 0.1, N = 3.4, Cab = 20, Cw = 0.002, Cp = 0.001, deg = 20)

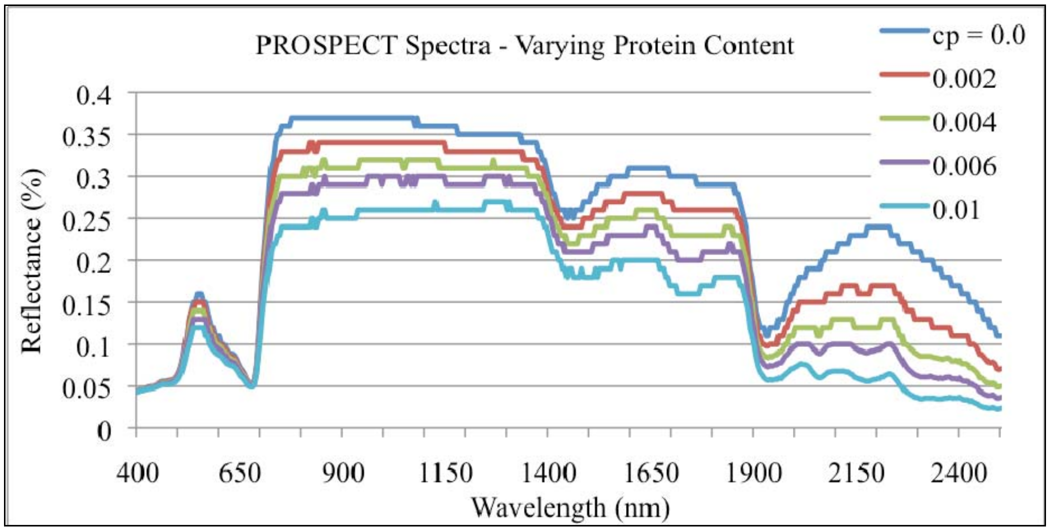


Figure 24. PROSPECT model output. (Cp varying from 0.0 to 0.1, N = 3.4, Cab = 20, Cw = 0.002, Cc = 0.001, deg = 20)

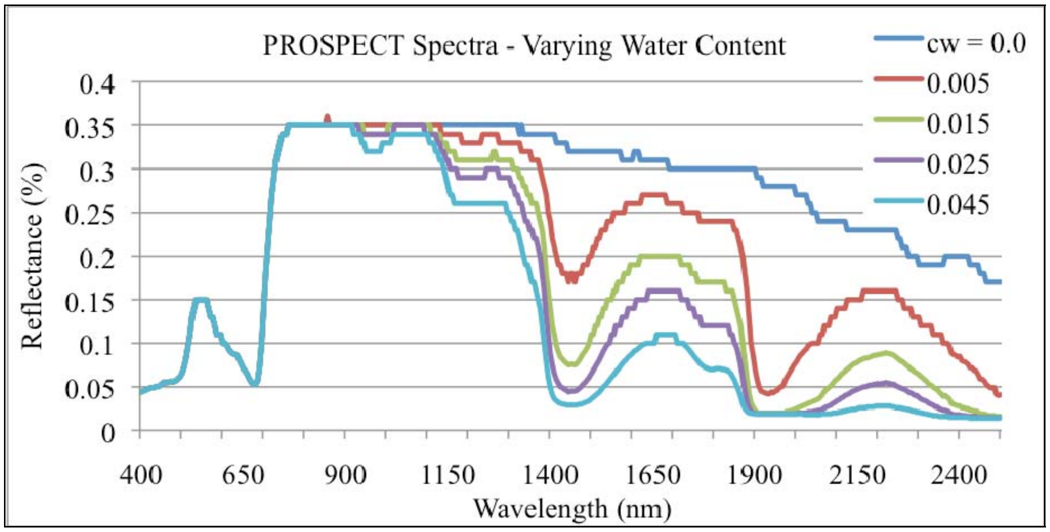


Figure 25. PROSPECT model output. (Cw varying from 0.0 to 0.05, N = 3.4, Cab = 20, Cp = 0.001, Cc = 0.001, deg = 20)

THIS PAGE INTENTIONALLY LEFT BLANK

III. RESULTS AND ANALYSIS

A. THE MODEL PARAMETERS

Each of the models is coded in the Interactive Data Language (IDL) from ITT Visual Information Solutions.

The following section contains the default settings used to test the models. Each of these settings can be adjusted to simulate different collection parameters.

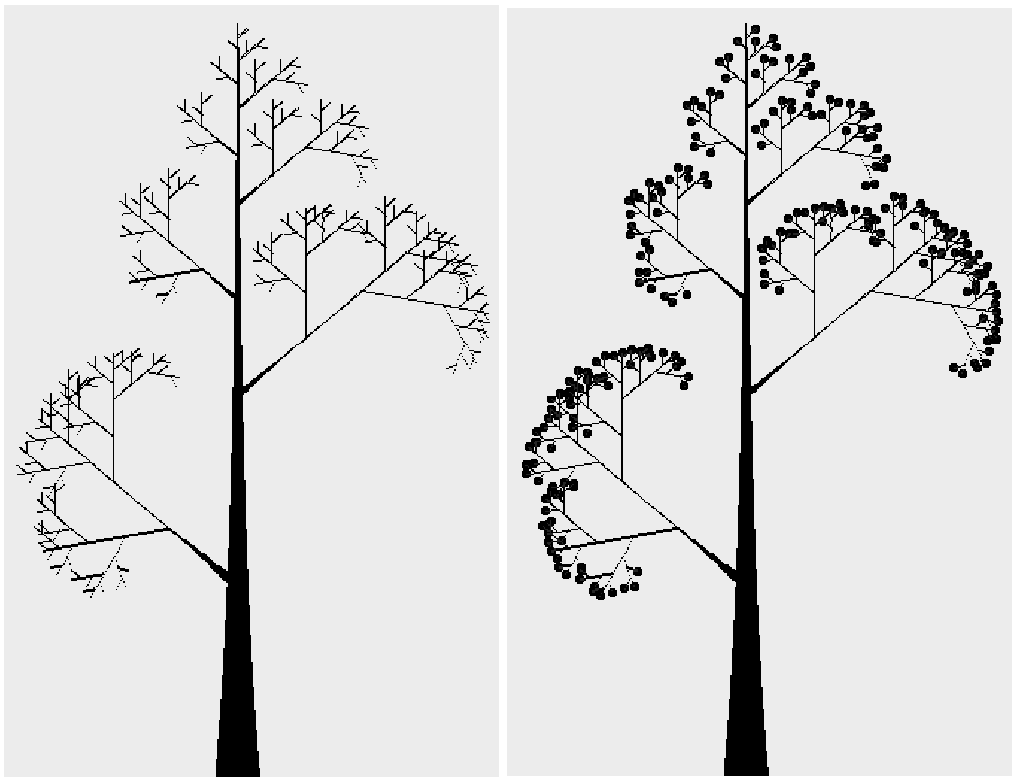


Figure 26. L-Sytem tree model output

1. L-system Tree Model Parameters

L-System Tree Model Settings	
Number of generations	5
Branch angle ‘δ’	50º
Original segment length, width	8 units, 30 units
Percent decrease length, width (per step)	2%, 4%
Axiom “ω”	0
Production rules “P”	0 -> 1 [0] 1 (0) 0 1 -> 1 1 [], () -> [], ()
Leaf clump radius	2 units

Table 1. Parameters and settings used to create the L-System tree

The “units” of measurement are given a physical meaning when the overall height of the tree is defined within the LIDAR simulation program.

The L-System code generates a series of files, which are used as inputs to the LIDAR propagation model.

L-System Tree Model Outputs	
Air.dat	Probability of air at each pixel location
Bark.dat	Probability of bark at each pixel location
Leaf.dat	Probability of leaves at each pixel location
(Generation #).gif	Image file showing the tree at generation n
Cos_map.dat	Cosine of angle of branch, measured counter-clockwise from horizontal
Bud_xcoord.dat, bud_ycoord.dat	x and y pixel locations of buds – where leaves may grow
L_system_string.txt	Definition of the L-System, and string created by the model

Table 2. Output files from the L-System tree program

2. LIDAR Propagation Model Parameters

The following settings were used as default values for LIDAR simulation test cases.

Tree Parameters	
Tree height – base of trunk to tree top	15 m
Tree trunk width at base	0.89 m
Smallest branches modeled	0.03 m
Leaf clump diameter	0.06 m

Table 3. Physical units of L-System tree dimensions used in the LIDAR propagation model

Radiometric Properties	
Atmosphere	15 km visibility – clear
Bark transmittance, reflectance, absorbance	532 nm – 0%, 10%, 90% 1064 nm – 0%, 35%, 65% 1550 nm – 0%, 45%, 55%
Leaf transmittance, reflectance, absorbance	532 nm – 8%, 36%, 56% 1064 nm – 30%, 36%, 5% 1550 nm – 27%, 57%, 16%
Ground (green grass) transmittance, reflectance, absorbance	532 nm – 0%, 7%, 93% 1064 nm – 0%, 40%, 60% 1550 nm – 0%, 23%, 67%

Table 4. Radiometric properties of materials in the LIDAR propagation model

LIDAR System Parameters	
Flying altitude above ground	1000 m
Pulse width	6 ns
Beam divergence	0.3 mrad
Aperture of receiving optics	0.2 m
Laser pointing angle (degrees from nadir)	0º

Table 5. LIDAR system parameters and settings used in the LIDAR propagation model

The outputs from the LIDAR propagation model are as follows:

LIDAR Propagation Model Outputs	
Initial_pulse.txt	Energy per bin of transmitted LIDAR pulse
Sampled_initial_pulse.txt	Transmitted LIDAR pulse resampled at the specified LIDAR system sampling rate
All_pulses.txt	List of energy and time-of-flight for each LIDAR pulse element
Waveform.txt	Simulated waveform; energy vs. time-of-flight – average response of all runs
Sampled_wf.txt	Simulated waveform resampled at the specified LIDAR system sampling rate
Metadata.txt	Text file containing all the parameter settings
Hit_map.gif	Image file showing the path traced by each LIDAR pulse element throughout the workspace

Table 6. Output files from the LIDAR propagation model

3. PROSPECT v2.01 Leaf Reflectance Model Parameters

PROSPECT v2.01 Parameters	
Leaf equivalent thickness (n)	3.4
Chlorophyll a+b (cab)	20.0
Water (cw)	0.002
Protein (cp)	0.001
Structural biochemicals (cc)	0.001
Leaf angle (deg)	0º

Table 7. Parameters and settings used to simulate the leaf reflectance spectra using PROSPECT v2.01

PROSPECT v2.01 Outputs	
PROSPECT_output.csv	Comma Separated Values file containing PROSPECT parameter settings, and leaf reflectance and transmittance values vs. wavelength

Table 8. Output file from the PROSPECT v2.01 leaf reflectance model

B. THE MODEL OPERATION

The L-System tree is defined and rendered, and the output files are created, including a map of bark and leaf locations and a map of branch angles. These files are used as inputs to the LIDAR simulation program.

The LIDAR simulation uses the information in the tree files to define the geometric layout of the model space. The height of the tree is a user-defined quantity, and this value is used to determine the physical dimensions of model space. For example, the simulated tree used in the simulations (shown in Figure 26) is 504 pixels tall. The user-defined height of the tree is 15 m, and so each pixel corresponds to 2.976 cm.

The radiometric properties of the ground and bark are user-defined values, while the PROSPECT model is called to determine the radiometric properties of the leaves. The PROSPECT model depends on physical properties of the leaf, as explained in Section A, 3. "PROSPECT v2.01 Leaf Reflectance Model." The atmospheric absorption for each location in the image space is calculated according to Beer's Law.

The probability of each of the materials in the scene can be adjusted. For example, if a pixel contains a leaf, and the probability of the leaf map is set to 50%, there is a 50% chance that the pixel will be treated as a leaf, and a 50% chance that the pixel will be treated as air.

To create the simulated waveform, multiple LIDAR pulses are propagated through the system, and the average response is output as the simulated waveform. Each LIDAR pulse is defined as having a certain length, energy level, and beam divergence. The pulse is divided into multiple small bins, with each bin being treated as a separate pulse having an initial energy, location, and propagation direction determined by its location in the overall LIDAR pulse. The initial start time is also adjusted based on the bin location; for a 6 ns pulse, the initial bin pulse will be transmitted at time 0, while the last bin will be transmitted just prior to 6 ns.

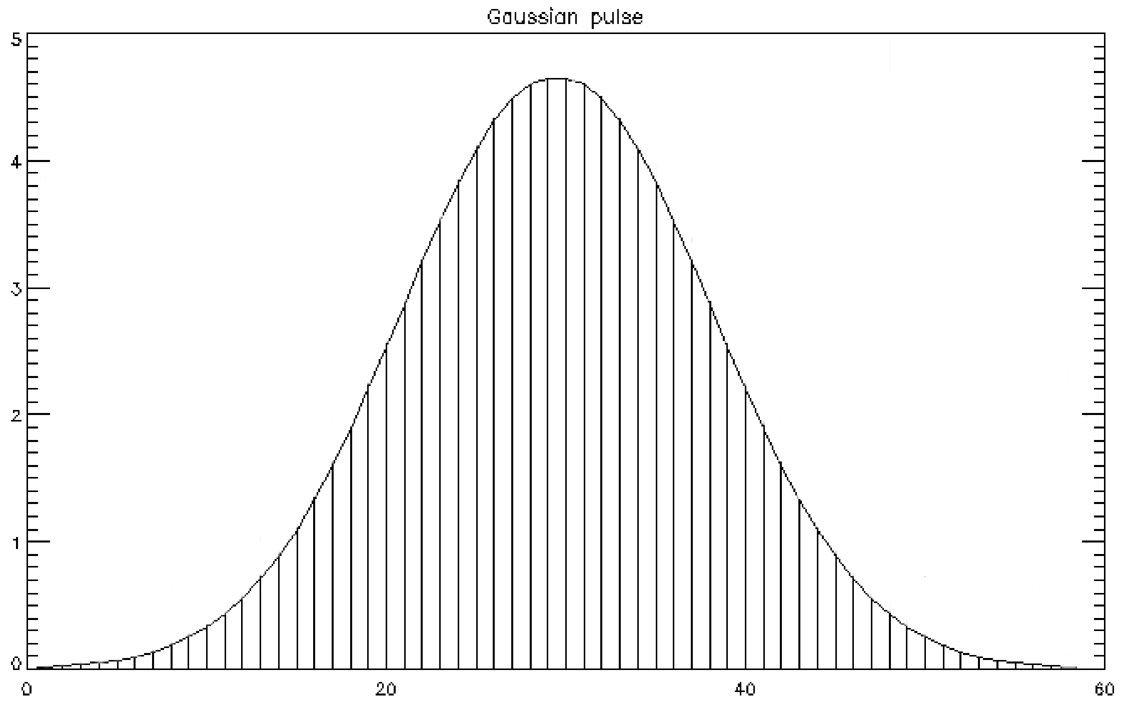


Figure 27. An example of a transmitted LIDAR pulse, showing multiple small bins dividing the initial pulse. Each bin is treated as a separate pulse having an energy equal to its area.

Each pulse bin is propagated with a user-defined step size. At each step, the location of the LIDAR pulse is tracked, and the material interaction is determined probabilistically according to the material probability maps, and reflectance properties. When the pulse interacts with material having an absorbance greater than 0, the energy of the pulse is reduced by that amount. The remaining energy is allowed to continue propagating. If the energy is transmitted, the angle of propagation is not changed. If the energy is reflected, the angle of propagation is updated. The reflection can be either Lambertian or specular. If the reflection is Lambertian, every angle of reflection between 0 and π radians from the material surface is equally likely, so a random angle of reflection is chosen. In the specular case, the reflection is assumed to be a mirrored reflection from the plane of the material being impacted. Figure 28 contains an illustration of the different types of scattering interactions.

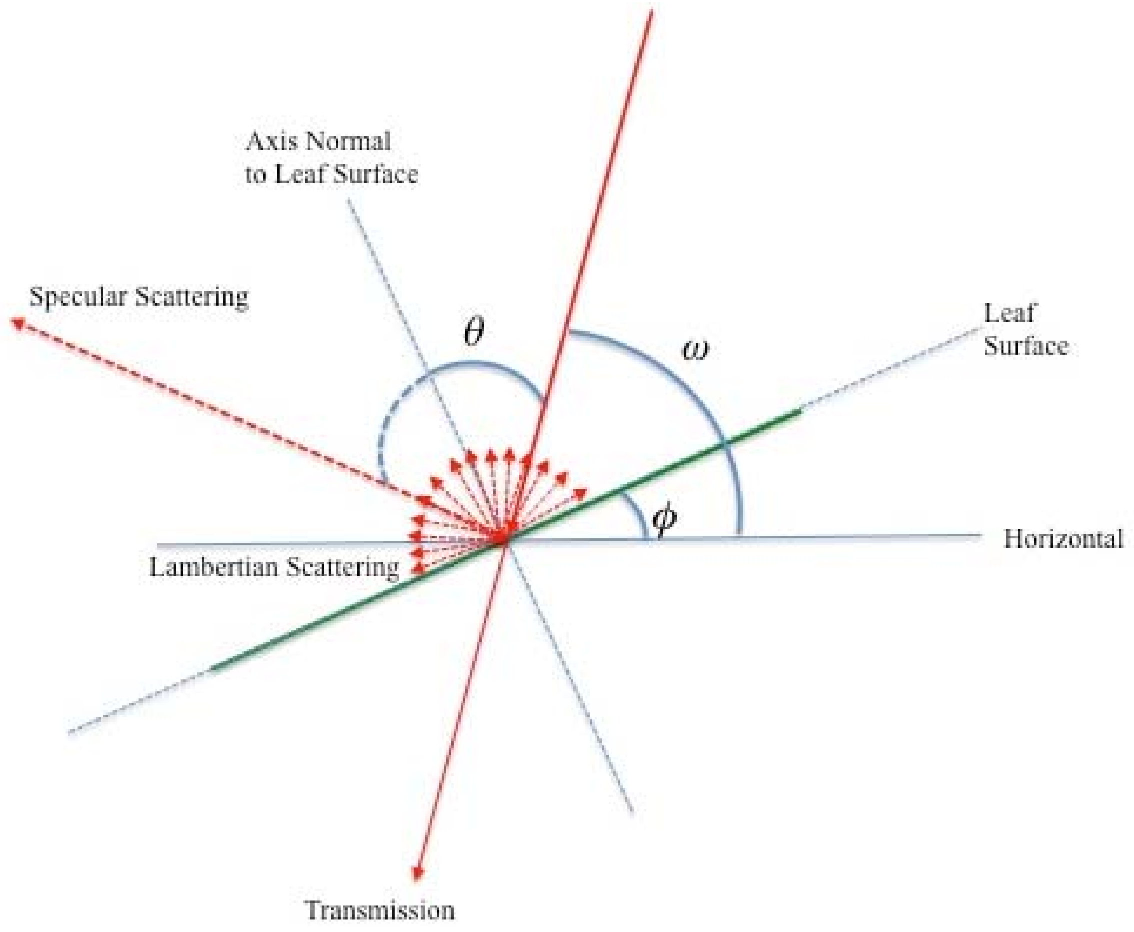


Figure 28. Illustration of the interactions of light via transmission, or specular or Lambertian scattering from a leaf surface

In Figure 28, ϕ is the angle of the leaf surface, measured counterclockwise from the horizontal plane. The angle of incoming LIDAR radiation, ω, is also measured counterclockwise from the horizontal plane. The axis normal to the leaf surface is $\frac{\pi}{2}+\phi$ radians counterclockwise from the horizontal plane, and is the axis about which energy is reflected. The angle between the incoming LIDAR radiation and the normal axis is $\theta=\frac{\pi}{2}-(\phi-\omega)$ = $\frac{\pi}{2}-\phi+\omega$. Therefore, the angle of reflection is $\left(\frac{\pi}{2}+\phi\right)-\theta=\left(\frac{\pi}{2}+\phi\right)-\left(\frac{\pi}{2}-\phi+\omega\right)=2\phi-\omega$.

Each pulse bin of energy is tracked until the energy is returned to the sensor, scattered out of the workspace and lost, or absorbed by materials within the scene. Some of the pulse bins may have an initial energy level below the detection level of the LIDAR system—this energy is considered lost. There is also an option to set a maximum number of allowed scattering events; if a pulse interacts more times than allowed, the energy is considered lost. If the energy is returned to the sensor, the time of return and the amount of energy is recorded. This process is repeated for each pulse bin. Returns occurring at exactly the same time are averaged, and a waveform is created for each pulse. This process is repeated for multiple pulses, and the waveform from each pulse is averaged to create the simulated waveform. The waveform is resampled so that the total energy contained in the waveform matches the amount of energy returned to the sensor.

C. THE SIMULATED COLLECTION

Based on the speed of light being 299,792,458 m/s, a LIDAR pulse transmitted from a sensor 1000 m above the ground will take approximately 6671 ns to traverse from the sensor, to the ground, and back. Likewise, for a tree top 15 m above ground, the LIDAR pulse will take 6571 ns to reach the top of the tree and be reflected back to the sensor. Figure 29 shows the simulated tree, and expected time-of-flight values for a LIDAR system located 1000 m above ground level.

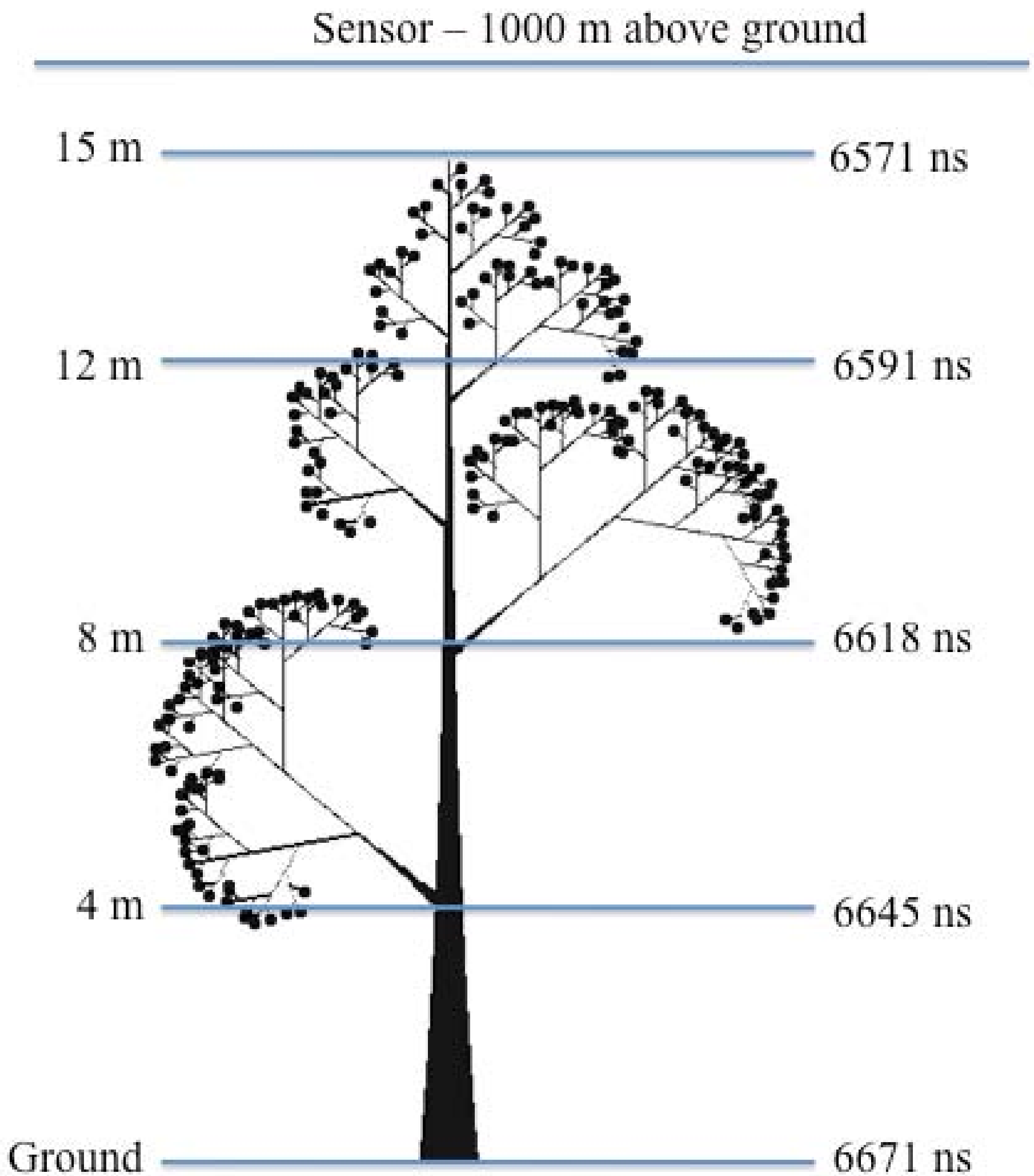


Figure 29. The simulated tree, with approximate heights and pulse time-of-flight values for a LIDAR sensor located 1000 m above the ground

D. THE SIMULATED LIDAR WAVEFORMS

In all of the test cases, the transmitted LIDAR pulse was defined as having the Gaussian shape as shown below. The probability distribution function for a Gaussian or normal distribution is defined as $\frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$,

where the variance σ = 14 and the mean μ = 0.

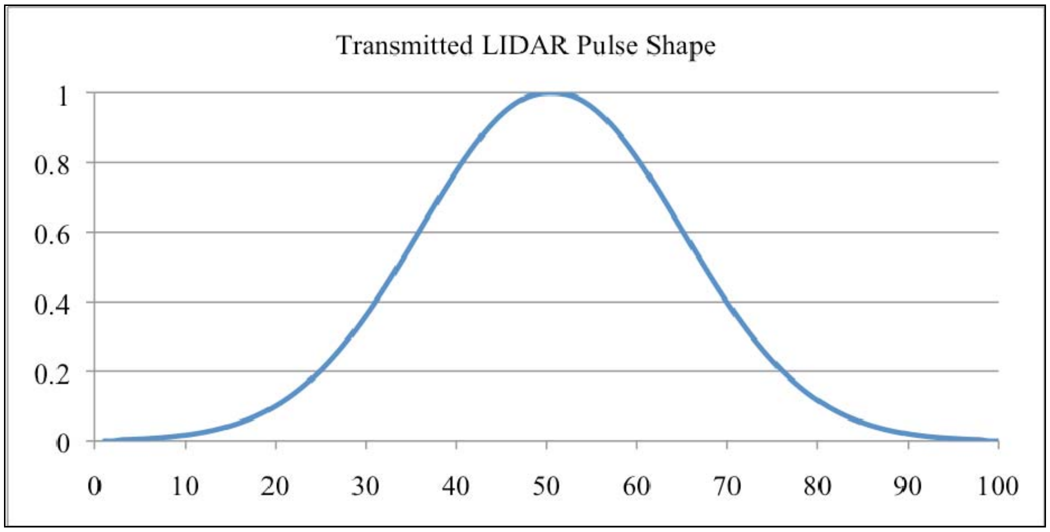


Figure 30. Gaussian curve shape used to model the transmitted LIDAR pulse

The total energy and length of the pulse are variable within the simulation. The length of the pulse is varied by interpolating the Gaussian curve shape into as many points as are needed. For example, if the model has a time step resolution of 0.5 ns, and the transmitted pulse is assumed to be 6 ns long, and the initial energy is assumed to be 100%, the initial LIDAR pulse will be as shown in Figure 31.

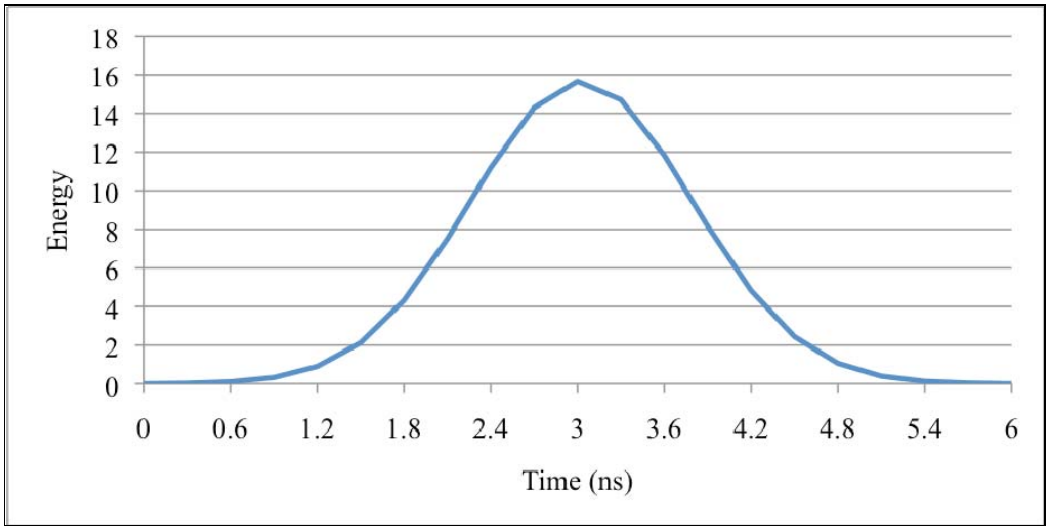


Figure 31. Simulated transmitted LIDAR pulse with a pulse length of 6 ns, an initial energy of 100%, and a simulation temporal sampling of 0.3 ns per step

1. Test Case 1–Interaction with a Ground Surface

In this case, the LIDAR pulse was directed directly at the ground, with no obstructions between the sensor and the ground. The reflectance of the ground was assumed to be 70%, and the absorbance was set at 30%. The purpose of this test was to ensure the transmitted Gaussian pulse shape was returned without any modification, the correct amount of energy was returned to the system, and to check the timing of the system.

Because the altitude of the sensor above the ground is set at 1000 m, the time taken for a pulse of light to travel from the sensor to the ground and back is approximately 6671 ns. In each of the test cases, the height above ground of returned points is calculated assuming that the ground is detected at 6671 ns. Pulses that take less than 6671 ns to travel back and forth must be returned from locations above ground. Likewise, pulses that take longer than 6671 ns appear to be below ground level. No adjustment has been made for the length of the pulse, so energy returned from the trailing edge of the transmitted Gaussian pulse will appear to take longer than 6671 ns, because time-of-flight is measured from the time of transmission of the leading edge of the pulse.

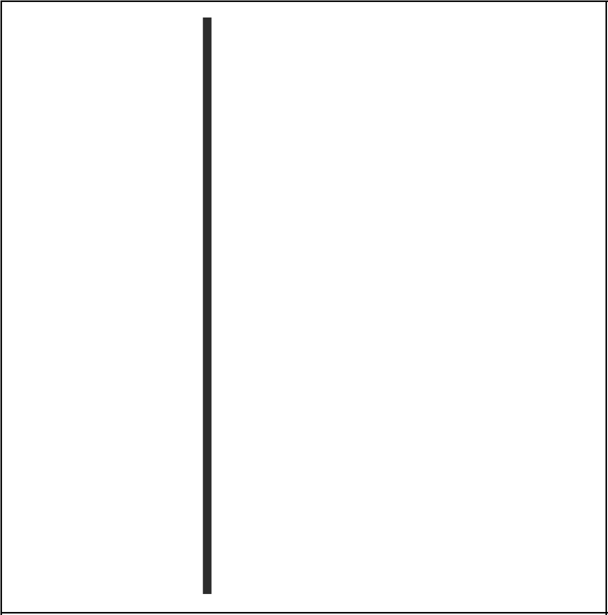
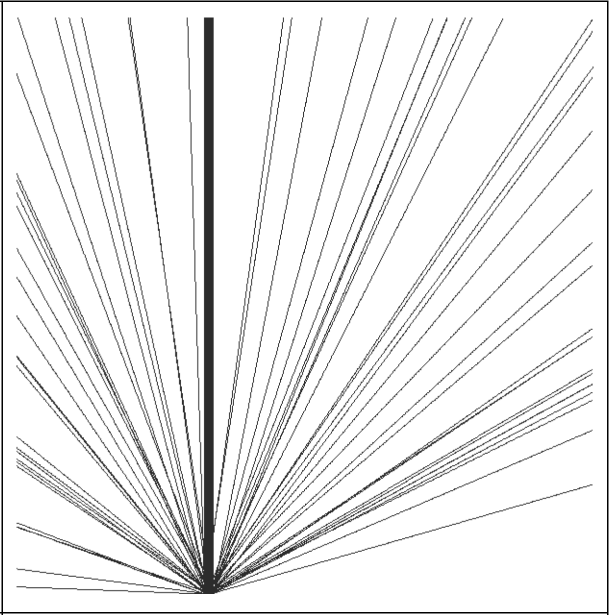
	
Specular Surface	Lambertian Surface

Figure 32. Interaction of first pulse of LIDAR energy, with the ground modeled as being a Lambertian surface and as a specular surface

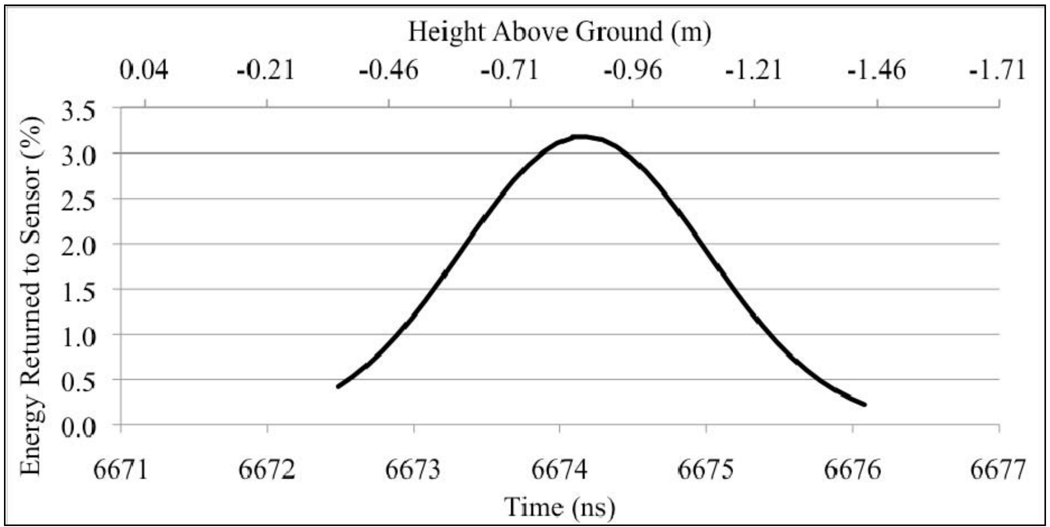


Figure 33. The simulated waveform returned from a specular surface

Of the total energy transmitted, with the ground being modeled as a specular surface, 68.46% of the energy is returned to the sensor, 1.54% is scattered out of the workspace, and 30% of the energy is absorbed. The 1.53% energy lost to scattering is due to the 0.3 mrad angular beam spread, and the fact that the receiving optics aperture is only 0.2 m in diameter.

The first energy to return detected at the sensor occurs at approximately 6672 ns after transmission, while the first energy returns are expected at 6671 ns. This difference is explained by the 1.53% energy that is lost due to scattering. The model is designed such that the transmitted LIDAR pulse is dispersed over the required angular beam spread. The initial and final bins of energy that are transmitted will have the greatest variance from the initial beam direction and, after a specular reflection, will end up furthest from the point of transmission.

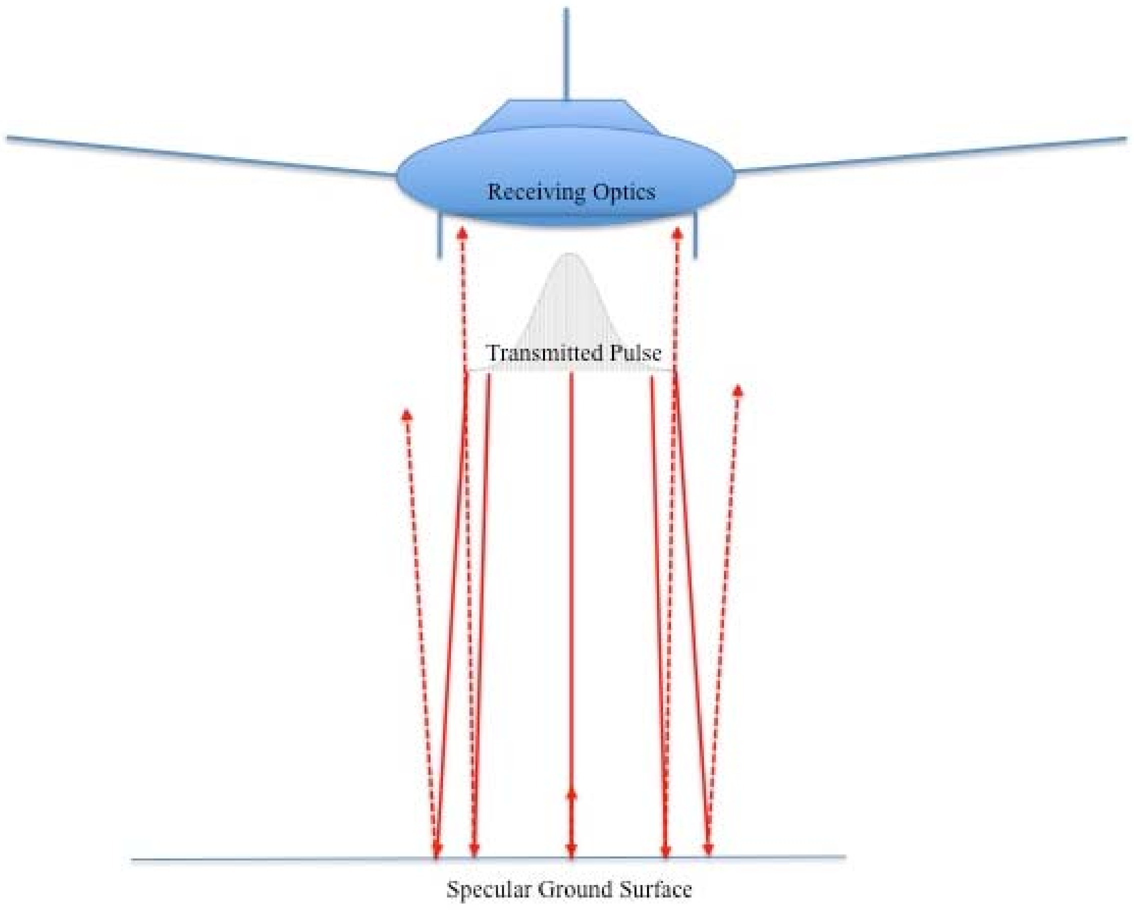


Figure 34. Illustration of energy being scattered away from the sensor due to the angular beam spread of the LIDAR system

The transmitted LIDAR pulse in this example is 6 ns long, and the last energy is returned to the sensor just less than 6 ns after the initial energy. The difference between the final return time and the expected value of 6677 ns can again be explained by the scattering of the final energy bins outside the diameter of the receiving optics of the sensor. The peak of the transmitted LIDAR pulse occurs as expected at approximately 6674 ns—3 ns after the expected start of returned waveform.

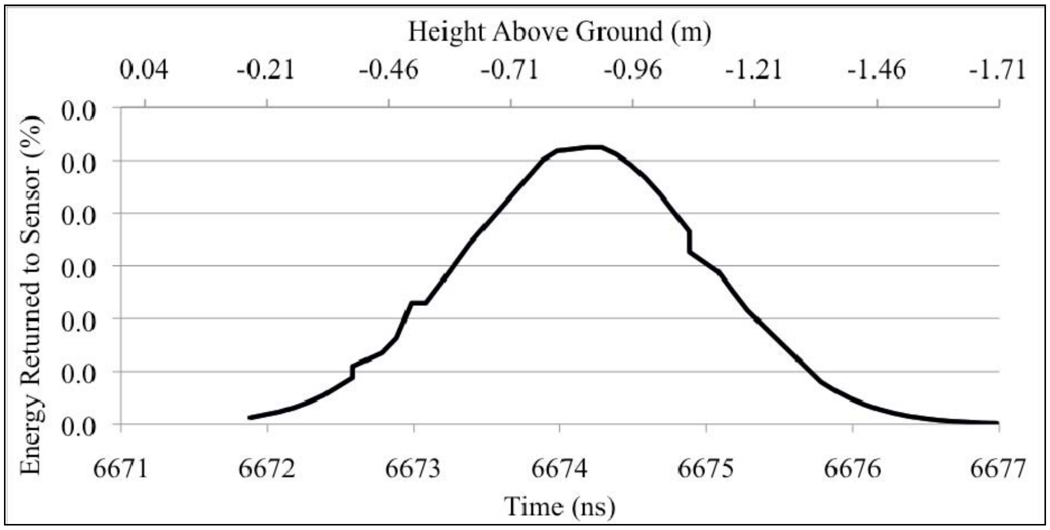


Figure 35. The simulated waveform from a Lambertian surface. The total energy returned to the sensor is 0.15% of the transmitted energy

When the ground is modeled as a Lambertian surface, light is scattered in all directions off the ground, regardless of angle of transmission. Therefore, the initial pulse bins are not necessarily scattered away from the sensor, and the first energy returned to the sensor occurs at 6671 ns, as expected. The last energy is recorded at 6677 ns.

Of the total energy transmitted, only 0.15% of the energy is returned to the sensor, while 30% is absorbed by the ground, and 69.85% is scattered out of the space. The overall Gaussian shape of the transmitted pulse is maintained, but the amount of energy returned is significantly decreased.

A second test was performed with the LIDAR interacting with a sloped ground surface. The slope of the ground is approximately 11º.

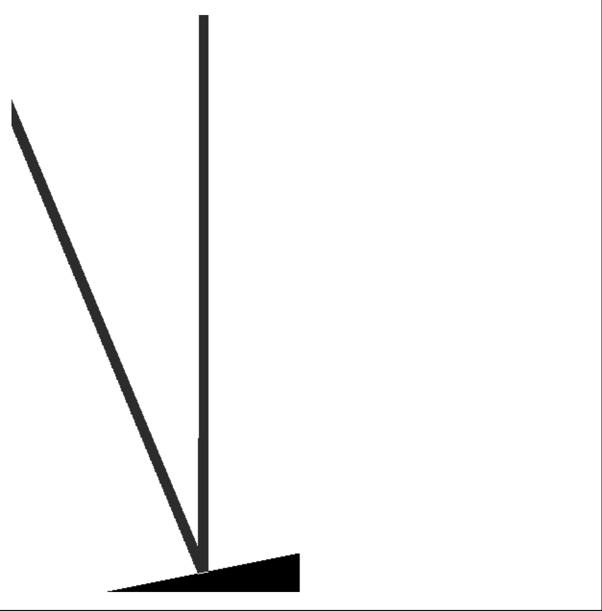
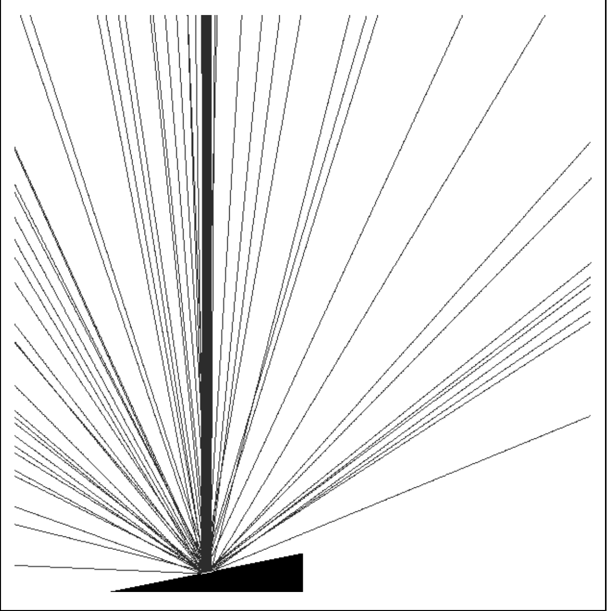
	
Specular Surface	Lambertian Surface

Figure 36. Interaction of first pulse of LIDAR energy

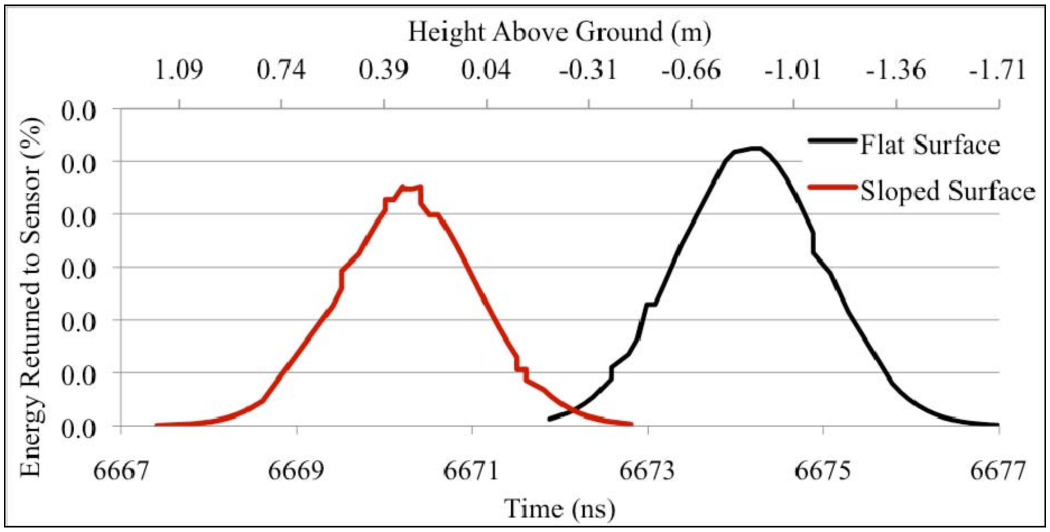


Figure 37. The simulated waveforms from a sloped and flat Lambertian surface

When the ground is modeled as a specular surface, all of the energy is reflected out of the space. When the ground is modeled as a sloped Lambertian surface, the initial peak of the waveform occurs at the correct time (about 4 ns earlier than the peak from the flat surface due to the slope surface being above ground level—about 60 cm at the center of the slope).

The total energy returned to the sensor from a sloped Lambertian surface is 0.19% of the total initial energy transmitted, as compared to 0.15% of the energy from the flat Lambertian surface. The curve of the waveform from the flat surface appears higher due to the return of fewer, higher intensity returns, but the total energy returned is actually lower from the flat Lambertian surface.

The return of more energy from a sloped Lambertian surface than from a flat surface could be due to secondary reflections of energy. Energy that is scattered off the sloped surface towards the flat ground interacts with the ground, and has a chance of being reflected back towards the sensor.

The LIDAR footprint in this case is not large compared to the slope of the ground, and the angle of the slope is relatively small, so there is not any noticeable spread in time of the returned waveform. It is expected that a larger footprint, or a more extreme slope, will be cause the returned waveform to be lengthened in time.

2. Test Case 2–Varying Number of Iterations

In this test, the effect of the number of model iterations was examined. The graphs of the returned waveforms become smoother as the number of iterations is increased (see Figure 38). The overall energy returned to the sensor stays approximately the same, but the number of distinct returns increases with more iterations (see Table 9).

Number of Iterations	Total Energy Returned to Sensor (%)	Number of Distinct Times of Return
100	0.094%	13
200	0.115%	23
300	0.149%	33
400	0.0921	29
500	0.113%	42

Table 9. Total energy returned to the sensor and numbers of distinct returns after interatction with a flat Lambertian surface for a varying number of model iterations

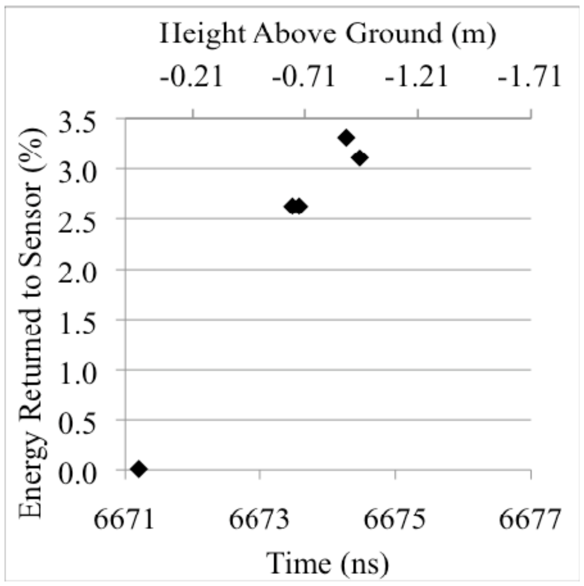
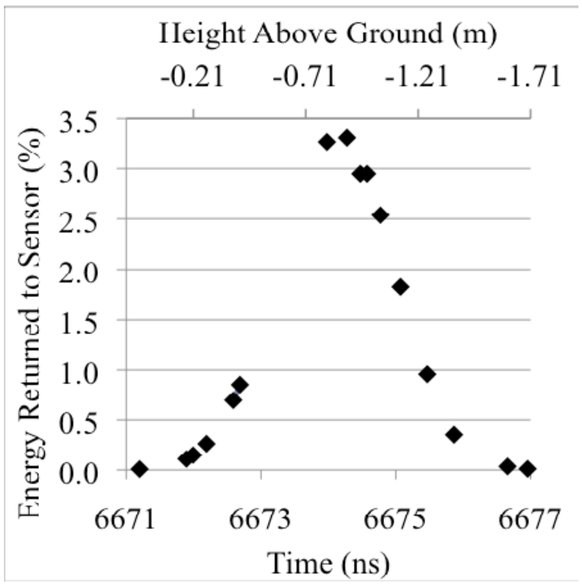
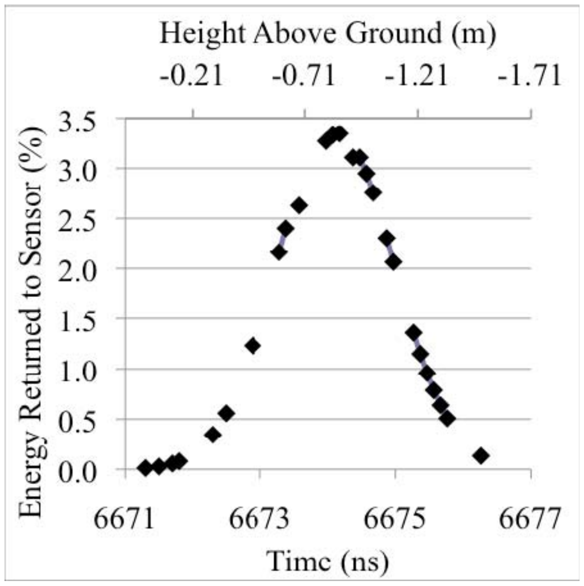
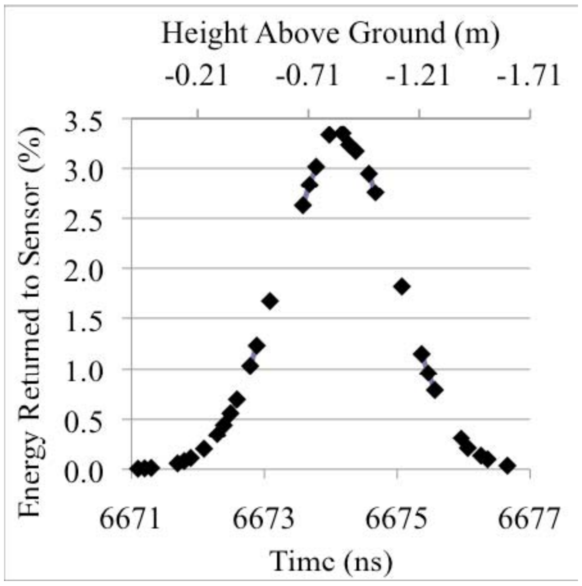
	
50 Iterations	100 Iterations
	
300 Iterations	500 Iterations

Figure 38. Simulated waveforms for LIDAR interacting with a flat Lambertian surface

The waveforms in Figure 38 were sampled to give a consistent step-size. This sampling was only done in this test case, and was used to demonstrate the effect of varying the number of model iterations. In other test cases, the returned waveform does not have a constant step-size; whenever a photon is returned to the sensor, the time-of-flight and amount of energy are recorded. The number of iterations used in the other test cases was fixed at 500 iterations, unless otherwise noted.

3. Test Case 3–Interaction of LIDAR with a Leaf

The purpose of this test case was to demonstrate the interaction of LIDAR with a leaf. A single pixel (2.98 cm) leaf "plate" was simulated approximately 89 cm above the bottom of the model space, with the sensor 1000 m above the bottom of the model space. In this first test, there is no ground surface. The energy that is transmitted through the leaf continues out of the bottom of the workspace and is not returned to the sensor.

The beam was not allowed to spread with transmission.

The time-of-flight to and from the LIDAR sensor to the leaf plate should be approximately $\frac{2*(1000m-0.89m)}{299,792,458m/s} = 6665\times10^{-9}s$.

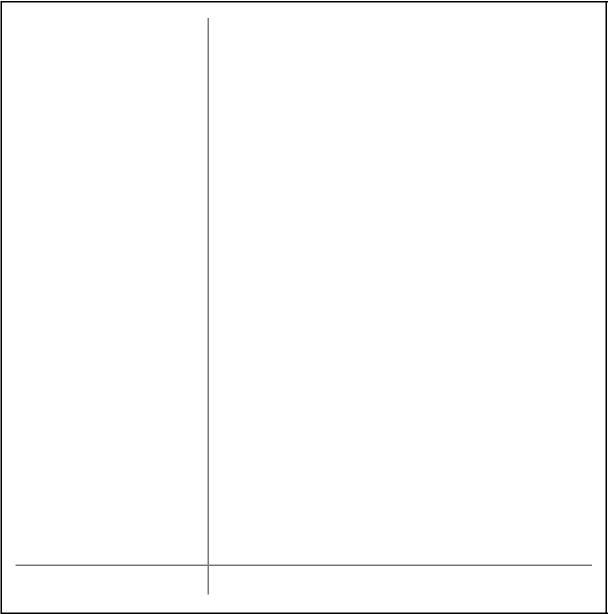
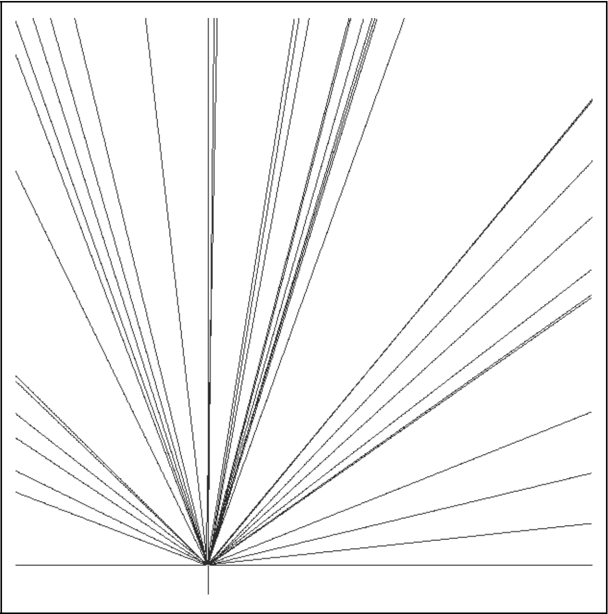
	
Specular Leaf (No ground surface)	Lambertian Leaf (No ground surface)

Figure 39. Interaction of first pulse of LIDAR energy with a simulated "leaf plate"

In the first case, the leaf is modeled as a specular surface, with a reflectance of 65.57%, transmittance of 29.83%, and absorbance of 4.6%. When the simulation is run, of the total energy transmitted, 65.72% is returned to the sensor, 29.68% is lost, either by being transmitted through the leaf or by being reflected back towards the sensor, but outside of the receiving optics aperture. 4.6% of the energy is absorbed.

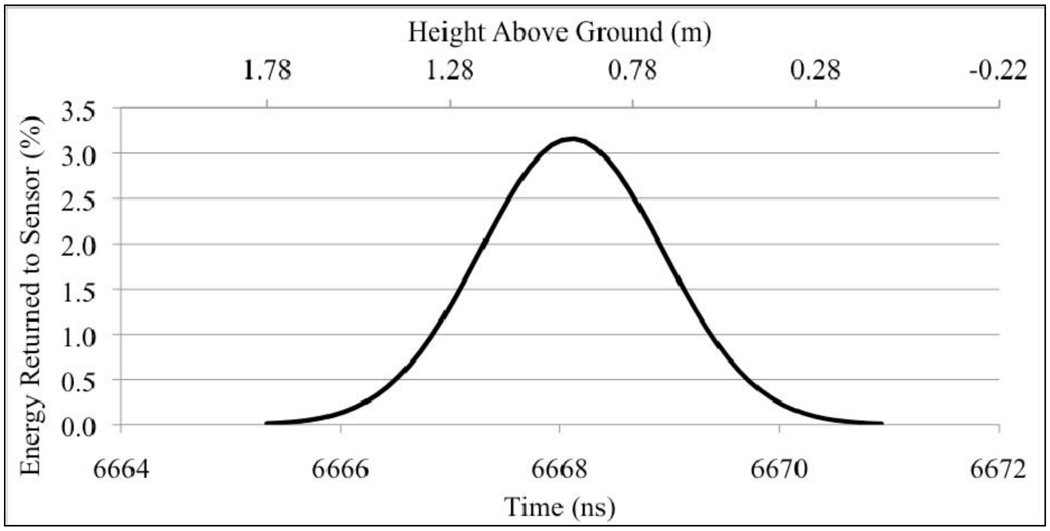


Figure 40. Waveform from LIDAR energy returned to the sensor after interaction with a simulated specular surface "leaf plate"

Similarly, the overall Gaussian shape of the transmitted LIDAR pulse is maintained when the surface is Lambertian, although the energy is returned to the sensor less consistently than in the specular case. Ninety-two percent of the initial pulse energy is scattered out of the space, and only 0.1% is returned to the sensor. The remaining 7.9% of the energy is absorbed by the leaf.

The amount of energy absorbed (7.9%) is higher than the modeled absorbance of the leaf (4.6%). This could be due to the fact, when the surface is modeled as a Lambertian surface, reflections are allowed to occur in any direction off the leaf's surface (between 0 and π radians as measured counter-clockwise from the plane of the leaf surface). Some of the reflected energy will be reflected at 0 and π radians, and this energy then has a second chance of being absorbed. The higher absorption is due to energy reflecting along the surface of the leaf, and being subsequently absorbed.

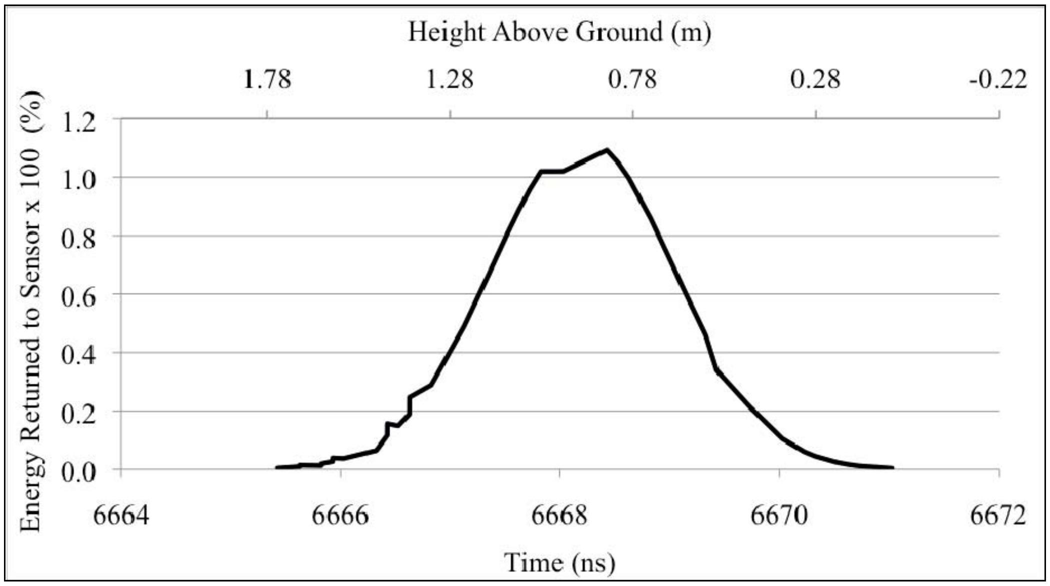


Figure 41. Waveform from LIDAR energy returned to the sensor after interaction with a simulated Lambertian surface "leaf plate"

This test case demonstrates that the model is correctly simulating the way LIDAR energy interacts with a leaf. It is also of interest to determine how the ground underneath the leaf affects the returned waveform.

To explore the effect of the ground underneath the leaf on the LIDAR waveform, a ground surface was added beneath the leaf plate. The reflectance, transmittance, and absorbance values of the leaf remain the same as above (65.57%, 29.83%, and 4.6% respectively), while the ground surface is modeled with a reflectance of 70% and an absorbance of 30%. Energy is not allowed to transmit through the ground. Energy can transmit through the leaf, interact with the ground, and then be transmitted through the leaf again back towards the sensor.

Four situations were modeled:

1. Lambertian leaf over specular ground
2. Lambertian leaf over Lambertian ground
3. Specular leaf over specular ground
4. Specular leaf over Lambertian ground

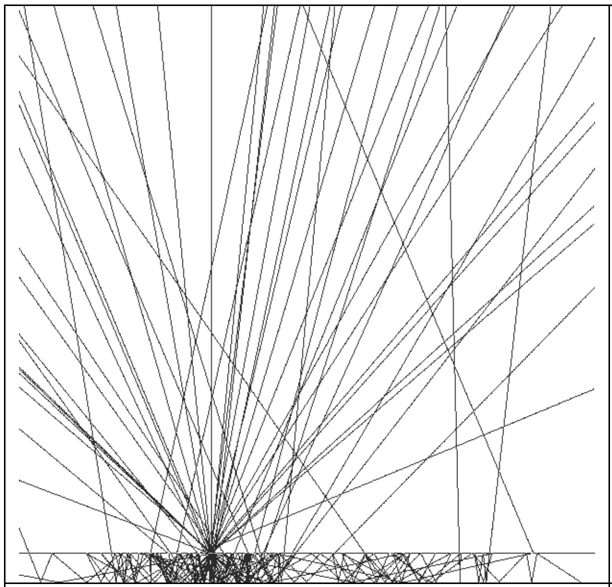
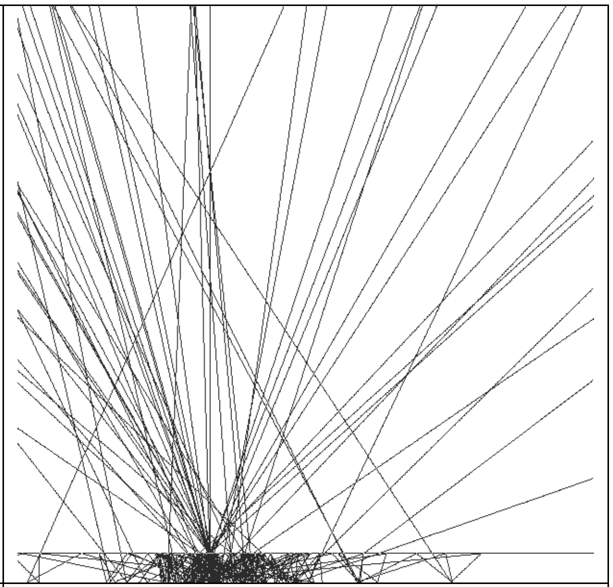
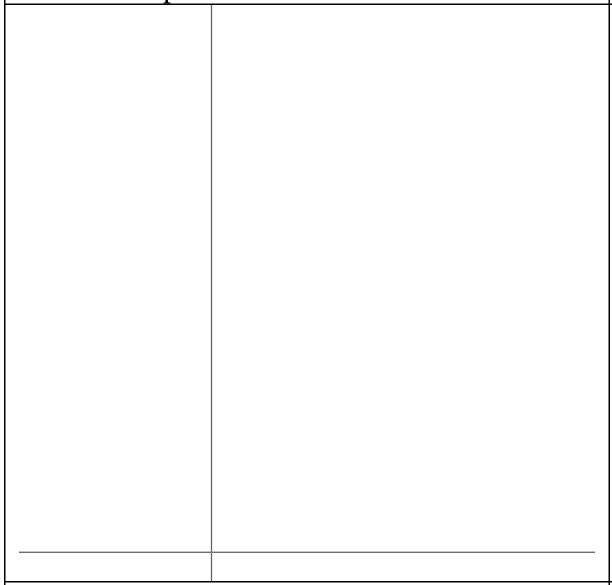
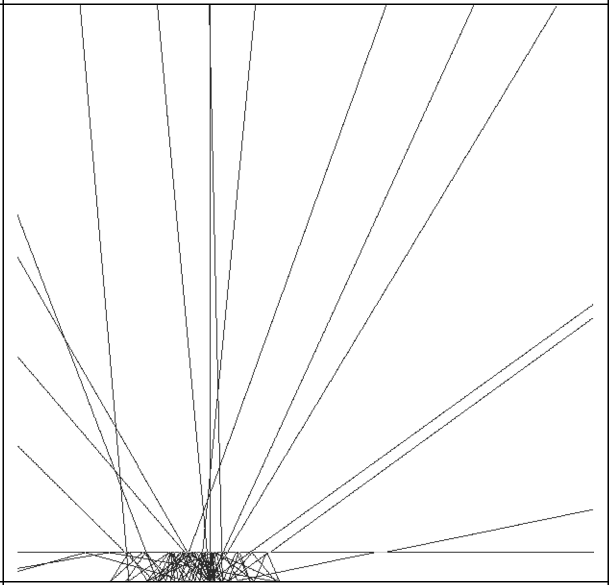
	
Lambertian Leaf Plate above a Specular Ground Surface	Lambertian Leaf Plate above a Lambertian Ground Surface
	
Specular Leaf Plate above a Specular Ground Surface	Specular Leaf Plate above a Lambertian Ground Surface

Figure 42. Interaction of first pulse of LIDAR energy with a simulated “leaf plate” and ground surface

The time-of-flight for a LIDAR pulse to interact with the leaf surface is 6665 ns, and 6671 ns for the ground. In the calculations of “Height Above Ground,” no adjustment has been made for the length of the pulse. As explained previously, time-of-flight is measured from the transmission of the leading edge of the Gaussian pulse, so energy returned from the trailing edge of the pulse will appear to be returned from below ground level, as the measured time will be longer than 6671 ns.

In Figures 43 through 46, the first two peaks correspond to the return of energy from the leaf, and the return of energy from the ground.

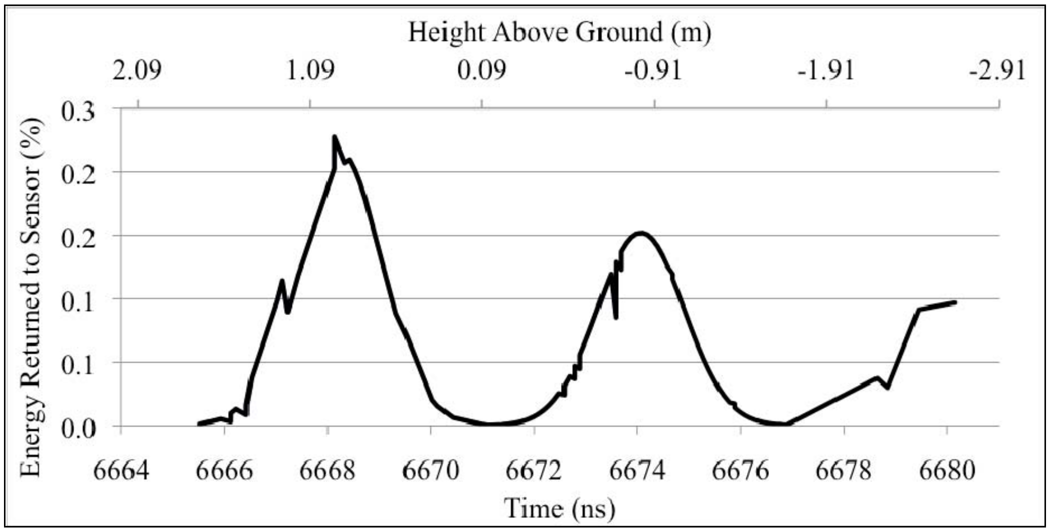


Figure 43. Simulated waveform for a Lambertian leaf over specular ground surface

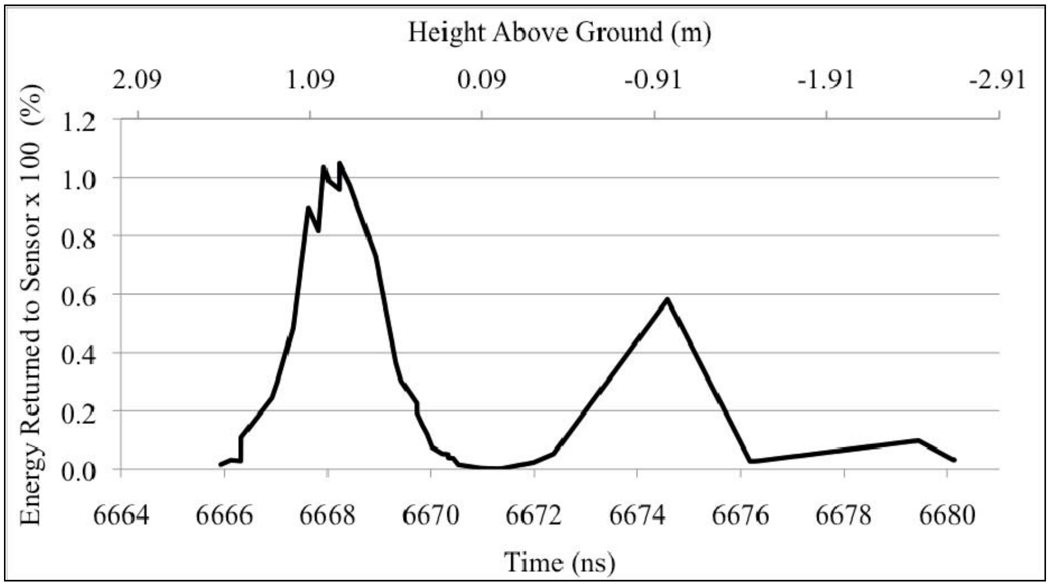


Figure 44. Simulated waveform for a Lambertian leaf over Lambertian ground surface

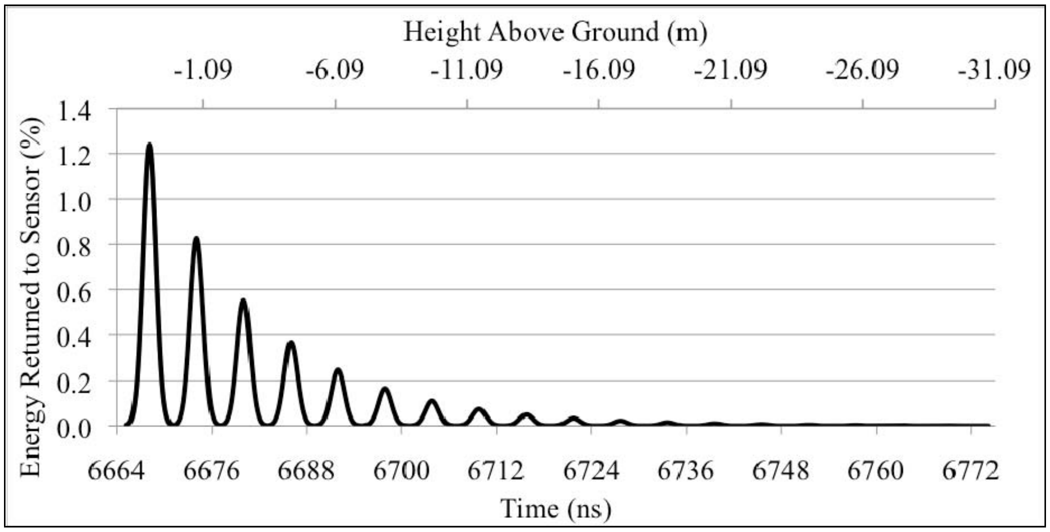


Figure 45. Simulated waveform for a specular leaf over a specular ground surface

In Figure 45, both the leaf surface and the ground surface are modeled as specular surfaces. The first two peaks in the waveform occurring at 6668 ns and 6674 ns are the initial responses from the leaf and the ground surface. The following diminishing peaks correspond to energy that is reflected once, twice, three times, etc., between the leaf and the ground before being returned to the sensor. These multiple reflections give the appearance that energy is being returned from below ground level, whereas in fact the negative values are due to the excess time taken for the light to reflect back and forth between the leaf plate and the ground.

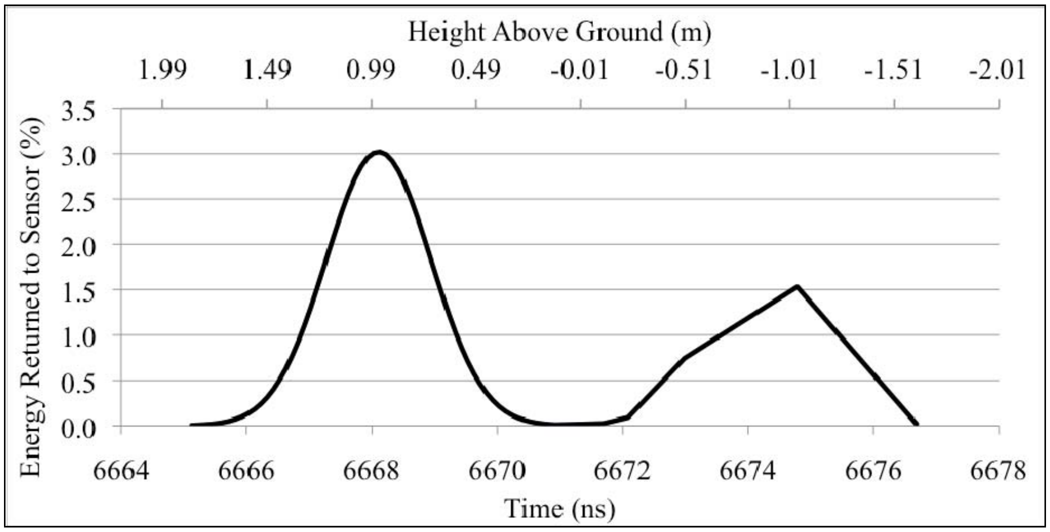


Figure 46. Simulated waveform for a specular leaf over Lambertian ground surface

Table 10 summarizes the amount of energy returned to the sensor in each simulation.

Simulation	Total Energy Returned
Lambertian leaf only	0.1%
Specular leaf only	65.72%
Lambertian leaf over specular ground	6.25%
Lambertian leaf over Lambertian ground	0.11%
Specular leaf over specular ground	77.2%
Specular leaf over Lambertian ground	65.6%

Table 10. Amount of energy returned to the sensor for cases of a leaf over ground

The addition of a ground surface under the leaf can have a fairly significant impact if the ground is modeled as a specular surface (Note the increase in energy returned from 65.72% to 77.2% when a specular ground surface is added under a specular leaf). When the ground is modeled as a Lambertian surface, there is relatively little impact on the total energy received, but the shape of the waveform changes significantly as seen in Figures 43 through 46.

In all cases, the first return is higher in energy than secondary returns.

Significantly negative values for “Height Above Ground” are due to longer amounts of time taken for energy to bounce between the leaf plate and the ground.

4. Test Case 4–Interaction with a Tree

This test case was used to demonstrate how the simulated LIDAR interacts with the tree. All materials were assumed to have Lambertian scattering properties.

The likelihood of the bark map was reduced to 10%, and the probability of the leaf map was reduced to 3%. These values were chosen after multiple simulations showed that these values would allow energy to interact with the ground. In the author’s experience, LIDAR data collected over a tree will include returns from the ground surface, even in cases of dense canopy.

When a LIDAR pulse encounters a pixel that contains a leaf, there is only a 3% of interaction with a leaf, and a 97% chance of interaction with air. In the locations where the leaf and bark maps overlap, there will be a 3% chance of interaction with a leaf, a 10% chance of interaction with bark, and a 87% chance of interaction with air.

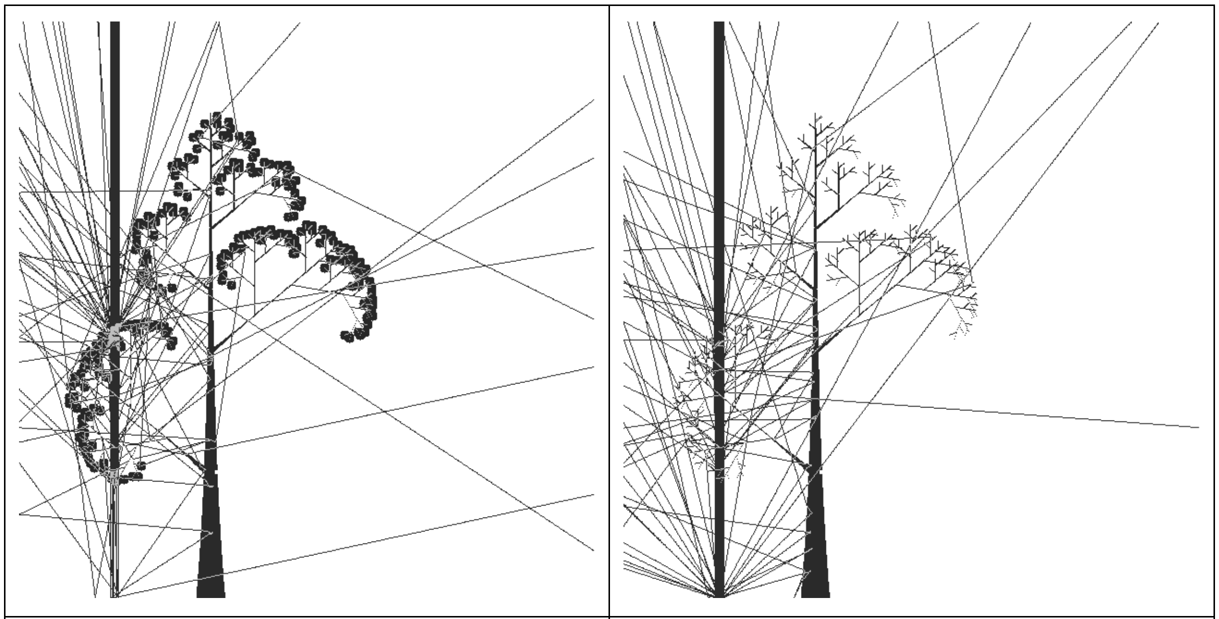


Figure 47. Interaction of the first pulse of LIDAR energy with a tree

Using Figure 29 as a reference, the returns from the branches should occur between 6615 and 6645 ns after pulse transmission. As can be seen in the simulated waveforms in Figure 48, in the “leaf-on” situation, energy is only being returned from the leaves. The probability of the leaf map was reduced to 3% to ensure energy could be detected after interaction with the ground, but combining the 3% leaf map with the 11% bark map means that the ground is not detected in the “leaf-on” case. In the “leaf-off” case, energy is returned from the branches, and also from the ground.

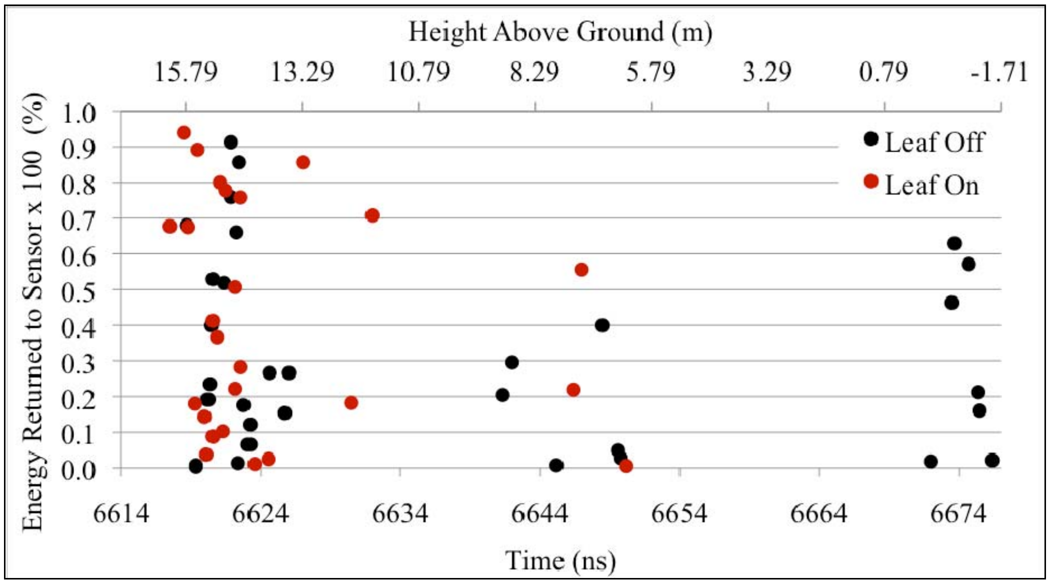


Figure 48. Simulated waveforms for a tree with and without leaves

In both the leaf-on and leaf-off situations, there is an initial return occurring at approximately 6615 ns. This corresponds to returns from the branches and leaves of the tree. A second peak can be seen in the leaf-off waveform at approximately 6675 ns, which corresponds to the ground return. There is significant scattering from the branches and leaves, as evidenced by the spread of returns from 6615 to 6660 ns. Returns occurring after 6675 ns are due to laser energy reflecting from the ground, and being scattered within the tree canopy, and then back to the sensor.

The total energy returned to the sensor in the "leaf on" case was 10.5% of the energy transmitted. In the "leaf off" case, 0.1% of the transmitted energy was returned.

In this test case, a wide peak corresponding to returns from the canopy was expected, rather than a series of very distinct points corresponding to the branch returns as shown in Figure 48 between 6615 and 6650 ns. These waveforms were created after 500 model iterations. It is expected that these simulated waveforms would become smoother with many more model iterations, as the number of distinct returns increases with more iterations as discussed in "Test Case 2–Varying Number of Iterations." At the time of this writing, attempts were being made to run the simulation on high performance computing resources, but some recoding of the program will be required to realize the benefit of multiple processor availability.

5. Test Case 5–Single versus Multiple Scattering

Each time a pulse of light interacts with a material in the scene, a scattering event is counted. When a LIDAR pulse is simulated, the Gaussian pulse is divided into multiple small bins, and each bin is propagated separately as a LIDAR pulse. Figure 49 shows the number of interactions for each of 60 Gaussian bin elements for a LIDAR pulse interacting with a tree without leaves (Test Case 3). The average number of interactions is 2.85. When the tree has leaves, the average number of interactions increases to 3.28. It should be noted that the number of interactions is also dependant on the probability maps of the materials in the scene. As in Test Case 3, the probability of the bark map was reduced to 10%, and the leaf map was reduced to 3%.

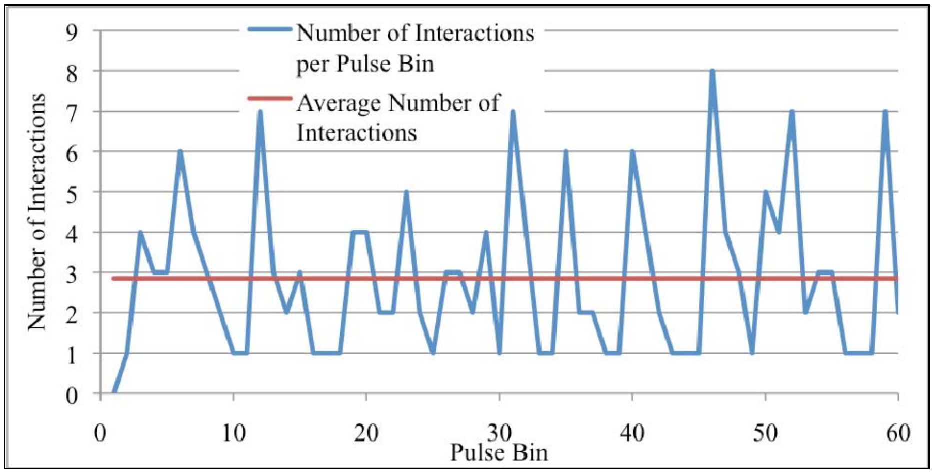


Figure 49. The number of LIDAR/material interactions per Gaussian bin element

6. Test Case 6–Varying Ground Reflectance Values

The purpose of this test is to determine how the reflectivity of the ground underneath the tree affects the LIDAR waveform. It is expected that more of a ground return will be seen when the ground is more reflective.

In these test cases, the leaf map is again reduced to 3% probability, while the bark map is reduced to 10% probability. The reflectance of the ground is varied from 10% to 70%. The transmittance of the ground is 0%, and so the absorbance varies from 90% to

30%. The reflectance properties of the leaves of the tree are kept constant at 35% reflectance, 8% transmittance, and 56% absorbance.

Just as in “Test Case 3–Interaction with a tree,” the returns from the branches should occur between 6615 and 6645 ns after pulse transmission. Interaction with the ground should occur starting at 6671 ns.

A plot of the PROSPECT leaf reflectance spectra with each laser wavelength overlaid is shown in Figure 50 to indicate the brightness of the leaves. The reflectance of the ground is varied from 10% to 70%, and so will sometimes be brighter than the leaves. The bark is modeled as having 10% reflectance at 532 nm, 35% at 1064 nm, and 45% at 1550 nm (see Table 4).

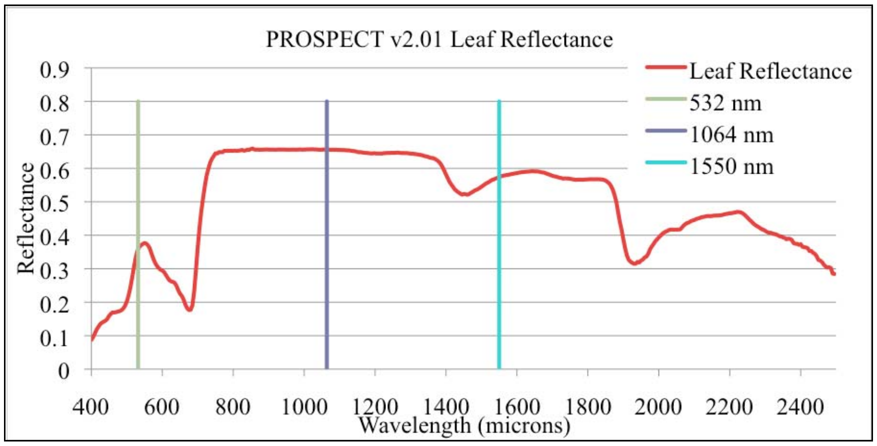


Figure 50. PROSPECT leaf reflectance spectra overlaid with lines indicating 532 nm, 1064 nm and 1550 nm

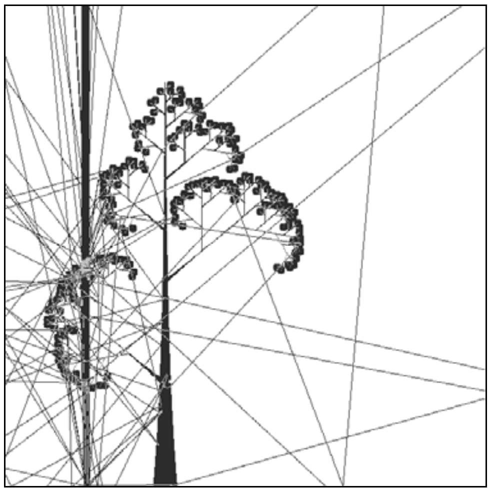
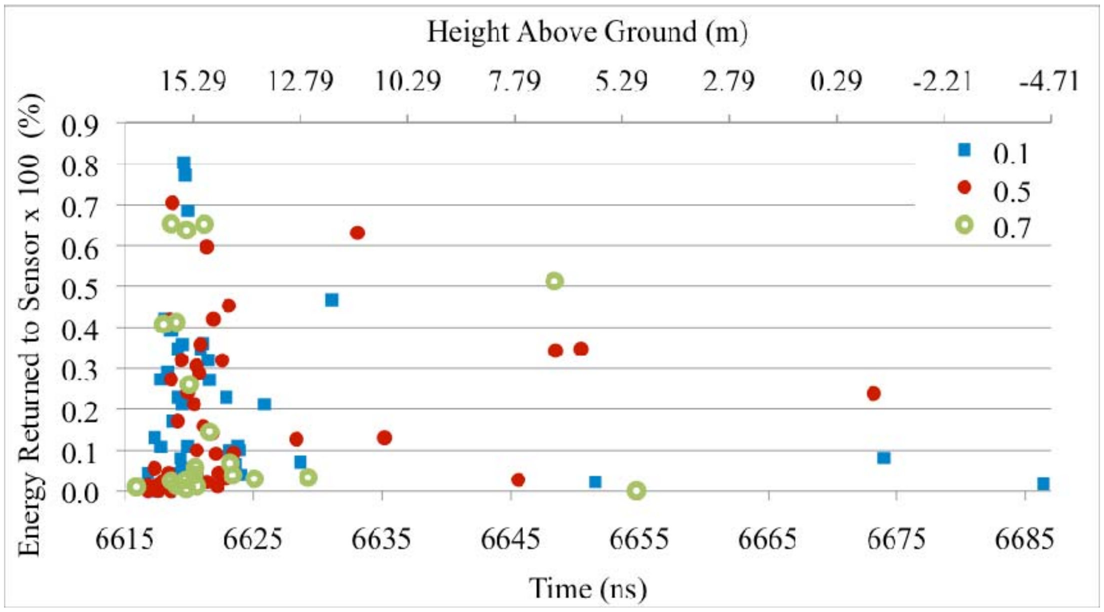
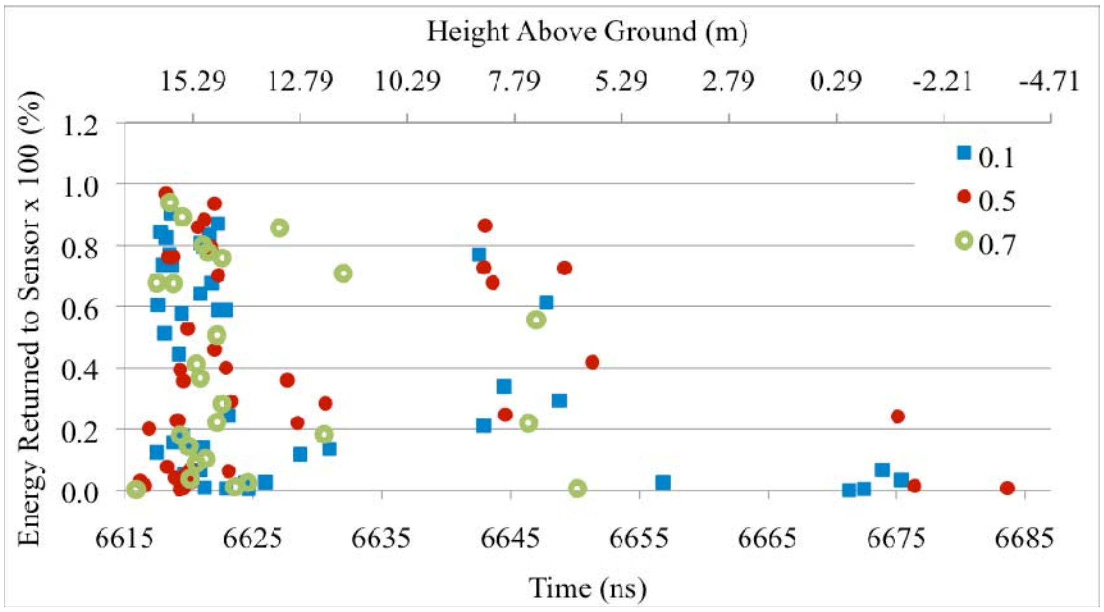


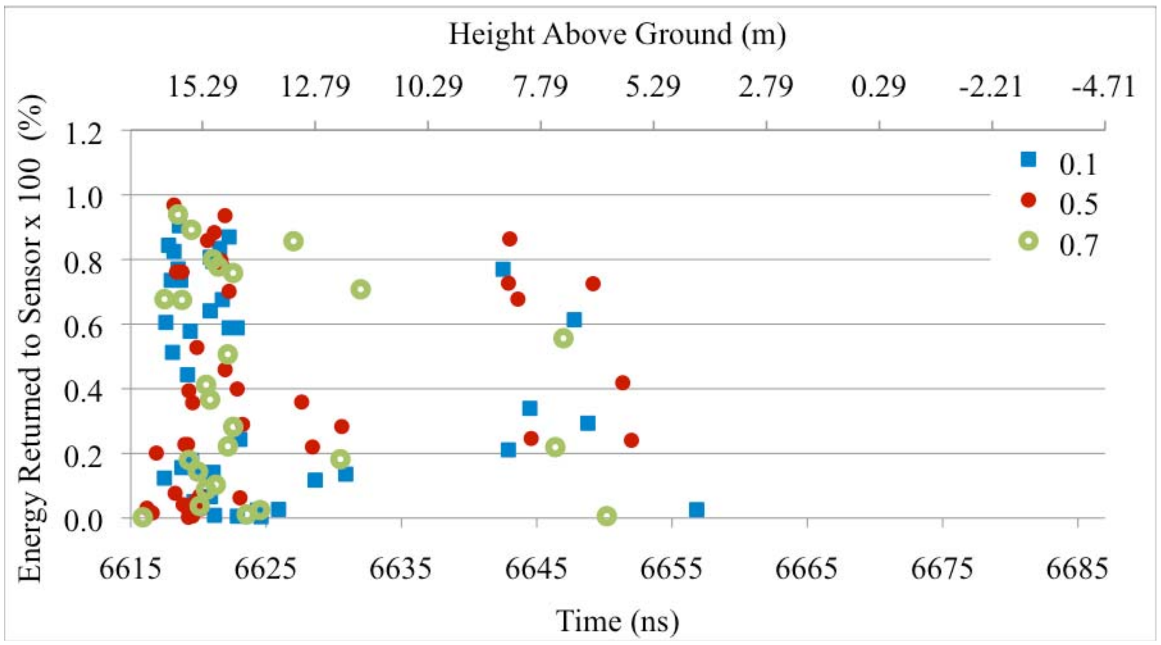
Figure 51. Interaction of the first pulse bin of LIDAR energy



Laser operating at 532 nm



Laser operating at 1064 nm



Laser operating at 1550 nm

Figure 52. Simulated waveforms for a LIDAR operating at 532, 1064, and 1550 nm. The reflectance of the ground is varied from 10% to 70% (and absorbance varies from 90% to 30%)

Surprisingly, when the laser is operating at 532 and 1064 nm, the ground can be detected when the reflectance of the ground is low, but not when the ground is brighter (70% reflective).

Again, these waveforms were not as smooth as expected. These waveforms were created from 500 model iterations.

7. Test Case 7–Varying Atmosphere

According to Mallet and Bretar, atmospheric effects are negligible unless there is rain (Mallet, 2008). For all of the test cases, the effects of the atmosphere in the immediate vicinity of the tree were ignored, but the total energy of the transmitted pulse was reduced according to the atmospheric absorption and the altitude of the sensor.

To test the effects of the atmosphere, the LIDAR was modeled as interacting with a flat specular ground surface having a reflectance of 70%, and an absorbance of 30%.

As demonstrated in Test Case 1, this simulation returns energy as expected. In this test case, the angular beam spread was set to 0 to avoid energy being lost due to scattering outside the receiving optics aperture.

The amount of energy transmitted through the atmosphere diminishes exponentially with altitude.

$$E_{abs} = E_0 - E_0 e^{-kz}$$

where E_{abs} is the amount of energy absorbed in the atmosphere, E_0 is the initial energy of the transmitted LIDAR pulse, *k* is the atmospheric absorption coefficient, and z is the distance traveled through the atmosphere.

According to Measures, a clear atmosphere (15 miles visibility) has a *k* value of $0.018*10^{-5}$ (Measures, 1992).

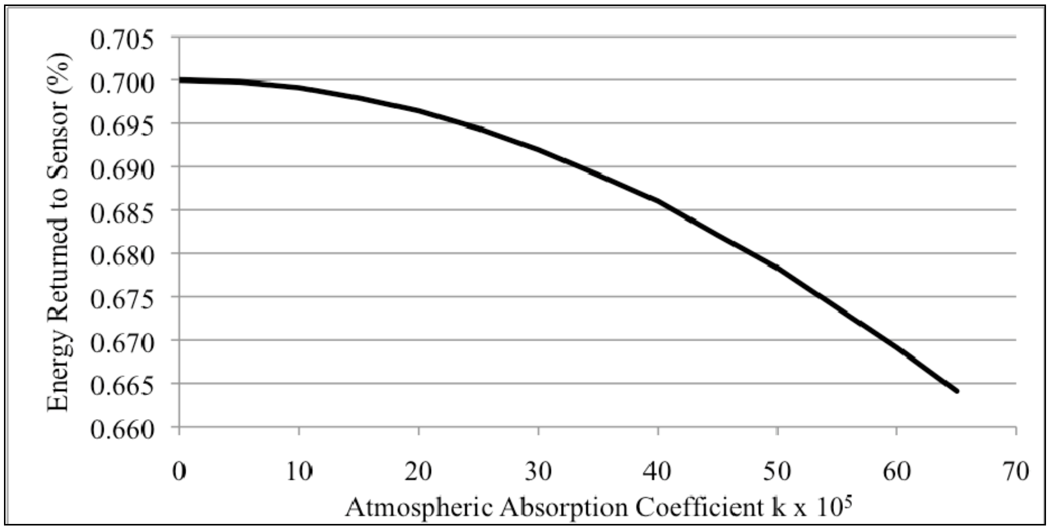


Figure 53. Plot of energy returned to the sensor after reflection from a 70% reflective specular surface under varying atmospheric conditions

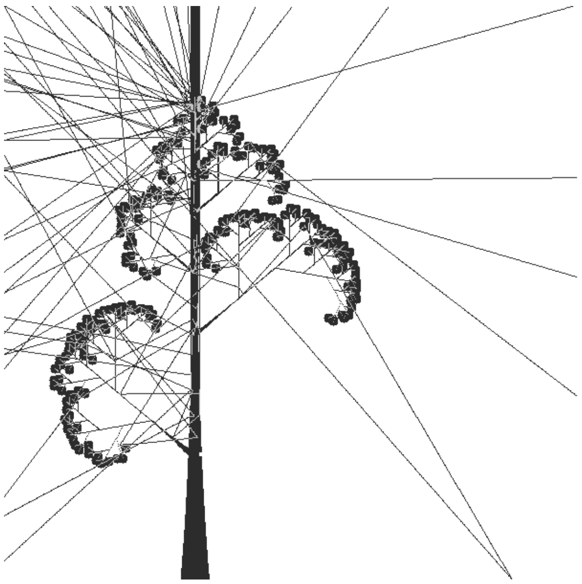
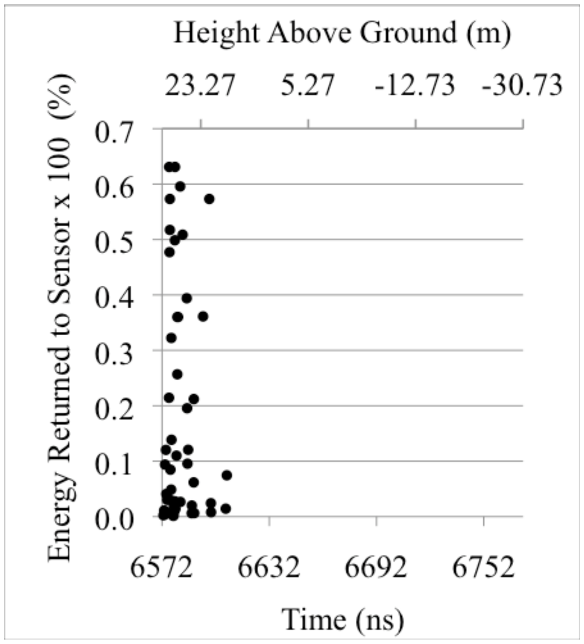
Adjusting the atmospheric absorption coefficient has very little effect until the atmospheric absorption coefficient *k* is increased by a factor of 1000.

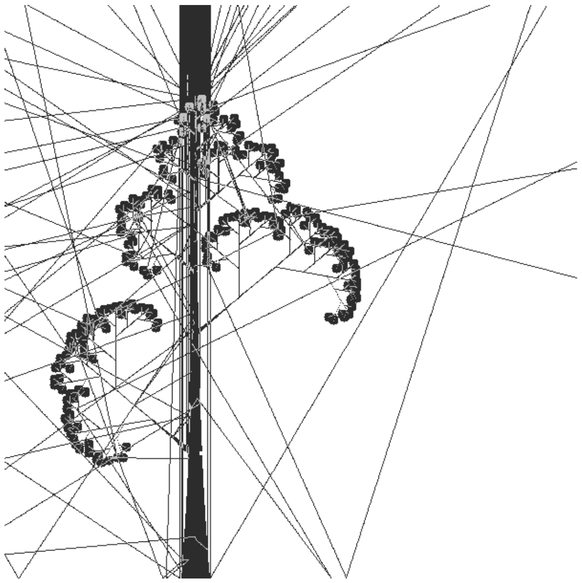
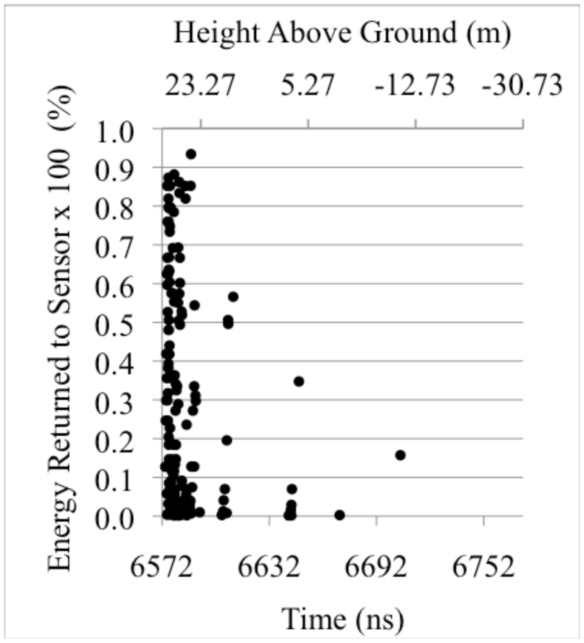
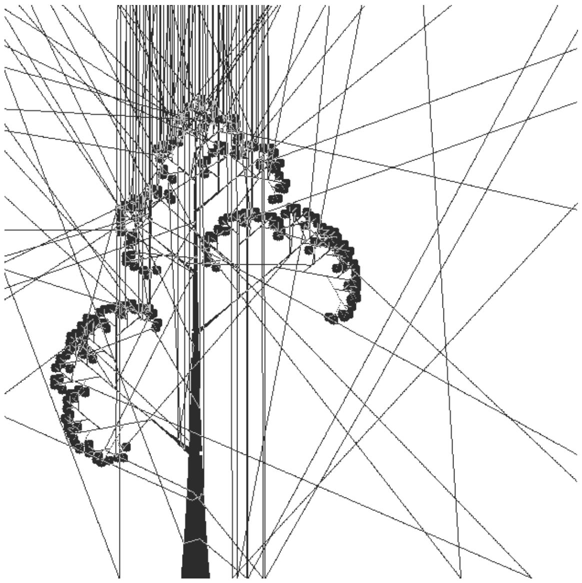
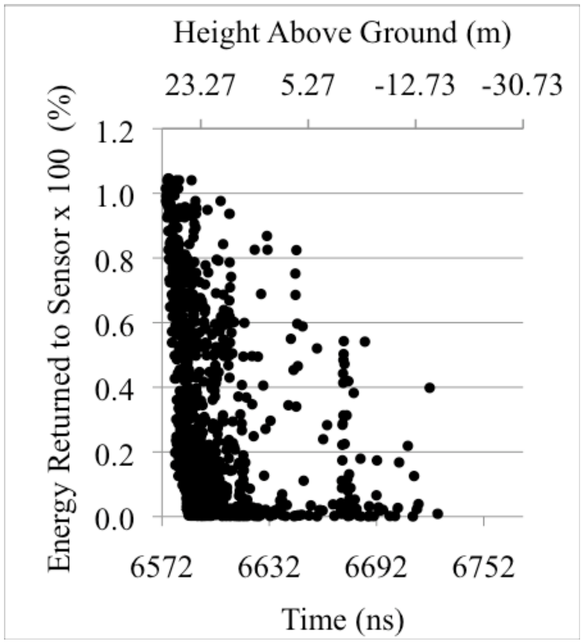
8. Test Case 8–LIDAR Footprint Size

Small footprint systems (0.3−2 m diameter) provide accurate measures of small objects within the laser footprint, but these systems may miss tree-tops, or, may hit tree-tops but miss the ground. The returns from large footprint systems (10–70 m diameter) will include information from all of the objects within the footprint, which could lead to confusion. However, the large footprint system is more likely to hit both the tree-top and the ground within the same footprint.

To simulate larger footprint sizes, the angular spread of the beam and the receiving optics aperture of the LIDAR system were increased. The LIDAR was simulated directly overtop the tree.

The top of the tree is 15 m tall, and the sensor is 1000 m above ground level, and so energy returned from the top of the tree should arrive after 6571 ns. The ground should be detected starting at 6671 ns.

	
Beam spread = 0.3 mrad, Beam diameter on ground = 0.3 m	Total energy returned to sensor = 0.09% Average number of interactions = 3.67

	
Beam spread = 1.0 mrad, Beam diameter on ground = 1.0 m	Total energy returned to sensor = 0.49% Average number of interactions = 3.47
	
Beam spread = 5.0 mrad, Beam diameter on ground = 5.0 m	Total energy returned to sensor = 3.87% Average number of interactions = 3.85

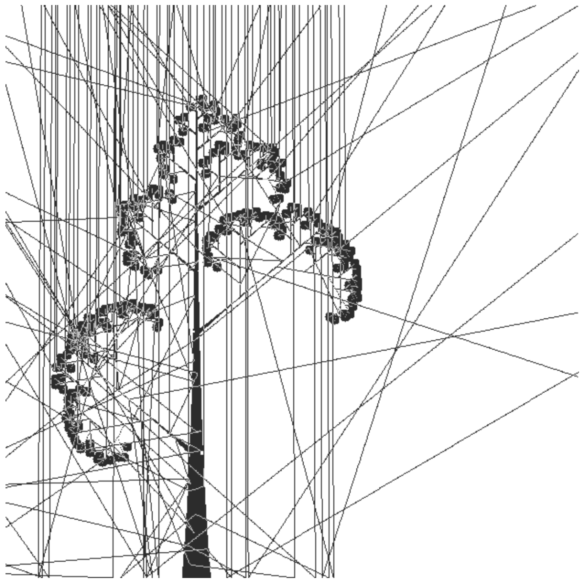
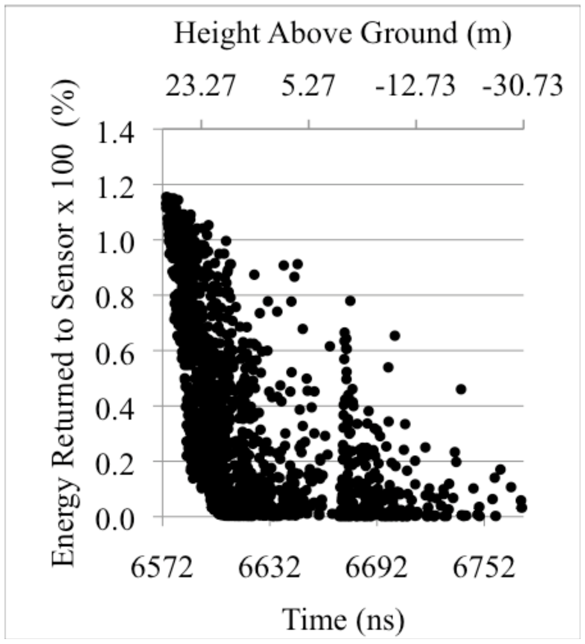
	
Beam spread = 10.0 mrad, Beam diameter on ground = 10.0 m	Total energy returned to sensor = 7.18% Average number of interactions = 3.85

Figure 54. Interactions of first pulses of LIDAR energy and simulated waveforms for increasing LIDAR footprint sizes

When the beam spread is 0.3 mrad, the footprint of the LIDAR is only 30 cm, and the LIDAR energy does not reach the ground. When the LIDAR footprint is increased, the ground is detected.

In the larger footprint simulations (5 m, 10 m), there is a significant amount of energy returned to the sensor after 6671 ns, and these returns appear as being returned from locations below ground level. It is possible that this time-spread is caused by energy being bounced around the tree canopy before being returned to the sensor. In the case of LIDAR interacting with a flat Lambertian ground surface, as in Test Case 1, the last energy returned to the sensor was detected at 6676 ns. With the large footprint simulations, LIDAR energy is returning almost 100 ns after this. Light travels approximately 30 m in 100 ns. The average number of interactions in the 5 m and 10 m footprint simulations is 3.85, which is slightly higher than the average number of interactions in the small footprint simulations.

9. Test Case 9–Off-nadir Sensing Geometry

This test case demonstrates the capability of the model to simulate off-nadir sensing geometries. This is an important feature because almost always, LIDAR systems use some type of scanning, and so data is collected at off-nadir directions.

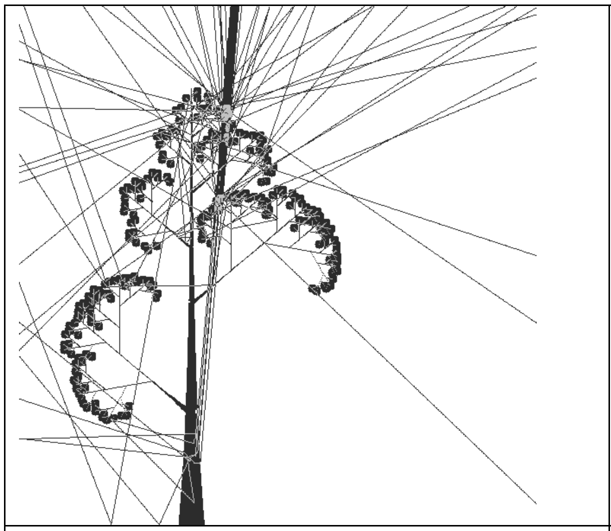
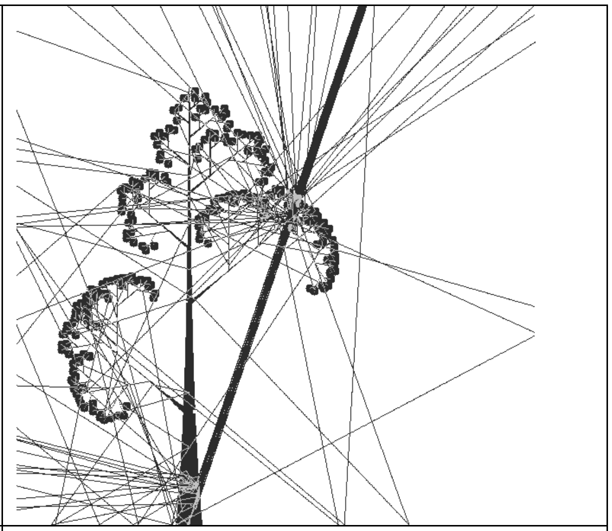
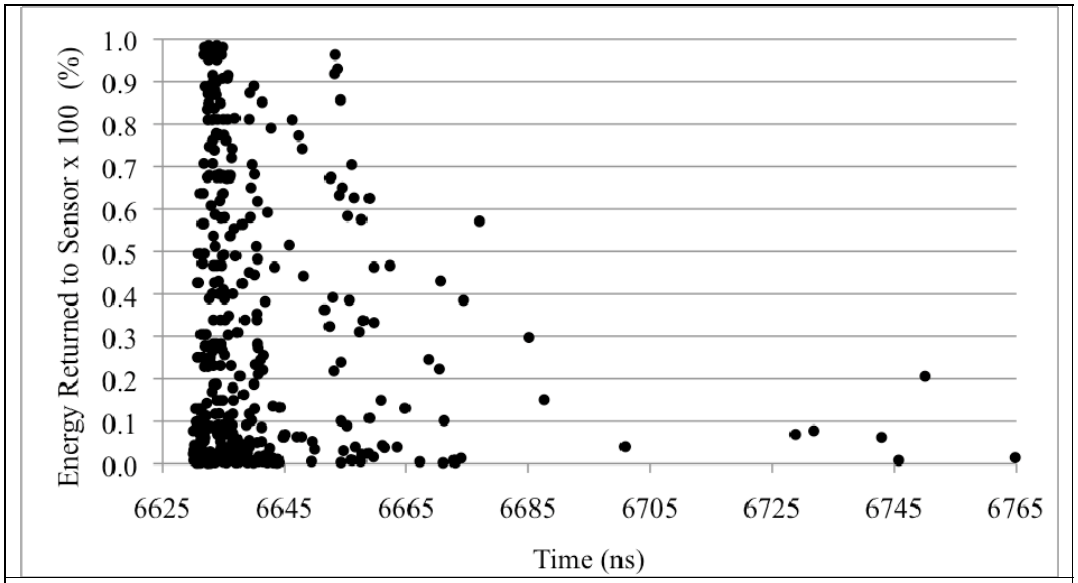
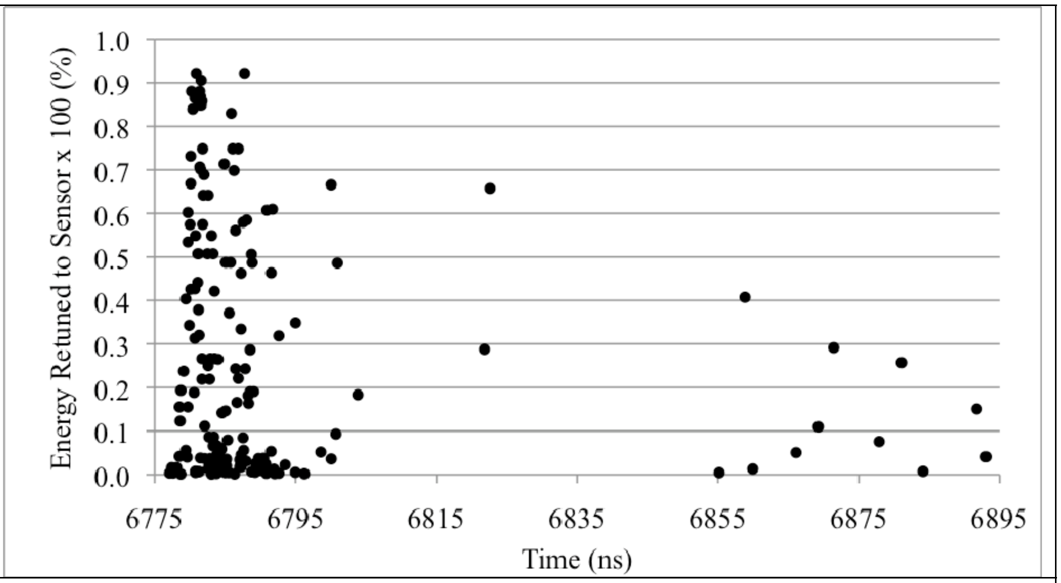
	
Sensor pointing 4.9º off-nadir	Sensor pointing 18.8º off-nadir

Figure 55. Interactions of first pulses of LIDAR energy for a LIDAR sensor pointing off-nadir



Sensor pointing 4.9º off-nadir



Sensor pointing 18.8º off-nadir

Figure 56. Simulated waveforms for a LIDAR sensor pointing off-nadir

THIS PAGE INTENTIONALLY LEFT BLANK

IV. CONCLUSIONS AND FUTURE WORK

The model presented in this thesis seems to reliably predict full-waveform LIDAR signatures for the relatively simple situations explored so far. It is expected that the waveforms from LIDAR interaction with a tree will be more realistically simulated by using many more model iterations. The simulated waveforms need to be compared to real LIDAR waveforms. Because of the relative simplicity and flexibility of the model, it can be used to determine the source of differences between the simulated and actual full-waveform LIDAR data, and this information will be valuable as algorithms for the exploitation of full-waveform LIDAR data are developed.

The model as presented here is a very simplistic representation of a complex situation. Further realism could be achieved by expanding the model to 3 dimensions, and by incorporating a more physically realistic vegetation model.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. IDL CODE FOR LIDAR SIMULATION

```
pro LIDAR_model
;Take tree model as input
;Propagate lidar pulse through the tree
;Output lidar waveform

;*************************
;Input filenames - tree model
;*************************
   dir = '/Users/admin/Documents/'
   datein = ' 9Sep' ;date of files to reuse
   barkin = dir + 'bark_'+datein+'.dat'
   leafin = dir + 'leaf_'+datein+'.dat'
   airin = dir + 'air_'+datein+'.dat'
   anglein = dir + 'angle_map_'+datein+'.dat'
   xcoordin = dir + 'bud_xcoord_'+datein+'.dat'
   ycoordin = dir + 'bud_ycoord_'+datein+'.dat'

;*************************
;Output filenames
;*************************
  outdir = '/Users/admin/Documents/tree/
  temp = systime()
  month = strmid(temp,4,3)
  day = strmid(temp, 8,2)
  date = strtrim(day)+month+'Lidar_model_output'
  print, 'Date:', date
  hit_map_file = outdir + 'hit_map_'+date+'.dat'
  waveform_file = outdir + 'waveform_'+date+'.txt'
  sampled_wf_file = outdir + 'sampled_wf_'+date+'.txt'
  metadata_file = outdir + 'metadata_'+date+'.txt'
  all_pulses_file = outdir + 'all_pulses_'+date+'.txt'
  gaussian_file = outdir + 'initial_pulse_'+date+'.txt'
  sampled_gaussian_file = outdir + 'sampled_initial_pulse_'+date+'.txt'

;*************************
;Define workspace parameters
;*************************
;Work Space - must be same as or bigger than tree size
  ;x = horizontal axis, oriented from L to R
  ;y = vertical axis, oriented from bottom to top
  x = 600
  y = 600
  tree_height_m = 15
  ground = fltarr(x,y) & ground(*,*) = 0 ;pixels above bottom of workspace
  ground_height = 0
  ground_angle_map = fltarr(x,y) & ground_angle_map(*,*) = 0 ;ground is flat unless
otherwise defined
  visibility_km = 15 ;clear - p. 143 Measures
  npulses = 500 ;number of separate pulses - response will be combined

  show = 1 ;set to 1 to display hit-maps every nth_step pulse
  nth_step = 10

  mult_scatter = 1 ;set to 0 to restrict number of scatters allowed
    max_scatters = 1 ;set number of scattering events allowed
  bark_prob = 0.1 ;Probability that a pixel containing bark will be treated as bark
  leaf_prob = 0.03
  air_abs_prob = 0.
  ground_scatter = 1 ;0 - Specular scattering; 1 - Lambertian scattering
  leaf_scatter = 1
  bark_scatter = 1

;*************************
;Define LIDAR parameters
```

```
;************************
  init_energy = 100d ;Percent
  min_detect_energy = .001
  pulse_length_ns = 6d ;time-spread of initial lidar pulse
  beam_spread_rad = 0.3/1000d  ;0.3 - 2.5 mrad - typical values for ALS
  sensor_alt_m = 1000 ;sensor altitude in meters
  aperture_diam_m = 0.2
  ;Pulse initial location in pixels and angle
  x_loc_orig = 200
  y_loc_orig = y-1
  angle_orig = 3*!pi/2. ;measured CCW from horizontal
  wavelength_nm = 1064
  sample_rate_GHz = 5. ;15GHz = 1cm ranging resolution, Toth p. 15

;************************
;Define PROSPECT leaf parameters
;Jacquemoud - 1996
;************************
n = 3.4  ;leaf equivalent thickness - lower n = higher transmittance - varies from 1 to 5
cab = 20.0 ;chlorophyll a+b - varies from 0 to 100
cw = 0.002 ;water - varies from 0 to 0.050
cp = 0.001 ;protein - varies from 0 to 0.01
cc = 0.001 ;structural biochemicals-hemicellolose, cellulose, lignin-varies from 0-0.01
deg = 0    ;leaf angle - varies from 0 to 90 degrees CCW from horizontal

;************************
;Open and read tree input files
;************************
  bark_map = fltarr(x,y)
  leaf_map = fltarr(x,y)
  xcoord = fltarr(x,y)
  ycoord = fltarr(x,y)
  angle_map = fltarr(x,y)
  OPENR, 1, barkin & READU, 1, bark_map & CLOSE, 1
  OPENR, 1, leafin & READU, 1, leaf_map & CLOSE, 1
  OPENR, 1, xcoordin & READU, 1, xcoord & CLOSE, 1
  OPENR, 1, ycoordin & READU, 1, ycoord & CLOSE, 1
  OPENR, 1, anglein & READU, 1, angle_map & CLOSE, 1

;************************
;Determine step sizes
;************************
  cm_per_pixel = double(tree_height_m)*100d/double(max(ycoord))
  c_cm_per_s = 2.9979246e+10 ;299 792 458 m / s
  c_cm_per_ns = 29.979246
  ;Make time_step_ns so that dist_step_pix = 1
  time_step_ns = cm_per_pixel/c_cm_per_ns
  dist_step_cm = time_step_ns*c_cm_per_ns
  dist_step_pix = dist_step_cm/cm_per_pixel
  model_sample_rate = 1./time_step_ns
  aperture_diam_pix = aperture_diam_m*100.*(1./cm_per_pixel)
  print, 'Time step (ns):', time_step_ns
  print, 'Distance (cm) pulse will travel in one time step:', dist_step_cm
  print, 'Distance (pixels) pulse will travel in one time step:', dist_step_pix
  print, 'Centimeters per pixel:', cm_per_pixel
  print, 'Sample rate of model (GHz):',model_sample_rate

;************************
;Determine atmospheric absorption above workspace - assuming uniform atmosphere
; Beer-Lambert -> I/I(0) = e^-(kz)
; k = attenuation coefficient
;************************
workspace_ht_m = y*cm_per_pixel/100.
k_m = 0.018*10^(-5d) ;m^-1
z_m = sensor_alt_m - workspace_ht_m
atm_abs = init_energy - init_energy*exp(-k_m*z_m)

;************************
;Define material interaction parameters - %
```

```
;*************************
leaf_result = call_function('prospect2_nogui_fxn', n, cab, cw, cp, cc, deg,
wavelength_nm)
leaf_refl = leaf_result(0) & leaf_trans = 1.0-leaf_result(1) & leaf_abs = leaf_result(2)
bark_trans = 0.17 & bark_abs = 0.13 & bark_refl = 0.6
  k_pix = k_m/100.*cm_per_pixel
;Create air absorbance map based on altitude
air_abs = fltarr(x,y)
FOR i=0, y-1 DO BEGIN
  z_m = cm_per_pixel*(i+1)/100.
  air_abs(*,y-i-1) = exp(-k_m*z_m)
ENDFOR
air_abs = air_abs*air_abs_prob
air_trans = 1.-air_abs
ground_refl = .7 & ground_abs = .3 & ground_trans = 0.

;*************************
;Determine footprint size on ground and at top of workspace
;*************************
beam_width_ground_cm = (2.*sensor_alt_m*tan(beam_spread_rad/2.))*100.
beam_spread_top_cm = (2.*(sensor_alt_m-workspace_ht_m)*tan(beam_spread_rad/2.))*100.
beam_spread_top_pix = beam_spread_top_cm/cm_per_pixel
print, 'Beam width at top of workspace (cm):', beam_spread_top_cm

;***************************
;Create material probability maps
;***************************
  bark_map = bark_map*bark_prob
  leaf_map = (1.-bark_map)*leaf_map*leaf_prob
  air_map = 1. - (bark_map + leaf_map)
  IF show EQ 1 THEN BEGIN
    WINDOW, 0, xsize = x, ysize = y
    temp_tree = fltarr(x,y,3)
    TVSCL, bark_map/max(bark_map)+leaf_map/max(leaf_map)
  ENDIF

  ;Create 'hit_map' to show where and interaction occurs
  hit_map = fltarr(x,y)
  hit_map(*,*) = 0
;*************************
;Define pulse shape
;Vary pulse energy according to a Gaussian distribution
;*************************
  mean = 0.
  variance = 14.
  gaussian = dblarr(100)
  i = 0.
  WHILE (i LT n_elements(gaussian)) DO BEGIN
    gaussian(i) = (1./sqrt(variance*2.*!pi))* $
       exp(-1.*((i-n_elements(gaussian)/2.)^2)/(2.*(variance^2)))
    i = i+1
  ENDWHILE
  npulse_bins = fix(pulse_length_ns/time_step_ns)
  gaussian = interpol(gaussian, npulse_bins)
  gaussian = (gaussian/total(gaussian))*(init_energy-2.*atm_abs)
  gauss_bin_width = n_elements(gaussian)/float(npulse_bins)
  gauss_bin_width_ns = pulse_length_ns/npulse_bins ;should be 1 bin per time step
  gauss_bin_width_pix = gauss_bin_width_ns*c_cm_per_ns*dist_step_pix

  IF max(gaussian) LE min_detect_energy THEN stop

;*************************
;Propagate the pulse(s)
;*************************
pulse_return_tot = dblarr(2*npulses,npulse_bins)
pulse_width_step = beam_spread_top_pix/npulse_bins
beam_spread_step = beam_spread_rad/npulse_bins
nflags_tot = dblarr(5, npulse_bins)
     flag_key = ['flag 1: exceeding max scatters', $
```

```
                  'flag 2: absorbed', $
                  'flag 3: lost in space', $
                  'flag 4: RTS', $
                  'flag 5: too low initially']

nbounces_pulse = 0
FOR p=0, npulses-1 DO BEGIN
nflags = dblarr(5, npulse_bins)

print, 'Pulse #',p+1, ' of', npulses
  ;Create array to store waveform
  pulse_return2 = dblarr(2,npulse_bins) & pulse_return2(*,*) = 0

  ;Determine length of waveform
  max_n_steps = 0

  x_loc = x_loc_orig
  y_loc = y_loc_orig

  xloc_init = x_loc
  yloc_init = y_loc
  angle_init = angle_orig
  bar = (indgen(npulse_bins)+1)*gauss_bin_width
  total_energy=0
 ;Find time taken for pulse to travel through atm
  atm_travel_time = 2.*(sensor_alt_m-workspace_ht_m)*100./c_cm_per_ns   bin = 0.
  returns = 0
  nbounces = 0
  WHILE (bin LT fix(npulse_bins)-1) DO BEGIN

    ;Track number of bounces
    bounce = 0
    pulse_init_time_steps = bin*gauss_bin_width_ns ;number of steps

  ;******* Vary spatial start location to allow random start within pulse_width_step area
  temp = randomu(seed)-0.5 ;temp ranges between -0.5 and +0.5
    x_loc = xloc_init - beam_spread_top_pix/2. + $
       pulse_width_step*bin + temp*pulse_width_step
    angle = angle_init - beam_spread_rad/2 + beam_spread_step*bin + temp*beam_spread_step

    y_loc = yloc_init
    x_dir = cos(angle)
    y_dir = sin(angle) ;-1 = pulse directed directly down towards ground

    energy = total(gaussian(gauss_bin_width*bin:(bar(bin)-1)))
    bin_energy = energy
    total_energy = total_energy+energy
    IF total_energy GT init_energy then stop
    n_steps = 0
    flag = 0
    path = intarr(x,y)
    n = 0l
    rts = 0 ;Change to 1 if photon exits space and returns to sensor
    nabs = 0l
    WHILE (flag EQ 0) DO BEGIN
      IF energy LE min_detect_energy THEN BEGIN
        flag = 2
        nflags(4,bin) = nflags(4,bin)+energy
        xloc_new = x_loc & yloc_new = y_loc
        GOTO, JUMP1
      ENDIF
      ;***************
      ;Move the photon
      xloc_new = x_loc + dist_step_pix*x_dir
      yloc_new = y_loc + dist_step_pix*y_dir

      n = n+1 ;steps

      IF (xloc_new GE fix(x) OR xloc_new LT 0 OR yloc_new GE fix(y)) THEN GOTO, JUMP1
      ;************************
      ;Impact with ground
      ;************************
      ;find max height of ground at xloc_new
      ground_height = max([max(where(ground(xloc_new, *) EQ 1)), 0])
      IF yloc_new LE ground_height THEN BEGIN
        bounce = bounce+1
        IF (mult_scatter EQ 0 AND bounce GE max_scatters) THEN BEGIN
          flag = 1
          nflags(0,bin) = nflags(0,bin)+energy
          GOTO, JUMP1
        ENDIF

        ground_angle = ground_angle_map(xloc_new, ground_height)
        abs = ground_abs
        refl = ground_refl
        trans = ground_trans
       ;track amount absorbed, propagate remaining
        nflags(1,bin) = nflags(1,bin)+energy*abs
        energy = energy - energy*abs ;less percentage absorbed
        IF energy LT min_detect_energy THEN BEGIN
          flag = 2 ;energy absorbed
          GOTO, JUMP1
        ENDIF
        ;Rest is reflected
        IF ground_scatter EQ 0 THEN BEGIN ;specular reflection
           yloc_new = ABS(yloc_new-ground_height) + ground_height
           axis = ground_angle + !pi/2. ;axis to reflect about
           ;angle at which pulse strikes - CW from axis
           diff = !pi/2. - (ground_angle - angle)
           angle = axis - diff
        ENDIF ELSE BEGIN ;Lambertian scatter
          yloc_new = ABS(yloc_new-ground_height) + ground_height
          temp = randomu(seed)*2.-1 ;-1<temp<1
          angle = acos(temp) + ground_angle
        ENDELSE
        ;Move the photon
        x_dir = cos(angle) & y_dir = sin(angle)
        xloc_new = x_loc + dist_step_pix*x_dir
        yloc_new = y_loc + dist_step_pix*y_dir
        n = n+1 ;steps
      ENDIF
  ;If outside bounds of tree, determine if photon will be counted as 'returned to sensor'
      IF (xloc_new GE fix(x) OR xloc_new LT 0) THEN GOTO, JUMP1
      ground_height = max([max(where(ground(xloc_new, *) EQ 1)), 0])
      IF (yloc_new GE fix(y) OR yloc_new LT ground_height) THEN GOTO, JUMP1

      ;Trace photon path and create 'hit_map'
      startx = min([x_loc, xloc_new]) & endx = max([x_loc, xloc_new])
      starty = min([y_loc, yloc_new]) & endy = max([y_loc, yloc_new])
        IF (fix(endx) - fix(startx)) GT 0 THEN BEGIN
          m = (yloc_new - y_loc)/(xloc_new - x_loc)
          b = y_loc - m*x_loc
          FOR i = startx, endx DO BEGIN
            hit_map(i,m*i+b) = 1
          ENDFOR
          IF (fix(yloc_new) - fix(y_loc)) GT 0 THEN BEGIN
            FOR i = y_loc, yloc_new DO BEGIN
              hit_map((i-b)/m,i) = 1
            ENDFOR
          ENDIF
       ENDIF ELSE BEGIN
         hit_map(x_loc, starty:endy)=1
       ENDELSE

      ;Update pulse location
      x_loc = xloc_new & y_loc = yloc_new

      ;Determine probability of material interaction
```

```
  ;measured at nearest integer pixel location
leaf_weight = leaf_map(x_loc, y_loc)
bark_weight = bark_map(x_loc, y_loc)
air_weight  = air_map(x_loc, y_loc)

;Determine light pulse interaction
;Randomly choose number between 0 and 1
temp = randomu(seed)
IF (temp LE leaf_weight) THEN impact = 0 ;leaf
IF (temp GT leaf_weight AND temp LE leaf_weight+bark_weight) THEN impact = 1 ;bark
IF (temp GT leaf_weight+bark_weight) THEN impact = 2 ;air
;******************
;* Impact with leaf
;******************
IF (impact EQ 0) THEN BEGIN ;impact with leaf
  bounce = bounce+1
  IF (mult_scatter EQ 0 AND bounce GE max_scatters) THEN BEGIN
    flag = 1
    nflags(0,bin) = nflags(0,bin)+energy
    GOTO, JUMP1
  ENDIF

  trans = leaf_trans
  abs   = leaf_abs
  refl  = leaf_refl

  ;Track energy being absorbed...
  nflags(1,bin) = nflags(1,bin)+energy*abs
  ;Subtract photons being absorbed
  energy = energy - energy*abs
  IF energy LT min_detect_energy THEN BEGIN
    nflags(1,bin) = nflags(1,bin)+energy
    flag = 2
    GOTO, JUMP1
  ENDIF
  ;Determine transmission or reflection of remaining energy
  temp = randomu(seed)*(refl+trans)
  IF (temp LE trans) THEN interact = 0 ;transmission - nothing changes
  IF (temp GT trans) THEN interact = 2 ;reflection - change angle

  IF (interact EQ 2) THEN BEGIN ;photon is reflected
    leaf_angle = deg*(!pi/180.) ;angle CCW of leaf from horizontal
    IF leaf_scatter EQ 0 THEN BEGIN ;Specular reflection
      ;Reflect from plane of leaf - leaf_angle
      axis = leaf_angle + !pi/2. ;axis to reflect about
     ; angle at which pulse strikes - CW from axis
      diff = !pi/2. - (leaf_angle - angle)
      angle = axis - diff
    ENDIF ELSE BEGIN ;Lambertian scattering from leaf surface plane
      ;angle at which pulse strikes - CW from leaf surface
      diff =  angle - leaf_angle
      temp = randomu(seed)*2.-1
         ;temp varies between -1 and +1., acos(temp) varies from 0 to !pi
      IF (diff GT !pi and diff LT 2*!pi) THEN BEGIN ;incoming to top of leaf
        angle = acos(temp) + leaf_angle ;should depend on incoming angle as well...
      ENDIF ELSE BEGIN ;If underneath leaf, then reflect down...
        angle = acos(temp) + leaf_angle + !pi
      ENDELSE
    ENDELSE
  ENDIF
  x_dir = cos(angle) & y_dir = sin(angle)
ENDIF ELSE BEGIN
;******************
;* Impact with bark
;******************
IF (impact EQ 1) THEN BEGIN ;impact with bark
  bounce = bounce+1
  IF (mult_scatter EQ 0 AND bounce GE max_scatters) THEN BEGIN
    flag = 1
```

```
      nflags(0,bin) = nflags(0,bin)+energy
      GOTO, JUMP1
    ENDIF
    temp=randomu(seed)
    trans = bark_trans
    abs   = bark_abs
    refl  = bark_refl

    ;Track energy being absorbed
    nflags(1,bin) = nflags(1,bin)+energy*abs
    ;Subtract photons being absorbed
    energy = energy - energy*abs
    IF energy LT min_detect_energy THEN BEGIN
      nflags(1,bin) = nflags(1,bin)+energy
      flag = 2
      GOTO, JUMP1
    ENDIF
    ;Determine transmission or reflection of remaining energy
    temp = randomu(seed)*(refl+trans)
    IF (temp LE trans) THEN interact = 0 ;transmission - nothing changes
    IF (temp GT trans) THEN interact = 2 ;reflection - change angle
    IF (interact EQ 2) THEN BEGIN ;photon is reflected
      branch_angle = angle_map(x_loc, y_loc) ;angle CCW of branch from horizontal
      IF bark_scatter EQ 0 THEN BEGIN ;Specular reflection
        ;Reflect from plane of bark surface
        axis = branch_angle + !pi/2. ;axis to reflect about
        ; angle at which pulse strikes - CW from axis
        diff = !pi/2. - (branch_angle - angle)
        angle = axis - diff
      ENDIF ELSE BEGIN
        temp = randomu(seed)*2.-1 ;-1<temp<1
        angle = acos(temp) + branch_angle   ;acos(temp) varies between 0 and !pi
      ENDELSE
    ENDIF
    x_dir = cos(angle) & y_dir = sin(angle)
  ENDIF ELSE BEGIN

  ;******************
  ;* Interaction with air
  ;******************
  IF (impact EQ 2) THEN BEGIN ;interaction with air
    temp=randomu(seed)
    abs = (1.-air_abs(x_loc, y_loc))*air_abs_prob
    trans = 1.-abs
  IF (temp LE trans) THEN interact = 0 ;transmission
    IF (temp GT trans) THEN interact = 1 ;absorption

    IF (interact EQ 0) THEN BEGIN ;photon is transmitted
    ENDIF ELSE BEGIN
    IF (interact EQ 1) THEN BEGIN ;photon is absorbed
      nflags(1,bin) = nflags(1,bin)+energy*abs
      energy = energy - energy*abs ; - photon_energy
      IF energy LT min_detect_energy THEN BEGIN
        nflags(1,bin) = nflags(1,bin) + energy
        flag = 2
        GOTO, JUMP1
      ENDIF
    ENDIF
 ENDELSE
    x_dir = cos(angle) & y_dir = sin(angle)
  ENDIF
  ENDELSE
  ENDELSE
  n_steps = n_steps+1
  IF n_steps GT max_n_steps THEN max_n_steps = n_steps

  IF (mult_scatter EQ 0 AND bounce GE max_scatters) THEN BEGIN
    flag = 1
    nflags(0,bin) = nflags(0,bin)+energy
```

```
        GOTO, JUMP1
      ENDIF

JUMP1: IF flag NE 1 AND flag NE 2 AND (xloc_new GE x OR xloc_new LE 0 OR yloc_new LT
ground_height OR yloc_new GE y) THEN BEGIN
           ;Photons lost in space
           flag = 3 ;leave while loop - start new pulse
           nflags(2,bin) = nflags(2,bin) + energy
       ENDIF
      IF flag EQ 3 AND (yloc_new GE y AND xloc_new LE x_loc_orig+aperture_diam_pix/2. $
        AND xloc_new GE x_loc_orig-aperture_diam_pix/2.) THEN BEGIN
 ;Project photon through atm at current heading & check if still within aperture diameter
        xloc_new = xloc_new + (sensor_alt_m - workspace_ht_m)/tan(angle)
        IF (xloc_new LE x_loc_orig+aperture_diam_pix/2. AND $
               xloc_new GE x_loc_orig-aperture_diam_pix/2.) THEN BEGIN

          rts = 1 ;count photon as 'returned to sensor'
          flag = 4
          ;subtract from total exiting workspace but not rts
          nflags(2,bin) = nflags(2,bin)-energy
          nflags(3,bin) = nflags(3,bin)+energy
          ;Add atmospheric travel time
          dist_traveled_time = n_steps * time_step_ns + atm_travel_time
          pulse_return2(*,bin) = $
             [dist_traveled_time+pulse_init_time_steps, nflags(3,bin)]
        ENDIF
      ENDIF
    ENDWHILE ;flag NE 0

     bin = bin+1.
     ;Track number of bounces per pulse bin
     nbounces = [nbounces, bounce]
   ENDWHILE ;bin LT npulse_bins
   nflags_tot = nflags_tot + nflags

  ;Find percentages of points lost
  flag1_pct = total(nflags_tot(0,*))/(total_energy*npulses)
  flag2_pct = total(nflags_tot(1,*))/(total_energy*npulses)
  flag3_pct = total(nflags_tot(2,*))/(total_energy*npulses)
  flag4_pct = total(nflags_tot(3,*))/(total_energy*npulses)
  flag5_pct = total(nflags_tot(4,*))/(total_energy*npulses)

  ;Sort pulse_return2 according to time of RTS
  pulse_return2 = $
  [pulse_return2(0,sort(pulse_return2(0,*))), pulse_return2(1,sort(pulse_return2(0,*)))]
  pulse_return_tot(2*p:2*p+1,*) = pulse_return2

  IF show EQ 1 THEN BEGIN
    IF p mod nth_step EQ 0 THEN BEGIN
      tvscl, byte(((1-hit_map)+(1-bark_map)+(1-leaf_map))*100)
    ENDIF
  ENDIF

  IF p EQ 0 THEN BEGIN
    WRITE_GIF, outdir+'hit_map_GIF_1st_Pulse_'+date+'.gif', $
       byte((hit_map+bark_map+leaf_map+(1-ground))*150)
  ENDIF
  ;Find average number of bounces per bin for each pulse
  nbounces_pulse = $
      [nbounces_pulse, float(total(nbounces))/float(n_elements(nbounces-1.))]
ENDFOR ;p pulse

index = indgen(npulses)

;Delete rows containing only zeros
non_zero = where(total(pulse_return_tot,1) NE 0)
avg_waveform = [0.,0.]
sampled_wf = [0.,0.]
sampled_init = [0.,0.]
```

```
IF non_zero(0) NE -1 THEN BEGIN
iplot, linestyle=6, xtitle='Time (ns)', ytitle='Intensity', $
     view_TITLE = 'Avg Return (blue), Sampled Waveform;'+string(sample_rate_GHz)+ $
     ' GHz (black)', tickfontsize = 12
;Sum returns to sensor occuring at same times for all pulses
  times = pulse_return_tot(2*index,*)
  intens_values = pulse_return_tot(2*index+1,*)
  temp_time = MIN(times(WHERE(times NE 0))) ;find min non-zero time
  min_time = temp_time & max_time = max(times)
  time_axis = dblarr(ceil((max_time - min_time)/time_step_ns+5000))
  intens_tot = time_axis & intens_tot(*) = 0
  t=0
  WHILE temp_time GT 0 DO BEGIN
    same_time = WHERE(float(times) eq float(temp_time))
    IF same_time(0) NE -1 then begin
      intens_tot(t) = total(intens_values(same_time))/(n_elements(same_time))
      times(same_time) = 0
    ENDIF
    time_axis(t) = temp_time
    t=t+1
    temp_time = -1
    pq = where(times NE 0)
    IF pq(0) NE -1 THEN temp_time = MIN(times(pq))
  ENDWHILE

   ;************************
   ;* Match waveform to total energy
   ;************************
   intens_tot = (intens_tot/total(intens_tot))*total_energy*flag4_pct
   avg_waveform = transpose([[time_axis],[intens_tot]])
   ;Get rid of zeros in array
   avg_waveform = avg_waveform(*,where(total(avg_waveform,1) NE 0))

  ;Plot waveform
  iplot, avg_waveform(0,*), avg_waveform(1,*), color = [0,255,255], /overplot
  ;**********************************************
  ;Sample waveform & initial pulse at sample rate
  ;**********************************************
  max_nsteps = ceil((max_time-min_time)*sample_rate_GHz
  sampled_wf = dblarr(2*nrates, max_nsteps)
  max_nsteps_init = ceil(pulse_length_ns*sample_rate_GHz
  sampled_init = dblarr(2*nrates, max_nsteps_init)

  ;Resample so that each bin contains one time step time_step_ns
  nsteps = ceil((max_time-min_time)*sample_rate_GHz)
  rintens = congrid(reform(avg_waveform(1,*)), nsteps, $
                      /center)*(n_elements(avg_waveform(1,*))/float(nsteps))
  rtime = congrid(reform(avg_waveform(0,*)), nsteps, /center)

   sampled_wf(rate*2+1,0:nsteps-1) = rintens
   sampled_wf(rate*2,0:nsteps-1) = rtime
   iplot, rtime, rintens, /overplot

    nsteps_init = ceil(pulse_length_ns*sample_rate_GHz)
    rintens_init = congrid(gaussian, nsteps_init, $
                       /center)*(n_elements(gaussian)/float(nsteps))
    rtime_init = interpol(indgen(pulse_length_ns)+1,nsteps_init)

    sampled_init(rate*2+1,0:nsteps_init-1) = rintens_init
    sampled_init(rate*2,0:nsteps_init-1) = rtime_init
ENDIF
print, '% pts excessively scattered, absorbed, lost in space, too low initially:'
print, flag1_pct, flag2_pct, flag3_pct, flag5_pct
print, '% pts returned to sensor:', flag4_pct
print, 'Ground refl, ground abs:', ground_refl, ground_abs
print, 'Leaf refl, trans, abs:', leaf_refl, leaf_trans, leaf_abs

;**********************************************
;Create output files
```

```
;**********************************************
WRITE_GIF, outdir+'hit_map_GIF_'+date+'.gif', $
       byte(((1-hit_map)+(1-bark_map)+(1-leaf_map))*100)
openw, 1, hit_map_file
writeu, 1, hit_map/max(hit_map) + bark_map/max(bark_map) + leaf_map/max(leaf_map)
close, 1

openw, 1, waveform_file
printf, 1, 'Energy, time (ns); created from',npulses,' pulses.'
printf, 1, 'total energy:', total(avg_waveform(1,*))
printf, 1, avg_waveform
close, 1

openw, 1, sampled_wf_file
printf, 1, 'Created from',npulses,' pulses, sampled at', sample_rate_GHz, ' GHz'
printf, 1, 'Total energy:',total(sampled_wf(1,*))
printf, 1, 'Energy, time (ns)'
printf, 1, sampled_wf
close, 1

openw, 1, all_pulses_file
  non_zero = where(total(pulse_return_tot,1) NE 0)
  pulse_return_tot_out = [0.,0.]
  IF non_zero(0) NE -1 THEN BEGIN
    pulse_return_tot_out = $
       pulse_return_tot(*,non_zero(0):non_zero(n_elements(non_zero)-1))
  ENDIF
  writeu, 1, pulse_return_tot_out
close, 1

openw, 1, gaussian_file
printf, 1, 'Energy per sample'
printf, 1, 'total energy:', total(gaussian)
printf, 1, transpose(gaussian)
close, 1

openw, 1, sampled_gaussian_file
printf, 1, 'Energy per sample'
printf, 1, 'Sample rate GHz:', sample_rate_GHz
printf, 1, 'total energy:', total(gaussian)
printf, 1, [sampled_init(0,*), sampled_init(1,*)]
close, 1

openw, 1, metadata_file
printf, 1, 'Date of input files:  ', datein
printf, 1, 'Date of output files:  ', date
IF mult_scatter EQ 1 THEN printf, 1, 'Multiple scattering allowed'
IF mult_scatter EQ 0 THEN BEGIN
  printf, 1, 'Number of scattering events restricted'
  printf, 1, 'Maximum number of scattering events allowed:', max_scatters
ENDIF
printf, 1, 'Lidar parameters:'
printf, 1, '     Laser Wavelength (nm):', wavelength_nm
printf, 1, '     Pulse length (ns):', pulse_length_ns
printf, 1, '     Beam angular spread (rad):', beam_spread_rad
printf, 1, '     Sensor altitude (m):’, sensor_alt_m
printf, 1, '     Transmitted pulse energy:', init_energy
printf, 1, '     Minimal detection energy:', min_detect_energy
printf, 1, '     Initial pulse angle (degress cc from horizontal):', angle_orig*180/!pi
printf, 1, '     Initial pulse location (x,y) pixels:', x_loc_orig,',',y_loc_orig
printf, 1, ''
printf, 1, '     Centimeters per pixel:', cm_per_pixel
printf, 1, '     Time step (ns):', time_step_ns
printf, 1, '     Speed of light c (cm/s):', c_cm_per_s
printf, 1, '     Speed of light c (cm/ns):', c_cm_per_ns
printf, 1, '     Distance traveled per step (cm):',  dist_step_cm
printf, 1, '     Distance traveled per step (pixels):', dist_step_pix
printf, 1, '     Beam diameter (cm) at top of workspace:', beam_spread_top_cm
printf, 1, '     Beam diameter (cm) on ground:', beam_width_ground_cm
```

```
printf, 1, ''
printf, 1, 'Tree height (m):', tree_height_m
printf, 1, 'Workspace dimensions (pixels, (x,y)):', x,',',y
printf, 1, 'Workspace dimensions (meters, (x,y)):', $
            cm_per_pixel*x/100.,',', cm_per_pixel*y/100.
printf, 1, ''
printf, 1, 'Bark probability:', bark_prob
printf, 1, 'Leaf probability:', leaf_prob
printf, 1, 'Air probability:', air_abs_prob
printf, 1, 'Number of pulses:', npulses
printf, 1, ''
printf, 1, 'Leaf reflectance, transmittance, absorbance:'
printf, 1, leaf_refl,', ', leaf_trans,', ', leaf_abs
printf, 1, 'Bark reflectance, transmittance, absorbance:'
printf, 1, bark_refl,', ', bark_trans,', ', bark_abs
printf, 1, 'Ground reflectance, transmittance, absorbance:'
printf, 1, ground_refl, string(1-1), ground_abs
printf, 1, ''
printf, 1, 'Distribution of points by fate (% of total):'
printf, 1, '  Points exceeding maximum number of scatters allowed:', flag1_pct
printf, 1, '  Points having energy lower than min detection threshold:', flag2_pct
printf, 1, '  Points exiting the workspace, but not returning to the sensor:', flag3_pct
printf, 1, '  Points returning to the sensor:', flag4_pct
printf, 1, '  Points below min detection threshold before propagation:', flag5_pct
printf, 1, ''
printf, 1, 'number of bounces'
printf, 1, transpose(nbounces)
close, 1

print, 'FINISHED'
END
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. PROSPECT LEAF REFLECTANCE MODEL

```
;*********************************************************
function prospect2_noGUI_fxn, n, cab, cw, cp, cc, deg, wl
;*************************************************************
; IDL version of the Prospect Leaf Reflectance Model
;
; Jacquemoud S., Ustin S.L., Verdebout J., Schmuck G., Andreoli G.,
; Hosgood B. (1996), Estimating leaf biochemistry using the PROSPECT
; leaf optical properties model, Remote Sens. Environ., 56:194-202
;
; Jacquemoud S., Baret F. (1990), PROSPECT: a model of leaf optical
; properties spectra, Remote Sens. Environ., 34:75-91.
;
; Glenn Newnham 09.03.00
; last updated 14.04.00
; amp 30 Jul 09
;
scrap = strarr(1)
data = dblarr(7,421)
wave = dblarr(421)
refra = dblarr(421)
ke = dblarr(421)
kab = dblarr(421)
kw = dblarr(421)
kp = dblarr(421)
kc = dblarr(421)
k = dblarr(421)
tau = dblarr(421)
tav1 = dblarr(421)
tav2 = dblarr(421)

refl = fltarr(421)
tran = fltarr(421)
ref_tran = fltarr(2,421)
modelled = fltarr(3,421)

measw=findgen(421)*5+400

;read in wavelength, leaf refractive index and specific absorption coefficients
valeur='/Users/admin/Documents/prospect_idl/valeur2.txt'
openr, 1, valeur
readf, 1, scrap
readf, 1, data
 wave=data(0,*)
 refra=data(1,*)
 ke=data(2,*)
 kab=data(3,*)
 kw=data(4,*)
 kp=data(5,*)
 kc=data(6,*)
close, 1

iplot, fltarr(421), fltarr(421), xtitle='Wavelength (nm)', $
  ytitle='Reflectance / 1-Transmittance', view_TITLE = 'Leaf Reflectance'; Varying
Parameter:  ';+param ;, $
mfile = ''

;compute the total leaf absorption coefficient from biochemical components
 k=ke+(cab*kab+cw*kw+cp*kp+cc*kc)/n

;compute the total leaf transmission coefficient for leaf internal material
tau = call_function('leaf',k)

;find the average interface transmittance for isotropic light
alpha = 90*(!pi/180)
```

```
tav1 = call_function('avg',refra,alpha)

;find the average interface transmittance for the solid angle incident on the leaf
surface
alpha = deg*(!pi/180)
tav2 = call_function('avg',refra ,alpha)

;compute the reflectance and transmittance for a single layer
;Allen et al, 1969
;Jacquemoud and Baret, 1990
x1=1-tav1
x2=tav1^2*tau^2*(refra^2-tav1)
x3=tav1^2*tau*refra^2
x4=refra^4-tau^2*(refra^2-tav1)^2
x5=tav2/tav1
x6=x5*(tav1-1)+1-tav2
r=x1+x2/x4
t=x3/x4
ra=x5*r+x6
ta=x5*t

;compute the reflectance and transmittance for n leaf layers
;Stokes, 1862
delta=(t^2-r^2-1)^2-4*r^2
alfa=(1+r^2-t^2+sqrt(delta))/(2*r)
beta=(1+r^2-t^2-sqrt(delta))/(2*r)

va=(1+r^2-t^2+sqrt(delta))/(2*r)
vb=sqrt(beta*(alfa-r)/(alfa*(beta-r)))

s1=ra*(va*vb^(n-1)-va^(-1)*vb^(-(n-1))) $
  +(ta*t-ra*r)*(vb^(n-1)-vb^(-(n-1)))
s2=ta*(va-va^(-1))
s3=va*vb^(n-1)-va^(-1)*vb^(-(n-1)) $
  -r*(vb^(n-1)-vb^(-(n-1)))

refl=s1/s3
tran=s2/s3

step = 10
color = [255-(step+2)*20, 255-(step+1)*10, step+80]
iplot, measw, refl, /overplot, COLOR = color
iplot, measw, 1-tran, /overplot, COLOR = color

trani = reform(1-tran)

;Print values to a text file
text_file = '/Users/admin/Documents/prospect_idl/PROSPECT_output_'+systime()+'.csv'
openw, 1, text_file
printf, 1, 'PROSPECT parameters:'
printf, 1, '  n =',n
printf, 1, '  cab =', cab
printf, 1, '  cw =', cw
printf, 1, '  cp = ', cp
printf, 1, '  cc = ', cc
printf, 1, '  deg = ', deg
printf, 1, ''
printf, 1, 'Wavelength, Reflectance (x'+string(n_elements(param_array))+'), Transmittance
(x'+string(n_elements(param_array))+'),'
FOR j=0, 420 DO BEGIN
  refls = string(refl, format = '(f10.7)')
  trans = string(trani, format = '(f10.7)')
  printf, 1, string(measw(j))+','+refls(j)+','+trans(j)
ENDFOR
close, 1

band = where(measw EQ wl)
IF band(0) EQ -1 THEN BEGIN
  ;interpolate
  band2 = n_elements(where(measw lt wl))
  band1 = n_elements(where(measw lt wl))-1
  refl_wl = ((wl-measw(band1))/(measw(band2)-measw(band1)))*(refl(band2)-
refl(band1))+refl(band1)
  trani_wl = ((wl-measw(band1))/(measw(band2)-measw(band1)))*(trani(band2)-
trani(band1))+trani(band1)
ENDIF ELSE BEGIN
  refl_wl = refl(band)
  trani_wl = trani(band)
ENDELSE
refl_tran_abs = [refl_wl, trani_wl, 1.-(refl_wl+(1.-trani_wl))]

return, refl_tran_abs
end

;*****************************************************************************
function leaf, k
;
; Transmission coefficient for the leaf internal material
; Allen et al, 1969, eq.14
;
; Uses the NAG Fortran routine s13aaf
;
; Glenn Newnham 09.03.00
; last updated 14.04.00
;
i=0
x=dblarr(421)
y=dblarr(421)
exint = dblarr(421)
tau = dblarr(421)
deg = 0.0
alpha = 0.0

;compute transmission coefficient tau (Allen et al. 1969, eq.14)

for i=0,420 do begin

case 1 of

  (k(i) le 0): tau(i)= 1

    (k(i) gt 0) and (k(i) le 4): begin
    x=(0.5*k(i))-1

    y=(((((((((((((((-3.60311230482612224d-13 $
    *x+3.46348526554087424d-12)*x-2.99627399604128973d-11) $
        *x+2.57747807106988589d-10)*x-2.09330568435488303d-9) $
        *x+1.59501329936987818d-8)*x-1.13717900285428895d-7) $
        *x+7.55292885309152956d-7)*x-4.64980751480619431d-6) $
        *x+2.63830365675408129d-5)*x-1.37089870978830576d-4) $
        *x+6.47686503728103400d-4)*x-2.76060141343627983d-3) $
        *x+1.05306034687449505d-2)*x-3.57191348753631956d-2) $
        *x+1.07774527938978692d-1)*x-2.96997075145080963d-1

        y=(y*x+8.64664716763387311d-1)*x+7.42047691268006429d-1
        exint=y-alog(k(i))
    tau(i) = (1.0-k(i))*exp(-k(i))+k(i)^2*exint
    end

  (k(i) gt 4) and (k(i) lt 85): begin
    x=14.5/(k(i)+3.25)-1

    y=(((((((((((((((-1.62806570868460749d-12 $
        *x-8.95400579318284288d-13)*x-4.08352702838151578d-12) $
        *x-1.45132988248537498d-11)*x-8.35086918940757852d-11) $
        *x-2.13638678953766289d-10)*x-1.10302431467069770d-9) $
        *x-3.67128915633455484d-9)*x-1.66980544304104726d-8) $
        *x-6.11774386401295125d-8)*x-2.70306163610271497d-7) $
        *x-1.05565006992891261d-6)*x-4.72090467203711484d-6) $
        *x-1.95076375089955937d-5)*x-9.16450482931221453d-5) $
        *x-4.05892130452128677d-4)*x-2.14213055000334718d-3

        y=((y*x-1.06374875116569657d-2)*x-8.50699154984571871d-2)*x $
        +9.23755307807784058d-1
    exint=exp(-k(i))*y/k(i)
    tau(i) = (1-k(i))*exp(-k(i))+k(i)^2*exint
    end

  (k ge 85): tau(i) = (1-k(i))*exp(-k(i))

endcase

endfor

return, tau

end


;*************************************************************************************
function avg, refra, alpha
;
; Evaluate the average transmittance across an interface between two dielectrics
; for any solid angle of incidence
;
; Stern, 1964
; Allen, 1973
;
; Glenn Newnham 09.03.00
; last updated 14.04.00
;
a=0.0
b=0.0
ds=0.0
b0=dblarr(421)
b1=dblarr(421)
b2=dblarr(421)
ts=dblarr(421)  ;parallel polarised transmittance
tp1=dblarr(421)
tp2=dblarr(421)
tp3=dblarr(421)
tp4=dblarr(421)
tp5=dblarr(421)
tp=dblarr(421)  ;perpendicular polarised transmittance
tav=dblarr(421) ;total transmittance for the solid angle of incidence

;compute the single interface average transmittance for the solid angle of incidence
a=((refra+1)^2)/2
b=-(((refra^2)-1)^2)/4
ds=sin(alpha)

case 1 of

  (alpha eq 0): tav=4*refra/(refra+1)^2

  (alpha ne 0): begin
    if (alpha eq !pi/2) then begin
      b1=0
    endif else begin
      b1=sqrt((ds^2-(refra^2+1)/2)^2+b)
    endelse

    b2=ds^2-(refra^2+1)/2
    b0=b1-b2

    ts=(b^2/(6*b0^3)+(b/b0)-(b0/2))-(b^2/(6*a^3))-(b/a)+(a/2)
    tp1=-2*(refra^2)*(b0-a)/(refra^2+1)^2
    tp2=-2*(refra^2)*(refra^2+1)*alog(b0/a)/(refra^2-1)^2
    tp3=refra^2*(1/b0-1/a)/2
    tp4=(16*refra^4*(refra^4+1))*alog((2*(refra^2+1)*b0-(refra^2-1)^2)/(2*(refra^2+1) $
      *a-(refra^2-1)^2))/((refra^2+1)^3*(refra^2-1)^2)
    tp5=16*refra^6*(1/(2*(refra^2+1)*b0-(refra^2-1)^2)-1/(2*(refra^2+1) $
      *a-(refra^2-1)^2))/(refra^2+1)^3
    tp=tp1+tp2+tp3+tp4+tp5

    tav=(ts+tp)/(2*ds^2)
    end
endcase

return, tav

end
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. IDL CODE FOR L-SYSTEM TREE MODEL

```
;L_system_tree
;*****************************************************
;* Define the L_system
;*
;* V = {
;*  1, 0 = generates a straight segment of length d
;*  3 = [ = start a branch at an angle counterlockwise 'branch_angle'
;*          degrees from current heading
;*  4 = ] = end a branch and return to base of branch
;*  5 = ( = start a branch at an angle clockwise 'branch_angle'
;*          degrees from current heading
;*  6 = ) = end a branch and return to base of branch
;*   }
;* P = {
;*  0 -> string1
;*  1 -> string2
;*  [ -> [, ] -> ], ( -> (, ) -> )
;*   }
;*
;*  'axiom_orig' = axiom to start the L-system generation
;*
;*  'branch_angle' = angle turtle will turn when a branch is started
;*
;*  'ngens' = number of generations to grow
;*
;*  'length_orig', 'thick_orig' = original dimensions of a growth unit
;*
;*  'decrease_orig_length','decrease_orig_thick' = percentage decrease of the
;*      growth unit dimensions per growth unit
;*
;*  'xdim','ydim' = dimensions of the tree growth space
;*
;*  'xloc_orig','yloc_orig' = location in space where tree starts growing
;*****************************************************
axiom_orig = 0 ;gen1
ngens = 5 ;number of generations

;0 -> string1
string1 = string([1,3,0,4,1,5,0,6,0])

;1 -> string2
string2 = string([1,1])

  branch_angle = 50 ;degrees
  thick_orig = 30
  length_orig = 8
  decrease_orig_length = 0.02 ;percentage
  decrease_orig_thick = 0.04

  xdim = 600 & ydim = 600
  xloc_orig = 200 & yloc_orig = 0
  leaf_rad = 2. ;7 = 3.5 ;radius of circular leaf clumps
  flat = 0 ;flat leaves - change to '1' to activate selection
  circle = 1 ;0 ;circular leaf clumps

  output_file = '/Users/admin/Documents/L_system_string_'

  temp = systime()
  month = strmid(temp,4,3)
  day = strmid(temp, 8,2)
  date = strtrim(day)+month
  pos1 = strpos(output_file,'/', /Reverse_search)
  dir = strmid(output_file, 0, pos1+1)
  output_file_temp = dir+output_file +date+ '.dat
  wait = 0 ;number of seconds to pause for viewing between generations
;*****************************************************

;*****************************************************
;*  Write information to metadata file
;*****************************************************
  openw, 1, output_file+date+'.txt'
  printf, 1, 'Axiom:  ', strtrim(axiom_orig,1)
  printf, 1, 'Number of generations:  ', strtrim(ngens,1)
  printf, 1, 'Branch angle:  ', strtrim(branch_angle,1)
  printf, 1, 'Original segment length:  ', strtrim(length_orig,1)
  printf, 1, 'Original segment width:  ',strtrim(thick_orig,1)
  printf, 1, 'Percent decrease length:  ', strtrim(decrease_orig_length,1)
  printf, 1, 'Percent decrease thickness:  ', strtrim(decrease_orig_thick,1)
  printf, 1, 'Percent decrease = decrease * thickness (or length) * age'
  printf, 1, ''
  printf, 1, 'IDL strings:'
  printf, 1, strtrim(string1,1)
  printf, 1, strtrim(string2,1)
  printf, 1, ''
  temp = string(string1)
  opens = where(temp EQ 3)
  closes = where(temp EQ 4)
  opensr = where(temp EQ 5)
  closesr = where(temp EQ 6)
  IF opens(0) NE -1 THEN temp(opens) = '['
  IF closes(0) NE -1 THEN temp(closes) = ']'
  IF opensr(0) NE -1 THEN temp(opensr) = '('
  IF closesr(0) NE -1 THEN temp(closesr) = ')'
  temp = strtrim(temp, 1) ;remove leading spaces
  printf, 1, '0 -> ', temp
  temp = string(string2)
  opens = where(temp EQ 3)
  closes = where(temp EQ 4)
  opensr = where(temp EQ 5)
  closesr = where(temp EQ 6)
  IF opens(0) NE -1 THEN temp(opens) = '['
  IF closes(0) NE -1 THEN temp(closes) = ']'
  IF opensr(0) NE -1 THEN temp(opensr) = '('
  IF closesr(0) NE -1 THEN temp(closesr) = ')'
  temp = strtrim(temp, 1) ;remove leading spaces
  printf, 1, '1 -> ', temp
  printf, 1, ''

;*****************************************************
;* Generate L-System Tree
;*****************************************************
FOR k1=0, ngens-1 DO BEGIN
  ngen = k1+1
  axiom = axiom_orig

tree = 0
IF axiom EQ 0 THEN BEGIN ;string1
  tree = string1 ;gen2
  FOR n=2, ngen DO BEGIN ;iterate through remaining number of generations
  new_tree = 0
    FOR j=0, n_elements(tree)-1 DO BEGIN
      axiom = tree(j)
      IF axiom EQ 0 THEN BEGIN
        new_tree = [new_tree,string1]
      ENDIF ELSE BEGIN
        IF axiom EQ 1 THEN BEGIN
          new_tree = [new_tree, string2]
        ENDIF
        IF axiom EQ 3 OR axiom EQ 4 OR axiom EQ 5 OR axiom EQ 6 THEN BEGIN
          new_tree = [new_tree, axiom]
        ENDIF
      ENDELSE
    ENDFOR
    tree = new_tree(1:n_elements(new_tree)-1)
  ENDFOR
ENDIF ELSE BEGIN
IF axiom EQ 1 THEN BEGIN ;string2
  tree = string2 ;gen2
  FOR n=2, ngen DO BEGIN ;iterate through remaining number of generations
  new_tree = 0
    FOR j=0, n_elements(tree)-1 DO BEGIN
      axiom = tree(j)
      IF axiom EQ 0 THEN BEGIN
        new_tree = [new_tree,string1]
      ENDIF ELSE BEGIN
        IF axiom EQ 1 THEN BEGIN
          new_tree = [new_tree, string2]
        ENDIF
        IF (axiom EQ 3) OR (axiom EQ 4) OR axiom EQ 5 OR axiom EQ 6 THEN BEGIN
          new_tree = [new_tree, axiom]
        ENDIF
      ENDELSE
    ENDFOR
    tree = new_tree
  ENDFOR
ENDIF
ENDELSE

;*****************************************************
;* Write information to metadata file
;*****************************************************
  printf, 1, ''
  printf, 1, 'Tree, generation:  ', strtrim(k1)
  temp = string(tree)
  opens = where(temp EQ 3)
  closes = where(temp EQ 4)
  opensr = where(temp EQ 5)
  closesr = where(temp EQ 6)
  IF opens(0) NE -1 THEN temp(opens) = '['
  IF closes(0) NE -1 THEN temp(closes) = ']'
  IF opensr(0) NE -1 THEN temp(opensr) = '('
  IF closesr(0) NE -1 THEN temp(closesr) = ')'
  temp = strtrim(temp, 1) ;remove leading spaces
  printf, 1, temp

;*****************************************************
;* Render the tree
;*****************************************************
  xloc = xloc_orig & yloc = yloc_orig
  thick = thick_orig ;Final thickness after n generations
  length = length_orig ;Final branch length

  ;Decrease the starting thickness and length based on generation -
  ;   to create more 'reasistic' looking growth patterns
  ;  Final generation is unaffected
  pct_thick = thick_orig^(1./(ngens-k1))
  thick = pct_thick
  pct_length = length_orig^(1./(ngens-k1))
  length = pct_length
  decrease_length = decrease_orig_length^(1./(ngens-k1))
  decrease_thick = decrease_orig_thick^(1./(ngens-k1))
  decrease_thick = decrease_orig_thick
  decrease_length = decrease_orig_length

;Create arrays to store information about the tree - to be exported to LIDAR modeling
program
x = fltarr(xdim,ydim)
stem = x
branch = x
xcoord = x & ycoord = x
angle_map = x
```

```
slope = x
apial_node = x ;locations where leaves will occur
leaves = x & leaves(*,*) = 0
air = x

xloc_new = xloc & yloc_new = yloc
branch_start = fltarr(2,3000)
branch_age = intarr(3000)
branch_ang = fltarr(3000)
branch_diam_order = fltarr(3000)
branch_orig_ang = fltarr(3000)
branch_orig_ang(0) = branch_angle*!pi/180.
order = 0
ang = branch_angle * (!pi/180.)
temp_ang = !pi/2.
length = length_orig
branch_diam = 0
leaf_loc = [0.,0.]
thick_orig2 = thick
age = 0
FOR k=0, n_elements(tree)-1 DO BEGIN
  new_decrease_length = decrease_length*age
  new_decrease_thick = decrease_thick
  length = max([length - length*order*new_decrease_length,1])
  thick = max([thick - new_decrease_thick, 1])
  IF tree(k) EQ 0 OR tree(k) EQ 1 THEN BEGIN
    age = age+1
    xloc_new = xloc + length*cos(temp_ang)
    yloc_new = yloc + length*sin(temp_ang)
    xcoord(xloc_new, yloc_new) = xloc_new
    ycoord(xloc_new, yloc_new) = yloc_new
    ; Create stem between buds
    startx = min([xloc, xloc_new])
    endx = max([xloc, xloc_new])
    run = xloc_new - xloc
    rise = yloc_new - yloc
    starty = min([yloc, yloc_new])
    endy = max([yloc, yloc_new])

    IF (fix(endx) - fix(startx)) GT 0 THEN BEGIN ;run GT 0
       m = rise/run
       b = yloc - m*xloc
       pct_decrease_thick = new_decrease_thick/length
       FOR i = startx, endx DO BEGIN
          stem(i,m*i+b) = 1
          angle_map(i,m*i+b) = atan(rise/run)
          current_thickness = thick - thick*pct_decrease_thick*(i+1-xloc)
          branch_diam = max([current_thickness*abs(sin(angle_map(i,m*i+b))), 0.])

          ;Fill in branch thickness in y-direction
          FOR j = max([0,m*i+b-branch_diam/2.]), min([m*i+b+branch_diam/2.,ydim]) DO
BEGIN
            branch(i,j) = 1
            angle_map(i,j) = atan(((j-yloc)*pct_decrease_thick+rise)/run)
          ENDFOR
       ENDFOR
       IF (fix(yloc_new) - fix(yloc)) GT 0 THEN BEGIN ;run GT 0 AND rise GT 0
         FOR i = yloc, yloc_new DO BEGIN
            stem((i-b)/m,i) = 1
            current_thickness = thick - thick*pct_decrease_thick*(i+1-yloc)
            branch_diam = max([current_thickness*abs(cos(angle_map((i-b)/m,i))), 0.])
            ;Fill in branch thickness in x-direction
            FOR j = (i-b)/m-branch_diam/2., min([(i-b)/m+branch_diam/2.,xdim]) DO BEGIN
              branch(j,i) = 1
              angle_map(j,i) = atan(length/((j-xloc)*pct_decrease_thick+run));(rise+(i-
yloc)))
            ENDFOR
         ENDFOR
       ENDIF ;end if run GT 0 AND rise GT 0
```

```
      ENDIF ELSE BEGIN IF length GT 0 THEN BEGIN ;end if run GT 0; begin if rise GT 0
        stem(xloc, starty:endy)=1
        pct_decrease_thick = new_decrease_thick/length ;amount to decrease thickness per
growth length
        FOR i=starty, endy DO BEGIN
          current_thickness = thick - thick*pct_decrease_thick*(i+1-yloc)
          branch_diam = max([current_thickness*abs(cos(angle_map(xloc,i))),0.])
          FOR j=xloc-branch_diam/2., xloc+branch_diam/2. DO BEGIN
            branch(j,i) = 1
            IF j EQ xloc THEN angle_map(j,i) = !pi/2.
            IF j NE xloc THEN angle_map(j,i) = !pi/2.+atan((j-
xloc)*pct_decrease_thick/length)
          ENDFOR
        ENDFOR
        ENDIF
      ENDELSE
      xloc = xloc_new & yloc = yloc_new
      branch_age(order) = age
  ENDIF ;tree(m) eq 0 or 1

  IF (tree(k) EQ 3 OR tree(k) EQ 5) THEN BEGIN ;start a branch
    age = 0
    branch_diam_order(order) = branch_diam
    n1 = 2
    IF tree(k) EQ 5 then n1 = 1 ;Start a branch in Clockwise direction from current
heading
    branch_ang(order) = temp_ang
    temp_ang = temp_ang + ang*((-1)^n1)
    branch_start(*,order) = [xloc_new, yloc_new]
    order = order+1
    branch_age(order) = age
  ENDIF

  IF (tree(k) EQ 4 OR tree(k) EQ 6) THEN BEGIN ;end a branch
    order = order-1
    apial_node(xloc,yloc) = 1
    leaf_loc = [[leaf_loc], [xloc, yloc]]
    xloc = branch_start(0,order) & yloc = branch_start(1,order) ;return to base of branch
    age = branch_age(order)
    temp_ang = branch_ang(order)
    branch_diam = branch_diam_order(order)
  ENDIF
length = length_orig
thick = min([branch_diam,thick_orig2])
ENDFOR ;number of elements of tree

   WINDOW, 1, xsize=xdim, ysize=ydim, xpos=xdim+10, ypos=100
   TVSCL, branch

;*****************************************************
;* Add circular leaves to apial nodes
;*****************************************************
  ;trim leading zeros from leaf_loc
  leaf_loc = leaf_loc(*,1:n_elements(leaf_loc(0,*))-1)
IF circle eq 1 THEN BEGIN
  FOR m = 0, n_elements(leaf_loc(0,*))-1 DO BEGIN
    xm = leaf_loc(0,m)
    ym = leaf_loc(1,m)
    FOR n = max([0,xm-(leaf_rad)]), min([xm+(leaf_rad)-1,xdim-1]) DO BEGIN
       FOR p = max([0,ym-(leaf_rad)]), min([ym+(leaf_rad)-1,ydim-1]) DO BEGIN
          distance = sqrt((xm-n)^2 + (ym-p)^2)
          IF distance LE leaf_rad THEN leaves(n,p)=1
       ENDFOR
     ENDFOR
   ENDFOR
ENDIF

;*****************************************************
;* Add flat leaves to apial nodes
```

```
;*****************************************************
IF flat eq 1 THEN BEGIN
  FOR m = 0, n_elements(leaf_loc(0,*))-1 DO BEGIN
    xm = leaf_loc(0,m)
    ym = leaf_loc(1,m)
    FOR n = max([0,xm-(leaf_rad)]), min([xm+(leaf_rad)-1,xdim-1]) DO BEGIN
       p =  ym
       distance = sqrt((xm-n)^2 + (ym-p)^2)
       IF distance LT leaf_rad THEN leaves(n,p)=1
     ENDFOR
   ENDFOR
ENDIF

;*****************************************************
;* Create .gif image file of the tree
;*****************************************************
  pos = strpos(output_file, '.')
  short = STRMID(output_file, 0, pos)
  IF k1 LE 9 THEN output_file_gif = short + string(k1,"(I1)") + '.gif' $
  ELSE output_file_gif = short + string(k1,"(I2)") + '.gif'
  IF k1 LE 9 THEN output_file_gif2 = short + string(k1,"(I1)") + '_thick.gif' $
  ELSE output_file_gif2 = short + string(k1,"(I2)") + '_thick.gif'

  ;Invert the color scheme - make background white
  temp_stem = byte(stem+1)
  temp_stem(where(temp_stem EQ 2)) = 0
  temp_branch = byte(branch+1)
  temp_branch(where(temp_branch EQ 2)) = 0

  write_gif, dir+output_file_gif, temp_stem*255b
  write_gif, dir+output_file_gif2, temp_branch*255b

;*****************************************************
;* Display image of the tree on the screen
;*****************************************************
   WINDOW, 2, xsize=xdim, ysize=ydim, xpos=xdim+10, ypos=100
   TVSCL, branch+leaves
   wait, wait

ENDFOR ;k = number of generations

;*****************************************************
;* Create output files of the tree for use with LIDAR simulation
;*****************************************************
  pos = strpos(output_file, '.')
  short = STRMID(output_file, 0, pos)
  temp = systime()
  month = strmid(temp,4,3)
  day = strmid(temp, 8,2)
  date = strtrim(day)+month
  pos1 = strpos(output_file,'/', /Reverse_search)
  dir = strmid(output_file, 0, pos1+1)
  output_file_air = dir +'air_'+date+ '.dat'
  output_file_bark = dir+'bark_'+date+ '.dat'
  output_file_leaf = dir +'leaf_'+date+ '.dat'
  output_file_xcoord = dir +'bud_xcoord_'+date+ '.dat'
  output_file_ycoord = dir +'bud_ycoord_'+date+ '.dat'
  output_file_angle_map = dir +'angle_map_'+date+ '.dat'

  openw, 2, output_file_air
  air(where(leaves+branch EQ 0)) = 1
  writeu, 2, air
  close, 2

  openw, 2, output_file_bark
  writeu, 2, branch
  close, 2

  openw, 2, output_file_leaf
  writeu, 2, leaves
  close, 2

  openw, 2, output_file_xcoord
  writeu, 2, xcoord
  close, 2

  openw, 2, output_file_ycoord
  writeu, 2, ycoord
  close, 2

  openw, 2, output_file_angle_map
  writeu, 2, angle_map
  close, 2

close, 1
print, 'FINISHED'

END
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Abbot, R. I., Mulholland, J. D., Silverberg, E. C., & Shelus, P. J. (1973). Laser observations of the moon: Identification and construction of normal points for 1969–1971. *The Astronomical Journal, 78(8),* 784–793.

Abshire, J. B., McGarry, J. F., Pacini, L. K., Blair, J. B., & Elman, G. C. (1994). Laser altimetry simulator, version 3.0 user's guide.

Allen, W. A., & Richardson, A. J. (1968). Interaction of light with a plant canopy. *Journal of the Optical Society of America, 58(8),* 1023–1028.

Andersen, H.-E., McGaughey, R. J., & Reutebuch, S. E. (2005). Estimating forest canopy fuel parameters using LIDAR data. *Remote Sensing of Environment, 94,* 441–449.

Andersen, H.-E., McGaughey, R. J., Carson, W. W., Reutebuch, S. E., Mercer, B., & Allan, J. (2003). A comparison of forest canopy models derived from LIDAR and INSAR data in a Pacific Northwest conifer forest. *International Archives of Photogrammetry and Remote Sensing, 34(3),* 211–217.

Andrews, L. C. & Phillips, R. L. (2005). *Laser beam propagation through random media* (2nd ed.). Bellingham, WA: SPIE Press.

Baltsavias, E. (1999a). Airborne laser scanning: existing systems and firms and other resources. *ISPRS Journal of Photogrammetry and Remote Sensing, 54,* 164–198.

Baltsavias, E. (1999b). Airborne laser scanning: Basic relations and formulas. *ISPRS Journal of Photogrammetry and Remote Sensing, 54,* 199–214.

Baltsavias, E. (2008). *Introduction to airborne lidar and physical principles of lidar technology* [PowerPoint slides]. Retrieved September 20, 2009, from Lecture Notes Online Web site: http://home.iitk.ac.in/~blohani/LiDARSchool2008/downloads.html

Brandtberg, T., Warner, T. A., Landenberger, R. E. & McGraw, J. B. (2003). Detection and analysis of individual leaf-off tree crowns in small footprint, high sampling density lidar data from the eastern deciduous forest in North America. *Remote Sensing of Environment, 85,* 290–303.

Clark., R. N., Swayze, G. A., Gallagher, A. J., King, T. V. V., & Calvin, W. M. (1993). *The U. S. Geological Survey, Digital Spectral Library: Version 1: 0.2 to 3.0 microns,* (U.S. Geological Survey Open File Report 93–592). Retrieved April 18, 2009, from http://speclab.cr.usgs.gov/spectral.lib04/clark1993/spectral_lib.html

Chomsky, N. (1956). Three models for the description of language. *IRE Transcations on Information Theory, 2*, 113–124.

Degnan, J. J. (2002). Photon-counting multikilohertz microlaser altimeters for airborne and spaceborne topographic measurements. *Journal of Geodynamics, 34,* 503–549.

Dubayah, R. O., & Drake, J. B. (2000). Lidar remote sensing for forestry. *Journal of Forestry, 98(6),* 44–46.

Foley, J., van Dam, A., Feiner, S., & Hughes, J. (1990). *Computer Graphics, Principles and Practices* (2nd ed.). Reading, Massachusetts: Addison-Wesley Publishing Company.

Hall, R. N., Fenner, G. E., Kingsley, J. D., Solyts, T. J., & Carlson, R. O. (1962). Coherent light emission form GaAs junctions. *Physical Review Letters, 9(9),* 366–368.

Hammond, A. L. (1970). Laser ranging: Measuring the moon’s distance. *Science, 170(3964),* 1289–1290.

Harding, D. J., Abshire, J. B., Dabney, P. W., Scambos, A. A., Shuman, C. A., & Sun, X. (2007). The swath imaging multi-polarization photon-counting LIDAR (SIMPL): A technology demonstration for space-based laser altimeter swath mapping. *Proceedings from the NASA Science Technology Conference 2007.* Retrieved September 1, 2009, from http://esto.nasa.gov/conferences/nstc2007/papers/Harding_David_B7P4_NSTC-07–0150.pdf

Harding, D. J., Leftsky, M. A., & Blair, J. B. (2001). Laser altimeter canopy height profiles – Methods and validation for closed-canopy, broadleaf forests. *Remote Sensing of Environment, 76,* 283–297.

Jacquemoud, S., & Baret, F. (1990). PROSPECT: a model of leaf optical properties spectra. *Remote Sensing of Environment, 34*, 75–91.

Javan, A., Bennett Jr., W. R., & Herriott, D. R. (1961). Population inversion and continuous optical maser oscillation in a gas discharge containing a He-Ne mixture. *Physical Review Letters, 6(3),* 106–110.

Krabill, W. B., Collins, J. G., Link, L. E., Swift, R. N., Butler & R. M. (1984). Airborne laser topographic mapping results. *Photogrammetric Engineering and Remote Sensing, 50(6),* 685–694.

Kuusk, A., & Nilson, T. (2000). A directional multispectral forest reflectance model. *Remote Sensing of Environment, 72,* 244–252.

Lucke, R., L. & Rickard, L. J. (2002). Photon-limited synthetic-aperture imaging for planet surface studies. *Applied Optics, 41(24),* 5084–5095.

Maiman, T. H. (1960). Stimulated optical radiation in ruby. *Nature, 187(4736),* 493–494.

Mallet, C. & Bretar, F. (2008). Full-waveform topographic lidar. *ISPRS Journal of Photogrammetry and Remote Sensing, 64,* 1–16.

Marino, R. M., Richardson, J., Garnier, R., Ireland, D., Bickmeier, L., Siracusa, C., et. al. Photon-counting LIDAR for aerosol detection and 3-D imaging. *Proceedings of SPIE, 7323.*

Measures, R. M. (1992). *Laser remote sensing: Fundamentals and applications* (reprint ed.). Malabar, FL: Krieger Publishing Company.

Mitchell, R. (2008). *Airborne oceanographic lidar history.* Retrieved June 20, 2008, from http://aol.wff.nasa.gov/

Mõttus, M. (2007). Photon recollision probability in discrete crown canopies. *Remote Sensing of Environment, 110,* 176–185.

NASA (1997). [Graph of representative SLA-02 waveforms 1997]. *Shuttle Laser Altimeter-02.* Retrieved September 10, 2009 from http://core2.gsfc.nasa.gov/lapf/sla02/results.html.

North, P. R. J. (1996). Three-dimensional forest light interaction model using a monte carlo method. *IEEE Transactions on Geoscience and Remote Sensing, 34(4),* 946–956.

Optech Incorporated (2004). *ALTM 3100: The next level of performance.*

Parker, G. G., Lefsky, M. A., & Harding, D. J. (2001). Light transmittance in forest canopies determined using airborne laser altimetry and in-canopy quantum measurements. *Remote Sensing of Environment, 76,* 298–309.

Prusinkiewicz, P. & Lindenmayer, A. (2004). *The Algorithmic Beauty of Plants.* New York: Springer-Verlag.

Pfeifer, N. (2008). *Principle and exploitation of full waveform ALS* [PowerPoint slides]. Retrieved September 8, 2008, from http://home.iitk.ac.in/~blohani/LiDARSchool2008/downloads.html

Rautiainen, M., & Stenberg, P. (2005). Application of photon recollision probability in coniferous canopy reflectance simulations. *Remote Sensing of Environment, 96,* 98–107.

de Reffye, P., Edelin, C., Jaeger, M., & Puech, C. (1988). Plant models faithful to botanical structure and development. *Computer Graphics, 22,* 151–158.

Riegl Laser Measurement Systems GmbH (2007). *Airborne data processing software RiAnalyze 560 for full waveform analysis.* Retrieved September 10, 2009, from http://www.rieglusa.com/products/airborne/lms-q560/index.shtml

Schawlow, A. L. & Townes, C. H. (1958). Infrared and optical masers. *Physical Review, 112(6),* 1940–1949.

Shabanov, N. V., Huang, D., Knjazikhin, Y., Dickinson, R. E., & Myneni, R. B. (2007). Stochastic radiative transfer model for mixture of discontinuous vegetation canopies. *Journal of Quantitative Spectroscopy & Radiative Transfer, 107,* 236–262.

Shan, J. & Toth, C. K. (Eds.). (2009). *Topographic laser ranging and scanning: principles and processing.* Boca Raton, FL: CRC Press.

Silverberg, E. C., & Currie, D. G. (1972). A description of the lunar ranging station at McDonald Observatory. *Proceedings from Open Meeting of Working Groups of the 14th Plenary Meetings of COSPAR: Space Research XII, 1,* 219–233.

Smolander, S., & Stenberg, P. (2005). Simple parameterizations of the radiation budget of uniform broadleaved and coniferous canopies. *Remote Sensing of Environment, 94,* 355–363.

Stenberg, P. (2007). Simple analytical formula for calculating average photon recollision probability in vegetation canopies. *Remote Sensing of Environment, 109,* 221–224.

Stokes, G. G. (1862). On the intensity of light reflected from or transmitted through a pile of plates. *Proceedings of the Royal Society of London, 11,* 545–556.

Velodyne Acoustics, Inc. (2007). *Velodyne's HLD-64E: A high definition lidar sensor for 3-D applications.*

Wang, Y., Buermann, W., Stenberg, P., Smolander, H., Ha”me, T., Tian, Y., et al. (2003). A new parameterization of canopy spectral response to incident solar radiation: case study with hyperspectral data from pine dominant forest. *Remote Sensing of Environment, 85,* 304–315.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Carlos F. Borges
   Naval Postgraduate School
   Monterey, California

4. Richard C. Olsen
   Naval Postgraduate School
   Monterey, California