5-16-1990

3-D longwave infrared synthetic scene simulation

Eric H. Shor

3-D Longwave Infrared Synthetic Scene Simulation

by
Eric H. Shor

Rochester Institute of Technology
Center for Imaging Science

May 16, 1990

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in the Center for Imaging Science in the College of Graphic Arts and Photography of the Rochester Institute of Technology

CENTER FOR IMAGING SCIENCE
COLLEGE OF GRAPHIC ARTS AND PHOTOGRAPHY
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

M. S. DEGREE THESIS

The M. S. Degree Thesis of Eric H. Shor
has been examined and approved
by the thesis committee as satisfactory
for the thesis requirement for the
Master of Science Degree

Dr. John R. Schott, Thesis Advisor

Mr. Carl Salvaggio

Dr. Mark Fairchild

THESIS RELEASE PERMISSION FORM

ROCHESTER INSTITUTE OF TECHNOLOGY
COLLEGE OF GRAPHIC ARTS AND PHOTOGRAPHY
CENTER FOR IMAGING SCIENCE

3-D Longwave Infrared Synthetic Scene Simulation

I, Eric H. Shor, hereby grant permission to the Wallace Memorial Library of R.I.T. to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.

Date May 16, 1990

Abstract

A technique for thermal infrared (8-14 μm) synthetic image generation (SIG) was demonstrated that yields improved radiometric accuracy. This process utilizes the LOWTRAN 6 atmospheric propagation model and computer graphics ray-tracing techniques. A scene is created by placing faceted objects into world coordinates with rotation, translation, and scaling parameters. Each facet is assigned a material index and temperature. This index points to angular emissivity data for that material. LOWTRAN 6 can incorporate a sensor response function when calculating data files for the atmospheric transmission, upwelled and downwelled radiances, and temperature-to-radiance conversions. Ray-traced imagery is generated and discussed. The image is then further processed using convolution to represent the modulation transfer function of the imaging system. The final infrared synthetic image is then compared to an actual thermal image. An average apparent temperature difference of 2.50°C is reported with a 1.52°C standard deviation. These temperatures fall within predicted error analysis limits.

Acknowledgements

I would like to thank a few special people who contributed to this work. Without their assistance, it may have never been completed.

Tom Servos

James Warnick

Carl Salvaggio

Donna Rankin

Dedication

Carl, this is all your fault...

Table of Contents

List of Figures

List Of Tables

1.0 Introduction

Over the last 25 years, the need has developed for artificially created images in such applications as animation, computer-aided design and manufacturing (CAD/CAM), and flight simulation. Most of these applications have been associated with the computer graphics community. The algorithms for creating such imagery are collectively known as Synthetic Image/Scene Generation (SIG). All of these techniques have been applied in the visible part of the electromagnetic spectrum.

There are other bands (windows) in the electromagnetic spectrum used for imaging. The longwave infrared (LWIR) band 8-14 μm is used to image at night. This region is also referred to as thermal IR because object's temperature and material characteristics dominate the measured radiance. The unique properties of thermal IR imaging ensure its use will grow, and that IR SIG will be needed. The factors to consider in the thermal IR are different than those in visible wavelengths, making it a challenge to produce radiometrically accurate synthetic IR images.

In this thesis it was demonstrated that a radiometrically accurate synthetic image can be generated by combining the techniques of IR SIG, atmospheric modeling, and ray-tracing. Objects (represented as facets) are placed into a scene. DIRTRAN (which is interfaced to the LOWTRAN 6 atmospheric propagation model) provided the synthetic image generator with atmospheric and radiometric information. A 2-D image was created by tracing flux paths from the sensor into the scene. Angular emissivity data were incorporated in the determination of target and background radiances.

Robustness of the simulator was demonstrated by the creation of complex image sequences depicting a vehicle on a road with trees in the background. These image sequences highlighted a few of the parameters that can be varied in the creation of synthetic images.

This technique was radiometrically tested by creating a simple outdoor scene consisting of simple objects. Thermistors were placed around the scene to get actual object temperature values. These temperature values were then entered into the SIG model. Temperature calibration of the IR camera was done using a variable-temperature blackbody.

Digital image processing was used for local area histogramming on the resulting imagery. An error analysis was done to ensure the apparent temperature difference fell within accountable limits. The results showed that predicted apparent temperatures had an absolute average difference of 2.50°C which was less than the predicted temperature error of ± 2.85°.

This thesis provides a good foundation for additional work. There are considerable improvements required in the area of texture, expansion of the model to the 3 -5 µm window, and improved sensor models.

2.0 Background

2.1 Synthetic Image Generation

Creation of a credible computer image requires realistic 3-D modeling. Realistic appearance of objects in the scene is based largely on material properties and lighting conditions. Simulation of proper illumination conditions is a critical and complex task. Lighting conditions are measured in terms of radiant flux (watts). The direction of propagation of radiant flux from the source to the scene is usually called the *path*. The flux can be absorbed, transmitted, and reflected/scattered from objects along the path. To follow the propagation of radiant flux along a path, the technique of ray-tracing has been developed (this should not be confused with optical ray-tracing used in lens design).

Radiation can be absorbed, transmitted, or reflected/scattered within the atmosphere. We therefore need to include atmospheric propagation in a realistic scene simulation. Atmospheric models such as LOWTRAN 6 [Keneizys (1983)] and FASCOD2 [Clough (1986)] have been used to mimic the effects of the atmosphere.

The radiance of an object in the infrared is primarily due to its temperature and its material characteristics. The thermal environment of an object also has major impact on its IR radiance. The thermal response measured by remote sensors often permit assessment of the type and/or condition of features. The specific set of these responses are often called *thermal signatures* [Lillesand (1987)]. Sometimes the objects' environment may cause a change in the object's temperature and thus alter the thermal signature. For example, object temperature may rise due to exposure to sunlight, being near other hot objects, or from internal heat sources (*e.g.* an engine). Other additional environmental variables that effect object temperature include wind speed and cloud cover.

Previous work in IR SIG has created very detailed and complex computer simulation code. This study addresses a small portion of the process in an attempt to improve on the radiometry and atmospheric propagation effects. Diurnal cycles, background clutter, texture, heat flux, and terrain elevation data (*e.g.* from the Digital Elevation Model data (DEM) from the Defense Mapping Agency (DMA)) are not planned for inclusion at this time. However, the code may be modified to incorporate any of these factors in the future.

Advancements in computer graphics hardware have temporarily rendered many software algorithms obsolete. For example, rotation, translation, and scaling of objects can now be done by using hardware matrix multipliers. Z-buffering, a technique for hidden surface removal, can also be implemented in hardware. These advancements in hardware were taken advantage of in this attempt to create a realistic LWIR image.

The process of IR SIG incorporates many different disciplines such as geometrical representation, thermodynamics, and atmospherics. Many different parameters or variables are included in the various models. Most of the literature available at the academic level does not go into technical detail on how the process works; rather overall processes are described. However, a detailed description of all possible parameters is not the intention of this section. The goal is to introduce a few SIG techniques and present some basic information required to generate a synthetic image.

Most existing IR SIG requires similar types of data for input. To properly simulate an object in an environment, one must have the following: a geometrical representation of an object (usually computer graphics), some form of an atmospheric transmission model (such as obtained from LOWTRAN or an equation such as Beer's Law), and material characteristics (usually emissivity). The more elaborate simulators also include thermodynamic models and texturing ability.

2.2 GTVISIT: A Case Study

A good example of IR SIG is from Georgia Tech. This section will begin by using their work as a case study and then mention others who have done significantly different work.

The Electromagnetics Laboratory of Georgia Tech Research Institute , [Cathcart and Sheffer (1988)] and [Sheffer and Cathcart (1988)], has been involved in IR SIG for over thirteen years. Their program for generating synthetic images of scenes is called Georgia Tech Visible and Infrared Synthetic Imagery Testbed (GTVISIT) . It requires the data obtained from many other modeling programs such as MAX, GTSIG, IRMA, and MODELIR.

Scenes can be thought of as having two component types: A gridded terrain background, and a faceted object placed into that scene. A gridded data base means that the area has been divided into cells or grid blocks that may relate directly to Universal Transverse Mercator (UTM) coordinates. Backgrounds may be terrain/ocean, sky, or a combination of the two. Sky backgrounds can be simulated with 2-D textures or 3-D cloud models.

GTVISIT uses four gridded databases: feature (material type), elevation, radiance, and IR reflectance. The feature and elevation data may come from real-world sources (such as satellite imagery and DMA measurements), from synthetic data, or from a hybrid. The radiance and IR reflectance databases are generated from the temperature and/or reflectance of each material. Such assignments typically are obtained from measured data or thermal predictions.

GTVISIT creates images by using a Z-buffer algorithm. Z-buffering is a computer graphics technique used to display (or not display) hidden surfaces. First, elevation databases are projected upon a view screen, the range values for the overlapping pixels are compared (in the event of multiple projections on a pixel). Second, the objects are projected in the same way. Visibility tests using 'containing boxes' (Fig. 10) are used. If any part of the box is visible, the scene element is projected facet by facet. Then the atmospheric attenuation and emission along the viewing path are computed.

To save computation time GTVISIT pre-computes radiance values for each of 12 orientations of each object in the scene. Radiance values are also precomputed and assigned to each vertex of a facet. If a pixel lies inside the facet, the pixel radiance is computed by bilinear interpolation. The code can include backgrounds and diurnal variation, as well as dynamic processes such as fire, smoke, and dust clouds.

GTSIG computes the thermal radiance of objects from first-principles thermal prediction code for objects. It employs a 3-D thermal network analyzer and a multi-surface radiosity technique in the computations; radiosity is similar to ray-tracing. The physical processes included in the thermal model are solar and sky radiation (direct and occluded), mass-transfer processes (evaporation, condensation, sublimination, and precipitation), fluid flow effects, shadowing, and multi-surface reflections.

IRMA [Botkin (1981)] was developed for the Air Force by Grumman Aerospace Corporation. It is a semi-empirical, one-dimensional heat transfer approach to compute the radiance for each facet of an object based on thermal history of the object. An IRMA model is easier to construct then a GTSIG model; it is used when solar radiance and self-shadowing effects are important.

Computations for GTSIG are robust but they do not account for angular emissivities, which are critical in the infrared [Schott (1986)]. GTSIGs' pre-computation of radiance values is acceptable, but may not be appropriate when attempting to generate an exact radiometric image. Finally, GTSIG does not address target-background interactions; however, Georgia Tech has suggested this improvement.

2.3 Other Approaches to SIG

Evans and Sutherland Computer Corporation, [Biesel (1987)], has developed a real-time system to simulate Forward Looking Infrared (FLIR) imagery. By assuming thermal equilibrium of the object with its environment, and limiting temperature to 256 distinct values, real-time scene simulation may be achieved. Object warming may be due only to solar radiation or diffuse sky contributions. Most importantly, specular reflection is neglected and atmospheric attenuation is calculated using Beer's law approximations. The method being proposed will not have the real-time constraints.

Nichols Research Corporation, [Dunn (1987)], has simulated high spectral and spatial resolution of a downlooking IR sensor. Their technique addresses hardware limitations of large-scale images. Z-buffering techniques are used to superimpose the contribution of each object to a final IR synthetic image. The simulator was constrained by hardware requirements (only 8 Mbytes). The simulation generated in this thesis has no such memory constraints.

The Night Vision and Electro-Optics Laboratory (NVEOL), [Kornfeld (1987)], has developed an IR model that predicts the thermal radiation distributions of a scene. The thermal signatures are calculated via an empirical model based on measured data. A specialized atmospheric attenuation routine (LTR) was validated against LOWTRAN 6. Because they were tasked to develop code of less than 2 Mbytes, computer storage problems were also discussed. Their technique relied on the artistic talents of cartoonists to create individual frames. Overlays are used to simulate background objects (*e.g.* mountains). It is this artistic requirement that makes this approach unattractive. This study relies on the user's ability to construct objects on a CAD workstation.

Grumman Corporate Research Center, [Gardner (1987)], used mathematical textural functions along with simple surfaces to simulate features such as hills, trees, and clouds. The U.S. Army Tank Automotive Command (TACOM) has extended this technique to the IR by substituting statistical characteristics of measured IR data for the texture function. As successful as their texture mapping technique is, they are still developing targets to place into the scenes. This project illustrated the important role played by texture in IR images. However, the inclusion of textural information is not being considered in this study.

Aerodyne Research, Inc. (ARI), [Stets (1988)], have developed their own code (AERIE) for simulating aircraft with various backgrounds. AERIE is very specialized and has the ability to calculate the thermal plume given off by aircraft engines. They can also include the influence of clouds on a target. The radiometry is calculated using SPIRITS 2.0, (Spectral Infrared Imaging of Targets and Scenes). Once again, LOWTRAN 6 is used for atmospheric attenuation computation. At the time of this writing SPIRITS is being modified to include background objects in the radiance calculations.

Schott and Salvaggio (1986) developed a simulator that used 2-D images acquired by actual scale models and atmospheric transmission calculated by DIRTRAN. DIRTRAN was developed as an easier interface to LOWTRAN 6. However, their method is not very exact since angles are estimated and it uses a "cut and paste" method to extract textures and mask them onto their synthetic image Their report recommended 3-D scene simulation as a future improvement.

This literature review suggests potential improvements in IR scene simulation. Many similarities exist between the previous work and in the SIG presented here. For example, faceted data was chosen to represent objects. This model also makes use of the popular LOWTRAN 6 atmospheric propagation model.

The LWIR SIG method made use of recent advances in computer graphics to radiometrically model any object in any scene at any viewing location. The use of ray-tracing improves the computation of target radiance by properly including background radiance. It interacts with a program called DIRTRAN, which relies on data from the LOWTRAN 6 atmospheric propagation model. It uses angular emissivity data to characterize materials. It runs on relatively common computer equipment (VAX/VMS). It interfaces to programs that have previously been used in the one- and two-dimensional cases. It is hoped that this new SIG program will be the foundation for other academic research to be built upon.

3.0 Technical Approach

This section will discuss how radiant energy is modeled, beginning with the basic theory of atmospheric propagation. Equations are then presented which represent how radiant energy interacts with objects. Finally, ray-tracing techniques are presented that are used to implement these equations.

3.1.1 Radiation Transfer

All objects reflect, absorb, or transmit incident radiation (irradiation). The one that tends to dominant is determined by the physical characteristics of the material and its environment. Before radiation from a source reaches a sensor, it will be selectively absorbed by atmospheric gases and scattered away from the line of sight by small particles suspended in the atmosphere. Atmospheric transmittance depends on temperature, pressure, path length, concentration of absorbing gases, and the wavelength of radiation.

Observed electromagnetic radiation will be a combination of reflected energy from direct sunlight (specular irradiation), reflected energy from scattered sunlight (skylight or diffuse irradiation) and a self-emitted component. Figure 1 shows the possible sources of reflected radiation that could reach a sensor. Path A represents the exoatmospheric energy being propagated through an attenuating atmosphere, reflected from the Earth's surface, and propagated back through an attenuating atmosphere to the sensor. Some of the solar radiation is scattered in the atmosphere, reflected from the earth and propagated through an attenuating atmosphere to the sensor (path B). Finally, some of the solar radiation is scattered in the atmosphere to the sensor independent of specific interaction with the Earth (path C).

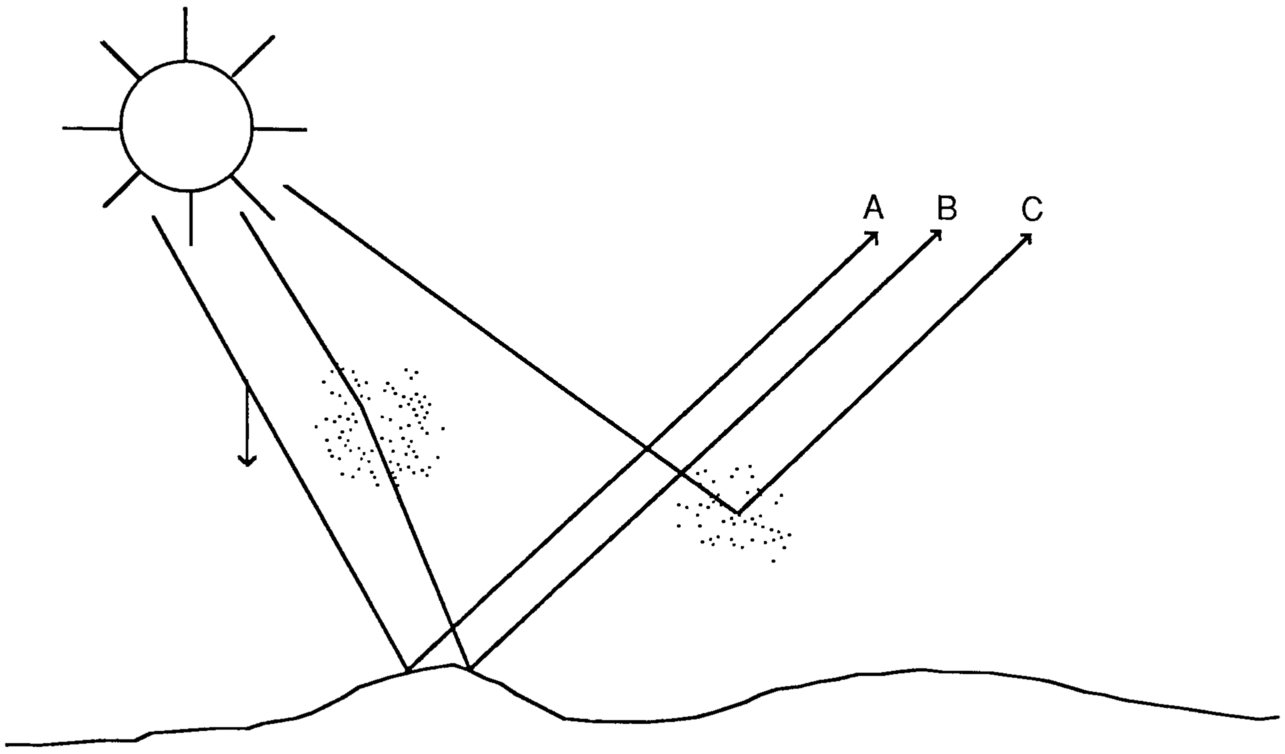


Figure 1- Reflected components of radiance reaching a sensor

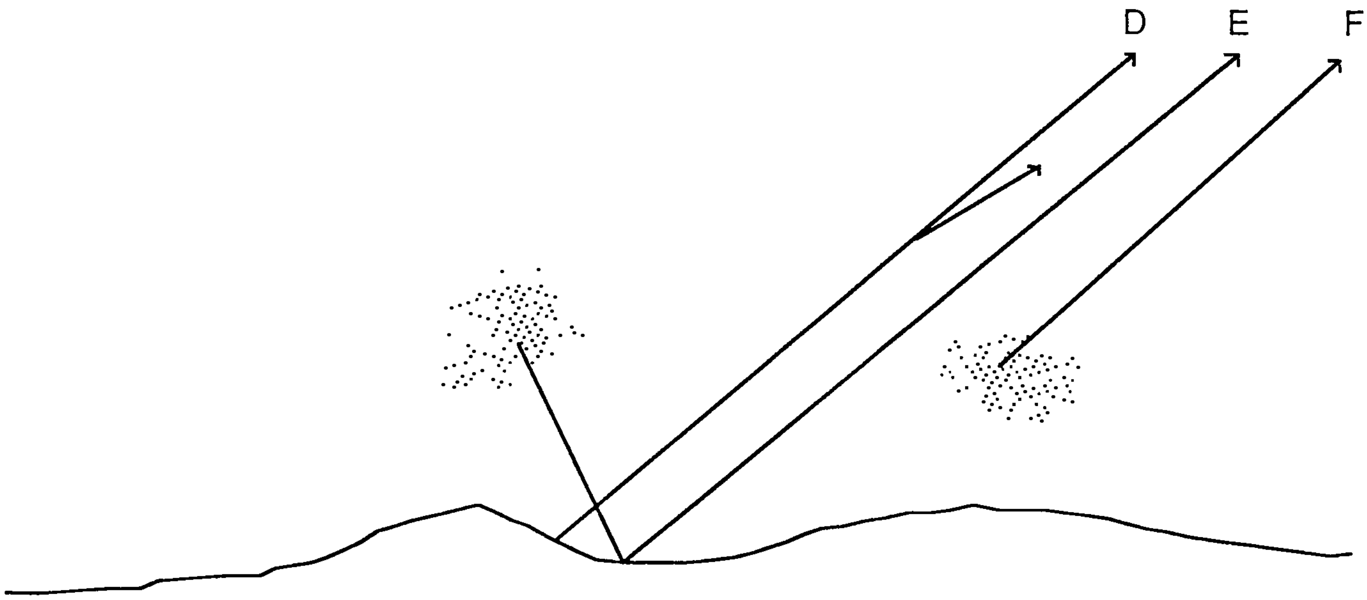


Figure 2 - Emitted components of the radiance reaching a sensor

Figure 2 shows the energy paths associated with self-emitted radiation. The Earth radiates a broad spectrum of energy due to both its temperature and its radiation efficiency. This energy is propagated through an attenuating atmosphere to the sensor (path D). The atmosphere itself radiates energy (due to its temperature) that is propagated to the Earth, reflected, and propagated back to the sensor (path E). Finally, the atmosphere between the sensor and the Earth radiates energy that can propagate to the sensor (path F).

The amount of energy reaching the sensor from each path shown in Figures 1 and 2 is a function of wavelength, and the relative importance of each path is also wavelength-dependent. When modeling these sources of energy, the following simplified equations are used. Assuming lambertian reflectors, the radiance reaching a sensor is:

$$L_r = \text{path A} + \text{path B} + \text{path C} = \frac{E_s r \tau}{\pi} + \frac{E_{sky} r \tau}{\pi} + L_{u_{vis}} \quad (1a)$$

where:

E_S = irradiance from the sun reaching the surface,

r = reflectance of target,

τ = atmospheric transmission,

E_{sky} = irradiance from the sky,

$L_{u_{vis}}$ = upwelled radiance reaching a sensor caused by scattering in the intervening air column in the reflective wavelength region.

The emitted radiance reaching a sensor is:

$$L_e = \text{path D} + \text{path E} + \text{path F} = \varepsilon L_T \tau + r L_d \tau + L_{u_{ir}} \quad (1b)$$

where:

ε = emissivity of target,

L_T = radiance associated with the target's kinetic temperature (T),

L_d = downwelled radiance from the hemisphere above the target.

$L_{u_{ir}}$ = upwelled radiance in the infrared

However, in the 8-14 µm (LWIR) window, solar effects are negligible (because of low solar irradiance) and E_s, E_{sky}, and $L_{u_{vis}}$ are all approximately equal to 0 so Equation 1a reduces to 0. Although some people assume it, solar effects are **not** negligible in the 3-5 µm window.

Equation 1b can be expanded to explicitly incorporate the dependence of the observed radiance on altitude (h) and view angle (θ). In this case:

$$L(h,\theta) = \tau(h,\theta)[\, \varepsilon(\phi)L_T + r(\phi)L_d\,] + L_u(h,\theta) \qquad (2)$$

where:

$\varepsilon(\phi)$ = angular emissivity in the band of interest,

$r(\phi)$ = angular reflectance in the band of interest,

$\tau(h,\theta)$ = atmospheric transmission to altitude h through view angle θ,

$L_u(h,\theta)$ = upwelled radiance reaching a sensor caused by self-emission and scattering in the intervening air column.

3.1.2 Definitions of Emissivity and Reflectance

It is important for the reader to understand the meaning of the emissivity and reflectance of a material. These values are difficult to measure because of the scope of dependent values involved in measurement instrumentation. The following brief paragraphs describes these parameters.

A blackbody is a hypothetical, ideal radiator that totally absorbs and reemits all energy incident upon it. This ideal condition is only an approximation for actual objects, which emit only a fraction of the energy of a blackbody at the same temperature. This fraction is called the emissivity, ε, and ranges between 0 and 1. Emissivity is a function of wavelength and view angle, as is reflectance.

Reflectance can be defined as the ratio of incident energy leaving a surface to the energy incident on the surface. The reflectance of a material is a very difficult property to characterize. At one extreme is specular reflection, where energy is directly reflected from the surface of the object (for example, a mirror). At the other extreme is diffuse reflector for which, the reflected radiance is equal in all directions (*c.f* Figure 3). The complete reflectance function of a real-world surface is called the bidirectional reflectance distribution function (BRDF) and is defined as a function of source - object - sensor angles. In this study it is assumed that materials are either perfect specular or diffuse reflectors.

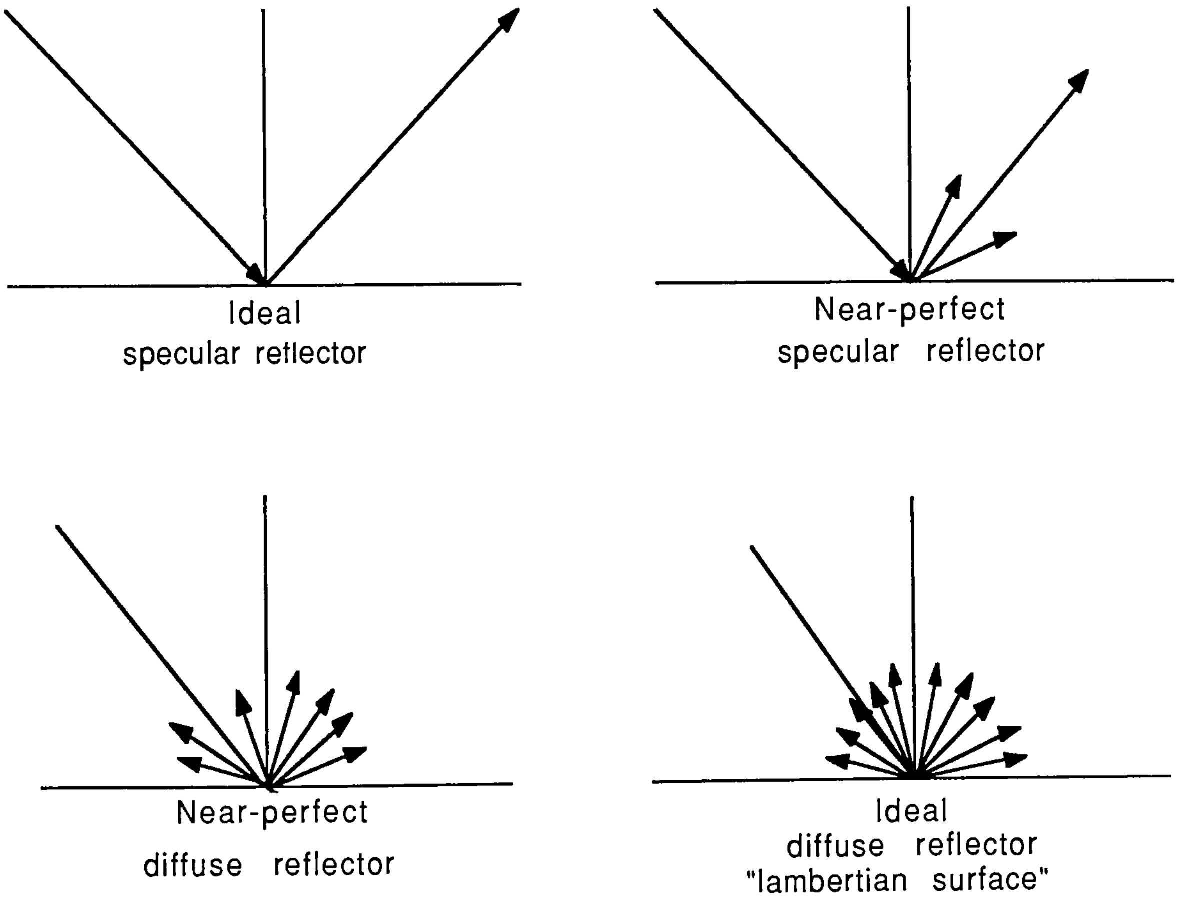


Figure 3 - Specular versus diffuse reflectance

The total downwelled radiance onto the target is computed from the contribution of each element of solid angle above the target. The irradiance on the target can then be expressed as:

$$E_d = \int L_d(\theta,\Phi) \cos(\theta)\, d\Omega \qquad (3)$$

$$\text{where } d\Omega = \sin(\theta)\, d\theta\, d\Phi$$

where the integral is over the solid angle Ω in the hemisphere above the target. It can be re-expressed in terms of azimuth angle as shown in Figure 4. In this example, θ and Φ are arbitrary angles demonstrating the hemispheric downwelled radiance computation.

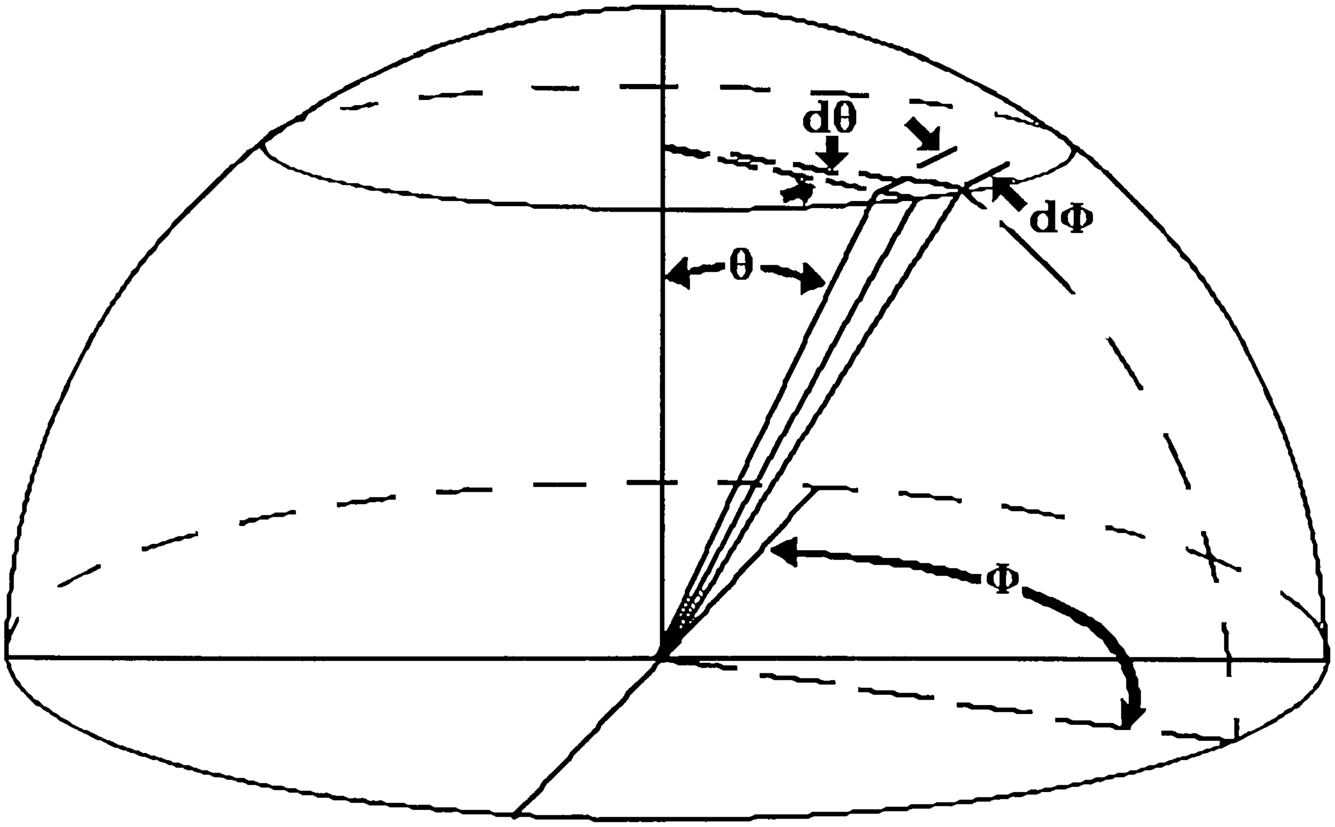


Figure 4 - Integration of downwelled radiance

$$E_d = \int_{\Phi=0}^{\Phi=2\pi} \int_{\theta=0}^{\theta=\frac{\pi}{2}} L_d(\theta,\Phi)\cos(\theta)\sin(\theta)d\theta d\Phi \qquad (4)$$

$$E_d = 2\pi \int_{\theta=0}^{\theta=\frac{\pi}{2}} L_d(\theta,\Phi)\cos(\theta)\sin(\theta)d\theta \qquad (5)$$

By assuming diffuse reflection from the surface, the total downwelled radiance reflected can be expressed as:

$$L_d = \frac{E_d}{\pi} \qquad (6)$$

Moreover, if the target is assumed to be a pure specular reflector, the reflected radiance can be expressed as $r(\theta)\ L_d(\theta)$ (*cf.* Figure 5.)

3.1.3 Downwelled and Background Radiances

In many cases the diffuse downwelled radiance incident on a reflecting surface is not from the sky alone. In the case of varying topography, the element being observed may also be irradiated by reflected or emitted radiance from adjacent terrain elements. In this case, the irradiance incident on the reflecting element can still be expressed as in Equation 3 if we observe that the downwelled radiance L_d will have components due to reflections and emissions from neighboring surfaces. In many cases, the downwelled radiance from the sky (L'_d) and non-sky background (L_b) are approximately constant with respect to θ and a simple shape factor approach can be used to estimate downwelled radiance. In this case, Equation 2 can be expressed as:

$$L(h,\theta) = \tau(h,\theta)[\varepsilon(\phi)L_T + (FL'_d + (1-F)L_b)r(\phi)] + L_u(h,\theta) \quad (6)$$

where:

L_b = blackbody radiance from a background surface at temperature b,

F = shape factor, *i.e.* the fraction of the hemisphere above the reflecting surface which is sky.

More rigorous derivations of Equations 1 through 6 can be found in Schott and Salavggio (1987), Turner and Keller (1975), and Slater (1980).

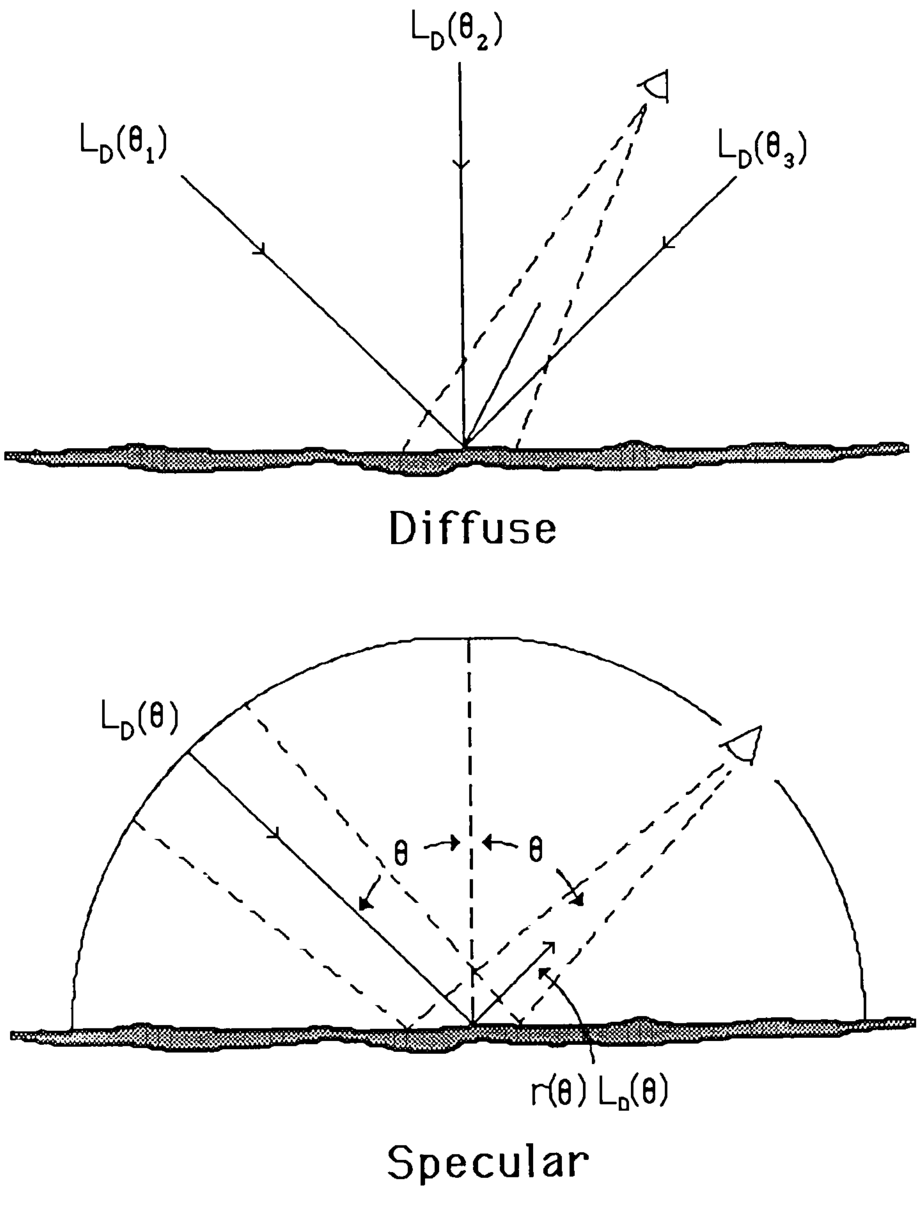


Figure 5 - Comparison of diffuse and specular reflectance

3.1.4 Approach to Solving the Radiance Equation

Before energy is propagated through the atmosphere to a sensor, we need to calculate the radiance leaving the target at the ground in the direction of the sensor, $L(0,\theta)$. To do this, we do not need to consider τ and L_u, they can be added later. If we ignore the shape factor (see Appendix B) Equation 6 becomes:

$$L(0,\theta) = \varepsilon(\phi)L_T + r(\phi)[L_d + L_b] \qquad (7)$$

It is necessary to obtain values for $L(0,\theta)$. $\varepsilon(\phi)$ is obtained through experimental data and stored in a look-up-table. Assuming opaque objects are being modelled, the transmission term would equal zero, then Kirchhoff's law states that the sum of a material's reflectance and emissivity is unity. The temperature, T, for a particular facet has been previously entered by the user. L_T, the radiance at temperature T, is pre-calculated (in the passband of interest) via Planck's equation . Ray-tracing will be used to calculate L_d and L_b because the downwelled radiance was expanded into two parts: that due to the sky and the background.

It was shown previously that L_d was calculated from the integration of hemispheric radiance. Calculating background radiance is one of the primary reasons the ray-tracer is being utilized in this approach. The question arises as to how many background objects are significant to the measured radiometry of a target. Appendix A shows that for completely specular objects, the measured radiance does not significantly change after one reflection. Thus, the calculation of L_b need only consider one bounce.

With this final assumption in place, the methodology for solving Equation 7 starts with a projection of a primary ray representing the flux path. If it hits a diffuse object, the integrated downwelled radiance L_d is calculated. Note that the shape factor is being ignored (see Appendix B). If the object is specular, a secondary ray (reflection) is projected. If that secondary ray hits an object, then $L_b = L_{T_b}$. However, if the secondary ray does not hit an object, the angular downwelled radiance $L_d(\theta)$ will be calculated (*cf.* Figure 6).

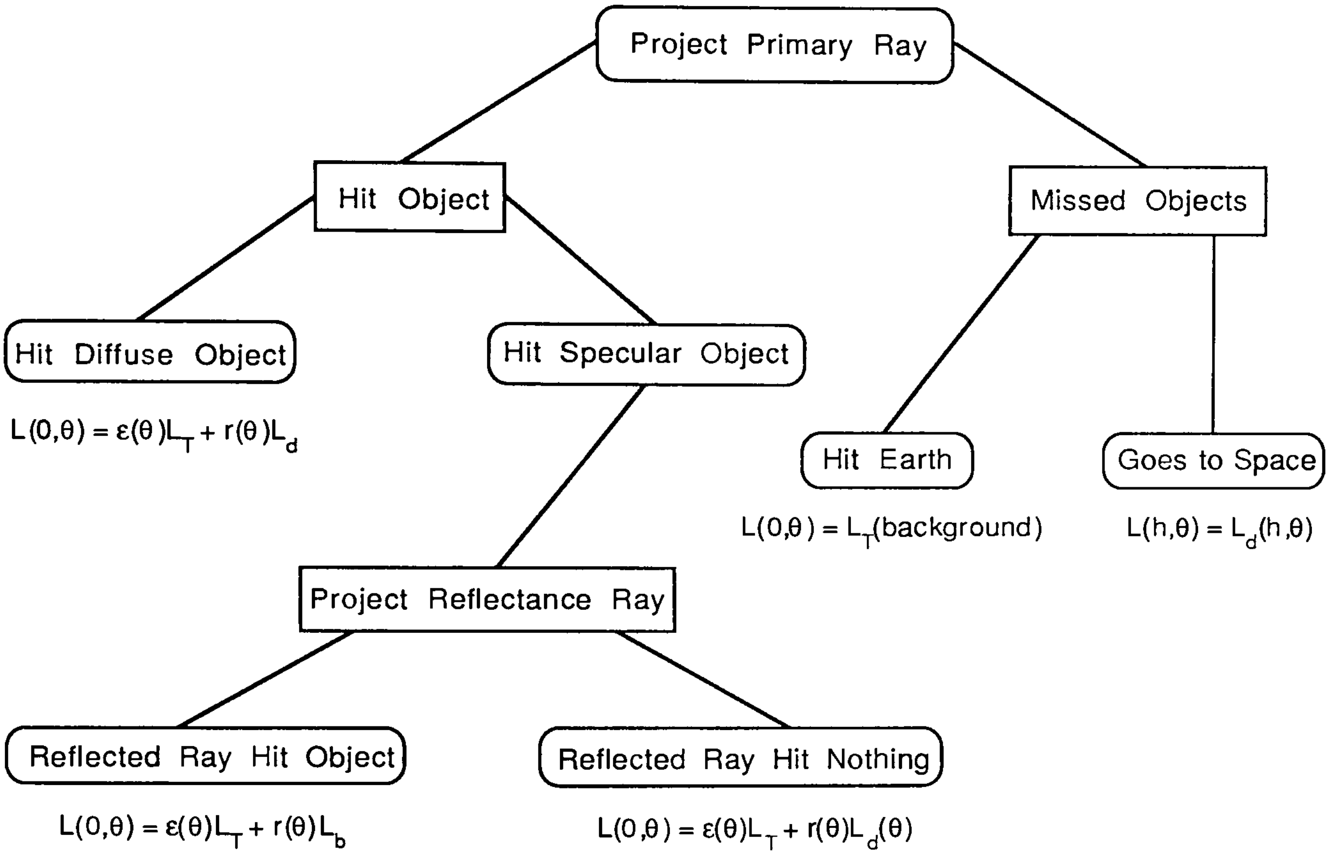

Project Primary Ray
Hit Object
Missed Objects
Hit Diffuse Object
$L(0,\theta) = \varepsilon(\theta)L_T + r(\theta)L_d$
Hit Specular Object
Hit Earth
$L(0,\theta) = L_T(\text{background})$
Goes to Space
$L(h,\theta) = L_d(h,\theta)$
Project Reflectance Ray
Reflected Ray Hit Object
$L(0,\theta) = \varepsilon(\theta)L_T + r(\theta)L_b$
Reflected Ray Hit Nothing
$L(0,\theta) = \varepsilon(\theta)L_T + r(\theta)L_d(\theta)$

Figure 6 - Ray projection flowchart

The test to find out if a ray goes to space compares the current ray angle against a "critical" angle. The critical angle is calculated at the beginning of the SIG process when the altitude (AGL) of the sensor is specified. Figure 7 illustrates this computation.

If the ray did **not** go to space then the surface-leaving radiance $L(0,\theta)$ will be multiplied by the transmission factor $\tau(h,\theta)$. Finally, the upwelled radiance $L_u(h,\theta)$ is added. These values have been calculated using the atmospheric model LOWTRAN via DIRTRAN. However, if the ray **did** go to space the sensor would see the path radiance from the current altitude to the top of the atmosphere. This will be calculated with a modification of the upwelled radiance routine in DIRTRAN . If the current angle cannot be calculated by DIRTRAN (angles > 85^{o}) then a second-order polynomial extrapolation is performed (see Appendix E).

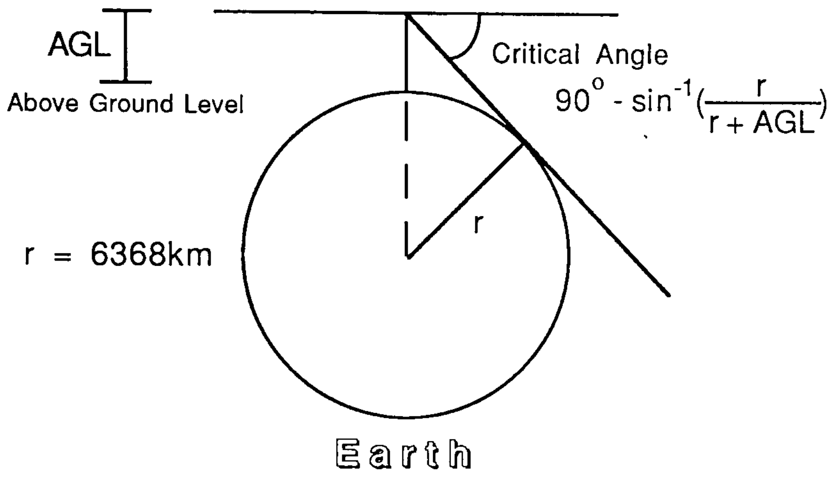


Figure 7 - Calculation of critical angle

This section has presented how radiometric quantities such as τ and L_u are used. The following section presents how the SIG obtains these values from the atmospheric propagation model. It also illustrates how this data is formatted for use by the SIG.

3.2 Atmospheric Propagation Models

Atmospheric propagation models are used to predict the effect of the atmosphere on the propagation of radiant energy. They are used in weather modeling, laser transmission studies, and (as in the present study) flux transmission. There are general types of these models: spectral line and spectral band. The line model operates at the quantum-mechanical level and it evaluates the influence of each type of atom in the atmospheric path. A band model uses integrated estimates of the line model.

3.2.1 LOWTRAN

LOWTRAN [Kneizys (1983)] is a band model that uses empirical data to estimate the atmospheric transmission through each of N homogeneous layers of the atmosphere. The effect of various atmospheric constituents is included in the calculation of effective transmission terms which are calculated for each layer through empirical relationships between transmission, spectral absorption cross sections, and number densities. The number densities within each layer of the atmosphere are estimated from empirical data. For detailed quantitative work, absorption calculations can be based on radiosonde data. Radiosonde data is obtained using a balloon containing transducers and a transmitter which sends down the temperature and pressure as a function of the balloon's altitude.

The upwelled radiance at any angle θ and altitude h is computed from:

$$L_U(h,\theta) = \sum_{i=1}^{N} L_T(i)[1 - \tau_i(h_i, \theta)]\tau_j(h_i,\theta) \qquad (8)$$

where:

N = number of homogeneous layers between the source and the sensor (typically 20), i = the atmospheric layer being considered,

$L_T(i)$ = radiance associated with the mean temperature of layer i,

$\tau_i(h_i,\theta)$ = transmission along the slant path through the ith layer to the sensor,

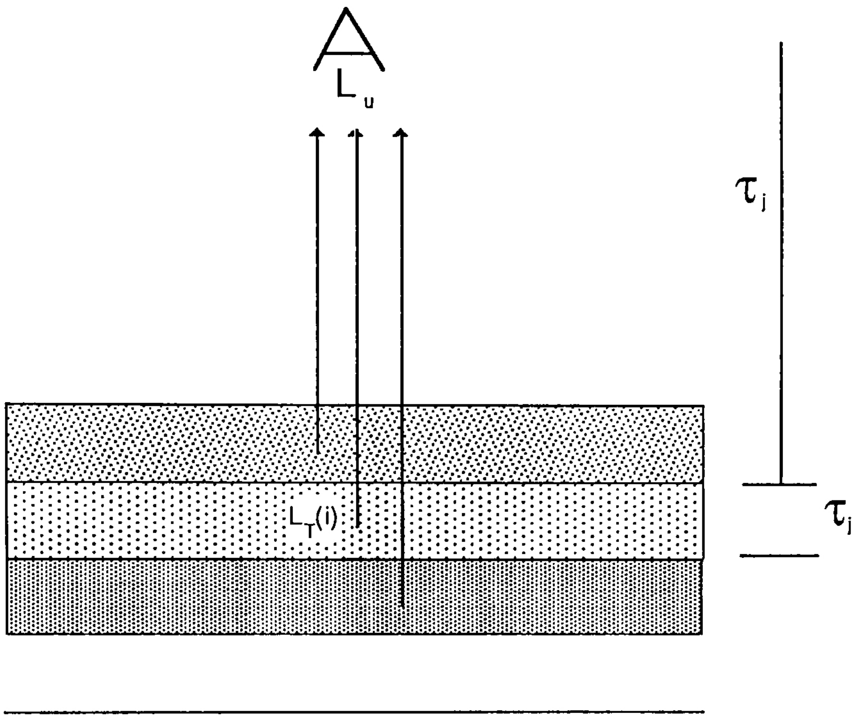
$\tau_j(h_i,\theta)$ is the transmission along the slant path from the top of the ith layer to the sensor which is calculated by multiplying transmissions of layers located above layer i (Equation 9).

$$\tau_j = \prod_{x=i+1}^{N} \tau_x \qquad (9)$$

The term $[1-\tau_i(h,\theta)]$ can be thought of as the effective absorbtivity and therefore the effective emissivity of the ith layer when it is treated as a blackbody. This is equivalent to assuming that the radiance scattered out of the layer is exactly replaced in magnitude and direction by the radiance scattered into the layer. Ben-Shalom et al. (1980) have suggested that this method is an improvement over the LOWTRAN 6 method for computing path radiance. Figure 8 illustrates the summation of contributions from each layer in the atmosphere to obtain the cumulative path radiance.

The downwelled radiance at any angle θ can be computed from LOWTRAN by defining a path from ground to space; this term is used for specular reflectors. If the target of interest is a lambertian reflector, LOWTRAN can be run over a series of angles to compute the downwelled radiance and numerically integrated over the hemisphere according to Equation 5.

The total transmission from the sensor at altitude h and view angle θ is a slight modification of Equation 9. The limits of the running product would go from the targets' layer to the sensors' layer.



$$L_u = \sum_{i=1}^{N} L_T(i)\,[1-\tau_i]\,\tau_j$$

Figure 8 - Summation of the contributions from each layer in the atmosphere to obtain the cumulative path radiance reaching a sensor.

3.2.2 Computational Considerations and Formats

The values required to solve Equation 8 were calculated using LOWTRAN 6. The necessary files are listed in Figure 9. The angular emissivity $\varepsilon(\phi)$ was linearly interpolated from an emissivity file [.ems]. This file contains the angular emissivity values from 0° to 90° in arbitrary increments (typically 10°). These data are assumed to be rotationally symmetric.

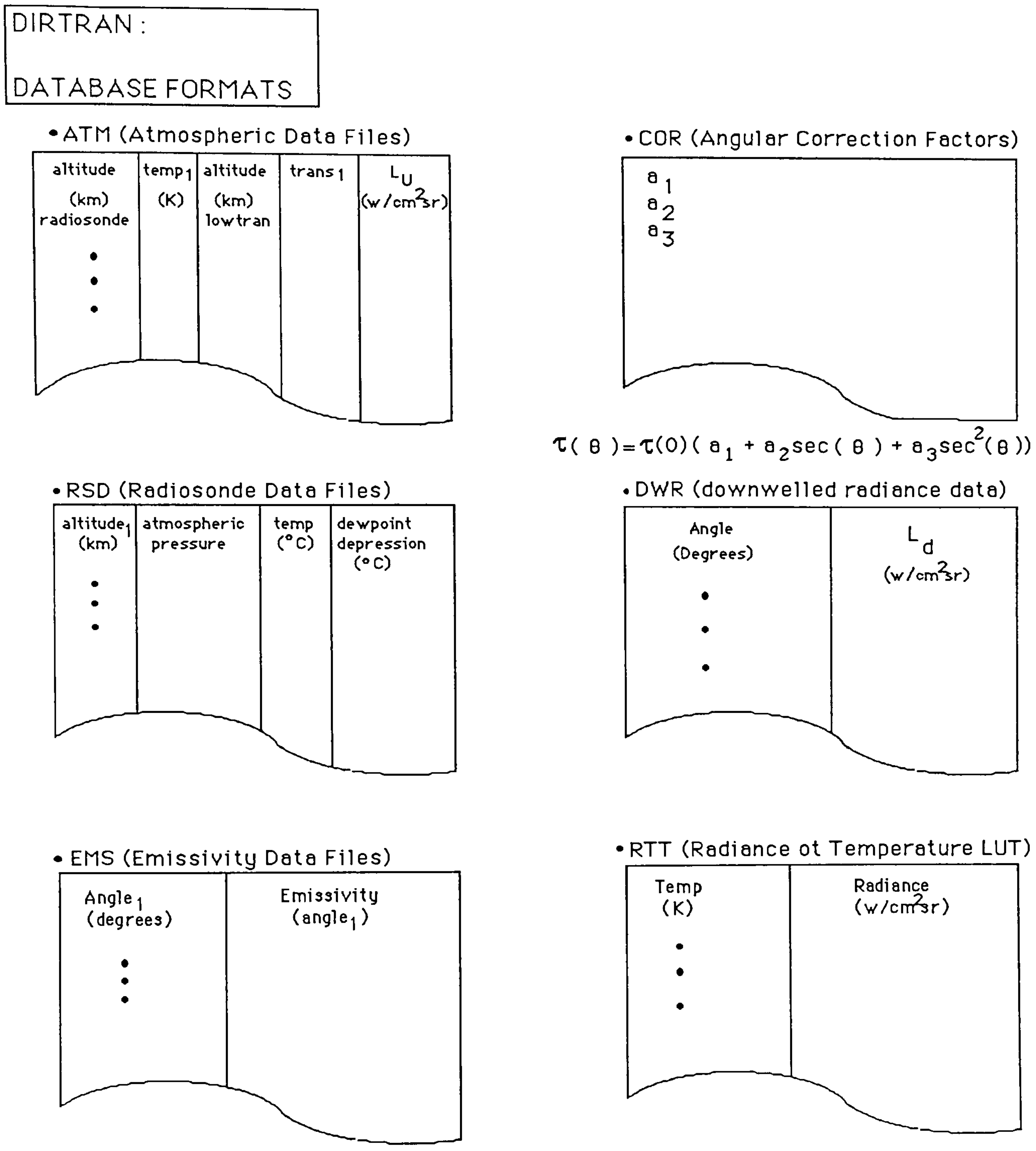


Figure 9 - Data files produced by DIRTRAN

To save computation time, a pre-calculated temperature-to-radiance file [.rtt] was used, that contains temperatures (Kelvin) and the equivalent radiance (W/m^2 sr) in the bandpass of interest obtained via Planck's equation. The required radiance was piecewise linearly interpolated from these data.

The downwelled radiance and the transmission factors are given in an atmosphere file [.atm]. This file specifies 5 parameters at many different altitudes: h_{sonde}, T_h, h, τ_h, and L_u. As a reminder, h_{sonde} is the altitude above sea level, whereas h is relative to the current position. The transmission at an angle θ is easily calculated by using a correction file [.cor]. This file contains three values (a_{1-3}) which are least-squares-regression coefficients used in the trigonometric polynomial:

$$\tau(\theta) = \tau(0)\ [a_1 + a_2\sec\theta + a_3\sec^2\theta] \qquad (10)$$

The upwelled radiance is calculated with a simple sec(θ) modification (*cf.* Figure 10):

$$L_u(h,\theta) = L_u(h,0)\sec\theta \qquad (11)$$

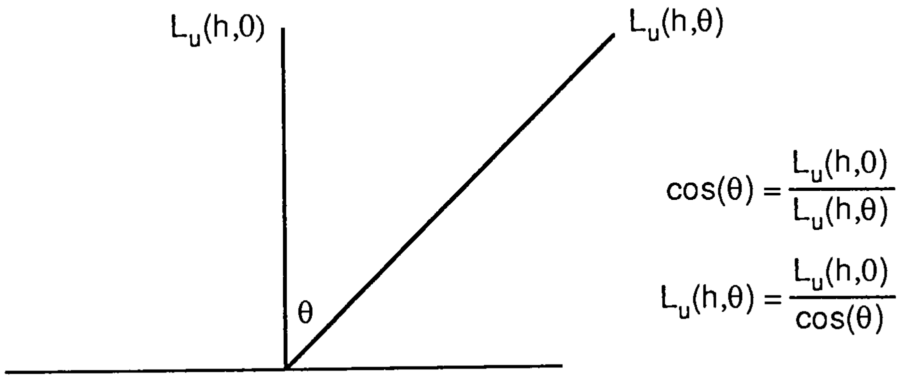

$L_u(h,0)$
$L_u(h,\theta)$
$\cos(\theta) = \frac{L_u(h,0)}{L_u(h,\theta)}$
$L_u(h,\theta) = \frac{L_u(h,0)}{\cos(\theta)}$
θ

Figure 10 - Calculation of angular upwelled radiance values

The downwelled radiance can be considered in two parts: the integrated downwelled radiance that is over the entire hemisphere and the angular downwelled radiance used for a specular reflector. The integrated downwelled radiance is calculated using LOWTRAN and Equation 5. The angular radiances are also computed via LOWTRAN and placed in a file [.dwr]. This file contains angular downwelled radiance at 5° increments. A piecewise linear interpolation is used on these values to obtain intermediate values.

At this point the reader has seen all the variables used in calculating radiance reaching a sensor in the LWIR and how they are determined. The next section presents the techniques used in the creation of the synthetic image.

3.3 Image Generation

This section discusses the process of creating a synthetic image. The goal of SIG is to create a realistic image. Ray-tracing is a way to create the appearance of such phenomena as reflection, shadowing, and translucence.

3.3.1 Theory of Ray-tracing

Radiant energy propagation may be modeled by tracing rays from sources to the detector. Even on the fastest machines, a complete ray-tracing calculation would take weeks to perform. However, only rays that eventually reach the detector are of interest. This *a priori* knowledge allows rays to be traced backwards from the detector, to the sources. Ray-tracing, therefore, can be appropriately called backward ray-tracing. A complete explanation of ray-tracing and image synthesis techniques can be found in Glassner (1987), Magnenat-Thalmann (1987), Blinn (1977), and Cook *et al.* (1987).

Each of the rays projected has to be tested for intersection with every object in the scene. Obviously, a single ray will then take longer to trace if there are more objects in the scene. However, short-cut techniques are available and will be discussed. Rays that hit objects can be reflected and/or refracted (transmitted). Therefore, from a single ray, two new rays could be generated for each single ray, and then must be traced separately into the scene. Multiple energy sources further complicate the illumination calculations. Therefore, the complexity involved in creating a properly illuminated SIG should be apparent.

In computer graphics, most complex objects can be composed of simple parts called primitive or core objects that are based on simple lines and arcs. Primitive objects such as cubes, spheres, and triangles are arranged into more complex objects (*e.g.* a house). In order to ray-trace these objects, several different kinds of testing routines are used. For example, ray-sphere and ray-plane intersections. Complex objects such as polyhedrons also need separate testing.

Over the past decade, graphic computers have progressed to the point where thousands of polygons per second can be displayed. Because of these developments and the simplicity involved in testing only one kind of intersection, it has been decided to create only faceted objects. All objects will be created with triangular approximations. Since facets are planer surfaces, their normals can be calculated with relative ease. These normals are used to calculate radiometric properties of the surface.

3.3.2 Radiometric Assumptions Affecting the Ray-tracer

All objects generated in this work will be opaque, *i.e.* not translucent. Therefore, there will be no secondary transmission rays. Surfaces will only be characterized as specular or diffuse. Due to the complexity involved in ray-tracing a diffuse material (*i.e.* shape factor), only specular materials will be ray-traced. In addition, because of radiometric attenuation, only one bounce need be accounted for (see Appendix A).

However, even with all these simplifications, additional factors in IR SIG complicate the task. The radiant exitance in the longwave infrared is so low that most radiance is from self-emittance. Therefore, every object acts as both object and source. Conventional ray-tracers may have many light sources.

3.3.3 Technical Approach to IR Ray-tracer

The main purpose of a ray-tracer is to determine the intersection of a ray with objects in the world. The ray is represented as a vector with a known initial point, direction and an unknown length. The intersection of the ray with all facets in the scene must be computed. To improve computation time, bounding volumes are used which range from simple spheres to complex parallelepipeds composed of planes (*cf.* Figure 11). These volumes are tested for ray intersection before individual facets are tested. A bounding cube will be used in this method because most of the rays will originate overhead and intersection with the top surface of a cube are easy to locate.

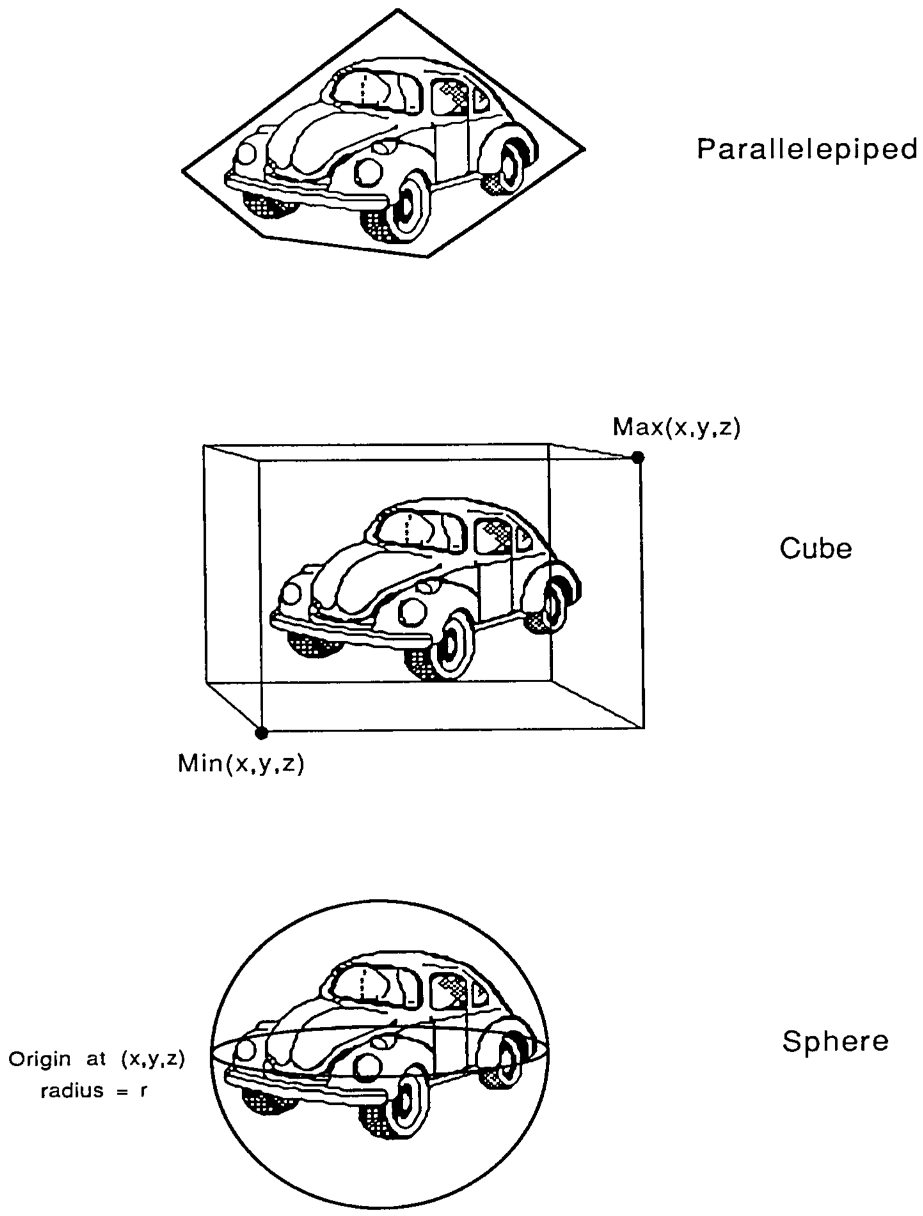


Figure 11 Illustration of bounding volumes for objects

Figure 12 shows some of the parameters the ray-tracer will use. The field of view of a sensing platform will be limited in both horizontal and vertical directions. The image plane consisted of N x M pixels. If N is equal to M and the FOV's are equal then a 1:1 aspect ratio is achieved. Different aspect ratios may be achieved with slight modification of the program. The rays generated all originated from the eye (viewpoint) and point to the center of each pixel.

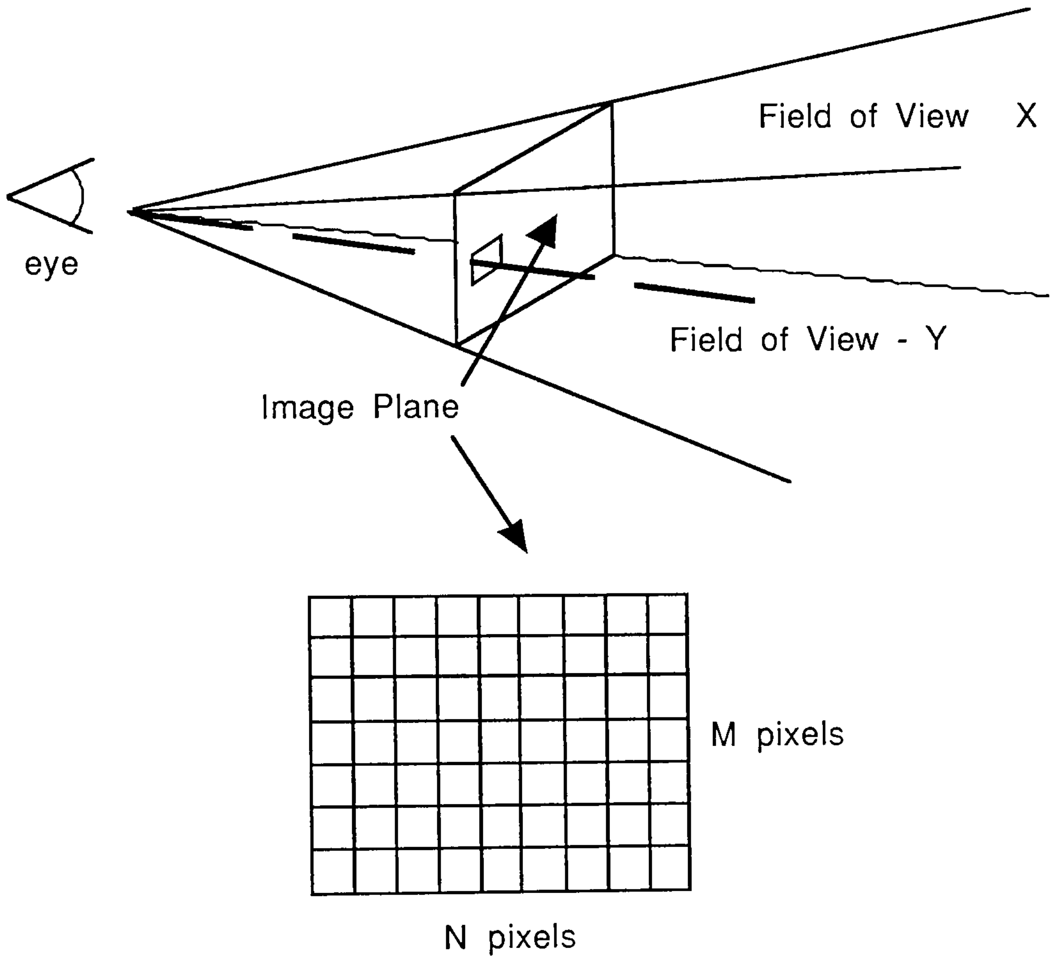


Figure 12 - Viewing pyramid and image plane

The test for whether a ray hit the bounding cube can be thought of as a true or false condition. If the bound volume condition is true, the next step is to test each plane of every facet in the bounded object. The test returns the intersection point in the plane being tested.

Appendices C and D describe the cube intersection test and the plane intersection test in detail.

If the plane test is true then the final polygon test is performed. The facet is projected into a 2-D plane, from which a hypothetical ray is sent in any direction. If the number of intersections is odd, then the point was inside the facet (*i.e.* the primary ray hit the current facet). If the number is even, then the plane's intersection point was not inside the polygon. Figure 13 shows a picture of this test.

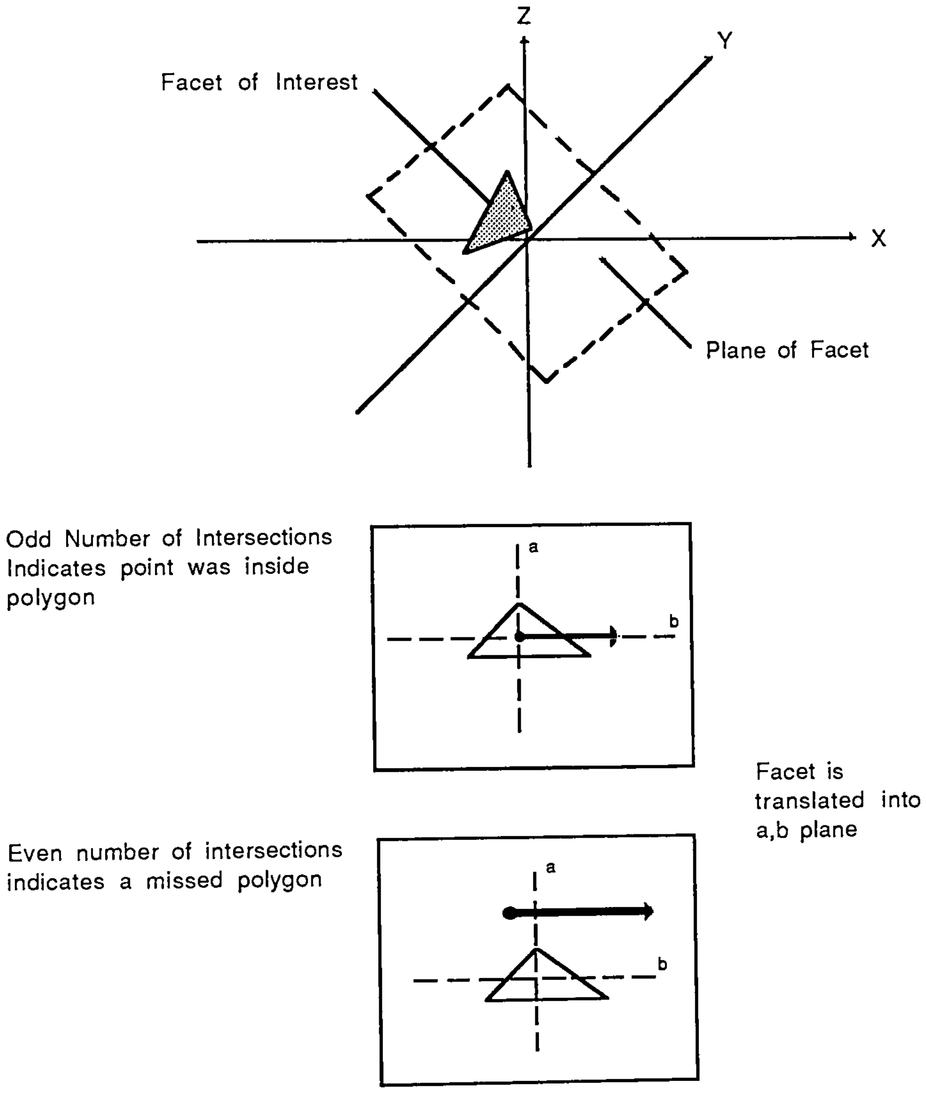

Z
Y
X
Facet of Interest
Plane of Facet
Odd Number of Intersections Indicates point was inside polygon
a
b
Facet is translated into a,b plane
Even number of intersections indicates a missed polygon
a
b

Figure 13 - Facet intersection test

It was previously assumed that the facet can be only specular or diffuse, (in the ideal computation complete BRDF data are needed). If the facet material is diffuse, its radiance is calculated. If it is specular, a second ray projection test is performed to check for background objects intersecting with the reflected ray.

3.3.4 Final Considerations

Ray-tracing samples image points and therefore can suffer from effects in space and time (spatial and temporal aliasing). However, there are countermeasures (anti-aliasing) [Crow (1977) and Dippe (1985)] that can be taken to prevent unwanted "jaggies" appearing in the image which include: supersampling, adaptive supersampling, distributed ray-tracing, and statistical supersampling. It is unknown at this time to what degree the imaging process will suffer due to aliasing, but it is expected to be minimal because the created images will have a much higher resolution than needed by a sensor.

The ray tracer yields two images, one image of the radiometric radiance that reaches the image plane and a interaction map that contains what interactions happened at each pixel. In the future, the class map can be used to add pre-computed texture information to the radiance image; but for now it just assists the user in analyzing the synthetic imagery.

Once the proper radiance field has been produced at the front end of the sensing system, further degradation due to the sensor can be introduced in order to generate a more realistic representation of the final image. These degradations are introduced and discussed in the following section.

3.4 Sensor Response and Modulation Transfer Functions

A sensor converts radiant energy in the focal plane into electrical power. Image degradations are caused by noise and the spread functions due to the optics and the finite detector size. Noise can be due to statistical fluctuations in measured flux, electronic amplification, and nonuniform detector characteristics [Kornfeld (1986)].

3.4.1 Sensor Response Function

The sensor response function (SRF) describes how a sensor reacts to radiant energy at a given wavelength. In this SIG process, only LOWTRAN 6 contains individual wavelength information. DIRTRAN uses the pre-computed integrated estimates of transmission and path radiance. Therefore, the sensor's effect can only be properly incorporated at the beginning of the SIG process rather then at the logical end. As LOWTRAN 6 computes the radiance for each wavelength, it is multiplied by a SRF value between 0 and 1. This results in an integrated effective radiance which is supplied to DIRTRAN.

3.4.2 Modulation Transfer Function

The modulation transfer function (MTF) is used to characterize the resolution of imaging systems. The MTF of some previously mentioned system components can be cascaded by performing a frequency-by-frequency multiplication. For example,

$$MTF_{overall}(f) = MTF_{optics}(f) \bullet MTF_{detector}(f) \bullet MTF_{electronics}(f) \bullet \ldots \quad (12)$$

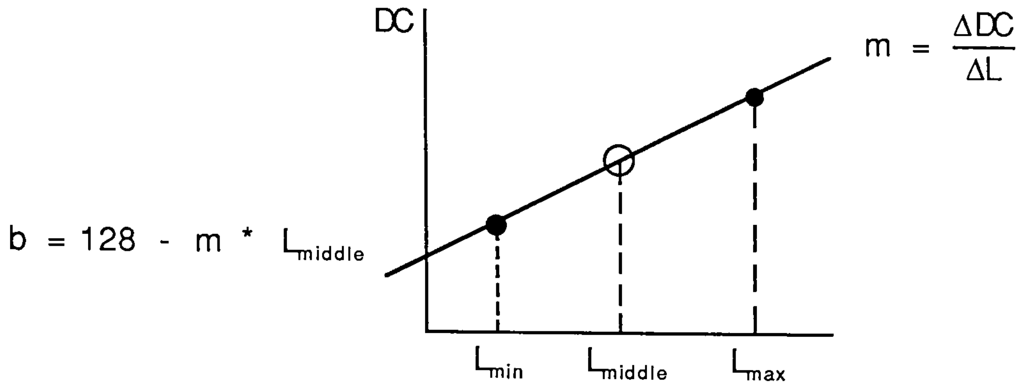
The task presented here has concentrated more on the entire system and not on individual components. There are many ways of simulating an MTF. One such approach can be found in Schott & Salvaggio (1987). Their technique forms a two-dimensional convolution kernel by convolving one-dimensional kernels specified for the across- and along-track directions. The one-dimensional kernels are originally defined as a 1-D power spectrum with a specified half-power frequency. However, in this thesis, final images were simply blurred by convolving with a 2-D square wave function representing the instantaneous field of view of a sensing element. This results in the final synthetic image that simulates an actual image generated from a system with equivalent MTF.

3.4.3 Conversion from Radiance to Temperature

At this point the radiometric image represents radiance values and must be converted to digital counts representing temperature via a calibration for the sensor. The gain of the sensor can be represented as an approximate digital count per degree (DC/T) value. The word approximate is used because there is a non-linear relationship between temperature and radiance. The DC/T value has to be converted to represent digital count per radiance unit which can be applied to convert the radiance image to temperature at each pixel image. This process is shown in Figure 14.

This allows the user to enter a gain with respect to degrees Kelvin and have the conversion carried out in radiance units. This technique causes the bias (y-intercept) to shift when the L_{min} and Lmax differ from scene to scene. However, it does have the advantage of pseduo autocontrast.

This concludes the theory on the SIG process. The following sections discuss how it was tested and some of the output it has provided.



select midrange radiance $L_{middle} = \frac{L_{max} - L_{min}}{2} + L_{min}$

convert to temperature via Planck's equation $L_{middle} \Rightarrow T_{middle}$

increment by 1K $T_{middle+1} = T_{middle} + 1$

convert to radiance via Planck's equation $T_{middle+1} \Rightarrow L_{middle+1}$

find ΔL $\Delta L = L_{middle+1} - L_{middle}$

Figure 14 - Conversion from radiance to approximate temperature with DC/degree

3.5 Experimental Approach

In order to verify such a robust simulator, it was decided to isolate and analyze a few of the parts that contribute to the model. The first part focuses on the geometric capabilities and the second part concentrates on the radiometric accuracy.

The geometrical examples consist of a complex scene depicting a vehicle on a road with a background composed of trees and grass. Many images were generated to demonstrate the robustness of the simulator. These include variation of the sensor location, gain, and FOV parameters.

Due to the complexity involved with physically creating a scene like the one described above, a simpler scene was created in order to test the radiometry. This simpler scene shows IR phenomena that could happen in a complex scene. The experimental scene consisted of a wooden pyramid structure, a pool of water, and a warm background. The background object was a tank of hot water that, in turn, heated a sheet of steel that was painted black. All of these objects were placed on a concrete surface.

In order to get actual temperatures, YSI high-precision thermistors (devises that change their electrical resistance as a function of temperature) were placed on the various objects around the scene. These thermistors are pre-calibrated so there is no need for calibration in this step. A random sample of these thermistors were tested in a water bath to be sure they matched the provided calibration data. The real IR image was acquired with an Inframetrics 600L camera. Although the camera contains a built-in temperature reading, it still had to be calibrated in order to achieve the desired accuracy. The calibration made use of a CI Systems SR 80 Extended Area Infrared Radiation Source (*i.e.* a blackbody) settable from 0°C to 50°C (depending on the dew point). The IR camera imaged the blackbody over a few different temperatures, and a transfer function was calculated. This function is used to function is used to convert digital counts to temperatures. Once the IR image has been acquired, local areas were averaged and converted to temperatures via the transfer function. These temperature values were compared with the values predicted by the SIG.

In order to create the synthetic IR image of the experimental scene, the atmospheric conditions were entered into the model. This was done by using (previously mentioned) radiosonde data on the evening the experiment was performed. In addition, the temperatures of the objects, obtained from the thermistor data, were also entered. The viewing parameters were easily calculated from the measured scene geometry. The FOV of the IR camera was given along with the image size. The gain of the sensor was estimated from knowing what the temperature range of the scene would be. The final radiometric image was then convolved with a 5 x 5 equal weight smoothing kemel to represent degradation due to sensor and optical effects.

4.0 Results

4.1 Description of the Computer Resources and Model Inputs

The SIG process began with the creation of objects on an Intergraph (modified DEC VAX 11/785) running Intergraph-RandMicas (IRM) finite element modeler/analyser software. An object was constructed by connecting lines, planes, and surfaces together. Once a three-dimensional object was satisfactory, the object was processed with a finite element analysis program. This program also tagged the facets with a material name. An Intergraph data file contained the vertices and polygon connections that recreate the object in cartesian coordinate space.

The Intergraph file was then parsed with a program on the VAX into the exact coordinates and material indices. A material index file is required in order to read the name of a material (asphalt, dirt, water, etc...) and substitute it with an integer value. The material index was used in displaying the scene and assigning emissivity values. A sample of the "object" file is shown below:

```
4                                Number of facets

6                                Material index
0.000000 0.000000 0.000000       Coordinates
-4.096910 -3.133200 0.000000
-5.000670 -1.810670 0.000000

6
-4.100130 -1.886120 0.000000
-5.000670 -1.810670 0.000000
-4.096910 -3.133200 0.000000

6
-4.096910 -3.133200 0.000000
```

```
-3.181800 -3.181830 0.000000
-3.199730 -1.955010 0.000000

6
-3.199730 -1.955010 0.000000
-4.100130 -1.886120 0.000000
-4.096910 -3.133200 0.000000
```

The object files were then transferred to a Silicon Graphics Iris 3020 workstation. This is where the "scenes" were created. A scene file was created by writing the name of the object with three rotation, translation, and scaling values (9 values all together). There can be as many objects in the scene as desired. An example is shown on the following page.

```
object name
ξ, ψ, υ                 rotation about x, y, & z axes in degrees
tx, ty, tz              translation from x, y, & z axes
sx, sy, sz              scaling of the data in the x, y, & z directions
```

The scene creation utility shows the user four windows. The first window is where the viewer wants to be (x,y,z) and with the desired FOV. The other three windows are views looking down the three major axes. The user selects an object to be modified. The selected object can be rotated, translated, and scaled. These parameters are displayed in an additional window. When the user is finished modifying the object parameters, he/she can save these parameters to a new scene file.

The final step in the data preparation was converting the object's data with its rotation, translation, and scaling parameters to "raw" data. That is the cartesian point after it has been rotated, translated, and scaled. In addition, the user was queried for an entire object temperature that were assigned at this point. This final file was moved to a DEC VAX computer where the IR ray-tracer runs.

The ray-tracer needed many files in addition to the scene file. An execute file is used to tell the ray-tracer which viewing file to use, which scene file to use, and the names of the two output images. The reason for this file format is so "fly-by" scenarios may be created with relative ease. The material index file is again required along with the angular emissivity files. DIRTRAN will have been previously run to generate the six data files needed. Any inputs to DIRTRAN/LOWTRAN will be added at this stage (*e.g.* radiosonde data, sensor response function).

The "viewing" file has the following format:

Vx, Vy, Vz	Location of viewer [km]
M, N	Number of pixels in X & Y
α, β, γ	Rotation of image plane about axes [degrees]
fovx, fovy	Fields of view in X & Y [degrees]
DC/T	System Gain

4.2 Description of the Output Imagery and Associated Data Files

The first image to be created was the interaction map. This map was created as the ray-tracer calculates each pixel. For each pixel, the following happens: The material index of the first object a ray hits is recorded and shifted four bits to the left (*i.e.* the upper nibble). If the material was specular, the reflection ray is cast out. The material index of the reflection ray's object is placed in the lower four bits of the byte for the current pixel. If no objects were hit, then a zero DC is placed in that pixel. This technique allows for 15 different materials (4 bits) giving 256 (8 bits) unique interactions.

The second image created was the synthetic image. This is what the scene would look like through the given atmosphere, with the defined viewing parameters. A Look-Up-Table (LUT) file is also created that gives the radiance, and temperature for each digital count in the image.

4.3 Preliminary Testing

Before believing any of the output imagery, basic testing was done on easy objects. Small planes (2 facets/plane) were placed in a checker-board lay-out. Each of the angles from the chosen viewpoint were easily computed by hand. Different temperatures and emissivites were assigned to each plane. As the ray-tracer ran, transmission, angular emissivity, and radiance values were validated against DIRTRAN. This insured the fact that the SIG was reproducing DIRTRAN results.

4.4 Example Imagery

Figures 16 through 19 demonstrate a few of the capabilities of the image generator. The scene depicts a vehicle on a road with a background composed of trees and grass. The input specifications ,and location of, the objects are given in Table 1 and Figure 15 respectively.

TABLE 1
Scene Input Specifications

Object	Material	# of Facets	Temperature*
Vehicle	Steel	500	290.30 K min/338.70 K max
	Glass	**	292.77 K
	Tires	**	300.00 K
Road	Asphalt	4	293.15 K
Trees	Conifer	72	289.15 K min/292.30 K max
Field	Grass	4	286.18 K

* Object temperatures based on data from the TEEX/TRIO thermodynamic model with min and max based on statistical distribution of temperatures for that object.

** Included in 500 facets

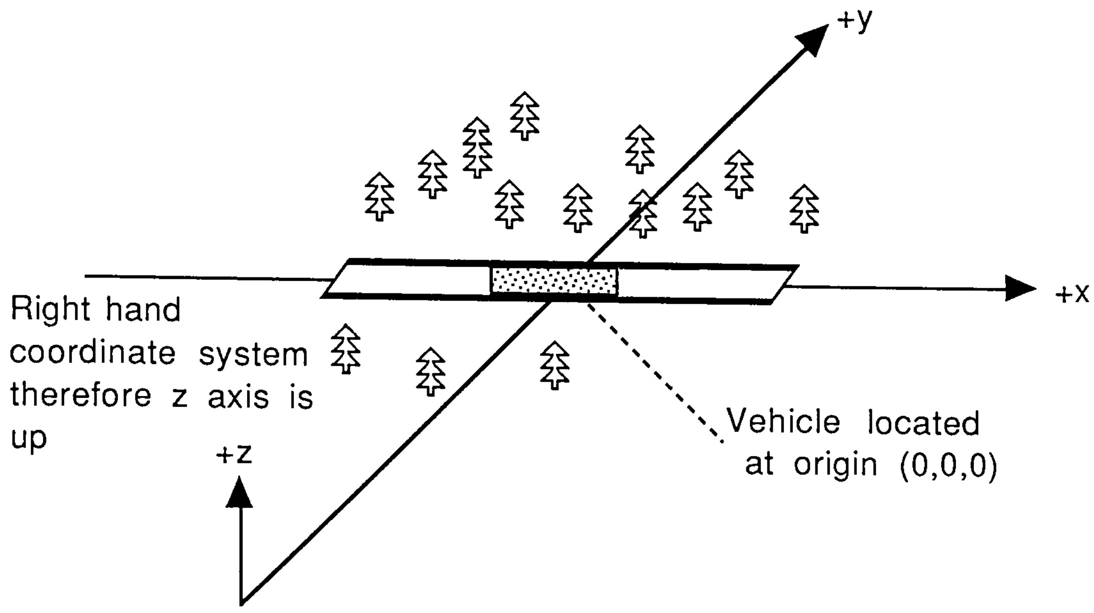

+y
+x
Right hand coordinate system therefore z axis is up
+z
Vehicle located at origin (0,0,0)

Figure 15 - Diagram of the relationship of objects used as input to the image generation algorithm

Table 2 shows the viewing, field of view, and sensor gain parameters for each frame in Figures 16-20. These images were all 256 x 256 pixels. Therefore, each horizontal (X) IFOV shown is calculated using 256 pixels. Frame 16 is a horizontal "fly-by" image sequence. Frame 17 is a perpendicular "fly-by" sequence, Frame 18 demonstrates the gain parameter, and Frame 19 changes the field of views.

Each frame contains both the interaction map and the corresponding radiance image. The radiance images have not been processed with noise and sensor effects in order to solely demonstrate the ray-tracing technique.

Figure 16 contains frames 1 through 8. This sequence is a fly-by parallel to the road. The viewing data is interpreted in the following manner. In frame 1, the camera is located at (1.6, -3.0, 0.7) which means 1.6 km to the right of the origin (+ x), 3 km away from the road (- y), at an altitude of 700 m (+ z). In order for the image plane to face the origin it has to be rotated 78.36° about the x axis and 28.07° about the z axis. The calculation of these angles is shown below:

$$\text{Rotation}(x) = \text{acos}\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right) \tag{13}$$

and

$$\text{Rotation}(z) = \text{atan}\left(\frac{x}{|y|}\right) \tag{14}$$

There will be a change in digital count for every 0.5° K change and there is a 1° FOV in both horizontal and vertical directions of the sensor.

There are many anomalies in thermal images that these few synthetic images succeed in demonstrating. The results of the ray-tracing are evident in the radiance from the trees, vehicle tires, and engine reflected off of the asphalt road which appears specular at grazing angles. The field was modelled as thick grass (lambertian surface) and shows no reflection characteristics. Close examination of the images reveal the effects of angular emissivity on radiance to the sensor from the vehicle body panels. Angular emissivity effects of asphalt are easily noticed in some images as the appearance of the road tends to completely blend in with the grass. Frames 15 and 16 of Figure 17 appear up-side-down which would be correct unless the camera was inverted. Notice that the trees are now in front of the mobile missile carrier. Frame 17 also demonstrates the important role the atmosphere has in IR imagery. The object radiance in Frames 9 and 10 are completely dominated by atmospheric attenuation and self-emission.

As nice as these images appear, they are only beneficial if their radiometry is correct. Therefore, the remainder of this section concentrates on the radiometric verification of the model.

TABLE 2

Viewing data for Figures 16-20

Frame #	Sensor Location X	Y	Z	Rotations x-axis	y-axis	z-axis	Gain	Field of View X	Y	IFOV X
		(km)			(degrees)		(DC/K)			(degrees)
Figure 16										
1	1.60	-3.00	0.70	78.3664	0.00	28.0725	2.0	1.0	1.0	0.039
2	0.80	-3.00	0.70	77.2948	0.00	14.9314	2.0	1.0	1.0	0.039
3	0.40	-3.00	0.70	76.9772	0.00	7.5946	2.0	1.0	1.0	0.039
4	0.00	-3.00	0.70	76.8660	0.00	0.0000	2.0	1.0	1.0	0.039
5	-0.40	-3.00	0.70	76.9772	0.00	-7.5946	2.0	1.0	1.0	0.039
6	-0.80	-3.00	0.70	77.2948	0.00	-14.9314	2.0	1.0	1.0	0.039
7	-1.60	-3.00	0.70	78.3664	0.00	-28.0725	2.0	1.0	1.0	0.039
8	-3.20	-3.00	0.70	80.9328	0.00	-46.8476	2.0	1.0	1.0	0.039
Figure 17										
9	0.00	-12.00	0.70	86.6615	0.00	0.0000	2.0	1.0	1.0	0.039
10	0.00	-9.00	0.70	85.5526	0.00	0.0000	2.0	1.0	1.0	0.039
11	0.00	-6.00	0.70	83.3456	0.00	0.0000	2.0	1.0	1.0	0.039
12	0.00	-3.00	0.70	76.8660	0.00	0.0000	2.0	1.0	1.0	0.039
13	0.00	-1.50	0.70	64.9831	0.00	0.0000	2.0	1.0	1.0	0.039
14	0.00	0.00	0.70	0.0000	0.00	0.0000	2.0	1.0	1.0	0.039
15	0.00	1.50	0.70	64.9831	0.00	0.0000	2.0	1.0	1.0	0.039
16	0.00	3.00	0.70	76.8660	0.00	0.0000	2.0	1.0	1.0	0.039
Figure 18										
17	-0.80	-3.00	0.70	77.2948	0.00	-14.9314	1.00	1.0	1.0	0.039
18	-0.80	-3.00	0.70	77.2948	0.00	-14.9314	2.00	1.0	1.0	0.039
19	-0.80	-3.00	0.70	77.2948	0.00	-14.9314	3.00	1.0	1.0	0.039
20	-0.80	-3.00	0.70	77.2948	0.00	-14.9314	4.00	1.0	1.0	0.039
Figure 19										
21	0.00	-3.00	0.70	76.8660	0.00	0.0000	2.0	1.0	2.0	0.039
22	0.00	-3.00	0.70	76.8660	0.00	0.0000	2.0	2.0	1.0	0.078
23	0.00	-3.00	0.70	76.8660	0.00	0.0000	2.0	3.0	3.0	0.0117
24	0.00	-3.00	0.70	76.8660	0.00	0.0000	2.0	4.0	2.0	0.0156

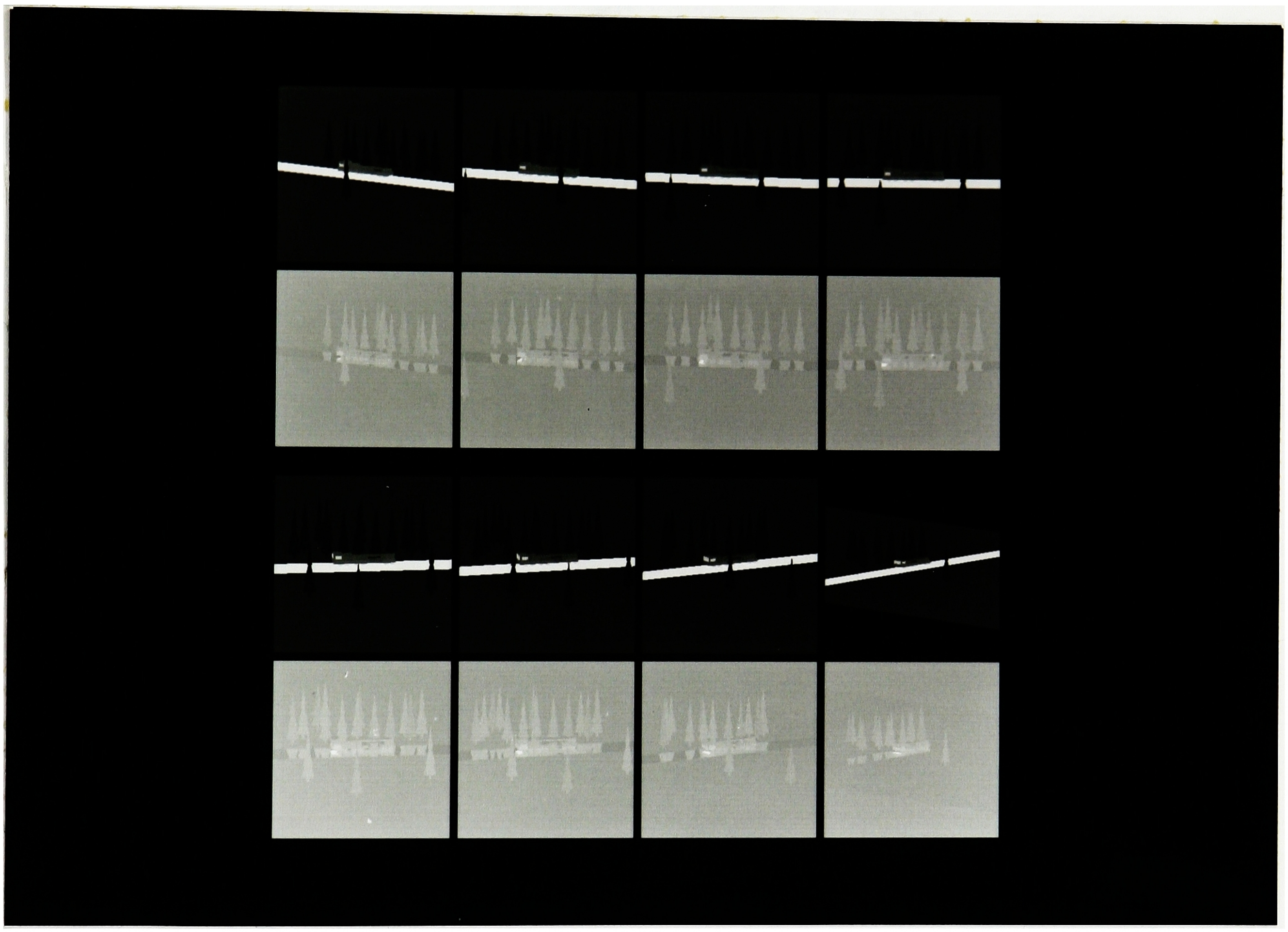


Figure 16 - Parallel fly-by, Frames 1 - 8

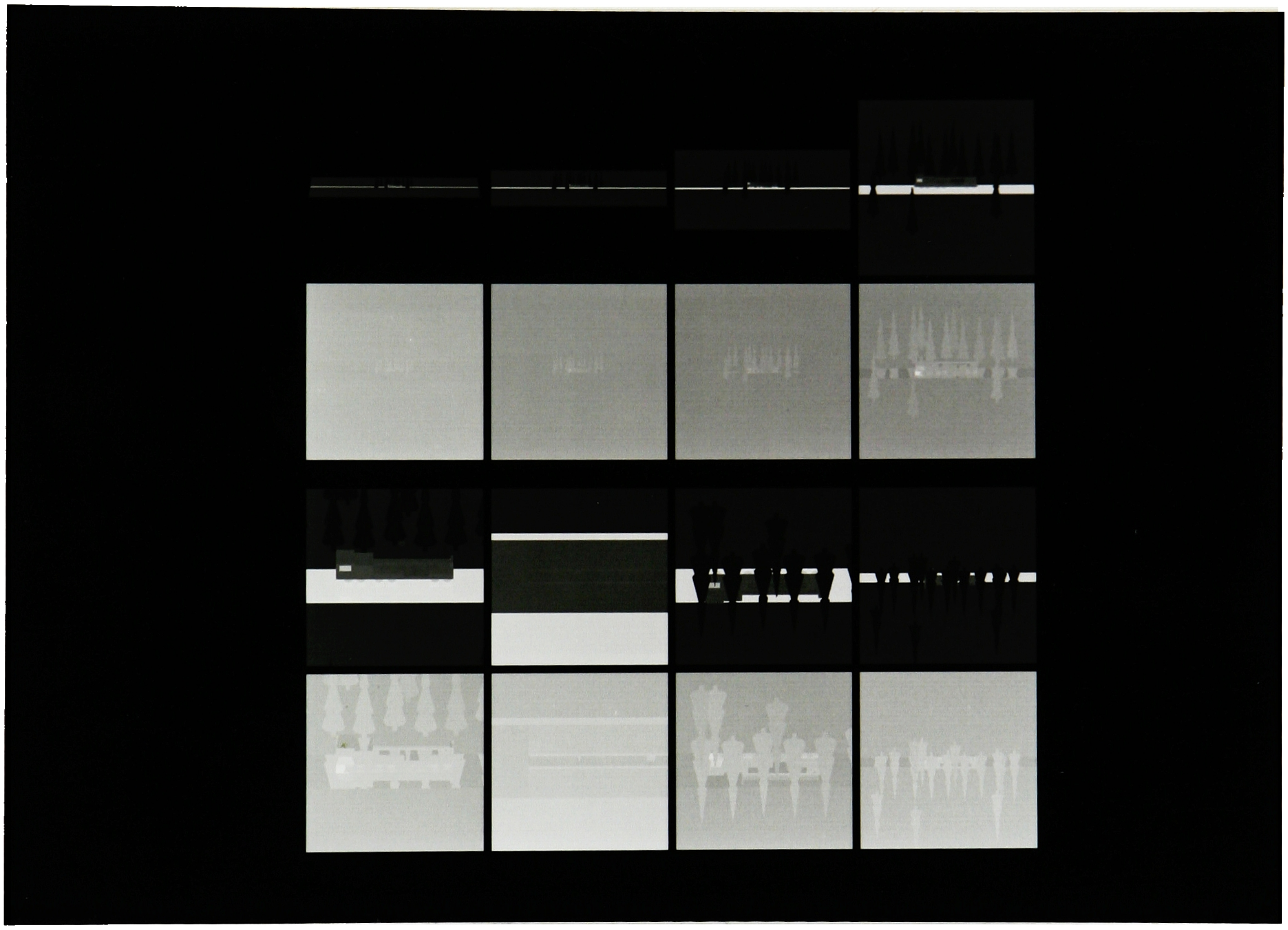


Figure 17 - Perpendicular fly-by, Frames 9 - 16

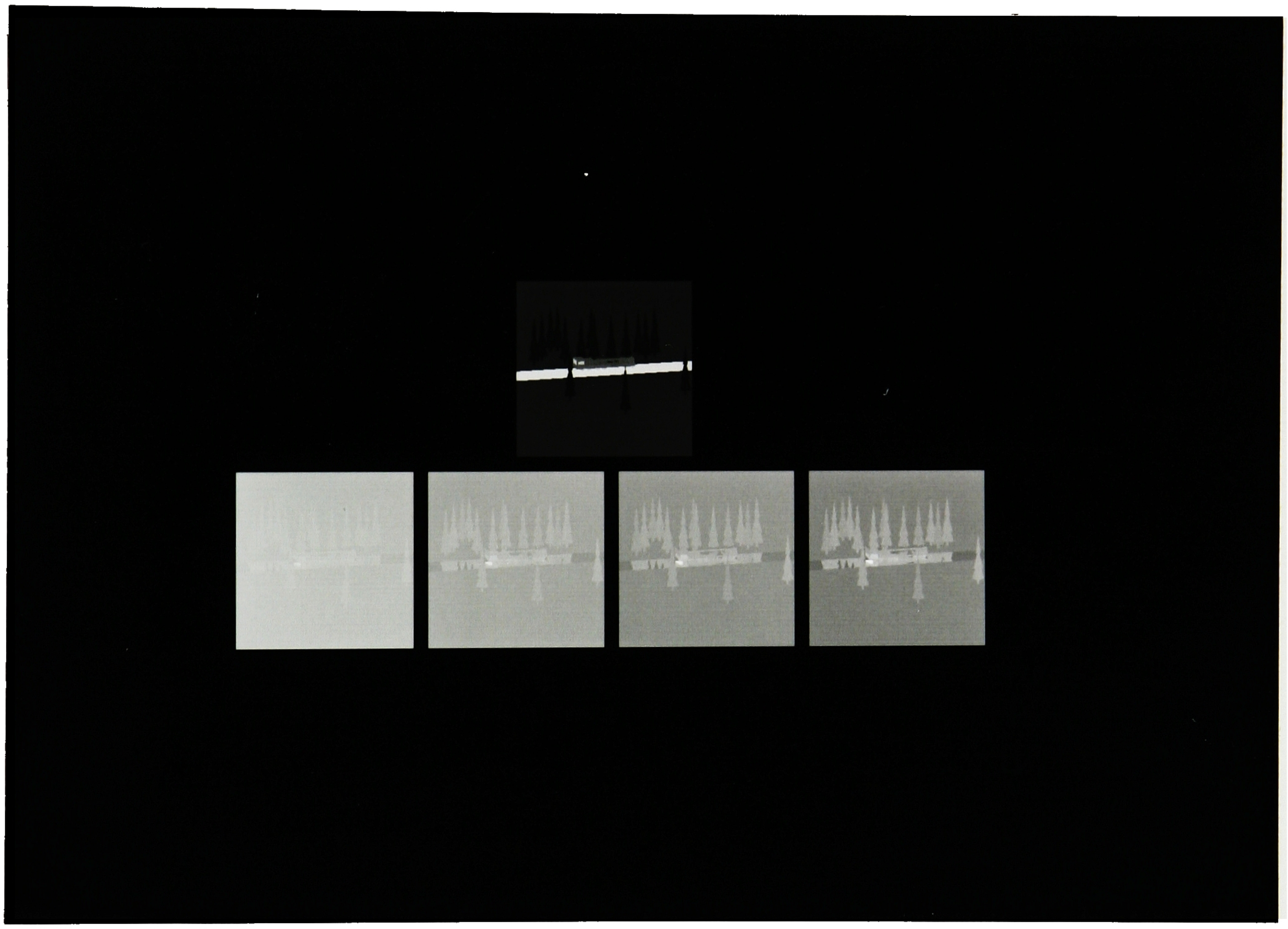


Figure 18 - Sensor Gain of 1,2, 3, and 4 DC/T, Frames 17 - 20

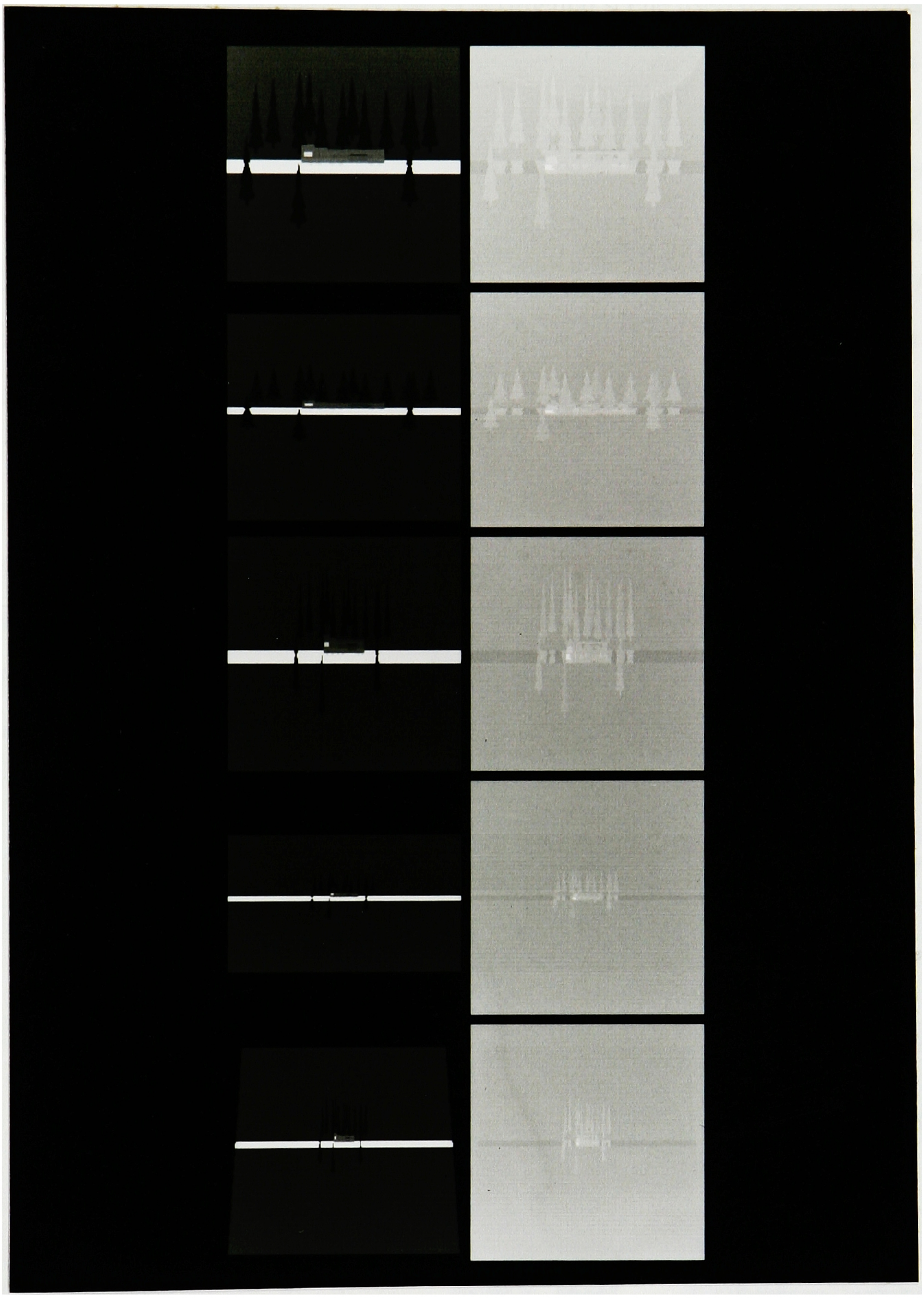


Figure 19 Frames 20-24 with changing field of views

4.5 Experimental Radiometric Results

Figure 20 shows an approximation of the experimental scene used for the radiometric verification. Actually, the pool and the background were switched. The scene was imaged through an Inframetrics IR camera (*cf.* Figure 21) at approximately 7:10 pm on March 8, 1990. The Inframetrics camera contains a single Mercury/Cadmium/Telluride detector cooled by liquid nitrogen to 77 K within an cryogenic dewar. It is sensitive to energy in the 8-12 μm bandpass. The camera incorporates electromechanical servos (galvanometers) to preform horizontal and vertical scanning. Horizontal scanning is performed at a very high rate (4 KHz) in a resonant (sinusoidal) mode. Vertical scanning is done in a sawtooth pattern commensurate with standard TV formats.

A portable Compaq computer running Werner Frei software was used to frame-grab the images on an Image Technology board. From this point on, when frame-grabbed IR images are referred to, they were frame-averaged 64 times to reduce noise effects as much as possible.

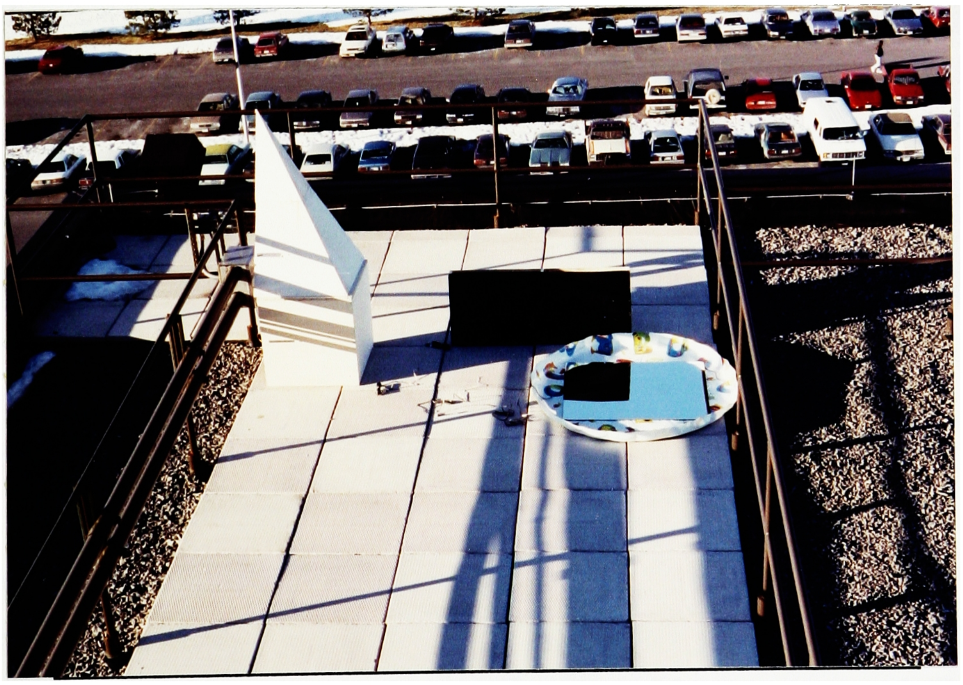


Figure 20 - Experimental scene set-up

Thermistors will be referred by their locations which were left and right side of the pool, top and bottom of the wooden pyramid structure, left concrete which was in front of the pyramid, right concrete which was in front of the Blackbody, and left and right sides of the blackbody. The camera was 4.77 m above and 8.61 m horizontally away from the objects. Parameters for the Inframetrics and Werner Frei can be found in Appendix F. Please note that the parking lot in the background is not part of the experimental scene and should be ignored.

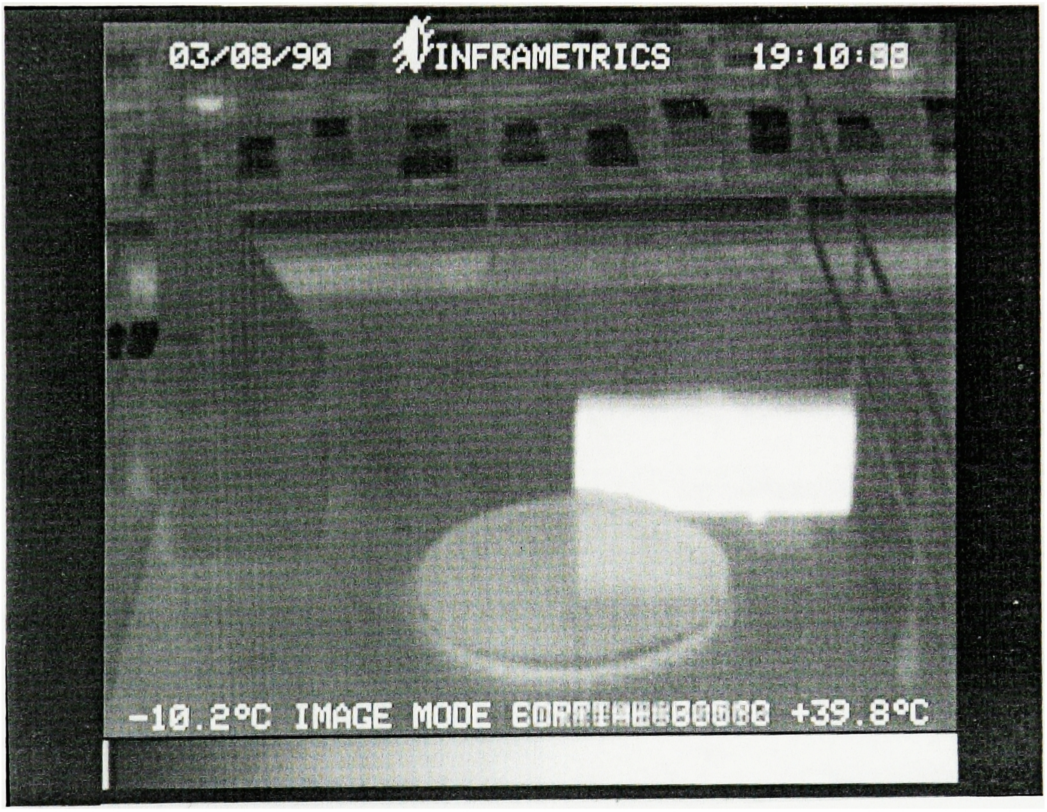


Figure 21 - Actual IR image of experimental scene

4.6 IR Camera Calibration

As previously discussed, a calibration must be done in order to be sure of the digital count to temperature relationship. Ideally, the camera should be checked for its spatial integrity, however, it was assumed to be spatially calibrated from previous work [Warnick (1990)]. In this experiment, a three point temperature calibration was performed. The blackbody (BB)was set to 5°, 15°, and 25° Celsius and images were taken of the BB while at these temperatures. Figure 22 shows an image of the BB.

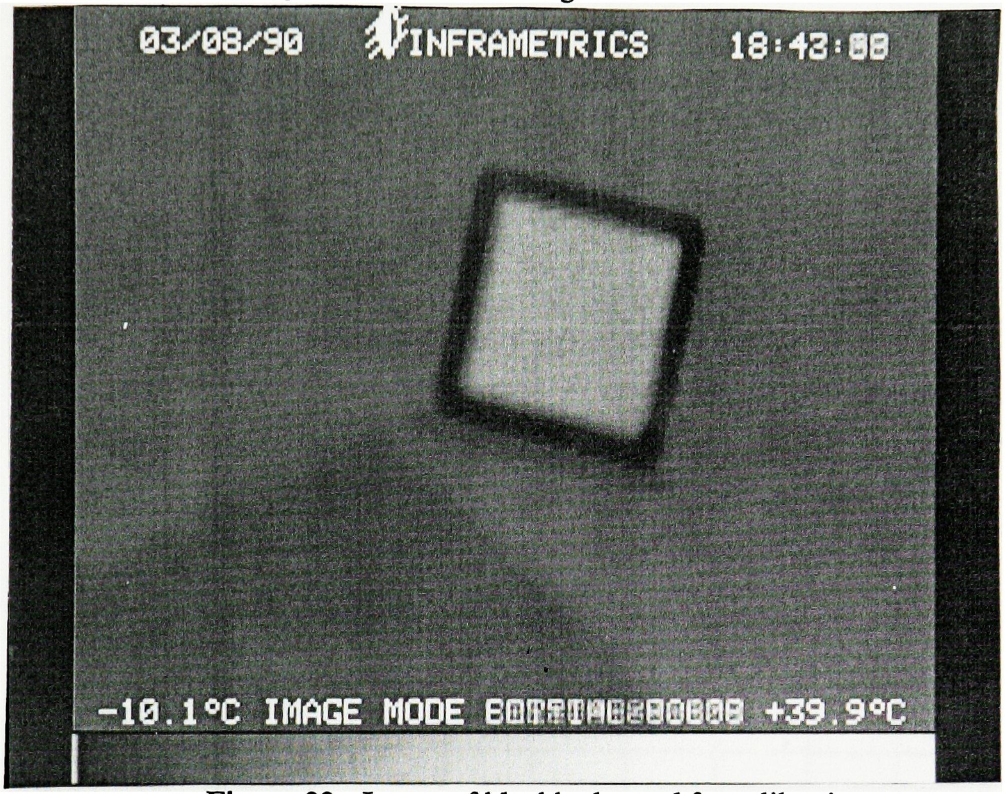


Figure 22 - Image of blackbody used for calibration

Local area histograms were done on all three of the BB images. The blackbody was assumed to have unit emissivity in order to simplify computations. The results of the local area histograms are shown in Table 3. Manufacturers specifications list 0.968 as the emissivity in the 8-12 μm region. The only way the remaining 3.2% reflectance could be properly handled is to use the downwelled radiance and exactly know the shape factor around the BB. The three calibration temperatures would be converted to apparent calibration temperatures when multiplied by the 0.968 emissivity.

The data in Table 3 was plotted and then linearly regressed to get a transfer function for DC to temperature. This graph and the corresponding linear equation are shown in Figure 23.

TABLE 3
Calibration Data

Blackbody Temperature	Mean Digital Count	Standard Deviation
5 ^{o}C	66	1
15 ^{o}C	117	1
25 ^{o}C	176	1

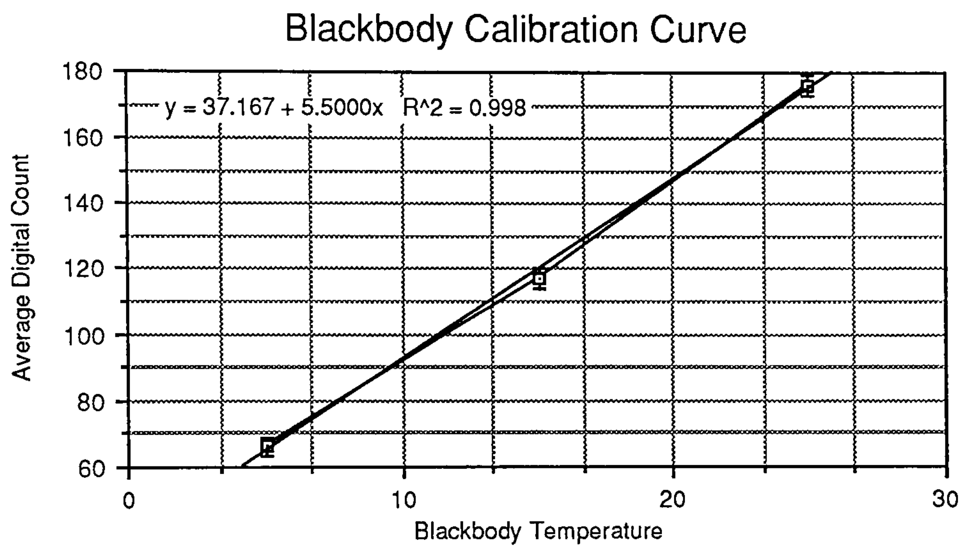


Figure 23 - Calibration curve for blackbody and IR images

Because the standard deviation of the averages was equal to one (instead of the desired zero), the calibration curve was also calculated for each of these y-bar errors. Using DCs of 65 and 177, the equation is:

$$y = 37.00 + 5.5x$$

and using DCs of 67 and 175 the equation is:

$$y = 40.00 + 5.4x$$

The uncertainty can be easily illustrated with a simple example. For a DC of 128 the temperature of 16.51° could also be 16.25° or 16.29° using the error bar equations.

However, these values should be on both sides of the mean value. The reason they are not is because the mean value used three points to calculate its regression equation while the error bars used only two. Therefore, the mean curve was recalculated with only temperatures of 5° and 25°. This resulting equation (shown below) is only to be used for error bar comparison only.

$$y = 38.5 + 5.5x$$

For a DC of 128, the temperature is 16.27° which is ± 0.02° between the error bar calculations. This shows that the error due to window sampling the BB image could only result in a temperature difference of ± 0.02° different than the actual temperature.

If the BB was not assumed to have unit emissivity the mean calibration equation would have been

$$y = 37.167 + 5.68x$$

using a DC of 128, the temperature would have been 15.99° which is 0.52° different. However, this value does not take the shape factor and downwelled radiance into account.

4.7 Thermistor Data

There were 8 high precision thermistors placed on various objects in the scene. Once a resistance was measured it was linearly interpolated with the calibration data to yield approximate temperatures. These temperatures are the "truth" for the experiment and are used as inputs to the synthetic scene simulator. Averages of objects were taken and entered into the model as the entire object temperature. The data are listed in Table 4.

TABLE 4
Thermistor Data

Thermistor Location	Resistance [Ω]	Temperature[^{o}C]	Average [^{o}C]
Left Pool	3.75K	13.77	13.35
Right Pool	3.90K	12.93	13.35
Left Concrete	6.44K	2.63	3.235
Right Concrete	6.06K	3.84	3.235
Bottom Pyramid	7.00K	0.97	
Top Pyramid	7.62K	-0.68	
Left Blackbody	1293	38.15	34.99
Right Blackbody	1680	31.83	34.99

4.8 Temperatures Extracted from the Actual IR Image

Once the regression equation from DC to temperature has been calculated, other DC's may be converted via the transfer function. Table 5 contains the average DC (acquired from Figure 21) for a region on the listed object. These region averages are then converted and their corresponding apparent temperatures are shown. These temperatures are the ones that will be used for a comparison to the synthetic scene simulator's predictions.

TABLE 5
Converted actual IR Temperatures

Thermistor Location	Average DC	Std. Dev.	Temperature [^{o}C]
Left Pool	90	1	9.61
Right Pool	108	3	12.88
Left Concrete	50	1	2.36
Right Concrete	52	3	2.72
Bottom Pyramid	49	1	2.17
Top Pyramid	35	1	-0.37
Left Blackbody	241	5	37.02
Right Blackbody	233	4	35.57

The standard deviations associated with the average digital counts make it difficult to present a single correct temperature for comparison. However, these values can also aid in error analysis. Using Left Blackbody's statistics as an example, a DC of 236 (241-5) would yield a temperature of 36.11°C through the calibration curve, and a DC of 246 would yield a temperature of 37.93°C. Therefore, in the extreme case, there is at least 1°C error due to the window averaging.

4.9 The Synthetic Image

The computer simulation has been previously discussed and the reader should be aware of how the simulator works. Figures 24a and 24b show the interaction map and its corresponding radiometric image. Table 6 contains the analysis of similar regions of interest used previously in the real IR image, the average DC of that region of interest, and the representative temperature of that DC, as calculated by the SIG via the LUT produced.

TABLE 6
Temperatures from the SIG

Thermistor Location	Average DC	Std. Dev.	Effective Temperature [°C]
Left Pool	103	1	10.04
Right Pool	128	1	15.31
Left Concrete	63	1	1.05
Right Concrete	63	1	1.05
Bottom Pyramid	64	1	1.20
Top Pyramid	43	1	-3.97
Left Blackbody	221	1	32.34
Right Blackbody	215	1	31.29

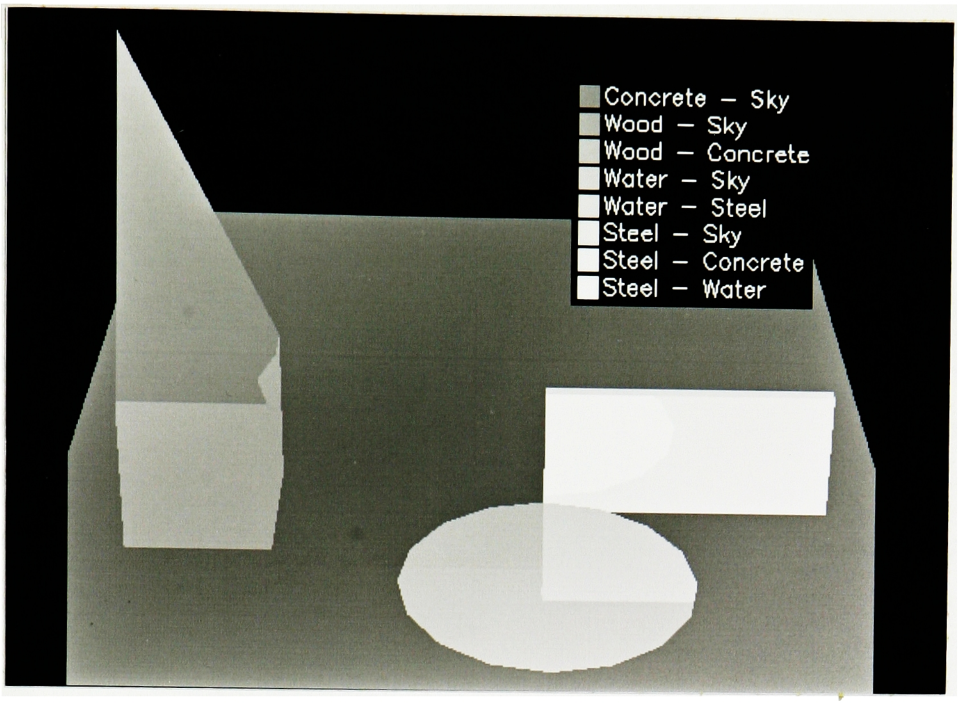


Figure 24a - Interaction map from the SIG

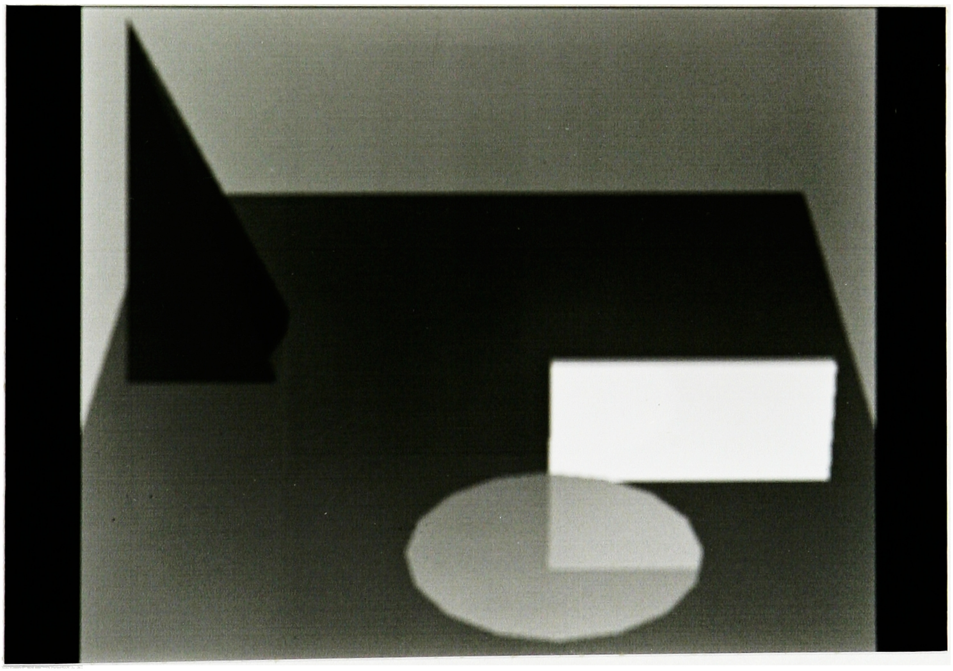


Figure 24b - Radiometric synthetic image

5.0 Conclusions and Recommendations

A visual comparison of the real and synthetic images shows that they appear very similar. The geometrical capabilities of the simulator are exemplified by the similarity of the two images. The SIG process being demonstrated did show many of the IR anomalies that occur in actual IR imagery. One can see reflection from the background upon the pool and no reflection on the concrete to the right of the pool. Also, the gradient on the right side of the pyramid on the predicted IR image can easily be attributed to angular emissivity effects since the temperature was modelled uniformly.

5.1 Error Analysis

In order to understand the radiometric accuracy of the simulator, further discussion is necessary. Table 7 contains all the temperatures of the objects throughout the experiment in rank order. In addition, an approximate value for the emissivity is listed.

TABLE 7

Radiometric Test Data (in ^{o}C)

	Assigned Temperature		Apparent Temperature of		
Location	Thermistor	Average	Real IR	$\varepsilon(60^{o})$	Synthetic IR
Top Pyramid	-0.68		-0.36	0.843	-3.97
Left Concrete	2.63	3.23	2.36	0.944	1.05
Right Concrete	3.84	3.23	2.71	0.944	1.05
Bottom Pyramid	0.97		2.17	0.843	1.20
Left Pool	13.77	13.35	9.62	0.925	10.04
Right Pool	12.93	13.35	12.88	0.925	15.31
Left Blackbody	38.15	34.99	37.02	0.747	32.34
Right Blackbody	31.83	34.99	35.57	0.747	31.29

Absolute temperature measurement is a very difficult task. Doing it outdoors just increases the complexity involved in getting accurate results. However, there was an attempt to keep many of the usual outdoor problems to a minimum. For example, the experiment was done after sunset so that sunlight could not increase the temperature. However, water was used in both the pool and in the background object and some water did get on object's surfaces which changes their emissivity. Obviously, the wind can not be controlled and its affect on apparent temperature causes both increasing and decreasing values towards the air ambient temperature.

There are many individual reasons that all contribute to temperature errors. There is an average error of ± 2.50°C with a standard deviation of 1.526°C. They were calculated by taking the absolute difference between apparent real IR and synthetic IR temperatures. The temperatures reported for the actual IR image were calculated using local area sampling and a calibration curve. This calibration curve was also a function of the window sampling operation. In the case of the calibration curve, a standard deviation of one was reported. It was then shown that errors due to the standard deviation would only account for a few tenths of a degree Celsius . The local area histogram operation had standard deviations ranging from 1 to 5. It was shown previously that this could contribute up to ± 1°C errors.

Appendix B exemplified the fact that errors as large as ± 1.5°C can be easily introduced by not including the shape factor. Concrete was the only diffuse material. The IR camera was on a wall 4.77 m above the experimental scene. There was definitely a shape factor involved (approximately 75%, and ignored) with a wall that close to the scene.

A significant reason the temperatures are different is because only eight thermistors were used. A single centimeter point was used as a representative sample of an entire surface. These thermistor temperatures were the inputs to the entire model. The manufacturer's (YSI Incorporated) specification of the thermistors listed them as accurate within ± 0.1°C.

In addition, the emissivity values used had a ± 0.01 error associated with them.

The total temperature error can be summarized as:

± 0.1°C thermistors ± 0.02°C calibration ± 1.0°C local area histogramming

± 1.5°C shape factor ± 0.25°C one bounce assumption = ± 2.87°C

along with some angular emissivity error which was reported low (± 0.01).

5.2 Recommendations

The SIG process presented here is only applicable to 8-14 μm imagery. In order to make the process run in the 3-5 μm wavelengths, solar flux must be accounted for. The code was written for easy incorporation of this modification.

The current code does not contain the following recommended improvements. The description of the objects is still one of the hardest parts of the entire SIG process. A decision must be made whether objects composed of surfaces (*i.e.* facets) should be solely 3 vertex or improved to include 4 vertices. Many surfaces can be rendered with 4 points rather than 6 points (2 facets). This would result in less data for some objects. It is a definite recommendation that object-editing code (*i.e.* the ability to address individual facets) must be able to address each facet of an object and be capable of changing its material index and its temperature.

The shape factor (F) parameter that was ignored in this SIG model was shown to be in some cases a 1.5°C source of error. This parameter could be calculated and accounted for in the following manor. For facets flat on the ground, rays could be sent out (a few degrees above the Earth's surface) 360° (azimuthal) around the facet of interest. A list of (azimuthal) angles that hit objects would be recorded. Then for each azimuthal angle, rays for elevation angles would be continually cast out until the object was no longer being hit. Then move on to the next azimuthal angle in the hit list. For facets at some angle to the

Earth, a modification of this approach would have to be implemented.

The parameter used for system gain (DC/T) should be changed to have a sensor's NEΔT. This parameter is more widely specified for imaging systems. NEΔT is the sensor's noise equivalent power (NEP) converted to the equivalent step in the temperature of a blackbody which would cause the same increase in power at the detector. In addition, a system offset should be used so that the auto-biasing does not occur.

Finally, verification procedures need to be further examined. If more detailed, laboratory experiments were designed, then perhaps the model could be verified in small pieces instead of one scene. For example, imagine concrete in an open area with no background objects (and a very low shape factor). Next look solely at the pool of water. Bring in a more stable background object then the one used in this thesis. An ideal object would be a blackbody, however, one of considerable size (2-4 ft.) would be quite costly. Many more thermistors/surface need to be used in order to get better "truth" temperatures.

6.0 References

Ben-Shalom, A., "Sky Radiance at Wavelengths between 7 and 14 Macrometers: Measurement Calculation and Comparison with LOWTRAN-4 Predictions," *Applied Optics,* Vol. 19, No. 6, pp. 838-839 (1980)

Heiner Biesel, and T. Rohlfing, "Real-time Simulated Forward Looking Infrared (FLIR) Imagery for Training," *Infrared Image Processing and Enhancement*, Marshal R. Weathersby, Editor,Proc. SPIE 781, pp. 71-80 (1987)

Blinn, J. F., "Models of Light Reflection for Computer Synthesized Pictures," *Computer Graphics* (Proc. SIGGRAPH 77) 11:2, pp.192-198 (1977)

Botkin, E. *et. al.*, "Infrared Modeling and Analyses (IRMA)," Report AFATL-TR-81-65, Grumman Aerospace Corporation, Bethpage, NY (1981)

Cathcart, J.M., and A.D. Sheffer, "Target and Background Infrared Signature Modeling for Complex Synthetic Scenes," *Infrared Systems and Components II*, H.M. Liaw, Editor, Proc. SPIE 890, pp. 95-103 (1988)

Clough, S.A., F.X. Kneizys, E.P. Shettle, and G.P. Anderson, "Atmospheric radiance transmittance: FASCOD2," *Proc. of the Sixth Conference on Atmospheric Radiation*," Williamsburg, VA American Meteorological Society, Boston, MA 141-144 (1986).

Crow, F. C., "The Aliasing Problem in Computer-Generated Shaded Images," *Communications of the ACM*, Vol. 20, No. 11, November 1977

Dippe, M.A.Z., and E.H. Wold,"Antialiasing Through Stochastic Sampling," SIGGRAPH 1985

Draper, N.R. and H. Smith, Applied Regression Analysis, 2nd Edition, Wiley and Sons Inc., New York, NY, p.117 (1981)

Dunn, A.R., G. Lindquist, and J.R. Meyer, "High Resolution of IR Scene Model," *Infrared Image Processing and Enhancement*, Marshal R. Weathersby, Editor, Proc. SPIE 781, pp. 95-98 (1987)

Gardner, G. Y., J. Mendelsohn, J. Kim, and W. Reynolds, "A Digital Scene Model for Simulation of Visual and Infrared Imagery," *Infrared Image Processing and Enhancement*, Marshal R. Weathersby, Editor, Proc. SPIE 781, pp. 81-86 (1987)

Glassner, A. "Ray Tracing in Computer Graphics," *Computers in Science"*, pp. 18 25, Sept/Oct (1987)

Gonzalez, R. C., and P. Wintz, Digital Image Procesing, Addison-Wesley Publishing Company, Reading, MA (1987)

Cook, R, A. Glassner, E. Haines, P. Hanrahan, P. Heckbert, and L.R. Speer, "Introduction to Ray Tracing," *Computer Graphics* SIGGRAPH '87 Course Notes

Kay, T.L., and J.T. Kajiya, "Ray Tracing Complex Scenes," SIGGRAPH 1986, pp. 269-278

Kneizys, F.X., E.P. Shettle, W.O. Galley, J.H. Chetwynd, Jr., L.W. Abrev, J.E.A. Selby, S.A. Clough, and R.W. Fenn, "Atmospheric Transmittance/Radiance Computer Code LOWTRAN 6, Air Force Geophysics Laboratory, Optical Physics Division, 1983

Kornfeld, G. H., "Digital Simulation of Precise Sensor Degradations Including Non-Linearities and Shift Variance," *Infrared Image Processing and Enhancement*, Marshal R. Weathersby, Editor, Proc. SPIE 781, pp. 63-70 (1987)

Lillesand, T.M., and R.W. Kiefer, 1987, Remote Sensing and Image Interpretation, John Wiley and Sons, Inc., New York

Magnenat-Thalmann, N., and D. Thalmann, "An Indexed Bibliography on Image Synthesis," IEEE CG&A, pp. 27-38, Aug. 1987

Schott, J.R., "Temperature Measurement of Cooling Water Dishcharged from Power Plants," *Photogrammetric Engineering and Remote Sensing,* Vol. 45, No. 6, pp. 753-761 (1979)

Schott, J.R., "Incorporation of Angular Emissivity Effects in Longwave Infrared Image Models," *Infrared Technology XII*, Proc. SPIE 685 (1986)

Schott, J.R., and C. Salvaggio, "L.W.I.R. Radiometric Modeling for use with Synthetic Scene Generation," Final Report Prepared for Contract RD-86-6843 (Task 3), Report No. RIT/DIRS 87/88-51-122

Schowengerdt, R.A., Techniques for Image Processing and Classification in Remote Sensing, Academic Press, Inc., New York (1983)

Sheffer, A.D., and J.M. Cathcart, "Computer generated IR imagery: a first principles modeling approach," Proc. SPIE 933 (1988)

Slater, P.N., Remote Sensing: Optics and Optical Systems, Addison Wesley, Reading, MA (1980)

Stets J, J. Conant, J. Gruninger, and B. Ryali., "Synthetic IR Scene Generation," *Infrared Systems and Components II*, H.M. Liaw, Editor, Proc. SPIE 890, pp. 130-146 (1988)

Turner, R.E. and T.L. Keller, "Analysis of Infrared Target Signature Models," Final Report, Science Applications International Corporation, Dayton, Ohio, April 1986

Warnick, J, "A Quantitative Analysis of a Self-emitting Thermal IR Synthetic Scene System" Masters Thesis, Rochester Institute of Technology, (1990).

Appendix A - 1 Bounce Assumption

The purpose of this appendix is to show how the "one bounce" assumption was made. Figure A-1 shows a scene with three elements: ground, house, and tree.

The target in this example is the ground, which is assumed to be specular in the infrared (*i.e.* such as water). The radiance leaving the ground in the direction of the sensor is L(0,θ). Since this is a specular object, the background radiance in this case is due to the house. If the house were also specular, its background radiance would be due to the tree. The question under consideration is how many bounces are significant ?

There are three major variables to test: target temperature, target emissivity, and background temperature. A FORTRAN program was written to calculate the radiance under the following conditions. Four cases were examined. The column headings are defined as follows:

a) **no bounce**: direct radiation from the target; (the ground).

b) **one bounce**: the flux was allowed to bounce from the ground and hit the house.

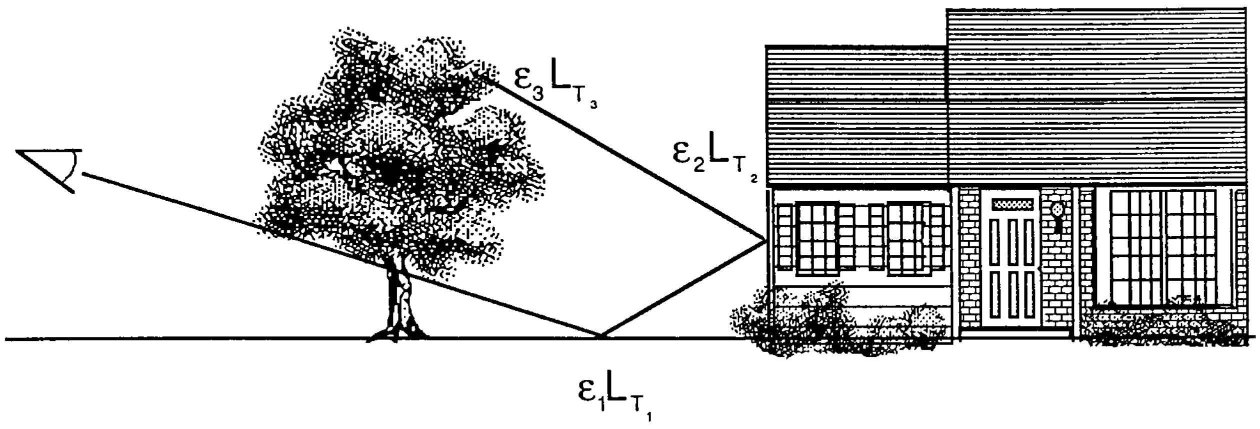
c) **two bounces**: a bounce from the ground to the house and into the tree (for simplicity, both the house and tree are assigned the same emissivity and temperature).

d) **three or more**: bounces off additional trees were possible.

The results are presented in Tables A-1, A-2, and A-3.

In each of the scenarios, the values do not change significantly after two or more bounces. The temperature falls within 0.25°K of the final (four-bounce) value after one bounce, except for extreme conditions (such as very high background temperatures and low emissivities). The results show that the background radiance reflected from a specular object can generally be represented by the radiance from the first background object only. This result is only applicable to earth ambient temperatures and more bounces have to be accounted for if "hot" objects (such as engines and plumes) were trying to be modeled.

Multiple Scattering with Specular Materials



$$L(0,\theta) = \varepsilon_1 L_{T_1} + r_1 L_{b_1}$$

$$L_{b_1} = \varepsilon_2 L_{T_2} + r_2 L_{b_2}$$

$$L_{b_2} = \varepsilon_3 L_{T_3}$$

$$L(0,\theta) = \varepsilon_1 L_{T_1} + r_1(\varepsilon_2 L_{T_2} + r_2 L_{b_2})$$

$$L(0,\theta) = \varepsilon_1 L_{T_1} + r_1(\varepsilon_2 L_{T_2} + r_2 \varepsilon_3 L_{T_3})$$

Figure A-1

Target Emissivity = 0.98 Background Temperature = 300.00K

Background Emissivity = 0.95

Temperature	No Bounces	1 Bounce	2 Bounces	3 Bounces	4 Bounces
260.0000	258.9804	260.9005	260.9953	261.0000	261.0002
270.0000	268.9044	270.6074	270.6916	270.6959	270.6961
280.0000	278.8316	280.3563	280.4319	280.4356	280.4358
290.0000	288.7520	290.1292	290.1976	290.2011	290.2013
300.0000	298.6722	299.9259	299.9884	299.9913	299.9915
310.0000	308.5894	309.7388	309.7960	309.7988	309.7989
320.0000	318.5048	319.5652	319.6180	319.6206	319.6208
330.0000	328.4170	329.4012	329.4503	329.4528	329.4529
340.0000	338.3282	339.2466	339.2923	339.2946	339.2947
350.0000	348.2372	349.0980	349.1408	349.1429	349.1431
360.0000	358.1433	358.9538	358.9943	358.9963	358.9964
370.0000	368.0480	368.8143	368.8525	368.8543	368.8543
380.0000	377.9501	378.6772	378.7134	378.7153	378.7154
390.0000	387.8518	388.5440	388.5785	388.5803	388.5804
400.0000	397.7500	398.4109	398.4441	398.4456	398.4457

Table A-1 Variable Target Temperature

Background Temperature = 300.0K Background Emissivity = 0.95

Object Temperature = 300.00K

Emissivity	No Bounces	1 Bounce	2 Bounces	3 Bounces	4 Bounces
0.60	269.6323	298.6722	299.9259	299.9884	299.9913
0.62	271.4088	298.7386	299.9292	299.9885	299.9913
0.64	273.1497	298.8049	299.9325	299.9886	299.9913
0.66	274.8568	298.8712	299.9358	299.9887	299.9913
0.68	276.5318	298.9376	299.9391	299.9889	299.9913
0.70	278.1763	299.0036	299.9424	299.9891	299.9913
0.72	279.7919	299.0697	299.9456	299.9893	299.9913
0.74	281.3799	299.1358	299.9489	299.9895	299.9914
0.76	282.9417	299.2019	299.9523	299.9897	299.9914
0.78	284.4782	299.2678	299.9554	299.9897	299.9914
0.80	285.9908	299.3340	299.9587	299.9899	299.9914
0.82	287.4804	299.3999	299.9620	299.9900	299.9914
0.84	288.9479	299.4658	299.9653	299.9902	299.9914
0.86	290.3943	299.5315	299.9685	299.9903	299.9914
0.88	291.8203	299.5974	299.9718	299.9905	299.9914
0.90	293.2270	299.6632	299.9752	299.9907	299.9914
0.92	294.6147	299.7289	299.9784	299.9909	299.9914
0.94	295.9845	299.7946	299.9817	299.9911	299.9914
0.96	297.3367	299.8603	299.9850	299.9911	299.9915
0.98	298.6722	299.9259	299.9884	299.9913	299.9915
1.00	299.9914	299.9915	299.9915	299.9915	299.9915

Table A-2 Variable Target Emissivities

Target emissivity = 0.98 Background Emissivity = 0.95

Target Temperature = 300.00K

Temperature	No Bounces	1 Bounce	2 Bounces	3 Bounces	4 Bounces
240.0000	298.6722	299.0815	299.1019	299.1029	299.1029
250.0000	298.6722	299.1827	299.2082	299.2095	299.2095
260.0000	298.6722	299.2992	299.3304	299.3320	299.3321
270.0000	298.6722	299.4312	299.4691	299.4709	299.4710
280.0000	298.6722	299.5795	299.6247	299.6268	299.6270
290.0000	298.6722	299.7443	299.7975	299.8003	299.8005
300.0000	298.6722	299.9259	299.9884	299.9913	299.9915
310.0000	298.6722	300.1246	300.1967	300.2002	300.2000
320.0000	298.6722	300.3401	300.4229	300.4271	300.4273
330.0000	298.6722	300.5729	300.6669	300.6717	300.6718
340.0000	298.6722	300.8223	300.9288	300.9340	300.9343
350.0000	298.6722	301.0886	301.2080	301.2140	301.2142
360.0000	298.6722	301.3712	301.5044	301.5110	301.5114
370.0000	298.6722	301.6701	301.8178	301.8252	301.8256
380.0000	298.6722	301.9847	302.1476	302.1558	302.1563
390.0000	298.6722	302.3148	302.4939	302.5028	302.5032
400.0000	298.6722	302.6599	302.8557	302.8654	302.8659

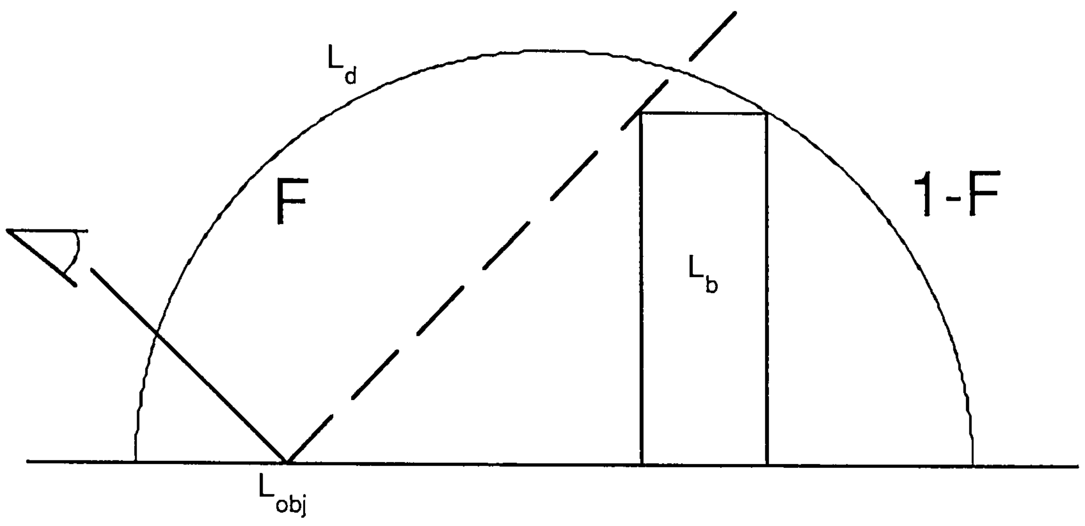
Table A-3 Variable Background Temperature

Program Bounce

```
        Implicit None

        Real*4 e(6), Lt(6), T(6), L(6), Emiss, Temp
        Real*4 Planck, Inv_Planck, Reflect
        Integer*2       I,J

        e(1) = 0.98     !Target emissivity

        Do I = 2, 6
           e(I) = 0.95  !Background Emissivity
        End Do

        T(1) = 300      !Target Temperature

*       Do I = 2, 6
*          T(I) = 300   !Background Temperature
*       End Do

        Lt(1) = Planck( T(1) )  !Target Radiance
*       Lt(2) = Planck( T(2) )  !Background Radiances

*       Do I = 3, 6
*          Lt(I) = Lt(2)        !Copy to others
*       End Do

        Open (2, Name = 'ans.out', Status = 'new')
        Write(2,*) 'Temperature'

*       Do Emiss = 0.6, 1.00, 0.02

        Do Temp = 240.0, 400.0, 10.0
           T(2) = Temp
           Lt(2) = Planck( T(2) )       !Background Radiance

           Do I = 3, 6
              Lt(I) = Lt(2)     !Copy to others
           End Do

           L(1) = e(1) * Lt(1)

           Do I = 2, 5
              L(I) = L(I-1) + Reflect(e, I) * e(I)*Lt(I)
           End Do

           Write(2,99) T(2), ( Inv_Planck( L(J) ), J = 1, 5 )

99      Format(F8.2, 5F10.4)

        End Do
        Close(2)

        End

****************************************************************

        Real*4 Function Reflect(e, N)

        Real*4  e(6)
        Integer*2 N

        Reflect = 1

        Do I = 1, N-1
          Reflect = Reflect * (1-e(I))
        End Do

        Return
        End

****************************************************************

        Real*4 Function Planck(T)

        Real*4 R, C, H, K, W1, W2, T, J, RT, Inc
        Parameter ( C = 2.997925E8, H = 6.6256E-34, K = 1.38054E-23 )
        Parameter ( Pi = 3.141517, Inc = .0000000001 )

        W1 = 8 * .000001
        W2 = 14 * .000001

        R = 0.0
        Do J = W1, W2, Inc
           RT = 1.0/( J**5 * ( Exp((H*C)/(J*K*T)) - 1 ) ) * Inc
           R = R + RT
        End Do

        Planck = R * 2.0 * C**2 * H

        Return

        End

****************************************************************

        Real*4 Function Inv_Planck(Radnce)

        Real*4     C, H, K, Pi, Inc, Limit
        Real*4     Radnce, Lambda, Lamda1, Lamda2, TmpInc, Temp, AccRad

        Parameter ( C = 2.997925E8, H = 6.6256E-34, K = 1.38054E-23 )
        Parameter ( Pi = 3.141517, Inc = .000000001, Limit = .00001 )
        Lamda1 = 8 * .000001
        Lamda2 = 14 * .000001

*
***  Initialize the temperature and the temperature increment
*
        TmpInc = 100.0
        Temp   = 0.0
*
********  Search for the required temperature
*

  40    Temp   = Temp + TmpInc
        AccRad = 0.0

        Do 50 Lambda = Lamda1, Lamda2, Inc
           Extnce = ((2.0 * Pi * H * C**2) / (Lambda**5)) *
     1             (1.0 / (Exp((H * C) / (Lambda * K * Temp)) - 1)) *
     2             (Inc / Pi)
           AccRad = AccRad + Extnce
  50    Continue

*
********  Check if limits are met
*

        If ( AccRad .Lt. Radnce ) GoTo 40

        TmpInc = TmpInc / 10.0
        Temp = Temp - TmpInc - ( TmpInc * 10.0 )
        If ( TmpInc .Lt. Limit ) GoTo 70

        GoTo 40

 70     Inv_Planck = Temp
        Return
        End
```

Appendix B - Shape Factor Calculations

Figure B-1 shows a case where the shape factor may be significant.



$$L = \varepsilon L_{obj} + (1-\varepsilon)[FL_d + (1-F)L_b]$$

Figure B-1 Illustration of shape factor

In the scene, the sensor is viewing a diffuse target. As stated previously, the integrated downwelled radiance should be added to the radiance of the object. However, notice that only a fraction F of downwelled radiance should be added and an additional (1-F) of background radiance should also be included.

For us to apply this analysis to the ray-tracer, the complete BRDF of all the materials must be known. If that data were available, rays could be projected out to look for appropriate objects in the BRDF's path. However, BRDF data are not known and so the shape factor in Equation 6 was assumed to be negligible.

The implications of this can be seen from the data calculated in Table B-1. Values were calculated for a variety of target and background temperatures, emissivities, with F values of 1.00, 0.75, 0.50, 0.25, and 0.00. Because there are many conditions illustrated, analysis can only be done in a step-by-step basis.

The effect of emissivity can be seen by selecting a single target temperature and one background temperature. For a shape factor of 0.5, the temperatures range from 298.34°K to 293.23°K. This difference exemplifies the fact that the reflectance term (equal to 1 - ε) really effects the target radiance.

A second example comes from analyzing data with a single emissivity value. The additional amount of downwelled radiance is a function of the shape factor. For a cold target temperature (240.0 °K and F = 0.75),the target actually appears warmer (241.43 °K) due the the downwelled radiance. As the target temperature increases (360.0 °K), the target appears colder (356.52 °K).

The implication resulting from ignoring F from the radiance calculations is that there could be at least a 1.5 °K temperature error. This number was arrived at by averaging many of the cases in Table B-1 for F=1.0 versus F=0.75 for Earth ambient conditions. The errors introduced by ignoring this parameter are scene dependent. It is a function of the objects' location, temperature, and emissivity.

```
Enter Ld
0.0014
Enter emissivity
0.95
                ============= Tobject 240.00 =============
  Tback        1.00        0.75        0.50        0.25        0.0
240.0000     237.7909    238.3503    238.9049    239.4548    239.9999
280.0000     237.7909    239.0277    240.2411    241.4324    242.6027
320.0000     237.7909    240.0546    242.2423    244.3605    246.4146
360.0000     237.7909    241.4325    244.8838    248.1689    251.3073
                ============= Tobject 270.00 =============
  Tback        1.00        0.75        0.50        0.25        0.0
240.0000     267.2572    267.6315    268.0041    268.3751    268.7445
280.0000     267.2572    268.0868    268.9083    269.7219    270.5278
320.0000     267.2572    268.7816    270.2791    271.7507    273.1979
```

```
360.0000    267.2572   269.7220   272.1176   274.4490   276.7209
             ============== Tobject 290.00 ==============
  Tback       1.00       0.75       0.50       0.25       0.0
290.0000    286.8724   287.6636   288.4485   289.2273   289.9999
295.0000    286.8724   287.7291   288.5785   289.4206   290.2558
300.0000    286.8724   287.7978   288.7146   289.6231   290.5234
305.0000    286.8724   287.8696   288.8569   289.8345   290.8028
310.0000    286.8724   287.9447   289.0053   290.0549   291.0938
315.0000    286.8724   288.0228   289.1599   290.2843   291.3964
320.0000    286.8724   288.1040   289.3206   290.5226   291.7106
             ============== Tobject 295.00 ==============
  Tback       1.00       0.75       0.50       0.25       0.0
290.0000    291.7728   292.5266   293.2748   294.0178   294.7556
295.0000    291.7728   292.5890   293.3989   294.2025   294.9999
300.0000    291.7728   292.6545   293.5287   294.3957   295.2556
305.0000    291.7728   292.7229   293.6644   294.5975   295.5224
310.0000    291.7728   292.7944   293.8061   294.8080   295.8006
315.0000    291.7728   292.8689   293.9536   295.0271   296.0899
320.0000    291.7728   292.9464   294.1070   295.2548   296.3903
             ============== Tobject 300.00 ==============
  Tback       1.00       0.75       0.50       0.25       0.0
240.0000    296.6717   296.9460   297.2197   297.4927   297.7650
280.0000    296.6717   297.2805   297.8858   298.4878   299.0864
290.0000    296.6717   297.3913   298.1060   298.8161   299.5216
295.0000    296.6717   297.4509   298.2245   298.9926   299.7554
300.0000    296.6717   297.5133   298.3486   299.1774   299.9999
305.0000    296.6717   297.5788   298.4783   299.3705   300.2554
310.0000    296.6717   297.6470   298.6137   299.5717   300.5216
315.0000    296.6717   297.7182   298.7547   299.7814   300.7986
320.0000    296.6717   297.7922   298.9013   299.9992   301.0862
360.0000    296.6717   298.4878   300.2740   302.0319   303.7622
             ============== Tobject 305.00 ==============
  Tback       1.00       0.75       0.50       0.25       0.0
290.0000    301.5692   302.2574   302.9413   303.6211   304.2968
295.0000    301.5692   302.3144   303.0547   303.7901   304.5208
300.0000    301.5692   302.3742   303.1735   303.9671   304.7552
305.0000    301.5692   302.4367   303.2977   304.1520   305.0000
310.0000    301.5692   302.5020   303.4273   304.3448   305.2552
315.0000    301.5692   302.5702   303.5623   304.5457   305.5207
             ============== Tobject 330.00 ==============
  Tback       1.00       0.75       0.50       0.25       0.0
240.0000    326.0366   326.2513   326.4658   326.6799   326.8936
280.0000    326.0366   326.5134   326.9886   327.4622   327.9340
320.0000    326.0366   326.9152   327.7880   328.6553   329.5172
360.0000    326.0366   327.4622   328.8730   330.2695   331.6520
```

```
             ============== Tobject 360.00 ==============
  Tback        1.00       0.75       0.50       0.25       0.0
240.0000     355.3541   355.5305   355.7069   355.8831   356.0590
280.0000     355.3541   355.7461   356.1373   356.5275   356.9168
320.0000     355.3541   356.0768   356.7964   357.5129   358.2264
360.0000     355.3541   356.5276   357.6929   358.8504   359.9998

Enter Ld
0.0014
Enter emissivity
0.90
             ============== Tobject 240.00 ==============
  Tback        1.00       0.75       0.50       0.25       0.0
240.0000     235.5027   236.6572   237.7909   238.9049   239.9999
280.0000     235.5027   238.0404   240.4813   242.8345   245.1077
320.0000     235.5027   240.1092   244.4117   248.4577   252.2840
360.0000     235.5027   242.8347   249.4411   255.4854   261.0795
             ============== Tobject 270.00 ==============
  Tback        1.00       0.75       0.50       0.25       0.0
240.0000     264.4212   265.1941   265.9597   266.7184   267.4701
280.0000     264.4212   266.1291   267.8024   269.4429   271.0525
320.0000     264.4212   267.5454   270.5571   273.4669   276.2836
360.0000     264.4212   269.4432   274.1849   278.6859   282.9770
             ============== Tobject 290.00 ==============
  Tback        1.00       0.75       0.50       0.25       0.0
290.0000     283.6425   285.2710   286.8724   288.4485   289.9999
295.0000     283.6425   285.4052   287.1364   288.8380   290.5110
300.0000     283.6425   285.5457   287.4125   289.2447   291.0443
305.0000     283.6425   285.6927   287.7006   289.6688   291.5993
310.0000     283.6425   285.8459   288.0007   290.1099   292.1759
315.0000     283.6425   286.0055   288.3127   290.5678   292.7740
320.0000     283.6425   286.1714   288.6366   291.0426   293.3931
             ============== Tobject 295.00 ==============
  Tback        1.00       0.75       0.50       0.25       0.0
290.0000     288.4408   289.9925   291.5207   293.0264   294.5107
295.0000     288.4408   290.1206   291.7728   293.3989   294.9999
300.0000     288.4408   290.2545   292.0364   293.7879   295.5106
305.0000     288.4408   290.3947   292.3117   294.1936   296.0424
310.0000     288.4408   290.5409   292.5984   294.6158   296.5952
315.0000     288.4408   290.6931   292.8966   295.0544   297.1687
320.0000     288.4408   290.8515   293.2063   295.5090   297.7628
             ============== Tobject 300.00 ==============
  Tback        1.00       0.75       0.50       0.25       0.0
240.0000     293.2365   293.8025   294.3655   294.9255   295.4825
280.0000     293.2365   294.4903   295.7294   296.9542   298.1651
```

```
290.0000     293.2365    294.7177    296.1783    297.6192    299.0411
295.0000     293.2365    294.8400    296.4195    297.9759    299.5102
300.0000     293.2365    294.9680    296.6717    298.3486    299.9999
305.0000     293.2365    295.1019    296.9350    298.7373    300.5101
310.0000     293.2365    295.2416    297.2095    299.1420    301.0407
315.0000     293.2365    295.3871    297.4950    299.5624    301.5914
320.0000     293.2365    295.5384    297.7914    299.9985    302.1619
360.0000     293.2365    296.9543    300.5475    304.0273    307.4033
            ============== Tobject 305.00 ==============
  Tback        1.00        0.75        0.50        0.25        0.0
290.0000     298.0292    299.4457    300.8442    302.2252    303.5892
295.0000     298.0292    299.5627    301.0751    302.5671    304.0396
300.0000     298.0292    299.6853    301.3168    302.9247    304.5098
305.0000     298.0292    299.8134    301.5692    303.2977    305.0000
310.0000     298.0292    299.9471    301.8323    303.6860    305.5098
315.0000     298.0292    300.0865    302.1060    304.0896    306.0391
320.0000     298.0292    300.2314    302.3903    304.5084    306.5877
            ============== Tobject 310.00 ==============
  Tback        1.00        0.75        0.50        0.25        0.0
290.0000     302.8192    304.1761    305.5171    306.8426    308.1531
295.0000     302.8192    304.2884    305.7388    307.1710    308.5859
300.0000     302.8192    304.4059    305.9707    307.5144    309.0381
305.0000     302.8192    304.5287    306.2129    307.8728    309.5095
310.0000     302.8192    304.6569    306.4654    308.2460    310.0000
315.0000     302.8192    304.7904    306.7282    308.6340    310.5094
320.0000     302.8192    304.9293    307.0012    309.0366    311.0375
            ============== Tobject 315.00 ==============
  Tback        1.00        0.75        0.50        0.25        0.0
290.0000     307.6065    308.9085    310.1961    311.4701    312.7307
295.0000     307.6065    309.0161    310.4091    311.7861    313.1473
300.0000     307.6065    309.1289    310.6320    312.1163    313.5827
305.0000     307.6065    309.2468    310.8648    312.4611    314.0365
310.0000     307.6065    309.3700    311.1075    312.8203    314.5091
315.0000     307.6065    309.4982    311.3602    313.1937    314.9999
320.0000     307.6065    309.6315    311.6226    313.5813    315.5090
            ============== Tobject 320.00 ==============
  Tback        1.00        0.75        0.50        0.25        0.0
290.0000     312.3912    313.6422    314.8806    316.1068    317.3209
295.0000     312.3912    313.7457    315.0855    316.4109    317.7224
300.0000     312.3912    313.8542    315.2999    316.7289    318.1420
305.0000     312.3912    313.9676    315.5240    317.0612    318.5796
310.0000     312.3912    314.0859    315.7576    317.4071    319.0352
315.0000     312.3912    314.2093    316.0008    317.7670    319.5087
320.0000     312.3912    314.3376    316.2535    318.1406    319.9999
           ============== Tobject 330.00 ==============
  Tback        1.00       0.75       0.50       0.25       0.0
240.0000    321.9526   322.3951   322.8363   323.2759   323.7141
280.0000    321.9526   322.9342   323.9087   324.8760   325.8363
320.0000    321.9526   323.7582   325.5394   327.2973   329.0328
360.0000    321.9526   324.8761   327.7367   330.5384   333.2847
          ============== Tobject 360.00 ==============
  Tback        1.00       0.75       0.50       0.25       0.0
240.0000    350.5746   350.9379   351.3003   351.6620   352.0229
280.0000    350.5746   351.3808   352.1832   352.9816   353.7762
320.0000    350.5746   352.0592   353.5305   354.9887   356.4341
360.0000    350.5746   352.9817   355.3541   357.6929   359.9998


Enter Ld
0.0014
Enter emissivity
0.85
          ============== Tobject 240.00 ==============
  Tback        1.00       0.75       0.50       0.25       0.0
240.0000    233.1275   234.9173   236.6572   238.3503   239.9999
280.0000    233.1275   237.0378   240.7207   244.2078   247.5242
320.0000    233.1275   240.1637   246.5138   252.3298   257.7165
360.0000    233.1275   244.2082   253.7240   262.1520   269.7729
          ============== Tobject 270.00 ==============
  Tback        1.00       0.75       0.50       0.25       0.0
240.0000    261.4829   262.6820   263.8634   265.0279   266.1759
280.0000    261.4829   264.1239   266.6817   269.1631   271.5739
320.0000    261.4829   266.2906   270.8344   275.1502   279.2672
360.0000    261.4829   269.1634   276.2052   282.7363   288.8481
          ============== Tobject 290.00 ==============
  Tback        1.00       0.75       0.50       0.25       0.0
290.0000    280.3002   282.8179   285.2710   287.6636   289.9999
295.0000    280.3002   283.0245   285.6732   288.2519   290.7656
300.0000    280.3002   283.2405   286.0931   288.8651   291.5624
305.0000    280.3002   283.4660   286.5307   289.5028   292.3896
310.0000    280.3002   283.7012   286.9858   290.1646   293.2469
315.0000    280.3002   283.9458   287.4583   290.8506   294.1335
320.0000    280.3002   284.1998   287.9479   291.5599   295.0489
          ============== Tobject 295.00 ==============
  Tback        1.00       0.75       0.50       0.25       0.0
290.0000    284.9940   287.3933   289.7359   292.0254   294.2651
295.0000    284.9940   287.5904   290.1206   292.5890   294.9999
300.0000    284.9940   287.7965   290.5222   293.1767   295.7650
305.0000    284.9940   288.0117   290.9409   293.7882   296.5599
```

```
310.0000    284.9940    288.2363    291.3765    294.4232    297.3839
315.0000    284.9940    288.4699    291.8289    295.0815    298.2368
320.0000    284.9940    288.7124    292.2978    295.7627    299.1179
            ============== Tobject 300.00 ==============
  Tback       1.00        0.75        0.50        0.25        0.0
240.0000    289.6836    290.5613    291.4314    292.2941    293.1498
280.0000    289.6836    291.6239    293.5283    295.3983    297.2360
290.0000    289.6836    291.9743    294.2151    296.4089    298.5585
295.0000    289.6836    292.1626    294.5833    296.9495    299.2646
300.0000    289.6836    292.3596    294.9680    297.5133    299.9999
305.0000    289.6836    292.5655    295.3691    298.1005    300.7644
310.0000    289.6836    292.7801    295.7866    298.7104    301.5574
315.0000    289.6836    293.0035    296.2203    299.3429    302.3786
320.0000    289.6836    293.2355    296.6701    299.9977    303.2274
360.0000    289.6836    295.3985    300.8203    305.9883    310.9339
            ============== Tobject 305.00 ==============
  Tback       1.00        0.75        0.50        0.25        0.0
290.0000    294.3691    296.5599    298.7065    300.8116    302.8772
295.0000    294.3691    296.7401    299.0596    301.3307    303.5563
300.0000    294.3691    296.9288    299.4286    301.8726    304.2640
305.0000    294.3691    297.1259    299.8134    302.4367    305.0000
310.0000    294.3691    297.3313    300.2141    303.0232    305.7639
315.0000    294.3691    297.5454    300.6304    303.6316    306.5551
320.0000    294.3691    297.7677    301.0623    304.2618    307.3735
            ============== Tobject 310.00 ==============
  Tback       1.00        0.75        0.50        0.25        0.0
290.0000    299.0506    301.1491    303.2086    305.2311    307.2185
295.0000    299.0506    301.3220    303.5477    305.7303    307.8724
300.0000    299.0506    301.5029    303.9021    306.2516    308.5542
305.0000    299.0506    301.6919    304.2718    306.7945    309.2635
310.0000    299.0506    301.8890    304.6570    307.3591    310.0000
315.0000    299.0506    302.0943    305.0571    307.9450    310.7633
320.0000    299.0506    302.3076    305.4723    308.5520    311.5531
            ============== Tobject 315.00 ==============
  Tback       1.00        0.75        0.50        0.25        0.0
290.0000    303.7279    305.7414    307.7201    309.6658    311.5801
295.0000    303.7279    305.9073    308.0460    310.1465    312.2104
300.0000    303.7279    306.0810    308.3869    310.6483    312.8678
305.0000    303.7279    306.2625    308.7426    311.1714    313.5521
310.0000    303.7279    306.4518    309.1130    311.7155    314.2630
315.0000    303.7279    306.6490    309.4982    312.2803    314.9999
320.0000    303.7279    306.8540    309.8980    312.8657    315.7628
            ============== Tobject 320.00 ==============
  Tback       1.00        0.75        0.50        0.25        0.0
290.0000    308.4014    310.3360    312.2397    314.1138    315.9597
```

```
295.0000    308.4014    310.4955    312.5535    314.5771    316.5680
300.0000    308.4014    310.6626    312.8817    315.0611    317.2027
305.0000    308.4014    310.8371    313.2243    315.5655    317.8634
310.0000    308.4014    311.0193    313.5812    316.0905    318.5502
315.0000    308.4014    311.2090    313.9523    316.6356    319.2624
320.0000    308.4014    311.4062    314.3376    317.2006    319.9999
             ============== Tobject 330.00 ==============
  Tback       1.00        0.75        0.50        0.25        0.0
240.0000    317.7363    318.4217    319.1035    319.7817    320.4562
280.0000    317.7363    319.2546    320.7551    322.2385    323.7052
320.0000    317.7363    320.5239    323.2525    325.9257    328.5466
360.0000    317.7363    322.2386    326.5909    330.8068    334.8988
             ============== Tobject 360.00 ==============
  Tback       1.00        0.75        0.50        0.25        0.0
240.0000    345.6484    346.2102    346.7701    347.3279    347.8839
280.0000    345.6484    346.8943    348.1306    349.3571    350.5744
320.0000    345.6484    347.9397    350.1982    352.4254    354.6225
360.0000    345.6484    349.3573    352.9817    356.5276    359.9998


Enter Ld
0.0014
Enter emissivity
0.80
             ============== Tobject 240.00 ==============
  Tback       1.00        0.75        0.50        0.25        0.0
240.0000    230.6559    233.1275    235.5027    237.7909    239.9999
280.0000    230.6559    236.0190    240.9591    245.5538    249.8600
320.0000    230.6559    240.2183    248.5541    256.0069    262.7894
360.0000    230.6559    245.5542    257.7725    268.3039    277.6606
             ============== Tobject 270.00 ==============
  Tback       1.00        0.75        0.50        0.25        0.0
240.0000    258.4317    260.0887    261.7114    263.3018    264.8613
280.0000    258.4317    262.0679    265.5457    268.8822    272.0923
320.0000    258.4317    265.0167    271.1107    276.8024    282.1577
360.0000    258.4317    268.8826    278.1812    286.6216    294.3940
             ============== Tobject 290.00 ==============
  Tback       1.00        0.75        0.50        0.25        0.0
290.0000    276.8337    280.3002    283.6425    286.8724    290.0000
295.0000    276.8337    280.5829    284.1876    287.6624    291.0195
300.0000    276.8337    280.8783    284.7557    288.4839    292.0779
305.0000    276.8337    281.1865    285.3467    289.3365    293.1740
310.0000    276.8337    281.5074    285.9605    290.2195    294.3069
315.0000    276.8337    281.8411    286.5964    291.1325    295.4757
```

```
320.0000    276.8337   282.1871   287.2545   292.0746   296.6792
            ============== Tobject 295.00 ==============
  Tback       1.00       0.75       0.50       0.25       0.0
290.0000    281.4202   284.7242   287.9191   291.0147   294.0191
295.0000    281.4202   284.9940   288.4408   291.7728   294.9999
300.0000    281.4202   285.2761   288.9850   292.5617   296.0188
305.0000    281.4202   285.5706   289.5515   293.3811   297.0747
310.0000    281.4202   285.8772   290.1397   294.2302   298.1670
315.0000    281.4202   286.1960   290.7498   295.1086   299.2945
320.0000    281.4202   286.5268   291.3812   296.0156   300.4564
            ============== Tobject 300.00 ==============
  Tback       1.00       0.75       0.50       0.25       0.0
240.0000    286.0012   287.2133   288.4108   289.5940   290.7635
280.0000    286.0012   288.6751   291.2795   293.8193   296.2987
290.0000    286.0012   289.1558   292.2143   295.1847   298.0738
295.0000    286.0012   289.4138   292.7145   295.9132   299.0182
300.0000    286.0012   289.6836   293.2365   296.6717   299.9999
305.0000    286.0012   289.9652   293.7797   297.4599   301.0181
310.0000    286.0012   290.2587   294.3445   298.2771   302.0718
315.0000    286.0012   290.5638   294.9303   299.1230   303.1602
320.0000    286.0012   290.8805   295.5368   299.9968   304.2827
360.0000    286.0012   293.8195   301.0924   307.9164   314.3634
            ============== Tobject 305.00 ==============
  Tback       1.00       0.75       0.50       0.25       0.0
290.0000    290.5768   293.5938   296.5259   299.3796   302.1606
295.0000    290.5768   293.8409   297.0060   300.0804   303.0710
300.0000    290.5768   294.0993   297.5072   300.8103   304.0176
305.0000    290.5768   294.3691   298.0292   301.5692   305.0000
310.0000    290.5768   294.6503   298.5718   302.3566   306.0173
315.0000    290.5768   294.9427   299.1350   303.1718   307.0689
320.0000    290.5768   295.2465   299.7183   304.0147   308.1540
            ============== Tobject 310.00 ==============
  Tback       1.00       0.75       0.50       0.25       0.0
290.0000    295.1469   298.0370   300.8518   303.5967   306.2764
295.0000    295.1469   298.2740   301.3132   304.2715   307.1545
300.0000    295.1469   298.5219   301.7951   304.9747   308.0682
305.0000    295.1469   298.7809   302.2971   305.7062   309.0169
310.0000    295.1469   299.0506   302.8192   306.4654   310.0000
315.0000    295.1469   299.3313   303.3612   307.2520   311.0167
320.0000    295.1469   299.6228   303.9227   308.0654   312.0663
            ============== Tobject 315.00 ==============
  Tback       1.00       0.75       0.50       0.25       0.0
290.0000    299.7116   302.4844   305.1902   307.8335   310.4183
295.0000    299.7116   302.7120   305.6343   308.4839   311.2660
300.0000    299.7116   302.9502   306.0981   309.1622   312.1487
```

```
305.0000     299.7116     303.1989     306.5815     309.8678     313.0656
310.0000     299.7116     303.4582     307.0843     310.6006     314.0162
315.0000     299.7116     303.7279     307.6065     311.3602     314.9999
320.0000     299.7116     304.0081     308.1477     312.1460     316.0161
              ============== Tobject 320.00 ==============
  Tback        1.00         0.75         0.50         0.25         0.0
290.0000     304.2710     306.9352     309.5396     312.0879     314.5835
295.0000     304.2710     307.1542     309.9674     312.7155     315.4029
300.0000     304.2710     307.3832     310.4143     313.3702     316.2562
305.0000     304.2710     307.6225     310.8803     314.0518     317.1432
310.0000     304.2710     307.8719     311.3653     314.7597     318.0632
315.0000     304.2710     308.1316     311.8689     315.4938     319.0156
320.0000     304.2710     308.4014     312.3912     316.2535     319.9999
              ============== Tobject 330.00 ==============
  Tback        1.00         0.75         0.50         0.25         0.0
240.0000     313.3741     314.3197     315.2580     316.1893     317.1137
280.0000     313.3741     315.4658     317.5228     319.5467     321.5390
320.0000     313.3741     317.2063     320.9251     324.5398     328.0587
360.0000     313.3741     319.5468     325.4351     331.0748     336.4951
              ============== Tobject 360.00 ==============
  Tback        1.00         0.75         0.50         0.25         0.0
240.0000     340.5605     341.3345     342.1046     342.8708     343.6331
280.0000     340.5605     342.2754     343.9711     345.6481     347.3073
320.0000     340.5605     343.7097     346.7953     349.8212     352.7907
360.0000     340.5605     345.6483     350.5746     355.3541     359.9998
```

Table B-1 Effect of F on observed radiance.

Appendix C - Ray/Plane Intersection

The algorithms used to test the intersection (Plane, Box, and Facet) were taken from the "Introduction to Ray Tracing" SIGGRAPH Course Notes [Cook *et al.* (1987)].

A plane can be defined as the linear equation:

$$Ax + By + Cz + D = 0 \qquad \text{(C1)}$$

such that A, B, C are not all zero. The unit vector normal to this plane is defined as:

$$n = (A, B, C)$$

such that $A^2 + B^2 + C^2 = 1$.
The distance from the origin of the coordinate system to the plane is D. The sign of D determines on which side of the plane the origin lies.

A ray is defined as:

$$R_{origin} = R_o = [\, x_o, y_o, z_o \,]$$
$$R_{direction} = R_d = [\, x_d, y_d, z_d \,]$$

which defines a ray to be:

$$\text{set of points on ray} = R(T) = R_o + R_d(T) \qquad \text{(C2)}$$

where $T > 0$.

The distance from the origin of the ray to the intersection with the plane is derived by simply substituting the expansion of Equation C2 into the plane equation (C1) and solving for T:

$$A^*(X_o+X_d{}^*T) + B^*(Y_o+Y_d{}^*T) + C^*(Z_o+Z_d{}^*T) \qquad \text{(C3)}$$

$$T = -\frac{(A^*X_o + B^*Y_o + C^*Z_o + D)}{(A^*X_d + B^*Y_d + C^*Z_d)}$$

More formally, this equation is:

$$T = -\frac{(P_n \bullet R_o + D)}{P_n \bullet R_d} \qquad \text{(C4)}$$

To use (C3) more efficiently, first calculate the dot product:

dot product value with ray direction =

$$\text{Vprd} = A^*X_d + B^*Y_d + C^*Z_d \qquad \text{(C5)}$$

If $V_{prd} = 0$, then the ray is parallel to the plane and no intersection occurs. If $V_{prd} > 0$, the normal of the plane is pointing away from the ray. Since there objects are planar with only one side, our test ends here. If $V_{prd} < 0$, we say the ray hit the plane and then can calculate the intersection point:

$$R_i = (X_i, Y_i, Z_i) = [\, X_o+X_d{}^*T,\ Y_o+Y_d{}^*T,\ Z_o+Z_d{}^*T \,] \qquad \text{(C6)}$$

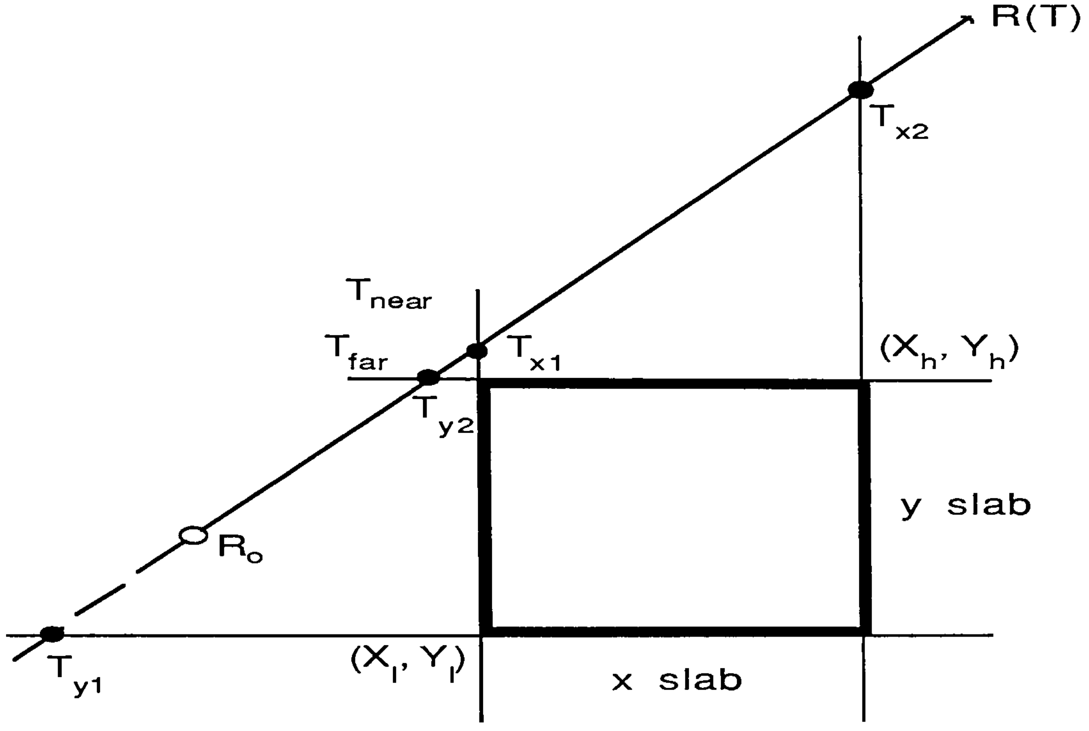
Appendix D - Ray/Box Intersection

A detailed explanation of the ray/box intersection test was presented by Kay and Kajiya (1986).

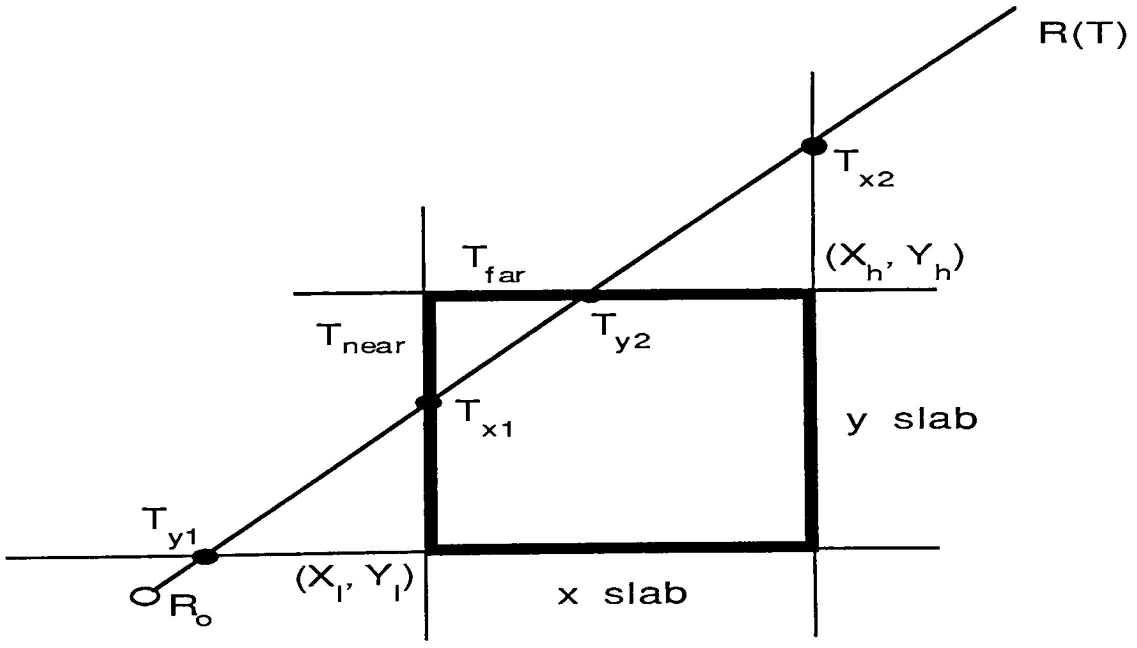
Define the orthogonal box by two coordinates:

$$\begin{aligned} \text{minimum extent of box} &= B_l = [X_l\ Y_l\ Z_l] \\ \text{maximum extent of box} &= B_h = [X_h\ Y_h\ Z_h] \end{aligned} \qquad \text{(D1)}$$

Figure D-1 shows pictorially the action of the algorithm. It will test first the z slabs (since most rays originate overhead), then the x slabs, and finally the y slabs. If the box is hit, the near and far intersection points distance is equal to T_{near}, and T_{far}, respectively.



$T_{near} > T_{far}$, so ray misses box



$T_{near} < T_{far}$, and $T_{near} > 0$, so T_{near} is intersection distance

Figure D-1

Appendix E- Path Extrapolation Technique

Figure E-1 shows a typical scenario for a grazing angle type of picture.

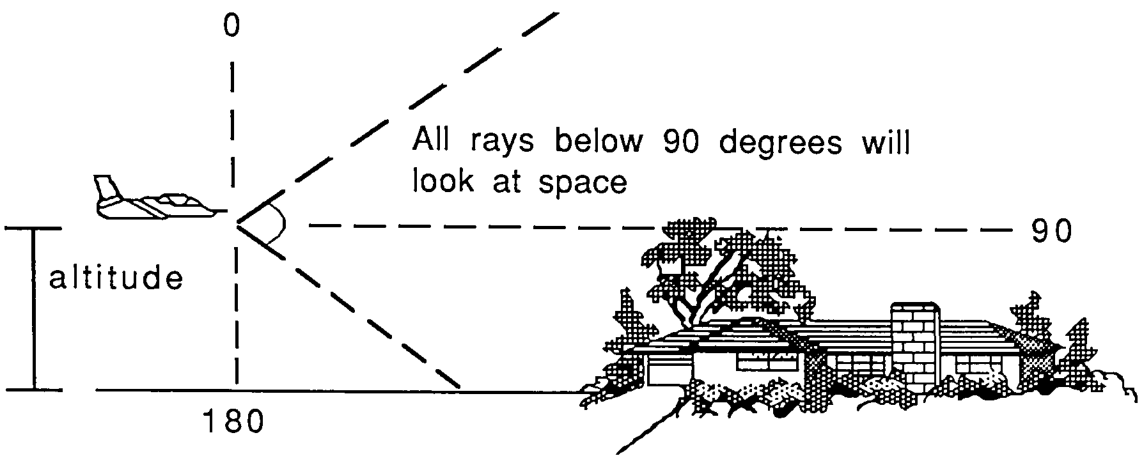


Figure E-1 Scenario for path radiance at some altitude

A limitation of the DIRTRAN program is that it only calculates radiance values when viewing the Earth. Using Figure E-1 as an illustration, these angles would range from 180 to around 95 degrees (depending on the altitude). Because of the scenario illustrated above, angles less than 95 are needed. Path radiance was calculated similarly to the way downwelled radiance is calculated. However, instead of summing atmospheric layers from the top down to the ground, this calculation stops at the sensors' altitude. A second-order polynomial extrapolation is then applied to the data for extreme angles.

The technique described was taken from [Draper and Smith (1981)]. The function is modeled by the equation:

$$y = a + bx + cx^2 \tag{E1}$$

where x is the angle from the normal and a,b, and c are the regression coefficients. The coefficients are obtained through the following matrix equation:

$$B = (X^tX)^{-1}X^tY \tag{E2}$$

where B, X, and Y are:

$$B = \begin{bmatrix} a \\ b \\ c \end{bmatrix}; \quad X = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ : & : & : \\ : & : & : \\ 1 & x_n & x_n^2 \end{bmatrix}; \quad Y = \begin{bmatrix} y_1 \\ y_2 \\ : \\ : \\ y_n \end{bmatrix}$$

However, because of the data's linearity for angles less than 60 degrees, the values used to calculate the coefficients begin at 60 degrees and stop at 83. The original data along with the extrapolated values are shown in the following two figures. Extrapolation values begin after 83 degrees.

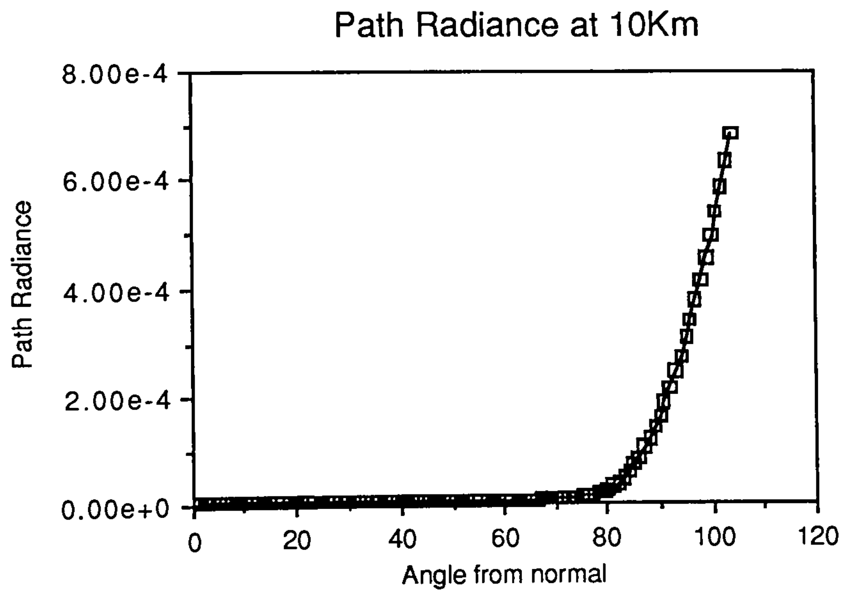


Figure E-2 Path radiance as a function of angle away from the normal at 10km

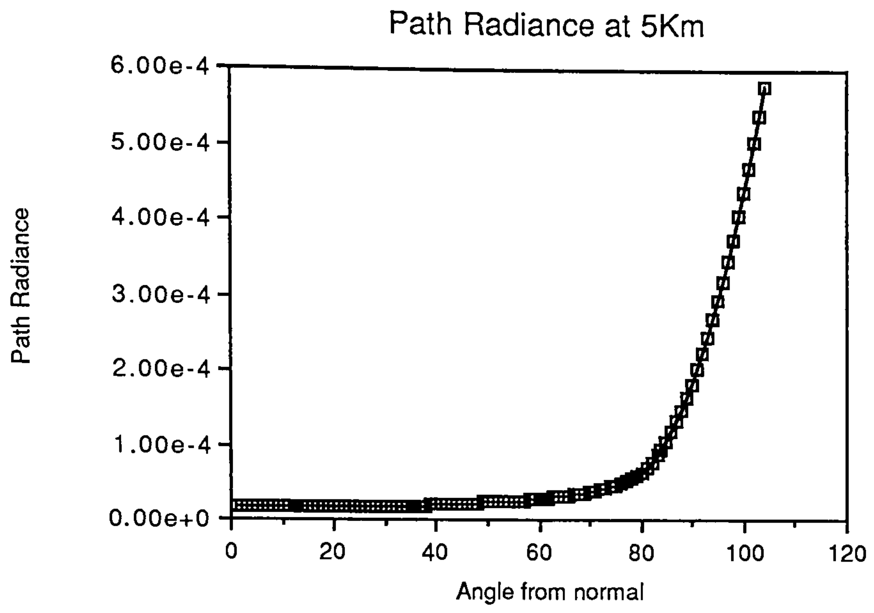


Figure E-3 Path radiance as a function of view angle away from the normal at 5km

There was an attempt to test how good of an approximations this extrapolation is. However, there were computational problems doing this. This model uses the data files provided by DIRTRAN. It does not calculate radiance values the same way as LOWTRAN 6. In order to make a similar comparison, the sensor has to be placed at the ground. This causes the extrapolation subroutine to include all atmospheric layers down to the ground thereby giving angular downwelled radiance values. This is the exact way DIRTRAN calculates downwelled radiance. Therefore, the comparison could have be made between LOWTRAN 6 and DIRTRAN. In addition, there is a loop control round off discrepancy between DIRTRAN (written in FORTRAN) and the ray-tracer (written in C) that causes the bottom layer to be included in DIRTRAN and not included in the ray-tracer. Therefore, due to the difficulties in getting LOWTRAN 6, DIRTRAN, and the ray-tracer to give exact numbers to compare rather then trends, it was decided not to compare this part of the code. Besides, in this thesis, it was not an original goal to include horizontal paths that didn't hit the Earth. This case was accidentally ran into while testing the versatility of the ray-tracer.

Appendix F Emissivity Data

The angular emissivity data that was used in throughout this SIG process has been presented in Table F-1. The only diffuse material in this study is concrete.

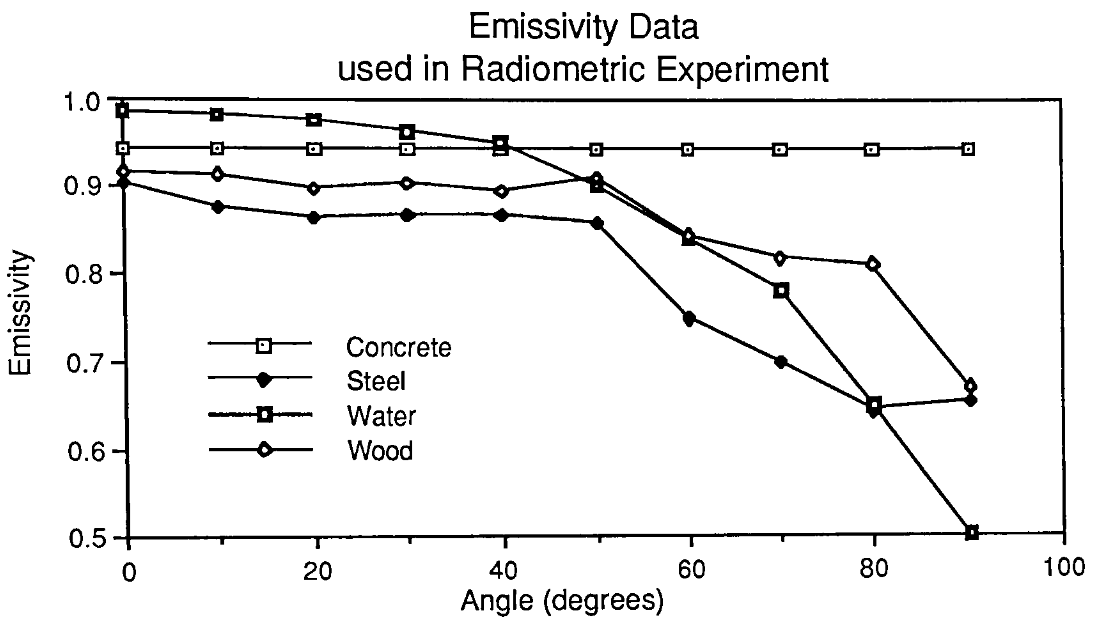


Table F-1 Graph of Material Emissivities

Appendix G Inframetrics / Werner Frei Specifications

Inframetrics, Inc.

IR Camera	Model 600
Spectral Bandpass	8-14 μm
Detector Type	HgCdTe (LN_2 Cooled)
Field of View	15 Deg. Vertical by 20 Deg. Horizontal
Resolution	2 mRad - 175 IFOV's Horizontal Resampled to 256 Pixels / Line 2 mRad - 130 IFOV's Vertical Resampled to 200 Lines / Frame
Dynamic Range	7 bits - 128 Grey Levels

Werner Frei Associates

Software	Imagelab
Dynamic Range	8 bits - 256 Grey Levels
Resolution	512 Horizontal Pixels 464 Vertical Pixels
Combined Resolution	256 Inframetrics Pixels Resampled to 499 Werner Frei Pixels (Horizontal) 200 Inframetrics Pixels Resampled to 408 Werner Frei Pixels (Vertical)

Appendix G - Selected code from LWIR ray-tracer

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <stddef.h>

#define MAX_OBJECTS   30           /* used for array allocation    */
#define MAX_MATERIALS 15           /* used for array allocation    */
#define NUMVERTS      3            /* Number of vertecies on each side.*/
#define M_PI          3.141592

#define max(x,y)      ( (x) < (y) ? (y) : (x) )
#define min(x,y)      ( (x) > (y) ? (y) : (x) )
#define sqr(x)        ( (x) * (x) )
#define cross(r,a,b)  { (r).x = (a).y * (b).z - (a).z * (b).y;\
                        (r).y = (a).z * (b).x - (a).x * (b).z;\
                        (r).z = (a).x * (b).y - (a).y * (b).x; }
#define is_zero(x)    ( (x) == (0.0) ? (1) : (0) )
#define sign(x)              ( (x) < (0.0) ? (-1) : (1) )

/* Define a point to be 3 coordinates */

typedef struct {
   double x, y, z;
} point_type;

typedef struct {
   double a,b,c,d;
} plane_type;

/* A Ray has an origin and direction */

typedef struct {
   point_type o, d;
} ray_type;

/* Define a facet to have a material type, temperature & radiance,
 * 3 points and a normal
 */

typedef struct {
   unsigned short matindex;
   point_type points[3];
   point_type normal;
   double temp;
   double radiance;
} facet_type;
```

```
/* Define a box as our bounding volume. */

typedef struct {
   point_type pmin, pmax;
} box_type;

/* Define an object to be an unknown number of facets */

typedef struct {
   long numfacets;
   facet_type *facets;       /* equivalent to facets[] */
   box_type box;
   char filname[15];
} object_type;

object_type scene[MAX_OBJECTS];

/*
typedef struct {
   long gain;
   double stddev;
   long blksize;
   char data_file[80];
   char title[30];
   long kernel;
   long dc[20];       /* two more the needed so the colormap ranges from 0 to
90 * /
} dirtran_format;
*/

typedef struct {
   point_type viewpoint;     /* Eye point */
   long xsize, ysize;        /* Number of pixels */
   double omega;             /* rotation about x axis */
   double phi;               /* rotation about y axis*/
   double kappa;             /* rotation about z axis */
   double fovx, fovy;        /* Fields of view */
   double dcpd;                    /* Digital Count per Degree */
} view_type;

view_type view;

/* When a ray hits something, we record vital information */

typedef struct {
   long object;
   long facet;
   double dist;
   double angle;
} hit_type;
```

```
typedef struct {
   char name[16];
   double angle[10], emissivity[10];
   long specular;
} emis_type;

emis_type materials[MAX_MATERIALS];

typedef struct {
   double temp, radiance;
} planck_type;

planck_type t2ltbl[200];

typedef struct {
   double sonde;
   double temp;
   double altit;
   double tau;
   double Lu;
} atm_type;

atm_type atmtbl[51];

typedef struct {
   double angle, dwrad;
} dwr_type;

struct dscr {
   short length;
   char class, type;
   char *buffer;
};

dwr_type dwr[20];     /* Downwelled Radiance Array at ground */
double DwRad[90];     /* Downwelled Radiance Array at altitude Z */

double corfactors[3]; /* Transmission Correction Factors */

/* dirtran_format class[MAX_MATERIALS]; */

long   DEBUG, numobjects, nummaterials;

double  downwelled_radiance(double), ang_emissivity(emis_type, double);
double path(double, double), altrans( double ), Calc_DwRad( double, double );
double calc_dwra(ray_type),  R2K( double ), K2R( double ), alTemp( double );

void read_emis(), read_rtt(), read_cor(), read_atm(), read_dwr();

void print_ray(ray_type), print_box(box_type), print_hit(hit_type);
void print_plane(plane_type), print_facet(facet_type);
void get_data(object_type[], FILE[]);

double Limage[512][512];

static char *PATH = "user29:[ehs0630.dirtran]";
char viewfile[20], scenefile[20], mapimage[20], outimage[20], lutfile[20];
double CritAng;



/* Trace --
 *    This is the main program used to generate the synthetic
 *  image.  It calls routines that read in the data files.
 *  It then calculates primaray rays and sends them to the tracer.
 *  Upon return it calculates the appropiate radiance for the pixel.
 *  The Last routine will convert the radiance to a temperature values
 *  and create an image in Digital Counts.

 *  DEBUG Levels:
 *      1) Show object data on read in
 *      2) Shows data file data
 *      3) Shows primary projection rays (NOT Normalized)
 *      4) Hit data (primary and secondary)
 *      5) Bounding boxes
 *      6) Plane and facet data
 *      7) Event cases
 *      8) Radiance calculations
 *      9) Reflection Rays
 *      10) Transmission values
 *      11) Dirtran: Path subroutine
 *      12) Dirtran: Compute Upwelled Radiance
 *      13) DIRTRAN: Calc_DwRad when ray misses Earth
 */

#include "defs2.h"

/* These functions are used for image creation on the Deanza a la S. Schultz
*/
long ipi_setpicops();
long ipi_setpicdimension();
long ipi_openfile();
long ipi_diskpic();
long ipi_setpicdata();
long ipi_errorcheck();
long ipi_returnpicblocks();

main(argc, argv)
int argc;
char *argv[];
{
   long i, j;                        /* Loop Control Variables */
   ray_type ray, rray, calray;
   hit_type hit, hit2;
   double alpha, beta;              /* angular pixel incrementals */
   double minval, maxval;           /* radiance values */
   double px, py, fl, aspect;       /* artifical focal length */
   long event, emis_index, Limit;
   double a, b, c;                  /* Extrapolation Coefficients */
   double m[5][5];                  /* Rotation Matrix */
   double o, p, k;                  /* omega, phi, and kappa : axis rotations
*/
   double L, Lb, T2Sens, UpwR;  /* Radiance, Background Radiance, transmission
                              to sensor, and upwelled Radiance */
   double dwra;                     /* downwelled radiance angle */

   /* Functions */
   double bob( point_type, ray_type ), calc_crit_angle( double ), CurAng;
   double Interpolate_DwRad( double, short, double, double, double );

   /* variables for the ipi routines */
   unsigned long status, picops, fileblk, fileptr, bandmask;
   unsigned long sizeinblocks, check;
   unsigned short DC;
   unsigned long X, Y, interp_flag;

   /* Descriptors */
   struct dscr infile;

   /* Initialize */
   DEBUG = 0; minval = 1000.0; maxval = -1000.0; interp_flag = FALSE;

   if ( argc == 1 ) {
      printf("Usage: trace execfile [DEBUG]\n");
      exit(0);
   }

   if (argc == 3)
      DEBUG = atoi(argv[2]);

   read_exec_file(argv[1]);
   read_viewing_data(viewfile, &fl, &aspect);
   CritAng = calc_crit_angle( view.viewpoint.z );
   if (DEBUG != 0)
      printf("fl: %lf aspect: %lf critical angle: %lf\n", fl, aspect,
CritAng);
   read_scene_data(scenefile);
   printf("\n\nFinished reading data files\n");

   loaddscr( &infile, mapimage );
   load_DwRad( view.viewpoint.z , &Limit, &a, &b, &c );
   if (DEBUG != 0)
      printf("Limit: %d a: %lf b: %lf c: %lf\n", Limit, a, b, c);

   /* Initialize DeAnza Stuff */
   picops = fileblk = 0;
```

```
check = 6;

status = ipi_setpicops ( &picops );
ipi_errorcheck( &status, 0, 0, 0, &check );

status = ipi_setpicdimens ( &picops, &view.xsize, &view.ysize );
ipi_errorcheck( &status, 0, 0, 0, &check );

sizeinblocks = ipi_returnpicblocks( &picops );

status = ipi_openfile ( &fileblk, &infile, &sizeinblocks );
ipi_errorcheck( &status, 0, 0, 0, &check );

status = ipi_diskpic ( &fileblk, &fileptr, &picops );
ipi_errorcheck( &status, 0, 0, 0, &check );

bandmask = 1;

/* Load Matrix */

if (DEBUG == 1)
   printf("Scene contains %d objects\n", numobjects - 1);

calray.o.x = ray.o.x = view.viewpoint.x;
calray.o.y = ray.o.y = view.viewpoint.y;
calray.o.z = ray.o.z = view.viewpoint.z;

o = view.omega * M_PI / 180.0;   /* X axis */
p = view.phi * M_PI / 180.0;            /* Y axis */
k = view.kappa * M_PI / 180.0;   /* Z axis */

m[1][1] = cos(p)*cos(k);
m[1][2] = cos(p)*sin(k);
m[1][3] = -sin(p);
m[1][4] = 0.0;
m[2][1] = sin(o)*sin(p)*cos(k) - cos(o)*sin(k);
m[2][2] = sin(o)*sin(p)*sin(k) + cos(o)*cos(k);
m[2][3] = sin(o)*cos(p);
m[2][4] = 0.0;
m[3][1] = cos(o)*sin(p)*cos(k) + sin(o)*sin(k);
m[3][2] = cos(o)*sin(p)*sin(k) - sin(o)*cos(k);
m[3][3] = cos(o)*cos(p);
m[3][4] = 0.0;
m[4][1] = view.viewpoint.x;
m[4][2] = view.viewpoint.y;
m[4][3] = view.viewpoint.z;
m[4][4] = 1.0;

/* Print the matrix */

if (DEBUG == 3) {
   for(i=1; i<5; i++) {
      for(j=1; j<5; j++)
```

```
            printf("%9.6f ", m[i][j]);
          printf("\n");
        }
    }

    /* This is the main loop that goes through every pixel in the image */

    for(i = -view.xsize/2; i < view.xsize/2; i++) {
       printf("\r%d%% finished ...", (long)
         (100.0 * (double) (i + view.xsize/2) / (double) view.xsize) );
       for(j = view.ysize/2; j > -view.ysize/2; j--) {

         DC = 0; X = view.xsize/2 - i - 1; Y = j + view.ysize/2 - 1;
          status=ipi_setpicdata(fileptr, &picops, &bandmask, &DC, &X, &Y );
          ipi_errorcheck( &status, 0, 0, 0, &check );

         alpha = atan ( i / f1 );    /* increments for fov */
         beta = atan ( j*aspect / f1 );

         ray.d.x = tan(alpha) * (-f1);
         ray.d.y = tan(beta) * (-f1);
         ray.d.z = -f1;

         calray.d.x = (ray.d.x * m[1][1]) + (ray.d.y * m[2][1]) +
                      (ray.d.z * m[3][1]) + m[4][1];
         calray.d.y = (ray.d.x * m[1][2]) + (ray.d.y * m[2][2]) +
                      (ray.d.z * m[3][2]) + m[4][2];
         calray.d.z = (ray.d.x * m[1][3]) + (ray.d.y * m[2][3]) +
                      (ray.d.z * m[3][3]) + m[4][3];

         CurAng = bob( view.viewpoint, calray );

          if ( (DEBUG >= 3) && (DEBUG < 7) ) {
            printf("\ni: %d j: %d X: %d Y: %d\n", i, j, X, Y);
/*          printf("\ni: %d j: %d\n", i, j);        */
            printf("alpha: %lf beta: %lf\n", alpha*180.0/M_PI,
beta*180.0/M_PI);
            printf("Current Angle: %lf\n", CurAng );
            print_ray(calray); printf("\n");
          }

          normalize_direction(&calray);

          if ( trace_scene(&calray, &hit) ) {
            if (DEBUG == 4) {
               printf("\nprimary Hit!\n");
               print_ray(calray);
               print_hit(hit);
            }

            DC = scene[hit.object].facets[hit.facet].matindex;
            DC = DC<<4;       /* shift to upper 4 bits of byte */
            status=ipi_setpicdata(fileptr, &picops, &bandmask, &DC, &X, &Y );
            ipi_errorcheck( &status, 0, 0, 0, &check );

           if ( materials[ scene[hit.object].facets[hit.facet].matindex
].specular ) {

calc_reflect_ray(calray,&rray,scene[hit.object].facets[hit.facet].normal);

              if ( trace_scene(&rray, &hit2) ) {
                DC = DC | scene[hit2.object].facets[hit2.facet].matindex;
/*              DC = scene[hit.object].facets[hit.facet].matindex
                     | scene[hit2.object].facets[hit2.facet].matindex; */
                 status=ipi_setpicdata(fileptr, &picops, &bandmask, &DC, &X,
&Y );

                 ipi_errorcheck( &status, 0, 0, 0, &check );

                 if (DEBUG == 4) {
                    printf("\nSecondary Hit!\n");
                    print_ray(rray);
                    print_hit(hit2);
                 }
                 event = 4;
              }

              else {
                 event = 3;
/*                  if ( scene[hit.object].facets[hit.facet].matindex == 3 )
                  printf("\ni: %d j: %d X: %d Y: %d\n", i, j, X, Y); */
              }

           }
           else
              event = 2;
         }
         else
           event = 1;

         switch (event) {
         case 1: /* Hit Nothing */
          if ( CurAng  < CritAng ) {
             L = K2R( 286.0 );            /* Tbackground */
             if (DEBUG == 7) printf("case 1 -- Hit Earth\n");
          }
          else {
             event = 0;
             if (DEBUG == 7) printf("case 1 -- Missed Earth\n");
          }
          break;

         case 2: /* Hit diffuse Object */
          emis_index = scene[hit.object].facets[hit.facet].matindex;
          L = materials[emis_index].emissivity[1]
              * scene[hit.object].facets[hit.facet].radiance +
              ( 1.0 - materials[emis_index].emissivity[1]) * dwr[19].dwrad;
          if (DEBUG == 7) {
             printf("case 2\n");
             printf("%s\n",materials[emis_index].name);
             printf("%lf\n",materials[emis_index].emissivity[1]);
             printf("radiance:
%lf\n",scene[hit.object].facets[hit.facet].radiance);
             printf("Integrated Ld: %lf\n", dwr[19].dwrad);
             printf("L = %lf\n", L);
          }
          break;

        case 3: /* Hit specular object - No second object */
          emis_index = scene[hit.object].facets[hit.facet].matindex;
          L = ang_emissivity(materials [ emis_index ], hit.angle)
              * scene[hit.object].facets[hit.facet].radiance;
          /* calculate downwelled radiance angle */
          dwra = calc_dwra(rray);
          L += ( 1.0 - ang_emissivity(materials[emis_index], hit.angle) ) *
              downwelled_radiance(dwra * 180.0 / M_PI);
          if (DEBUG == 7) {
             printf("case 3\n");
             printf("%s\n",materials[emis_index].name);
             printf("%lf\n",ang_emissivity(materials [ emis_index ],
hit.angle));
             printf("ang: %lf dwra: %lf dwr:%lf\n", hit.angle * 180.0/M_PI,
                           dwra * 180.0/M_PI, downwelled_radiance (dwra) );
             printf("L = %lf\n", L);
          }
          break;

        case 4: /* Hit specular object - Hit second Object */
/*        emis_index = scene[hit2.object].facets[hit2.facet].matindex;
          Lb = ang_emissivity(materials [ emis_index ], hit2.angle)
              * scene[hit2.object].facets[hit2.facet].radiance; */

          Lb = scene[hit2.object].facets[hit2.facet].radiance;
          if (DEBUG == 7) {
             printf("case 4\n");
             printf("%s\n",materials[emis_index].name);
             printf("%lf\n",ang_emissivity(materials [ emis_index ],
hit2.angle));
             printf("Lb = %lf\n", Lb);
          }

          emis_index = scene[hit.object].facets[hit.facet].matindex;
          L = ang_emissivity(materials [ emis_index ], hit.angle)
              * scene[hit.object].facets[hit.facet].radiance +
              ((1 - ang_emissivity(materials [ emis_index ], hit.angle)) *
Lb);
          if (DEBUG == 7) {
             printf("case 4\n");
```

```
              printf("%s\n",materials[emis_index].name);
              printf("%lf\n",ang_emissivity(materials [ emis_index ],
hit.angle));
              printf("L = %lf\n", L);
           }
           break;
          } /* switch */



          /* Multiply by tau and add Lu * /
          if ( event ) {
            cur( &T2Sens, view.viewpoint.z, CurAng, &UpwR);
            if ( (DEBUG == 7) || (DEBUG == 8) )
               printf("Before tau, L = %lf\n", L);
             L *= T2Sens;
            if ( (DEBUG == 7) || (DEBUG == 8) ) {
               printf("tau = %lf\t", T2Sens );
               printf("After tau, L = %lf\n", L);
            }
             L += UpwR;
            if ( (DEBUG == 7) || (DEBUG == 8) ){
               printf("Lu = %lf\t", UpwR);
               printf("After Lu, L = %lf\n", L);
            }
          }
          else {
            L = Interpolate_DwRad(CurAng, Limit, a, b, c);
            if ( (DEBUG == 7) || (DEBUG == 8) || (DEBUG == 13) )
               printf("in main: Current Angle: %lf L = %lf\n", CurAng, L);
          }

         Limage[X][Y] = L; minval = min(L, minval); maxval = max(L, maxval);

         if ( (i == 0) && (j == 0) )
            printf("\nCentral value = %lf\tTemperature %lf\n\n", L, R2K( L ) );

         if (L > 1.0) {
           printf ("\ni: %d j: %d alpha: %lf beta: %lf\n", i, j, alpha, beta);
           printf("L = %lf UpwR = %lf\n", L, UpwR);
           printf("Angle = %lf\n", view.omega + fabs(alpha*180.0/M_PI) );
           printf("event = %d\n", event);
         }

       } /* x loop */
    } /* y loop */
    save_image(minval, maxval);
 } /* main */

 /* This goes through all the objects and returns a hit or miss */

 int trace_scene(ray, hit)
 ray_type *ray;
```

```
hit_type *hit;
{
   int k;
   int facet;          /* Facet Hit */
   int hit_flag;       /* TRUE if any object was hit */
   double distance;    /* Distance that current object was hit at */
   double angle;       /* Angle that current object was hit at */
   point_type intpnt;  /* The point where the object was hit */
   ray_type tempray;

   /* Initialize */

   hit_flag = FALSE;
   (*hit).object = 0;
   (*hit).facet = 0;
   (*hit).dist = 10000.0;

   tempray = *ray;

   for (k=1; k<numobjects; k++) {
      if (DEBUG == 4) printf("====== Object %d ========\n", k);
      if ( hit_object(tempray, scene[k], &facet, &distance, &angle, &intpnt) )
{
         hit_flag = TRUE;
         if (DEBUG == 4)
            printf("Hit object# %d, facet# %d, dist = %lf\n", k, facet,
distance);
         if ( distance < (*hit).dist ) {
            (*hit).object = k;
            (*hit).facet = facet;
            (*hit).dist = distance;
            (*hit).angle = angle;
            (*ray).d.x = intpnt.x;
            (*ray).d.y = intpnt.y;
            (*ray).d.z = intpnt.z;
         }
      }
   }

/*    if (hit_flag)
       printf("Final Hit: object #%d, facet#%d, dist = %lf\n",
         (*hit).object, (*hit).facet, (*hit).dist); */

   return(hit_flag);
}

save_image(minval, maxval)
double minval, maxval;
{
   unsigned long x, y;
   double avl, avt, dL, Incr, dcpL, slope;
   short offset, DC;
```

```
FILE *fp;
double hit[256];

unsigned long status, picops, fileblk, fileptr, bandmask;
unsigned long sizeinblocks, check;
struct dscr infile;

loaddscr( &infile, outimage );
picops = fileblk = 0;
check = 6;

status = ipi_setpicops ( &picops );
ipi_errorcheck( &status, 0, 0, 0, &check );

status = ipi_setpicdimens ( &picops, &view.xsize, &view.ysize );
ipi_errorcheck( &status, 0, 0, 0, &check );

sizeinblocks = ipi_returnpicblocks( &picops );

status = ipi_openfile ( &fileblk, &infile, &sizeinblocks );
ipi_errorcheck( &status, 0, 0, 0, &check );

status = ipi_diskpic ( &fileblk, &fileptr, &picops );
ipi_errorcheck( &status, 0, 0, 0, &check );

bandmask = 1;

if ( (fp = fopen(lutfile, "w")) == NULL ) {
   printf("save_image: can't open lut\n");
   exit(1);
}

avl = ((maxval - minval) / 2) + minval; /* middle radiance */
avt = R2K( avl );                       /* average temperature */
dL = K2R( avt + 1 ) - avl;                     /* delta in radiance */
slope = view.dcpd / dL;                 /* m and b */
offset = 128 - (short) (slope * avl );
printf("\nSlope %lf offset %d\n", slope, offset);

for (x=0; x<256; x++)
   hit[x] = -1;

for (x=0; x<view.xsize; x++) {
   for(y=0; y<view.ysize; y++) {
     DC = (short) (slope * Limage[x][y]) + offset;
     if (DC > 255) DC = 255;
      status=ipi_setpicdata(fileptr, &picops, &bandmask, &DC, &x, &y );
      ipi_errorcheck( &status, 0, 0, 0, &check );
     /* fprintf(fp, "%lf %d\n", Limage[x][y], DC ); */
     if ( hit[DC] == -1 )
        hit[DC] = Limage[x][y];
   }
}
```

```
   for (DC=0; DC<256; DC++) {
      if (hit[DC] != -1)
         fprintf(fp, "%d %lf %lf %lf\n", DC, hit[DC], R2K( hit[DC] ),
                      R2K( hit[DC] ) - 273.15 );
   }
   close(fp);
}

/* Calculate the angle from the altitude's normal to the current ray
 * Why was it called bob, because we ran out of function names!
 */

double bob ( A, B )
point_type A;
ray_type B;
{
   double alpha;
   ray_type dummy;

   dummy.o.x = A.x; dummy.o.y = A.y; dummy.o.z = A.z;
   dummy.d.x = A.x; dummy.d.y = A.y; dummy.d.z = 0.0;
   normalize_direction( &dummy );
   normalize_direction( &B );

   alpha = acos( (dummy.d.x * B.d.x) + (dummy.d.y * B.d.y) + (dummy.d.z *
B.d.z) );

   return( 180.0/M_PI*alpha );
}



#include "defs2.h"

int hit_object(ray, object, facet_num, distance, angle, intpnt)
ray_type ray;
object_type object;
int *facet_num;
double *distance;
double *angle;
point_type *intpnt;
{
   int i, hit_flag, current_facet;
   plane_type plane;
   int reverse;
   double dist;        /* Distance to Plane */
   double min_dist;
   double ang;
   point_type intercept;
   hit_flag = current_facet = 0; min_dist = 10000.0;

   if ( hit_box(ray, object.box) ) {

      if ( (DEBUG == 5) || (DEBUG == 6) ){
        printf("Bounding Box Was Hit!\n");
      }
      for(i=0; i<object.numfacets; i++) {

        if (DEBUG == 6)
           printf("\tFacet #%d\n", i);
         calc_plane_equ(&plane, object.facets[i].points);
        /*  print_plane(plane); */
         if ( hit_plane(ray, plane, &intercept, &reverse, &dist, &ang) ) {

            if (DEBUG == 6) {
              printf("\t\tPlane Was Hit! distance = %lf angle = %lf\n", dist,
ang);
              printf("\t\t\tintpnt = %lf %lf %lf\n", intercept.x,
intercept.y,intercept.z);
              printf("\t\treverse = %d\n", reverse);
            }

           if (dist < min_dist) {
               if ( hit_tri(plane, object.facets[i], intercept) ) {
                 if (DEBUG == 6) printf("\t\t\tfacet Was Hit!\n");
                 min_dist = dist; hit_flag = 1; current_facet = i;
                  *angle = ang;
                 (*intpnt).x = intercept.x;
                 (*intpnt).y = intercept.y;
                 (*intpnt).z = intercept.z;
               }
           }
        }
      }
   }
   if (hit_flag) {
      *facet_num = current_facet;
      *distance = min_dist;
   }
   return(hit_flag);
}


calc_plane_equ(plane, tri)
plane_type *plane;
point_type tri[3];
{
   point_type u,v,norm;
   double length;

   u.x = tri[0].x - tri[2].x;
   u.y = tri[0].y - tri[2].y;
   u.z = tri[0].z - tri[2].z;
   v.x = tri[1].x - tri[2].x;
   v.y = tri[1].y - tri[2].y;
   v.z = tri[1].z - tri[2].z;
   cross(norm, u, v);
   length = sqrt ( sqr(norm.x) + sqr( norm.y) + sqr(norm.z) );
   (*plane).a = norm.x / length;
   (*plane).b = norm.y / length;
   (*plane).c = norm.z / length;
   (*plane).d = (*plane).a * -tri[0].x + (*plane).b * -tri[0].y +
                (*plane).c * -tri[0].z;
}

/* Essential Ray Tracing Algorithms -- Eric Haines
 * RAY/BOX INTERSECTION p. 29
 */

int hit_box(ray, box)
ray_type ray;
box_type box;
{
   double Tnear, Tfar, T1, T2;

   T1 = T2 = 0.0;

   if (DEBUG == 5) {
      printf("\nhit_box:\n");
      print_ray(ray);
      print_box(box);
   }

   Tnear = -10000.0; Tfar = 10000.0;

   if ( is_zero(ray.d.x) ) {
      if ((ray.o.x < box.pmin.x) || (ray.o.x > box.pmax.x))
         return(0);
   }
   else {
      T1 = ( box.pmin.x - ray.o.x ) / ray.d.x;
      T2 = ( box.pmax.x - ray.o.x ) / ray.d.x;
      if (T1 > T2)
         swap(&T1, &T2);
      if (T1 > Tnear)
         Tnear = T1;
      if (T2 < Tfar)
         Tfar = T2;
      if (Tnear > Tfar)       /* box is missed */
        return(0);
      if (Tfar < 0)    /* box is behind ray */
        return(0);
   }
   if ( is_zero(ray.d.y) ) {
      if ((ray.o.y < box.pmin.y) || (ray.o.y > box.pmax.y))
         return(0);
   }
   else {
      T1 = ( box.pmin.y - ray.o.y ) / ray.d.y;
      T2 = ( box.pmax.y - ray.o.y ) / ray.d.y;
      if (T1 > T2)
         swap(&T1, &T2);
      if (T1 > Tnear)
         Tnear = T1;
      if (T2 < Tfar)
         Tfar = T2;
      if (Tnear > Tfar)      /* box is missed */
        return(0);
      if (Tfar < 0)   /* box is behind ray */
        return(0);
   }

   if ( is_zero(ray.d.z) ) {
      if ((ray.o.z < box.pmin.z) || (ray.o.z > box.pmax.z))
         return(0);
   }
   else {
      T1 = ( box.pmin.z - ray.o.z ) / ray.d.z;
      T2 = ( box.pmax.z - ray.o.z ) / ray.d.z;
      if (T1 > T2)
         swap(&T1, &T2);
      if (T1 > Tnear)
         Tnear = T1;
      if (T2 < Tfar)
         Tfar = T2;
      if (Tnear > Tfar)      /* box is missed */
        return(0);
      if (Tfar < 0)          /* box is behind ray */
        return(0);
   }
   return(1);
}

swap(x,y)
double *x, *y;
{
   double temp;
   temp = *x;
   *x = *y;
   *y = temp;
}

/* Essential Ray Tracing Algorithms -- Eric Haines
 * Ray/Plane Intersection  p. 16
 */
```

```
int hit_plane(ray, plane, intpnt, reverse, dist, angle)
ray_type ray;
plane_type plane;
point_type *intpnt;
int *reverse;
double *dist;
double *angle;
{
   double Vprd, Vpro, T;
   point_type nray;

   *reverse = FALSE;

   Vprd = plane.a * ray.d.x + plane.b * ray.d.y + plane.c * ray.d.z;
   if (DEBUG == 6) {
      printf("\thit_plane: Vprd = %lf\n", Vprd);
   }
   if (Vprd == 0.0)
      return(0);
   else {
      Vpro = - (plane.a * ray.o.x + plane.b * ray.o.y + plane.c * ray.o.z
               + plane.d);
      T = Vpro / Vprd;
      *dist = T * sqrt( sqr(ray.d.x) + sqr(ray.d.y) + sqr(ray.d.z) );
      *angle = M_PI - acos( Vprd );
/*
 * These numbers were changed because the inputs to this ray-tracer
 * is in km (so that DIRTRAN altitudes would work right.  If data is
 * ever changed back to meters or something, these should be adjusted.
 */
      if ( (*dist > -0.0000001) && (*dist < 0.0000001) )
        return(0);
      if (T < 0.0)
        return(0);
      else if (T > 0.0) {
        (*intpnt).x = ray.o.x + ray.d.x * T;
        (*intpnt).y = ray.o.y + ray.d.y * T;
        (*intpnt).z = ray.o.z + ray.d.z * T;
      }
      if (Vprd > 0.0) {
        *reverse = TRUE; return(0);
      }
      return(1);
   }
}

void normalize_direction(ray)
ray_type *ray;
{
   double length;

   length = sqrt ( sqr( (*ray).d.x - (*ray).o.x ) +
                  sqr( (*ray).d.y - (*ray).o.y ) +
                  sqr( (*ray).d.z - (*ray).o.z ) );

   if (length != 0.0) {
      (*ray).d.x = ( (*ray).d.x - (*ray).o.x ) / length;
      (*ray).d.y = ( (*ray).d.y - (*ray).o.y ) / length;
      (*ray).d.z = ( (*ray).d.z - (*ray).o.z ) / length;
   }
   else {
      (*ray).d.x = 0.0;
      (*ray).d.y = 0.0;
      (*ray).d.z = 0.0;
   }

/*   if (DEBUG == 2) {
      printf("leaving normalize_direction:\n");
      print_ray(*ray);
   } */

}

/* Essential Ray Tracing Algorithms -- Eric Haines p.19
 * Point/Polygon Inside/Outside Testing
 */

int hit_tri(plane, tri, intpnt)
plane_type plane;
facet_type tri;
point_type intpnt;
{
   double u[3], v[3];
   int nc;     /* Number of Crossings */
   int sh;     /* Sign Holder */
   int nsh;    /* Next Sign Holder */
   double tip; /* True Intersection Point */
   int NV;     /* Number Verticies */
   int a, b, i;

   NV = 3;

/*   printf("In hit_tri\n");
   print_facet(tri);
   print_plane(plane); */

   if ( ( fabs(plane.a) > fabs(plane.b) ) && ( fabs(plane.a) > fabs(plane.c) )
) {
      /* X is the dominant coordinate */
      u[0] = tri.points[0].y - intpnt.y;
      u[1] = tri.points[1].y - intpnt.y;
      u[2] = tri.points[2].y - intpnt.y;
      v[0] = tri.points[0].z - intpnt.z;
      v[1] = tri.points[1].z - intpnt.z;
      v[2] = tri.points[2].z - intpnt.z;
   }
   else if ((fabs(plane.b) > fabs(plane.a)) && (fabs(plane.b) > fabs(plane.c)))
{
      /* Y is the dominant coordinate */
      u[0] = tri.points[0].x - intpnt.x;
      u[1] = tri.points[1].x - intpnt.x;
      u[2] = tri.points[2].x - intpnt.x;
      v[0] = tri.points[0].z - intpnt.z;
      v[1] = tri.points[1].z - intpnt.z;
      v[2] = tri.points[2].z - intpnt.z;
   }
   else {
      /* Z is the dominant coordinate */
      u[0] = tri.points[0].x - intpnt.x;
      u[1] = tri.points[1].x - intpnt.x;
      u[2] = tri.points[2].x - intpnt.x;
      v[0] = tri.points[0].y - intpnt.y;
      v[1] = tri.points[1].y - intpnt.y;
      v[2] = tri.points[2].y - intpnt.y;
   }

/*   for (i=0; i<3; i++)
      printf("%lf %lf\n", u[i], v[i]); */

   nc = 0;
   sh = sign(v[0]);

/* NOTE: Remember that a -1 is TRUE in 'C' -- it's non-zero */

   for (a=0; a <= NV-1; a++) {
      b = (int) fmod( (double) a + 1, (double) NV);
      nsh = sign(v[b]);
      if (sh != nsh) {
         if ( (sign(u[a]) == 1) && (sign(u[b]) == 1) )
            nc++;
         else if ( ( sign(u[a]) == 1 ) || ( sign(u[b]) == 1 ) ) {
                tip = u[a] - v[a] * ( u[b] - u[a] ) / ( v[b] - v[a] );
            if ( sign(tip) == 1 )
               nc++;
           }
      }
      sh = nsh;
   }
   /* This should be an even/odd test, but were working with tri's */
   if (nc == 1)
      return(1);
   else
      return(0);
 }
```