# AN INVESTIGATION INTO THE USE OF OPENGL AS A LIBRARY FOR THE SIMULATION OF INFRARED SCENARIOS

by

**Francois Petrus Jacobus le Roux**

Submitted in partial fulfillment of the requirements for the degree

Master of Engineering (Electronic)

in the

Faculty of Engineering

UNIVERSITY OF PRETORIA

September 2001

In the beginning God created the heaven and the earth.

And the earth was without form, and void; and darkness was upon the face of the deep. And the Spirit of God moved upon the face of the waters.

And God said, Let there be light: and there was light.

And God saw the light, that it was good: and God divided the light from the darkness.

And God called the light Day, and the darkness he called Night. And the evening and the morning were the first day.

*Genesis 1:1-5*

Now to him who is able to do immeasurably more than all we ask or imagine, according to his power that is at work within us,

to him be glory in the church and in Christ Jesus throughout all generations, for ever and ever! Amen.

*Ephesians 3:20-21*

# Acknowledgements

A number of people contributed to the dissertation and its contents:

- Dr. J.H.S. Roodt read through an early version of the document and supplied valuable inputs and suggestions on the further development of the study.
- Dr. J.J. Krüger read the document and supplied extensive inputs regarding the lay-out, presentation and contents of the dissertation. He also spent a lot of time discussing detail aspects of the work with me. It is a privelege to work with him.
- Prof. F.W. Leuschner supported the initial idea to carry out this work. He read through a number of versions of this dissertation and the structure and contents reflect his inputs.
- Mr. C.J. Willers's long experience in the field of radiometry shed additional light on a number of technical issues.
- My colleagues at Defencetek, CSIR supplied both a project environment to carry out the work and a sounding board to discuss new ideas and problems. The application of the results in future projects served as a constant motivator. Defencetek supplied the financial support to acquire computer hardware, infrastructure and reference material, as well as tuition fees.

The work presented in this dissertation was made possible by the establishment of SERA, the Southern Educational and Research Alliance, between the University of Pretoria and the CSIR.

## Summary

The generation of real-time infrared imagery was previously the domain of multi-million rand graphics supercomputers such as the SGI Onyx2. The purpose of this study is to investigate the implementation of radiometrically accurate infrared simulations on a personal computer, by adapting the widely available graphics library, OpenGL®.

An introduction is given into the radiometric and computer graphics principles related to this study. A technique is investigated to increase the dynamic range that is available in commercial graphics accelerators. It is shown that it is possible to map spectrally variant radiometric quantities to equivalent single-parameter variables that can be used in infrared simulations, bearing in mind certain constraints. The implementation of an infrared simulation shows that the current generation PC hardware can render images at real-time frame rates, but that extra processing is required to generate radiometrically accurate images at the required frame rates.

**Keywords:** OpenGL, infrared simulations, real-time scene generation, radiometric parameters, atmospheric transmittance, path radiance.

## Opsomming

Die intydse berekening van infrarooi beelde het voorheen 'n grafika superrekenaar soos die SGI Onyx2 benodig. Die doelwit van hierdie studie is om die implementering van radiometries akkurate infrarooi simulasies op 'n persoonlike rekenaar te ondersoek. OpenGL®, 'n algemeen beskikbare grafika biblioteek, word aangepas om hierdie simulasies te kan doen.

'n Oorsig word gegee van die radiometrie en rekenaargrafika wat betrekking het op hierdie studie. 'n Tegniek word ondersoek om die dinamiese bereik van kommersiële skermkaarte te verhoog. Daar word aangetoon dat dit moontlik is om spektrale radiometriese veranderlikes met enkel-parameter veranderlikes voor te stel indien daar aan sekere beperkings voldoen word. 'n Implementering van 'n simulasie toon aan dat huidige generasie persoonlike rekenaars wel die beelde teen intydse raamtempos kan genereer, maar dat ekstra prosessering nodig is om radiometries akkurate beelde te genereer.

**Sleutelwoorde:** OpenGL, infrarooi simulasies, intydse beeld generasie, radiometriese parameters, atmosferiese transmittansie, padradiansie.

# Contents

# List of Abbreviations and Symbols

| | |
|---|---|
| API | Application programming interface |
| ARB | Architectural review board |
| AUX | OpenGL *aux*iliary library |
| COTS | Commercial-off-the-shelf |
| CPU | Central processing unit |
| CRT | Cathode ray tube |
| DMA | Direct memory access |
| FLIR | Forward looking infrared - a thermal imager |
| GUI | Graphical user interface |
| OpenGL | Open graphics library. It is a trademark of SGI |
| PC | Personal computer - typically using an Intel processor |
| SGI | Trade name of the company previously trading as Silicon Graphics Inc. |

| | |
|---|---|
| $\alpha$ | Absorptivity. It is also used for the atmospheric transmittance coefficient |
| $c$ | Speed of light in $ms^{-1}$ |
| $E$ | Irradiance in $Wm^{-2}$ |
| $h$ | Planck's constant in $Js$ |
| $I$ | Intensity in $W sr^{-1}$ |
| $k$ | Boltzmann constant in $JK^{-1}$ |
| $\lambda$ | Wavelength |
| $L$ | Radiance in $Wm^{-2}sr^{-1}$ |
| $M$ | Exitance in $Wm^{-2}$ |
| $\Omega$ | Solid Angle |
| $\Phi$ | Radiant flux in $W$ |
| $\rho$ | Reflectivity |
| $\tau$ | Transmissivity |
| $T$ | Temperature |

# Chapter 1

# Introduction

## 1.1 Background

Physically realistic imagery is an important part of the design, simulation and evaluation of electro-optical systems. These images can be acquired during field trials or generated using the characteristics of the scenario and objects of interest. The generation of real-time (more than 75 frames per second) images used to be limited to graphics supercomputers such as the SGI Onyx2. These computers utilize dedicated hardware and multiple processors to generate images at the required rate. The development of a new generation of personal computer graphics cards during 1999/2000 made it possible to generate complex images at real-time frame rates, using just a personal computer and a commercial off-the-shelf graphics card. These graphics cards are often called graphics co-processors to indicate that a large part of the graphics processing is handled by the graphics card, instead of the CPU.

The development of the graphics cards are largely driven by the demands of computer game players for realistic imagery to accompany games. The cards are therefore developed to deliver images that are pleasing to the human eye. The generation of two-dimensional representations, as viewed on a computer monitor, of three-dimensional scenarios and objects place large demands on the computer programmer. The programme must transform the three-dimensional objects to two-dimensional representations, add lighting and atmospheric effects and compute surface parameters, all at real-time frame rates. Graphics libraries, such as OpenGL® , were developed to assist with this task.

OpenGL was originally developed to generate visual images, but it is possible to generate infrared images using OpenGL. The infrared parameters such as atmospheric transmittance and source radiance must be mapped to equivalent OpenGL parameters such as fog density and object colour.

Infrared imagery are used for the development of systems and automatic target recognition algorithms and the evaluation of systems. These evaluations include hardware-in-the-loop testing and the evaluation of operational systems prior to their use. Some examples are given in [1], [2] and [3].

## 1.2 Problem Statement

The generation of high fidelity infrared images from a three-dimensional model of an object would require the following steps:

- Determine the objects that are in the sensor's field of view. These objects are normally described by a collection of three-dimensional polygons. The number of polygons vary with the complexity of the object, but are typically not less than a few thousand polygons per object.
- Calculate the projected area of each surface element or polygon of the object as viewed from the sensor.
- Calculate the spectral radiance ($L$) of each surface element. The radiance could include reflected, emitted and transmitted energy. The reflected radiance can be from a source such as the sun or another source such as a lamp. The irradiance ($E$) is influenced by the atmospheric path between the source and the surface element. The radiance is further influenced by the reflectivity, absorptivity and transmissivity of the surface element. These parameters can all vary as a function of wavelength. This would imply that the radiance calculation must be carried out at each wavelength of interest.
- Calculate the spectral atmospheric transmittance between the surface element and the sensor using a model such as MODTRAN [4].
- Calculate the spectral energy emitted by the atmospheric path between the surface element and the sensor.
- Calculate the spectral irradiance as the sum of the irradiance from the object multiplied by the atmospheric transmittance and the irradiance due to path radiance.
- Multiply the spectral irradiance with the sensor system's spectral response and integrate the spectral irradiance to determine the irradiance on the sensor from the specific surface element.
- Repeat the process for all the surface elements in the sensor's field of view. The surface elements could be from terrain, target objects and sky background.

The calculation of each of the items mentioned above require a massive amount of computer power. The computer systems that can carry out such a task operate with multiple processors and costs multiple millions of dollars. Silicon Graphics developed a graphics library, OpenGL, to simplify the calculation of images in the visual band. The library is highly optimised to enable the calculation of images in real-time. The implementation of real-time calculations required that a few short-cuts had to be implemented. One of these was to limit the dynamic range of the final image. This did not have an effect on the results, when viewed by a human observer. Recent developments in graphics technology led to graphics co-processors that could implement graphics libraries such as OpenGL and Microsoft's DirectX in hardware. These graphics co-processors became available on consumer hardware at a fraction of the cost of a graphics super-computer. A shortcut in the development of these systems was to limit the dynamic range of the output images even further, to eight bits per colour for each of the three basic colours.

## 1.3 Methodology

The objective of this study is to investigate the implementation of a real-time infrared image generation system on a Pentium III class computer. Generating infrared imagery on a system that was developed to calculate visual imagery presented three problems:

- The graphics library was developed for visual images. It therefore requires the extension of the library to the infrared band. This can be accomplished by defining a mapping of spectral infrared parameters such as radiance and atmospheric transmittance to parameters that can be implemented in OpenGL or DirectX.
- The dynamic range of graphics accelerators is not sufficient to calculate high dynamic range infrared images. A technique is therefore required to "extend" the dynamic range of the graphics accelerator.
- The implementation of solutions to the first two problems might be so time consuming that the frame rate of the resulting images is too low to be useful. The solution to this problem will require the development of some additional techniques to reduce the number of required calculations.

## 1.4 Research Areas

To reach the objective the following areas were investigated:

- **Radiometry**
  The mapping of spectrally variant parameters such as atmospheric transmittance, surface emissivity and radiance to single value parameters that can be implemented as an equivalent model and used in an OpenGL environment. The results are compared with theoretically calculated data to determine the fidelity of the technique.
- **Rendering physically realistic imagery on low-cost computer platforms**
  The references found in the literature search all used high-end SGI machines with built-in hardware acceleration to generate images at real-time frame rates. Elements of a radiometric simulation were implemented on a personal computer with a graphics co-processor. Images were rendered using this system and the results are compared with theoretically calculated values. The study was limited to thermal self-emission (radiance) to reduce the number of variables in a specific scenario.
- **Increasing the dynamic range of commercial graphics accelerators**
  According to Makar *et al.* [5], the dynamic range of high-end image rendering systems, such as the SGI InfiniteReality2 is 12 bits per colour. The equivalent dynamic range on a personal computer is 8 bits per colour. The dynamic range limits the range of radiance levels that can be presented in a simulation. A technique to extend the dynamic range, proposed by Olsen *et al.* [6], was investigated.

## 1.5 Literature Survey

A literature search using the DIALOG system yielded four references related to the proposed research area.

Lorenzo *et al.* [7] describe a system that was developed to generate synthetic imagery for sensors such as FLIRs. The system implements OpenGL to generate the imagery, but runs on SGI hardware. The authors do not describe the accuracy of the method nor the correspondence to actual scenarios. Anding *et al.* [8] developed a product called SensorVision that uses OpenGL to generate infrared imagery. The authors do not describe the accuracy of the method nor the correspondence to actual scenarios. The simulation runs on SGI hardware.

Sundberg *et al.* [9] and Crow *et al.* [10] are using OpenGL to generate infrared images based on parametric databases.

## 1.6 Contribution

The literature survey shows that current infrared image generators runs on multi-million dollar graphics computers. There is a limited description of the techniques required extend OpenGL to generate infrared images. The contributions flowing from this study will be:

- It will be shown that it is possible to generate infrared imagery on a consumer personal computer.
- OpenGL will be extended to render infrared images on a personal computer.
- The accuracy of the mapping of infrared parameters to the visual band will be investigated and problem areas defined.
- It will be shown that a technique to extend the dynamic range of SGI machines is also applicable to personal computers.
- A technique will be developed that would enable the rendering of high-fidelity infrared images on a personal computer.

## 1.7 Dissertation Outline

The following are addressed in this document:

- **Chapter 2** is an overview of the radiometric principles that are relevant to this study.
- **Chapter 3** is an overview of the elements of computer graphics that are relevant to this study.
- **Chapter 4** describes an investigation of the technique proposed by Olsen *et al.* [6] to artificially increase the dynamic range of a graphics accelerator card. The difference between the rendered and calculated radiance of a test object is determined for different

positions on the object. The difference between the rendered intensity and the calculated intensity of the object is determined for different separation distances between the test object and the sensor.

- **Chapter 5** is a description of the determination of equivalent single parameter entities for atmospheric transmittance and path radiance.
- **Chapter 6** describes the practical implementation of a radiometric simulation in OpenGL. Problems with the implementation are highlighted and techniques are suggested to implement an accurate simulation.

# Chapter 2

# Radiometric Overview

The purpose of this chapter is to highlight the radiometric concepts that are relevant to this study. It is not intended as a complete introduction to radiometry and it is therefore assumed that the reader is familiar with the basic radiometric quantities.

## 2.1 Object Parameters

A blackbody (Planckian) function describes the maximum energy that can be emitted by an object in equilibrium at a specific temperature and wavelength. According to Zissis, *et al.* [11, p8], the spectral radiant excitance can be calculated using

$$M_{\lambda,T} = \frac{c_1}{\lambda^5(e^x - 1)}, \tag{2.1}$$

where

$$x = \frac{c_2}{\lambda T}. \tag{2.2}$$

In equations (2.1) and (2.2) $c_1$ is the first radiation constant, $c_2$ is the second radiation constant and $\lambda$ is the wavelength in $\mu m$. The values of $c_1$ and $c_2$ are given by Leuschner [12] as:

$$\begin{aligned} c_1 &= 2\pi c^2 h = 3.7417749 \times 10^8 \quad [W \mu m^4 m^{-2}], \\ c_2 &= \frac{hc}{k} = 14387.69 \quad [\mu m K]. \end{aligned}$$

A radiator under equilibrium conditions is limited by the blackbody relationship. Emissivity describes the efficiency with which it radiates energy, absorptivity the efficiency with which it absorbs energy, reflectivity the efficiency with which it reflects energy and transmissivity the efficiency with which it transmits energy.

Kirchhoff [11, p26] pointed out that under equilibrium the sum of the absorbed, reflected and transmitted power is equal to the incident power, so that one can write

$$\alpha + \rho + \tau = 1, \tag{2.3}$$

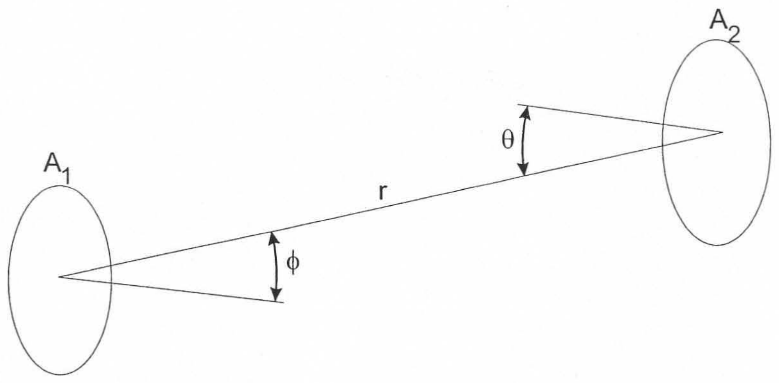

Figure 2.1: Flux transfer between a source and a detector

with $\alpha$ the absorptivity, $\rho$ the reflectivity and $\tau$ the transmissivity.

Equation (2.3) is true for total power and under equilibrium conditions it is true for spectral quantities.

## 2.2 Flux transfer

The flux that is transferred from Surface $A_1$ to Surface $A_2$ in Figure 2.1 is given by Wyatt [13, p54] as:

$$\partial^2\Phi_\lambda = \frac{L_{1,\lambda}\partial A_1 \cos\phi \partial A_2 \cos\theta \tau_\lambda}{r^2}, \tag{2.4}$$

with $L_1$ the radiance from surface $A_1$, $\tau$ the transmittance of the medium between $A_1$ and $A_2$ and $\phi$ and $\theta$ the angle between the surface's normal and the vector connecting the two surfaces. The subscript $\lambda$ is included to indicate spectrally variant sources and transmittance paths.

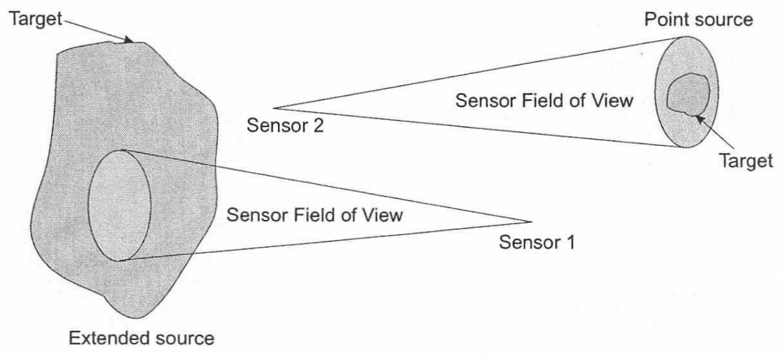

Figure 2.2: Comparison between a point and extended source

### 2.2.1 Irradiance on a pixel from an extended source

If the size of an object in a sensor's field of view is larger than the sensor's field of view, as shown in Figure 2.2, the object is called an extended source. In the case of an extended source, Equation (2.4) can be simplified to give

$$E_{source,\lambda} = L_{source,\lambda}\Omega_{sensor}\tau_{\lambda,atm}\tau_{\lambda,sys}. \tag{2.5}$$

The spectral irradiance on the sensor from the source is given by $E_\lambda$, $L_{source,\lambda}$ is the spectral radiance of the source, $\Omega_{sensor}$ is the solid angle in steradian viewed by the sensor and $\tau_{\lambda,atm}$ is the spectral transmittance of the medium between the source and the sensor. In the case of a spectrally variant detector, the detector's spectral response is included in the value of $\tau_{\lambda,sys}$. The irradiance due to the atmospheric path between the source and the sensor is given by Equation (2.6).

$$E_{path,\lambda} = L_{path,\lambda}\Omega_{sensor}\tau_{\lambda,sys}. \tag{2.6}$$

When the path is uniform, this term is given by

$$E_{path,\lambda} = L_{blackbody,\lambda}\ \Omega_{sensor}\tau_{\lambda,sys}\ (1-\tau_{\lambda,atm}). \tag{2.7}$$

$L_{blackbody,\lambda}$ is the radiance from a blackbody source at the temperature of the atmospheric path. The term $(1-\tau_{\lambda,atm})$ defines the emissivity of this source. The irradiance on the sensor varies as a function of the transmittance of the medium between the source and the sensor.

### 2.2.2 Irradiance on a pixel from a point source

When the source is subtending an angle smaller than the sensor's field of view, also shown in Figure 2.2 the irradiance is given by

$$E = E_{source} + E_{path,sourcetosensor} + E_{background} + E_{path,backgroundtosensor}. \tag{2.8}$$

The irradiance from the components of Equation (2.8) is given in Equation (2.10). The irradiance from the source is given by

$$E_{source,\lambda} = \frac{L_{source,\lambda}Area_{source}\tau_\lambda}{r^2}, \tag{2.9}$$

with $Area_{source}$ the projected area of the source, $L$ the radiance of the source and $r$ the distance between the sensor and the source. If the sensor's responsivity is spectrally variant, $\tau_\lambda$ includes the detector's spectral responsivity. The irradiance from the source varies as a function of the distance between the source and sensor, as well as the transmittance of the medium between the source and the sensor.

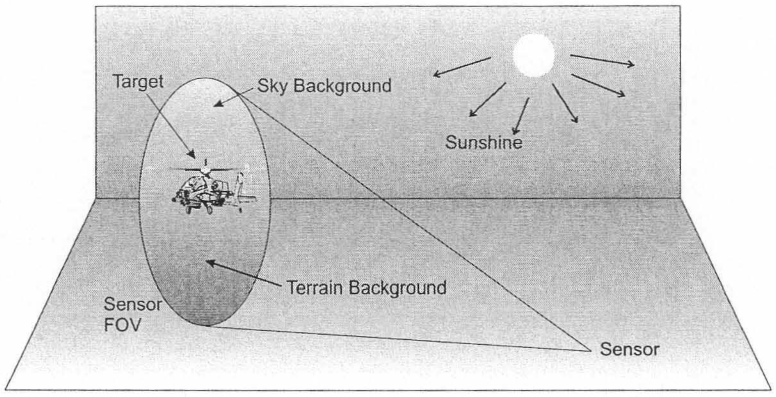

Figure 2.3: Some of the sources of radiance in a scenario

## 2.3 Elements of the radiometric environment

A radiometric scene can be a complex combination of sources, a detector and propagating medium. The radiance generated, reflected and transmitted by these objects combine to form the irradiance measured by the sensor. A typical scenario is shown in Figure 2.3.

### 2.3.1 Blackbody, graybody and spectral sources

In Section 2.1 a blackbody is defined as a perfect emittter, with an emissivity $\varepsilon = 1$. A graybody is defined as an emitter with a constant emissivity, at a value smaller than one. A spectral source is defined as an emitter with an emissivity that is varying as a function of wavelength. These concepts are illustrated in Figure 2.4.

### 2.3.2 Sources of radiance

The irradiance on a sensor is a combination of self-emitted and reflected radiance from the source, radiance from the background and the atmospheric path as well as radiance from objects in the scenario such as terrain that often contribute to the clutter in the scene. A possible sensor view is shown in Figure 2.3. The irradiance on the sensor is given by

$$\begin{aligned} E &= \int_{\lambda_0}^{\lambda_1} L(\lambda)_{target} \mathcal{S}_\lambda \Omega_{target} (\tau_{\lambda,target}) d\lambda \\ &\quad + \int_{\lambda_0}^{\lambda_1} L(\lambda)_{background} \mathcal{S}_\lambda \Omega_{background} (\tau_{\lambda,background}) d\lambda \\ &\quad + \int_{\lambda_0}^{\lambda_1} L(\lambda)_{path\ sensor\ to\ target} \mathcal{S}_\lambda \Omega_{target} (1 - \tau_{\lambda,target}) d\lambda \\ &\quad + \int_{\lambda_0}^{\lambda_1} L(\lambda)_{path\ sensor\ to\ background} \mathcal{S}_\lambda \Omega_{background} (1 - \tau_{\lambda,background}) d\lambda \end{aligned} \tag{2.10}$$

Figure 2.5 shows a simulation of the implementation of Equation (2.10). $\mathcal{S}_\lambda$ represents the spectral response of system. The irradiance from the different components were calculated

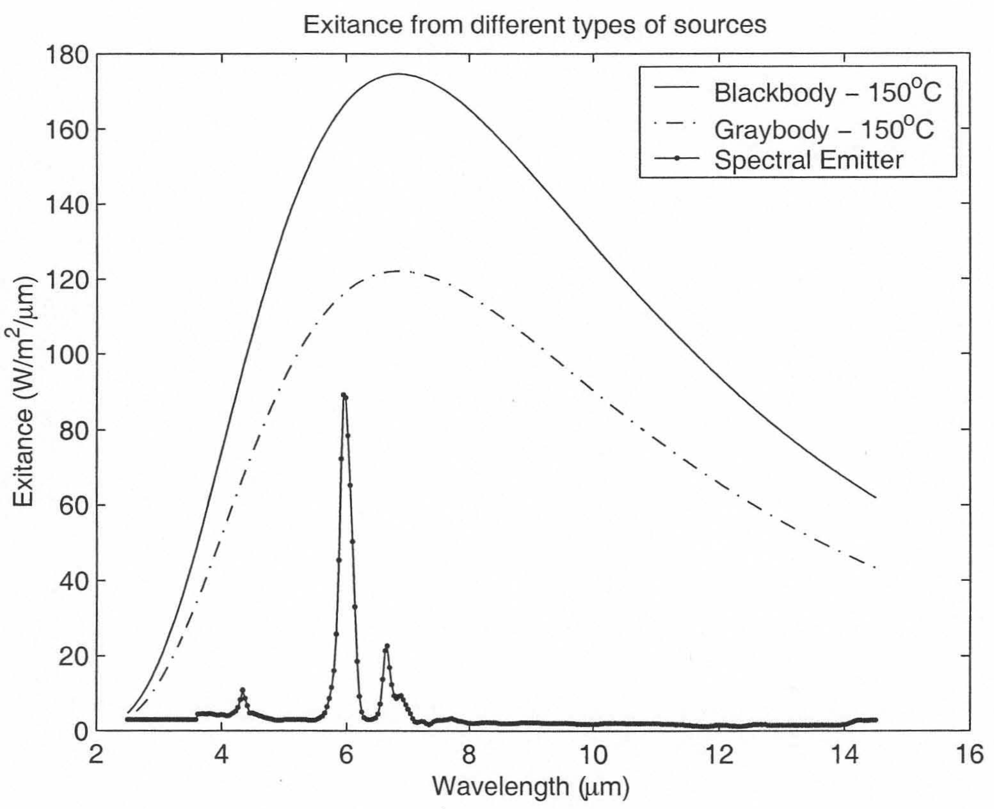

Figure 2.4: Spectral exitance of different types of sources

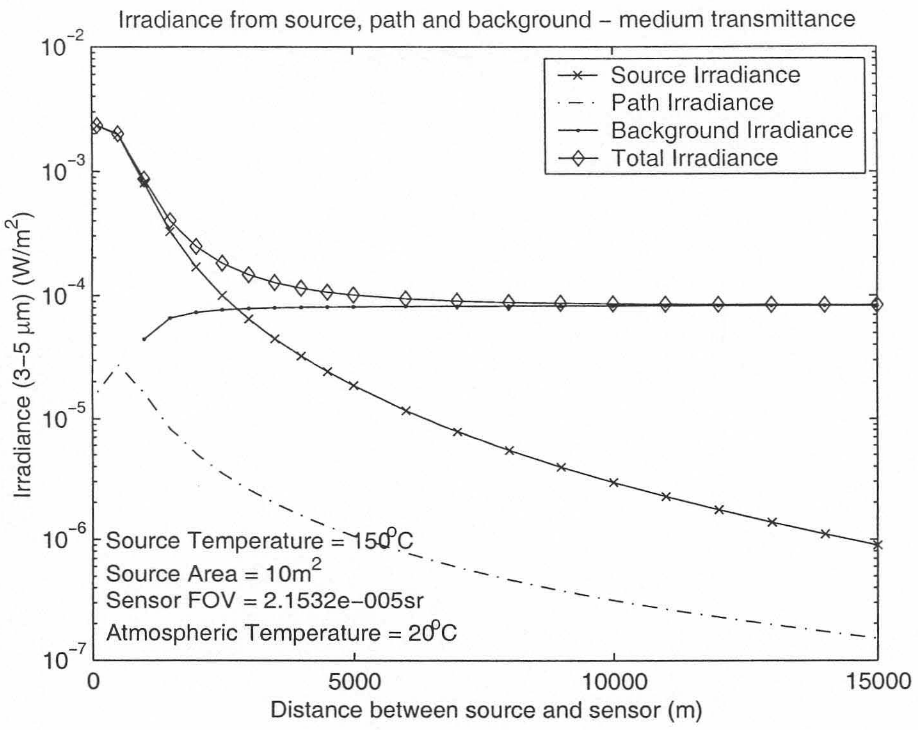

Figure 2.5: Simulation of the contribution of different sources to total irradiance at a sensor

spectrally and integrated to obtain the irradiance on a detector with the source at different distances from the detector. The input parameters that were used in the calculations are:

- Source temperature: 150°C
- Source Area: 10m$^2$
- Sensor Field of View: 2.1532x10$^{-5}$sr ≡ 0.3°
- Atmospheric Temperature: 20°C

The source was initially an extended source, but became a point source after 681m, when the angle subtended by the source became smaller than the sensor's field of view. The contribution from the different components of the irradiance are shown in Figure 2.5.

### 2.3.3 Atmosphere

The atmosphere is an important factor in determining the performance of most electro-optical systems. Absorption, scattering and turbulence often determine the operational capability of electro-optical systems operating in the earth's atmosphere. From Smith [14], the atmospheric transmittance at a single wavelength is given by

$$\tau(r) = e^{(-\alpha r)}, \tag{2.11}$$

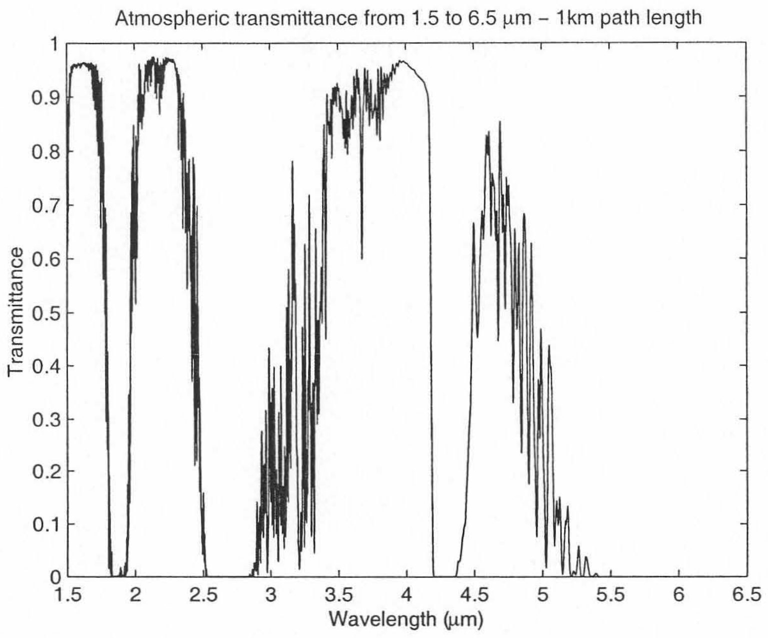

Figure 2.6: Example of atmospheric transmittance

with $\tau$ the atmospheric transmittance, $r$ the distance and $\alpha$ the extinction coefficient. The extinction coefficient is highly dependent on the wavelength and atmospheric parameters such as humidity, temperature and air pressure. A number of models for atmospheric transmittance exist, examples include LOWTRAN, MODTRAN and FASCODE. An example of atmospheric transmittance over a 1 km path length is shown in Figure 2.6. The data was generated using MODTRAN 3. The input parameters were selected to represent an atmosphere that can be expected in Pretoria, South Africa, in the summer.

### 2.3.4 Detector characteristics

The detector in an electro-optical system is a transducer that converts electro-magnetic energy to a form that can be used in further processing. According to Wyatt [13, p79], electro-optical systems use two detector types: thermal detectors and photon detectors.

Thermal sensors make use of the heating effect of radiation. Their response is therefore dependent on the amount of energy absorbed, but not on the spectral content of the absorbed energy. Thermal sensors are used in thermocouples and in new generation imaging sensors such as uncooled bolometer arrays.

Photon detectors respond to photons that have more than a certain minimum energy. The response to photons at a wavelength is proportional to the rate at which photons of that energy are absorbed. Photon detectors include photoemissive detectors such as photomultiplier tubes, semiconductor photoconductive and photovoltaic detectors and photographic film.

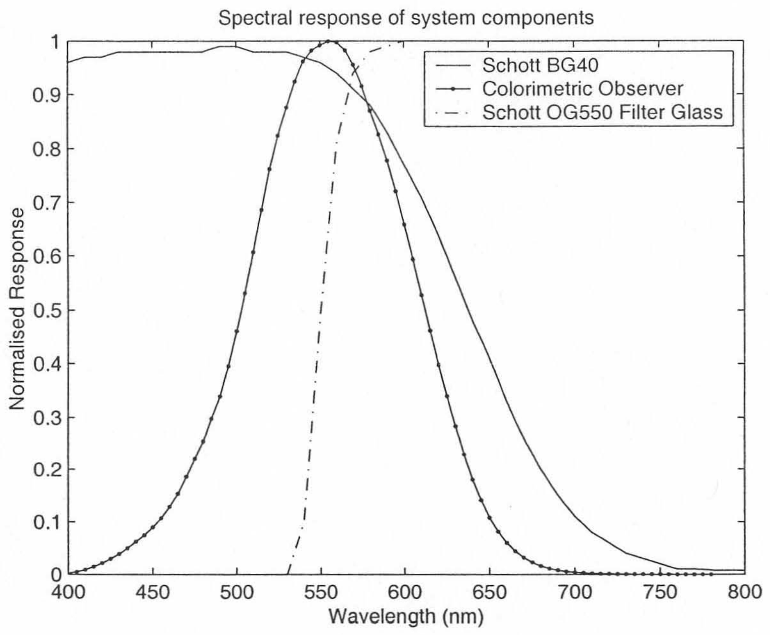

Figure 2.7: Normalised spectral response of the components of a sensor system

### 2.3.5 Spectral response of electro-optical sensors

The spectral response of a sensor system is determined by the response of the detector, the optical system and possible filter elements that can be introduced in the optical path. The response of the system is given by:

$$\mathcal{S} = \int_0^\infty R_{(\lambda,detector)} R_{(\lambda,opticalsystem)} R_{(\lambda,filter)} d\lambda. \tag{2.12}$$

The spectral responses for the components of a system are shown in Figure 2.7. The following components are used in this example system:

- **Schott BG40** is a filter glass passing short wavelengths with the normalised response shown in Figure 2.7.
- A **colorimetric observer** is a standard response based on the average response of the human eye. It is also known as the $V_\lambda$ curve. The detector in this case is therefore the human eye.
- **Schott OG550** is a filter glass passing longer wavelengths with the normalised response shown in Figure 2.7.

The system's spectral response is shown in Figure 2.8.

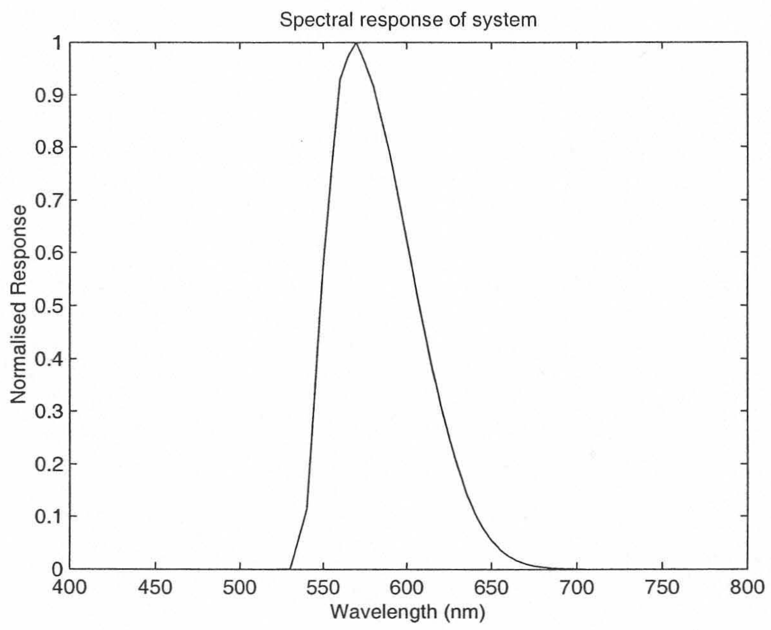

Figure 2.8: Normalised spectral response of a sensor system with the components shown in Figure 2.7

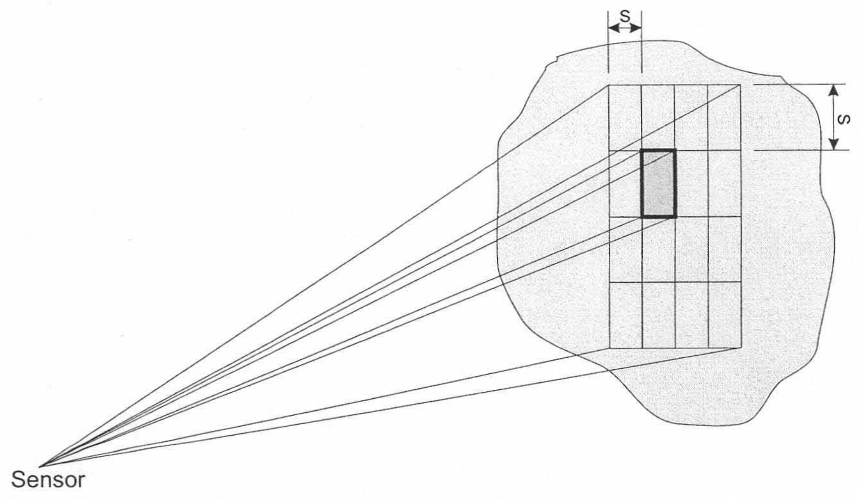

Figure 2.9: Image rendered at different pixel field of views

## 2.4 Irradiance conversion - different render resolutions

Rendering systems can be implemented where part of the scene is rendered at a higher resolution than the rest of the scene. The main advantage of such a scheme is that it can reduce the time required to render a scene, especially if parts of the scene do not contain a lot of detail. It is not necessary to render objects with low levels of detail at high resolution in order to generate realistic images. Figure 2.9 shows a case where a part of an image is rendered at two different resolutions. The high resolution pixel elements have a size of $s \times s$, whereas the low resolution pixel element has a resolution of $4s \times 4s$. The irradiance on the sensor for the low resolution case is given by:

$$\begin{aligned} E &= L\Omega\tau, & (2.13)\\ &= \frac{M}{\pi}\frac{(4s)^2}{r^2}\tau, & (2.14)\\ &= 16\frac{Ms^2\tau}{\pi r^2}. & (2.15) \end{aligned}$$

The radiance from a Lambertian source is given by $L = \frac{M}{\pi}$, with $M$ the exitance of the source [13, p44]. The radiance value was substituted in Equation (2.13).

The distance between the sensor and the object is given by $r$, whereas the transmittance of the atmosphere is given by $\tau$. The irradiance on the sensor in the high resolution case is given by:

$$\begin{aligned} E &= L\Omega\tau, & (2.16)\\ &= \frac{M}{\pi}\frac{(s)^2}{r^2}\tau, & (2.17)\\ &= \frac{Ms^2\tau}{\pi r^2}. & (2.18) \end{aligned}$$

The conclusion in Equation (2.18) can be generalised to the following statement: The irradiance on a lower resolution pixel is therefore the sum of the irradiance on higher resolution pixels, given that the field of view of the lower resolution pixel is the sum of the fields of view of the higher resolution pixels.

## 2.5 The generation of high-fidelity infrared images

Generating high-fidelity infrared images require the implementation of Equation (2.10) for each surface element in a detector's field of view. This process can be extremely time-consuming and the field of computer graphics developed to deal with this process. Chapter 3 is an overview of the elements of computer graphics that are relevant to this study. The implementation of an image rendering system in OpenGL is investigated. It was decided to implement the image rendering system in OpenGL due to its portability across operating systems such as Windows NT, Irix, Linux and Solaris, as well as hardware platforms such as personal computers, Sun machines and high-end graphics supercomputers from SGI.

# Chapter 3

# Computer Graphics Overview

## 3.1 Introduction

The calculation of computer generated images takes up a large amount of processor time. The field of computer graphics was developed to increase the speed and fidelity of computer generated images. The purpose of this chapter is to highlight some of the fundamentals of computer graphics that are relevant to this study. A short overview is given of the conventions used in computer graphics and matrix operations. Two and three-dimensional transforms are investigated, leading to a discussion on perspective and orthogonal projections. Texture maps are used to simplify the generation of physically realistic imagery. The main elements of the OpenGL API are discussed. This chapter is based on Foley *et al.* [15], except where otherwise noted.

## 3.2 History of computer graphics

A short history of computer graphics is given by Foley *et al.* [15].

> Plotting on hard-copy devices such as teletypes and line printers date from the earliest computers. The Whirlwind Computer, developed at MIT in 1950, used computer driven CRT (cathode ray tube) displays for output. The SAGE Air Defense System in the middle of the 1950s used CRT display consoles on which operators identified targets by pointing at them with light pens. Ivan Sutherland founded modern interactive computer graphics with the development of the Sketchpad drawing system as part of his Ph.D. work in 1963. He introduced data structures for storing symbol hierarchies which are built up by replication of standard components as well as interaction techniques for using the keyboard and a light-pen for choice-making, pointing and drawing. Computer graphics were then largely driven by developments in computer-aided design and computer-aided manufacturing. The display devices developed in the mid-sixties are called vector displays. They consist of a display processor, a display buffer memory and a CRT. The buffer stores the computer generated display-list or display program which contained point and line plotting commands as well as character plotting commands. These commands are interpreted by the display processor, which controls the display on the CRT. Vector displays of up to 4096x4096 points were

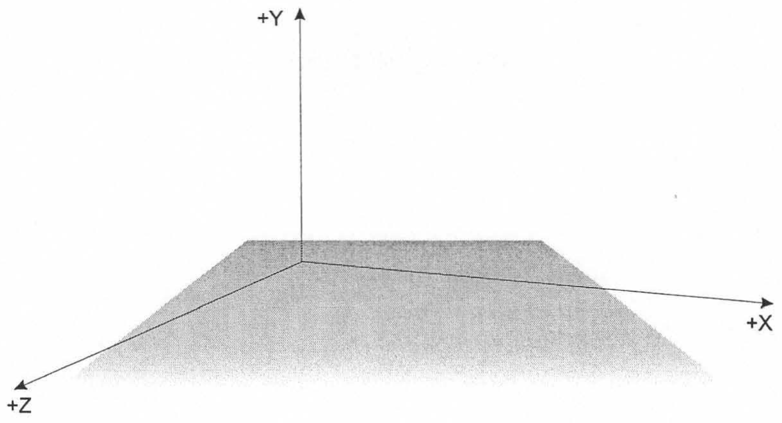

Figure 3.1: Right-handed coordinate system

developed.

In the seventies the generation of images was moved from the central computer to a mini-computer connected to the display device. The most significant development in the mid-seventies, however, was the appearance of low-cost solid state memory. This lead to the development of raster displays. In the raster displays, primitives such as lines, characters and polygons are stored in a refresh buffer in terms of their component points, called pixels. The image is formed from the raster. The raster is simply a matrix of pixels covering the display area.

OpenGL was developed from SGI's Iris GL [16]. Iris GL was used by SGI's high-end graphics workstations as a 3D API (application programming interface). Iris GL required specialised hardware to display graphics. OpenGL was developed to be a more portable graphics API than Iris GL. SGI made the OpenGL available to other vendors through licensing and the OpenGL ARB (architecture review board) was formed. The founding members of the OpenGL ARB were SGI, DEC, IBM, Intel and Microsoft. Version 1.0 of the OpenGL specification was introduced on July 1, 1992. The current specification of OpenGL, Version 1.2.1, was released on April 1, 1999.

## 3.3 Coordinate Convention

The OpenGL library uses a right-handed coordinate system as illustrated in Figure 3.1. This is from Wright *et al.* [16, p137].

## 3.4 Two-dimensional transformations

Matrix operations are fundamental to the implementation of a computer graphics system. Sections 3.4 and 3.5 are included to highlight the basic matrix transformations required for two- and three-dimensional computer graphics.

The multiplication of matrix $A = (a_{ij})$, a $m \times n$ matrix, and matrix $B = (b_{ij})$, a $n \times p$ matrix, is defined by de la Rosa *et al.*[17, p205] as the matrix $AB = (c_{ij})$ where

$$\begin{aligned} c_{ij} &= (a_{i1}, a_{i2}, \dots, a_{in}) \cdot (b_{1i}, b_{2i}, \dots, b_{3i}), \\ &= \sum_{t=1}^{n} a_{it} b_{tj}, \quad (1 \leq i \leq m, \quad 1 \leq j \leq p). \end{aligned} \tag{3.1}$$

$AB$ is an $m \times p$ matrix.

The discussion on two-dimensional transformations is based on Foley *et al.*[15]. Points in the $xy$-plane can be translated to new positions by adding translation amounts to the points. If the point is to be moved $kx$ units parallel to the $x$-axis and $ly$ units parallel to the $y$-axis, the coordinates of the new position is given by:

$$x' = x + kx, \tag{3.2}$$
$$y' = y + ly. \tag{3.3}$$

Similarly, an object can be translated by applying Equation (3.2) to each point of the object. All points on a line can be translated by translating the endpoints and drawing a line between the translated endpoints.

The points (as end-points of vectors) can also be scaled along the $x$-axis and $y$-axis by multiplying by $S_x$ and $S_y$ respectively. The new endpoints are given by:

$$x' = S_x \times x, \tag{3.4}$$
$$y' = S_y \times y. \tag{3.5}$$

In the case of differential scaling the values of $S_x$ and $S_y$ are not equal, leading to a change in the proportions of an object made up of multiple points.

The points can also be rotated through an angle $\theta$ about the origin. The coordinates of the new point are given by:

$$x' = x \times \cos\theta - y \times \sin\theta, \tag{3.6}$$
$$y' = x \times \sin\theta + y \times \cos\theta. \tag{3.7}$$

Equations (3.6) and (3.7) are given in matrix form by:

$$[x' \quad y'] = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}. \tag{3.8}$$

The translation, scaling and rotation of an object around the origin is illustrated in Figure 3.2.

The matrix representations for translation, scaling and rotation are:

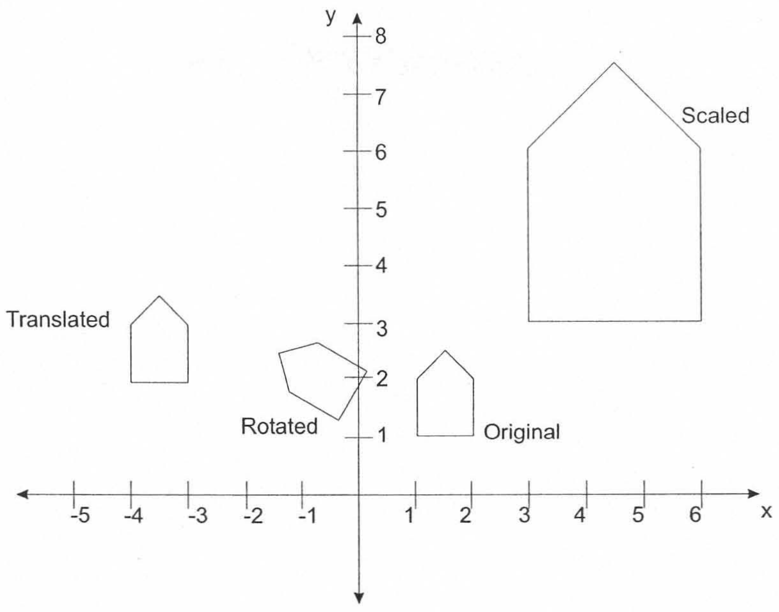

Figure 3.2: The translation, scaling and rotation of an object

$$\begin{aligned} P' &= P + T \quad \text{Translation,} &(3.9)\\ P' &= P \times S \quad \text{Scaling,} &(3.10)\\ P' &= P \times R \quad \text{Rotation.} &(3.11)\end{aligned}$$

Translation is the result of an addition, whereas scaling and rotation are the results of matrix multiplications. Ideally, all three transformations should be treated in a similar way, so that they can be combined in a single matrix operation. If the points are expressed in *homogeneous coordinates*, all three transformations can be treated as multiplications. In homogeneous coordinates, point $P(x, y)$ is represented as $P(W \times x, W \times y, W)$ for any scale factor $W \neq 0$. Given a homogeneous coordinate representation of a point $P(X, Y, W)$, the two-dimensional cartesian representation of the point is $x = \frac{X}{W}$ and $y = \frac{Y}{W}$. Points are now 3-element row vectors, leading to $3 \times 3$ transformation matrices to obtain another 3-element row vector. The translation matrix is now given by:

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x & D_y & 1 \end{bmatrix}, \tag{3.12}$$

with $D_x$ and $D_y$ the translation in $x$ and $y$, respectively.
The scaling matrix is given by:

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \times \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.13}$$

with $S_x$ and $S_y$ the scaling in the $x$ and $y$ dimensions, respectively.

Table 3.1: Scaled, rotated and translated coordinate points

| Original x | Original y | Scaled x | Scaled y | Rotated x | Rotated y | Translated x | Translated y |
|---|---|---|---|---|---|---|---|
| 1.000 | 1.000 | 3.000 | 3.000 | -0.366 | 1.366 | -3.000 | -2.000 |
| 2.000 | 1.000 | 6.000 | 3.000 | 0.134 | 2.232 | -2.000 | 2.000 |
| 2.000 | 2.000 | 6.000 | 6.000 | -0.732 | 2.732 | -2.000 | 3.000 |
| 1.500 | 2.500 | 4.500 | 7.500 | -1.415 | 2.549 | -2.500 | 3.500 |
| 1.000 | 2.000 | 3.000 | 6.000 | -1.232 | 1.866 | -3.000 | 3.000 |

The rotation matrix is given by:

$$[x' \quad y' \quad 1] = [x \quad y \quad 1] \times \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.14}$$

with $\theta$ the rotation around the origin. The effects of the different transformations are illustrated in Figure 3.2. The original coordinates of the polygon are given with the scaled, rotated and translated coordinates in Table 3.1. The polygon is scaled by *(2,2)*, rotated +60°around the origin and translated by *(-4,1)*, respectively.

## 3.5 Three-dimensional transformations

The discussion on three-dimensional transformations is based on Angel [18]. Points in the three-dimensional space can also be translated, rotated and scaled around a reference point. Translation moves a point **p** to a new point **p**$'$ using a displacement vector **d**. This can be written as:

$$\mathbf{p} = [x \quad y \quad z], \qquad \text{(the original point)} \tag{3.15}$$

$$\mathbf{d} = [\alpha_x \quad \alpha_y \quad \alpha_z], \qquad \text{(the translation vector)} \tag{3.16}$$

$$\mathbf{p}' = \mathbf{p} + \mathbf{d}, \qquad \text{(the new point)} \tag{3.17}$$

therefore

$$x' = x + \alpha_x, \tag{3.18}$$

$$y' = y + \alpha_y, \tag{3.19}$$

$$z' = z + \alpha_z. \tag{3.20}$$

In homogeneous coordinates this is given as:

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \times \begin{bmatrix} 1 & 0 & 0 & \alpha_x \\ 0 & 1 & 0 & \alpha_y \\ 0 & 0 & 1 & \alpha_z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.21}$$

i 16035665
b15433055

In the case of scaling and rotating, there is a fixed point that is unchanged by the transformation. For the cases where the fixed point is not the origin, transformations can be concatenated to obtain the transformation for the point. A point can be scaled, with the fixed point at the origin, using:

$$x' = \beta_x x, \tag{3.22}$$
$$y' = \beta_y y, \tag{3.23}$$
$$z' = \beta_z z. \tag{3.24}$$

These equations can be combined in homogeneous form as:

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \times \begin{bmatrix} \beta_x & 0 & 0 & 0 \\ 0 & \beta_y & 0 & 0 \\ 0 & 0 & \beta_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.25}$$

In rotating a point around the origin, there are three possible degrees of freedom, corresponding to the ability to rotate the point independently about the three coordinate axis. The sequence of rotation is important because a rotation about the $x$ axis by an angle $\theta$ followed by a rotation about the $y$ axis by an angle $\phi$ would not give the same results as when the order of rotations were reversed. The equations for three-dimensional rotation about an axis can be derived by noticing that, for example, rotating around the $z$ axis is equivalent to rotating in two dimensions in the $xy$ plane, while keeping $z$ constant. From Equation (3.14) the equations for rotation about the $z$ axis by an angle of $\theta$ is given by:

$$x' = x\cos\theta - y\sin\theta, \tag{3.26}$$
$$y' = x\sin\theta + y\cos\theta, \tag{3.27}$$
$$z' = z. \tag{3.28}$$

In matrix form, rotation about the $z$ axis can be written as:

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \times \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.29}$$

Rotation about the $x$ axis is given by:

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \times \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{3.30}$$

whereas rotation about the $y$ axis is given by:

$$[x' \quad y' \quad z' \quad 1] = [x \quad y \quad z \quad 1] \times \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.31}$$

Rotation of an object around a point $\mathbf{p_0}$ would start by a translation to the origin, rotating the object around the $x$, $y$ and $z$ axis as required and then translating the resulting object back to $\mathbf{p_0}$.

When it is required to do successive transformations on a point $\mathbf{p}$, the transformations can be concatenated. In Equation (3.32) three transformations, $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are carried out on $\mathbf{p}$, resulting in $\mathbf{q}$:

$$\begin{aligned} \mathbf{q} &= \mathbf{CBAp}, && (3.32) \\ &= (\mathbf{C}(\mathbf{B}(\mathbf{Ap}))). && (3.33) \end{aligned}$$

In the case of many points that must be transformed, a new transformation, $\mathbf{M}$, can be calculated, with

$$\mathbf{M} = \mathbf{CBA}. \tag{3.34}$$

The matrix in Equation (3.34) can be used to calculate $\mathbf{q}$, therefore

$$\mathbf{q} = \mathbf{Mp}. \tag{3.35}$$

The use of a transformation such as $\mathbf{M}$, can save a significant amount of calculations.

## 3.6 Orthogonal and perspective projections

Objects in a scene, such as shown in Figure 3.3, can be projected in two ways onto a display. The first is orthogonal projection as shown in Figure 3.4. In this case objects with the same dimensions appear the same size, regardless of their distance from the viewing plane. This type of projection is used in CAD drawings, architectural design and two-dimensional graphs.

The result of perspective projection is shown in Figure 3.5. This projection adds the effect that distant objects appear smaller than nearby objects. Perspective projection is widely used in simulation and three-dimensional animation and is responsible for a large part of the realism.

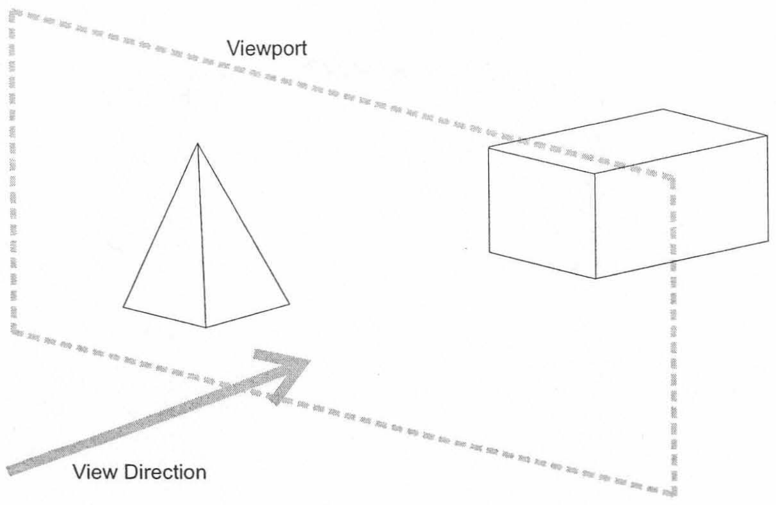

Figure 3.3: Terrain with two objects

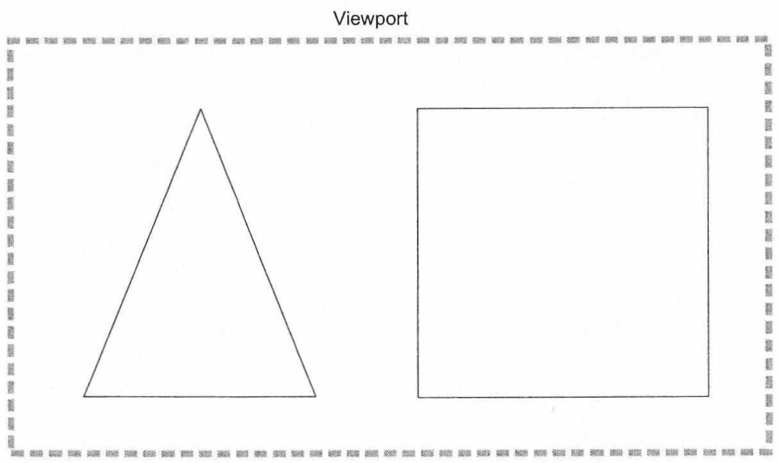

Figure 3.4: View using orthogonal projection

## 3.7 Texture maps

Wright *et al.* [16] regard texture mapping as one of the most significant developments in computer graphics in the past ten years. Texture mapping is used to supply realism to computer generated images. A texture map is an image that is fitted to polygons in a scene in order to generate realistic imagery. Texture mapping is used for applications that include anti-aliased text and reflection mapping. Texture mapping requires intensive calculations and can be slow on 2D graphics cards where all the calculations are done on the computer's CPU. Newer generation 3D graphics cards support texturing in hardware, leading to performance levels only found on graphics supercomputers a few years ago.

## 3.8 The open graphics library (OpenGL)

This section is based on Segal *et al.* [19]. OpenGL is concerned only with rendering into a framebuffer and reading values from the framebuffer. It does not provide support for devices such as mice and keyboards. OpenGL draws primitives subject to a number of selectable modes. Primitives include points, line segments, polygons and pixel rectangles. The primitives are defined by a group of one or more vertices. A vertex defines a point, an end-point of an edge, or a corner of a polygon where two edges meet. A vertex is defined by positional coordinates, colours, normals and texture coordinates.

The model for the interpretation of OpenGL commands is client-server. That means that a program (the client) issues commands and these commands are interpreted and processed by OpenGL (the server). The server and client may or may not be operating on the same computer. OpenGL is designed to run on a range of graphics platforms with varying capabilities and performance.

Commands are processed in the order in which they are received. A primitive must therefore be drawn completely before a subsequent one can influence the frame buffer. Another implication is that the results of pixel read operations are consistent when all previously invoked OpenGL commands are completed.

OpenGL commands are functions. Various groups of commands can perform the same operation, but differ in how arguments are supplied to them. The general syntax for a command declaration is:

$$\textit{rtype}\ \textbf{Name}\{\epsilon\textbf{1234}\}\{\epsilon\ \textbf{b s i f d ub us ui}\}\{\epsilon\textbf{v}\}$$
$$\textit{( [args ,] T arg1,...,T argN [, args] );}$$

*rtype* is the return type of the function. The braces ({}) enclose a series of characters of which one is selected. $\epsilon$ indicates no character is selected. The arguments enclosed in brackets may or may not be present. If the final character is not **v**, then the number of arguments is **1**, **2**, **3**

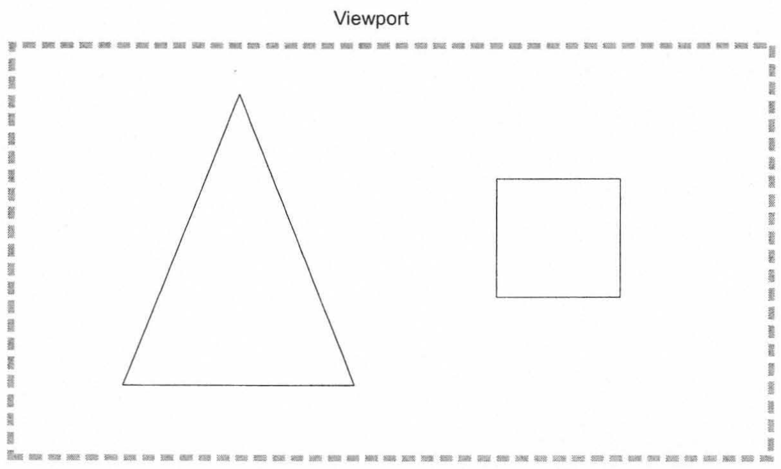

Figure 3.5: View using perspective projection

or **4**. If the final character is **v**, then only *arg1* is present and it is an array of $N$ values of the indicated type.

The syntax can be illustrated using vertex specification as an example. Vertices are specified by giving their coordinates in two, three or four dimensions. The general versions of the command are:

void **Vertex**{**234**}{**sifd**}( T *coords* ) ;
void **Vertex**{**234**}{**sifd**}**v**( T *coords* ) ;

A call to any **Vertex** command specifies four coordinates: $x$, $y$, $z$ and $w$. The $x$ coordinate is the first coordinate, $y$ the second, $z$ the third and $w$ the fourth. A call to **Vertex2** sets the $x$ and $y$ coordinates while the $z$ coordinate is implicitly set to zero and $w$ is set to one. **Vertex3** sets the $x$, $y$ and $z$ coordinates to the provided values, while $w$ is set to one. **Vertex4** sets all four coordinates, allowing the specification of a point in homogeneous coordinates. The type specifiers **s**, **i**, **f** and **d** represent the standard types short, integer, float or double.

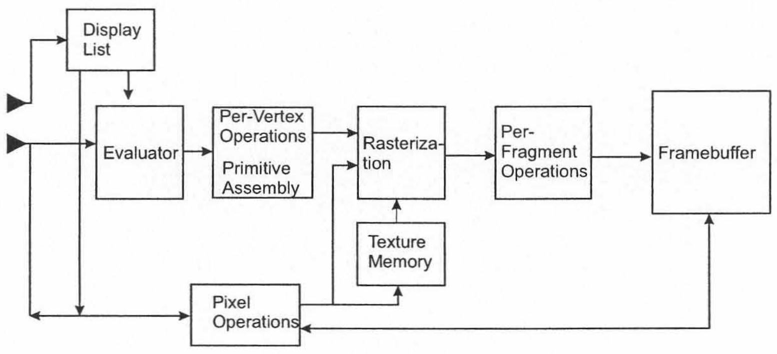

Figure 3.6: Block diagram of OpenGL

Figure 3.6 shows a schematic diagram of OpenGL. Some commands specify geometric objects to be drawn, whereas others control how the objects are handled by the various stages. Most commands may be accumulated in a display list for later processing. Commands that are not included in a display list are sent through a processing pipeline.

The first stage approximates curve and surface geometry by evaluating polynomial functions of input values. The next stage operates on geometric primitives described by vertices: points, line segments and polygons. In this stage vertices are transformed and lit and primitives are clipped to a viewing volume in preparation for the next stage. The rasterizer produces a series of framebuffer addresses and values using a two-dimensional description of a point, line segment or polygon. Each fragment produced by the rasterizer is fed to the next stage that performs operations on individual fragments before they finally alter the framebuffer. These operations include conditional updates into the framebuffer based on incoming and previously stored depth values, blending of incoming fragment colours with stored colours as well as masking and other logical operations on fragment values.

The vertex processing portion of the pipeline can be bypassed to send a block of fragments directly to the individual fragment operations, eventually causing a block of pixels to be written to the framebuffer. Values may also be read back from the framebuffer or copied from one portion of the framebuffer to another.

Vertices, normals and texture coordinates are transformed before their coordinates are used to produce an image in the framebuffer. Figure 3.7 shows the sequence of transformations. The vertex coordinates that are presented to OpenGL are termed *object coordinates*. The *model-view matrix* is applied to these coordinates to yield *eye coordinates*. The next matrix, the projection matrix, is applied to the eye coordinates to yield *clip coordinates*. A perspective division is carried out on clip coordinates to yield *normalised device coordinates*. A final *viewport* transformation is applied to convert these coordinates into *window coordinates*.

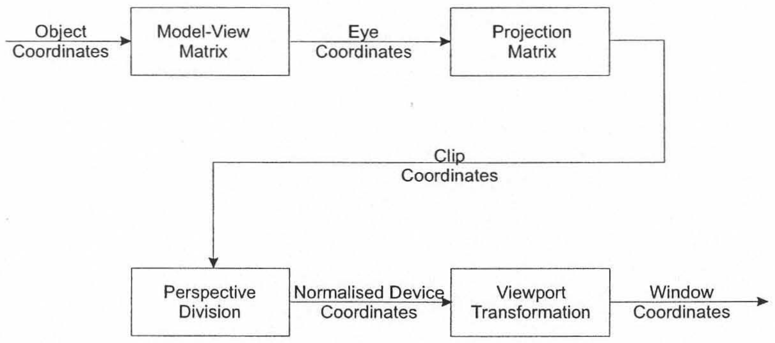

Figure 3.7: Vertex transformation sequence

## 3.9 OpenGL Utilities: GLUT

OpenGL was developed to be platform independent. It does not contain functions or commands for actions such as window management, keyboard input or mouse interaction. OpenGL was initially released with AUX, the OpenGL auxiliary library. The library was created to facilitate the learning and writing of OpenGL programs and did not contain basic GUI features.

AUX has been largely replaced by GLUT, the *OpenGL utility toolkit*. GLUT was written by Mark Kilgard at SGI. It includes pop-up menus, window management, keyboard and mouse interaction and even joy-stick support. GLUT makes it possible to port programs with relatively little effort between different platforms.

## 3.10 A short OpenGL program

This section shows a short OpenGL program that was written using GLUT. The listing is based on code from Wright *et al.*[16].

```
// A short OpenGL Program using GLUT

#include <windows.h>
```

```
#include <gl/glut.h>

// Main program entry point
void main(void)
        {
        // Set up a double-buffered display and specify
        // the pixel mode to be red, green and blue.
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
        // Creates the display window with title "Short Program"
        glutCreateWindow("Short Program");
        // The function called by glutMainLoop whenever a new
        // scene is to be rendered.
        glutDisplayFunc(RenderScene);
        // Setup the rendering context.  Parameters such as the
        // background colour, lighting and fog can be set up in
        // this function.
        SetupRC();
        // The glutMainLoop calls functions to handle events such
        // as keyboard inputs, timer ticks and scaling of windows.
        glutMainLoop();
        }
void RenderScene(void)
        {
        // Clear the window with current clear color
        glClear(GL_COLOR_BUFFER_BIT);
        // Draw a filled triangle with a dark gray color
        glBegin(GL_TRIANGLES);
                glColor3ub(150, 150, 150);
                glVertex2f(-0.5, -0.5);
                glVertex2f(0.5, -0.5);
                glVertex2f(0.0, 0.5);
        glEnd();
        // Flush drawing commands
        glutSwapBuffers();
        }
// Setup the rendering state
void SetupRC(void)
        {
        // Set clear color to light gray
        glClearColor(0.90f, 0.90f, 0.90f, 1.0f);
        }
```

The output of the program is shown in Figure 3.8.

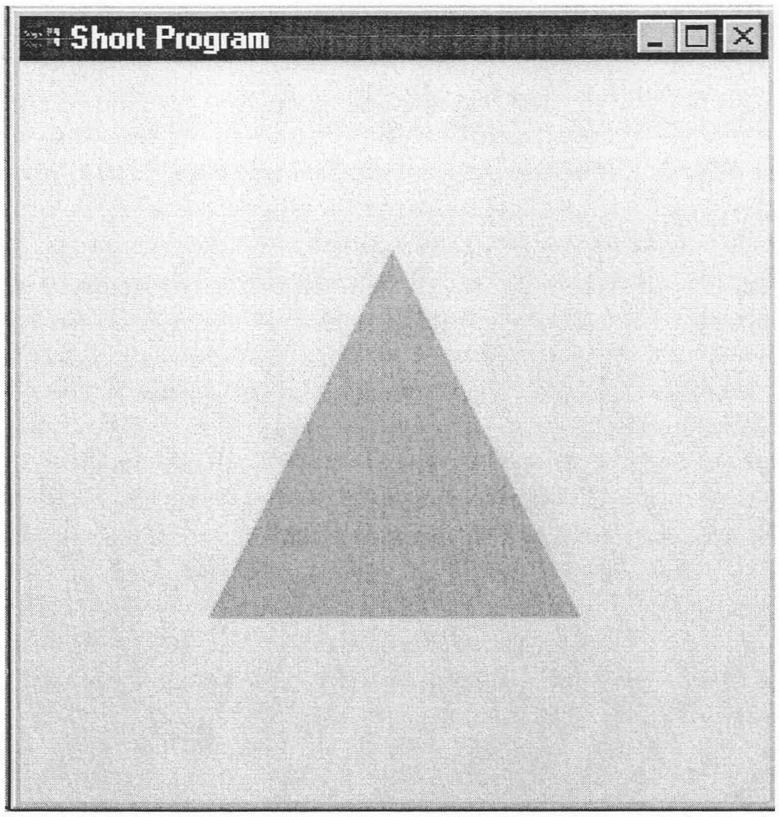

Figure 3.8: Image generated by using the short GLUT and OpenGL program

## 3.11 The use of OpenGL commands to simulate an infrared scenario

### 3.11.1 glColor

A colour can be defined for each vertex. If an infrared scenario is simulated, the colour assigned to a vertex can be replaced with a radiance value. The colours in OpenGL consist of a combination of red, green and blue. The radiance value can be assigned only to a single colour. It is also possible to assign the same radiance value to all three colours. This would lead to gray-scale images, which are similar to the output of typical thermal imagers. The "colour" output seen on some thermal imagers is a false colour map that is assigned according to the irradiance level of the pixel.

**The commercially available graphics accelerators for the PC are limited to a resolution of eight bits per colour. It is therefore only possible to assign 256 radiance levels to an image. This can lead to severe reductions in the radiometric accuracy of the rendered images. A technique to artificially increase the resolution will be investigated in Chapter 4.**

### 3.11.2 glFog

OpenGL provides depth cuing and atmospheric effects through the *glFog* function. Fog provides a way of adding a predefined colour to each vertex in the image, with the amplitude based on the distance between the vertex and the observer. OpenGL supports three kinds of fog namely:

**GL_LINEAR:** linear fog that is used for depth cuing.

**GL_EXP:** exponential fog that is used for heavy fog or clouds.

**GL_EXP2:** exponential fog that is used for smoke and weather haze.

Once a fog type is specified, the fog colour is specified using:

```
Glint fog_colour[4] = {red, green, blue, alpha};
glFogiv(GL_FOG_COLOR, fog_colour};
```

In addition to the fog colour, GL_EXP and GL_EXP2 have an additional density parameter:

```
glFogf(GL_FOG_DENSITY, fog_density};
```

The equivalent infrared atmospheric transmittance is defined using the *density* parameter, wheras the infrared path radiance is defined by specifying the *colour* of the fog. The effect of the three types of fog are shown in Figure 3.9. The data were generated using OpenGL with linear fog from 100m to 2000m, GL_EXP and GL_EXP2 fog with a density parameter of 0.001 and the fog colour set to {0, 0, 0}. The fog colour was black, resulting in no path radiance. The irradiance was normalised with the irradiance calculated for a scenario without fog.

Angel [18, p420] gives the equations for the different types of fog as:

$$\mathbf{GL\_LINEAR\ FOG}: \quad \mathrm{f} = 1 - dz, \tag{3.36}$$
$$\mathbf{GL\_EXP\ FOG}: \quad \mathrm{f} = e^{-dz}, \tag{3.37}$$
$$\mathbf{GL\_EXP2\ FOG}: \quad \mathrm{f} = e^{-(dz)^2}, \tag{3.38}$$

with $d$ the density factor and $z$ the distance. The linear fog is clamped between 1 and 0 using

```
glFogf(GL_FOG_START, start_of_fog);
glFogf(GL_FOG_END, end_of_fog);
```

with the transmittance 100% at the start of the fog and 0% at the end of the fog. Objects closer than GL_FOG_START are rendered without fog effects and objects further than GL_FOG_END are rendered with maximum fog effects. Equation (3.36) are modified by OpenGL to take this into account. The linear fog equation in this case becomes:

$$FOG = \begin{cases} 1 & \text{if z} < \text{GL\_FOG\_START}, \\ 1 - dz & \text{if GL\_FOG\_START} < \text{z} < \text{GL\_FOG\_END}, \\ 0 & \text{if z} > \text{GL\_FOG\_END}. \end{cases} \tag{3.39}$$

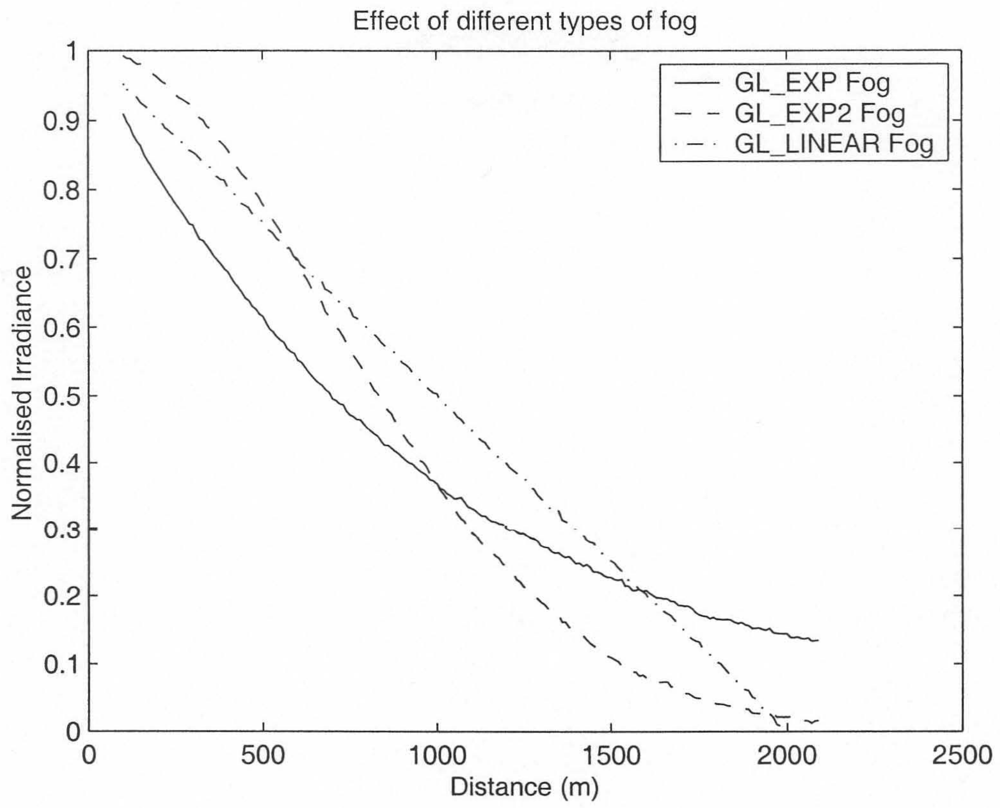

Figure 3.9: Demonstration of the reduction in normalised irradiance due to fog

## 3.11.3 The dynamic range of computer graphics hardware

The dynamic range of PC computer graphics hardware is limited to 8 bits per colour channel. It is not possible to increase the dynamic range in the hardware and another technique must be implemented to artificially increase the dynamic range. This technique is investigated in Chapter 4.

# Chapter 4

# The Dynamic Range of Graphics Accelerators

## 4.1 Introduction

Commercial-off-the-shelf renderers such as the GeForce 2 card from Nvidia can render images at resolutions of up to 24 colour bits per pixel. The pixel is made up of the three basic colours, red, green and blue. The colours are rendered at a resolution of 8 bits each and the three colour channels are independent of each other. The resolution of the rendered infrared images is therefore limited to the resolution of a single colour channel. Olsen, *et al.* [6] and [20], described a technique whereby the resolution of a radiometric image can be increased by combining the outputs from the three colour channels.

The increased dynamic range is obtained by assigning parts of the input dynamic range to the different colour channels. This is illustrated in Figure 4.1, from Olsen *et al.* [20]. Figure 4.1 shows the assignment as it will be used on a personal computer, with a colour resolution of 8 bits per channel and a 14-bit output resolution. The 14-bit input word, which is related to the input radiance, is split into the least significant 8 bits which are assigned to the red channel, the next most significant 3 bits which are padded with 5 zeros are assigned to the green channel and finally the most significant 3 bits which are padded with 5 zeros and assigned to the blue channel.

The image is then rendered in normal fashion and recombined in a single 14-bit word. The method of recombination is shown in Figure 4.2.

## 4.2 Dynamic range of a thermal image

The dynamic range of the radiance values in a thermal image is dependent on the input temperatures in the scene. The radiance was calculated for input temperatures from 0°C to 1000°C and a spectral band of 3 to 5.5 $\mu$m using:

$$L(T) = \int_3^{5.5} \frac{M(\lambda, T)}{\pi} d\lambda, \tag{4.1}$$

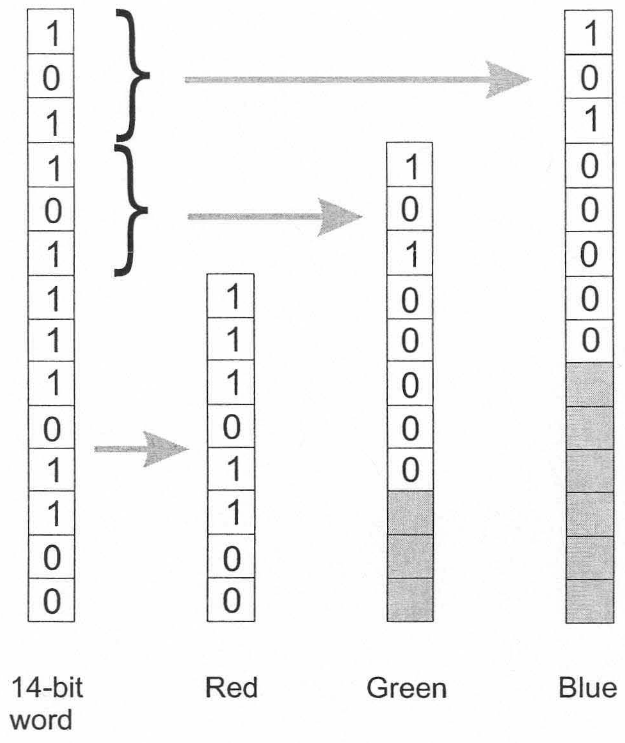

Figure 4.1: Assignment of radiance levels to RGB channels

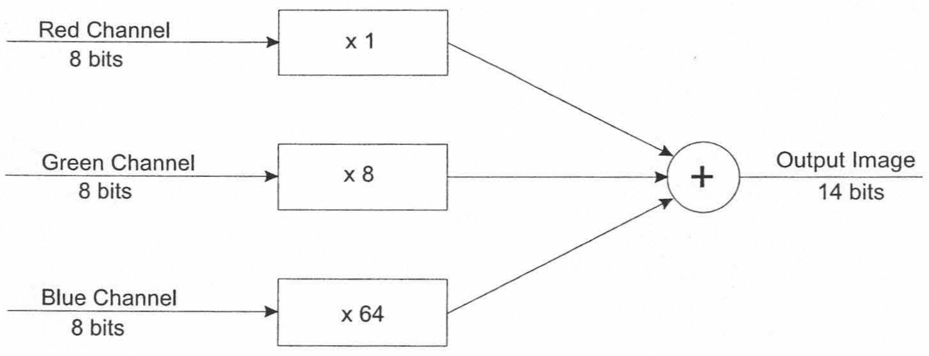

Figure 4.2: Recombination of RGB channels

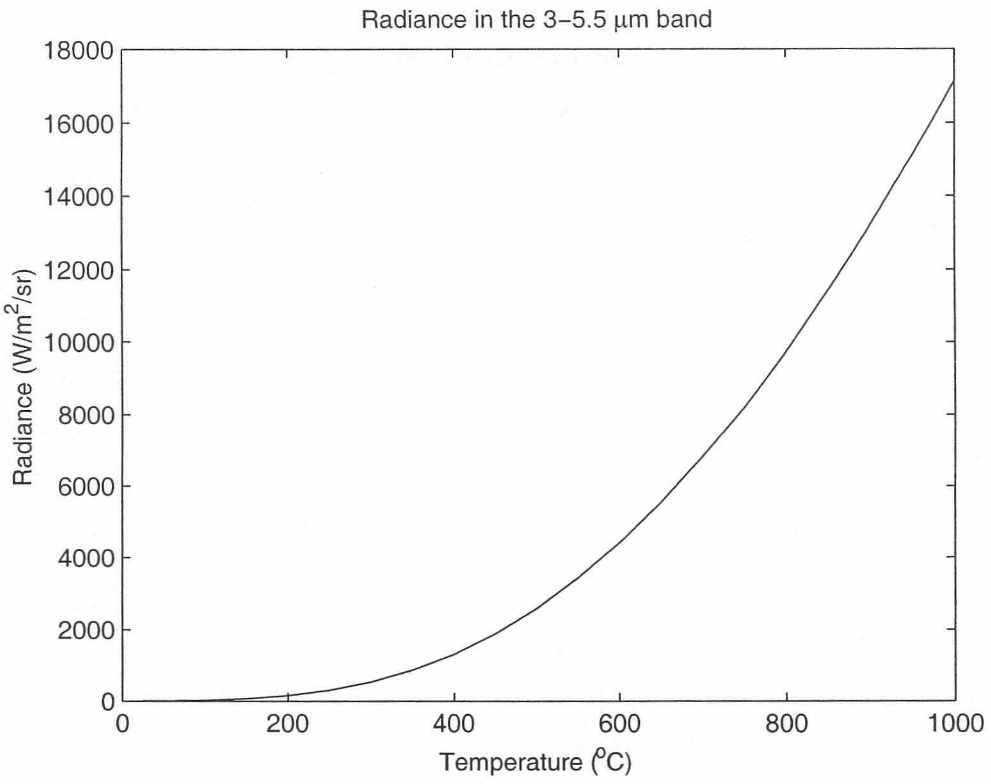

Figure 4.3: Radiance from sources at a range of temperatures

with the exitance as defined in Equation (2.1). The result is shown in Figure 4.3.

If it is assumed that the maximum render resolution is 14 bits (16384 levels) and that the mapping from radiance to gray levels is linear, the maximum temperature range that can be accommodated in 14 bits is from 0°C to 981°C. The temperature values were obtained by inspection of the data used to generate Figure 4.3. The requirement for a linear mapping is added to eliminate an additional processing step on the image, increasing the rendering speed. On an 8-bit rendering system radiance values of 0 to 255 $\mathrm{Wm^{-2}sr^{-1}}$ would be assigned to the red channel, 256 to 2048 $\mathrm{Wm^{-2}sr^{-1}}$ to the red and green channel and 2048 to 16384 $\mathrm{Wm^{-2}sr^{-1}}$ to the red, green and blue channel. The examples in Table 4.1 should clarify the technique.

## 4.3 Errors due to artificially extending the render resolution

In order to investigate the errors caused by artificially extending the rendering resolution, a test object was defined as shown in Figure 4.4. The test object was selected so that its radiometric properties could be calculated exactly, as shown in Equations (4.2) and (4.5). The calculated properties could then be compared with the object that was rendered using OpenGL. The triangle was rendered in OpenGL using:

```
glBegin(GL_TRIANGLES);
```

Table 4.1: Examples of mapping radiance values to colours

| Input Value | Input value in binary | Dynamic Range to be applied | Red Value (8-bits) | Green Value (8-bits) | Blue Value (8-bits) | Comment |
|---|---|---|---|---|---|---|
| 200 | 11001000 | 10-bit | 11001000 | 00000000 | 00000000 | |
| 812 | 1100101100 | 10-bit | 00101100 | 10000000 | 10000000 | |
| 812 | 1100101100 | 12-bit | 00101100 | 11000000 | 00000000 | |
| 812 | 1100101100 | 14-bit | 00101100 | 01100000 | 00000000 | |
| 812 | 1100101100 | 16-bit | 00101100 | 00110000 | 00000000 | |
| 1487 | 10111001111 | 10-bit | | | | > 10-bit |
| 1487 | 10111001111 | 12-bit | 11001111 | 01000000 | 10000000 | |
| 1487 | 10111001111 | 14-bit | 11001111 | 10100000 | 00000000 | |
| 6869 | 1101011010101 | 10-bit | | | | > 10-bit |
| 6869 | 1101011010101 | 12-bit | | | | > 12-bit |
| 6869 | 1101011010101 | 14-bit | 11010101 | 01000000 | 01100000 | |

```
        glColor3f(red1, green1, blue1);
        glVertex3f(-1.0, -1.0, 0.0);
        glColor3f(red2, green2, blue2);
        glVertex3f(1.0, -1.0, 0.0);
        glColor3f(red3, green3, blue3);
        glVertex3f(0.0, 1.0, 0.0);
glEnd();
```

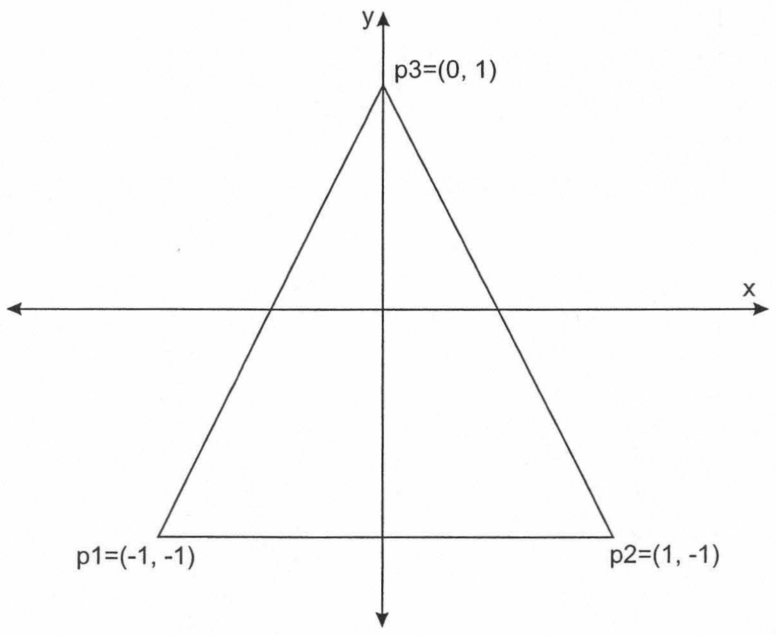

Figure 4.4: Definition of the test object

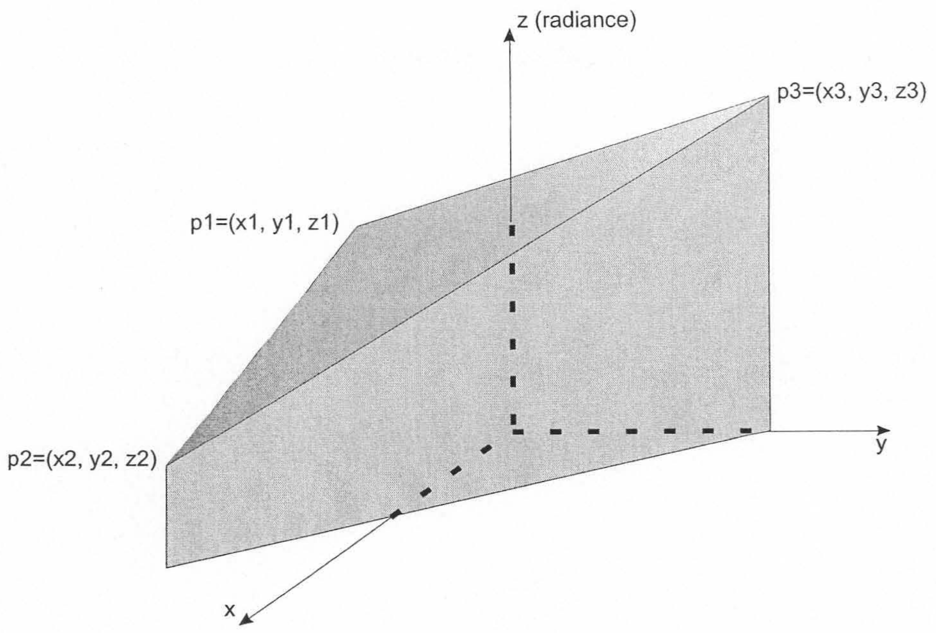

Figure 4.5: Test object with radiance values assigned to the z-axis

By assigning a colour related to radiance to each vertex, it was possible to calculate the exact radiance value of an x,y-point within the triangle, or to calculate the exact intensity of the triangle. An example of the triangle, with radiance values assigned to the z-axis, is shown in Figure 4.5.

The equation of the surface connecting points p1, p2 and p3 in Figure 4.5 is given by:

$$ax + by + cz = d. \tag{4.2}$$

It is therefore possible to find the radiance at any point $(x, y)$ in the triangle by using Equation (4.2). The intensity of the test object is given by:

$$\begin{aligned} I &= LA, && (4.3)\\ &= \int_{-1}^{1}\int_{\frac{y-1}{2}}^{\frac{1-y}{2}} LA \quad dxdy, && (4.4)\\ &= \frac{6d+2b}{3c}. && (4.5)\end{aligned}$$

$L$ is the radiance of the target, $A$ the projected area and $b$, $c$ and $d$ the constants defining the surface in Equation (4.2).

The values in Equation (4.5) were determined for

$$p1 = (-1, \quad -1, \quad z_1), \tag{4.6}$$
$$p2 = (1, \quad -1, \quad z_2) \quad \text{and} \tag{4.7}$$
$$p3 = (0, \quad 1, \quad z_3), \tag{4.8}$$

as shown in Figure 4.4. The triangle can be rendered in OpenGL and the resulting values compared with the theoretically calculated values. The following two experiments were carried out to determine the errors caused by increasing the dynamic range to more than 8-bits.

**Test object at a fixed spatial position.** Random radiance values are determined for each vertex of the test object. The radiance values are assigned to the three colours according to the technique described in Section 4.2. The image is rendered and the radiance value at a random pixel is determined. The radiance value at the same position is calculated using Equation (4.2). The difference between the two values is recorded.

**Test object at an increasing distance from the sensor.** The vertices of the test object are assigned three different radiance values. The radiance values are assigned to the three colours according to the technique described in Section 4.2. The test object is moved from a distance of 100m from the camera to a distance of 2100m with a step size of 10m. The image is rendered at each distance and the intensity is compared with the intensity calculated from Equation (4.5). The rendered intensity is recorded at each distance.

## 4.4 Results

### 4.4.1 Determining difference between rendered and calculated radiance

In this experiment the radiance of the three vertexes of the test object were assigned random radiance values. The radiance value on a random pixel were rendered and theoretically calculated and the results compared. The experiment was repeated 10000 times. The position of the pixel on the triangle and the radiance levels of the three vertexes were varied in each case. The difference between the calculated and the rendered radiance values were determined at resolutions of 8-bit, 10-bit, 12-bit, 14-bit and 16-bit. The differences between the theoretical and rendered radiance values and the error histograms for the 5 cases are plotted in Figure 4.7 to Figure 4.16. The figures are:

- **Figures 4.7 and 4.8** The difference between the rendered and calculated radiance is shown as absolute error and percentage error as function of radiance value in Figure 4.7. The histogram of the absolute error values is shown in Figure 4.8. The radiance resolution was 8-bit or 256 radiance levels.

- **Figures 4.9 and 4.10** The data was determined for a radiance resolution of 10-bit or 1024 gray levels and is presented in the same format as Figures 4.7 and 4.8.

- **Figures 4.11 and 4.12** The data was determined for a radiance resolution of 12-bit or 4096 gray levels and is presented in the same format as Figures 4.7 and 4.8.

- **Figures 4.13 and 4.14** The data was determined for a radiance resolution of 14-bit or 16384 gray levels and is presented in the same format as Figures 4.7 and 4.8.
- **Figures 4.15 and 4.16** The data was determined for a radiance resolution of 16-bit or 65536 gray levels and is presented in the same format as Figures 4.7 and 4.8.
- **Figure 4.6** shows the error histograms of all the dynamic range cases on one graph.

The errors can be ascribed to the following:

- **Rounding Errors** The radiance value of the pixels are calculated using integers. The radiance value is therefore immediately rounded to the nearest integer, leading to an error of $\pm$ 0.5 in the 8-bit case, which is red only. This is multiplied by the scaling value for the green and blue pixels in the cases where more than 8-bits are used. The maximum rounding error would then be: $\pm(0.5 + 0.5 * greenscalefactor + 0.5 * bluescalefactor)$. This is $\pm$3.5, $\pm$10.5, $\pm$36.5 and $\pm$136.5 for 10-bit, 12-bit, 14-bit and 16-bit respectively.
- **Value for pixel compared to value calculated for a point** The theoretical radiance value is calculated for a point, whereas the radiance of a pixel is determined by the radiance values at the four corners of the pixel. The amplitude of this error is determined by the equivalent position of the point in the pixel and the radiance values at the four corners of the pixel.

The maximum error percentage in all the cases is less than 20%. The error percentage drops to below 5% when the rendered radiance value is more than 20% of the maximum rendered value. It can therefore be concluded that it is possible to render images with a less than 5% error in more than 80% of the dynamic range of the card, even in cases where the dynamic range is artificially extended. The histograms of the errors of all the dynamic range cases are shown for comparison purposes in Figure 4.6.

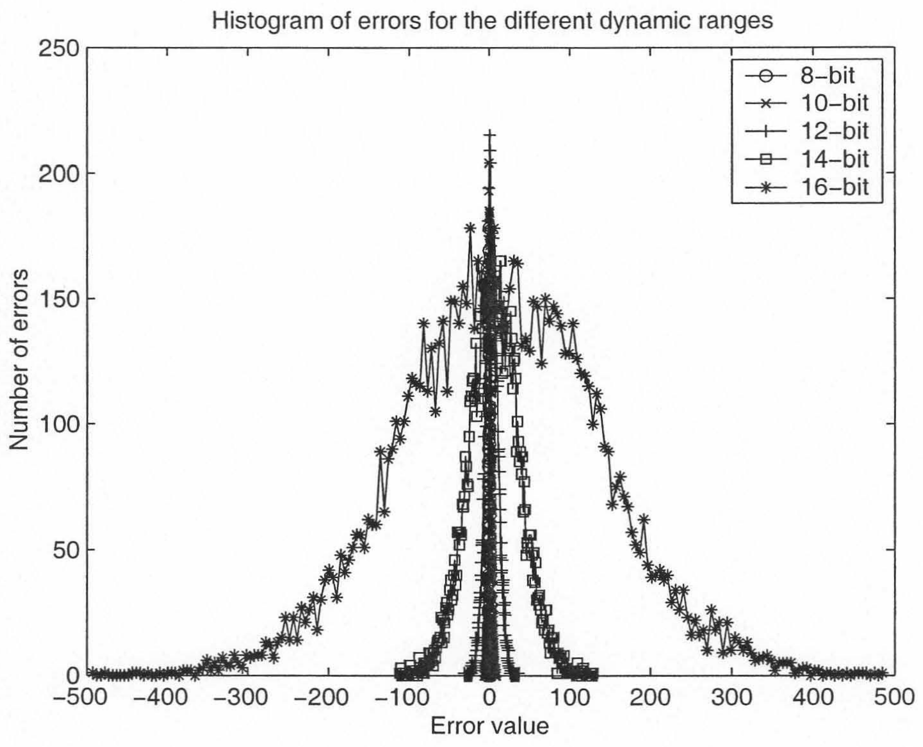

Figure 4.6: Histogram of the errors for different dynamic ranges

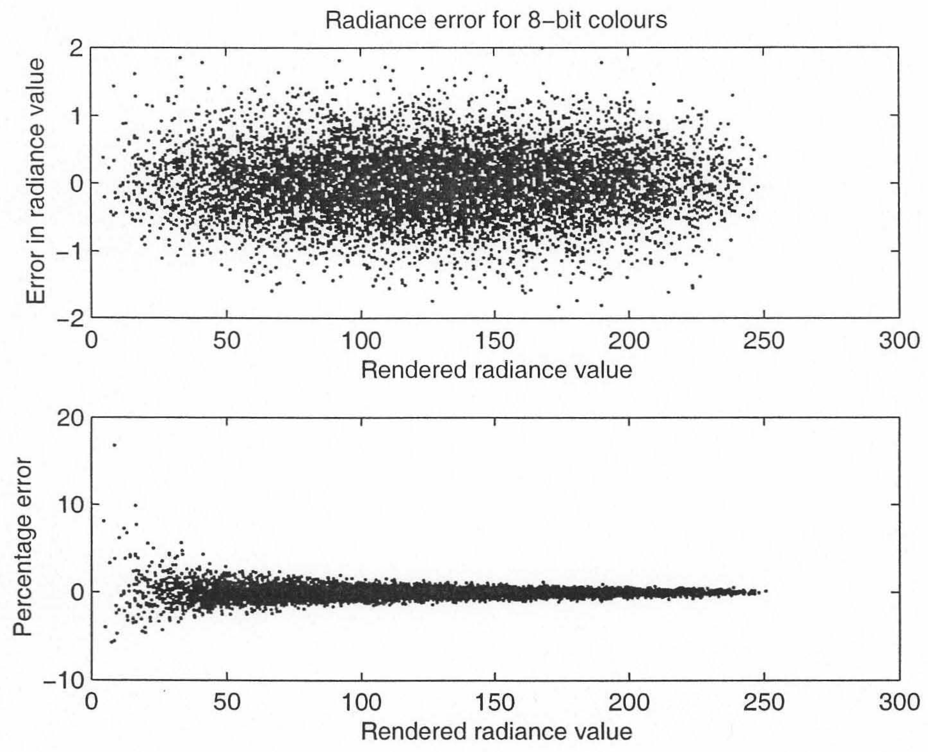

Figure 4.7: Difference between calculated and rendered radiance values - 8bit

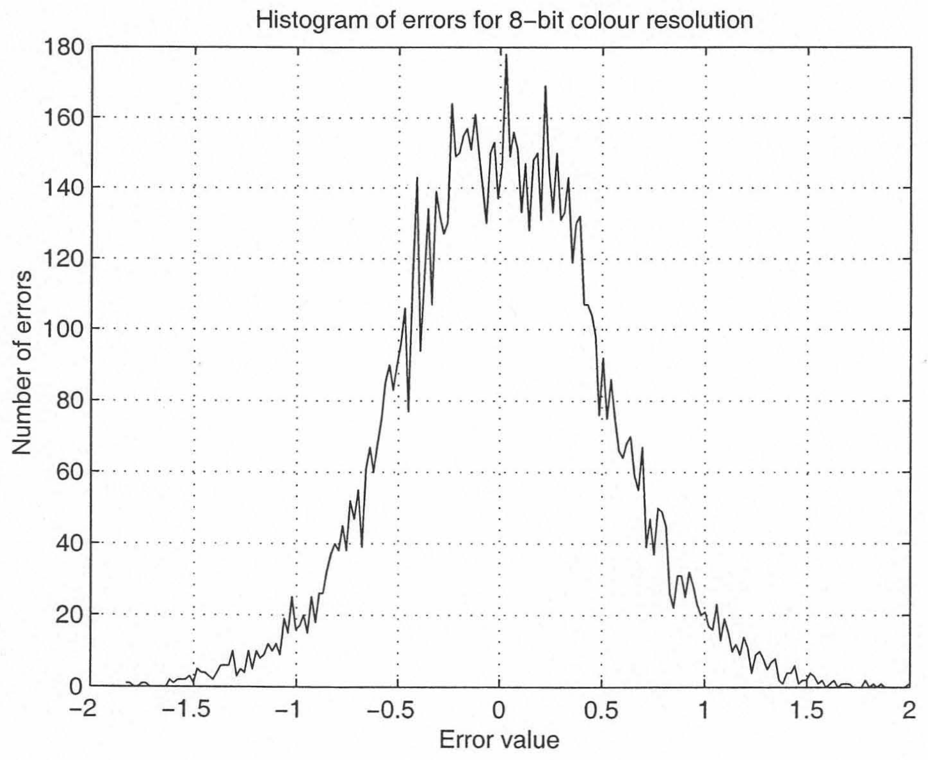

Figure 4.8: Histogram of difference between calculated and rendered radiance values - 8bit

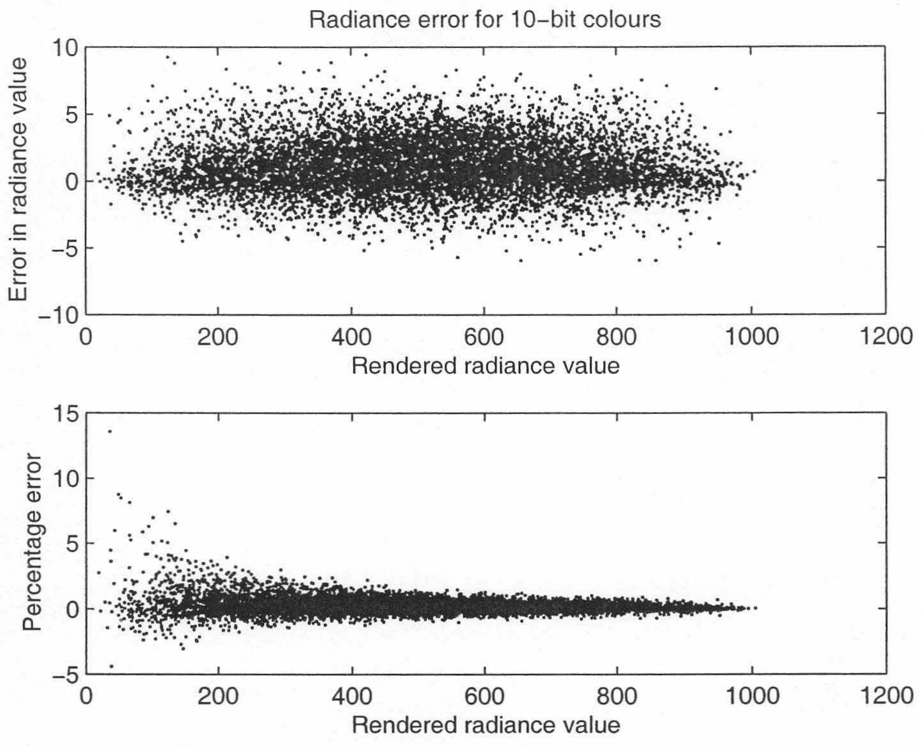

Figure 4.9: Difference between calculated and rendered radiance values - 10bit

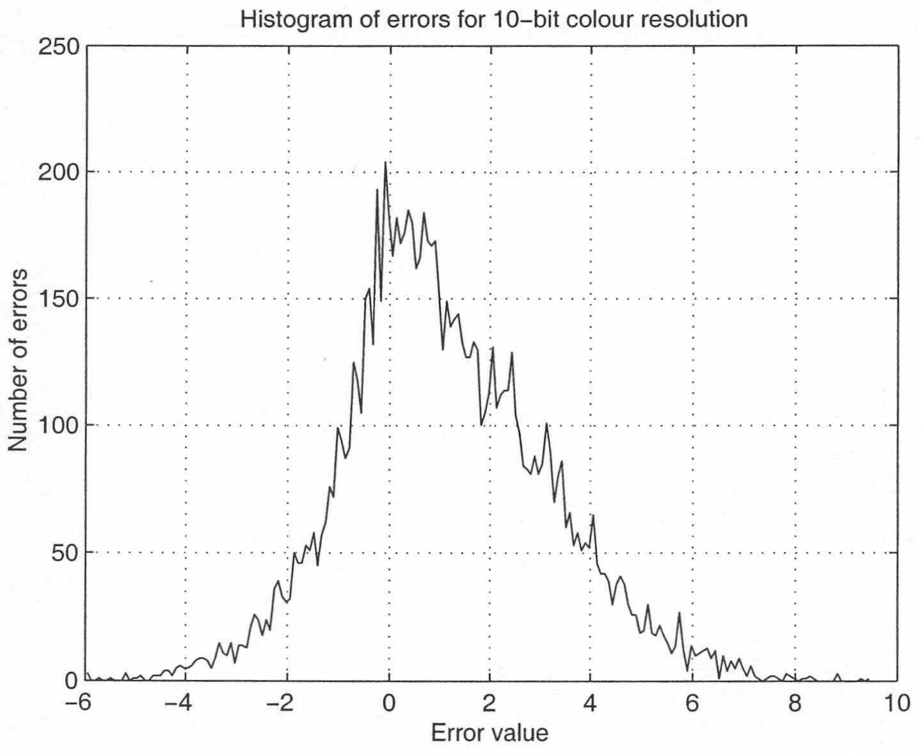

Figure 4.10: Histogram of difference between calculated and rendered radiance values - 10bit

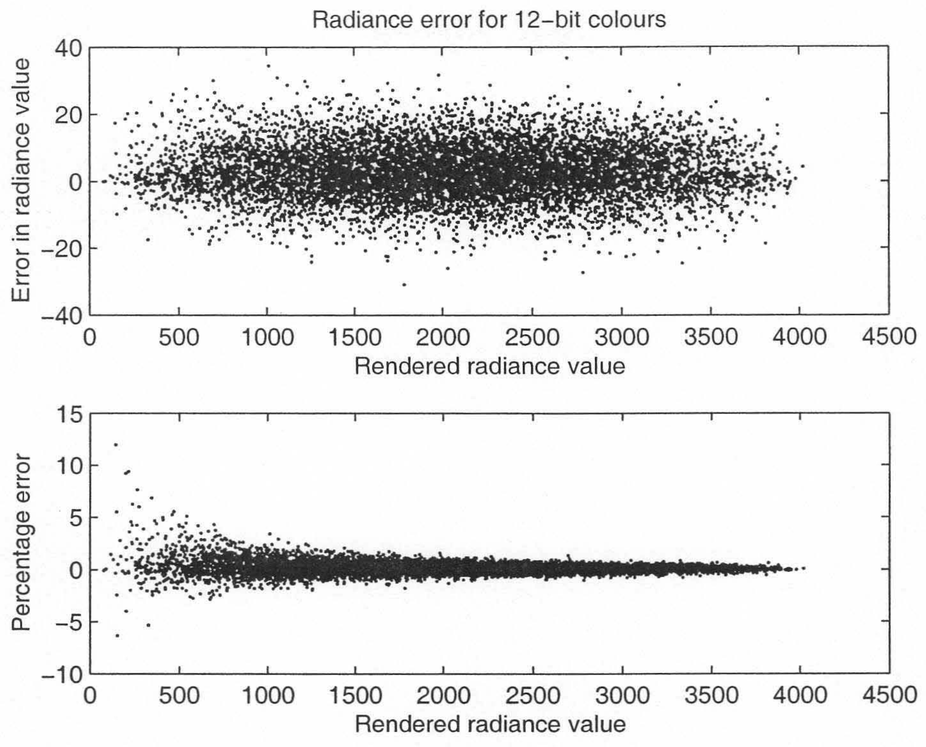

Figure 4.11: Difference between calculated and rendered radiance values - 12bit

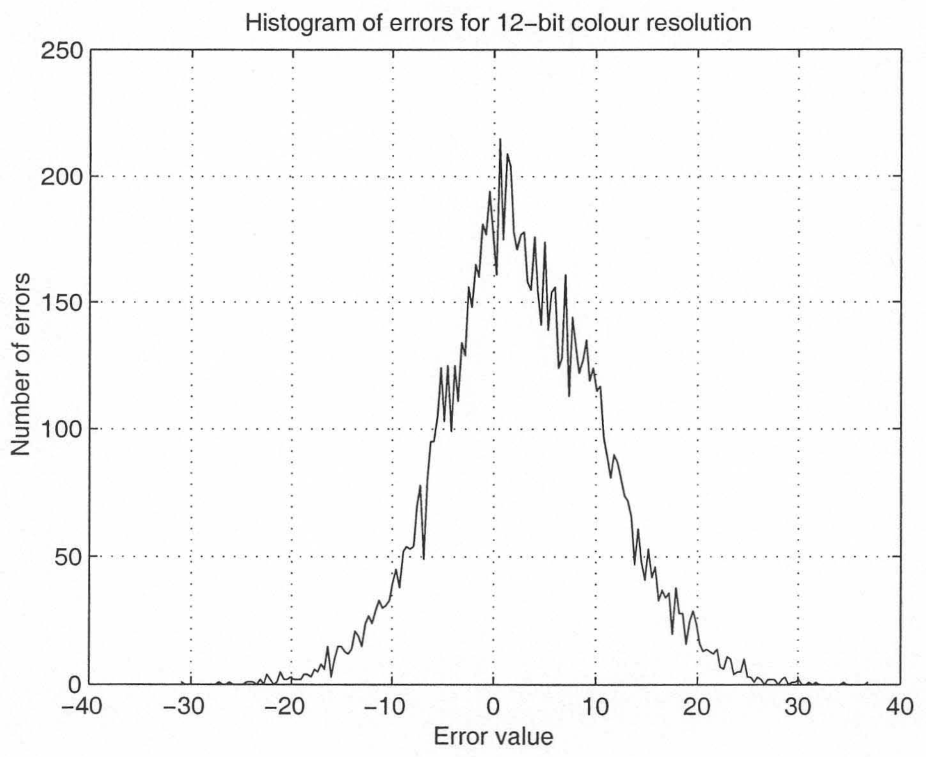

Figure 4.12: Histogram of difference between calculated and rendered radiance values - 12bit

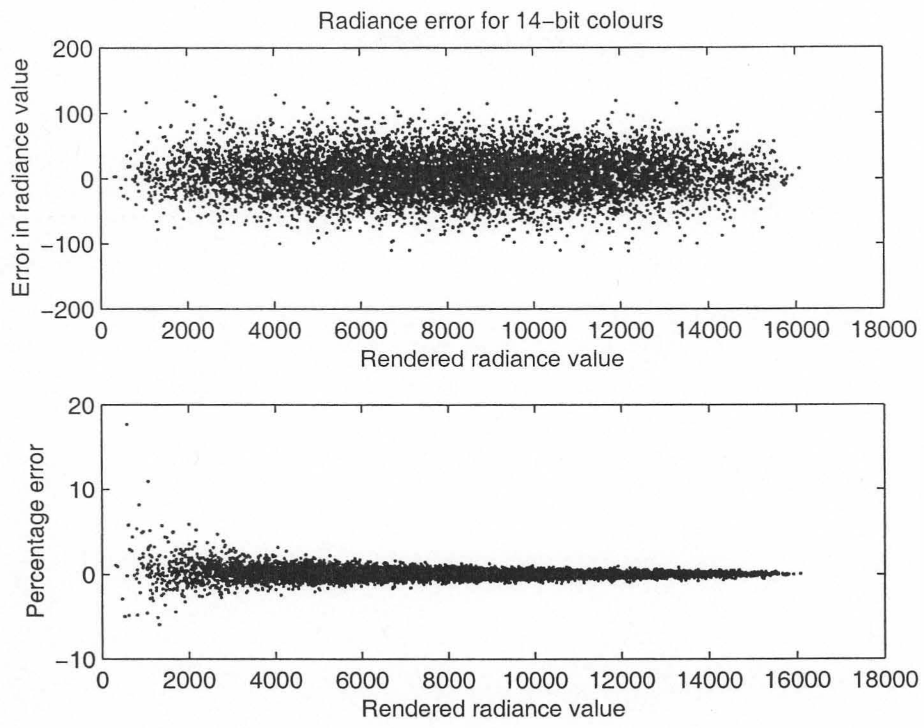

Figure 4.13: Difference between calculated and rendered radiance values - 14bit

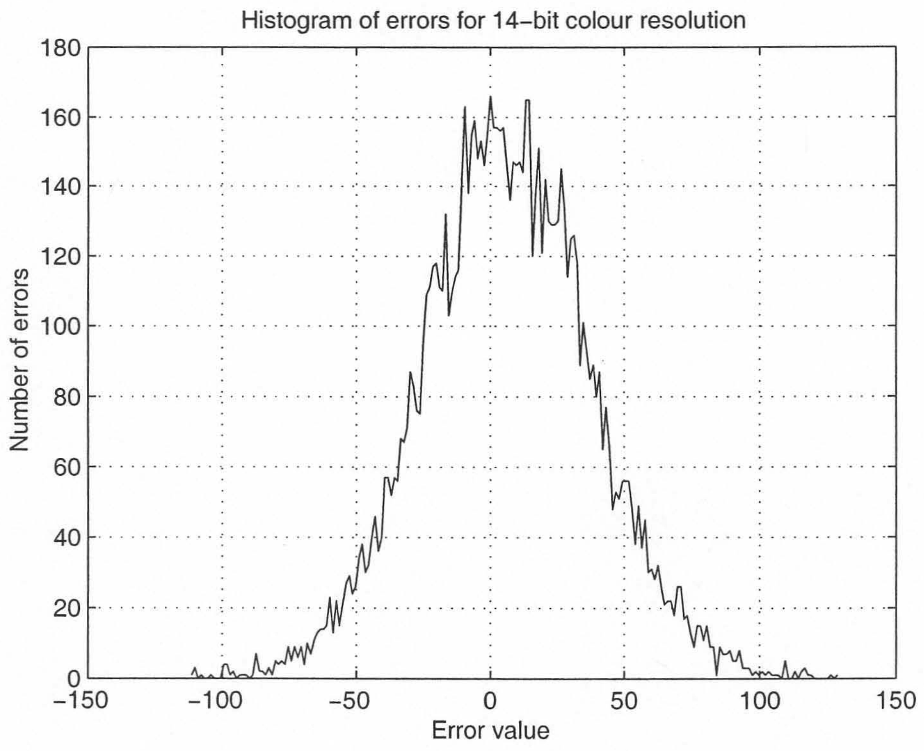

Figure 4.14: Histogram of difference between calculated and rendered radiance values - 14bit

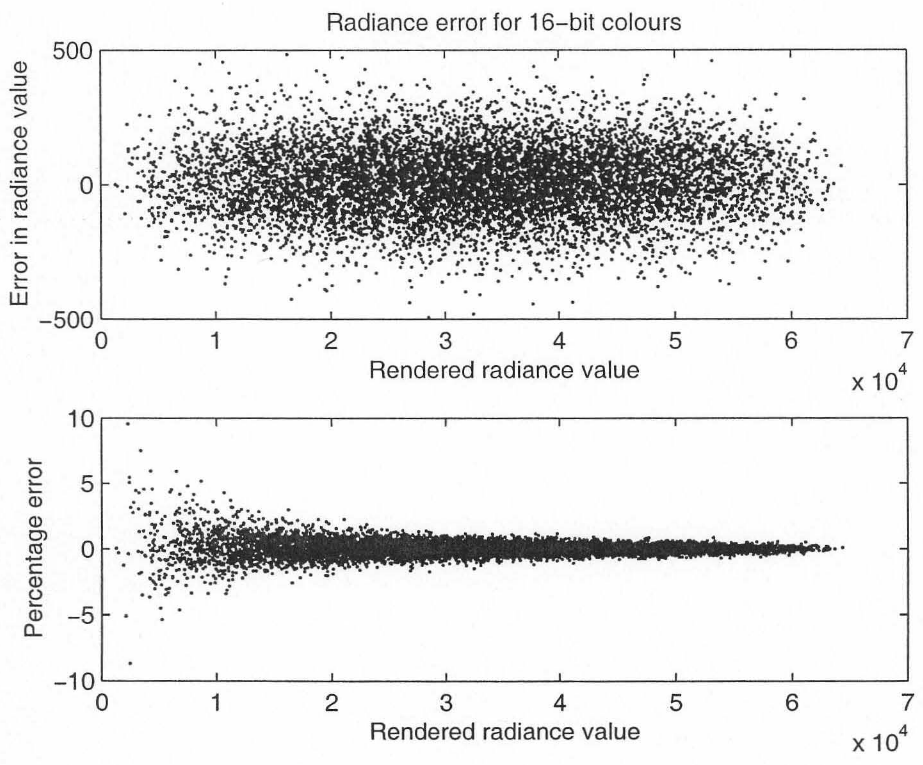

Figure 4.15: Difference between calculated and rendered radiance values - 16bit

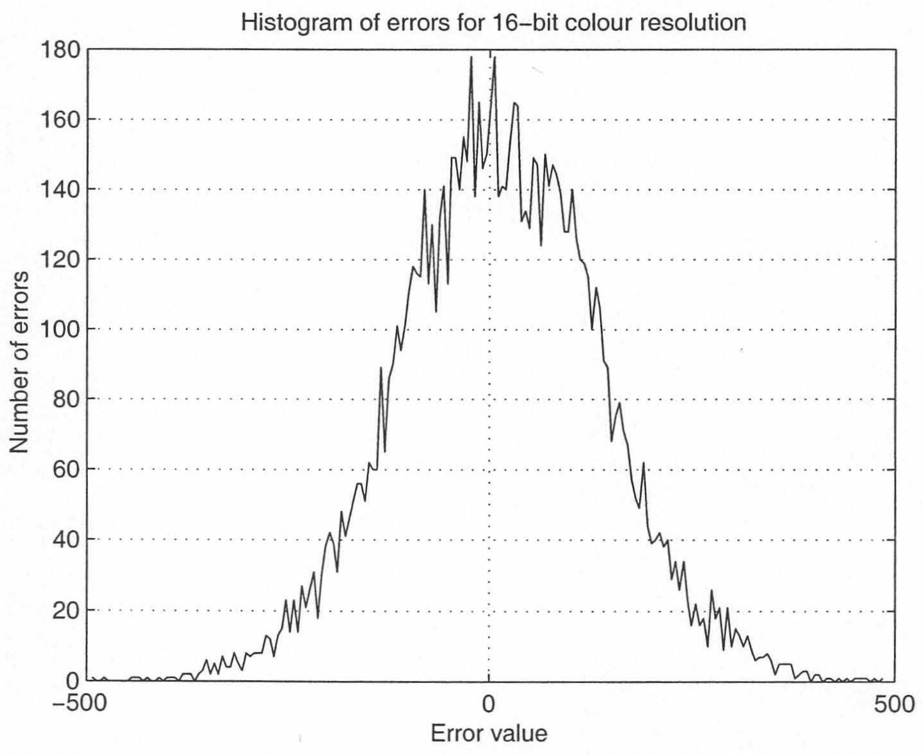

Figure 4.16: Histogram of difference between calculated and rendered radiance values - 16bit

### 4.4.2 Determining the error in intensity at increasing distances

The object was assigned the input radiance values shown in Table 4.2 for the second experiment. The intensity of the object was calculated using Equation (4.5) and the intensity was determined from the rendered image for different distances between the object and the sensor.

Table 4.2: Input values for the calculation of intensity

| Resolution (bits) | Radiance of vertex 1 ($Wm^{-2}sr^{-1}$) | Radiance of vertex 2 ($Wm^{-2}sr^{-1}$) | Radiance of vertex 3 ($Wm^{-2}sr^{-1}$) | Calculated Intensity ($Wsr^{-1}$) |
|---|---|---|---|---|
| 8 | 105 | 250 | 96 | 300.667 |
| 10 | 105 | 250 | 968 | 882 |
| 12 | 1050 | 3250 | 368 | 3112 |
| 14 | 1050 | 250 | 12048 | 8898.667 |
| 16 | 1050 | 5250 | 36840 | 28760 |

Figures 4.17 to 4.21 show the rendered intensity as a function of the distance between the source and the sensor. The calculated intensity is also shown in each of the graphs. A summary of the errors is shown in Table 4.3.

The object was rendered every 10m, from 100m to 2100m. The field of view used in the experiment was 5°and the screen resolution was 512×512 pixels. The data shows a similar sawtooth trend in all the cases. There is therefore a range of distances, even at large distances, where the rendered intensity is within 1% of the actual intensity. It is therefore not possible to conclude that the intensity calculation can only be used up to a specific distance. The intensity at short distances is within 2% of the actual value in all the cases. The technique can be used to determine the intensity, with less than a 2% error, of objects at distances up to 200m from the sensor for the set of input parameters used in this experiment. A smaller pixel field of view would increase this distance and a larger pixel field of view would reduce it. The saw-tooth shape of the data might be due to a sampling effect, where the number of pixels filled with the test object varies as the distance between the observer is increased.

Table 4.3: Errors in calculating the intensity at different distances

| Resolution (bits) | Maximum error ($Wsr^{-1}$) | Maximum error (%) |
|---|---|---|
| 8 | 46.6 | 15.5 |
| 10 | 132.54 | 15 |
| 12 | 519.7 | 16.7 |
| 14 | 1251 | 14.1 |
| 16 | 4051 | 14.1 |

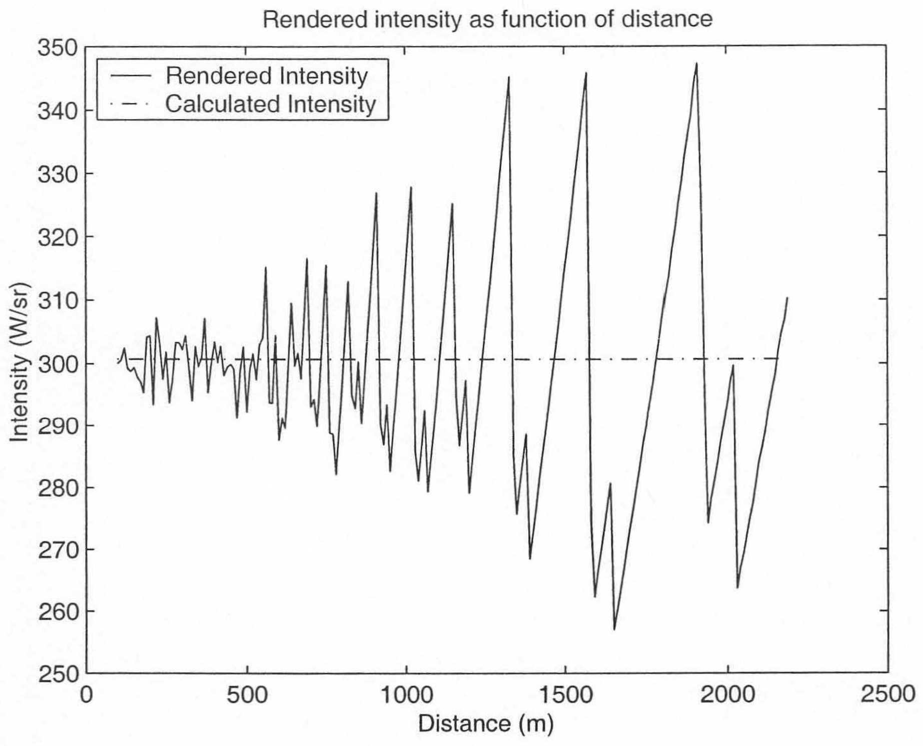

Figure 4.17: Rendered intensity values as function of distance - 8bit

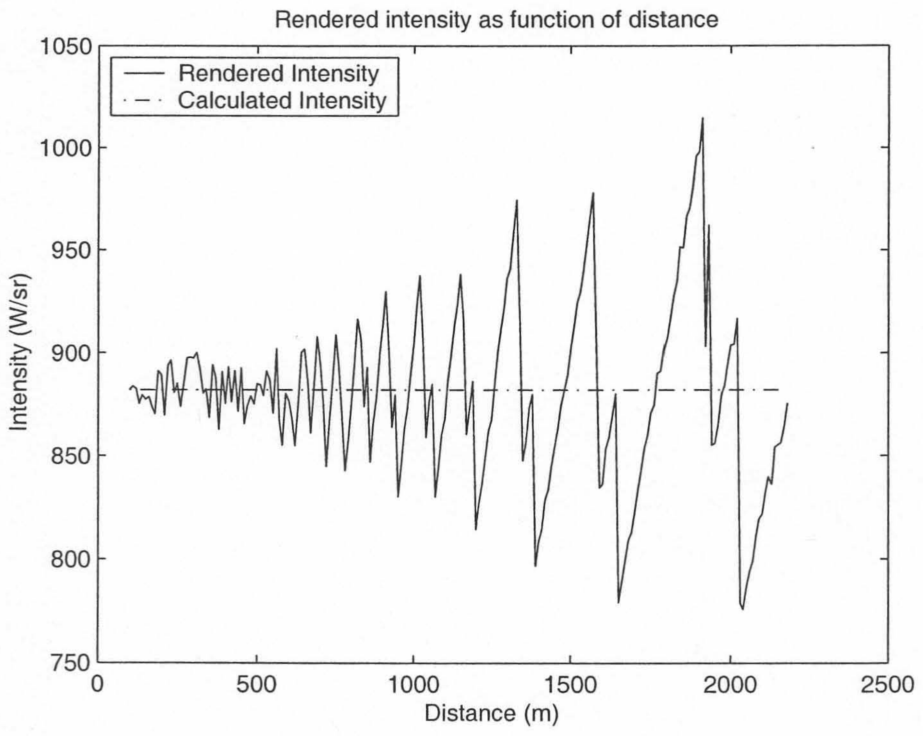

Figure 4.18: Rendered intensity values as function of distance - 10bit

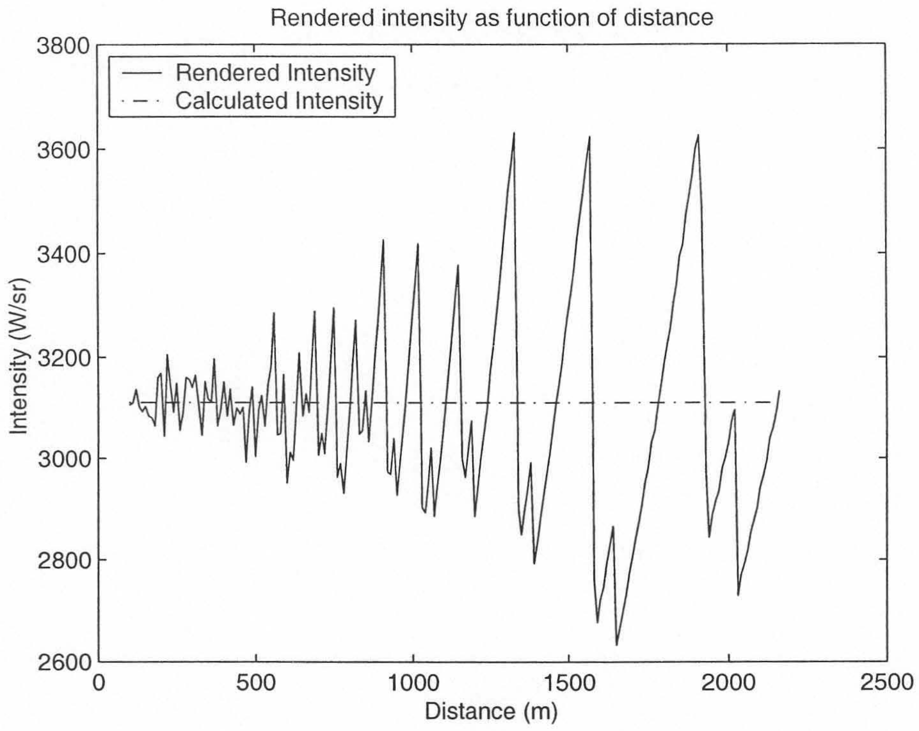

Figure 4.19: Rendered intensity values as function of distance - 12bit

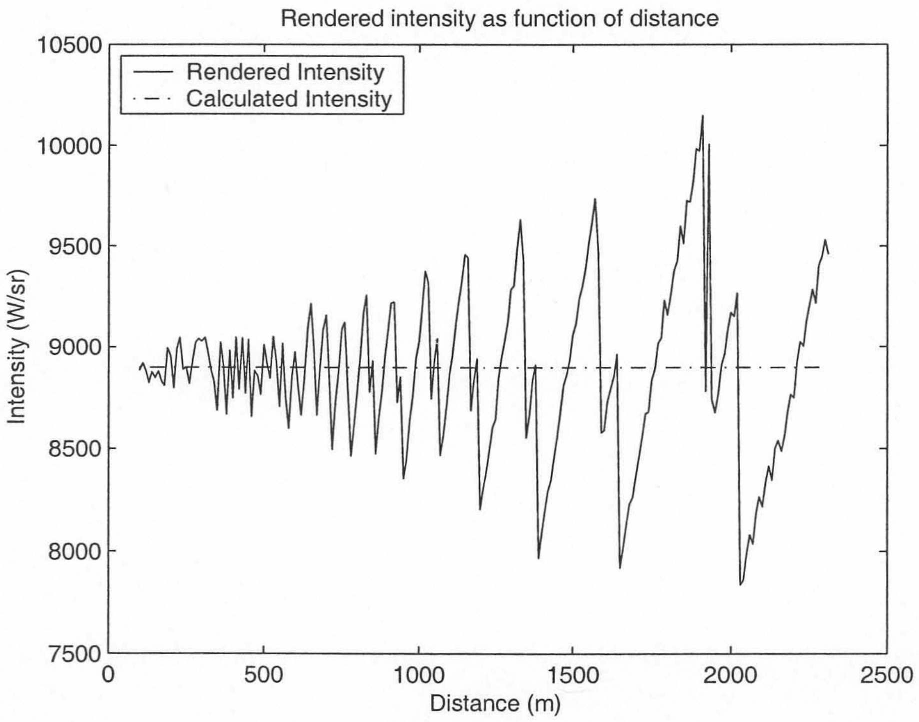

Figure 4.20: Rendered intensity values as function of distance - 14bit

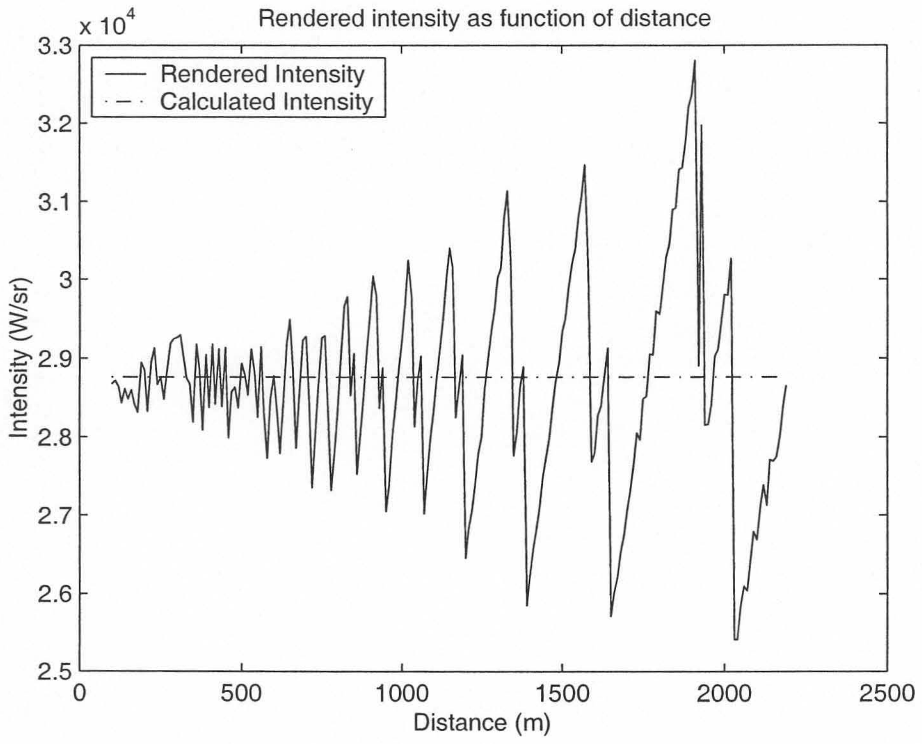

Figure 4.21: Rendered intensity values as function of distance - 16bit

# Chapter 5

# Single Parameter Equivalents of Radiometric Entities

## 5.1 Introduction

Radiometric entities such as radiance and atmospheric transmittance can be highly variable with wavelength. It is therefore not always accurate to replace such a spectrally variant entity with a single parameter that is required by OpenGL. The purpose of this chapter is to investigate the mapping of atmospheric transmittance, path radiance and sources with constant emissivity to single parameters that can be used in OpenGL.

The MODTRAN atmospheric transmittance model was used to generate atmospheric transmittance data at selected distances from 100m to 15km. The input parameters were selected to represent good, average and bad atmospheric transmittance conditions. The data was generated at a wavenumber increment of 2 (2 cm$^{-1}$). An example of atmospheric transmittance versus wavelength is shown in Figure 5.2.

## 5.2 Atmospheric transmittance

Effective atmospheric transmittance is determined by calculating the irradiance on the detector taking the atmospheric transmittance into account and dividing it with the irradiance calculated without the atmospheric transmittance, as shown in Equation (5.1). Equation (5.1) was adapted from Holst [21, p287].

$$\tau_{eff} = \frac{\int_{\lambda 1}^{\lambda 2} L(\lambda)\tau_{system}(\lambda)\tau_{atmosphere}(\lambda)d\lambda}{\int_{\lambda 1}^{\lambda 2} L(\lambda)\tau_{system}(\lambda)d\lambda}. \tag{5.1}$$

$L_\lambda$ is the spectral radiance of the source, $\tau_{system}(\lambda)$ is the system's spectral response and $\tau_{atmosphere}(\lambda)$ is the spectral atmospheric transmittance.

The effective atmospheric transmittance as function of distance were determined for three different spectral filters, namely 3.9-4.1 $\mu$m, 3.9-5 $\mu$m and 3-5 $\mu$m. The effective transmittance and the position of the filter in the atmospheric window are shown in Figure 5.1 to Figure

5.6. The source was a blackbody at 920°C. The filter bandwidths were selected to represent transmittance in a single atmospheric window (3.9-4.1$\mu$m), transmittance in a window including a strong absorption band (3.9-5$\mu$m), and transmittance in a wide band that includes transmittance windows, an absorption band and line absorption bands (3-5$\mu$m).

The equivalent effective transmittance was determined by fitting a line to the log of the effective transmittance curve using:

$$\alpha = \frac{\sum_i x_i log(\tau_i)}{\sum_i x_i^2}. \tag{5.2}$$

The equivalent effective transmittance is a single parameter equation that can be used to calculate the atmospheric transmittance at a distance $r$ by using

$$\tau = e^{\alpha r}, \tag{5.3}$$

with $\alpha$ the value determined in Equation (5.2). The technique was tested for the three filter bandwidths mentioned above. The results are shown in Figure 5.7 to Figure 5.12. The different figures are:

- Figure 5.7 shows a comparison between the effective transmittance that was calculated using Equation (5.1) and the single parameter equivalent that was calculated using Equation (5.2). Figure 5.8 shows the difference between the effective transmittance and the single parameter equivalent. The data was calculated for a 3.9-4.1 $\mu$m filter.
- Figure 5.9 shows a comparison between the effective transmittance that was calculated using Equation (5.1) and the single parameter equivalent that was calculated using Equation (5.2). Figure 5.10 shows the difference between the effective transmittance and the single parameter equivalent. The data was calculated for a 3.9-5 $\mu$m filter.
- Figure 5.11 shows a comparison between the effective transmittance that was calculated using Equation (5.1) and the single parameter equivalent that was calculated using Equation (5.2). Figure 5.12 shows the difference between the effective transmittance and the single parameter equivalent. The data was calculated for a 3-5 $\mu$m filter.

In Figure 5.7 to Figure 5.12 $\tau$ is used to indicate atmospheric transmittance. The atmospheric transmittance curves shown in Figures 5.2, 5.4 and 5.6 were calculated for a path length of 1km.

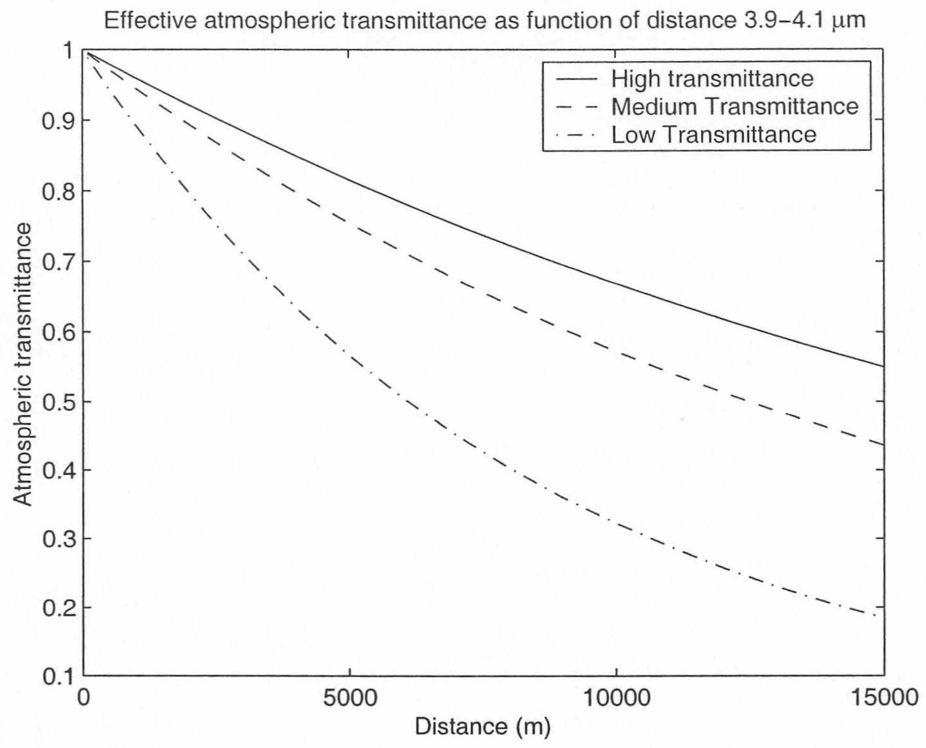

Figure 5.1: Effective atmospheric transmittance for a 3.9-4.1 $\mu$m system response

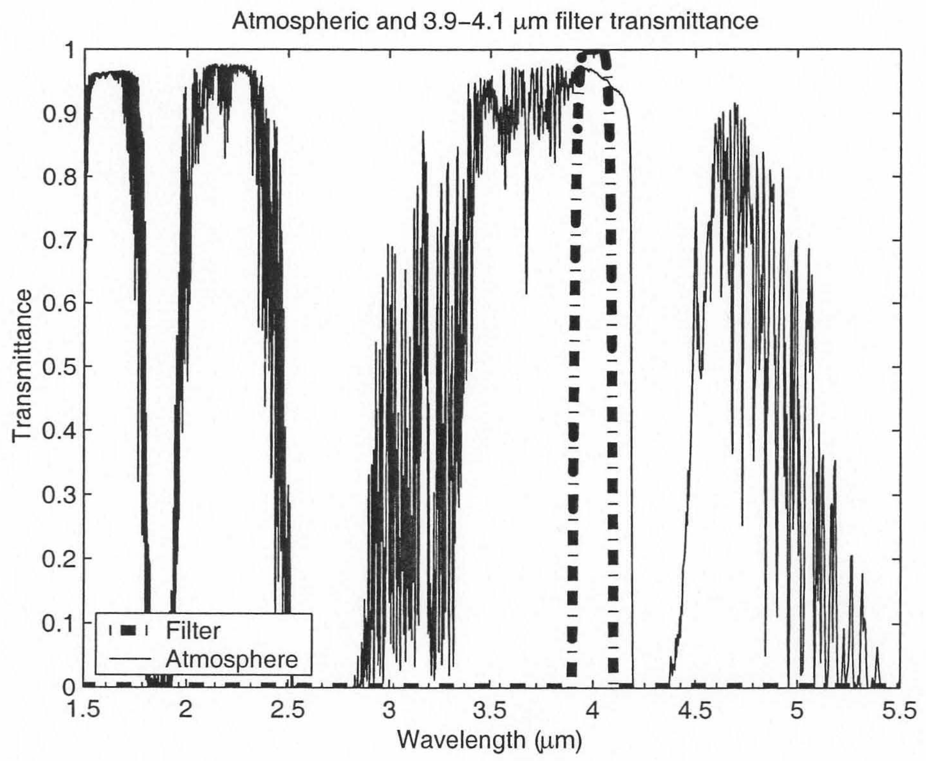

Figure 5.2: Position of the 3.9-4.1 $\mu$m bandpass filter

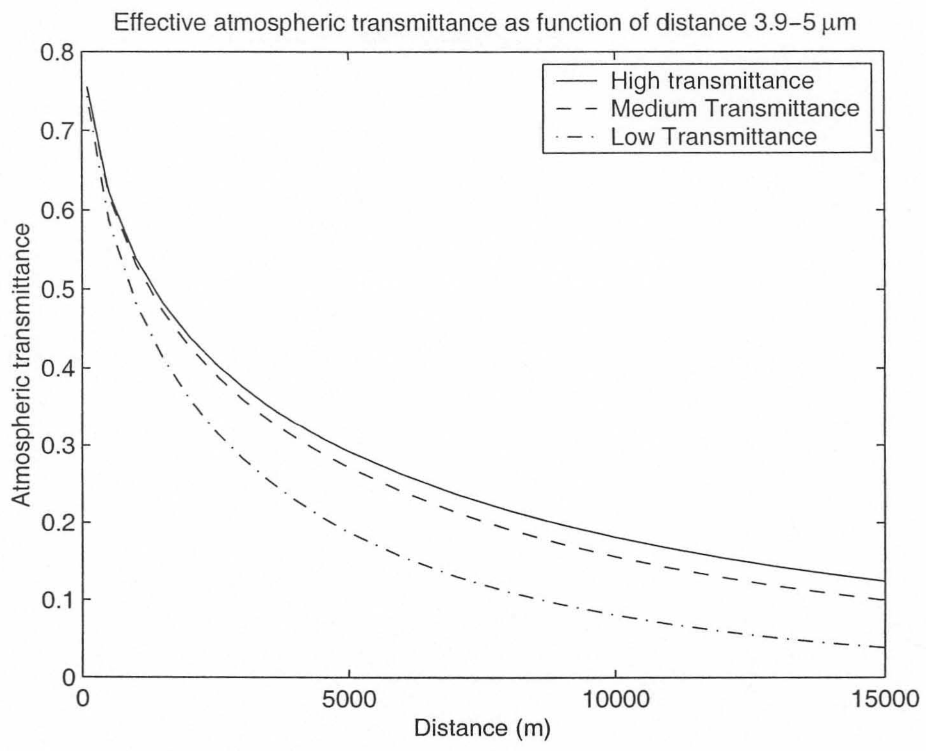

Figure 5.3: Effective atmospheric transmittance for a 3.9-5 $\mu$m system response

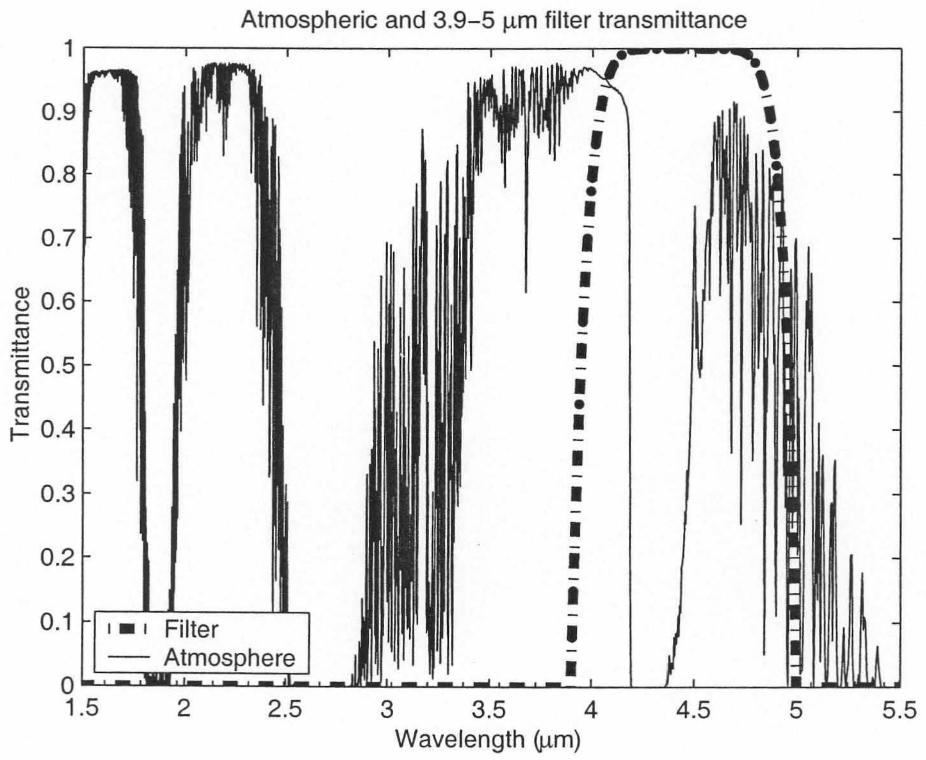

Figure 5.4: Position of the 3.9-5 $\mu$m bandpass filter

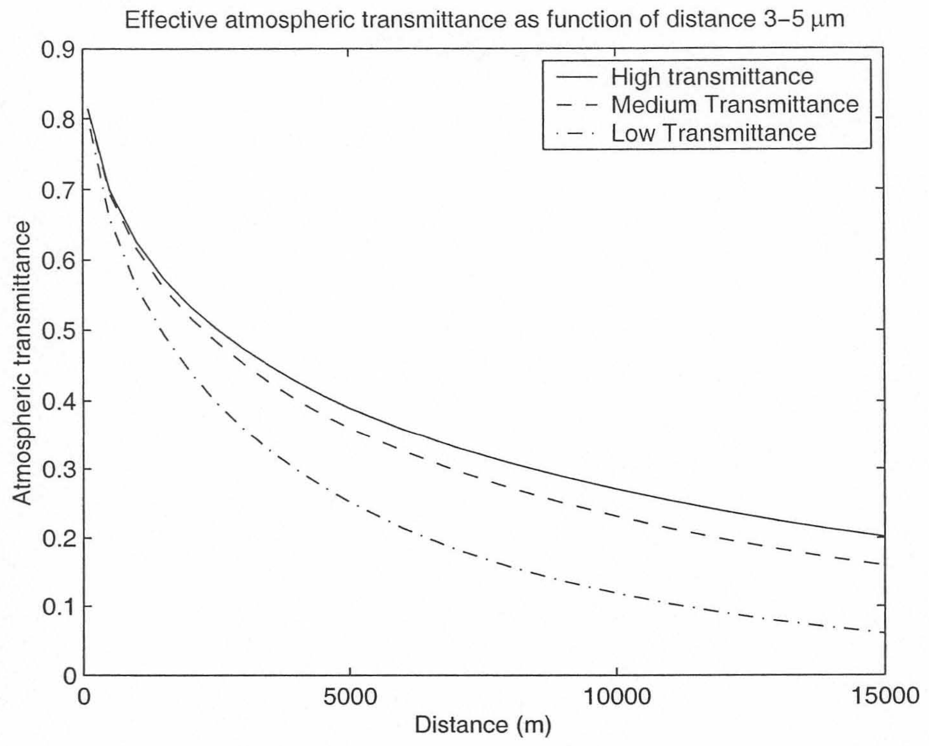

Figure 5.5: Effective atmospheric transmittance for a 3-5 $\mu$m system response

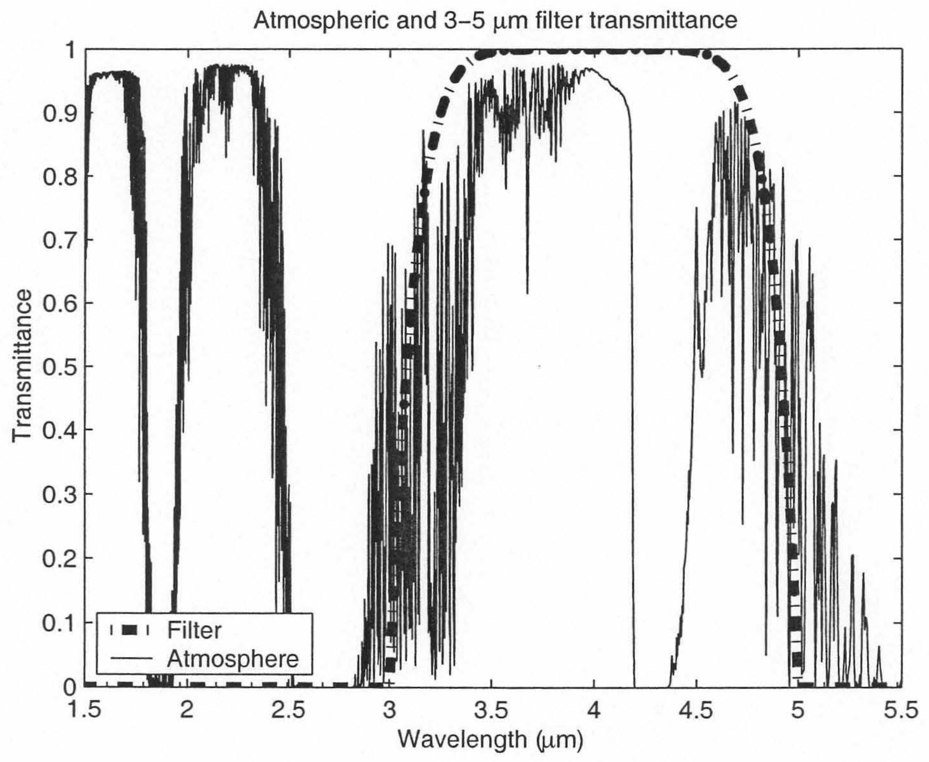

Figure 5.6: Position of the 3-5 $\mu$m bandpass filter

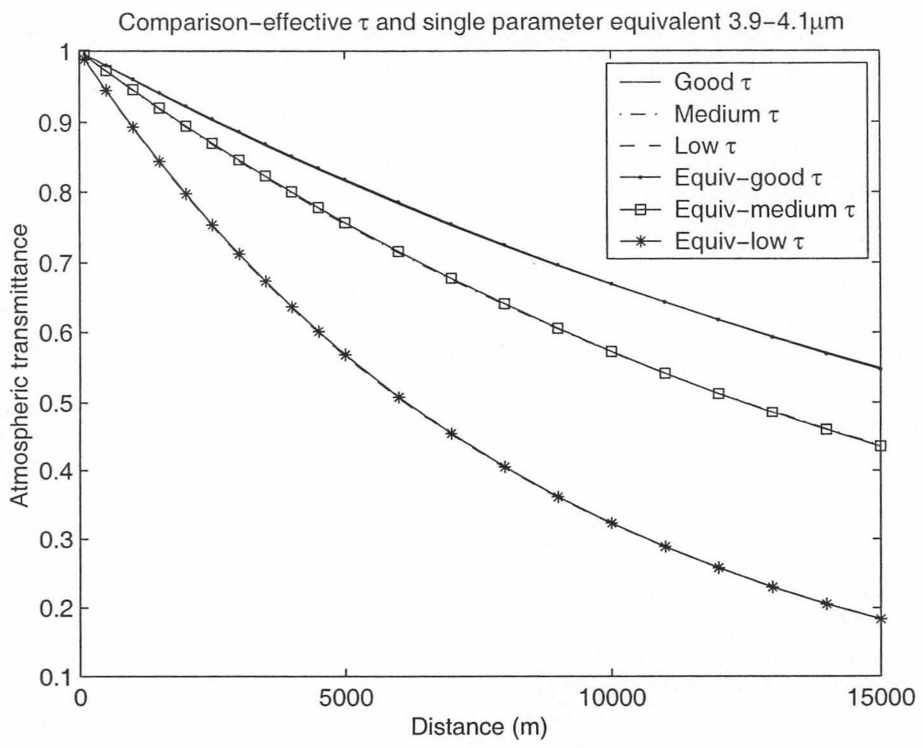

Figure 5.7: Atmospheric transmittance and equivalent transmittance - 3.9-4.1$\mu$m

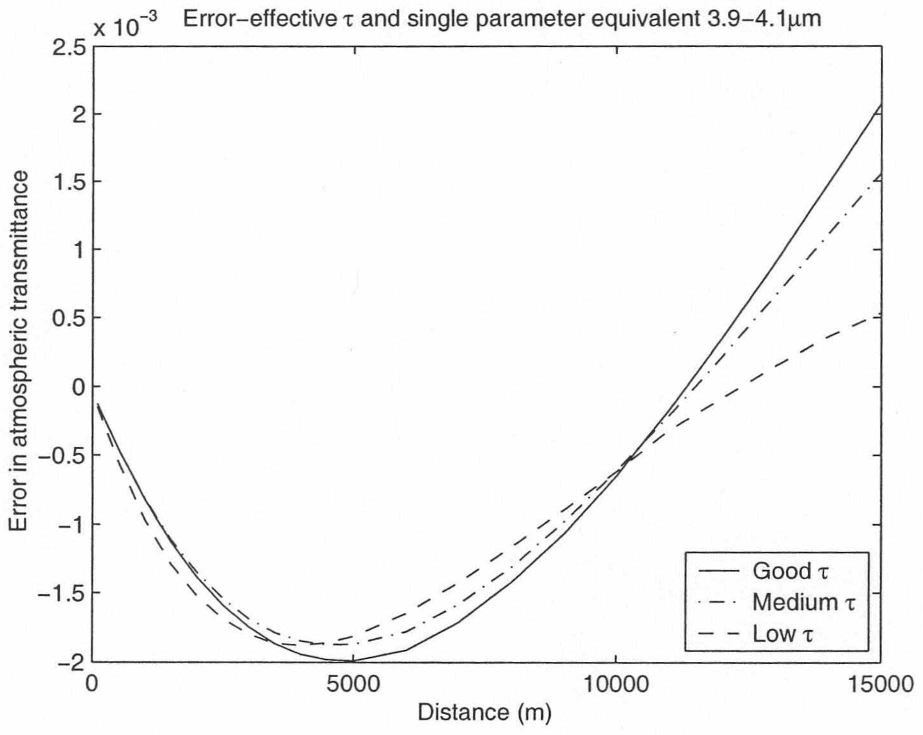

Figure 5.8: Error between actual and equivalent atmospheric transmittance - 3.9-4.1$\mu$m

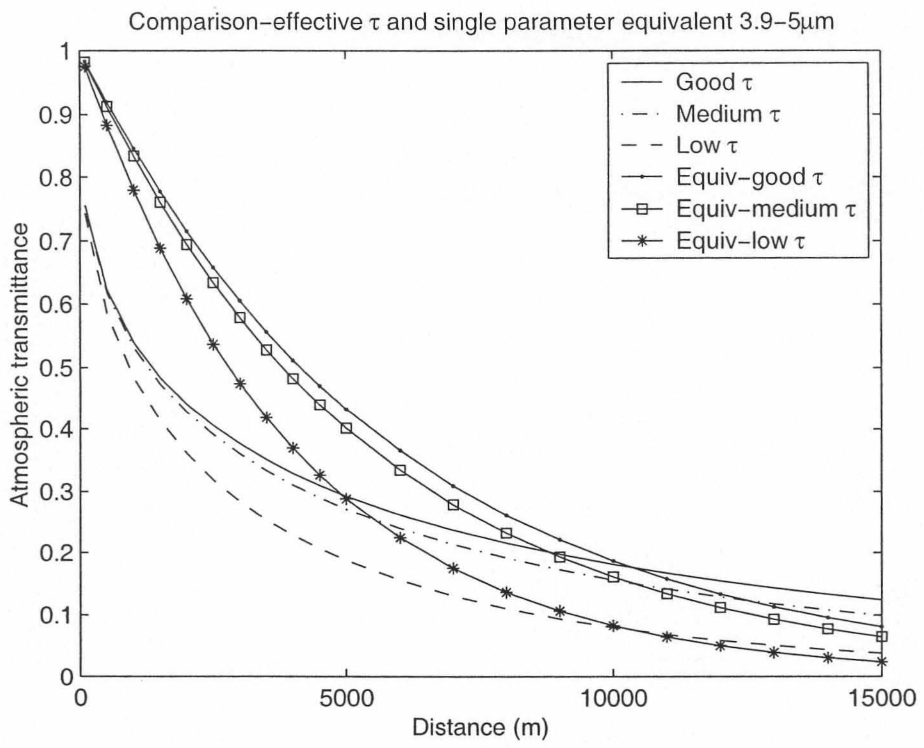

Figure 5.9: Atmospheric transmittance and equivalent transmittance - 3.9-5$\mu$m

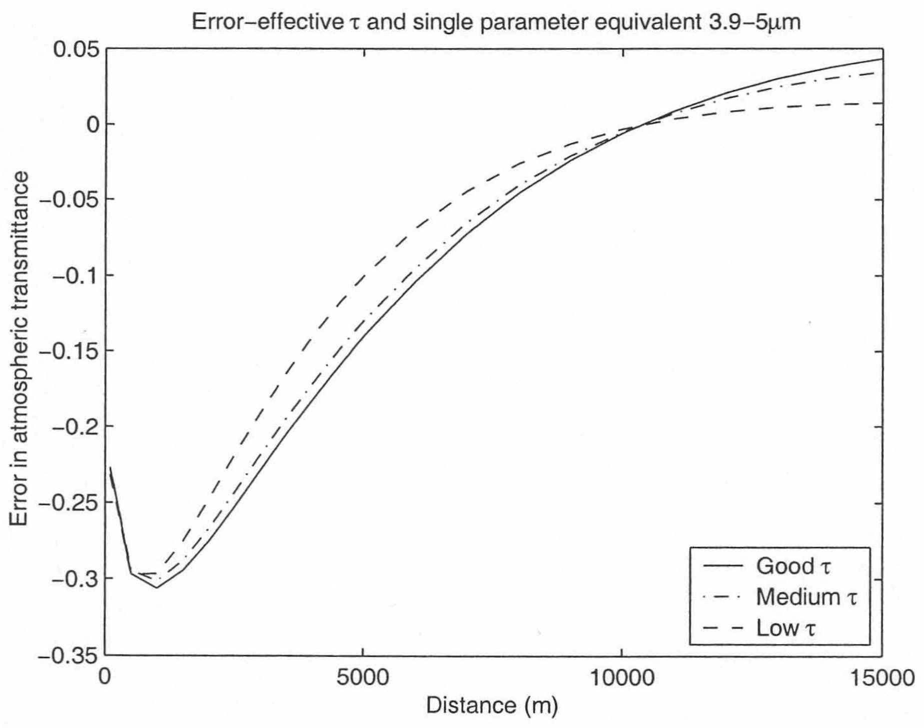

Figure 5.10: Error between actual and equivalent atmospheric transmittance - 3.9-5$\mu$m

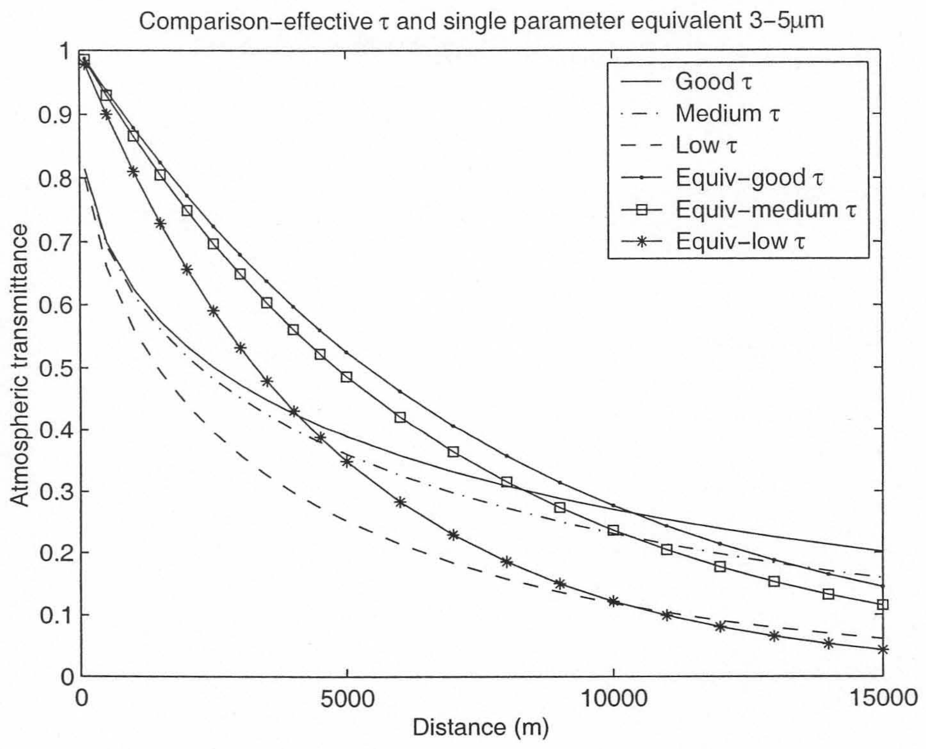

Figure 5.11: Atmospheric transmittance and equivalent transmittance - 3-5$\mu$m

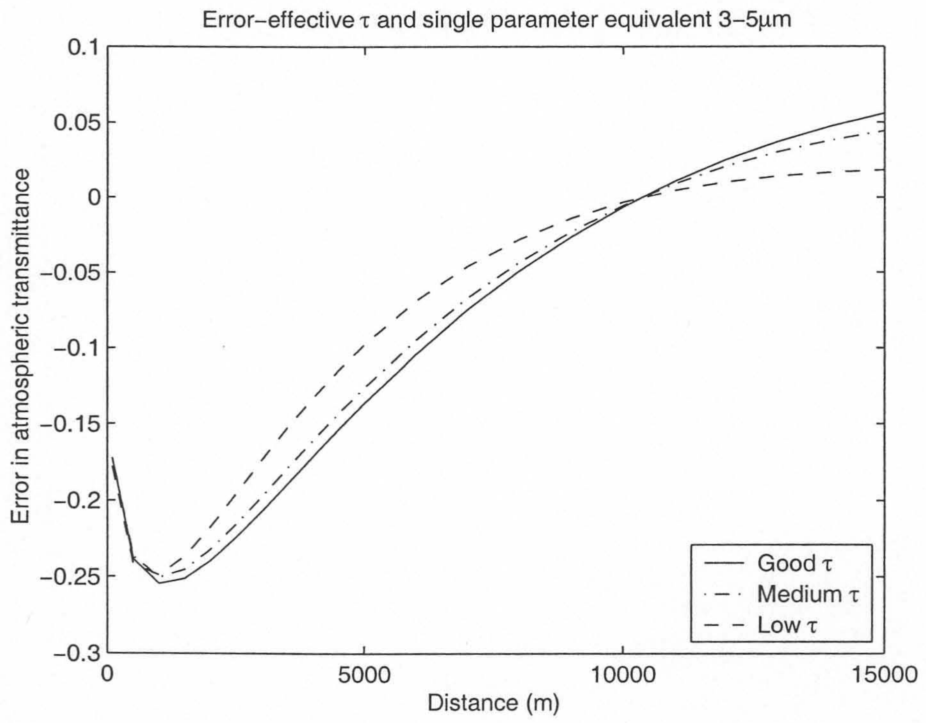

Figure 5.12: Error between actual and equivalent atmospheric transmittance - 3-5$\mu$m

## 5.3 Polynomial model of the effective atmospheric transmittance

It is possible to model the transmittance in the narrow band case, shown in Figure 5.7, with a low error, as shown in Figure 5.8, using an equation of the form $e^{-\alpha r}$, with $r$ the distance and $\alpha$ the extinction coefficient. The two wider band cases do not model well with a single exponential parameter. A higher order polynomial model was investigated to map the transmittance in the wide band cases. The higher order model cannot be used directly in OpenGL, because OpenGL uses a single input value for the extinction coefficient, but is useful in cases where the atmospheric transmittance is modeled as a second step in the image generation. The results shown in Figure 5.13 to Figure 5.16 were modeled with an equation of the form

$$\tau = e^{-\alpha_1 r^2 - \alpha_2 r - \alpha_3}, \tag{5.4}$$

with $r$ the distance, and $\alpha_1$, $\alpha_2$ and $\alpha_3$ the coefficients of a polynomial fitted to the logarithm of the transmittance data. The advantage of using a higher order polynomial instead of just a single parameter is shown in Table 5.1.

Table 5.1: Reduction in error after using a higher order polynomial

| Spectral band of the system | Atmospheric Transmittance | Maximum Error - single parameter (%) | Average of the absolute value of the error - single parameter (%) | Maximum Error - polynomial (%) | Average of the absolute value of the error - polynomial (%) |
|---|---|---|---|---|---|
| 3.9-5 $\mu$m | Good | -31.27 | 14.62 | 14.52 | 1.45 |
| 3.9-5 $\mu$m | Medium | -30.79 | 13.93 | 14.48 | 1.41 |
| 3.9-5 $\mu$m | Low | -30.46 | 12.19 | 15.42 | 1.40 |
| 3-5 $\mu$m | Good | -23.95 | 12.24 | 14.55 | 1.52 |
| 3-5 $\mu$m | Medium | -23.58 | 11.58 | 14.53 | 1.48 |
| 3-5 $\mu$m | Low | -23.90 | 10.20 | 15.91 | 1.48 |

The use of the polynomial model led to up to a 53% reduction in the error when calculating the equivalent atmospheric transmittance. This result is useful in cases where the atmospheric transmittance is calculated as a separate step in a simulation. It is not possible to map the polynomial approximation to the fog density parameter in OpenGL. The calculation of the atmospheric transmittance in an OpenGL-based simulation would require careful optimisation of the atmospheric model at the range of distances of interest in the simulation.

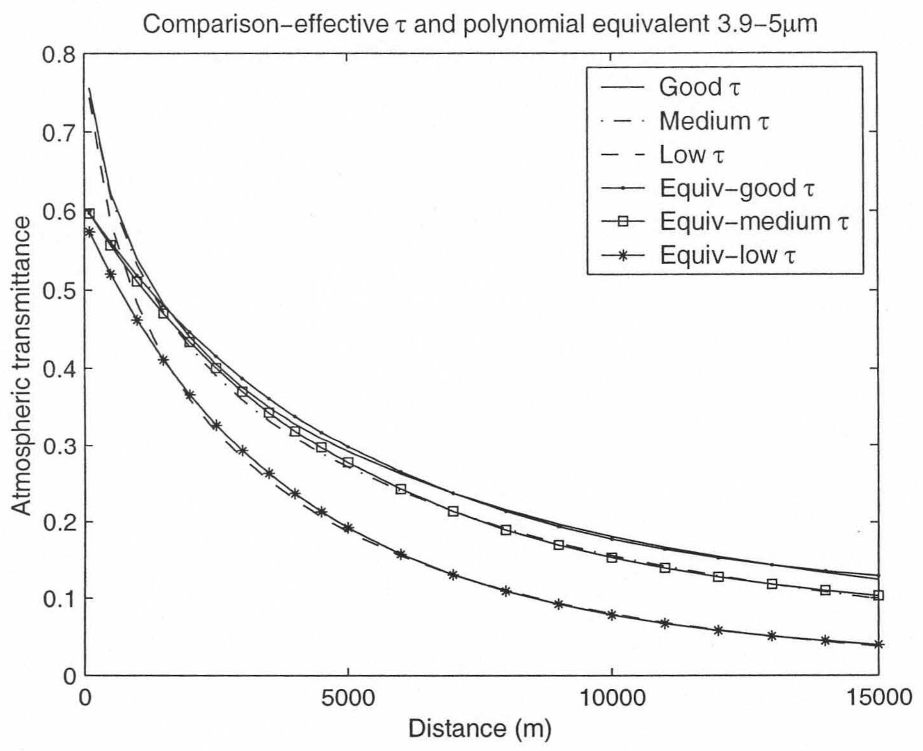

Figure 5.13: Atmospheric transmittance and equivalent transmittance - 3.9-5$\mu$m - polynomial fit

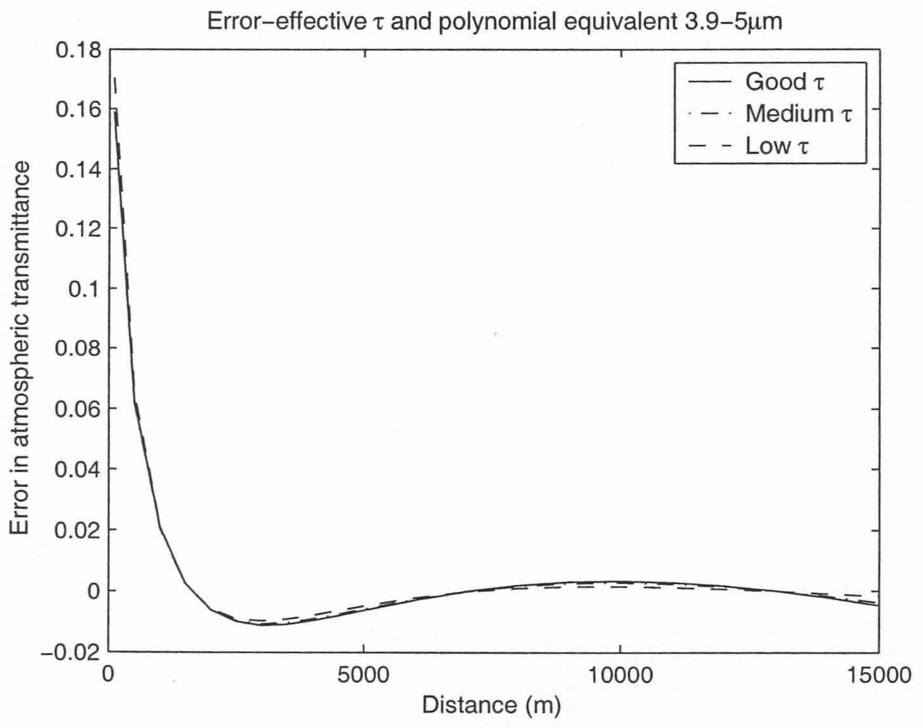

Figure 5.14: Error between actual and equivalent atmospheric transmittance - 3.9-5$\mu$m

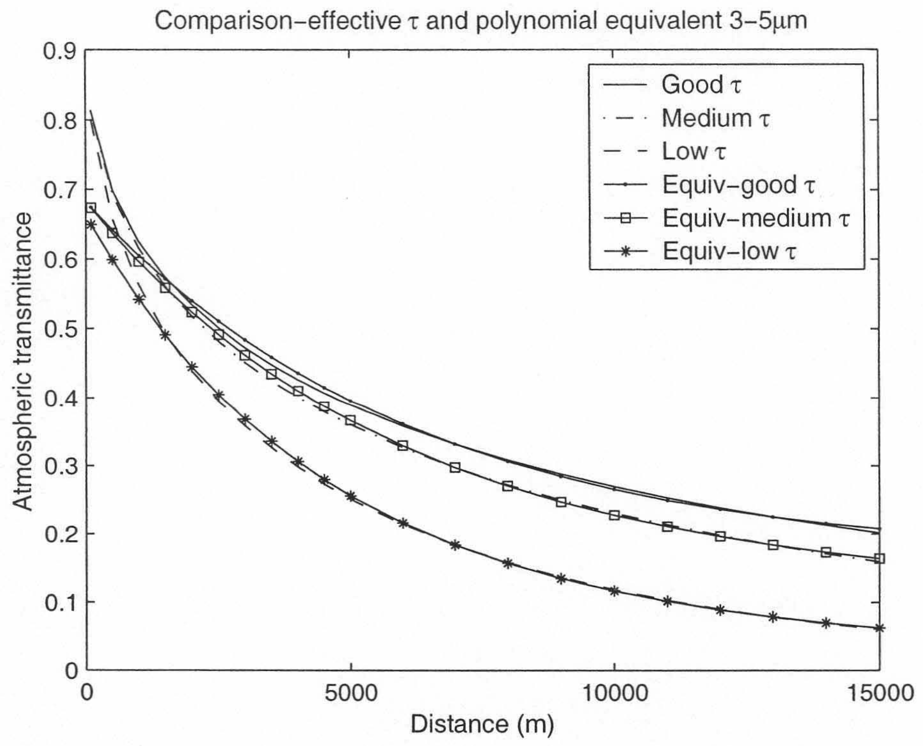

Figure 5.15: Atmospheric transmittance and equivalent transmittance - 3-5$\mu$m - polynomial fit

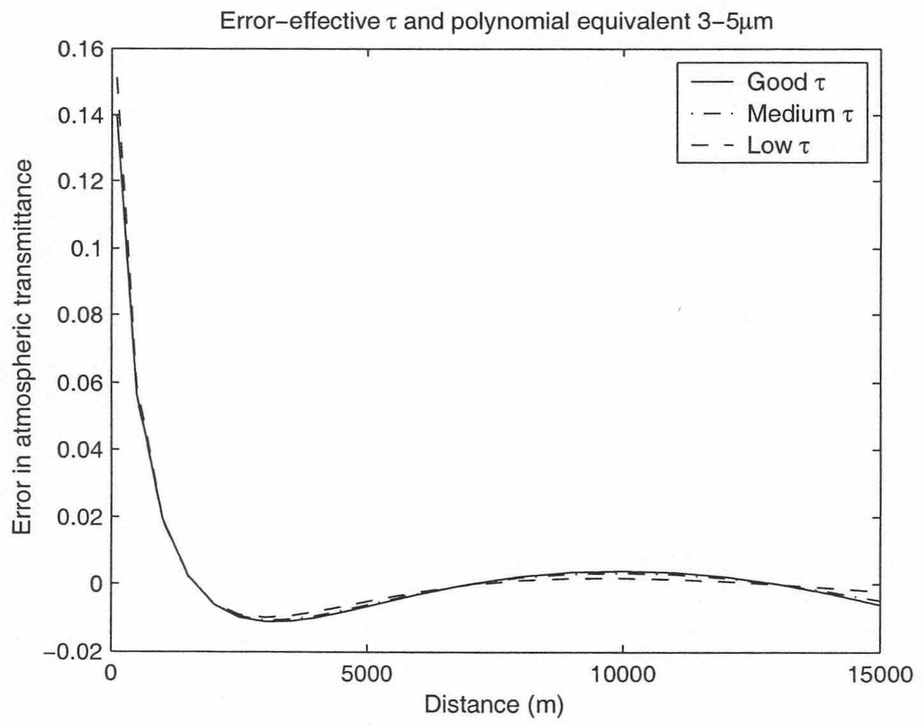

Figure 5.16: Error between actual and equivalent atmospheric transmittance - 3-5$\mu$m

## 5.4 The equivalent temperature of an object

The temperature of the source is an important parameter in determining the equivalent atmospheric transmittance. An additional complication arises when the object consists of a number of areas, each at a different temperature. In this case the different areas and temperatures must be combined into a projected area at an equivalent temperature. The average atmospheric transmittance for an object at a given temperature $T$ is given by Holst [21, p287] as:

$$\tau_{ave} = \frac{\int_{\lambda_1}^{\lambda_2} \Delta M_e(\lambda, T)[\tau_{atm}(\lambda)]^R \tau_{optics}(\lambda) R_d(\lambda) d\lambda}{\int_{\lambda_1}^{\lambda_2} \Delta M_e(\lambda, T)\tau_{optics}(\lambda) R_d(\lambda) d\lambda}, \tag{5.5}$$

with $\Delta M_e(\lambda, T)$ the difference in spectral radiant exitance between the object and the background, $\tau_{atm}$ is the spectral atmospheric transmittance at distance $R$, $R_d(\lambda)$ is the detector's spectral response and $\tau_{optics}(\lambda)$ is the optical system's spectral transmittance. Equation (5.1) is a simplified form of Equation (5.5). The average temperature for an object consisting of a number of areas at different temperatures is given by Lauber *et al.*[22] as:

$$T_{ave} = \frac{\sum_{i=1}^{N} A_i T_i}{\sum_{i=1}^{N} A_i}, \tag{5.6}$$

where $A_i$ is the area of each part, $T_i$ is its temperature and the object consists of N areas at constant temperature.

The effective transmittance is influenced by the source temperature. This is illustrated in Figure 5.17. The data in the figure were generated using medium atmospheric transmittance, a system response of 3-5$\mu$m and the source temperatures as indicated in the figure.

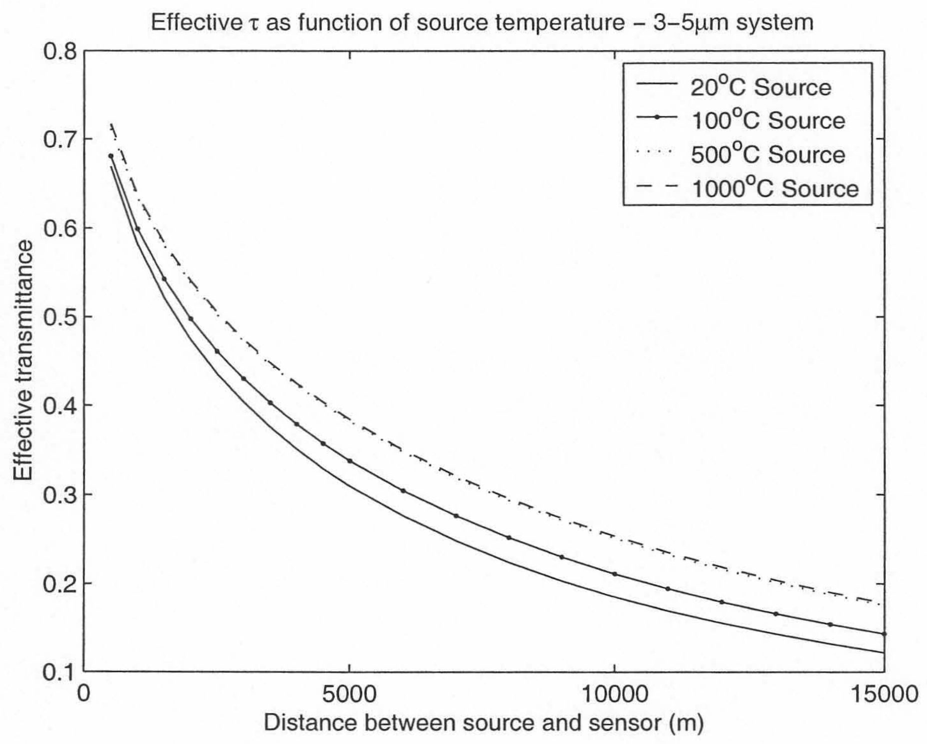

Figure 5.17: Effect of source temperature on the effective atmospheric transmittance

## 5.5 Path radiance

The purpose of this section is to investigate the accuracy of mapping path radiance to a single parameter. The path radiance can be be simulated in OpenGL by assigning a colour to the fog, as was discussed in Section 3.11.2.

The path radiance between the source and detector for a point source is given in Equation (2.10) as:

$$E_{path} = \int_{\lambda_0}^{\lambda_1} L(\lambda)_{path\ sensor\ to\ source} S_\lambda \Omega_{source}(1 - \tau_{\lambda,source})d\lambda. \tag{5.7}$$

Equation (5.7) can be re-written for a point-source after integration of spectral quantities as:

$$E_{path} = \frac{L_{path} A_{source}}{r^2}\varepsilon, \tag{5.8}$$

with $r$ the distance between the source and sensor, $L$ the radiance of a blackbody at the temperature of the path, $A_{source}$ the area of the source and $\varepsilon$ the equivalent emissivity. The area of the source is used because the path radiance is from the same area as that of the source. The path radiance can therefore be represented as a source at the temperature of the atmosphere with an emissivity equal to $(1-\tau_{\lambda,source})$, with $\tau_{\lambda,source}$ the atmospheric transmittance of the path between the sensor and the source. The path radiance is directly dependent on the atmospheric transmittance between the detector and the source.

The mapping of path radiance to an equation of the form:

$$E_{path} = ke^{\sigma r}, \tag{5.9}$$

was investigated for the three filter bandwidths and the three atmospheric transmittance conditions used previously in this chapter. The results are shown in Figure 5.18 to Figure 5.23. The figures are:

- Figure 5.18 shows the calculated and equivalent path radiance for the 3.9-4.1 $\mu$m filter. Figure 5.19 shows the error in the equivalent path radiance. A single parameter model was used to calculate $\sigma$ in Equation (5.9).
- Figure 5.20 shows the calculated and equivalent path radiance for the 3.9-5 $\mu$m filter. Figure 5.21 shows the error in the equivalent path radiance. A single parameter model was used to calculate $\sigma$ in Equation (5.9).
- Figure 5.22 shows the calculated and equivalent path radiance for the 3-5 $\mu$m filter. Figure 5.23 shows the error in the equivalent path radiance. A single parameter model was used to calculate $\sigma$ in Equation (5.9).

The values for $k$ and $\sigma$ in Equation (5.9) were calculated for a point source and is not valid for an extended source. The path radiance for an extended source is given by:

$$E_{path} = L_{path}\Omega_{sensor}\varepsilon, \tag{5.10}$$

with $\Omega$ the sensor's field-of-view. The values of $k$ and $\sigma$ must be determined again for an extended source.

The mapping of the path radiance to Equation (5.9) led to large errors in the cases investigated. The narrow-band system had errors of up to 100%, whereas the two wide-band systems had errors of more than 250%. The path radiances were re-calculated using a second order polynomial model for the extinction coefficient. The polynomial model is of the form:

$$\sigma = \alpha_1 r^2 + \alpha_2 r + \alpha 3. \tag{5.11}$$

The results are shown in Figure 5.24 to Figure 5.29. The figures are:

- Figure 5.24 shows the calculated and equivalent path radiance for the 3.9-4.1 $\mu$m system. Figure 5.25 shows the error in the equivalent path radiance. Equation (5.11) was used to calculate $\sigma r$ in Equation (5.9).
- Figure 5.26 shows the calculated and equivalent path radiance for the 3.9-5 $\mu$m system. Figure 5.27 shows the error in the equivalent path radiance. Equation (5.11) was used to calculate $\sigma r$ in Equation (5.9).
- Figure 5.28 shows the calculated and equivalent path radiance for the 3-5 $\mu$m system. Figure 5.29 shows the error in the equivalent path radiance. Equation (5.11) was used to calculate $\sigma r$ in Equation (5.9).

The polynomial model reduced the errors in the path radiance compared to the single parameter model. The comparison is shown in Table 5.2.

Table 5.2: Comparison in path radiance error when using a single parameter and a polynomial model

| Spectral band of the system | Maximum Error in single parameter model (%) | Maximum error in polynomial model (%) |
|---|---|---|
| 3.9-4.1 $\mu$m | -100 | 30 |
| 3.9-5 $\mu$m | -250 | 50 |
| 3-5 $\mu$m | -250 | 50 |

When a simulation is implemented in OpenGL, a potential source of errors is the change in effective transmittance with a change in temperature. The path radiance is calculated using the effective transmittance of the source, because it is only possible to specify a single value for fog density, but the effective transmittance required to calculate the path radiance might differ a lot due to large temperature differences between the source and the atmosphere.

The large errors generated by the single parameter model would again imply that care must be taken in setting up a simulation where the transmittance, and therefore the path radiance, is represented by a single parameter.

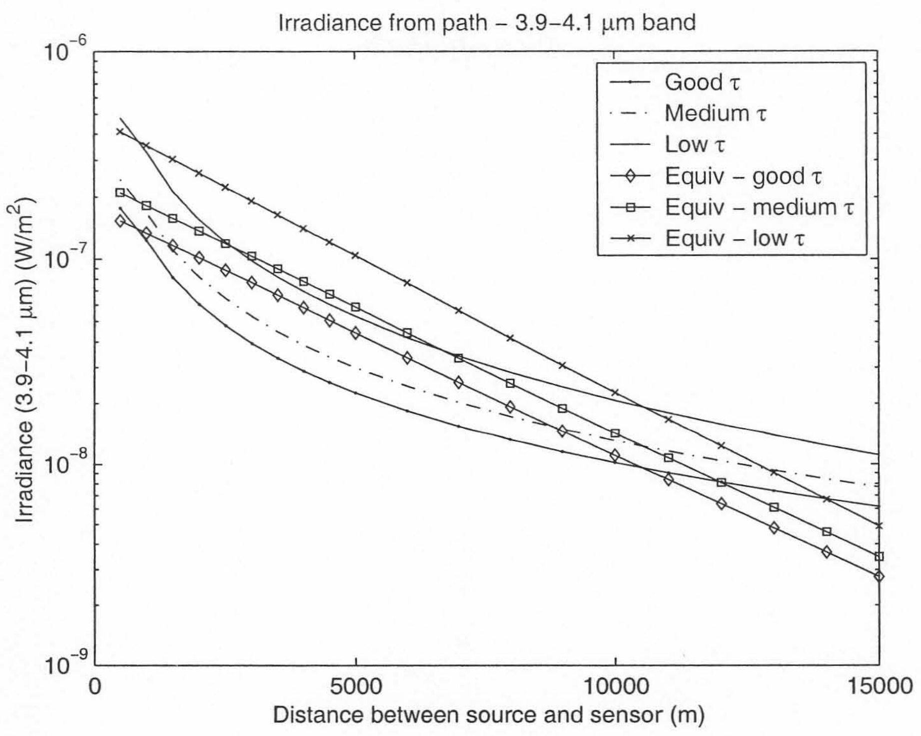

Figure 5.18: Irradiance due to path radiance for various atmospheric conditions, 3.9-4.1 $\mu$m system

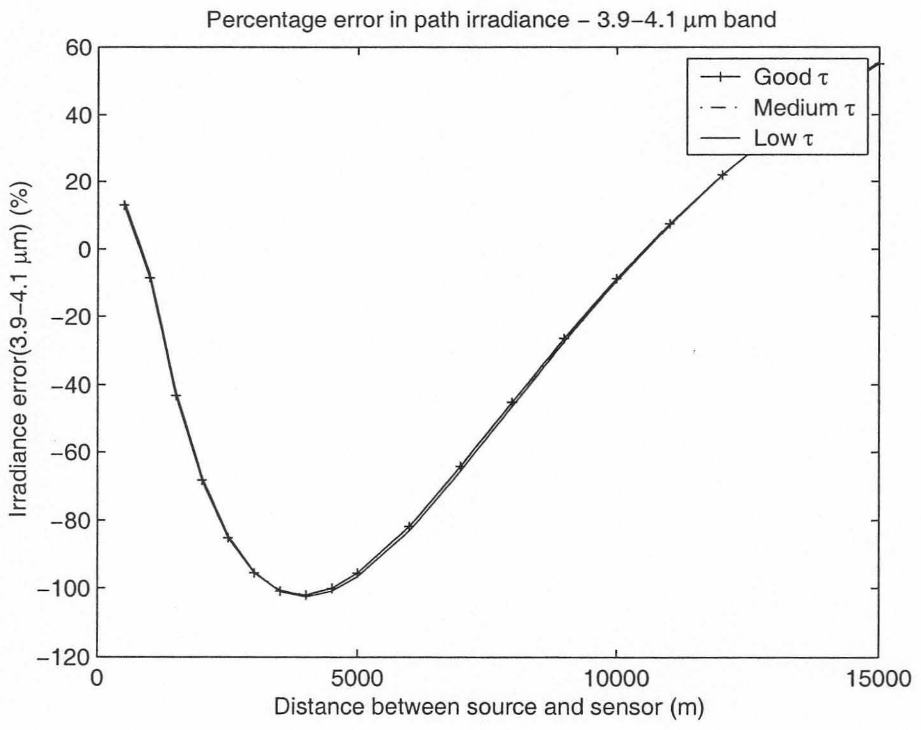

Figure 5.19: Error in irradiance due to path radiance for various atmospheric conditions, 3.9-4.1 $\mu$m system

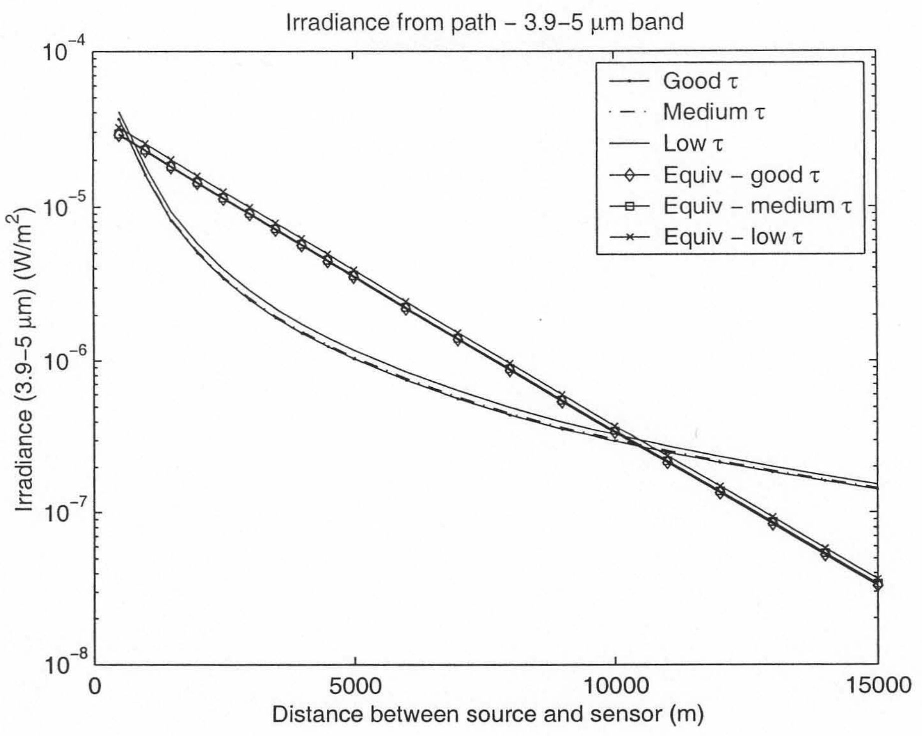

Figure 5.20: Irradiance due to path radiance for various atmospheric conditions, 3.9-5 $\mu$m system

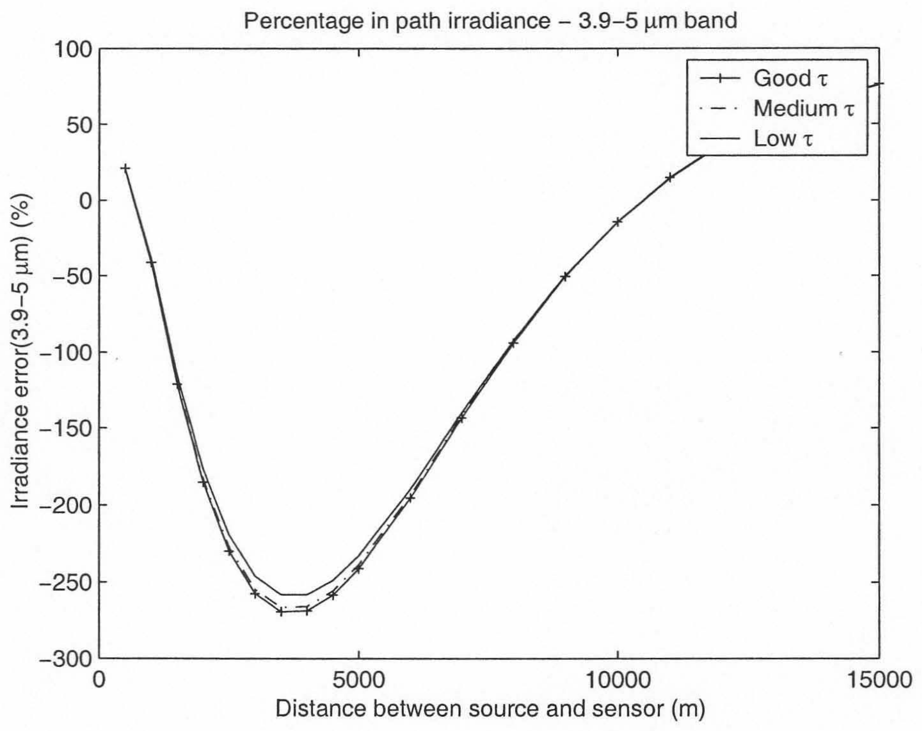

Figure 5.21: Error in irradiance due to path radiance for various atmospheric conditions, 3.9-5 $\mu$m system

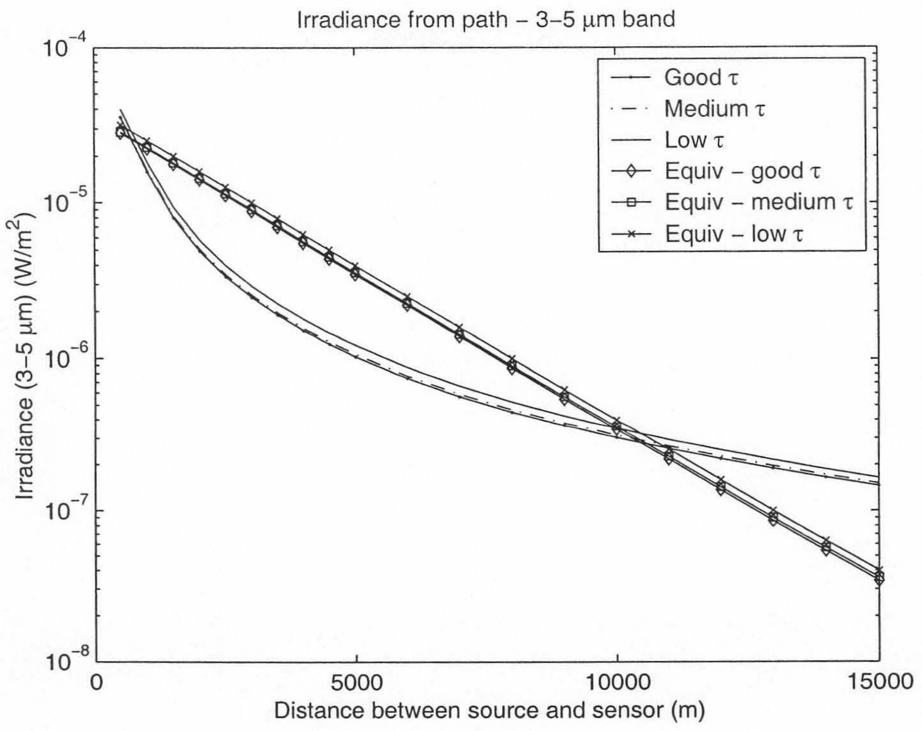

Figure 5.22: Irradiance due to path radiance for various atmospheric conditions, 3-5 $\mu$m system

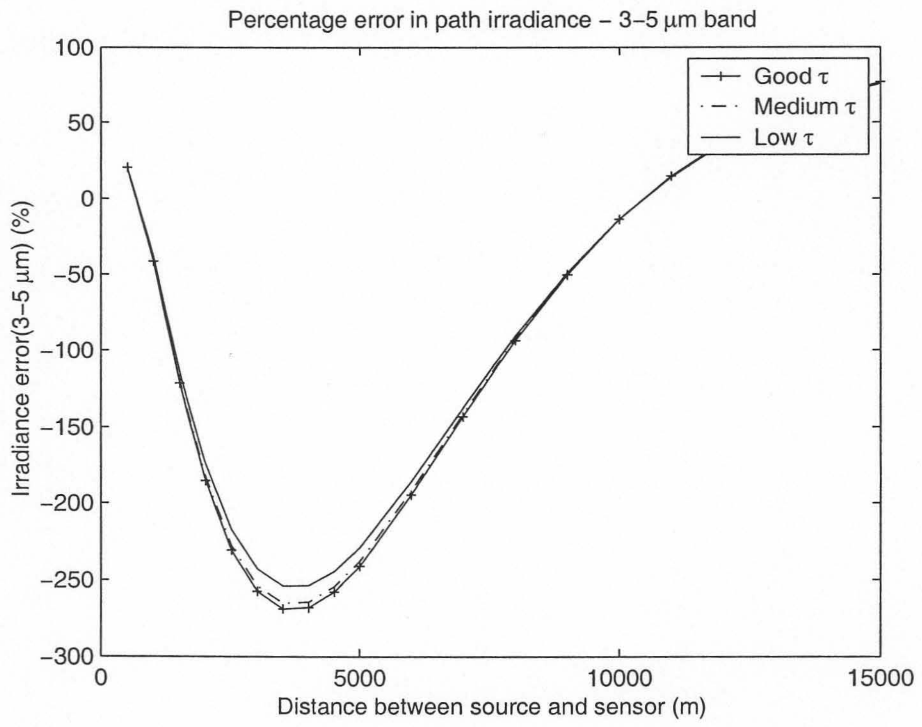

Figure 5.23: Error in irradiance due to path radiance for various atmospheric conditions, 3-5 $\mu$m system

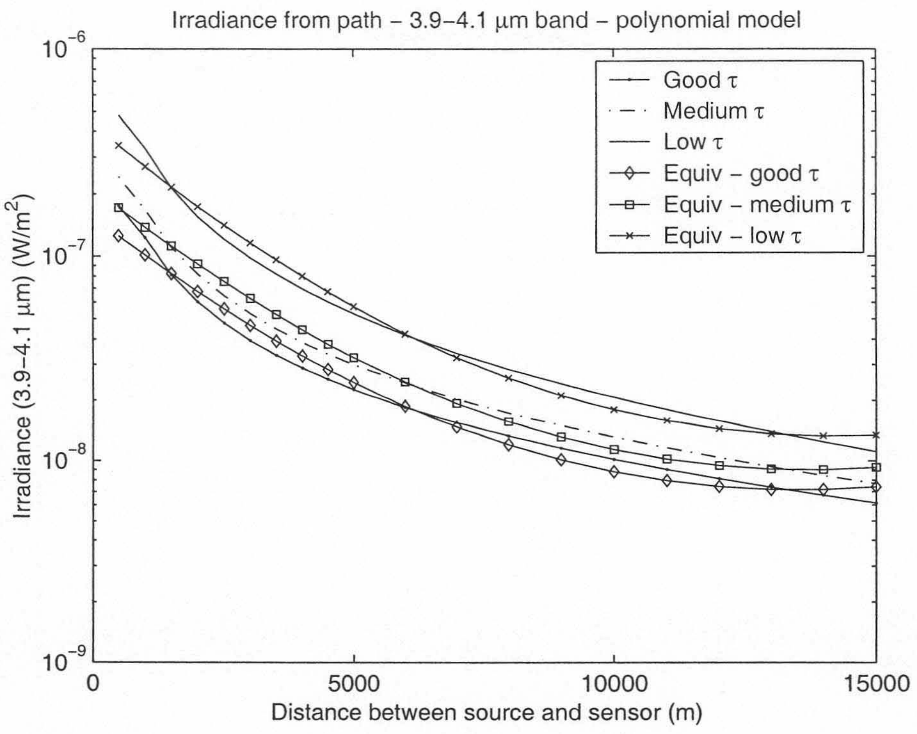

Figure 5.24: Irradiance due to path radiance - polynomial model for $\alpha$, 3.9-4.1 $\mu$m system

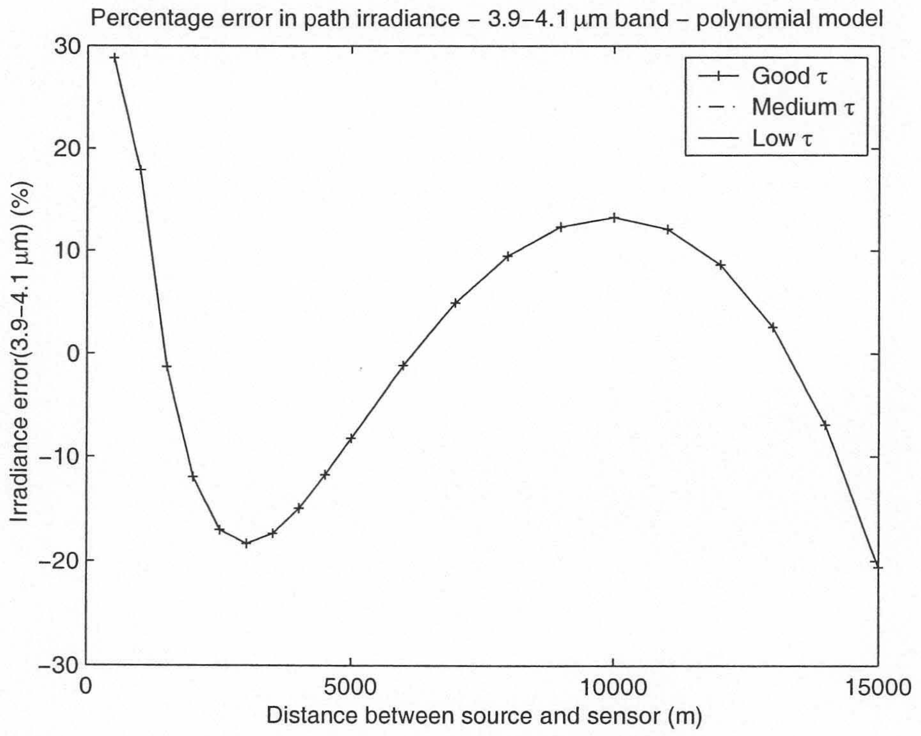

Figure 5.25: Error in irradiance due to path radiance - polynomial model for $\alpha$, 3.9-4.1 $\mu$m system

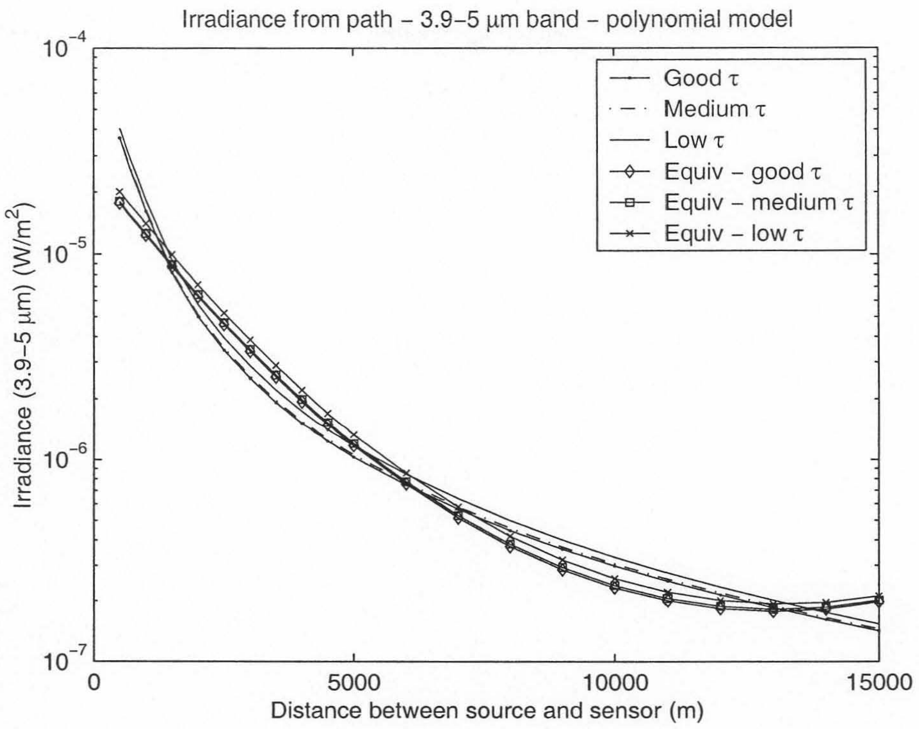

Figure 5.26: Irradiance due to path radiance - polynomial model for $\alpha$, 3.9-5 $\mu$m system

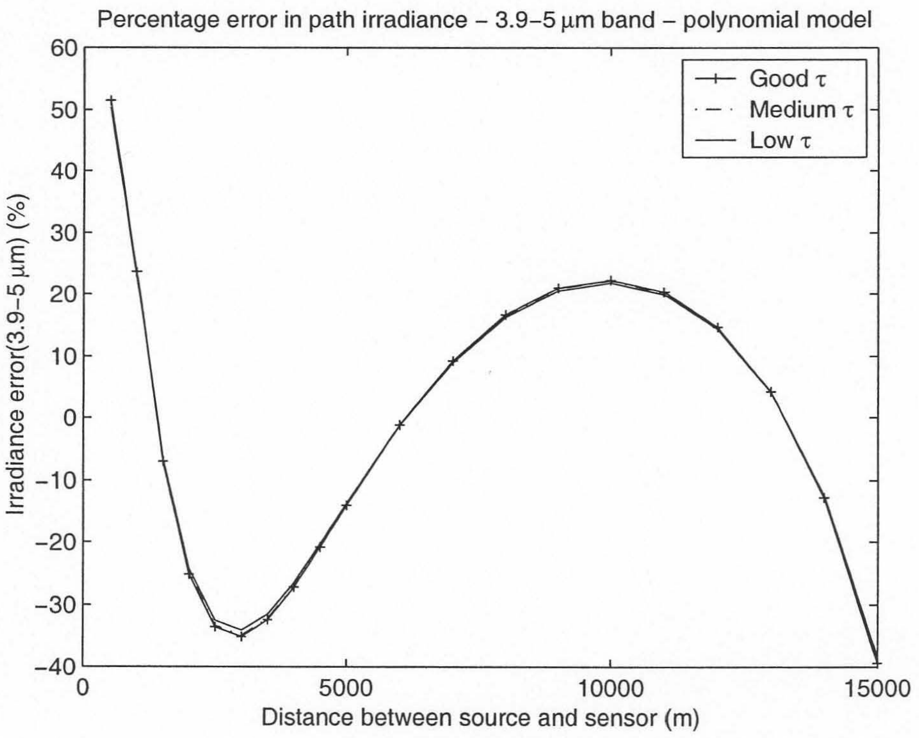

Figure 5.27: Error in irradiance due to path radiance - polynomial model for $\alpha$, 3.9-5 $\mu$m system

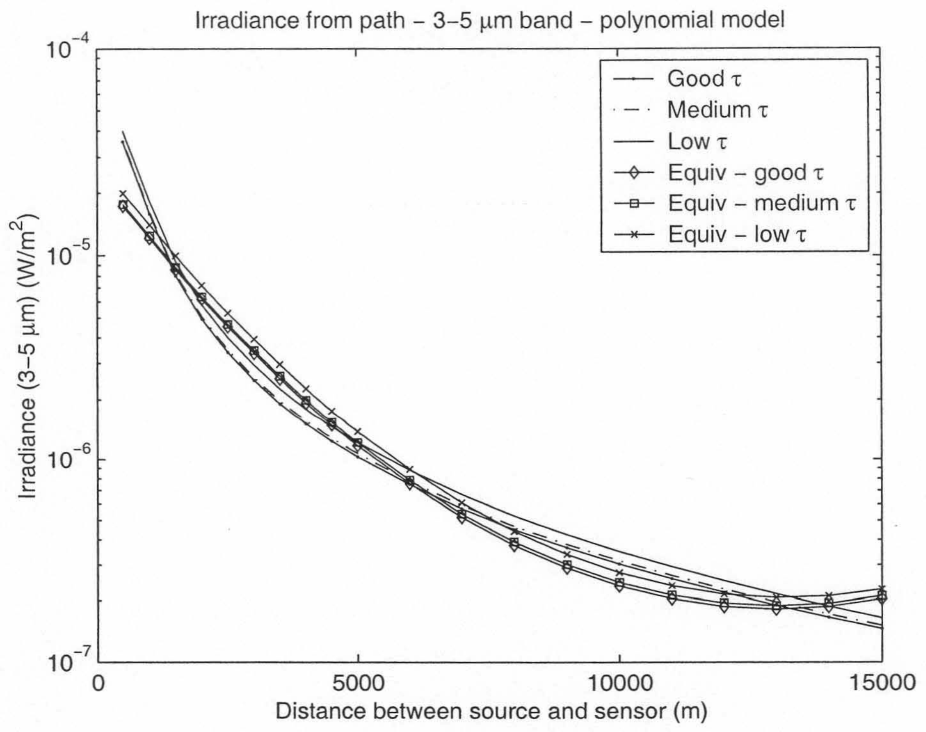

Figure 5.28: Irradiance due to path radiance - polynomial model for $\alpha$, 3-5 $\mu$m system

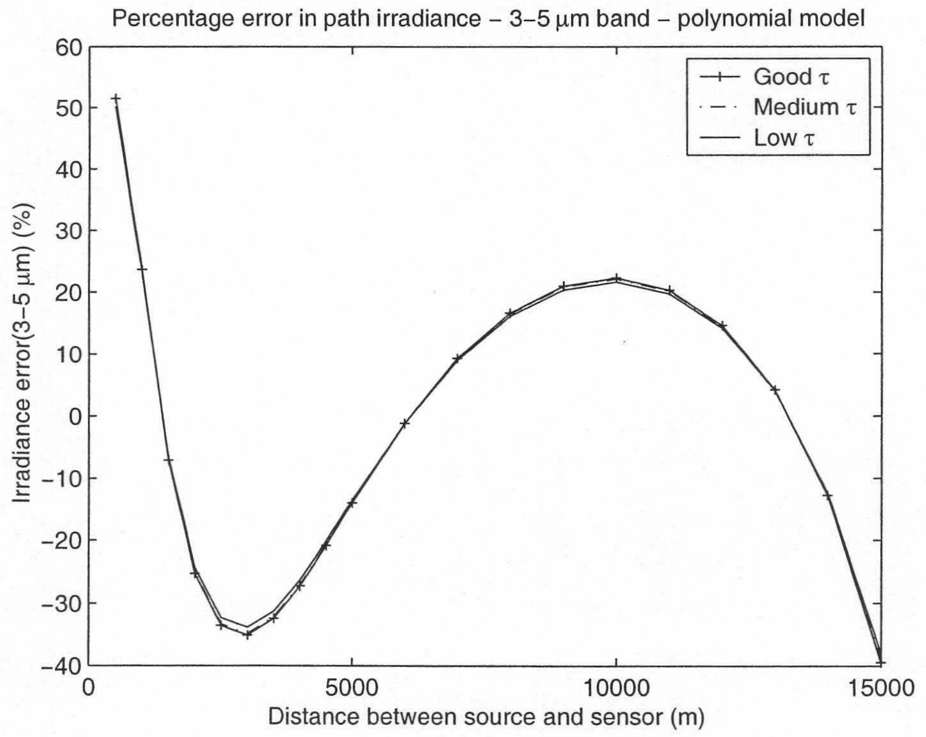

Figure 5.29: Error in irradiance due to path radiance - polynomial model for $\alpha$, 3-5 $\mu$m system

## 5.6 Conclusion

### 5.6.1 Mapping the atmospheric transmittance to a single parameter equivalent

The mapping of atmospheric transmittance to a single parameter is only successful in cases where the spectral band of interest is within an atmospheric transmittance window. This is illustrated in Figure 5.7 and Figure 5.8. The effect of absorption bands leads to errors in the cases where a transmission band and an absorption band are included in the spectral band of interest. This is mainly due to the $e^{-\alpha r}$ function that approaches 1 when the distance $r$ approaches 100 m. The 100 m distance was the shortest distance used in this evaluation. In cases where an absorption band is included, the transmittance at shorter distances, in this case 100 m, does not approach 1.

### 5.6.2 Mapping the atmospheric transmittance to a polynomial equivalent model

The mapping of the atmospheric transmittance to a polynomial model for the absorption coefficient, leads to the lower errors shown in Figure 5.13 to Figure 5.16. The reduction in errors is mainly due to the polynomial model that does not approach 1 when the distance approaches 100 m. The use of the polynomial model is not directly applicable to the mapping of atmospheric transmittance to OpenGL, but can be useful in simulations where the atmospheric transmittance is required as a function of distance and it is not possible to re-run the MODTRAN model.

### 5.6.3 Modeling the path radiance

There are a number of potential causes for errors in the modelling of the path radiance. The most important two are:

- **The effective transmittance.** The effective transmittance of the source and the atmospheric path can be substantially different due the the difference in temperature between the source and the path. This was illustrated in Section 5.5. The path radiance is calculated using the effective transmittance of the source, potentially leading to large errors.
- **The mapping of the atmospheric transmittance.** The path radiance is calculated from the effective atmospheric transmittance. The inaccuracy in calculating the atmospheric transmittance is reflected in the errors calculating the path radiance shown in Figure 5.18 to Figure 5.23. The use of a polynomial model for atmospheric transmittance leads to a reduction in the error, as illustrated in Figure 5.24 to Figure 5.29.

# Chapter 6

# Implementation of a Radiometric Simulation in OpenGL

The purpose of this chapter is to investigate the implementation of a practical infrared simulation in OpenGL. It will address a number of the problems identified in the previous chapters and suggest techniques to increase the accuracy of a simulation.

## 6.1 Dynamic range of the fog colour (path radiance)

In a typical South African scenario, the temperature of the atmosphere varies between 0°C and 40°C. This is equivalent to radiance levels of between 0.64 and 2.95 $\mathrm{Wm^{-2}sr^{-1}}$ for a bandwidth of 3-5 $\mu$m. The radiance values and their integer representations are shown in Figure 6.1. If these radiance values are to be represented by integers, the radiance from a source at 0°C to a source at 40°C are rounded to only 3 integers. The path radiance and atmospheric background radiance is therefore a potential source of errors in simulations. In detection systems where the pre-amplifier is AC-coupled to the detector, the DC-offset due to the path radiance and atmospheric background radiance is cancelled. These systems include some imaging systems and single-detector systems that use a reticle to chop the irradiance. If the purpose of the simulation is to simulate a DC-coupled system, such as a single detector radiometer, where the image is integrated and projected to a single point, the background and path radiance can lead to significant errors, especially in cases where the background radiance represents a large percentage of the image.

## 6.2 Simulation setup

A simulation was set up to investigate the effect of the different image components on the total error. The target object was defined as a sphere with a radius of 1m and a radiance value of 12278 $\mathrm{Wm^{-2}sr^{-1}}$, representing a surface temperature of 920°C. The binary representation of 12278 is 10111111110110, giving values in all three the "colours" of a synthetic display resolution of 14 bits. The colour of the fog was set to a *RedGreenBlue* value of *(1,0,0)*, representing a temperature of 11°C in the 3-5 $\mu$m band. The density of the fog was set to $1.3645\times10^{-4}$, the value obtained in Figure 5.11. The irradiance on the sensor was determined for a range of distances and compared with theoretically calculated values. The colour of the fog was

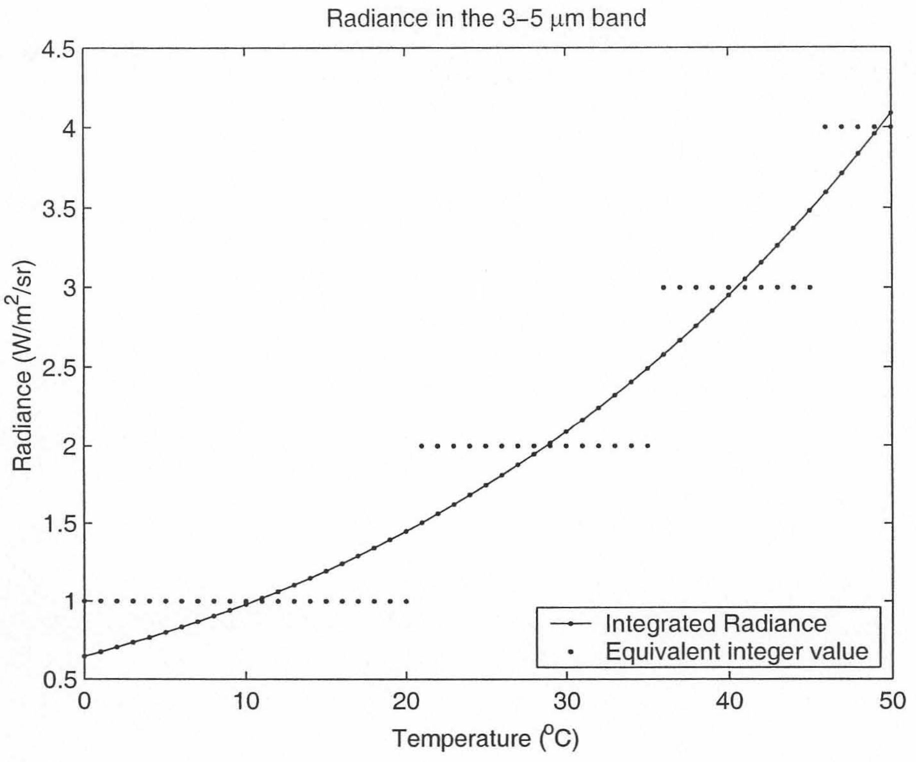

Figure 6.1: Radiance from sources at a range of temperatures

then changed to (*2,0,0*), (*3,0,0*) and (*4,0,0*), equivalent to an atmospheric temperature of 29°C, 41°C and 49°C, and the simulation repeated. The results are shown in Figure 6.2.

The irradiance values were below the calculated values for fog colours of (*1,0,0*) and (*2,0,0*), whereas a fog colour of (*3,0,0*) gave results similar to the predicted values at longer ranges. A fog colour of (*4,0,0*) gave irradiance values above the calculated values at longer distances. At short distances the calculated irradiance is higher than the rendered irradiance in all the cases. This indicates that the fog density factor is too high.

## 6.3 A simulation with an 8 bit dynamic range

A temperature range of 0°C to 250°C can be represented using an 8-bit radiance value, for radiance integrated from 3 to 5$\mu$m. The quantisation errors limit the temperature resolution in the area close to atmospheric temperatures, as mentioned in Section 6.1. The simulation in Section 6.2 was repeated with the source temperature set at 200°C, equivalent to a radiance value of 113 $Wm^{-2}sr^{-1}$. The fog density was set to $1.4410 \times 10^{-4}$. The density value was obtained by calculating the effective atmospheric transmittance for a source at 200°C and finding the equivalent extinction coefficient. The results are shown in Figure 6.4 and Figure 6.5. The fog radiance of *(3,0,0)* gave an error of less than $1 \times 10^{-3}$ $Wm^{-2}$ in irradiance at distances larger than 1000m. The simulation illustrates that 8-bit resolution can be used to simulate infrared scenarios in cases where the radiance levels are below 256 $Wm^{-2}sr^{-1}$. Careful selection of the input parameters are required to obtain accurate results.

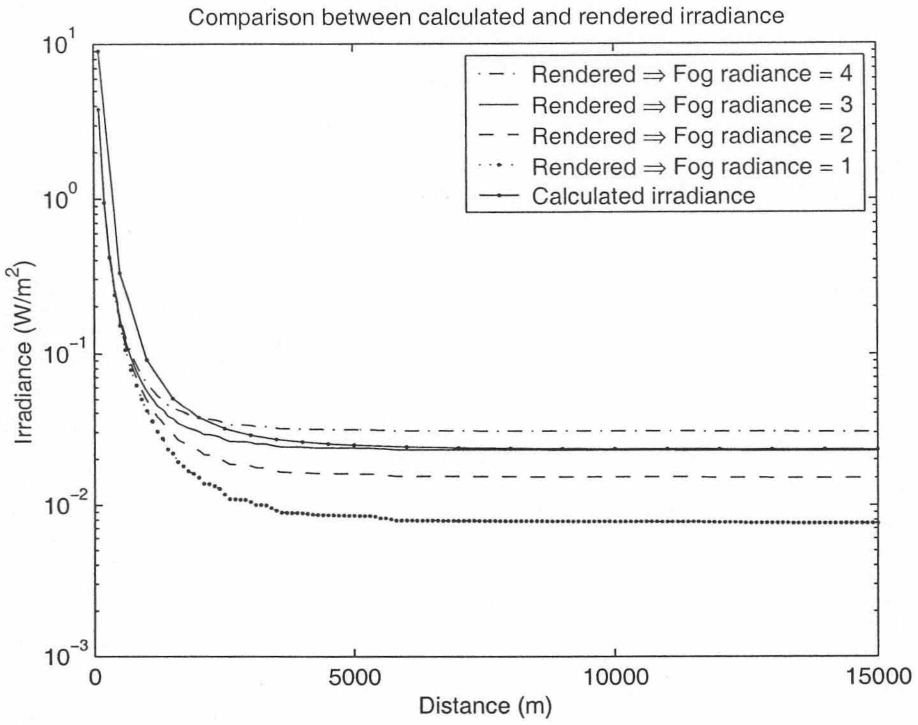

Figure 6.2: Irradiance from scenario with different fog radiances - 14bit

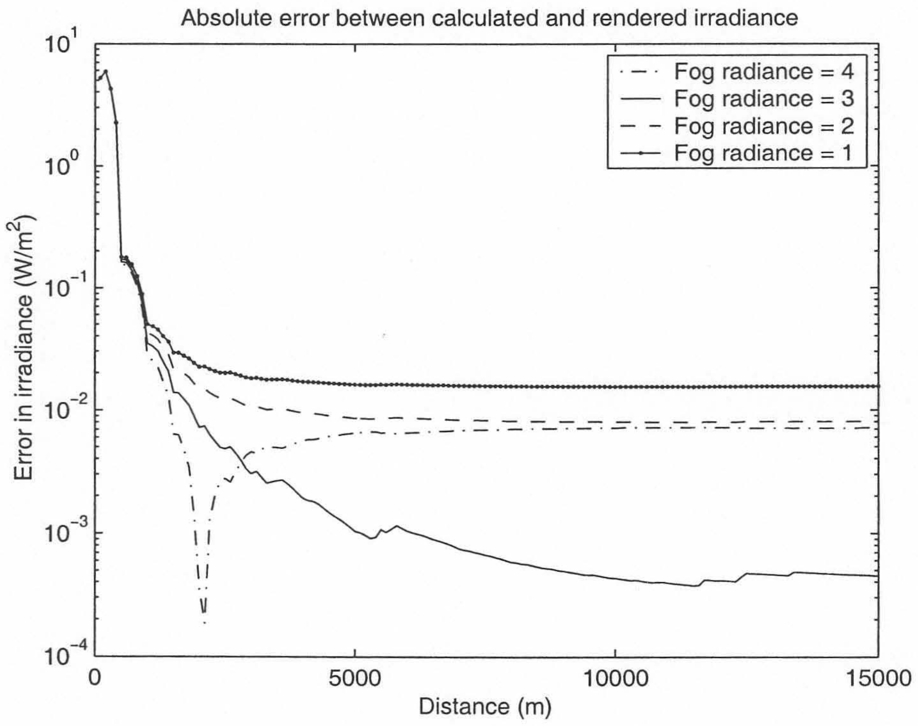

Figure 6.3: Irradiance error with different fog radiances - 14bit

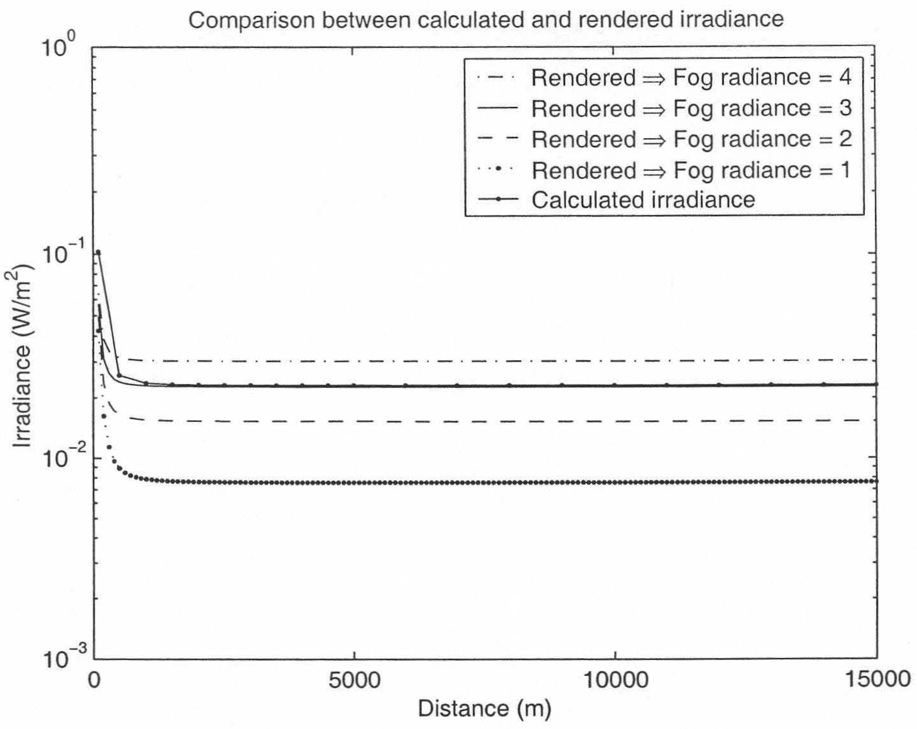

Figure 6.4: Irradiance from scenario with different fog radiances - 8bit

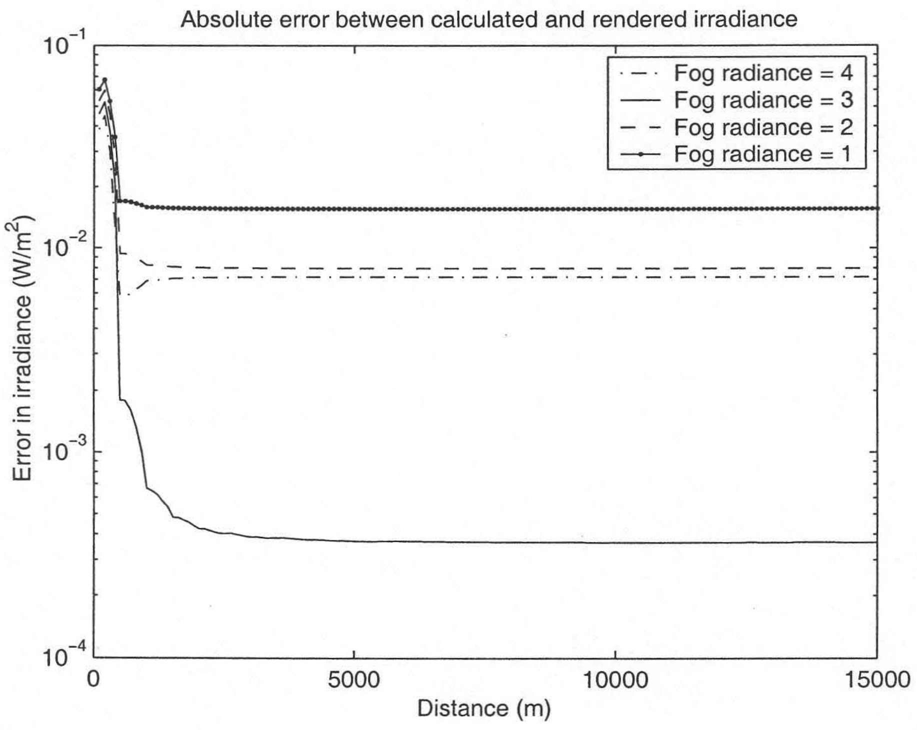

Figure 6.5: Irradiance error with different fog radiances - 8bit

## 6.4 Techniques to reduce the rendering errors

### 6.4.1 Super-resolution

A source of errors is that pixels are combined into larger pixels as the distance between the object and the sensor increases. If a single pixel is at a much higher radiance than its neighbours, the irradiance due to that pixel might be lost in the final image after combination with the neighbouring pixels. That can lead to large errors in the irradiance. It is possible to increase the accuracy of the rendered image by rendering the target object at a high resolution and re-sampling the image before adding the target object to the final image.

### 6.4.2 Mapping the dynamic range of the radiance to hardware's dynamic range

If simulations are limited to low temperatures, for example below 200°C, the input radiance values can be spread over the hardware's dynamic range. This can lead to greater accuracy in the resulting simulation, although an extra processing step is required to convert the image from gray scale to radiance values.

### 6.4.3 Multi-stage processing of the image

A typical infrared scenario consists of target radiance, background radiance, atmospheric transmittance and atmospheric path radiance. The results in Chapter 4 show that it is possible to render an object with reasonably high accuracy at short distances, using techniques such as extending the dynamic range. The results in Chapter 5 show that errors in calculating the atmospheric transmittance and atmospheric path radiance occur and can lead to large errors. The limited dynamic range of the rendering hardware is an important source of errors in calculating the path radiance.

The optimal method of rendering images would then be to render the target objects at a short distance using OpenGL. The spatial characteristics of the target are taken into account and the result is a bitmap viewed from the correct azimuth and elevation. The image can be converted from gray scales to radiance if the technique in Section 6.4.2 was used. The bitmap is modified using super-resolution to compensate for the distance between the target and the sensor in the actual scenario. The atmospheric transmittance is calculated using Equation (5.4) or a $e^{-\alpha r}$ format in cases where the simulation is for narrow bands. The target bitmap is scaled with the value of the atmospheric transmittance. The path radiance is calculated and added to the value of the target bitmap. The atmospheric background radiance is pre-calculated using MODTRAN and assigned to a bitmap. The two bitmaps are combined into a final rendered image. This image consists of 32-bit floats. Adding the atmospheric background radiance to the target would introduce large errors in cases where the target to background ratio is close to one. The correct technique would be to only add the background radiance values to the parts of the target image where the pixel values are zero. This would increase the time required to execute the simulation. The process is shown in Figure 6.6.

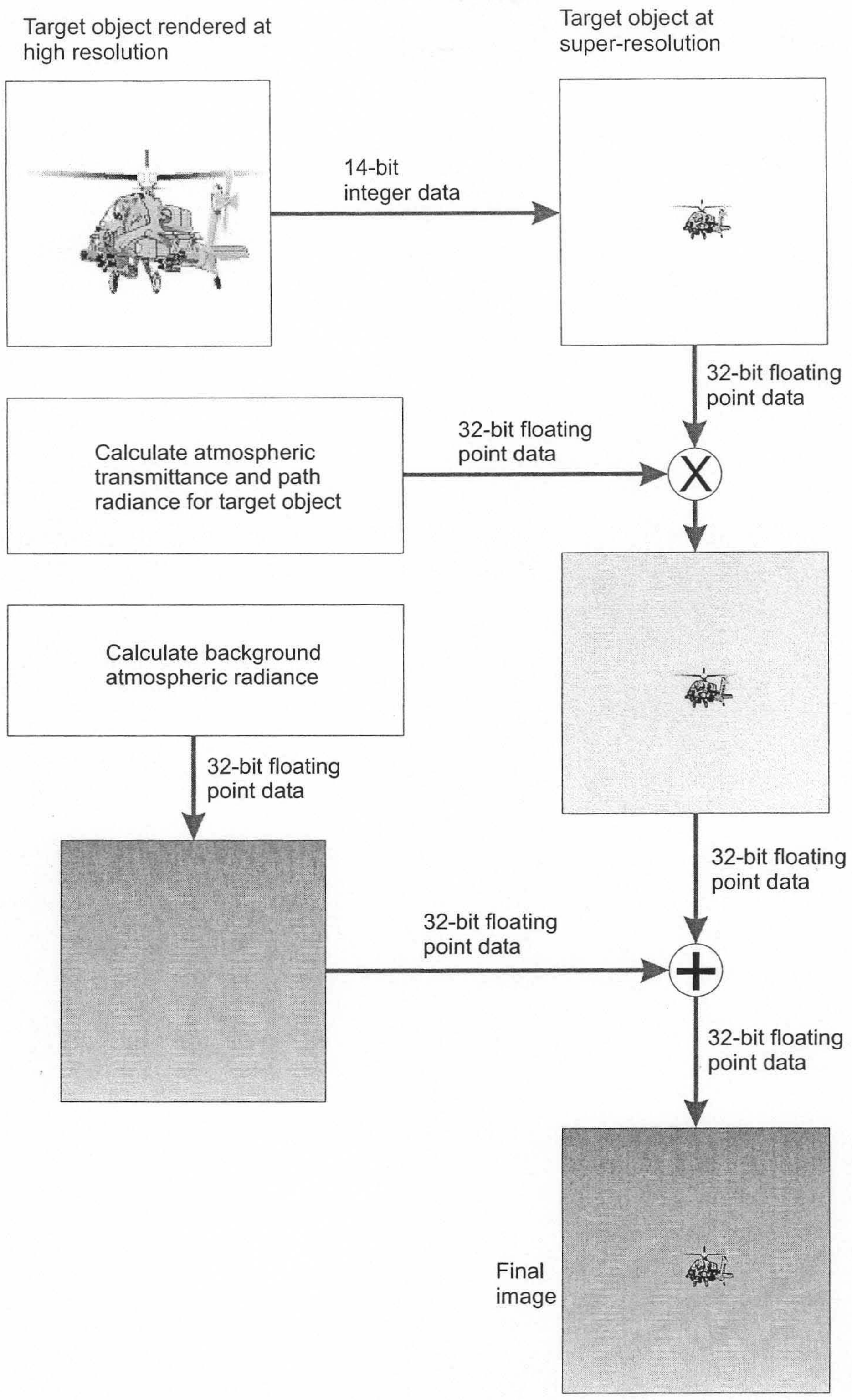
Target object rendered at high resolution
Target object at super-resolution
14-bit integer data
32-bit floating point data
Calculate atmospheric transmittance and path radiance for target object
32-bit floating point data
X
Calculate background atmospheric radiance
32-bit floating point data
32-bit floating point data
32-bit floating point data
+
32-bit floating point data
Final image

Figure 6.6: Rendering an image to obtain maximum accuracy

## 6.5 Benchmarks

A series of benchmarks were carried out to evaluate the frame rates that can be achieved in simulations. The test system was a 667 MHz Pentium III machine with a GeForce 2 GTS graphics card running Windows NT4. The following tests were carried out:

- **Render images** The GeForce card was used to render OpenGL images. The test object was a sphere consisting of 10000 polygons. The card rendered 123 frames per second.
- **Read images from the frame buffer** An image was rendered on the GeForce card and a 256×256 sub-image was read out using the *glReadPixels* command. It was possible to read out 14.7 sub-images per second.
- **Multiply a matrix with a constant** A 512×512 floating point matrix was defined in memory and the elements in the matrix were each multiplied with a floating point value. It was possible to repeat the operation 6.9 times per second.
- **Calculate 256×256 super-sampled image** A 256×256 floating point matrix was defined in memory and values were assigned to 5×5 elements in the centre of the matrix. The rest of the matrix was set to zero. The super-sampled equivalent of the matrix was calculated at various distances. The operation was repeated 26.8 times per second.
- **Render image, read result and calculate super-sampled image** A 512×512 image was rendered using the GeForce card, a 256×256 sub-image was read out from the card and the super-sampled equivalent of the sub-image was calculated. The frame rate was 5.4 frames per second.

# Chapter 7

# Conclusion

Modern graphics accelerators present exiting opportunities for low-cost simulations on personal computers. There are, however, a number of limitations that must be kept in mind when implementing a simulation. These limitations reduce the accuracy of simulations and techniques must be implemented to work around them.

## 7.1 The use of OpenGL as a library for generating infrared scenarios

OpenGL was designed to generate visually pleasing images at real-time frame rates. The required resolution of the images are therefore limited to values that would lead to images that "look right". Infrared scene generation has much higher requirements for scene fidelity. Infrared parameters such as radiance, atmospheric transmittance and atmospheric path radiance can be implemented directly on OpenGL. It is also possible to include reflected energy from sources such as the sun in simulations. The main drawback is that the resolution of hardware limits the accuracy of the resulting images. The dynamic range is a limitation even on systems such as the SGI Onyx2 with an InfiniteReality engine [5]. Parameters such as the atmospheric transmittance and path radiance must be calculated separately, and do not use the rendering hardware that is normally used for generating visual images.

## 7.2 Limitations of using a personal computer as simulation platform

Colours on the personal computer were defined to yield images that are visually acceptable. The colour space was therefore limited to 3×8-bit colours. The maximum number of gray scales on a personal computer is 256, placing a severe limitation on the dynamic range of the resulting infrared image. The graphics co-processors are designed to deliver images at real-time frame rates, but although the frame rate is sufficient for infrared simulations, the dynamic range is too low.

## 7.3 Benchmarks

The benchmarks indicated that the GeForce card can render images at more than the required frame rate. It is not possible to read the rendered images from the frame buffer at the required frame rates, using the available OpenGL commands. Calculating an image using the super-resolution technique is also too slow for practical implementation. The matrix multiplications that are required to implement the atmospheric transmittance and path radiance calculations are too slow to calculate on-the-fly during real-time simulations.

## 7.4 Implementing a physically realistic simulation on a personal computer

Commercial-off-the-shelf (COTS) systems are limited in their ability to render accurate radiometric images at real-time frame rates, using only their native hardware graphics acceleration. This statement is also valid for very expensive graphics supercomputers such as the SGI Onyx2. In all the cases the reason is the difference between the dynamic range required for visually acceptable images and the dynamic range required for radiometrically accurate infrared images.

Implementing an infrared simulation on a COTS system requires a number of compromises. Rendering images of objects at close ranges can be accomplished to a high level of accuracy, as was shown in Chapter 4. These object images can then be translated to other distances using the super-resolution technique described in Section 6.4.1. The image of the target object can be scaled with the atmospheric transmittance and combined with a pre-calculated bitmap representing the atmospheric background.

The time required to read an image from the frame buffer into the personal computer's main memory and the time required for matrix multiplications are major bottlenecks, reducing the simulation's frame rates. These bottlenecks must be removed before a real-time simulation can be successfully implemented. It is possible to do this by sampling the video card's analog outputs and performing the matrix multiplications on an external digital signal processing card.

It is therefore possible to generate physically realistic infrared images on a personal computer at real-time frame rates, given that techniques can be implemented to eliminate the problems with memory bandwidth and processing power on a personal computer.

## 7.5 Contribution

The study showed that it is possible to generate infrared imagery on a personal computer, as opposed to the expensive hardware normally used for this task. The task was accomplished by extending OpenGL to the infrared band through the mapping of infrared parameters to OpenGL. The mapping of infrared parameters to OpenGL can be inaccurate and techniques were developed to limit the errors. The techniques used are described in the article by le Roux *et al.* [23]. The technique to extend the dynamic range of SGI machines, as suggested by

Olsen *et al.* [6], was extended to a personal computer. A technique was suggested that would enable the generation of accurate infrared imagery on a personal computer. A summary of the work included in this dissertation was presented at the AeroSense 2001 conference [24].

## 7.6 Future Work

The biggest hardware problem with implementing the rendering on a personal computer is the speed with which data can be read from the graphics card to the PC's main memory. The focus of future work will therefore have to be to increase the speed at which data is read into the PC's memory. This can be done either by sampling the display card's video output or improving the current DMA (direct memory access) rates. A further possibility is to increase the part of the rendering that is done in software in the PC's main memory. This would become more feasible with the continued increase in processor speeds. A form of distributed rendering, where parts of the image are rendered on different computers, would also increase the frame rate. This would present new challenges in transferring the image data at the required frame rates.

# Chapter 8

# References

[1] J. A. Ray, G. A. Larson, and J. E. Terry, "Successful Hardware-in-the-loop Support of the Longbow/HELLFIRE Modular Missile System," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-loop testing V, SPIE Volume 4027*, (Orlando, Florida), pp. 82–90, April 2000.

[2] B. A. Weber and J. A. Penn, "Modeling Synthetic Infrared Data for Classifier Development," in *Proceedings of Conference on Automatic Target Recognition X, SPIE Volume 4050*, (Orlando, Florida), pp. 1–20, April 2000.

[3] B. Shetler, D. Mergens, C. Chang, F. Mertz, J. Scott, S. Brown, R. Strunce, F. Maher, S. Kubica, R. de Jonckheere, and B. Tousley, "A comprehensive hyperspectral system simulation I: Integrated sensor scene modeling and the simulation architecture," in *Proceedings of Conference on Algorithms for Multispectral, Hyperspectral and Ultraspectral Imagery VI, SPIE Volume 4049*, (Orlando, Florida), pp. 94–104, April 2000.

[4] A. Berk, L. Bernstein, and D. Robertson, *MODTRAN: a moderate resolution model for LOWTRAN 7*. GL-TR-89-0081, Air Force Geophysics Laboratory: Hanscom AFB, 1978.

[5] R. J. Makar and D. B. Howe, "Real-time IR/EO scene generation utilizing an optimized scene rendering subsystem," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-Loop Testing V, SPIE Volume 4027*, (Orlando, Florida), pp. 145–154, April 2000.

[6] E. M. Olsen, C. Coker, J. Coker, and D. Garbo, "A demonstration of innovative techniques used for real-time closed-loop infrared scene generation," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-loop testing III, SPIE Volume 3368*, (Orlando, Florida), pp. 321–327, April 1998.

[7] M. Lorenzo, E. Jacobs, R. Moulton, and J. Liu, "Optimized mapping of Radiometric Quantities into OpenGL," in *Proceedings of Conference on Modeling, Simulation and Visualization for Real and Virtual Environments, SPIE Volume 3694*, (Orlando, Florida), pp. 173–182, April 1999.

[8] D. C. Anding and A. Szabo, "Real-time image visualisation for sensors," in *Proceedings of Conference on Technologies for Synthetic environments: Hardware-in-the-loop SPIE Vol 2741*, (Orlando, Florida), pp. 232–241, April 1996.

[9] R. L. Sundberg, J. Gruninger, M. Nosek, and J. Burks, "Quick Image Display (QUID) model for rapid real-time target imagery and spectral signatures," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-loop testing II, SPIE Volume 3084*, (Orlando, Florida), pp. 272–281, April 1997.

[10] D. R. Crow, C. F. Coker, D. L. Garbo, and E. M. Olson, "A closed-loop real-time infrared scene generator," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-loop testing III, SPIE Vol 3368*, (Orlando, Florida), pp. 342–351, April 1998.

[11] G. J. Zissis, *Sources of Radiation*, vol. 1 of *The Infrared and Electro-optical Systems Handbook*. PO Box 10, Bellingham, Washington, USA: SPIE Optical Engineering Press, 1996.

[12] F. W. Leuschner, "Introduction to Photonics." Class notes for Photonics EEF410, 1999.

[13] C. L. Wyatt, *Radiometric System Design*. 886 Third Avenue, New York, NY: Macmillan Publishing Company, 1987.

[14] F. G. Smith, *Atmospheric Propagation of Radiation*, vol. 2 of *The Infrared and Electro-optical Systems Handbook*. PO Box 10, Bellingham, Washington, USA: SPIE Optical Engineering Press, 1996.

[15] J. D. Foley and A. van Dam, *Fundamentals of Interactive Computer Graphics*. Addison-Wesley Systems Programming Series, Reading, Massachusetts: Addison-Wesley Publishing Company, 1984.

[16] R. S. Wright and M. Sweet, *OpenGL SuperBible, Second Edition*. Waite Group Press, 1999.

[17] B. de la Rosa, D. Döman, D. Gildenhuys, P. J. Gräbe, T. de W. Jooste, J. G. Pretorius, L. M. Pretorius, and M. J. Schoeman, *Inleidende Algebra*. Johannesburg: McGraw-Hill Boekmaatskappy, 1987.

[18] E. Angel, *Interactive Computer Graphics, A top-down approach with OpenGL*. Addison-Wesley, 1997.

[19] M. Segal and K. Akeley, *The OpenGL Graphics System: A Specification (Version 1.2.1)*. Silicon Graphics, Inc., 1999.

[20] E. M. Olsen, O. M. Williams, R. L. Murrer, and J. R. Kircher, "Resolution and dynamic range capabilities of dynamic infrared scene projection systems," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-loop testing V, SPIE Volume 4027*, (Orlando, Florida), pp. 202–213, April 2000.

[21] G. C. Holst, *Electro-optical Imaging System Performance*. Bellingham, Washington, USA: SPIE Optical Engineering Press, 1995.

[22] Y. Z. Lauber and D. Braun, "Correct weighting of atmospheric transmittance and target temperature, applied to IR airborne reconnaissance systems," in *Proceedings of Conference on Targets and Backgrounds VI: Characterisation , Visualisation and the Detection Process, SPIE Volume 4029*, (Orlando, Florida), pp. 112–119, April 2000.

[23] F. P. J. le Roux and F. W. Leuschner, "Single parameter equivalents of atmospheric transmittance and atmospheric path radiance," *Journal of Electronic Imaging*, vol. Submitted for publication.

[24] F. P. J. le Roux, F. G. Collin, and F. W. Leuschner, "An investigation into the use of a personal computer for generating real-time infrared imagery," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-Loop Testing VI*, (Orlando, Florida), April 2001.

[25] J. A. Buford, A. M. Offutt, and T. M. Reynolds, "Development of a Real-Time Sensor Emulator System for Hardware-in-the-Loop Testing," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-loop testing V, SPIE Volume 4027*, (Orlando, Florida), pp. 163–170, April 2000.

[26] M. Bowden, J. A. Buford, and A. Mayhall, "Advanced real-time dynamic scene generation techniques for improved performance and fidelity," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-loop testing V, SPIE Volume 4027*, (Orlando, Florida), pp. 171–178, April 2000.

[27] A. J. Cantle, M. Devlin, E. Lord, and R. Chamberlain, "High Frame rate, Low Latency Hardware-in-the-Loop Image Generation - An illustration of the Particle Method and Dime," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-loop testing V, SPIE Vol 4027*, (Orlando, Florida), pp. 122–133, April 2000.

[28] N. Chiba, K. Muraoka, H. Takahashi, and M. Miura, "Two-dimensional visual simulation of flames, smoke and the spread of fire," *Journal of Visualization and Computer Animation*, vol. 5, pp. 37–53, 1994.

[29] D. Garbo, E. Olsen, D. Crow, C. Coker, and D. Cunard, "IR Model Development for a High-Speed Imaging Fuze," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-Loop Testing III, SPIE Volume 3368*, (Orlando, Florida), pp. 328–341, April 1998.

[30] A. le Goff, P. Kersaudy, J. Latger, T. Cathala, N. Stolte, and P. Barillot, "Automatic Temperature Computation for Realistic IR Simulation," in *Proceedings of Conference on Targets and Backgrounds VI: Characterization, Visualization, and the Detection Process, SPIE Volume 4029*, (Orlando, Florida), pp. 187–196, April 2000.

[31] J. P. Gourret and P. Afflard, "Three-dimensional texture generator supervised by a small number of parameters," *Journal of Visualisation and Computer Animation*, vol. 3, pp. 105–127, 1992.

[32] A. V. Gitin, "Radiometry. A comprehensive approach," *Journal of Optical Technology*, vol. 65, no. 2, pp. 132–140, 1998.

[33] A. Houlbrook, M. Gilmore, I. Moorhead, D. Filbee, C.Stroud, G. Hutchings, and A. Kirk, "Scene simulation for camouflage assessment," in *Proceedings of Conference on Targets and Backgrounds VI: Characterization, Visualization, and the Detection Process, SPIE Volume 4029*, (Orlando, Florida), pp. 247–255, April 2000.

[34] H.-K. Hong, S.-H. Han, G.-P. Hong, S.-G. Jahng, and J.-S. Choi, "IR Model of 3d Aircraft for simulation of Reticle Seekers," *Journal of Circuits, Systems and Computers*, vol. 7, no. 4, pp. 333–344, 1997.

[35] R. J. Makar, B. O'Toole, and P. Rogers, "Real-time radiometric calculations utilizing SGI symmetric multiprocessing architecture," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-Loop Testing II, SPIE Volume 3084*, (Orlando, Florida), pp. 260–270, April 1997.

[36] R. J. Makar and B. O'Toole, "Real-time synchronized multiple-sensor IR/EO scene generation utilizing the SGI Onyx2," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-Loop Testing III, SPIE Volume 3368*, (Orlando, Florida), pp. 300–309, April 1998.

[37] J. C. Paul, P. M. Deville, and C. Winkler, "Modelling radiative properties of light sources and surfaces," *Journal of Visualization and Computer Animation*, vol. 6, pp. 231–240, 1995.

[38] N. I. Pavlov, V. A. Shevoldin, Y. A. Shuba, and G. I. Yasinskiĭ, "Combined analysis of images of scenes in the thermal and visible regions, using physical models," *Journal of Optical Technology*, vol. 65, no. 12, pp. 1045–1048, 1998.

[39] N. I. Pavlov, Y. A. Shuba, and V. A. Shevoldin, "Interconnection of the radiance of objects in the IR and visible regions during natural heat exchange," *Journal of Optical Technology*, vol. 65, no. 3, pp. 204–206, 1998.

[40] J. Sanders and R. Roland, "Captive flight test-based infrared validation of a hardware-in-the-loop simulation," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-loop testing V, SPIE Volume 4027*, (Orlando, Florida), pp. 292–300, April 2000.

[41] O. D. Simmons, S. E. Jacobs, R. J. Makar, F. J. Stanley, T. W. Joyner, and K. B. Thiem, "Advancements in Real-Time IR/EO Scene Generation Utilizing the Silicon Graphics Onyx2," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-loop testing V, SPIE Volume 4027*, (Orlando, Florida), pp. 134–144, April 2000.

[42] J. S. Sanders and S. D. Brown, "Utilization of DIRSIG in Support of Real-Time Infrared Scene Generation," in *Proceedings of Conference on Targets and Backgrounds VI: Characterization, Visualization, and the Detection Process, SPIE Volume 4029*, (Orlando, Florida), pp. 278–285, April 2000.

[43] H. Shekarforoush and R. Chellappa, "Data-driven multichannel superresolution with application to video sequences," *Journal of the Optical Society of America A*, vol. 16, no. 3, pp. 481–492, 1999.

[44] T. Sheremet'eva, G. Filippov, and E. Kalyashev, "Methods of processing and representing the results of measurements of three-dimensional surfaces," *Journal of Optical Technology*, vol. 65, no. 5, pp. 403–405, 1998.

[45] M. Wegener and R. Drake, "High Fidelity Synthetic IR Imaging Model," in *Proceedings of Conference on Technologies for Synthetic Environments: Hardware-in-the-loop testing V, SPIE Volume 4027*, (Orlando, Florida), pp. 163–170, April 2000.

[46] M. Wellfare, D. Vechinski, J. Watson, J. Foster, J. Edwards, and M. Richards, "Irma 5.0 Multi-Sensor Signature Prediction Model," in *Proceedings of Conference on Targets and Backgrounds VI: Characterization, Visualization, and the Detection Process, SPIE Volume 4029*, (Orlando, Florida), pp. 187–196, April 2000.