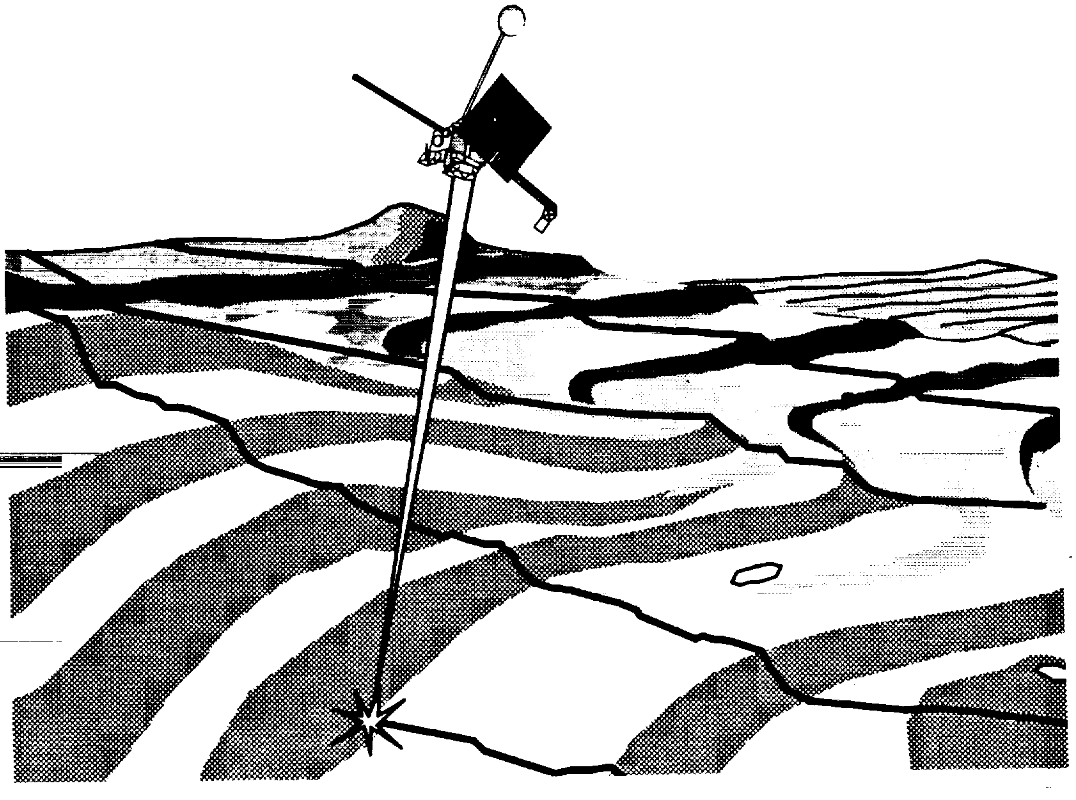
NASA Technical Memorandum 104588

Laser Altimetry Simulator, Version 3.0 User's Guide

James B. Abshire, Jan F. McGarry, Linda K. Pacini,
J. Bryan Blair, and Gregory C. Elman



January 1994

(NASA-TM-104588) LASER ALTIMETRY
SIMULATOR. VERSION 3.0: USER'S
GUIDE (NASA) 71 p

N94-26152

Unclas

G3/46 0208974

NASA Technical Memorandum 104588

Laser Altimetry Simulator, Version 3.0 User's Guide

James B. Abshire, Jan F. McGarry, Linda K. Pacini,
and J. Bryan Blair
Goddard Space Flight Center
Greenbelt, Maryland

Gregory C. Elman
Science Systems and Applications, Inc.
Lanham, Maryland

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

1994

CONTENTS

CONTENTS (cont.)

ACKNOWLEDGEMENTS

We are grateful to Ms. Debbie Sallitt of Science Systems and Applications, Inc., for her expert production work on this document, and to Mr. Hans Bauman of Ressler Associates, Inc., for developing and performing the simulator's verification tests. We are also grateful to Mr. Gregory F. Smith of the EOS Chemistry and Special Flight Project for supporting this work.

1.0 Introduction

Pulsed laser altimeters estimate the range to a terrain surface by measuring the round trip time-of-flight of a laser signal[1]. The measurement geometry is shown in Figure 1.1. Although different transmitted laser signals can be used[2], most direct detection laser altimeters transmit a single Q-switched laser pulse with a nearly Gaussian-shaped intensity profile. The laser altimeter receiver must detect the terrain reflected laser pulse in the presence of optical and electronic noise. Because the height variations of the terrain within the laser footprint spread the reflected pulse[3], the receiver must also estimate the "center" of the received pulse in order to accurately measure the range.

In airborne laser altimeters[4], the range is usually measured in two parts. The coarse range is typically measured as the time between the leading edges of the transmitted and received laser pulses. A fine range correction is then computed from a sampled version of the received optical waveform, and the correction is added to the coarse range to produce the final range estimate. In some planetary laser altimeter designs such as MOLA[5], the instrument's power is very constrained and only the coarse range is measured.[6,7] For these instruments, the primary task of the receiver is to maximize the probability of a successful measurement.[5,6]

The accuracy of the timing (or ranging) performance is governed by both the altimeter's design, its pointing angle, and the characteristics of the terrain surface.[3,8] Relevant laser transmitter parameter's include the laser energy, pulse width and beam divergence. Important parameters of the measurement geometry and terrain surface include the altimeter's altitude and pointing angle, and the terrain's surface slope, roughness and reflectivity. Relevant parameters in the receiver include the receiver telescope area, the detector's bandwidth, gain, noise and the design and sampling rate of the signal processor[9].

For some special cases, such as for flat or uniformly sloped terrain, closed form expressions can be given for the laser altimeter's detection statistics and timing performance.[8] However, when the surface topography is more complex, it is difficult to describe the altimeter's receiver signal shape, which is required to predict the receiver performance. As a consequence, for many realistic measurement scenarios it is difficult to develop a high performance altimeter receiver and to analyze its performance.

1.1 Overview

The Laser Altimetry Simulator was developed as a first-generation tool to explore the relationship between the altimeter's design, performance, and the terrain characteristics. It calculates the altimeter performance in a simplified two-dimensional (height versus along track distance) measurement geometry. As a complementary approach to the analytical calculations, it can produce performance estimates over a variety of conditions, including those where the theory

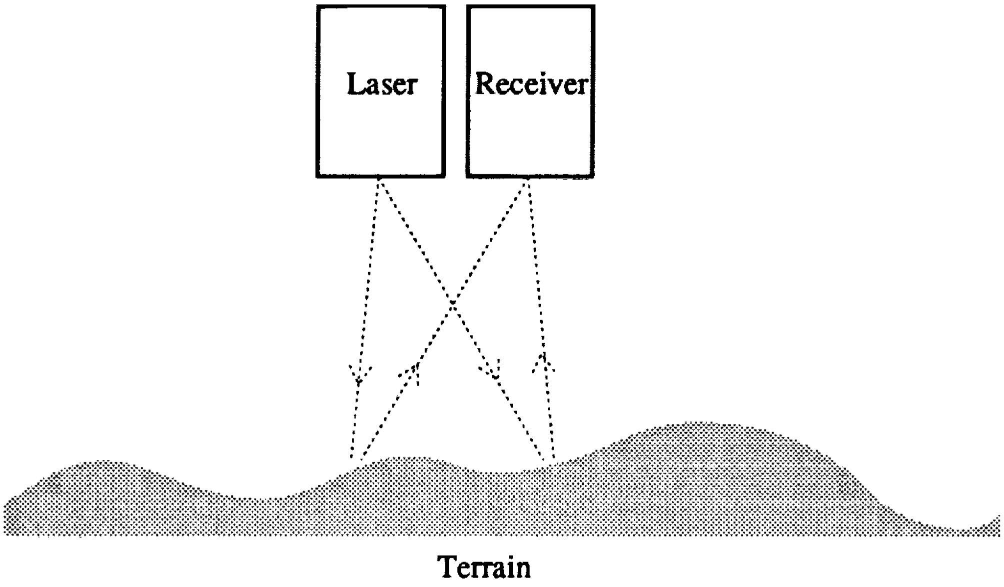


Figure 1.1. Laser Altimeter Measurement Geometry.

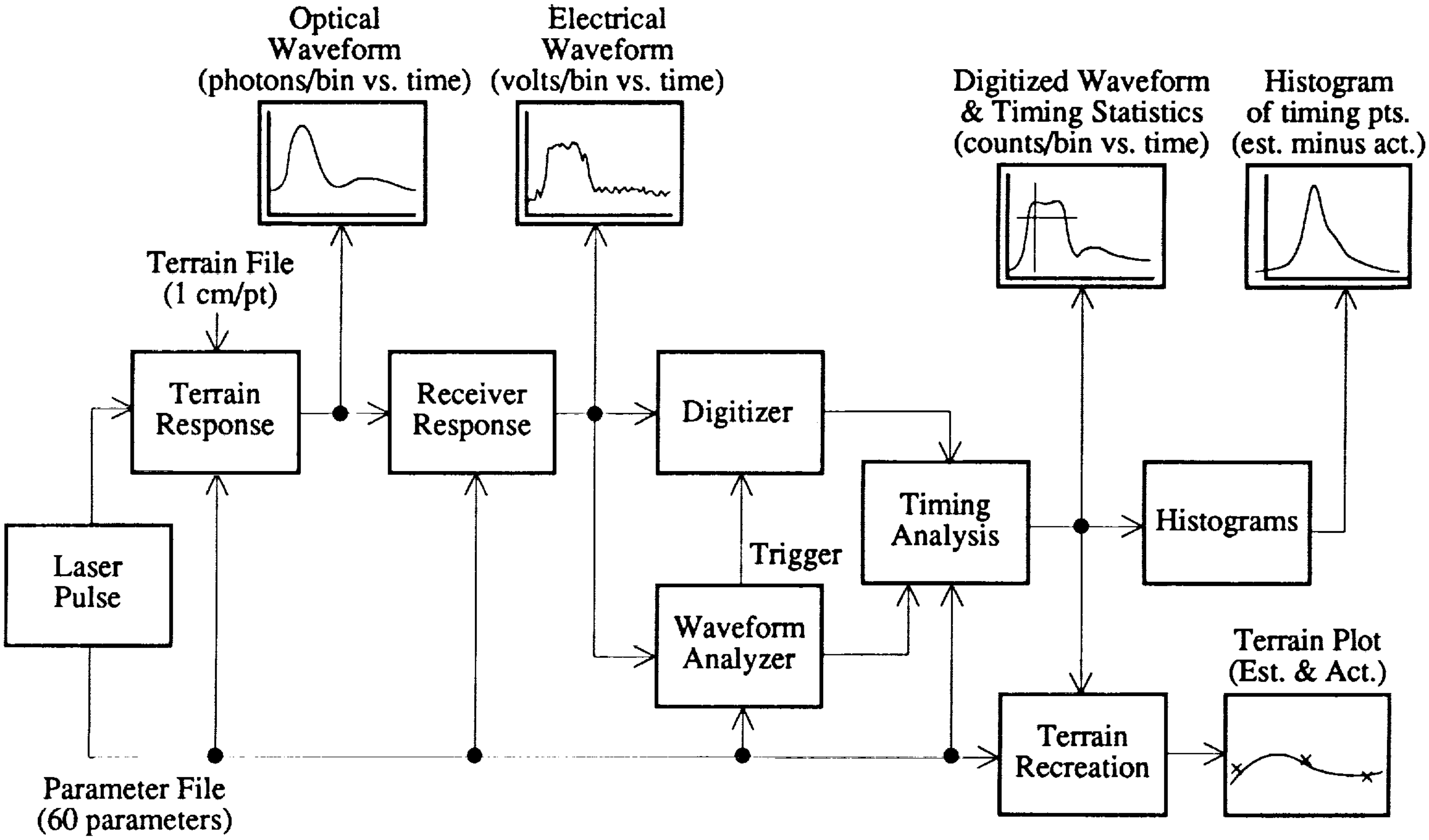


Figure 1.2. Laser Altimetry Simulator Block Diagram.

is intractable. The simulator can also calculate and plot the altimeter's signal and noise at the various stages within the altimeter receiver, which can yield insight into the altimeter's operation. Although this simulator was developed for the GLAS mission, it is flexible, and it can be used to analyze the performance of a variety of airborne and spaceborne laser altimeters.

In prior work, Abshire and McGarry[10] developed a simpler Monte-Carlo simulator to calculate the timing performance of short pulse two-color laser reflections[11-13] from a model of the ocean surface. It was used as a guide for this work. However, this simulator is more complete and encompasses the entire laser pulse and detector propagation paths.

This guide is intended for users who have working knowledge of how a laser altimeter operates as well as a working knowledge of UNIX, the NCAR graphics package and the SUN sparcstation.

1.2 Simulator Design

The simulator operates by calculating the laser altimeter's optical intensity waveform, as it propagates to and from the terrain surface and, after detection, through the altimeter's receiver. The simulator operates in two dimensions (along-track distance and height), and operates with time quantized into 100 psec bins, which correspond to 1.5 cm in range. It calculates the optical signal path in two dimensions, (height versus along-track distance) and uses a finite number of rays to approximate the laser's optical wavefront. A simplified flow diagram is shown in Figure 1.2.

The simulator does not include the effects of atmospheric refraction. The laser transmitter's wavelength, divergence angle and tilt angle of the altimeter are specified, along with its height above the terrain surface and the along-track velocity. The terrain surface profile can be specified flat, tilted or have a predefined height profile. The terrain surface is assumed to be a diffuse reflector, and its reflectivity and height can be specified for every centimeter of along-track distance.

The simulator's receiver includes a telescope, optical bandpass filter, either a photomultiplier or avalanche photodiode optical detector, a low pass filter with a raised cosine impulse response, a timing discriminator, a time interval unit and a waveform digitizer. The parameters of the optical detectors are specified in the parameter file, along with the impulse response time of the lowpass filter. The sampling rate, number of bits, and voltage scaling of the waveform digitizer are also specified.

The receiver waveform and coarse and fine timing estimates are calculated independently for each laser firing. The threshold setting of the receiver's threshold detector is calculated from part of the received waveform which contains only noise. The coarse range is calculated as the time interval reading between the laser firing and the receiver's first threshold crossing time. Several possible fine range corrections are calculated by using the digitized waveform. The receiver's fine timing estimators include 50% risetime, and the midpoint, center of area, mean and peak of the received waveform.

1.3 Operation

Typically, the simulator is used to calculate the altimeter's measurement response to a sequence of laser firings. Each laser shot is simulated as an independent event. For each laser firing, the simulator follows the following steps:

a). The simulator calculates the optical intensity waveform as its leaves the laser transmitter. The optical signal has a specified energy, angular width and angular pointing offset from nadir. The transmit beam's intensity and far-field angle are Gaussian. As the laser signal leaves the transmitter, it starts the receiver time interval unit used to measure the coarse range delay.

b). The laser signal propagates to the terrain surface. The simulator divides the transmitted beam into a finite number of rays in along-track angle. The reflected intensity and range delay are calculated independently for each ray. The simulator calculates the laser pulse propagation to the surface and the terrain reflection in the along-track distance and height dimensions.

c) The simulator calculates the terrain surface interaction. It does this by projecting the altimeter's laser beam in a line which is parallel to the along-track altimeter motion. In doing so, it ignores any cross-track terrain height variations. The terrain surface is assumed to be Lambertian reflector, with a height and diffuse reflectivity specified for each along-track point . The height and diffuse reflectivity can be specified independently for each location in the surface profile.

d) The terrain scattered signal collected at the receiver is calculated. The calculations are based on 3-dimensional diffuse scattering from each terrain element and a 3-dimensional receiver telescope. Solar illumination is also scattered by the terrain back to the receiver. The range delays for each transmitted ray are calculated. The reflected laser pulse from each reflected ray are summed with their appropriate range delay, producing the received optical waveform. When added with the background light, this produces the optical intensity waveform at the detector surface.

e). The simulator uses a Monte Carlo method to calculate the detector's output signal. The user can select either a Silicon Avalanche photodiode (Si APD) or a photomultiplier (PMT) detector. For the Si APD, the output detector statistics for both signal plus background and the background only are assumed to have a Gaussian distribution. For the PMT detector, the statistics have a Poisson distribution. Both detector models produce an electrical output waveform (voltage versus time) for each 100 psec time bin.

f). The detector's output waveform is filtered with an electrical lowpass filter. The filter's impulse response is modeled as a raised cosine. As long as the filter's impulse response is longer than the laser pulse width, the filter's output is a smoothed version of the input waveform.

g). The filter's output is sent to the receiver's threshold detector. If any voltage in waveform exceeds the receiver threshold, the received pulse is detected. This stops the time interval measurement, yielding the coarse range estimate. It also starts the waveform digitize used to compute the fine correction estimate. If for that laser firing the entire waveform remains below threshold, the firing is registered as a missed detection. The present version of the simulator has an ideal (no false alarm) threshold setting algorithm. However, future versions will incorporate more realistic threshold setting algorithms.

h) For each detected pulse, the simulator calculates the fine ranging correction by using the waveform digitizer data. Since the time interval unit triggers on the received pulse's leading edge, it always triggers early and underestimates the true range. The waveform data is used to compute various estimates of the center of the pulse to correct the coarse timing estimate. The pulse timing estimators include 50% risetime, peak, mid-point center of area, and pulse mean, as shown in Figure 1.3. The pulse area, which is proportional to received pulse energy, is also calculated.

1.4 Spacecraft Velocity

If the altimeter's velocity = 0, then the altimeter will not move along track between laser firings. The laser will illuminate the same surface profile on all shots. This mode is useful for calculating timing and waveform statistics at a specified point in the terrain profile.

If the altimeter's velocity > 0, then the altimeter "moves" along-track between laser shots. If the along-track distance moved between shots exceeds the laser spot diameter, then a new terrain surface will be illuminated with each laser firing. This is a typical mode of operation, since it allows the altimeter's performance to be calculated for a pass over given terrain.

1.5 Waveforms and Summary Statistics

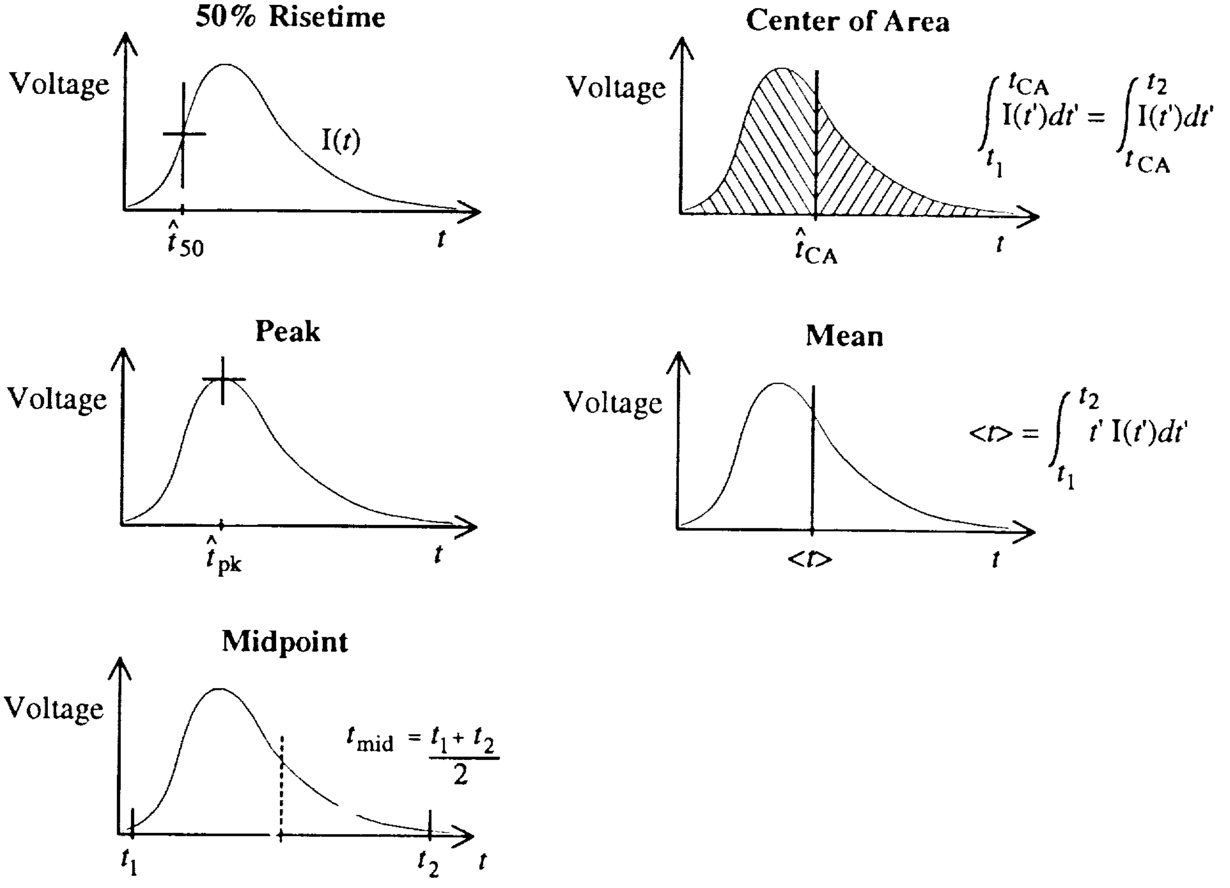
For each laser firing, the simulator calculates waveforms at several locations in the altimeter receiver. These include at the detector surface (photons vs. time), after the receiver electrical filter (volts vs. time), and after the altimeter waveform digitizer (counts vs. time). These waveforms can be plotted onto the screen or hardcopied.

Once a set of laser firings have been simulated, the results can be used to calculate the statistics of the altimeter's performance. These are computed by accumulating histograms of the desired timing or detection parameters. The histograms can be plotted and their mean and standard deviations can be calculated. Additionally, the detection and false alarm probabilities can be calculated.

1.6 Simulator Parameters

Most simulator settings can be changed with the parameter list. Those related to the altimeter instrument include the laser wavelength, laser energy, pulse width, transmitter beamwidth and off-axis pointing angle. Those related to the surface include diffuse reflectivity, background light illumination level and terrain height profile. Terrain profiles can be selected to be deterministic, including square waves and ramps with given slopes, or the terrain profile can be input as a height vs. distance data file. This feature is useful when calculating altimeter signals reflected from terrains which have been previously profiled with airborne altimeters.

Future versions of the simulator may include two different types of random terrain profiles. The first would permit adding a random roughness component (with a specified rms value) to a deterministic terrain profile. The power spectra (power versus along-track wavelength) of the random component can be specified by randomizing the profile's phase term and calculating the profile as an inverse Fourier transform. A completely random terrain profile, with a specified power spectrum and rms value, can also be calculated with this approach.


50% Risetime

Center of Area

Peak

Mean

Midpoint

Figure 1.3. Waveform Timing Estimators.

1.7 References

1. J.L. Bufton, "Laser altimetry measurements from aircraft and spacecraft," Proc. IEEE, Vol 77, 463 (1989).
2. X. Sun, J.B. Abshire and F.M. Davidson, "Design and performance of a multishot laser altimeter," Applied Optics, Vol. 32, 4578 (1993).
3. C.S. Gardner, "Target signatures for laser altimeters: an analysis," Applied Optics, Vol. 21, 448 (1982).
4. J.L. Bufton, J.B. Garvin, J.F. Cavanaugh, L. Ramos-Izquierdo, T.D. Clem and W.B. Krabill, "Airborne lidar for profiling of surface topography," Optical Engineering, Vol. 30, 72 (1991).
5. M.T. Zuber, D.E. Smith, S.C. Solomon, D.O. Muhleman, J.W. Head, J.B. Garvin, J.B. Abshire and J.L. Bufton, "The Mars Observer Laser Altimeter Investigation, " Journal of Geophysical Research,Vol. 97, 7781 (1992).
6. J.B. Abshire, S.S. Manizade, W. H. Schaefer, R.K. Zimmerman, J.S. Chitwood and J.C. Caldwell, "Design and performance of the receiver for the Mars Observer Laser Altimeter," Technical Digest, Conference on Lasers and Electro-Optics (CLEO'91), paper CFI4, Optical Society of America, Baltimore MD, May 1991.
7. J.F. McGarry, L.K. Pacini, J.B. Abshire, and J.B. Blair, "Design and performance of an autonomous tracking system for the Mars Observer Laser Altimeter receiver," Technical Digest, Conference on Lasers and Electro-Optics (CLEO'91), paper CThR27, Optical Society of America, Baltimore MD, May 1991.
8. C.S. Gardner, " Ranging Performance of Satellite Laser Altimeters," IEEE Transaction on Geoscience and Remote Sensing, Vol. 30, 1061 (1992).
9. X. Sun, F. M. Davidson, L. Boutsikaris and J.B. Abshire, "Receiver Characteristics of Laser Altimeters with Avalanche Photodiodes," IEEE Transactions on Aerospace and Electronic Systems, Vol. 28, 1 (1992).
10. J.B. Abshire and J.F. McGarry, "Estimating the Arrival Times of Photon-Limited Pulses in the Presence of Shot and Speckle Noise," J. Opt. Soc. Am. A, Vol. 4, 1080 (1987).
11. B.M. Tsai and C.S. Gardner, "Remote Sensing of Sea State Using Laser Altimeters", Applied Optics, Vol. 21, 3932 (1982).
12. J.B. Abshire and J.E. Kalshoven Jr., "Multicolor Laser Altimeter for Barometric Measurements over the Ocean: Experimental," Applied Optics, Vol. 22, 2578 (1983).
13. J.B. Abshire, J. F. McGarry, R.S. Chabot, and H.E. Rowe, "Airborne Measurements of Atmospheric Pressure with a Two-Color Streak Camera-Based Laser Altimeter," Proc. Conf. on Lasers and Electro-Optics (CLEO'85), Paper PD-6, Baltimore MD (May 1985).

2.0 Examples of Simulator Results

The outputs from the simulator for four sample terrains illustrate the simulator's operation. The examples "fire" a single laser pulse over four types of terrain for an altimeter with the nominal parameters. The four types of returns pulses for these examples are listed below.

FIGURE	WAVEFORM	TERRAIN PROFILE
2.1	Impulse	Flat, linear terrain.
2.2	Gaussian	10° sloped, linear terrain.
2.3	Symmetrical pulses	1 m single stepped, beam centered 50%. (35.25 m plateau)
2.4	Asymmetrical pulses	1 m single stepped, beam centered 25%. (17.625 m plateau)

These figures show five different graphs for each type of terrain. The text in parenthesis indicates the location of the waveform in the simulator. The graph types include:

a) Along-track terrain file.
b) SPACE_TIME subroutine output waveform (at the detector's surface).
c) RECEIVER subroutine output waveform (after the receiver electrical filter).
d) DIGITIZE subroutine output waveform (after the waveform digitizer).
e-i) Timing histograms.

For these examples, the simulator was run using its nominal parameters with the along-track velocity equal to zero.

2.1 Slope and Terrain

A set of uniform terrain slopes of 0, 1, 2, and 3 degrees was also used to test the simulator's statistical performance. One hundred laser shots were used for each terrain slope at transmit laser energies of 100, 50, 25 and 12.5 mJ. The performance of each of the five fine timing estimators was calculated and plotted versus energy level. Figure 2.5 shows the Mean of the five fine timing estimators and Figure 2.6 shows RMS jitter of the fine timing estimator plotted versus energy level. Each set of figures show the fine timing estimators for each of the four terrain profiles.

2.2 Ice Terrain

A similar test to the slope and terrain test was performed using ice terrain data. However, here the terrain file included a terrain segment from each of five classes of ice roughness. These are illustrated in Figure 2.7. The ice roughness classes are defined in Section 4.6.3. One hundred laser shots were simulated for each ice roughness class and the laser energies were 100, 50, 25 and 12.5 mJ. The Mean and RMS jitter of the five fine timing estimators are plotted versus signal level in Figure 2.8 and Figure 2.9. There is a plot for each of the terrain classes.

FIGURES 2.1 (A-D)

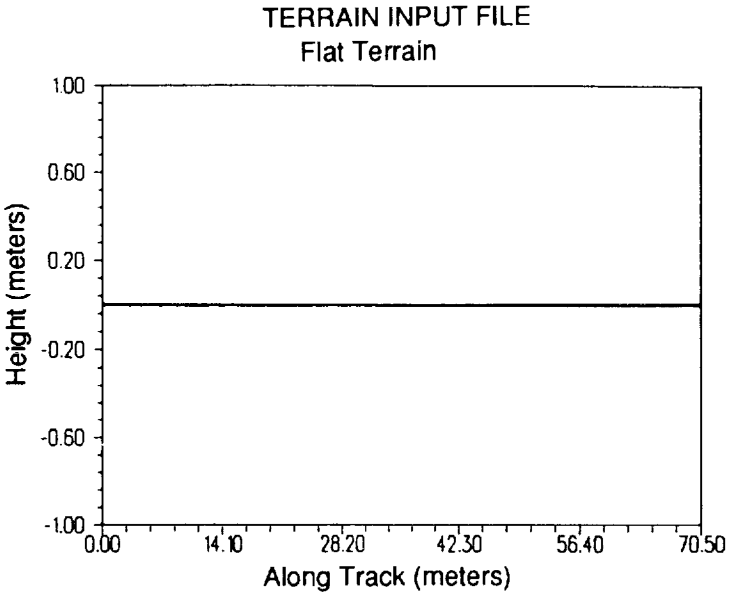

TERRAIN INPUT FILE
Flat Terrain

Figure 2.1A

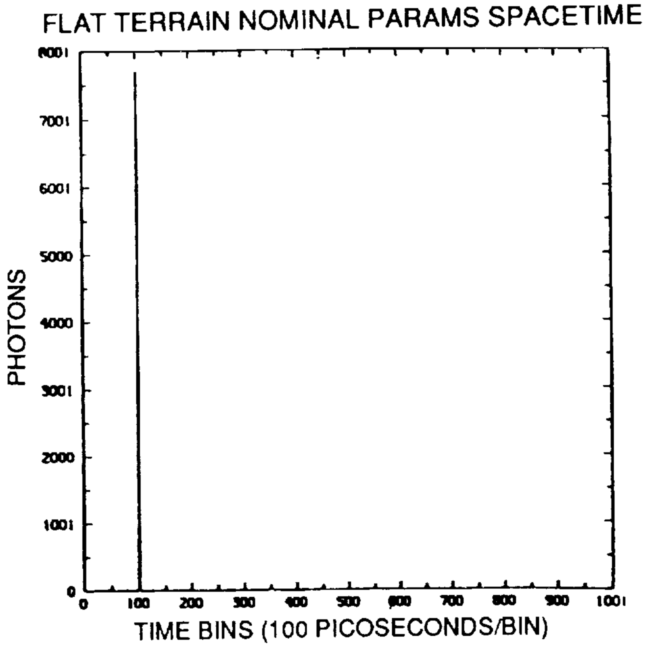

FLAT TERRAIN NOMINAL PARAMS SPACETIME

Figure 2.1B

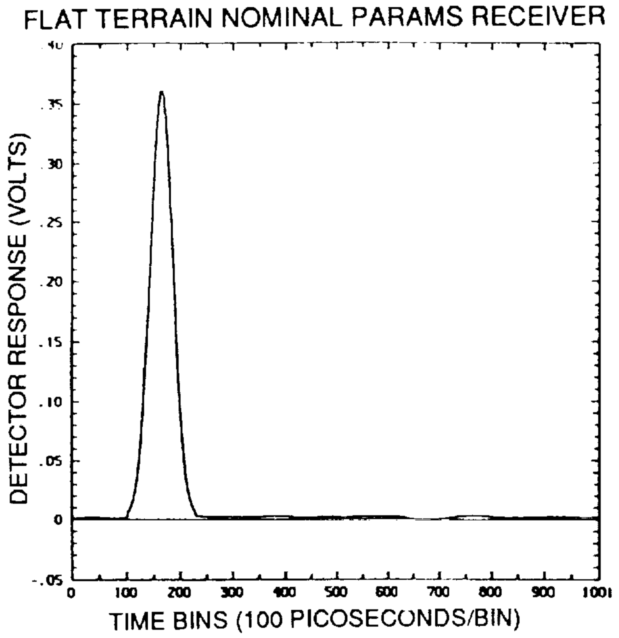

FLAT TERRAIN NOMINAL PARAMS RECEIVER

Figure 2.1C

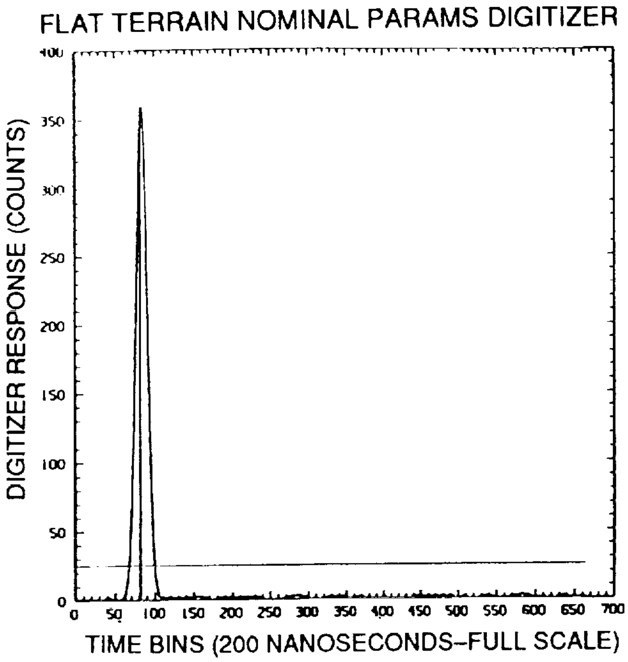

FLAT TERRAIN NOMINAL PARAMS DIGITIZER

Figure 2.1D

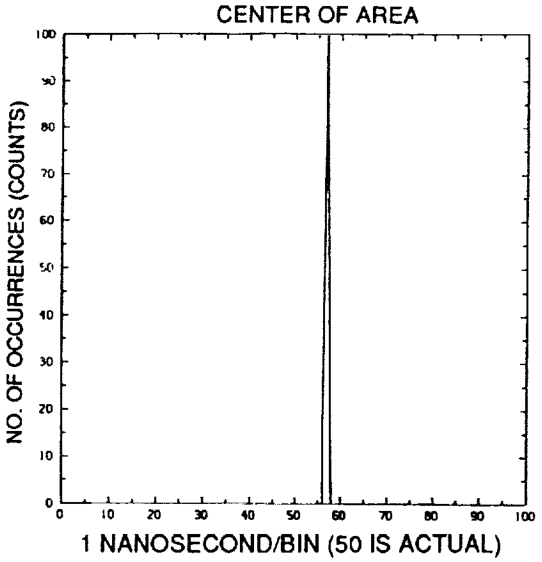


Figure 2.1E

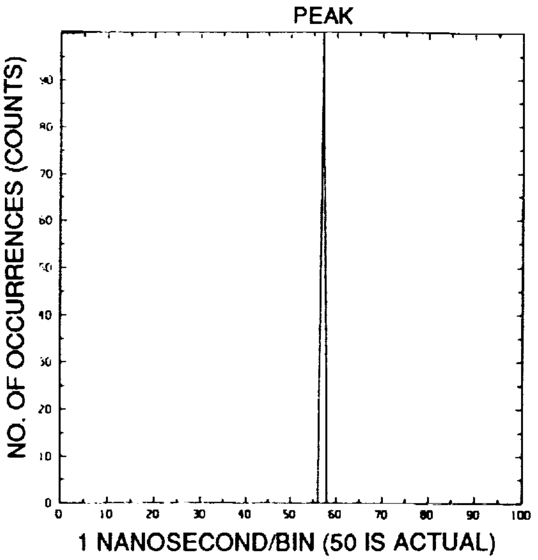


Figure 2.1F

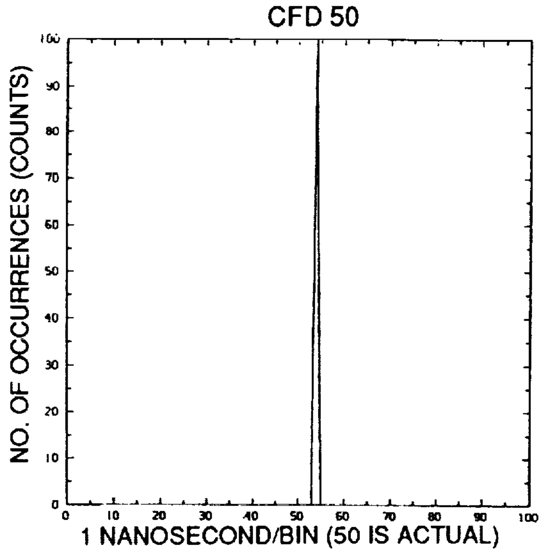


Figure 2.1G

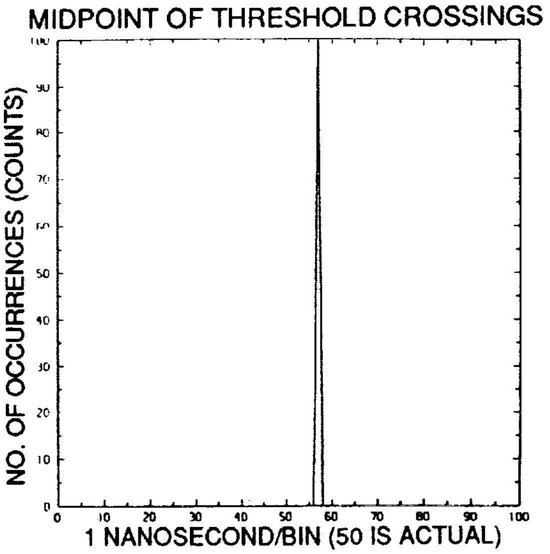


Figure 2.1H

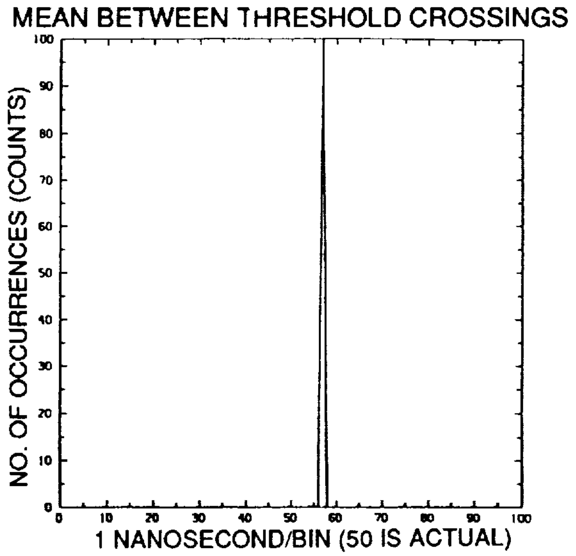


Figure 2.1I

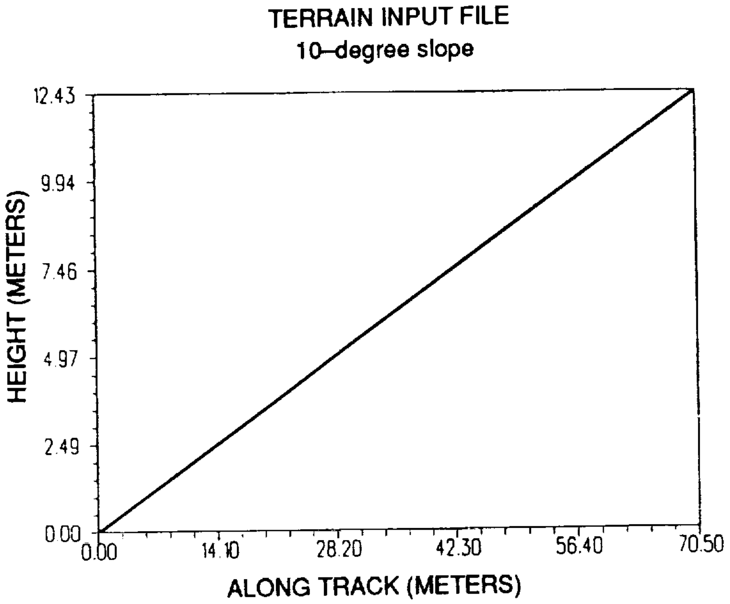

TERRAIN INPUT FILE
10–degree slope

Figure 2.2A

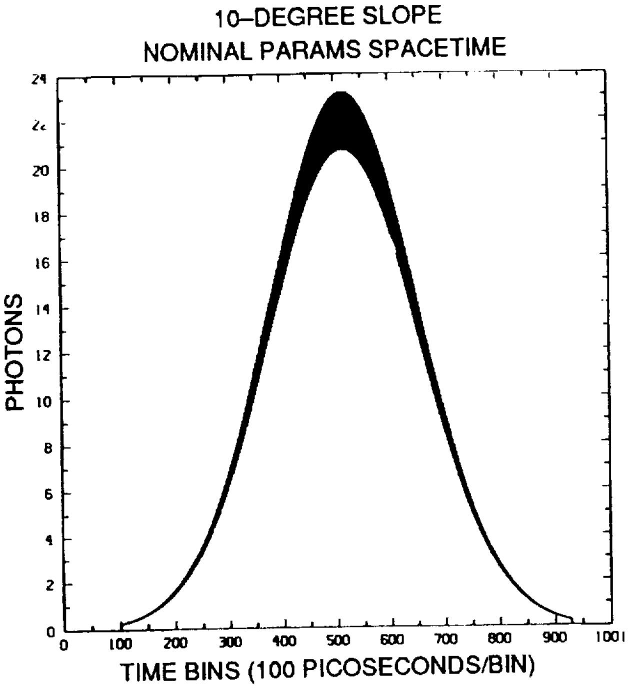

10–DEGREE SLOPE
NOMINAL PARAMS SPACETIME

Figure 2.2B

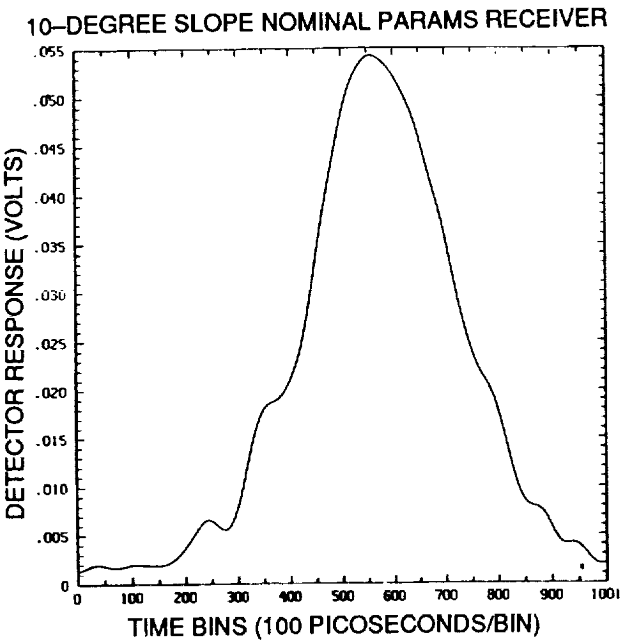

10–DEGREE SLOPE NOMINAL PARAMS RECEIVER

Figure 2.2C

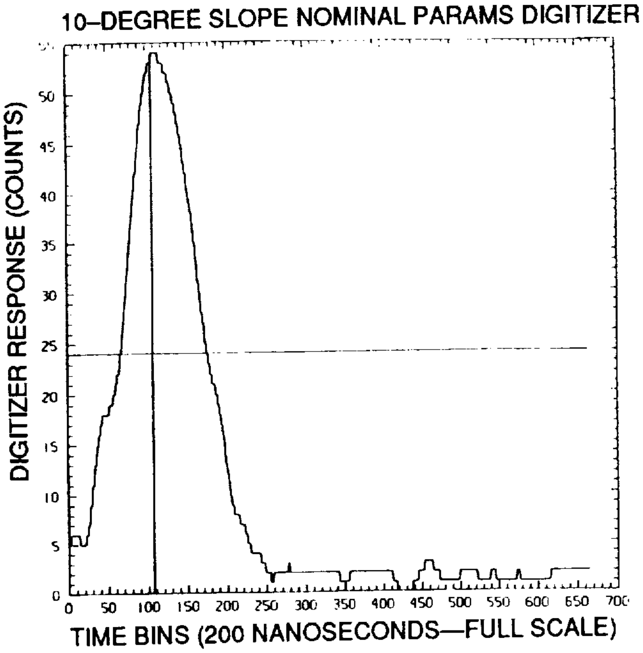

10–DEGREE SLOPE NOMINAL PARAMS DIGITIZER

Figure 2.2D

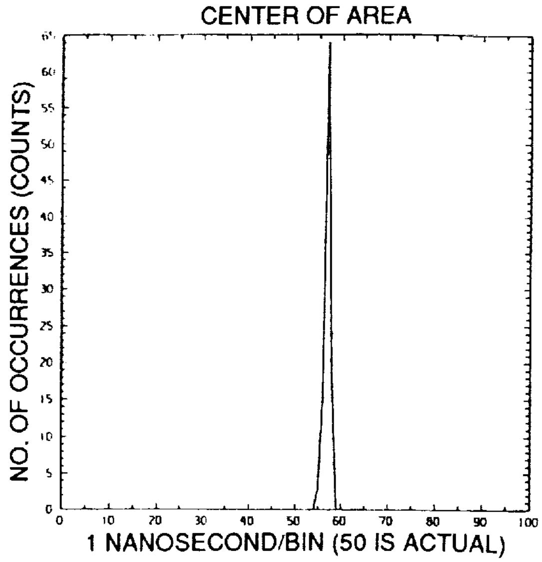


Figure 2.2E

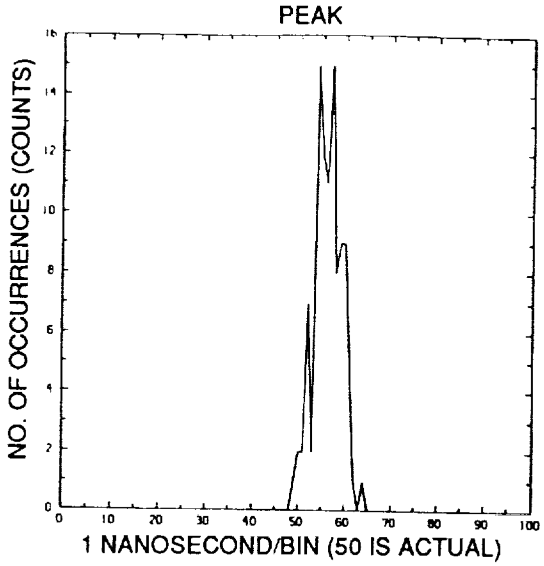


Figure 2.2F

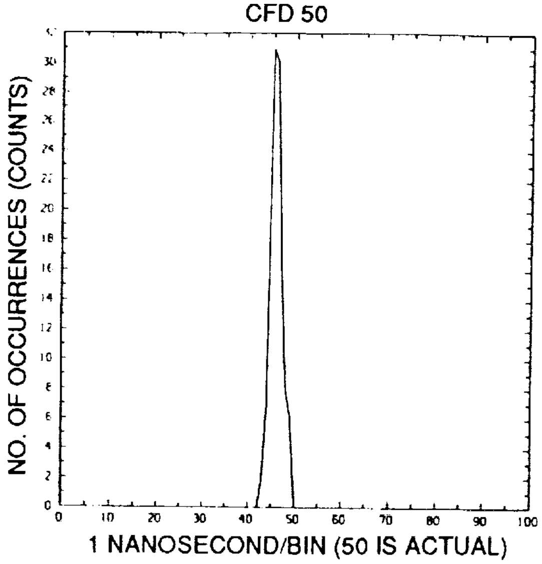


Figure 2.2G

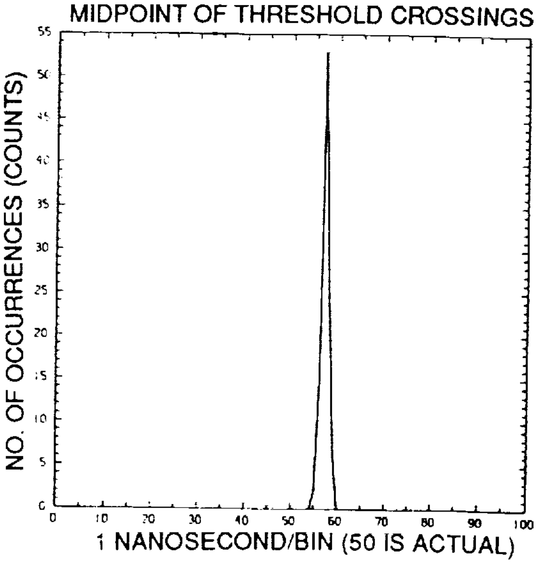


Figure 2.2H

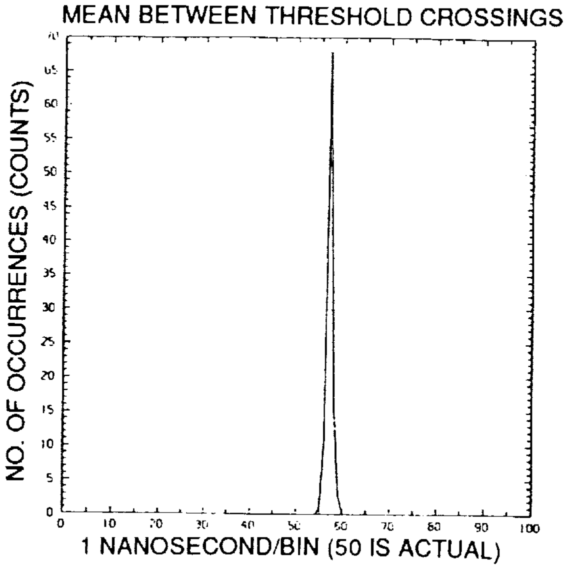


Figure 2.2I

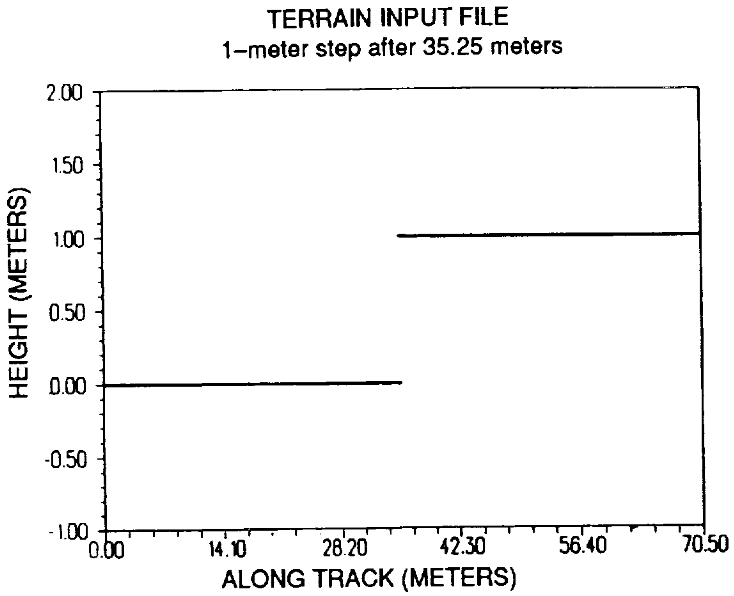


Figure 2.3A

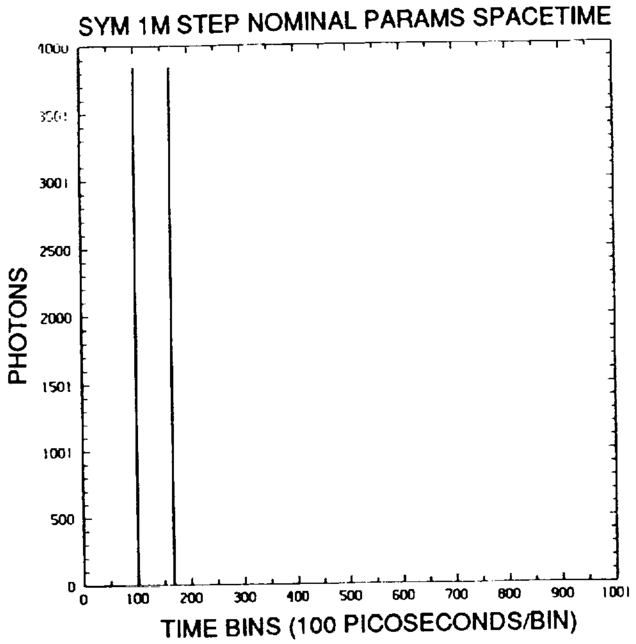


Figure 2.3B

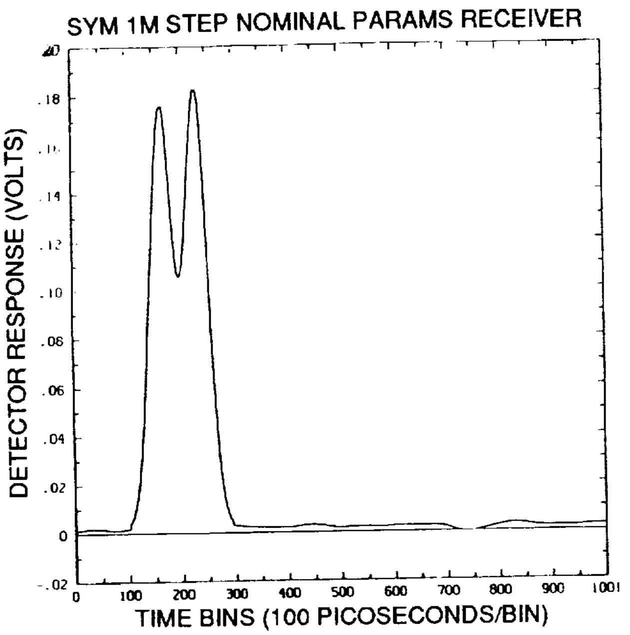


Figure 2.3C

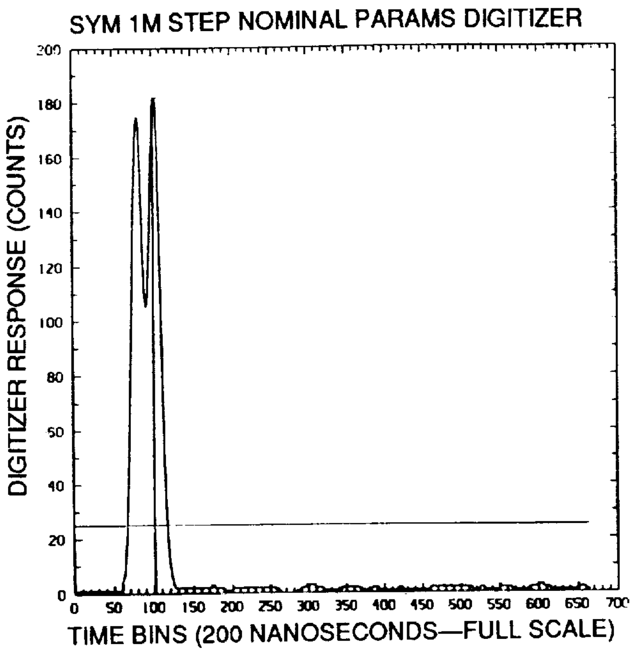


Figure 2.3D

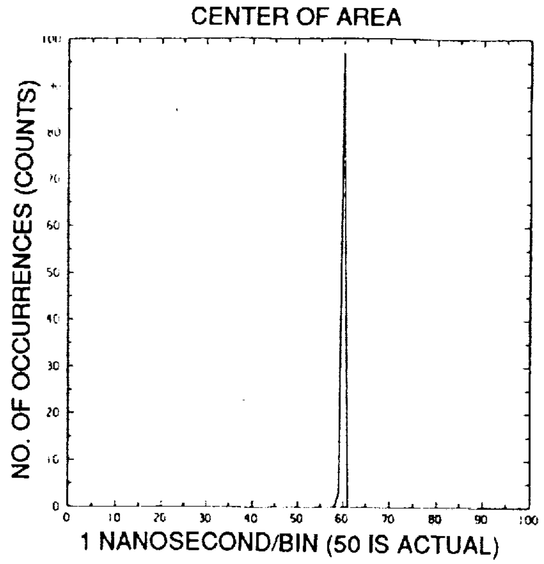

CENTER OF AREA

Figure 2.3E

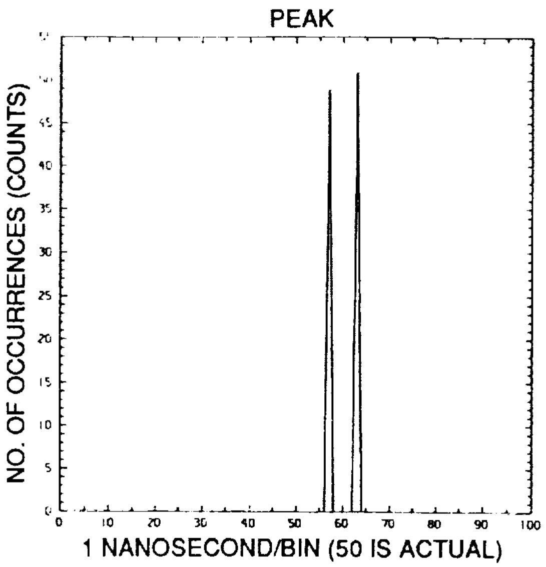

PEAK

Figure 2.3F

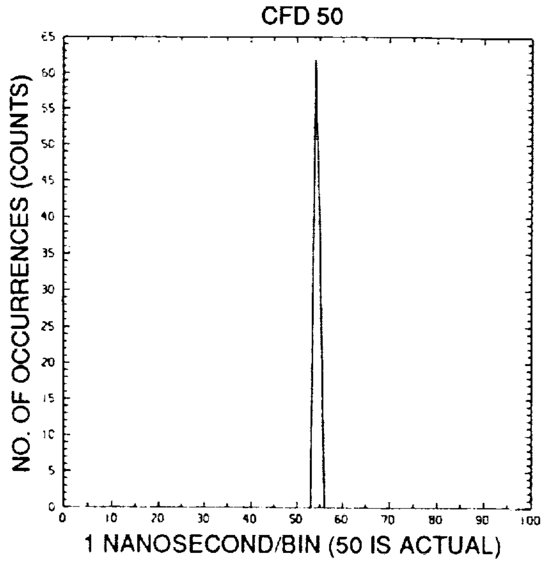

CFD 50

Figure 2.3G

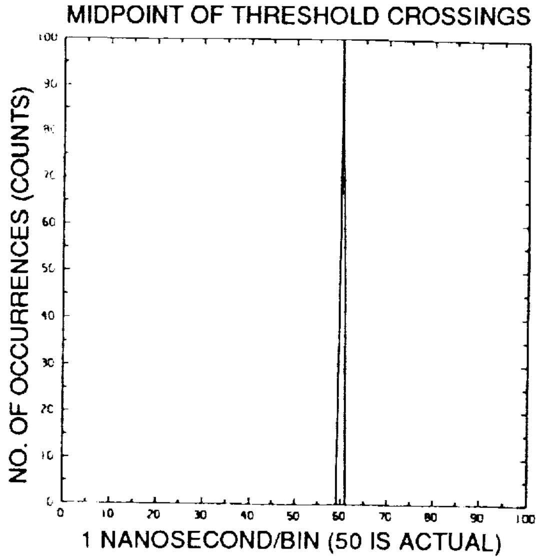

MIDPOINT OF THRESHOLD CROSSINGS

Figure 2.3H

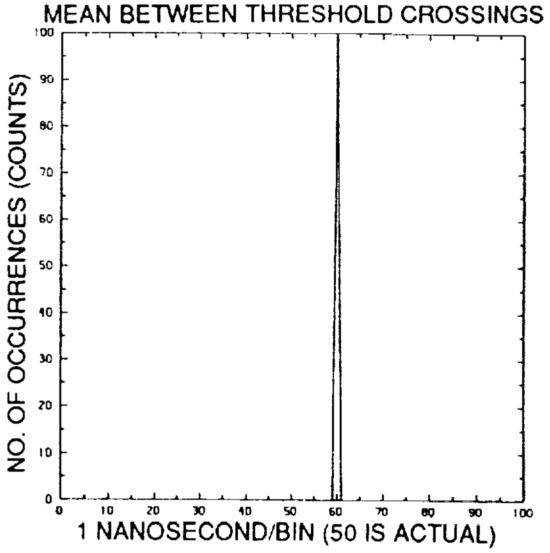

MEAN BETWEEN THRESHOLD CROSSINGS

Figure 2.3I

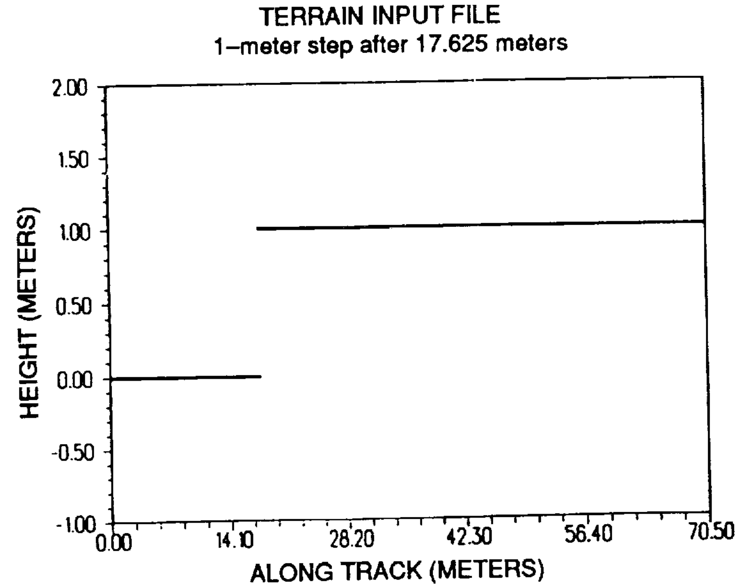


Figure 2.4A

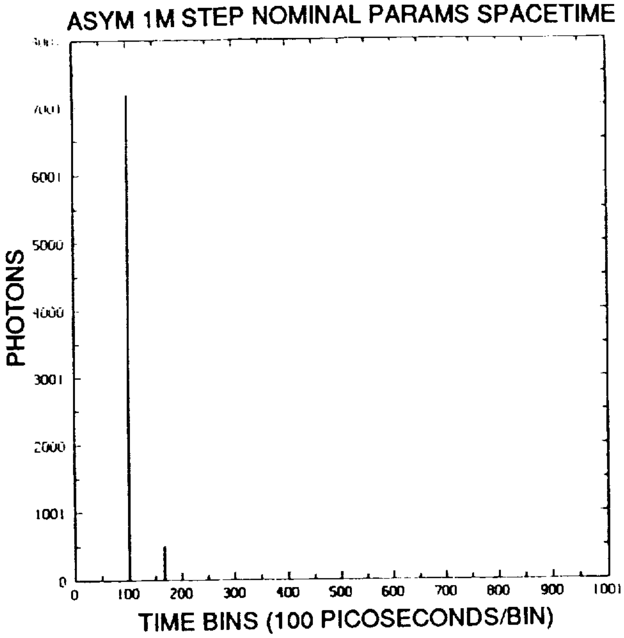


Figure 2.4B

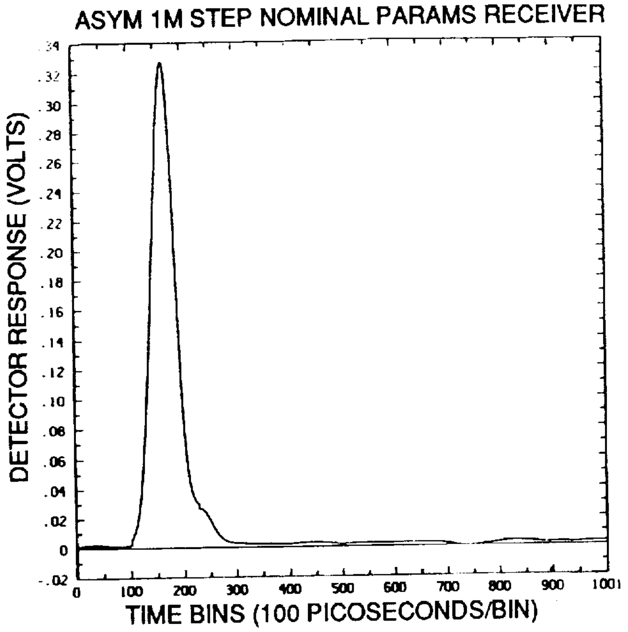


Figure 2.4C

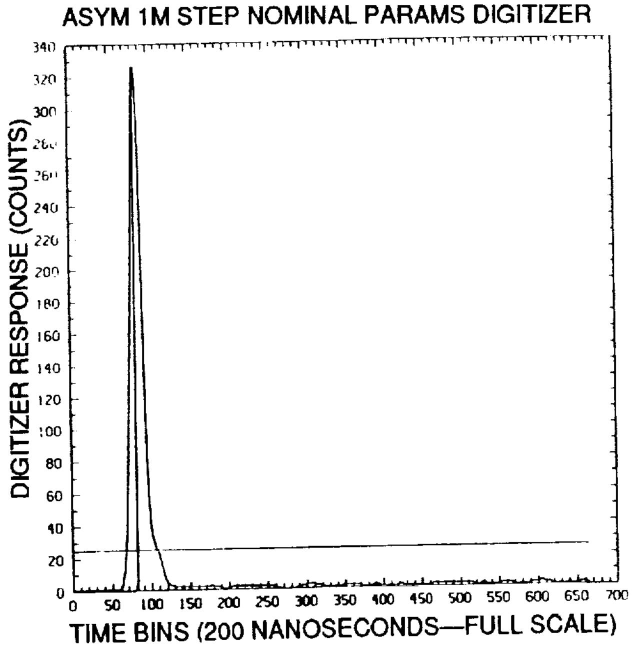


Figure 2.4D

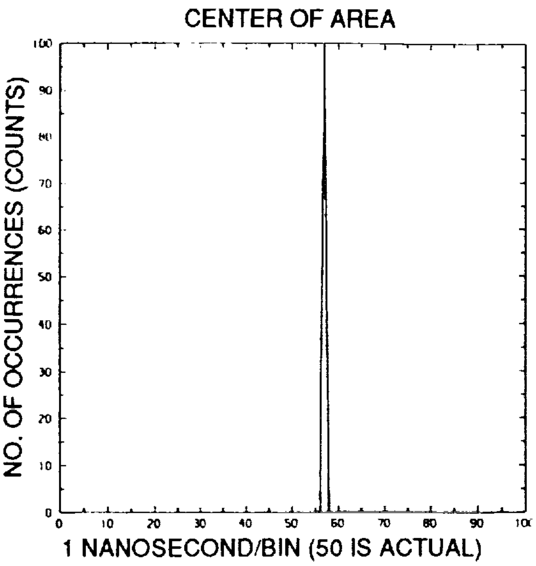


Figure 2.4E

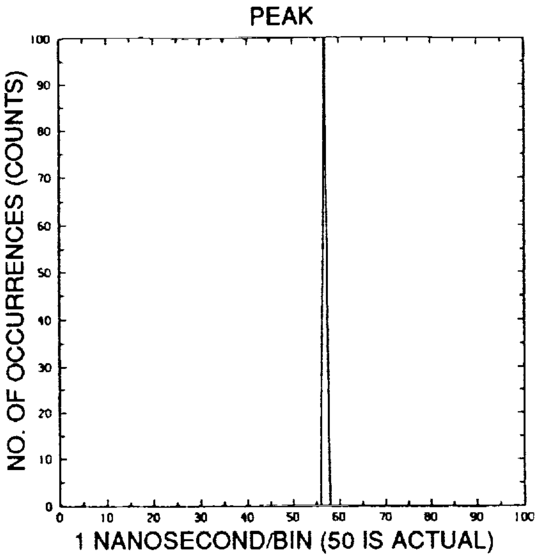


Figure 2.4F

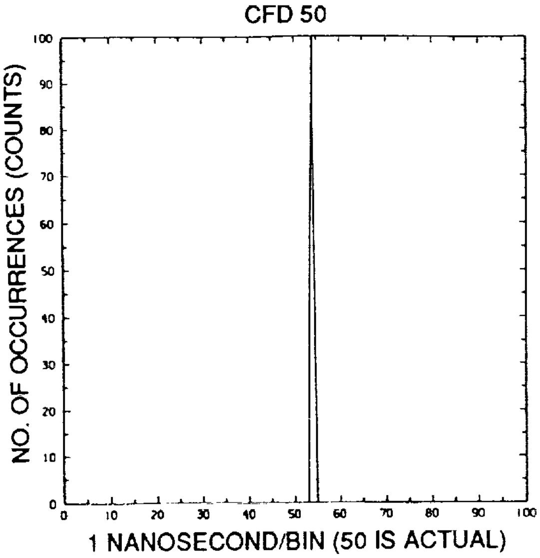


Figure 2.4G

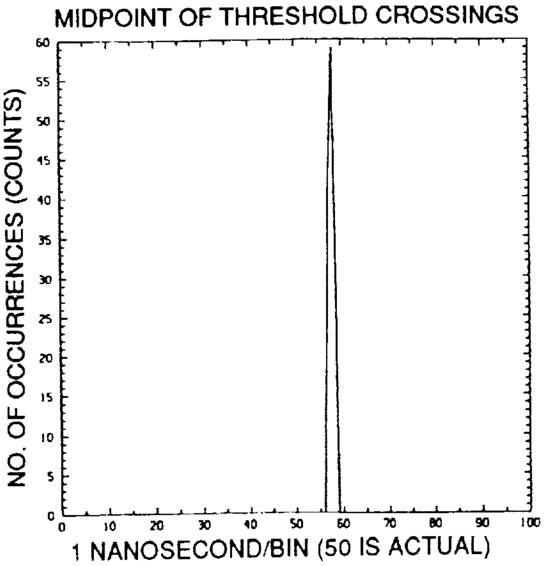


Figure 2.4H

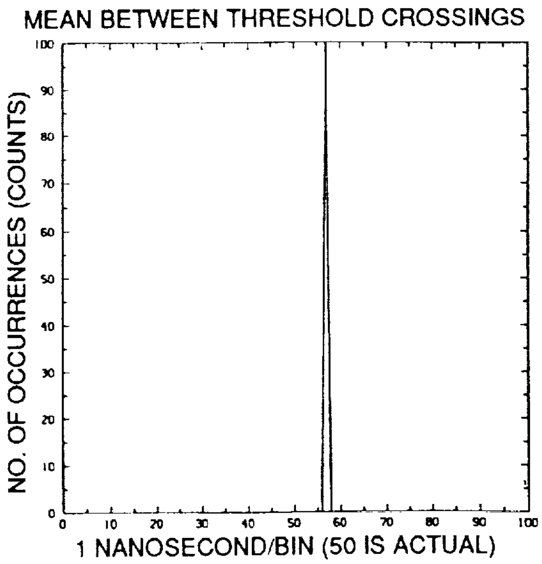


Figure 2.4I

The small height features in the rough ice occasionally cause an increase in the simulator's timing jitter. This is due to variations in photon counts near the receiver's threshold. Occasionally these cause the threshold crossing time to shift from one ice feature to another. When this occurs, the waveform timing point can shift considerably. The timing histograms of such data typically have a bimodal shape and a large RMS time jitter.

2.3 Terrain Re-creation

If the satellite altitude is known exactly, the terrain height estimates from the simulator can be used to "re-create" the height profile of the measured terrain. A sample segment of ice terrain, which included ice roughness classes 1, 3, and 5, was used to illustrate this mode using both the peak and mean fine timing estimators. For each estimator, the terrain height was determined at the center of the laser footprint for every laser firing. The satellite height was assumed to be known exactly. The following cases illustrate results from the Peak and Mean estimators:

1) Nominal parameters (given in Section 6).
2) Nominal parameters, receiver filter impulse response = 200 psec (nominal = 5 nsec).
3) Nominal parameters, receiver filter impulse response = 200 psec, laser divergence = 10 μrad.

The results for the Peak estimator are shown in Figure 2.10 and for the Mean estimator in Figure 2.11. The x's denote the simulator height estimates.

The figures show the abrupt transitions caused by the Peak estimator, and the time bias caused by the delay of the receiver impulse response. The spatial smoothing caused by the nominal beam divergence is also shown. As expected, the Mean estimator with the shortest impulse response time and narrowest beam divergence gives the best performance.

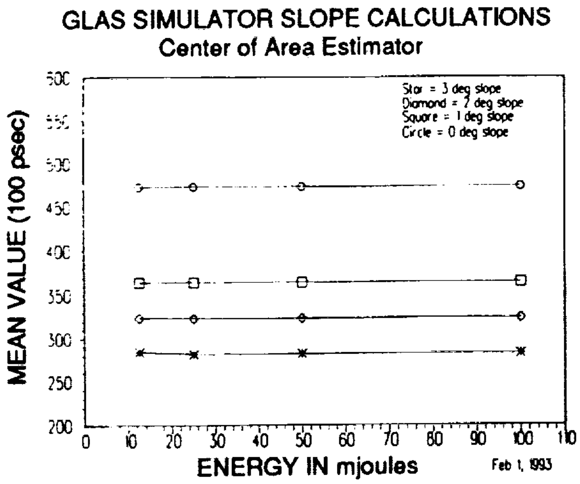


Figure 2.5A

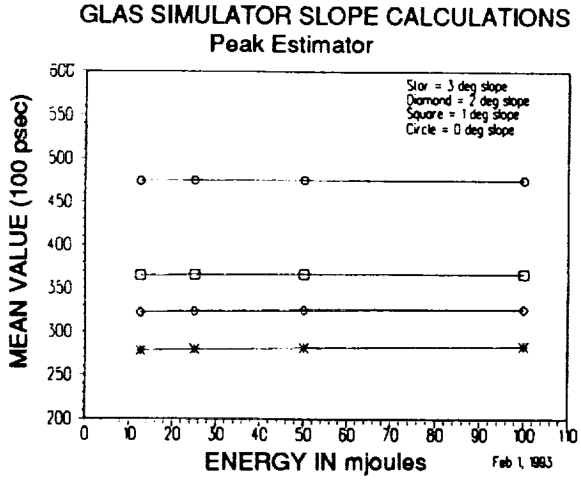


Figure 2.5B

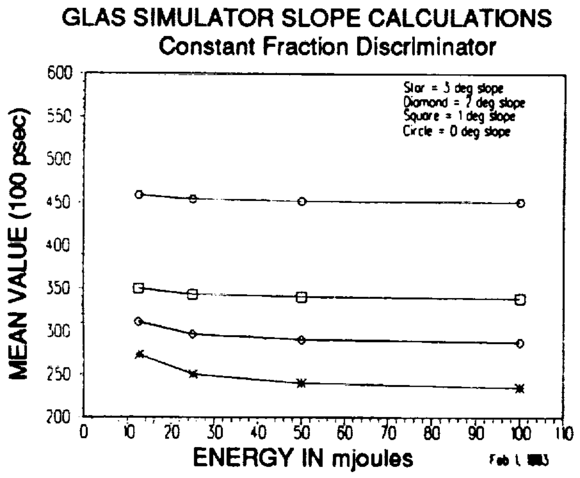


Figure 2.5C

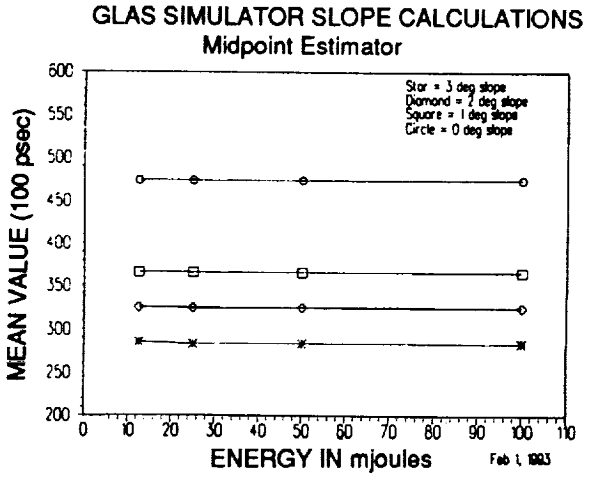


Figure 2.5D

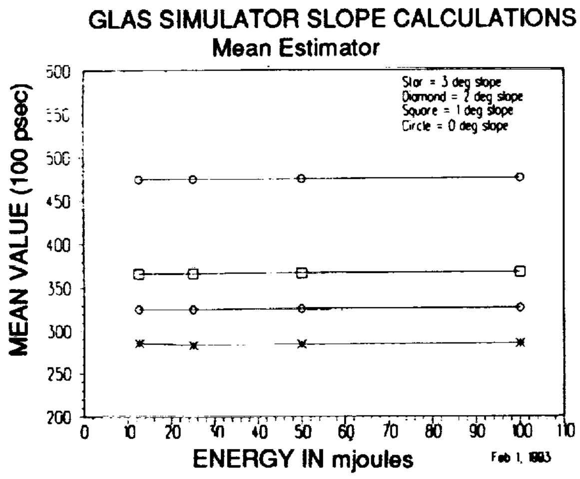


Figure 2.5E

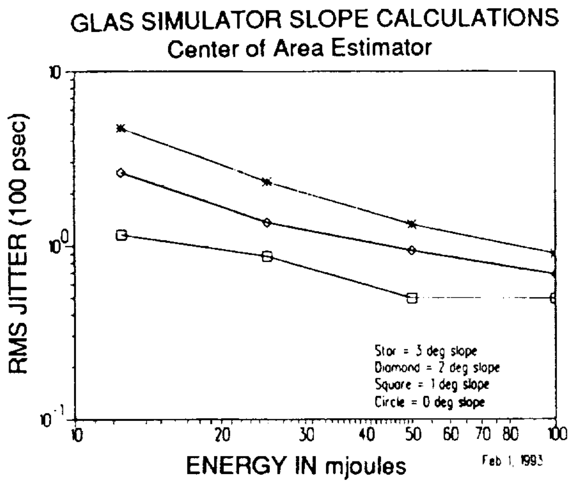


Figure 2.6A

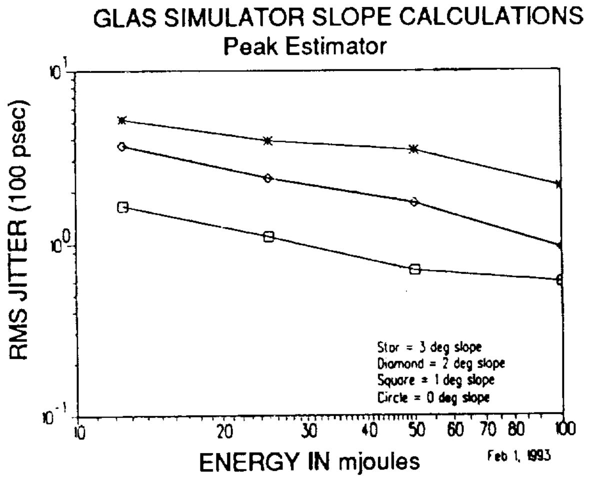


Figure 2.6B

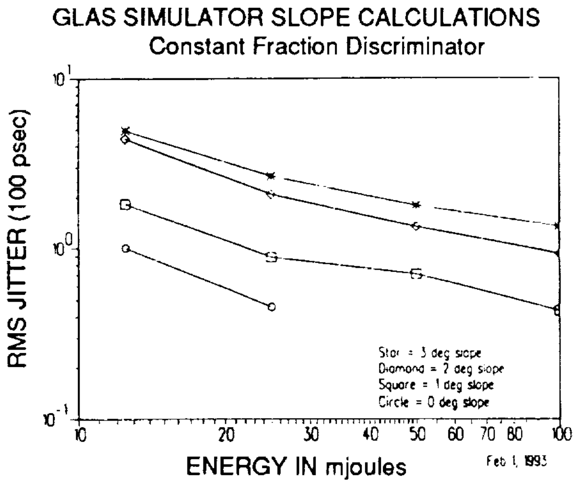


Figure 2.6C

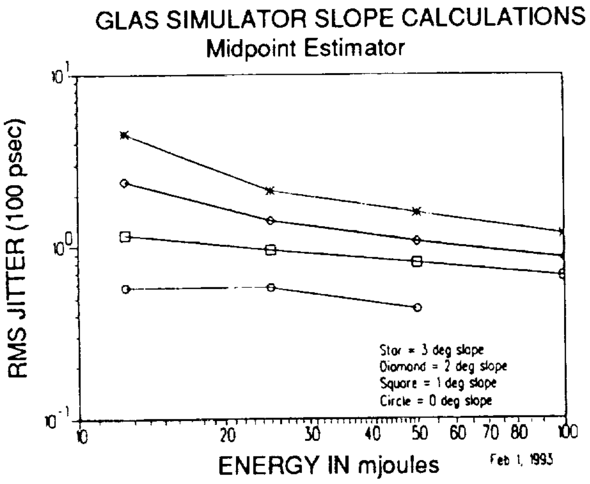


Figure 2.6D

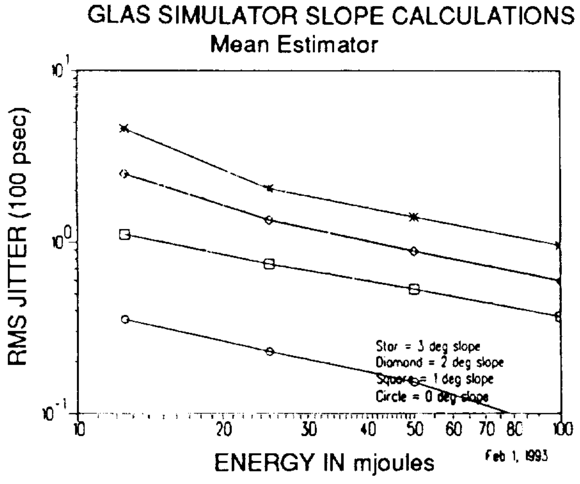


Figure 2.6E

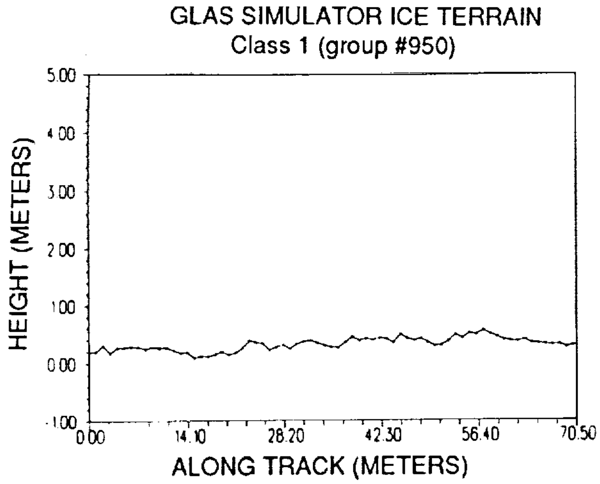

GLAS SIMULATOR ICE TERRAIN
Class 1 (group #950)

Figure 2.7A

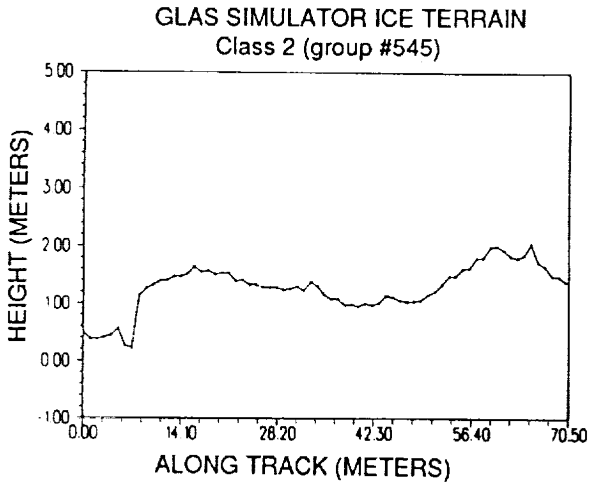

GLAS SIMULATOR ICE TERRAIN
Class 2 (group #545)

Figure 2.7B

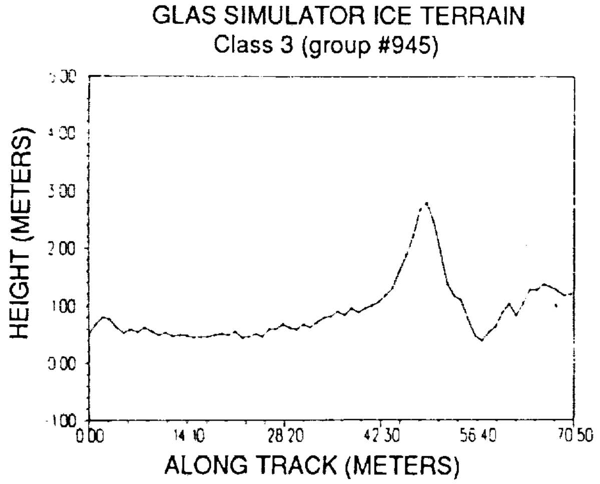

GLAS SIMULATOR ICE TERRAIN
Class 3 (group #945)

Figure 2.7C

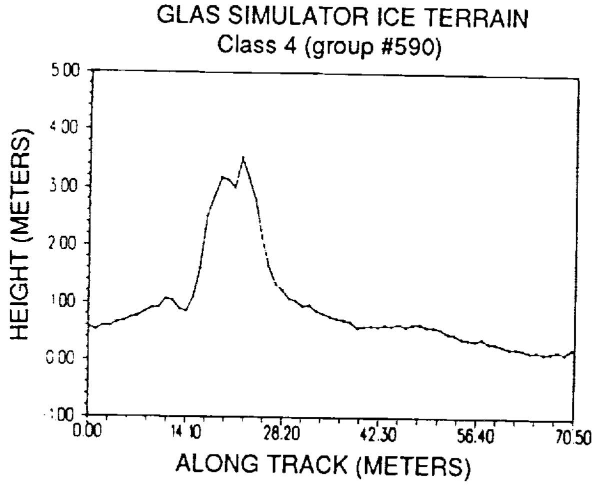

GLAS SIMULATOR ICE TERRAIN
Class 4 (group #590)

Figure 2.7D

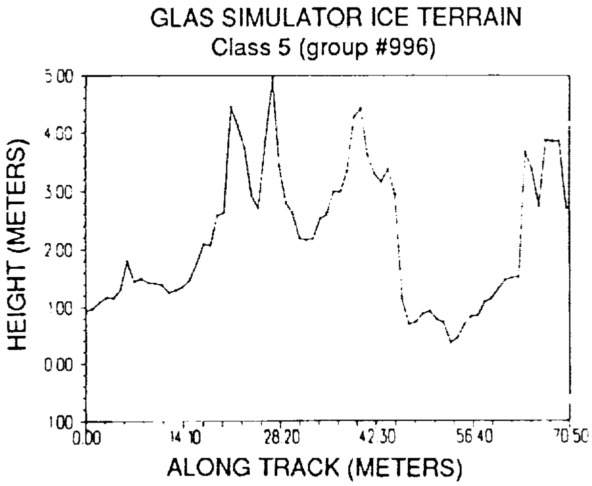

GLAS SIMULATOR ICE TERRAIN
Class 5 (group #996)

Figure 2.7E

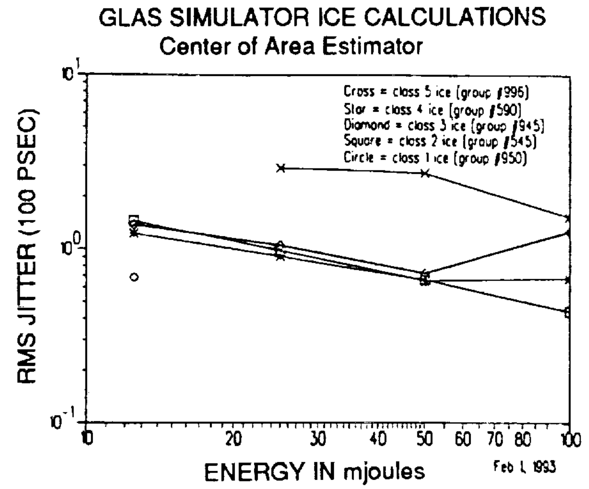


Figure 2.8A

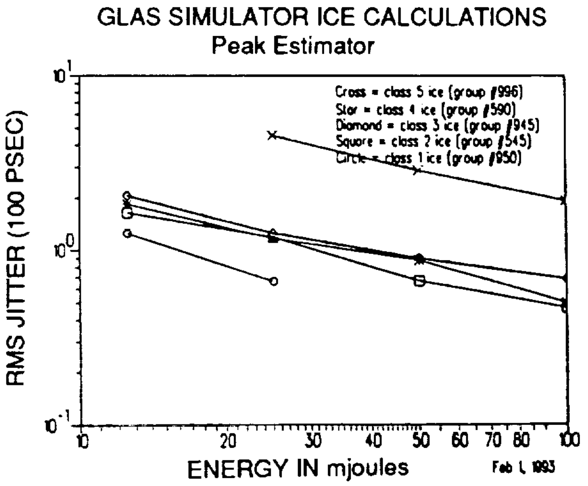


Figure 2.8B

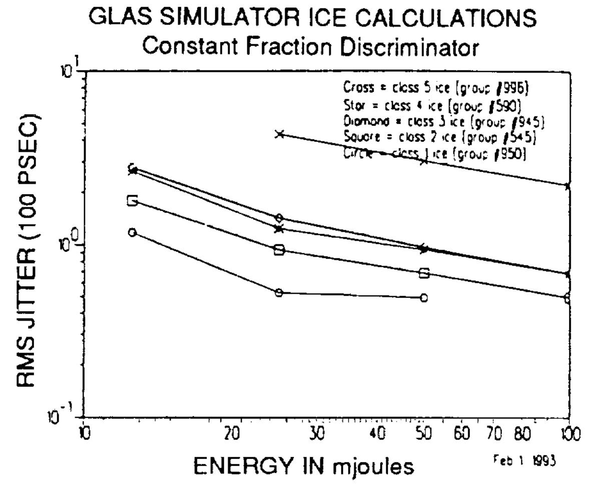


Figure 2.8C

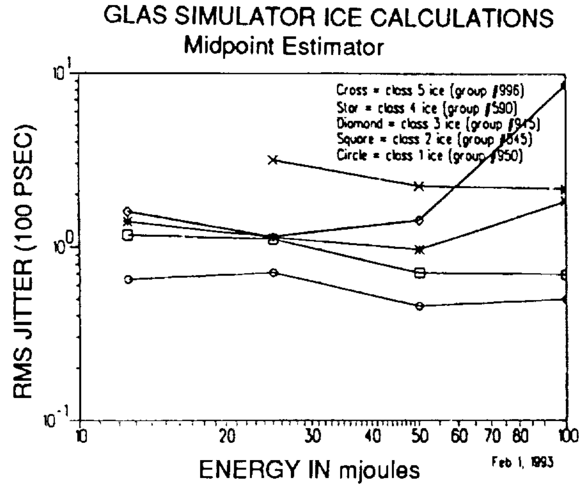


Figure 2.8D

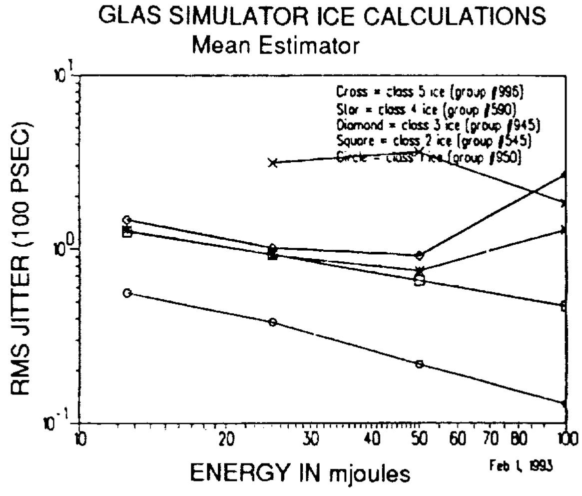


Figure 2.8E

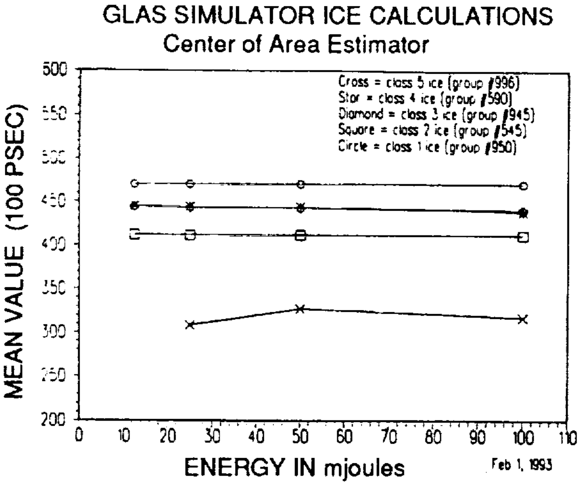


Figure 2.9A

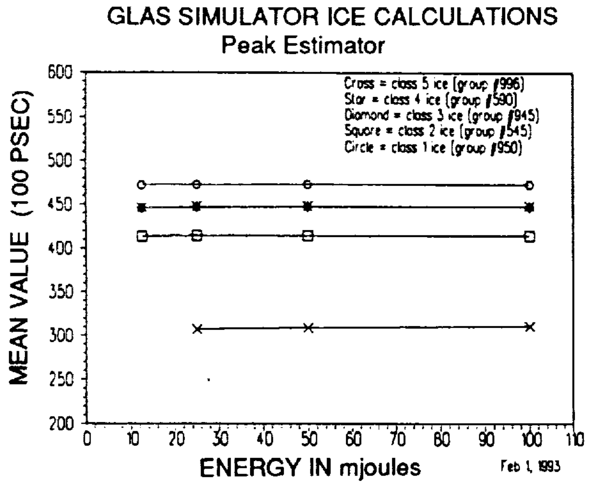


Figure 2.9B

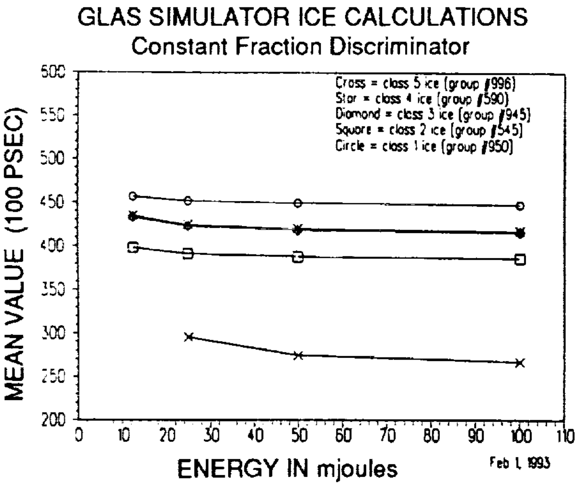


Figure 2.9C

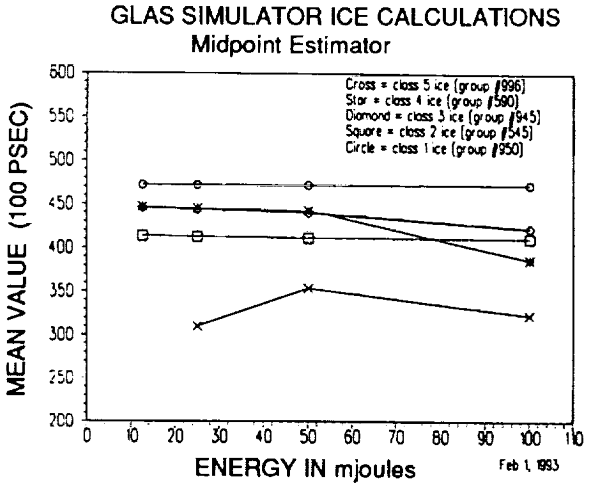


Figure 2.9D

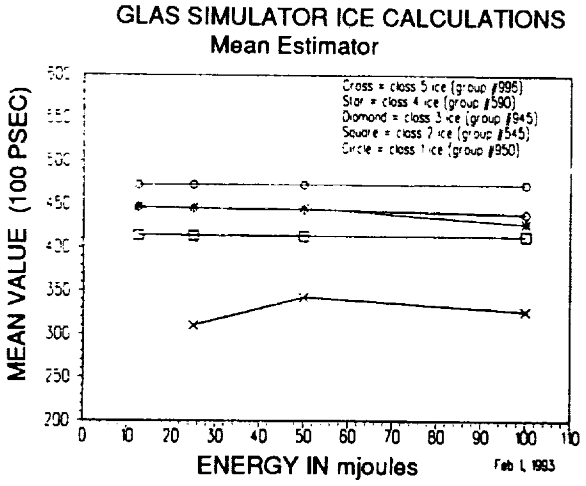


Figure 2.9E

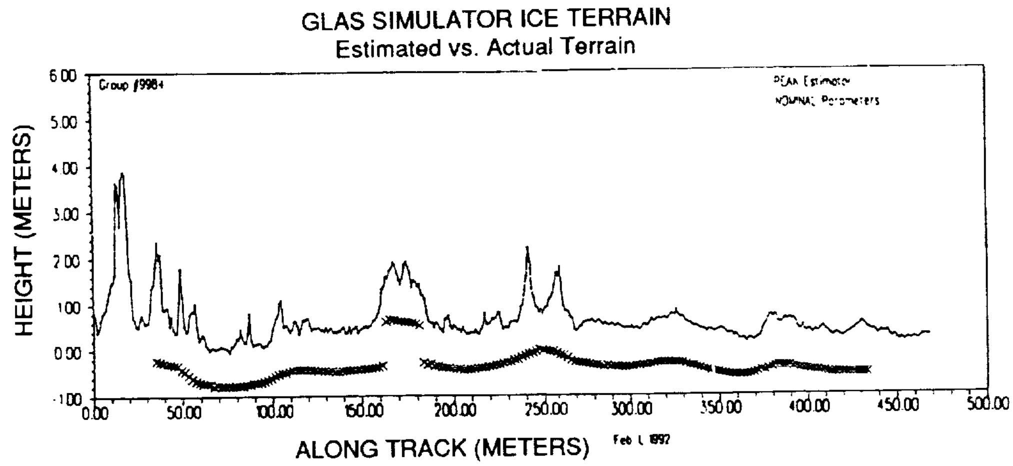


Figure 2.10A

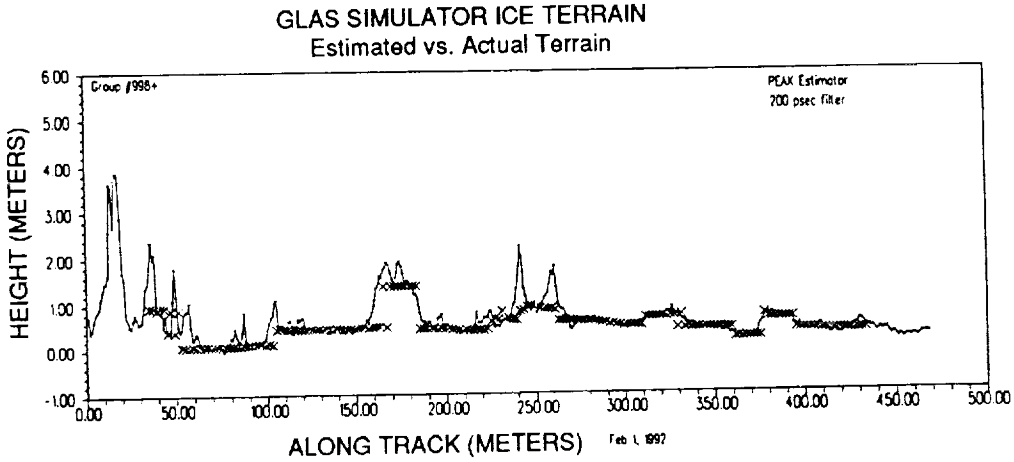


Figure 2.10B

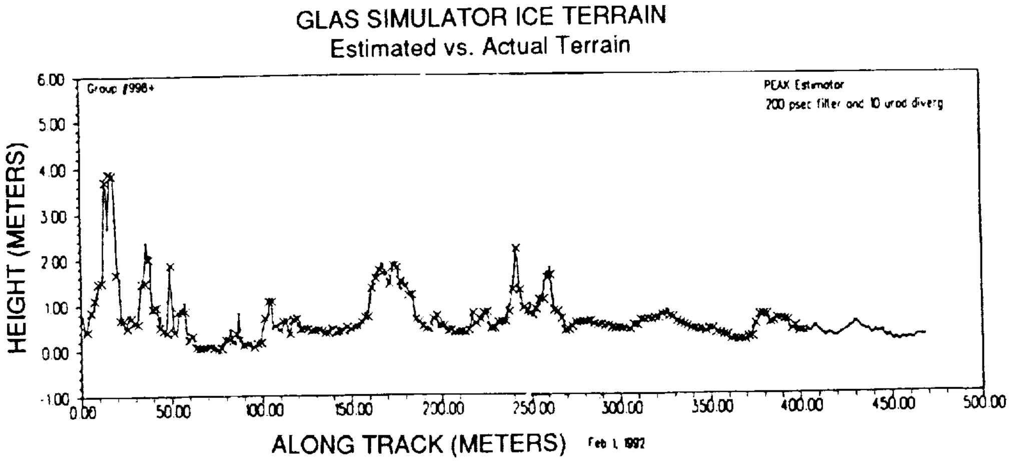


Figure 2.10C

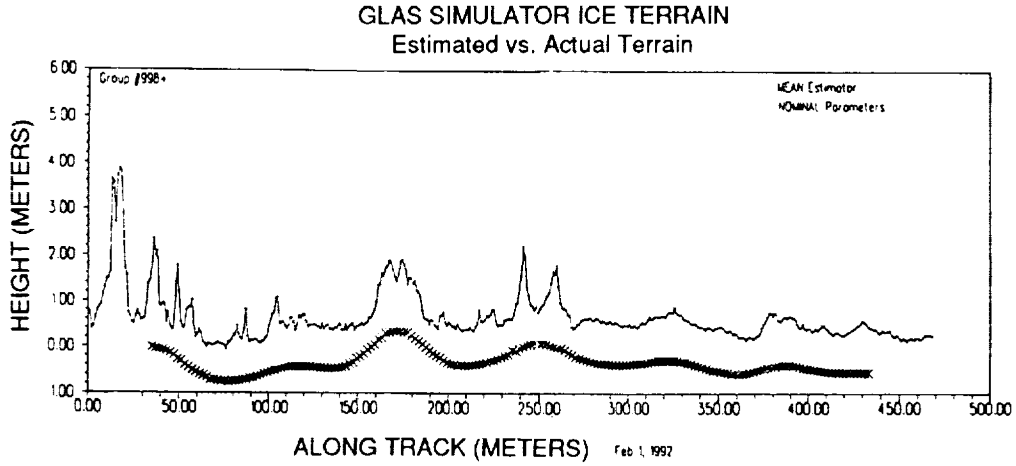


Figure 2.11A

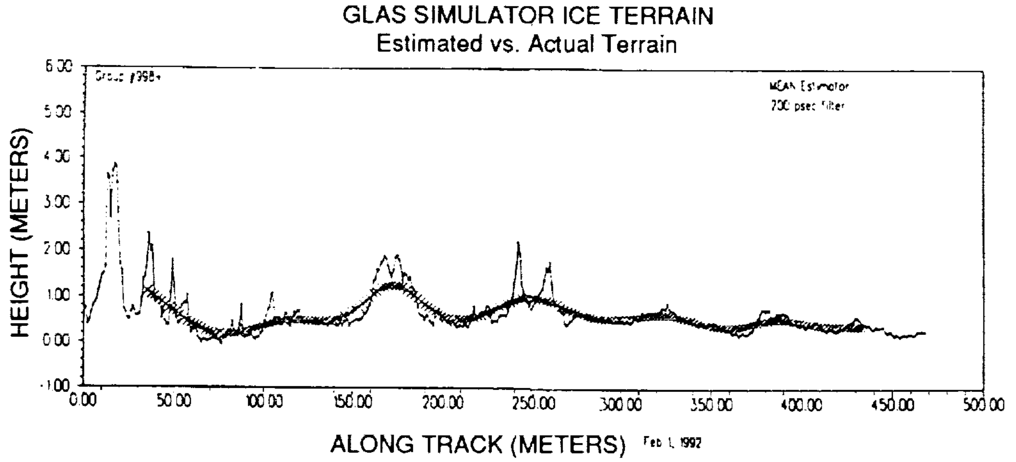


Figure 2.11B

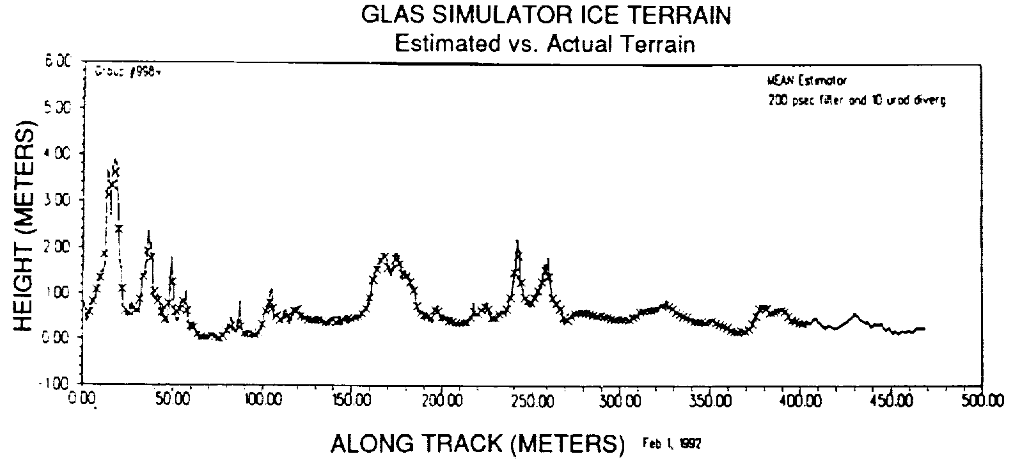


Figure 2.11C

3.0 Simulator Design Overview

To simulate the laser altimetry measurement, the main routine, ***sim***, first reads the parameter table and the terrain file. The simulator then calls the subroutines SPACE_TIME, RECEIVER, and DIGITIZE to simulate the altimetry measurement and a subroutine, TERGPH, to analyze the data.

The SPACE_TIME subroutine simulates the firing of a laser from a specified altitude to a given terrain profile. Using link equations, the returned photon energy from the laser is calculated as a function of time. A return waveform in the time domain for each shot is created as the subroutine output. This routine is described in Sections 3.1 and 4.2.

The RECEIVER subroutine adds system noise and then converts the waveform photons received from the SPACE_TIME subroutine into voltages for a single laser shot. In addition to the waveform, the subroutine returns a threshold for the DIGITIZE subroutine based on system noise statistics that are calculated in the RECEIVER subroutine. This routine is described further in Sections 3.2 and 4.3.

The DIGITIZE subroutine digitizes the waveform output from the RECEIVER subroutine. The waveform characteristics are specified in the parameter table. DIGITIZE also calculates the five timing correction estimators, and accumulates timing statistics for all shots. This routine is described in Sections 3.3 and 4.4.

TERGPH generates two files for plotting: the actual terrain profile covered by all of the shots, and the altimeter's height estimates. The user can compare the altimeter's measured profile against the terrain profile by plotting both of these files on the same graph. This routine is described in Sections 3.4 and 4.5.

3.1 SPACE_TIME Design

The SPACE_TIME routine is called once per shot. It propagates the laser pulse from the transmitter through the atmosphere, reflects it from the terrain, and propagates it back through the atmosphere to the receiver. The simulator does not include the effects of atmospheric refraction.

The laser pulse is transformed from the space to time domain when it interacts with the terrain. This subroutine breaks up the transmitted optical wavefront into a number of narrow optical rays. This model approach is valid as long as the terrain is in the far field of the laser transmitter. The number of rays depends on the satellite distance, the satellite off-nadir pointing angle, the transmitter beam divergence, the height of the terrain surface and the slope of the surface at each point. The individual rays have sufficiently small angular width so that each has negligible ($\leq$ 1 bin) pulse spreading after interacting with its terrain segment. The along-track surface profile is divided into discrete 1-centimeter segments. To simplify the computations, the rays are constrained to have their tips lie exactly on the transition of the terrain segments (see Figure 3.1).

This constraint causes the angular width of the rays to vary across the beam.

The terrain profile is analyzed by SPACE_TIME as shown in Figure 3.2. The size of the time bin ($\Delta t=100$ psec) and the terrain step size (=1 centimeter) are fundamental constants used in the simulator and cannot be changed.

The SPACE_TIME routine performs the following computations:

[A] Compute the ray angular width.

The ray's spatial width is chosen to be sufficiently small to ensure that all the light returning from a single ray will lie within one time bin. This means that $\Delta r = r_2 - r_1 \leq 1.5$cm (see Figure 3.3). To ensure these limits on Δr (as well as to simplify the software), each ray base is constrained to a single 1cm terrain segment. From the geometry shown in Figure 3.3:

$$\tan(\theta+d\theta) = x_2 / (Rsat - h_2)$$

$$\text{and } x_2 = x_1 + 0.01$$

which, when solved for the ray angular width, $d\theta$, yields

$$s_i = (Rsat - h_{i-1})*\tan\theta_i + 0.01$$

$$d\theta_i = \tan^{-1} (s_i/(Rsat - h_i)) - \theta_i$$

where:

Rsat is satellite altitude above geoid,
h_i is height above geoid,
θ_i is off-nadir angle to ray "i", and
all distances are given in meters.

[B] Read the terrain from file.

The number of terrain points read is NP, which is equal to the number of rays in the transmitter beam. The program reads NP terrain elements, starting at index "vsat*(k-1)*100+1" where "k" is the shot number and "vsat" is the velocity of the satellite in meters/shot.

[C] Compute the terrain slope angle of incidence.

The program uses 2 terrain points to compute the slope of the terrain within the ray, (see Figure 3.3):

$$\theta^i{}_T = \tan^{-1} \left(\frac{h_i - h_{i-1}}{0.01m}\right)$$

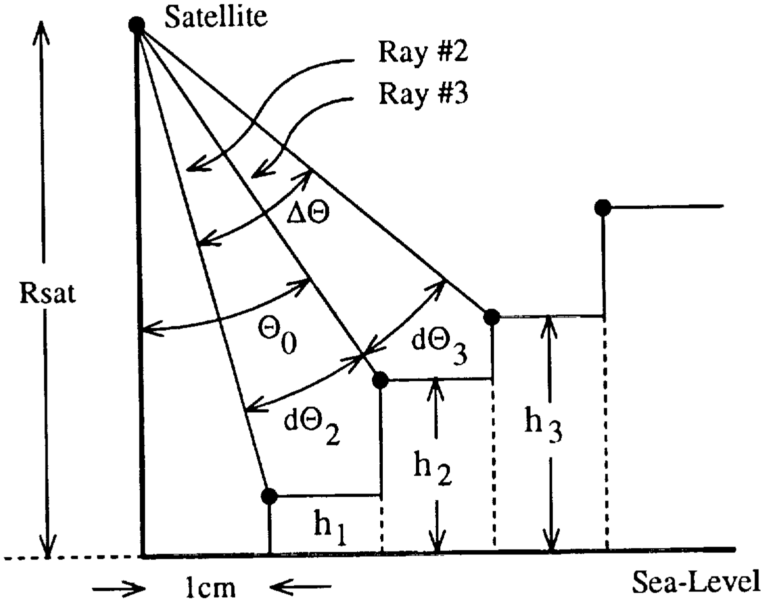


Figure 3.1. SPACETM Geometry.

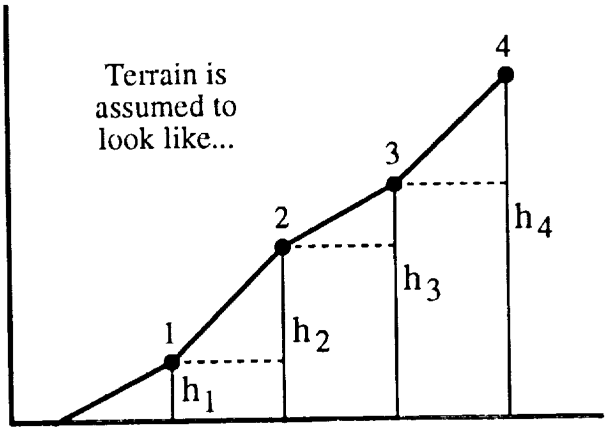


Figure 3.2. How SPACETM views terrain.

The angle of incidence of the beam with the terrain, ϕ_i (see Figure 3.4) is:

$$\phi_i = \theta_i - \theta^i{}_T$$

where:

$$\theta_i = \left(\theta_0 - \frac{\Delta\theta}{2}\right) + \sum_{j-1}^{i-1} d\theta_j$$

[D] Compute the reflectivity and energy in the ray.

The diffuse reflectivity (due to Lambertian scattering) is given by:

$r_i = r_i \cos\phi_i$ for $\phi < 50°$

$= 0$ for $\phi \geq 50°$

where r_i the diffuse reflectivity is read from the terrain file.

The Lambertian scattering angles were limited to < 50 deg. to prevent photons from a single ray falling outside of a single bin in time. Angles of incidence greater than 50 deg in combination with certain other parameter values would cause this to happen.

The spatial distribution of the transmitted energy is assumed to be Gaussian with distribution:

$$I(x,\sigma) = \frac{1}{\sqrt{2\pi}} \exp\left(\frac{-(x/\sigma)^2}{2}\right)$$

The energy for each ray is given by:

$$E_i = E_t \cdot I((\theta_i - \theta_0), \Delta\theta) \cdot \left(\frac{6 \cdot d\theta}{\Delta\theta}\right)$$

where the assumption has been made that $\Delta\theta/2$ is the 3σ point of the Gaussian distribution.

[E] Perform the link analysis.

The receiver-field-of-view area on the terrain is:

$$\text{Aspot} = \pi \bullet (\text{RECFOV} \bullet \text{Rsat}/2)^2$$

For the ith ray:

$$A_i = \text{Aspot} \bullet d\theta_i / \Delta\theta$$

where $d\theta_i$ is the ray angle and $\Delta\theta$ is the laser divergence.

This assigns an area to the ith ray proportional to its fraction of the along-track beam width. The average number of signal photons returning to the receiver in the ith ray is:

$$N_i = E_t \cdot \left(\frac{\lambda}{hc}\right) \cdot \left(\frac{Arec}{x_i^2}\right) \cdot \tau_{sys} \cdot \tau_{atm}{}^2 \left(\frac{r_i}{\pi}\right)$$

The average solar background rate (in photons/sec) seen by the receiver detector for this ray is:

$$B_i = I_{day} \cdot f \cdot \left(\frac{\lambda}{hc}\right) \cdot \Delta\lambda \cdot Arec \cdot \tau_{sys} \cdot \left(\frac{A_i}{x_i^2}\right) \cdot \left(\frac{r_i}{\pi}\right)$$

where:

Et is the total transmitted laser energy,
λ is the laser wavelength,
h is Planck's constant,
c is the speed of light,
Arec is the area of the receiver,
$x_i = (Rsat - H_i)/cos_i$ is the slant range,
τ_{sys} is the system transmission,
τ_{atm} is the 1-way atmospheric transmission,
H_i is the terrain height,
r_i is surface diffuse reflectivity,
I_{day} is the day solar irradiance at the Earth's surface (W/m^2 nm),
f is the night/day fraction, and
$\Delta\lambda$ is the receiver spectral bandpass (nm).

[F] Compute the time of arrival of the photons in this ray

Assuming that the altimeter's coarse clock starts when the laser fires, the arrival time of the i^{th} ray is given by

$$T_i = \frac{2s_i}{c}$$

where the i^{th} slant range, shown in (Figure 3.5), is

$$s_i = \frac{(R - h_i)}{\cos\theta_i}.$$

The arrival time T_i is used to create an index in the timing histogram

$$J_i = \frac{(T_i - T_{min})}{\Delta t}$$

where T_{min} is the minimum delay time expected and Δt = 100psec is the simulator's time resolution.

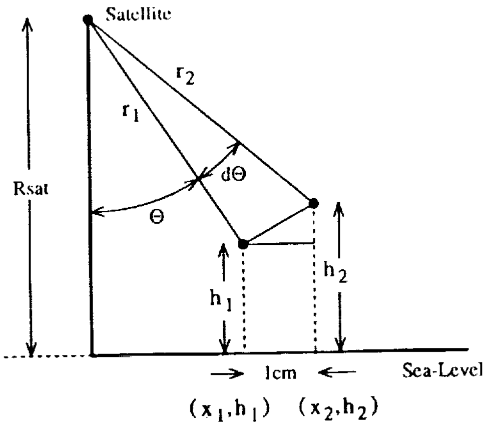


Figure 3.3. Ray/Terrain Connection.

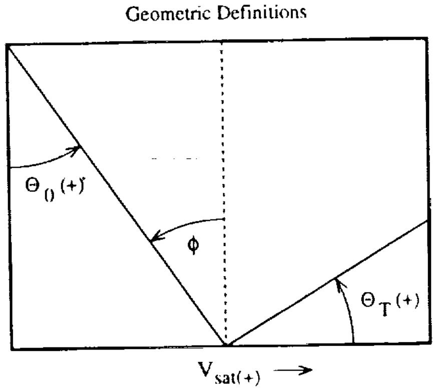


Figure 3.4 Geometric Sign Conventions.

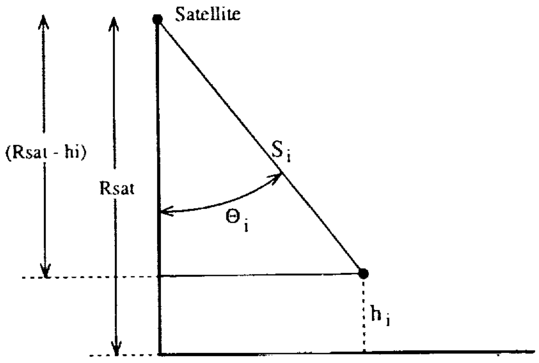


Figure 3.5. Computing Roundtrip Time.

3.2 RECEIVER Design

The RECEIVER subroutine simulates the response of the detector by processing the photon waveform from the SPACE_TIME subroutine and generating the detected electrical waveform. The detector is modeled as either an avalanche photodiode (APD) or a photomultiplier tube (PMT), followed by a low pass filter. Detector noise is included before, during and after the actual signal waveform. The noise only portion of the response which surrounds the signal and noise segment is used to calculate the threshold level for DIGITIZE subroutine.

APD Detector

For every time increment, Δt, RECEIVER calculates the mean signal response from the APD detector by using:

$$V_{out}(t,\Delta t) = Nph \cdot \eta \cdot R_L \cdot q \cdot G \qquad (3.2.1)$$

where the detector constants are:

Nph is the number of signal photons illuminating the detector at time (T) in the time bin Δt
T_p is the integration time
η is the quantum efficiency of the detector
R_L is the load resistor [ohms]
q is the electron charge 1.60×10^{-19} [C]
G is the Gain of the APD

The RECEIVER routine adds detector noise to the electrical waveform. The detector noise is modeled with Gaussian statistics. The APD noise moments are calculated in photoelectrons, referenced to the input of the APD just following detection. In a given time, the mean photoelectron count is:

$$<N>=N_s+N_{back}+N_{bulk} \qquad (3.2.2)$$

The standard deviation of the number of photo-emissions (in photoelectrons) is given by:

$$\sigma_N \equiv \sqrt{Var(N)} = \sqrt{F(N_{back} + N_{bulk}) + N_{thi}} \qquad (3.2.3)$$

The excess noise factor of the APD gain is given by:

$$F = k_{eff}G + (1-k_{eff})(2-\frac{1}{G})$$

The photoelectrons levels due to optical background and APD bulk leakage current in time Δt are:

$$N_{back} = \eta \cdot B_i \cdot \Delta t$$

and

$$N_{bulk} = \frac{i_{bulk} \cdot \Delta t}{q \cdot G}$$

The equivalent variation in photoelectron emissions caused by thermal noise in the APD preamp in time Δt is:

$$N_{thi} = \frac{2 \cdot K_B \cdot T_r \cdot \Delta t}{q^2 \cdot R_L \cdot G^2}$$

In these equations:

N_{back} is the background noise count in time Δt [photoelectrons],
η is the APD quantum efficiency
B_i is the background noise rate [photons/sec],
Δt is the resolution time = 100 psec,
N_{bulk} is the APD bulk leakage charge in time Δt, referenced to the input [photoelectrons],
i_{bulk} is the APD bulk leakage current measured at the output[A],
G is the APD gain,
N_{thi} is the preamp thermal noise referenced to the input [photoelectrons],
K_B is Boltzmans constant = 1.38×10^{-23} [C],
T_r is receiver pre-amplifier noise temperature [°K],
k_{eff} is the APD's effective ionization ratio, and
R_L is the preamplifier resistor.

PMT Detector

RECEIVER also includes a model for a PMT detector. The mean voltage response in a time bin Δt is given by (3.2.1). However, the detector fluctuations in each time bin are calculated using Poisson statistics. The number of counts in each bin Δt, referenced to just after the PMT photocathode are:

$$N_{tot} = N_{sig} + N_{back} + N_{dark} \qquad (3.2.4)$$

The number of counts caused by optical background and dark noise in time Δt are given by:

$$N_{back} = \eta \cdot B_i \cdot \Delta t$$

$$N_{dark} = i_d \cdot \Delta t / qG,$$

where i_d is the PMT dark current [A].

Filtering

The final segment of the receiver subroutine is the low pass filter. The output of the detector is passed through a unity gain electrical filter with a specified impulse response width at the full width at half maximum (FWHM). The filter impulse response can be selected to have either a square wave or Gaussian shape.

Receiver Threshold

The receiver threshold is set by finding the maximum output for the two noise only segments. The threshold is then set to slightly above (0.1%) this maximum value. This "no false alarm" algorithm will be modified in the next version of the simulator.

3.3 WAVEFORM DIGITIZER Design

The waveform digitizer subroutine, 'digitize', emulates an A/D converter digitizing the filtered detector receiver output signal. DIGITIZE calculates where the first and last threshold crossings occur and then samples the input waveform by averaging a number of input bins (data points) specified by time scale, parameter #56. The input waveform and threshold are supplied by the receiver subroutine. DIGITIZE scales each sampled bin from volts to A/D counts by using the specified A/D resolution, # of bits, and the expected maximum input voltage range of the A/D given in the parameter file.

Using the scaled waveform, DIGITIZE then calculates and returns the five waveform fine timing estimates. They include Center of Area, Peak, 50% Constant Fraction Discriminator (CFD), Mean, and Midpoint. The value of each fine timing estimator is the location in time (array index value) where the estimate occurs in the digitized waveform. The timing estimators are given in units of the 100 psec time bins and are referenced to the start of that waveform. The RMS width of the waveform is also calculated.

3.3.1 Waveform Estimators

The Figure 3.6 shows a digitized waveform example with sketches of the estimator values. The DIGITIZE routine samples the electrical waveform output by using:

$$v_D(i) = \frac{\sum_{j=i}^{i+T_D-1} v(j)}{T_D} \qquad 3.3.1$$

where v(j) is the output voltage from the filter and T_D is the number of samples averaged.

Each entry in the digitized array is then quantized into A/D counts by using:

$$M(i) = v_D(i) / \Delta v \qquad 3.3.2$$

where Δv = # of volts/bit specified in the parameter table. The array M(i) contains the waveform which is quantized in both time and voltage.

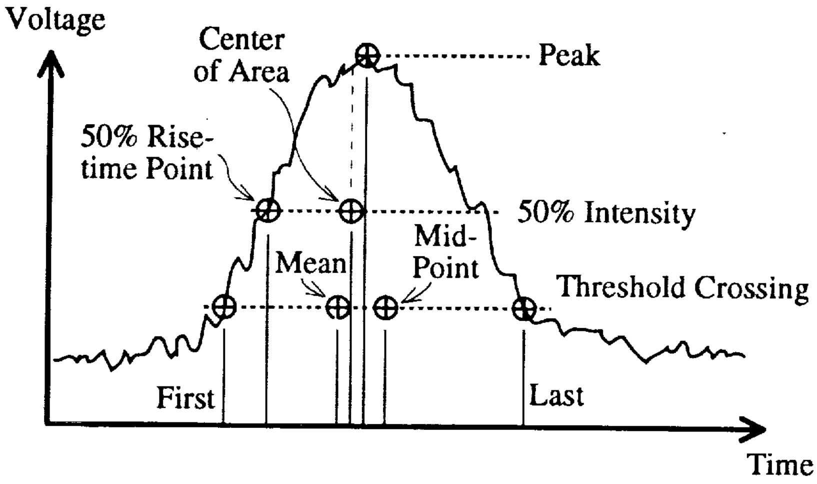


Figure 3.6. Digitized Waveform with Timing Estimators.

3.3.2 Timing Estimates

The fine timing estimates calculate the waveform fine timing points using the following formulas:

a. Center of Area (COA) - The COA index, **icoa**, is given by:

$$\sum_{i=I_1}^{icoa-1} M(i) < \frac{A}{2} and \sum_{i=I_1}^{icoa} M(i) \geq \frac{A}{2} \qquad 3.3.3$$

where:

icoa - is the index where the COA occurs in M(i)

A - is the area of M(i) between the threshold crossing points

I_1 - is the index of the first element in M(i) that exceeds the threshold crossing point.

An interpolation between indexes (i_{coa} & i_{coa}-1) is performed to determine the exact location in time, in bins, where the center_of_area occurs (See Section 4.4.3 step 7).

b. Peak - A quadratic fit, around the maximum value of digitized waveform, is used to calculate the interpolated index value where the peak of the waveform occurs.

$$Index(_{peak}) = \frac{-b}{2a} \qquad 3.3.4$$

where the constants a and b are calculated from:

$$a = \frac{(x_1 - x_2) * (i_1 - i_2) - (x_2 - x_3) * (i_1 - i_2)}{(i_1 - i_2) * (i_1 + i_2) * (i_2 - i_3) - (i_2 - i_3) * (i_2 + i_3) * (i_1 - i_2)}$$

$$b = \frac{(x_1 - x_2) - a * (i_1^2 - i_2^2)}{(i_1 - i_2)}$$

and:

$$\begin{aligned} i_1 &= i_{(\max)} - 1 & x_1 &= M(i_1) \\ i_2 &= i_{(\max)} & x_2 &= M(i_2) \\ i_3 &= i_{(\max)} + 1 & x_3 &= M(i_3) \end{aligned}$$

c. 50% Risetime Point - The index of the 50% risetime point, **icfd50+1**, satisfies the equations:

$$\begin{aligned} M(icfd50) &\geq (M(i_{\max}) - M_{TH}) / 2 + M_{TH} \\ M(icfd50 - 1) &< (M(i_{\max}) - M_{TH}) / 2 + M_{TH} \end{aligned} \qquad 3.3.5$$

where M_{TH} is the digitized threshold value.

d. Mean - The index of the Mean (or center of gravity) of the waveform is calculated by using:

$$Index(Mean) = \frac{\sum_{i=I_1}^{I_2} i \cdot M(i)}{\sum_{i=I_1}^{I_2} M(i)} \qquad 3.3.6$$

where

I_1 = index of first threshold crossing
I_2 = index of last threshold crossing

e. Midpoint - The index of the Midpoint of the waveform is calculated from:

$$Index(Mid) = \frac{I_1 + I_2}{2} \qquad 3.3.8$$

f. RMS Pulse Width - The RMS Pulse Width (in number of Δt elements) is calculated by using:

$$rms\_width = \sqrt{\frac{\sum_{i=I_1}^{I_2} M(i) \cdot i^2}{\sum_{i=I_1}^{I_2} M(i)} - mean^2} \qquad 3.3.7$$

3.4 TERGPH Design

TERGPH is used to recreate the terrain profile from the DIGITIZE subroutine's timing estimates. It also prepares the estimates to allow them to be plotted superimposed on the actual terrain profile. TERGPH is the last subroutine called.

TERGPH converts the round-trip time of flight timing estimates into terrain height estimates, and plots them on a graph of the terrain profile at the correct along-track distance. The along-track distances are plotted by placing the first estimate at a distance corresponding to one-half the beam's footprint. Every estimate after that is plotted at a distance corresponding to the velocity of the satellite.

Since each ray lies on exactly one terrain segment, the length of the beam's footprint in centimeters is equal to the number of rays in the beam. The number of rays in the laser beam is NP, where

$$\Delta\theta = \sum_{i=1}^{NP} dt_i$$

and $\Delta\theta$ is the laser divergence angle
$dt_i = \tan^{-1}(s_i/(Rsat - h_i)) - \theta_i$
$s_i = (Rsat - h_{i-1}) * \tan\theta_i + 0.01$meters.
θ_i is the off-nadir pointing angle of ray "i".

If X_i is denoted as the along-track distance of height estimate H_i, then

x_1 = NP/2 centimeters,

$x_i = x_1$ + vsat * (i-1) *100 centimeters,

and vsat is the distance in meters that the satellite moves between shots.

The height estimate H_i is computed from the timing estimate T_i by:

$$H_i = Rsat - \frac{c}{2} \cdot T_i \cdot \cos\theta_o$$

where:

Rsat is satellite altitude

c is velocity of light

θ_0 is the off-nadir pointing angle of the transmitter.

The simulator does not incorporate the effects of atmospheric refraction.

3.5 TERRAIN Program Design

TERRAIN generates the terrain profile data file used by the simulator. The terrain data file is in the format of (x,y, r). Here x is the linear distance along the surface track of the satellite (in cm, always starting at zero), y is the terrain height at location x, and r is the diffuse surface reflectivity at that point.

The terrain file divides x into 1cm segments. Each 1 cm segment has a constant height and constant reflectivity. The length of the segments was chosen to ensure that all photons in a single ray, which return from a segment, lie in a single 100 psec time bin.

In the current simulator, the terrain types produced are:

a. Flat
b. Uniform slopes
c. Single steps
d. Multiple steps
e. Ice terrains

The ice terrains are generated from sections of actual laser altimeter measurements over ice.

4.0 Simulator Implementation.

SIM, is comprised of a main FORTRAN program and four subroutines, SPACE_TIME, RECEIVER, DIGITIZE and TERGPH. The waveform array is passed between these programs and undergoes processing that changes both the array's contents (See Figure 1.2) and size. Figure 4.0 illustrates the changes that occur to the waveform array as each subroutine is executed by the simulator.

4.1. Main Routine

The main routine calls and coordinates the four simulator subroutines. It creates the output histograms and creates text output and most of the graphical outputs. If graphical outputs are selected, the graphs will be created with parameter 60, the graphlabel, as the label.

The main routine first opens and reads the parameter file, *PARAMETERS.SIM*. After reading the parameter file and initializing various variables, main prints the output text header.

Then the main routine names the parameters and opens files for the SPACE_TIME subroutine. Main also converts THETO from degrees to radians. SPACE_TIME is called with the appropriate parameters from *PARAMETERS.SIM*. After the subroutine has been called, main calculates the total number of photons by summing every element in the waveform array. The current shot number is printed, followed by important numerical outputs from the SPACE_TIME subroutine. If parameter 2 is negative or greater than or equal to the current shot number, the photon return is graphed using NCAR graphics.

The RECEIVER subroutine is called with parameter array values and the waveform and background noise output from the SPACE_TIME subroutine. The receiver energy is found by summing the array, and important numerical outputs are printed. If parameter 3 is negative or greater than or equal to the current shot number, the receiver output voltage is plotted.

DIGITIZE is called with the waveform output from RECEIVER and the appropriate parameters. The estimators are printed as text. If parameter 4 is negative or greater than or equal to the current shot number, then the digitized output is plotted.

After the three subroutines are called, the main routine updates the histograms. If the number of shots is greater than one, the main routine reruns the three subroutines once for each shot specified by parameter 36. The histograms are updated after each shot. If all shots have been fired, then, if parameter 5 is equal to 1, the main routine graphs the histograms of the estimators. If parameter 1 is greater than 0, the TERGPH subroutine is called to recreate data files of the actual and recreated terrain from the simulator output. Parameter 1 value indicates which of the 6 estimators to use. These terrains can be graphed using NCAR graphics routines.

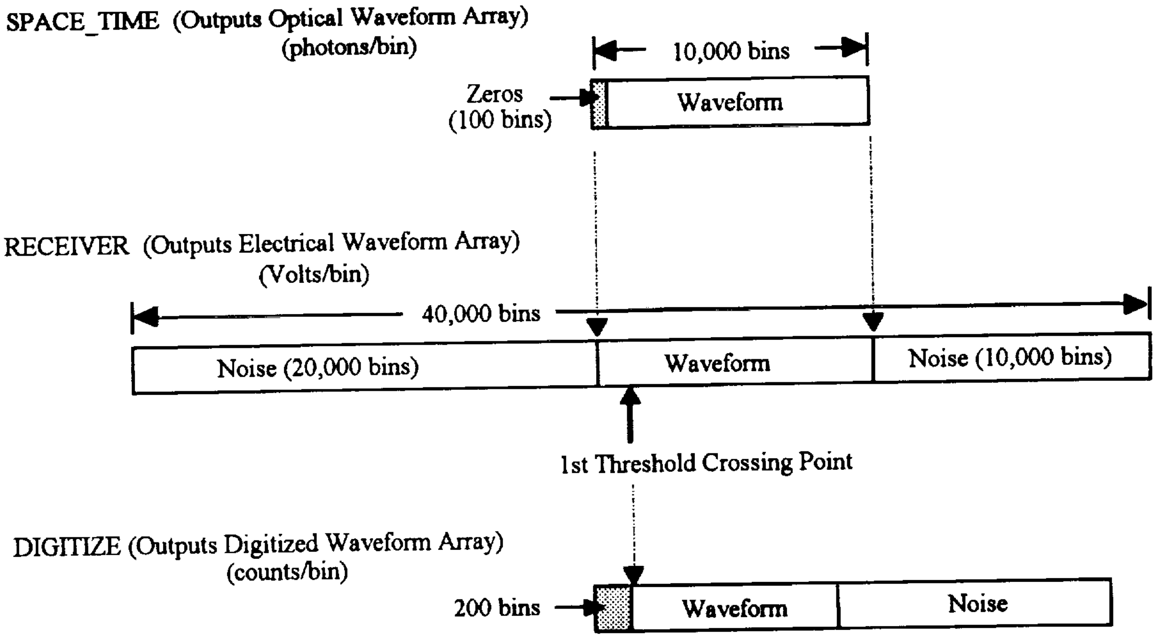


Figure 4.1. Waveform Array Representations.

4.2 SPACE_TIME Subroutine (Version 3.0)

The program reads the terrain from the terrain file corresponding to the entire spot size of the beam on the ground. The subroutine always begins at the first terrain record for shot #1. It computes $d\theta_i$ (the ray angular size) at each terrain segment. All segments are summed to obtain NP=n_θ (the number of rays). The minimum round-trip time of flight of all rays in the beam, T_{min} is next computed. 10nsec is subtracted from T_{min}, to start the range window 10nsec early.

The following is performed for i = 2,n_θ:

- Compute $d\theta_i$ from Section 3.1 [A].
- Compute the angle to this ray from the normal.
- Compute the slope at the terrain segment (x_i,h_i) associated with this ray. (Section 3.1 [C])
- Use the surface reflectivity of this segment (r_i) to compute the transmitted energy in this ray. (Section 3.1 [D])
- Perform a link calculation to obtain NS_i, $NBDOT_i$. (Section 3.1 [E])
- Compute T_i (round-trip time of this ray) from Section 3.1 [F] and place NS_i photons in the WAVE histogram at position J = (T_i - Tmin). See Figure 4.2 below.

Finally the program finds the maximum round-trip time of flight of all rays, Tmax, and sum up the background rate $NBDOT_i$. across all rays.

4.2.1 Space_Time Constraints

WHAT?	UNITS	WHY?
$10 \leq$ NHIST ≤ 1000	(elements)	Max DIMENSION is 10000 for all of the subroutines
1.D-10 $\leq$ DELTH	(radians)	Limited since it is used as divisor in SPACE_TIME
$100 \leq$ Rsat $\leq$ 2000.D3	(meters)	Constrained to make all returns from a given ray return within one 100 psec time bin.
$0 \leq$ THETO $\leq$ 0.175D0	(radians)	Negative angle isn't defined in SPACE_TIME Maximum limit is constrained by the DIMENSION on the number of rays
$2 \leq$ # rays $\leq$ MAXPEN		Computed from input data (see detailed description in ALGORITHM section).

4.2.2 Space_Time Constants

PI = π	3.141592654
LAMLIM	50.0 degrees (Lambertian scattering limit)
CVEL	299792500.0 m/sec. (velocity of light in vacuum)
H	6.625D-34 Joule-sec (Planck's constant)

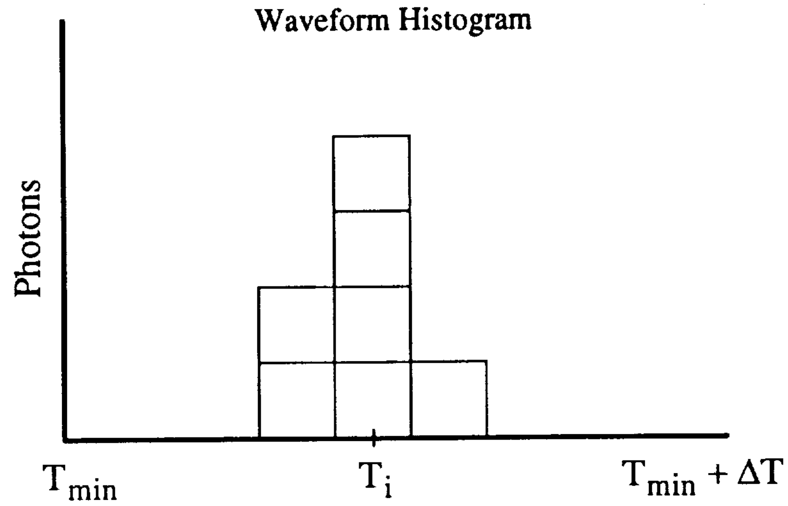


Figure 4.2. SPACETM Waveform in Time Domain.

4.2.3 Space_Time Inputs:

NHIST (I*4)	Max number of WAVE (time domain) bins allowable.
Rsat (R*8)	Orbital altitude from mean sea level (meters).
vsat (R*8)	Ground speed of laser spot (meters/shot).
WAVL (R*8)	Laser wavelength (meters).
THET0 (R*8) = θ_0	Off-nadir pointing angle from satellite to ground, defined from normal to center of pulse (radians).
DELTH (R*8) = $\Delta\theta$	Divergence of laser (full angle, in radians).
ITHFLG (I*4)	Beam intensity pattern option (1=Gaussian).
XEN (R*8)	Average transmitted laser energy (Joules per shot).
Arec (R*8)	Telescope receiver area (sq.meters).
TAUSYS (R*8)	System transmission (0 to 1).
TAUATM (R*8)	Atmospheric transmission (0 to 1).
SUNIR (R*8)	Solar irradiance (Watts/meter**3).
SUNF (R*8)	Solar illumination fraction (0 to 1).
RECFOV (R*8)	Receiver field of view (angular region).
B0 (R*8)	Receiver optical filter width (meters).
TEXT_FLAG (I*4)	Debug mode flag: 0 ==> no debug / 1 ==> output diagnostic messages to file.
NCAR_FLAG (I*4)	Debug mode flag: 0 ==> no debug / N>0 ==> output diagnostic data for plotting shot #N to file..

The subroutine also reads terrain data from a file which has been opened by the main calling subroutine as logical unit 10.

4.2.4 Space_Time Outputs:

Tmin (R*8)	TIU reading (start of WAVE histogram) in 2-way nanoseconds.
Tmin (R*8)	Longest round-trip time of all the rays in 2-way nanoseconds.
NBDOT (R*4)	Noise photon arrival rate (photons/sec).
WAVE (R*8)	Histogram of return times (waveform array). Each bin is 100 psec in width. Count of each bin is in photons.
TACT (R*8)	Actual round-trip time of flight in nanoseconds at the center of the beam.
HGTACT (R*8)	Actual terrain height in meters at the center of the beam.
IRETF (I*4)	Return flag indicating ERROR or END-OF-FILE, The response to all errors (except IRETF=3) is to return Tmin=Tmax=NBDOT=0 with WAVE=all zeros. -1 End of terrain file encountered. 0 Ok (expected return) 1 Error on terrain file read. 2 Number of calculated rays is out of range. 3 The angle (THETi) to one of the rays is 90deg or greater. 4 One of the following is out of limits: DELTH, Rsat, or THET0. 5 NHIST is out of range.

4.2.5 SPACE_TIME Diagnostics

The program can also write diagnostics to files on logical units 11 and 12. If TEXT_FLAG is non zero then on logical unit 11 the program will write 4 ASCII lines of text (two data, two header) for each ray:

LINE 1: header
LINE 2: ray#, height, reflectivity, time hist.bin#,
transmit energy of this ray, angular size of ray.
LINE 3: header
LINE 4: slope of terrain, angle from nadir to the ray,
#photons in this ray's part of transmit beam,
optical background noise rate see by this ray,
round-trip time associated with this ray's return.

If NCAR_FLAG is equal to N > 0 then at the Nth shot the time-domain waveform will be written to an ASCII file. This waveform will contain two lines at the beginning (one header, one data). The data line contains Tmin, Tmax, NBDOT, TACT, and HGTACT. The wave itself will be written one bin per line as: i, $WAVE_i$.

4.3 RECEIVER Subroutine (Version 4.1)

Called from the main routine, ***sim***, RECEIVER, process a photon waveform array created by SPACE_TIME and creates a new array representing an electrical detector response waveform which is passed to the DIGITIZE routine.

4.3.1 Constants/Parameters

Constants/Parameters Used in RECEIVER Subroutine

Index Number	Name	Description	units	nominal[1]	format[2]	module
20	NWID	Filter Width: 1 point = 100ps	100ps	10	I*4	RECEIVER
21	QE	PMT quantum efficiency	—	0.15	REAL*8	RECEIVER
22	ID	PMT dark current	amps	6.4E-12	REAL*8	RECEIVER
23	G	PMT multiplication gain	—	1.0E6	REAL*8	RECEIVER
24	RL	PMT load resistor	ohms	50.0	REAL*8	RECEIVER
25	TP/BAMT	Integration time/Bin Amount	sec	100.e-12	REAL*8	RECEIVER
26	F	APD excess noise factor	—	0.0065	REAL*8	RECEIVER
27	G	APD multiplication gain	—	194.0	REAL*8	RECEIVER
28	R	APD load resistor	ohms	22 000	REAL*8	RECEIVER
29	QE	APD quantum efficiency	—	0.35	REAL*8	RECEIVER
30	IBLK	APD bulk current	A	50.E-12	REAL*8	RECEIVER
31	IANS	Receiver Type Switch	—	1	I*4	RECEIVER
32	TR	APD Receiver Temperature	K	750.0	REAL*8	RECEIVER

Variable Names	Data Type	Size	Input/ Output	Source	Comments
WAVESIG1	REAL*8	10000	OUTPUT	SPACE_TIME SUB	Array containing received wave (photons/bin)
IWFMSIZE N	I*4	10000	INPUT	I/O	Number of elements in WAVE
NN	I*4	1	INPUT	DATA STMNT	Number of elements in RES array
NBDOT RBG	REAL*8	1	OUTPUT	SPACE_TIME SUB	Noise photon arrival rate (photons/sec)
ST	REAL*8	1	INPUT	DATA STMNT	Start time of signal in WAVE array always 0
TMAX SEND	REAL*8	1	OUTPUT	SPACE_TIME SUB	End TIU reading for wave 2-way (nsec)
RESP R	REAL*8	40000	OUTPUT	RECVR SUBROUTINE	Filtered output from detector of WAVE, with noise added
PARAMS	REAL*8	50	INPUT	FILE	Array containing input parameters for RCVR subroutine
THRESHOLD VT	REAL*8	1	OUTPUT	RECVR SUBROUTINE	Calculated threshold for waveform processing
IFLAG	I*4	1	OUTPUT	RECVR SUBROUTINE	Error flag for processing data, ERROR<0

4.3.2 Program Flow

1) Read input from PARAMETERS.SIM - 'receiver' initializes variables and constants with the values contained in PARAMETERS.SIM items 20-40.

2) Expanding the Signal Array - An array (WAVESIG1) whose length is determined by the SPACE_TIME subroutine is input to the RECEIVER subroutine. The output array from the subroutine (RESP R) is fixed at 40,000 time bins, representing 100 psec each. The RECEIVER subroutine places the input signal from the SPACE_TIME subroutine at position 20,001 in the expanded array and noise is added to all 40,000 array elements.

3) Add noise to the Response Waveform - As described in Section 3.2 equations 3.2.2 and 3.2.3, noise is added, Gaussian for an APD and Poison for a PMT, to each bin (40,000) of the detector response waveform.

4) Convert photo-electrons/bin to Volts/bin - RECEIVER converts the WAVESIG1 array from photo-electrons/Δt bin to Volts/Δt bin.

5) Apply Filter - 'Receiver' filters the detector response waveform by a square or Gaussian filter as specified in *PARAMETERS.SIM.*

6) Calculate Threshold Level - RECEIVER calculates the threshold level by finding the maximum value of the noise bins only, bins 1-20000 and bins 30001 to 40000.

4.4 DIGITIZE Subroutine (Version 1.3)

Called from the main routine, DIGITIZE, digitizes the output array from the 'receiver' and calculates the waveform timing estimators as described above in Section 3.4.

4.4.1 Parameters

The major variables and arrays used by the DIGITIZE routine are described in the following table.

Name	I/O Status	Description	Units
wfm_in	Input Output	Real 40,000 point array that represents the receiver/detector response data in terms of Voltage vs. Time. DIGITIZE processes wfm_in and **returns** an array that represents sampled values in digital Counts vs. Time. The digital counts range in value from 0 to 2^ (# of bits) - 1.	I-Volts O-Counts
length_in	Input	4 byte integer scalar representing the length of wfm_in[].	none
length	Output	4 byte integer scalar representing the length of the digitized array returned in wfm_in[]'s location. 'length' should always be less than or equal to length_in.	none
thresh	Input	double precision scalar representing the threshold value of the discriminator.	Volts
time-scale	Input	4 byte integer scalar representing the # of input bins (input bin is fixed at 100 ps/bin) contained or averaged into each output bin or sample.	100ps Bins
nbits	Input	4 byte integer scalar representing the # of A/D converter bits used in DIGITIZE, (nominally 16, See PARAMETERS.SIM item #57).	none
xmax	Input	8 byte real scalar representing expected max. value of the waveform and the max. of the A/D converter's input range. 'xmax' is used for scaling wfm_in in volts to xout in counts.	Volts
moffset	Output	4 byte integer scalar representing the offset pointer in bins from the start of wfm_in to the start of the output array (See Section 13).	100ps Bins
mcenterofarea	Output	4 byte integer scalar representing where the #1 'estimator', center of area index, occurs in the digitized waveform array wfm_in.	100ps Bins
mpeak	Output	4 byte integer scalar representing where the #2 'estimator', peak value index, occurs in the digitized waveform array wfm_in.	100ps Bins

mmultihit	Output	This parameter is not supported under this version.	
mcfd50	Output	4 byte integer scalar representing where the #4 'estimator', 50% CFD index, occurs in the digitized waveform array wfm_in. (units of bins)	100ps Bins
mmidpoint	Output	4 byte integer scalar representing where the #5 'estimator', midpoint of threshold crossings, occurs in the digitized waveform array wfm_in.	100ps Bins
meanx	Output	8 byte real scalar representing the returned mean value of the digitized waveform array wfm_in.	100ps Bins
rmswidth	Output	8 byte real scalar representing the returned rms width value (pulse width) of the digitized waveform array wfm_in	100ps Bins

4.4.2 Program Flow

The program flow of the DIGITIZE routine is given below:

1) Find the Location of First Threshold Crossing Point - DIGITIZE finds the first threshold crossing point in the wfm_in array by comparing the value of each wfm_in[i] to the threshold value thresh. When wfm_in [i] exceeds the threshold value DIGITIZE sets the first crossing point to [i].

2) Perform A/D Conversion (Sample) - DIGITIZE samples the raw data in wfm_in by time_scale to simulate digitizing the waveform with an integrating A/D converter. The simulated digitization begins at a starting point, referred to as 'start', which is set to 20ns (200 bins) before the first threshold crossing point. DIGITIZE then samples the waveform by averaging a number of input waveform bins (# = time_scale) to form one output bin as shown in Eqn 3.4.1.

The digitized (averaged) waveform data is stored in an array named xout, and the length of xout is stored in length. For each xout(i), dig_pointer represents the starting index of wfm_in that is being averaged. The value of dig_pointer is initialized to 'start', incremented by time_scale for each xout(i) and continues until the end of wfm_in is reached.

3) Transform Threshold (counts) - If no errors occurred in the digitization processes (2), then 'digitize' scales the threshold from volts to A/D counts by the following expression.

$$thres_{(counts)} = thres_{(volts)} * \left(\frac{2^{nbits}{}_{(counts)}}{x\max_{(volts)}} \right)$$

4) Convert digitized array from volts to counts - Translate the digitized array xout in volts to xtmp in A/D counts using Eqn 3.4.2. If any voltage values exceed the max. expected range of the A/D

(xmax) set those values to (xmax) before converting to counts.

5) Set first and last threshold crossings - Set/Determine the first and last threshold crossing points/indexes as they occur in xtmp. Since the value of 'start' (the starting point of xtmp) was equal to the first threshold crossing point in wfm_in minus 20ns (200 input bins), the first threshold crossing point in the digitized data xtmp is given by the following:

$$first_{(output\_bins)} = \frac{200_{(input\_bins)}}{time\_scale_{\left(\frac{input\_bins}{output\_bins}\right)}}$$

where:

last = i+1,

i > first and xtmp (i) < thresh

6) Calculate Area - With first and last, determine the waveform's area with the following expression:

$$area_{(counts)} = \sum_{i=first}^{last} xtmp(i)_{(counts)}$$

7) Calculate Center_of_Area (COA) - Calculate COA by determining the index (i_{coa}) of xtmp as shown in Eqn. 3.4.3. DIGITIZE then interpolates between indexes (i_{coa} & i_{coa}-1) using the difference in area of xtmp at i_{coa} and i_{coa}-1 to determine the exact location in time, in bins, where the center_of_area occurs.

$$x_{(coa)} = i_{coa} + \frac{\frac{area}{2} - sum}{xtmp(i)} \quad where \quad sum = \sum_{i=first}^{icoa} xtmp(i)$$

8) Calculate the peak - DIGITIZE finds the max. of xtmp and then calculates the peak of the digitized waveform, xtmp($i_{(peak)}$) with the expressions shown in Eqn. 3.4.5

9) Calculate the 50% Constant Fraction Discriminator (CFD) - DIGITIZE finds where the 50% CFD occurs as shown in Eqn. 3.4.6.

10) Calculate the Mean Index - DIGITIZE finds where the Mean Index occurs as shown in Eqn. 3.4.7.

11) Calculate the RMS Pulse Width - DIGITIZE finds the RMS Pulse Width using Eqn. 3.4.8.

12) Calculate the Midpoint - DIGITIZE finds where the Midpoint occurs using Eqn. 3.4.9

13) Correcting Timing Estimators - DIGITIZE calculates where the Center of Area, Peak, 50% CFD, mean, and Midpoint occur in the digitized waveform, xtmp. The estimator values are calculated from the start of xtmp and are given in terms of 100 ps bins. In order to make the estimator timing values consistent with timing reference or starting point of the other simulator routines, an offset factor is used to convert from digitize starting point, 'start', to the simulator starting point. The offset is given by the following:

$$\mathrm{offset}_{(100\text{ ps bins})} = \mathrm{firstcross} - 200$$

The estimator values were previously calculated in units of the output bins, where the value of the estimator occurs in xtmp. The timing estimators and RMS width are converted to a time position, in 100 ps bins, by the following formulas:

$$\mathrm{COA}_{(\text{inp bin})} = \mathrm{COA}_{(\text{out bin})} * \mathrm{time\_scale}_{(\text{inp bin/out bin})} + \mathrm{offset}_{(\text{inp bin})}$$

$$\mathrm{Peak}_{(\text{inp bin})} = \mathrm{Peak}_{(\text{out bin})} * \mathrm{time\_scale}_{(\text{inp bin/out bin})} + \mathrm{offset}_{(\text{inp bin})}$$

$$\mathrm{CFD50}_{(\text{inp bin})} = \mathrm{CFD50}_{(\text{out bin})} * \mathrm{time\_scale}_{(\text{inp bin/out bin})} + \mathrm{offset}_{(\text{inp bin})}$$

$$\mathrm{Mean}_{(\text{inp bin})} = \mathrm{Mean}_{(\text{out bin})} * \mathrm{time\_scale}_{(\text{inp bin/out bin})} + \mathrm{offset}_{(\text{inp bin})}$$

$$\mathrm{midpoint}_{(\text{inp bin})} = \mathrm{midpoint}_{(\text{out bin})} * \mathrm{time\_scale}_{(\text{inp bin/out bin})} + \mathrm{offset}_{(\text{inp bin})}$$

$$\mathrm{RMS\_W}_{(\text{inp bin})} = \mathrm{RMS\_W}_{(\text{out bin})} * \mathrm{time\_scale}_{(\text{inp bin/out bin})}$$

14) Store the digitized waveform - Store $\mathrm{xtmp}_{(\text{counts})}$ in the location of wfm_in.

4.4.3 Return Value

Status = 0 if all functions and processes were completed successfully.

4.5 TERGPH Subroutine (Version 1.5)

TERGPH reads the terrain file which has been opened by the main calling routine as logical unit 10. It determines the number of actual terrain points that it needs to plot, prior to plotting the first estimated height, by computing the number of rays in the first half of the laser beam.

TERGPH then reads and corrects the simulator's timing estimates as follows:

$$\mathrm{Tmeas}_i = \mathrm{TIU}_i - 2000 + \mathrm{Test}_i * \mathrm{TIMESCALE}/10.$$

where Test_i is the timing estimate from the DIGITIZE subroutine, and all terms units are nanoseconds. TIU is the round-trip time of flight from laser fire to the start of the SPACE_TIME window. Subtracting

2000 nsec from TIU gives the round-trip time of flight to the start of the DIGITIZE window. TIMESCALE is the number of bins in each DIGITIZE waveform element where each bin is equal to 100 psec. The last factor therefore converts Test from DIGITIZE units to nanoseconds. The subscript (i) represents the ith shot.

The terrain height is then computed from the corrected timing estimate, and this terrain estimate, an (x,y) Cartesian pair, is written to the file [TEREST.DAT] as $(XX_i, Hmeas_i)$. Here XX is the linear distance along the terrain track and Hmeas is the estimated height in meters above sea-level.

Every 10th point of the actual terrain data is written to file [TERACT.DAT]. The Cartesian coordinates for each record are (XX_i, HGT_i) where $XX_{i+1} - XX_i$ = 10 centimeters and HGT_i is the corresponding actual height for the given XX_i.

The file [TEREST.DAT] will contain "NSHOTS" number of points where "NSHOTS" is the number of shots fired. The file [TERACT.DAT] will contain the actual terrain covered from half a beam diameter prior to the first estimated terrain point to half a beam diameter after the last estimated point. The number of points then is approximately:

$$(vsat*100*NSHOTS + NP)/10$$

4.5.1 Tergph Inputs:

Test (R*8)	Array of estimated round-trip return times from waveform calculations (one entry per shot) in waveform digitize bin units.
NSHOTS (I*4)	Number of shots fired.
Rsat (R*8)	Orbital altitude from mean sea level (meters).
vsat (R*8)	Ground speed of laser spot (m/shot).
WAVL (R*8)	Laser wavelength (meters).
THET0=θ_0 (R*8)	Off nadir pointing angle from the satellite to the ground (radians).
DELTH=$\Delta\theta$ (R*8)	Divergence of laser (full angle / radians).
Tmin (R*8)	TIU reading from SPACETM in nsec. This is the start of SPACETM's window.
TIMESCALE (R*8)	DIGITIZE subroutines scale factor indicating number of SPACE_TIME bins used in each waveform digitize element.

4.5.2 Tergph Outputs:

IRETF (I*4)	Return flag... 0 ==> OK 1 ==> read or write error 2 ==> End-of-file

4.6 TERRAIN Program

The terrain file is a binary direct access file (16 bytes per record) which contains a height and a reflectivity for every 1 centimeter of linear distance. The relative linear distance, x, is not explicitly given, but is implicitly determined from the record number; record "i" (where i starts at 0) represents "i" centimeters of along-track distance.

The program has two major output options:

1) simulated terrain
2) terrain generated from actual laser altimeter ice data located in the file [ICEX.DAT].

A simulated terrain can consist of a slope or alternating slopes, a single step or a series of steps. The operator can chose the length of each section (in meters).

The terrain generated from the ice data can be chosen from anywhere in the ICEX.DAT file by selecting the starting record number.

The constraints on the inputs are that the slopes must be between +85 and -85 degrees, the reflectivity must be between 0 and 1 and the number of terrain points is arbitrarily limited to 100,000.

In this version, the reflectivity is a constant within the terrain file.

4.6.1 Inputs:

Operator entries and [ICEX.DAT] file.

4.6.2 Outputs

File of terrain data with operator selected name. The terrain file is a binary file containing one 16 byte record for each terrain point (8 bytes of floating point height and 8 bytes of floating point reflectivity).

4.6.3 Ice Terrain

A sample file of ice topography [ICEX.DAT] was furnished by Bob Swift/NASA Goddard-WFF as Airborne Oceanographic Lidar (AOL) data file MAY20A.DAT. This data was taken on May 20, 1987 in a NASA/WFF P-3 mission conducted north of Greenland. The measurements of ice topography were referenced to GPS and measured with the AOL's laser altimeter in the fixed nadir viewing mode. This data set was used in the simulator by converting the data from the Latitude-Longitude-Height reference frame to an Along_Track-Height reference frame (i.e. the 2-dimensional GLAS reference frame). The FORTRAN format for the [ICEX.DAT] file is:

```
XX,HGT ---> (1X,F10.2,1X,F10.4)
```

The [ICEX.DAT] file contains 128,000 records which span an along-track distance of ≈110 km (see Figure 4.3). The separation between recorded heights is approximately 1 meter and varies from point to point. The file statistics are:

	Latitude	Longitude	Height(m)
min.	84.0784	-17.2826	-0.60
max.	84.4616	-11.6655	4.90

To generate a terrain file with 1-centimeter-spaced data from the approximately 1-meter-separated ICEX.DAT data, the program fits a straight line to two ICEX.DAT points and then uses this line to interpolate between the two points.

Treating the height data as a probability function has allowed us to classify this large file into segments of varying roughness. We computed a mean and standard deviation (μ,σ) for each 100 point segment (group) and then divided the data into the 5 classes of roughness as shown in the table below and the following Figure 4.4.

	Fee Class	RMS roughness σ-range (m)	% of file in class
SMOOTHEST	1	0.00 - 0.25	62.7%
	2	0.25 - 0.50	28.2%
	3	0.50 - 0.75	7.2%
	4	0.75 - 1.00	1.3%
ROUGHEST	5	1.00 - 1.25	0.6%

It is interesting to note that ~90% of the ice surface sampled has an rms roughness of <0.5m.

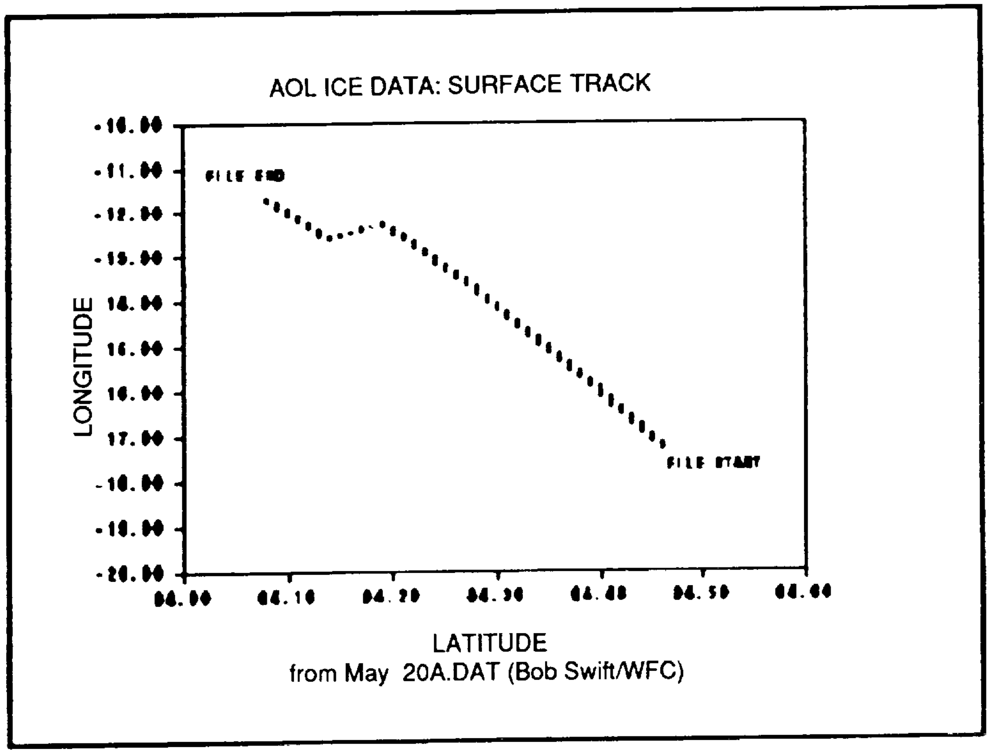


Figure 4.3. AOL Ice Data: Surface Track.

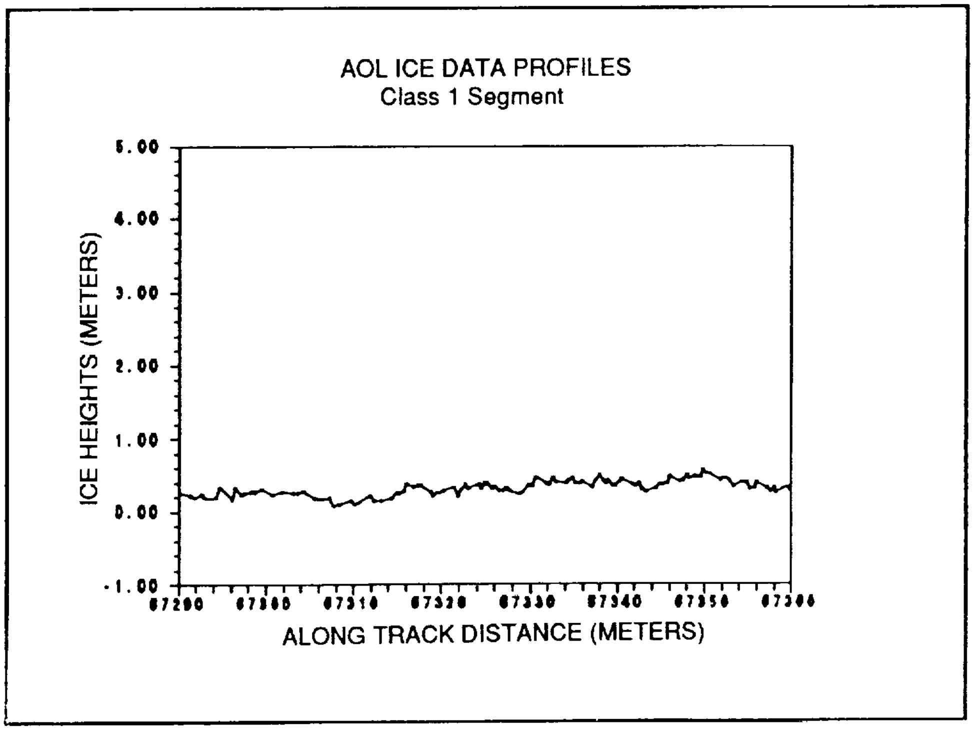


Figure 4.4A. AOL Ice Data Profiles, class 1 segment.

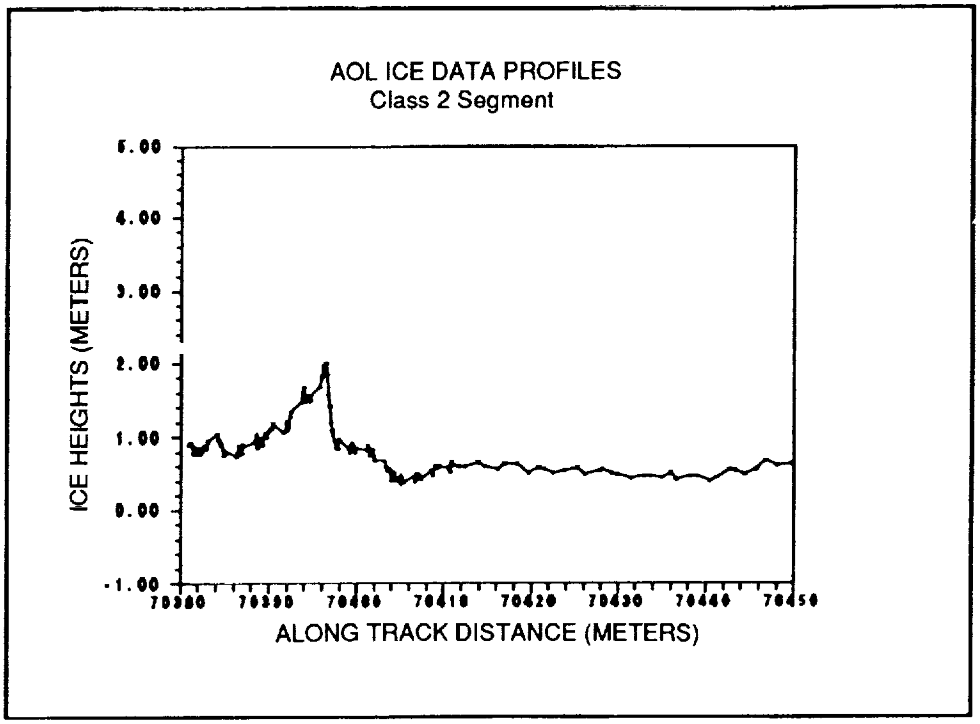


Figure 4.4B. AOL Ice Data Profiles, class 2 segment.

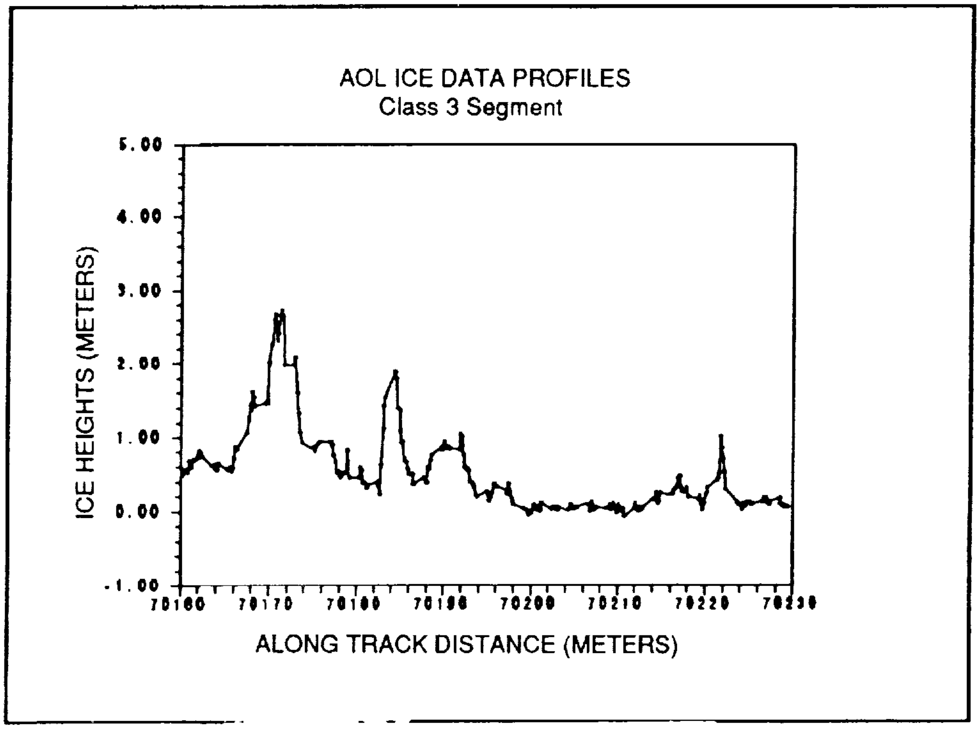


Figure 4.4C. AOL Ice Data Profiles, class 3 segment.

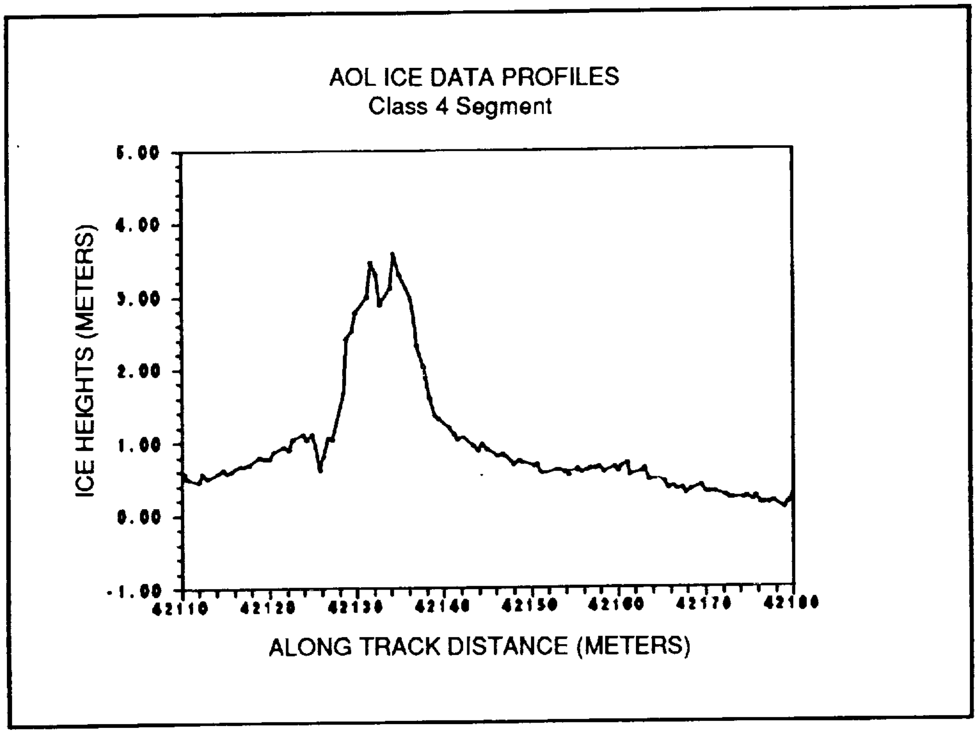


Figure 4.4D. AOL Ice Data Profiles, class 4 segment.

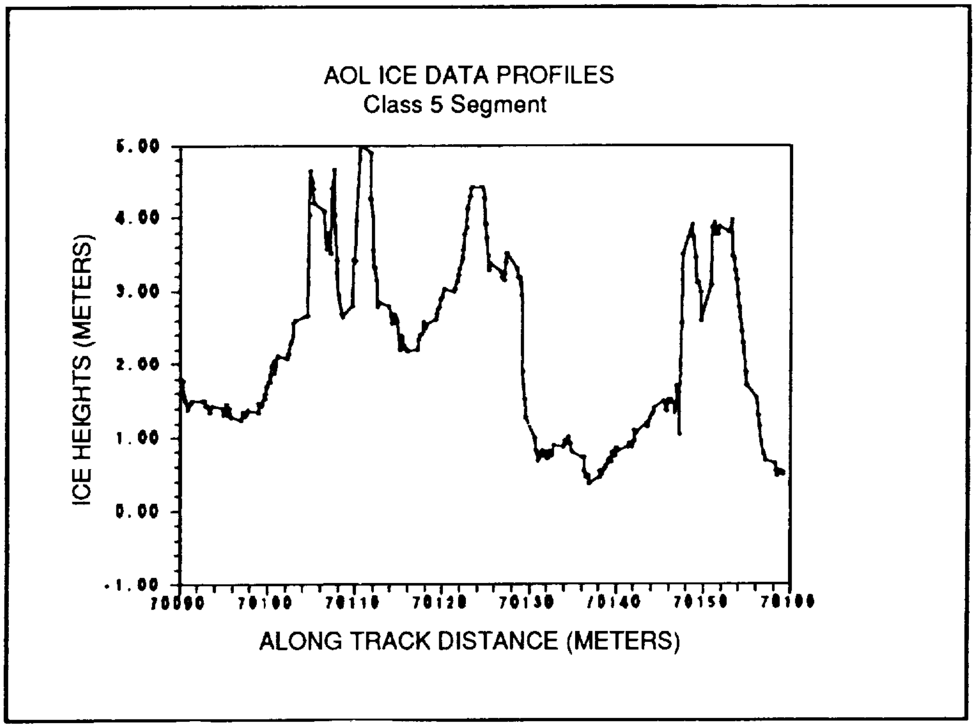


Figure 4.4E. AOL Ice Data Profiles, class 5 segment.

5.0 Computer Requirements

The Goddard Laser Altimetry Simulator is written specifically for use on the Sun SPARCstations 1+ or later version. Software packages including the ***mctrsum*** screen utility, NCAR graphics[1] and its associated postscript drivers are used to produce graphical outputs. The amount of available memory on the workstation determines several factors, including the execution time of simulations, the maximum size of the terrain files, and the number of simulations that may be run simultaneously. This simulator was developed on a SPARCstation which has 40 MB of memory.

The simulator requires the following executable and data files:

FILE NAME	SIZE	DESCRIPTION	VERSION
For simulator execution:			
sim	360448	Simulator Executable	3.0
PARAMETERS. SIM	≈500	Parameter File	
[terrain file]	variable	Terrain File	
i.e. *terrain.dat*	160016		
Additional Files:			
NOMINAL.SIM	507	Nominal Parameters	
terrain	163840	Terrain Generator	2.2
icex.dat	2048000	Real Ice Data	
param_edit	155648	Parameter Editor	2.1
NAMES.DOC	2104	**param_edit** Data	
mpost	148	Hardcopy Batch File	
mpost.cmd	16	mpost Auxiliary File	

These files require approximately 3.15 MB of disk space.

Note: Executable files names are printed in boldface italics, while data files are printed in normal italics. In addition, UNIX systems are case-sensitive, so file names must be entered as shown.

---

[1] The NCAR graphics package is a product of the National Center for Atmospheric Research, Boulder, Colorado. A version for the Sun workstation is available from MINEsoft, Ltd, 1801 Broadway, Suite 910, Denver, CO 80202-3837. (303) 292-6449.

6.0 How to Use the Simulator

This guide is intended for users that have a working knowledge of UNIX, the NCAR graphics package and the Sun SPARCstation.

Four steps are required to use simulator:

A) Create or edit the parameter table.
B) Create or edit a terrain file.
C) Run ***sim***, the simulator.
D) Generate and Analyze the outputs.

6.1 Creating/Editing the Parameter Table

The parameter file values determine the inputs to the simulator. It must be in the directory from which the simulator, ***sim***, is run and must be named *PARAMETERS.SIM*. Copies of the parameter file may be created under different names for runs of various test cases, but these files must be renamed *PARAMETERS.SIM* at run time in order to be recognized by the simulator. A copy of the nominal parameter table is included in the next section.

6.1.1 Parameter Listing

Each line in the following parameter file listing corresponds to a variable. Each of the user-variable parameters has a definition, sample test value, and useful range of values. For a more detailed description of each of these parameters, please see Section 4. For more information on use of each of these parameters, see Section 3 and 4. Some variables can be changed by the users, some are fixed, and some values are not used.

PARAMETER TABLE

Parameter Name:	Tested Range:	Nominal Value:
1. TERGPH Terrain Recreation:	0 = none n = estimator "n" where: n = 1..6 1 = Center of Area 2 = Peak 3 = Not used 4 = CFD_50% 5 = Midpoint 6 = Mean	0
2. SPACE_TIME Graph Option: |	(-1 = all shots graphed)	0
3. RECEIVER Graph Option: >	(0 = no shots graphed)	0
4. DIGITIZE Graph Option: |	(n = 1..n shots graphed)	0
5. Histogram Output:	1/yes 0/no	0
6. - 19. Unused.		
20. Filter width (100 psec units):	1 - 100	20
21. PMT quantum efficiency:	0.0 - 1.0	0.15
22. PMT dark current (amps):	0.0 - 1.0	6.4E-12
23. PMT multiplication gain:	1.E3 - 1.E10	1.0E6
24. PMT load resistor (ohms):	10.0 - 90.0	50.0
25. Fixed Value.		100.E-12
26. APD excess noise factor:	10.E-6 - 0.1	0.0065
27. APD multiplication gain:	1.0 - 1000.0	194.0
28. APD load resistor (ohms):	2.2E3 - 220.E3	22.0E3
29. APD quantum efficiency:	0.0 - 1.0	0.35
30. APD bulk current (amps):	0.0 - 1.E-9	50.E-12
31. Receiver type switch:	0/PMT 1/APD	1
32. APD detector preamp noise temp. (K):	500.0 - 1000.0	750.0
33. - 35. Unused.		
36. Number of Shots:	1 - 2E9	100
37. Fixed value.		10,000
38. Fixed value.		40,000

39. Fixed value.		40,000
40. Orbital altitude (m):	100.0 - 2.E6	705 000
41. Ground speed of shots (m/shot):	0.0 -	0.0
42. Laser wavelength (m):	100.E-9 - 10.E-6	1064.E-9
43. Laser pointing angle (deg):	1.E-7 - 10.E-3	0.0
44. Full angle laser divergence (rad):	0.0 - 0.175	1.E-4
45. Fixed value.		1
46. Transmitter laser energy (J):	0.0 - 100	100.E-3
47. Telescope receiver area (m^2):	1.E-4 - 10.	0.6
48. System transmission:	0.0 - 1.0	0.5
49. Atmospheric transmission:	0.0 - 1.0	0.5
50. Solar spectral at laser wavelength irradiance (W/m^3):	0.0 - 1.E12	0.6E9
51. Solar illumination factor:	0.0 - 1.0	1.0
52. Receiver field of view diameter (rad):	1.E-6 - 1.E-2	0.00025
53. Optical filter width (m):	10.E-12 - 1.E-6	2.E-9
54. Diagnostic use only.		0
55. Diagnostic use only.		0
56. Time scale (input/output bins):	1 - 400	3
57. Number of bits in A/D converter (bits):	2 - 16	16
58. Maximum range of A/D converter (volts):	0.1 - 200.0	65.536
59. Name of terrain file:	[any valid filename]	TERRAIN.DAT
60. Label for graphical output:	[up to 40 characters]	

6.1.2 Editing the Parameter Table using param_edit

The recommended way to edit the parameter table is with the simulator's utility program called ***param_edit***..

It is run by typing: ***param_edit***

The user is first prompted to enter the name of the parameter file to be edited. This parameter table must already exist, but may be saved under a different name when exiting. For example, the *NOMINAL.SIM* parameter table could be edited and then saved as *PARAMETERS.SIM* when finished editing. ***param_edit*** displays the parameters in four groups, by the portion of the simulator they affect: main, SPACE_TIME, RECEIVER, and DIGITIZE. The user has the options to edit or accept a set of parameters, exit ***param_edit*** (with or without save), or to reset to the top of the parameter listing. The program is designed to be self-explanatory and the available options are displayed at every point.

When saving the parameter file, the user has the option to save under a name other than *PARAMETERS.SIM*, in order to allow creating multiple parameter tables for different simulation cases. The user should rename the file *PARAMETERS.SIM* before running the simulator.

The simulator's graph is generated using NCAR graphics routines. These require spaces in graphlabel to be entered as the underscore character. Also, the characters available for NCAR graph labels are limited; refer to NCAR documentation for specifics.

6.1.3 Editing the Parameter Table using text editor

The parameter table may be edited using a simple text editor, such as ***vi***.. Familiarity with ***vi*** is essential when exercising this option. When using an editor, be certain to retain the basic format of the parameter table. Each line consists of a two-digit integer field specifying the parameter number, followed by two spaces, followed by the parameter value in any number format. Refer to the section above for limitations on possible entries for graphlabel.

6.2 Terrain Files

A terrain file must be created for the simulator. It must be in the same directory as the simulator, ***sim.***

6.2.1 Naming Requirements

The terrain file must be given the same name as is specified by parameter 59 in the *PARAMETERS.SIM* . The usual name for this file is *TERRAIN.DAT*.

6.2.2 Size of Terrain Files

The terrain file must be sufficiently long to accommodate all of the laser footprints required. Each element in the terrain file represents 1 cm of along-track terrain. A terrain file of 100,000 terrain points requires memory of approximately 1.6 MB, while 20,000 terrain points require 320 KB of memory. The present maximum terrain file size is limited to 100,000 points by the terrain generation file.

The required size of the terrain file in points is given approximately by:

$$N_{POINTS} = (100 \text{ x VSAT x } N_{SHOTS} + N_{\theta}) \text{ points}$$

where N_{θ} is the size of the footprint in centimeters, VSAT is the ground speed in meters/shot (parameter 41), and N_{SHOTS} is the number of shots (parameter #36). N_{θ} is given by:

$$N\theta = \frac{THDVRG}{\tan^{-1}\left(\frac{S}{RSAT - H1}\right) - THETO}\ centimeters$$

where

$S = (RSAT - H_2) \times \tan(THETO) + 0.01$ meters

H_1 is the 1st terrain height in meters

H_2 is the last terrain height in meters

THDVRG is the divergence of the laser (parameter #44)

RSAT is the altitude of the spacecraft (parameter #40)

THET0 is the laser off-axis pointing angle (parameter #43)

Using flat terrain with nominal parameters, and while pointing at nadir (pointing angle of 0 degrees) the footprint size is 70.5 meters. Thus, the minimum number of terrain points required:

$$N_{POINTS} = 100 \times 0 \times 1 + 7050 = 7050 \text{ points.}$$

This produces a terrain file of approximately 113 KB. If the off-nadir pointing angle were 10 degrees, a terrain file containing 7252 points would be required for a single shot, since the footprint size is 72.52 meters.

If the satellite altimeter is fixed (ground speed = 0), only a single footprint's worth of terrain is necessary. However, if multiple shots are specified and the satellite is moving, a greater number of points is required.

6.2.3 Creating the Terrain file

This section gives an overview of the ***terrain*** program. For a detailed description of how ***terrain*** generates terrain files, see Sections 3.5 and 4.6.

The ***terrain*** program first prompts for an output file. This file name may be any valid UNIX file name and may include a path. This name must match parameter 59, the terrain file name, in the appropriate *PARAMETERS.SIM* file. Note that if a file with the same name specified already exists, ***terrain*** will not be able to create a terrain file. So, to create a new *TERRAIN.DAT* if one already exists, delete the original *TERRAIN.DAT* before running ***terrain***.

The next input is surface's diffuse reflectivity which has a value from 0.0 to 1.0. The simulator cannot process surfaces with spectral components.

The user is given the option to generate a sloped or stepped terrain, or to use data from measured topography.

- If the sloped terrain option is chosen, the program prompts for the terrain slope in degrees. Alternating slopes are also available.

- If the step terrain option is chosen, the user is prompted for the step size and the plateau length. There is also an option to generate only one single step.

- Measured topography may be used. If this option is chosen, ***terrain*** will request a starting record for the data file *icex.dat*. For more information on this option, see Section 4.6.3.

The last parameter is the number of terrain points to generate. 20,000 points are usually sufficient, depending on the parameters and terrain conditions chosen. The maximum value of terrain parts is 100,000.

6.2.4 Example Terrain Creation

An example of terrain creation using ***terrain*** is given in this section. The terrain will be a flat plain which reflects 50% of the incident light. The file will be named *TERRAIN.DAT*. The generation program is run by typing

terrain

at the UNIX prompt. Upon being prompted for a file name, enter:

TERRAIN.DAT

For 50% surface diffuse reflectivity, at the prompt enter: 0.5

The terrain for this run will be flat, which will return an impulse response. So, at the sloped/stepped/real terrain prompt, enter: 0

terrain will now prompt for a starting surface height. Enter: 0

The terrain slope for a linear, level slope, in degrees, is simply: 0

Since we want a continuous linear slope, without any variation, at the alternating slopes prompt enter: n

Finally, choose the number of terrain points. This value must be large enough to cover all footprints. For this example, use: 20000

Upon entering this value, ***terrain*** will generate the requested terrain file. This example takes approximately 13 seconds to generate on the SPARCstation 1+.

6.3 Running the Simulator

If a proper *PARAMETERS.SIM* file has been created and a terrain file exists whose name matches that in the *PARAMETERS.SIM* file (parameter 59), then the simulator is ready to run. Ensure that ***sim***, *PARAMETERS.SIM*, and the terrain file are in that same directory or path.

The simulator is run by typing *sim*.

Run-time Options

There are no run-time options. All simulator parameters and output options are controlled by *PARAMETERS.SIM*. See the next section for a description of possible outputs, as determined by the parameter table, and redirection of outputs.

Execution Time

Using nominal values but with only a single shot and with the attached nominal *PARAMETERS.SIM* and a flat terrain, the simulator takes approximately 20 seconds to execute on the NASA Goddard SPARCstation 1+ (eib1).. Changing parameter values can significantly alter the execution time of the simulator. For instance, using a filter width of 100 instead of 10 in the RECEIVER subroutine increases the execution time to 50 seconds.

6.4 Simulator Outputs

As it is run, the simulator produces both text and graphical output. The text output includes calculated values for each shot, are sent to the display device, by default the monitor. These are the only numerical outputs from the simulator.

6.4.1 Description of Text Outputs

The outputs are defined below. More detailed descriptions of each of the outputs are given in Section 4. The output from the nominal case with the flat terrain created above follows this section.

The first time gives the version of the simulator being run and the date of this version.

SHOT# = Specifies the shot number for the text.

DATE EXECUTED = Gives date and time of the simulation.

SPACE_TIME Subroutine Outputs:

ITREF = Return flag indicating error if non zero.

TACT = Actual round-trip time of flight in nsec.

(TACT - TMIN) = Return offset from start of waveform.

TMIN = TIU Reading / to the start of the waveform window.

RECEIVER Subroutine Outputs:

NBDOT = Number of background photons in counts/sec.
SIGNAL PHOTONS = Number of total signal photons in counts.
IFLAG = RECEIVER error flag (zero - ok).
THRESHOLD = Threshold voltage level of the receiver.
AREA = Area underneath time response of output voltage.

DIGITIZE Subroutine Outputs:

The following are the return time estimators. Values are in nanoseconds. Add 2 μsec to compute the timing estimate from start of waveform window.

OFFSET = offset of the start of the digitizer window to the start of the waveform window.
CENTER OF AREA = Center of area of the digitized waveform.
PEAK = The peak of the digitized waveform.
MULTIHIT = The peak of the curve calculated by waveform analyzer.
CFD 50% = The 50% CFD of the digitized waveform.
MIDPOINT = The midpoint of the digitized waveform.
MEAN OF PULSE AT - Location of mean on digitized waveform.
RMS MEAN OF PULSE - Mean of the digitized waveform.
LABEL - Graphlabel to ensure proper text-graph matching.

6.4.1.a Redirecting the Text Output

The text output may be redirected into an output file using the pipe command after ***sim***.

e.g.. `sim >nominal.out`

To suppress text output, the output may be piped into the null device:

```
sim >/dev/null
```

6.4.2 Graphical Outputs

The simulator's graphical outputs are determined by parameters 1-5 in *PARAMETERS.SIM*. Each graph produced is labeled with the graphlabel, parameter 60, and with the shot number. The simulator places the NCAR graphs in a meta file named *sim.met*. This meta file must be renamed if future reference is desired, since each time the simulator is run a new *sim.met* is created. For detailed information on the graphs, see Section 6. The parameters' effects are as follows:

1. TGR: TERGPH Terrain Recreation
This option creates files that can later be plotted of the original terrain and recreated terrain. The estimator used in the recreated terrain file is selectable; zero here implies no terrain recreation will occur.

2. STGR: SPACE_TIME Graph Option
This option produces graph(s) of the output photon count of the SPACE_TIME subroutine for the first STGR shots, if STGR > 0, or for all shots if SGTR=-1.

3. RGR: RECEIVER Graph Option
This option produces graphs of the output voltage signal of the RECEIVER subroutine for the first RGR shots, if RGR > 0, or for all shots if RGR=-1.

4. DGR: DIGITIZE Graph Option
This option produces graphs of the digitized signal of the DIGITIZE subroutine for the first DGR shots, if DGR > 0, or for all shots if DGR=-1.

5. HGR: Histogram Output
Choosing this option (setting HGR=1) causes the simulator to create histograms of all of the fine tuning estimators values.

6.4.2.a NCAR Graphical Output

The NCAR meta files generated by the simulator may be viewed using the

```
mctrsun filename
```

command. The graphs of the most previous run are always stored under *sim.met*. To view the graphs from the previous run, type

```
mctrsun sim.met
```

mctrsun displays the graphs in the order they were generated. This utility is mouse driven, with a menu popping up in the lower left-hand corner when the mouse is clicked. If 'ERASE' is chosen, the current graph is erased and the next graph is plotted. If 'OVERLAY' is chosen, the next graph is plotted over the current one. ***mctrsun*** also has the option to 'ZOOM' into a portion of the graph.

6.4.3 Hardcopy Outputs

The NCAR utility ***mctrpost*** converts the NCAR meta file to a postscript file which may be printed on any laser printer with a postscript driver. The format for this command is

```
mctrpost [meta file] >[postscript file]
```

For example:

```
mctrpost nominal.met >nominal.ps
```

creates a postscript file, named *nominal.ps* of the NCAR graphs saved under the filename *nominal.met.*

If the text output has been redirected into a text file, then the graphical and text outputs may be combined, as detailed below.

A simple file conversion utility, ***mpost***, has been included which uses the ***vi*** editor to help consolidate the output. In order to use***mpost***, the simulator output must have be redirected into a text file with a *.out* extension. ***mpost*** concatenates the text and graphical outputs into one postscript file, using the ***pstext*** command. It then removes the page break from the text file by means of a batch-level***vi*** edit. The output postscript file is given a *.ps* extension. ***mpost*** will also save a copy of the NCAR meta output under a *.met* extension. The usage format is:

```
mpost [text output filename without extension]
```

For example, if the output has been redirected into a file named *nominal.out,* typing

```
mpost nominal
```

will cause ***mpost*** to produce a postscript file called *nominal.ps* of the graphical output with the text output above the first graph. A copy of the meta file, named *nominal.met,* will also be saved for that run.

To run ***mpost***, both ***mpost*** and *mpost.cmd* must be in the current root or path. *mpost.cmd* is a file containing the commands that ***mpost*** uses to edit the text output file. ***mpost*** is simply a batch file accessing postscript conversion utilities and may be easily edited to suit particular applications.

The text and graphical outputs in Section 7.0. were generated using the above technique. The text output and the NCAR meta file were combined using ***mpost***, as described above.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
January 1994

3. REPORT TYPE AND DATES COVERED
Technical Memorandum

4. TITLE AND SUBTITLE

Laser Altimetry Simulator, Version 3
User's Guide

5. FUNDING NUMBERS

Code 924

6. AUTHOR(S)

James B. Abshire, Jan F. McGarry, Linda K. Pacini, J. Bryan Blair,
and Gregory Elman

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Experimental Instrumentation Branch, Code 924
Goddard Space Flight Center
Greenbelt, Maryland 20771

8. PERFORMING ORGANIZATION REPORT NUMBER

94B00039

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

National Aeronautics and Space Administration
Washington, D.C. 20546–0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

NASA TM–104588

11. SUPPLEMENTARY NOTES

Gregory Elman: Science Systems and Applications, Inc., Lanham, MD.

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited
Subject Category 46
Report available from the NASA Center for AeroSpace Information, 800 Elkridge Landing Road, Linthicum Heights, MD 21090; (301) 621–0390.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

A numerical simulator of a pulsed, direct–detection laser altimeter has been developed to investigate the performance of space–based laser altimeters operating over surfaces with various height profiles. The simulator calculates the laser's optical intensity waveform as it propagates to and is reflected from the terrain surface and is collected by the receiver telescope. It also calculates the signal and noise waveforms output from the receiver's optical detector and waveform digitizer. Both avalanche and photodiode and photomultiplier detectors may be selected. Parameters of the detected signal including energy, the 50–percent risetime point, the mean timing point, and the centroid can be collected into histograms and statistics calculated after a number of laser firings. The laser altimeter can be selected to be fixed over the terrain at any altitude. Alternatively, it can move between laser shots to simulate the terrain profile measured with the laser altimeter.

14. SUBJECT TERMS
Laser Altimetry, Laser Ranging, Laser Remote Sensing, Ice Sheets,
Earth Observing System (EOS) Platforms, Surface Measurement

15. NUMBER OF PAGES
70

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
Unlimited