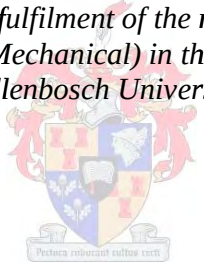
Development of a Ray Tracer for Concentrating Solar Power Systems

by
Sebastian-James Bode

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Mechanical) in the Faculty of Engineering at Stellenbosch University



Supervisor: Prof. Paul Gauché

December 2017

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: .....

Abstract

The world climate system is changing and recent research shows that man-made carbon dioxide emissions from processes such as coal-fired power generation is the most likely cause. In South Africa the bulk of electricity generation comes from coal power plants. In addition to the possible environmental damage caused by these plants, South Africa's reliance on coal exposes the country to possible fuel shortages and rising fuel prices. Concentrating Solar Power (CSP), which uses clean, renewable solar energy for electricity generation, can possibly mitigate these risks and reduce South Africa's dependence on coal. However, for CSP to play a significant role in electricity generation, its Levelised Cost of Electricity (LCOE) must be reduced. This can be achieved through research and development of CSP systems. Numerical software tools, such as ray tracing, play an essential role in design and optimisation of these systems.

The aim of this research was to develop, program, and validate a CSP ray tracer which can be used for ongoing research at the Solar Thermal Energy Research Group (STERG) at Stellenbosch University. The ray tracer, which has been named SUNRAY (Stellenbosch UNiversity RAY tracer), was written in C++. It was developed with realistic sunshape and reflection modules, a number of geometric shapes, automatic tracking algorithms, acceleration routines, and a method to simulate actual mirror surface profiles using scanned data. SUNRAY has been extensively validated using behavioural tests, comparative tests against previously validated ray tracers, and through experimental investigation. For the behavioural cases, it was found that SUNRAY was able to resolve the correct solution for every test. In the comparative tests the relative difference for the power and flux distribution between SUNRAY and validated ray tracers was, on average, no more than 0.4% and 3%, respectively. Furthermore, the execution times for the simulations were, in most cases, faster than that of the validated codes. In the experimental validation various mirror shape profiles were tested under different weather conditions. The experimental tests demonstrated that SUNRAY can be used to adequately determine the magnitude of flux and flux distribution on a target.

This thesis presents the theory behind the various features, algorithms, and routines of SUNRAY as well as the validation of these features. Two novel algorithms are also proposed in this thesis. The first is a method which reduces the number of missed rays by only generating rays above each object in the simulation. The second uses statistical tools to predict the number of rays which are needed in a simulation.

Uittreksel

Die wêreld se klimaatstelsel is besig om te verander en onlangse navorsing het bewys dat mensgemaakte koolstofdioksieduitlaatgasse, wat afkomstig is van die gebruik van steenkool as brandstof vir elektrisiteitsopwekking, waarskynlik die oorsaak daarvan is. Bykomend tot die skade wat deur hierdie aanlegte veroorsaak kan word, stel die feit dat Suid-Afrika tot 'n groot mate staatmaak op steenkool die land bloot aan 'n moontlike brandstoftekort en stygende brandstofpryse. Gekonsentreerde sonkrag (Concentrating Solar Power - CSP), wat skoon, hernubar sonenergie gebruik om elektrisiteit op te wek, kan moontlik hierdie risiko's teenwerk en Suid-Afrika se afhanklikheid van steenkool verminder. Om werklik 'n beduidende rol in elektrisiteitsopwekking te kan speel, moet die koste van elektrisiteit wat deur CSP opgewek word, egter verminder word. Dit kan bereik word deur CSP-stelsels na te vors en te ontwikkel. Numeriese programmatuurgereedskap, soos straalsporing, speel 'n noodsaaklike rol in die ontwerp en optimering van hierdie stelsels.

Die doel van hierdie navorsing was om 'n CSP-straalspoorder te ontwikkel, te programmeer en te sertifiseer, wat gebruik kan word in voortgesette navorsing deur die Sontermiese Energienavorsingsgroep (STERG) by Stellenbosch Universiteit. Dié straalspoorder, wat SUNRAY (Stellenbosch UNiversity RAY tracer) genoem word, is in C++ geskryf en ontwikkel met realistiese modelle van sonvorms, refleksiemodules, 'n aantal geometriese vorms, outomatiese naspooralgoritmes, versnellingsroetines en ′n metode om werklike spieëloppervlak-profiele te simuleer deur van geskandeerde data gebruik te maak. SUNRAY is uitgebreid getoets deur middel van gedragstoetse asook vergelykende toetse tussen SUNRAY en kommersiële straalspoorders wat hulself al bewys het. Verder is dit ook getoets deur eksperimentele ondersoeke. Tydens die gedragstoetse is daar bevind dat SUNRAY daartoe in staat is om die korrekte oplossing vir elke toets te vind. In die vergelykende toetse was die relatiewe verskil tussen SUNRAY en die getoetste straalspoorders se oplossings vir krag en vloedverspreiding gemiddeld nie meer as onderskeidelik 0.4% en 3% nie. Verder was die uitvoertye vir die simulasies in meeste gevalle vinniger as dié van die gekontroleerde sagteware. In die eksperimentele kontroletoetse is verskeie spieëlvorm-profiele tydens verskillende weersomstandighede getoets. Die eksperimentele toetse het daarop gedui dat SUNRAY gebruik kan word om die omvang en verspreiding van vloed op 'n teiken te bepaal.

In hierdie tesis word die teorie onderliggend aan SUNRAY se kenmerke, algoritmes en roetines bespreek en die sertifisering van hierdie aspekte word aangebied.

Twee nuwe algoritmes word ook in hierdie tesis voorgestel. Die eerste is ′n metode wat die verlies aan strale verminder deur slegs strale bo elke voorwerp in die simulasie te genereer en die tweede is ′n algoritme wat van statistiese metodes gebruik maak om die aantal strale wat in ‘n simulasie benodig word, te voorspel.

Acknowledgements

I would like to thank the following people whose support has made this thesis possible.

1. My parents and brother for providing me with the opportunity to obtain my degree and their continued guidance, love, inspiration, and support.

2. My supervisor, Paul Gauché, I would like to express my sincere gratitude for guiding, teaching, and mentoring me throughout the entire course of this thesis and providing the resources which made the thesis possible and making it possible to present this work at in SolarPACES, Las Vegas.

3. Willem Landman for his help and collaboration with the experimental validation.

4. My colleagues in STERG and the staff of solar roof laboratory for their continued assitance.

5. Thomas Roos and Derek Griffith that CSIR for assisting me with the experimental validation

I would also like to thank The Stellenbosch University Hope Project, The Department of Science and Technology of South Africa and Sasol Technology for their financial support.

Contents

List of Figures

List of Tables

Nomenclature

Constants

$\pi =$	3.14159265358979323846264338327 95	

Variables

A	Solar azimuth angle	[rad]
A	Area	[m^2]
$A^{\perp}$	Projected area	[m^2]
c	Speed of light	[m/s]
c_i	Spectral power distribution coefficient	[−]
d	Distance	[m]
d	Discriminant	[−]
E	Error term	[W or W/m^2]
E_T	Tolerable Error	[W or W/m^2]
E	Irradiance	[W/m^2]
F_r	Fresnel reflection coefficient	[−]
$f_{r,s}$	Bidirectional reflection distribution function	[−]
H	Heliostat	[−]
I	Radiant intensity	[W/m^2sr]
i	Cell index	[−]
K	Imperfect reflection weighting term	[−]
k	Absorbtion Coefficient	[m^-1]
k_r	Reflection Coefficient	[−]
L	Radiance	[W/m^2sr]
M	Radiant Exitance	[Wm^2]
M_{sun}	Transformation matrix	[−]
M	Transformation matrix	[−]
M'	Transformation matrix due to scaling and translation	[−]
m	Number of rays required in a simulation to reduce error to an acceptable level	[−]
m_{vg}	View grid multiplying factor	[−]

m_g	Grid multiplying factor	[−]
N	Number of rays in a simulation	[−]
N	Set of independently and identically distributed numbers	[−]
$N_{initial}$	Initial number of rays used in the PRNR method	[−]
n	Number of repeated simulations	[−]
$n_{x,y}$	Number of view grid cells in the x-,y-direction of the View Grid	[−]
Q	Point on the earth surface	[−]
Q_λ	Spectral power	[W]
q	Amount of energy carried by one photon	[J]
$R_{x,y,z}(\theta)$	Rotation transformation matrix of θ about the x,y or z axis	[rad]
(R, G, B)	Red, green, blue wave lengths	[−]
r	Radius of a sphere	[m]
$r_{max,min}$	Maximum/minimum radius of a partial sphere	[−]
$r_\parallel$	Fresnel reflection for parallel light	[−]
$r_\perp$	Fresnel reflection for perpendicular light	[−]
S	Slant distance	[m]
$S_{x,y,z}$	Scaling transformation matrix in the x,y or z directions	[m]
$Sh_{x,y,z}$	Shearing transformation matrix n the x,y or z directions	[m]
s	Stadard deviation	[−]
$T_{x,y,z}$	Translation transformation matrix in the x,y or z directions	[m]
t	Ray parameter, distance along the ray from the origin	[m]
$t_{\alpha/2}$	Student $t-$distribution	[−]
u	Two dimensional parameter	[−]
v	Two dimensional parameter	[−]
w	Two dimensional parameter	[−]
$w_{x,y}$	Dimmensions of the View Grid in the x-,y-direction	[−]
x	Coordinate	[m]
Y	The optical depth of the bottomm 250 m atmospheric layer	[m]

y	Coordinate	[m]
y_i	Results for a single simulation	[W or W/m^2]
$\bar{y}$	Mean results for a set of simulations	[W or W/m^2]
Z	Normally distributed random number	[−]
$z_{0,1}$	Minimum/maximum of the global bounding box	[m]
$z_{max,min}$	Maximum/minimum z-coordiante value for a sphere	[m]
z	Coordinate	[m]

Vectors and Three Dimmensional Points

$\mathbf{a}$	A point on a three dimmensional plane	[m]
$\mathbf{a}$	First corner of of a triangle	[m]
$\mathbf{b}_{0,1}$	Lower and upper corner coordinates of a bounding box	[m]
b	Second corner coordinate of a triangle	[m]
$\mathbf{c}$	Third corner coordinate of a triangle	[m]
$\mathbf{c}_r$	Spectral base function	[m]
$\mathbf{d}$	Direction of a ray in three dimmensional space	[m]
$\mathbf{l}$	Vector pointing in direction on incomming ray	[−]
$\mathbf{N}$	Normal vector	[−]
$\mathbf{N}_f$	Distribution of surface normal with slope errors	[−]
$\mathbf{o}$	Coordinates of the origin of a ray	[m]
$\mathbf{p}$	Coordinates of a point point	[−]
$\mathbf{r}$	Vector pointin in direction of reflected ray	[−]
$\mathbf{R}$	Reflection vector	[−]
$\mathbf{S}$	Vector pointing from collector centroid toward the sun	[−]

Greek Symbols

α	Solar altitude angle	[rad]
α	Barycentric coordinate	[m]
β	Solar subtend angle	[rad]
β	barycentric coordinate	[m]
γ	Barycentric coordinate	[m]
γ	Parameter in Buie Sunshape	[−]
γ_{AE}	Elevation angle for azimuth-elevation tracking	[rad]
γ_{SE}	Elevation angle for spin-elevation tracking	[rad]

δ	Dirac delta function	[−]
ΔT	Difference between the earth's rotational and terrestrial time	[H : M : S]
η	Refractive index	[−]
θ	Bisect angle between **R** and **S**	[rad]
θ	Altitude angle	[rad]
$\theta_{max,min}$	Maximum/minimum zenith angle for a sphere	[rad]
θ_i	Angle of incidence	[rad]
θ_o	Angle of reflection	[rad]
κ	Parameter in Buie Sunshape	[−]
λ	Wave length	[m]
λ	Facing angle of target aligned heliostat	[rad]
λ_{SA}	Single axis tilt angle	[rad]
μ	Mean valude for a Normal Gaussian distribution	[−]
μ_{sun}	Mean valude for a Normal Gaussian distribution for the Gaussian sunshape	[rad]
μ_∞	Mean result for repeating a simulation infinite times	[W or W/m^2]
ν	Degrees of freedom	[−]
ξ	Uniform random number	[−]
ρ	Reflectance	[−]
ρ	Directional-hemispherical reflectance	[−]
ρ_{AE}	Azimuth angle for azimuth-elevation tracking	[rad]
ρ_{SE}	Azimuth angle for spin-elevation tracking	[rad]
σ	Standard distribution for Gaussian distribution	[rad]
$\sigma_{combined}$	Combined mirror erros	[rad]
$\sigma_{tracking}$	Standard deviation for tracking errors	[rad]
σ_{sun}	Standard deviation for Gaussian distribution for the Gaussian sunshape	[rad]
$\sigma_{spec,1,2}$	Standard deviation for specular error for imperfect specular surface	[rad]
σ_{slope}	Standard deviation for slope erros for imperfect specular surface	[rad]
$\sigma_{a,e,s}$	Abosorption, scattering or extinction coefficient	[−]
σ_n	Variance for a simulation using n rays repeated infinite times	[W or W/m^2]

τ_a	Ray transmittence	[−]
Φ	Radiant flux	[W]
Φ_{cs}	Circumsolar region of the sky	[−]
Φ_i	Incident flux from the direct beam and aureole	[W]
χ	Circumsolar ratio	[−]
ϕ	Azimuth angle	[rad]
ϕ	Sunshape azimuth angle	[rad]
ϕ_{hit}	Azimuth angle for sphere hit point	[rad]
$\phi_{max,min}$	Maximum/minimum azimuth angle for sphere hit point	[rad]
$\boldsymbol{\omega}$	Ray	[−]
Ω	Solid angle	[sr]

Subscripts

i	Inicident
macro	Macro surface error
max	Maximum
micro	Micro surface error
min	Minimum
r	Reflected
sun	Sun shape
v	View Grid cell
x	Coordinate
xmax	Maximum value in the x direction
y	Coordinate
ymax	Maximum value in the y direction
z	Coordinate

Abbreviations

2D	2-dimensional
3D	3-dimensional
AE	Azimuth-Elevation

BTDF Bidirectional Transmittance Distribution Function

BRDF Bidirectional Reflection Distribution Function

BSDF Bidirectional Scattering Distribution Function

BVH Bounding Volume Hierarchies

CCD Charged Couple Device

cdf Cumulative Density Function

CMM Coordinate Measurement Machine

CMOS Complimentary Metal-Oxide Semi-Conductor

CPC Compound Parabolic Concentrator

CSIR Council for Scientific and Industrial Research

CSR Circumsolar Ratio

CSP Concentrating Solar Power

CSV Comma-Separated Value

$C0_2$ carbon dioxide

DHI Diffuse Horizontal Irradiance

DNI Direct Normal Irradiance

DLR German Aerospace Centre

GPU Graphic Processing Unit

GUI Graphical User Interface

HPC-SA High Performance Computing - Simulation Acceleration

IEEE Institute of Electrical and Electronics Engineers

iid independent and identically distributed

IPCC Intergovernmental Panel on Climate Change

LCOE Levelised Cost of Electricity

LTE Light Transport Equation

MIT Massachusetts Institute of Technology

NREL National Renewable Energy Laboratory

pbrt Physically Based Rendering Techniques

pdf Probability Density Function

PLY Polygon File Format

PRNG pseudo-random number generator

PRNR Predict the Required Number of Rays

PROMES-CNRS Processes, Materials and Solar Energy - National Centre for Scientific Research

PS10 Planta Solar 10

rms root mean square

SASEC South African Solar Energy Conference

SE Spin-Elevation

SEADS Spatially Enumerated Auxiliary Data Structure

SNR Signal-to-Noise Ratio

SOLPOS Solar Position Algorithm

SPA Solar Position and Intensity Algorithm 2.0

SPD Spectral Power Distribution

STERG Solar Thermal Energy Research Group

SUNRAY Stellenbosch UNiversity RAY tracer

SUNSPOT Stellenbosch UNiversity Solar POwer Thermodynamic

TDT Terrestrial Dynamic Time

TIN Triangular Irregular Networks

UHC University of Houston Field Codes

USA United States of America

1. INTRODUCTION

Warming of the climate system is unequivocal... Most of the global average warming over the past 50 years is extremely likely due to anthropogenic green house gas increases (IPCC, 2013).

1.1. Background

In 2007 the United Nations Intergovernmental Panel on Climate Change (IPCC) published a detailed summary of the climate change situation (Solomon, Qin, Manning, Chen, Marquis, Averyt, Tignor & Miller (eds.), 2007). A key finding of this report was the undeniable warming of the earth's climate system. This is extremely likely due to the increase in human-made greenhouse gas concentrations (IPCC, 2013). These greenhouse gasses, such as carbon dioxide ($C0_2$), originate primarily from the burning of fossil fuels for transport and power generation (Department of Environmental Affairs, 2012).

South Africa is the largest emitter of $C0_2$ in Africa and the 13^{th} largest in the world (Marland, Boden & Andres, 2011). In South Africa the predominant source (approximately 85%) of fossil fuel $C0_2$ emissions originate from coal (Marland *et al.*, 2011). The combustion of coal for electricity production forms the bulk (approximately 77%) of South Africa's power generation (van Niekerk, 2011).

South Africa's acute reliance on coal exposes the country to risks of fuel shortage and rising fuel prices. It has been argued that South Africa has already reached its peak in coal production and that the cost of coal extraction will rapidly increase, reaching the point where coal will become uneconomical to mine (Rutledge, 2011; Hartnady, 2010). This will have a serious negative effect on the growth of South Africa's economy (Heun, van Niekerk, Swilling, Meyer, Brent & Fluri, 2010).

Reducing South Africa's dependence on coal requires the development and deployment of alternative, renewable, and sustainable forms of electricity production (Heun *et al.*, 2010). Unfortunately, most renewable energy technologies rely on intermittent natural phenomena on for power generation, which means that energy production from these sources cannot be guaranteed. In order to match renewable electricity generation to the power demand profile, a form of energy storage is required (Gauché, Backström & Brent, 2012). Of all the renewable energy technologies, Concentrating Solar Power (CSP) offers one of the most effective forms of energy storage in the form of thermal energy (Boyle, 2004).

Recognised for its potential, CSP is, internationally, the fastest-growing renewable energy technology (Behar, Khellaf *&* Mohammedi, 2013). Despite this, the cost of CSP is still significantly higher than that of coal (Gauché *et al.*, 2012). For CSP to have a substantial contribution to South Africa's electricity supply, the Levelised Cost of Electricity (LCOE) needs to be reduced considerably. This can be achieved through focused research and further development of CSP.

Four primary types of CSP technologies exist, namely: parabolic trough, linear Fresnel, dish Stirling, and central receiver (Behar *et al.*, 2013). Of these technologies, the central receiver type is just entering the growth stage of its life-cycle and offers great potential for research into technical and cost improvements (Gauché *et al.*, 2012). In central receiver systems a number of solar collectors (mirrors), known as *heliostats*, concentrate sunlight onto a central receiver at the top of a tower, as illustrated in Figure 1.1.

Figure 1.1: Gemasolar Plant, Spain (Energy, 2010)

One central receiver concept currently being investigated in the Solar Thermal Energy Research Group (STERG) at the University of Stellenbosch, is the Stellenbosch UNiversity Solar POwer Thermodynamic (SUNSPOT) cycle proposed by Kröger (2012). In the SUNSPOT cycle (Figure 1.2) the receiver absorbs the concentrated solar radiation and transfers it as thermal energy into compressed air. The air then flows through a grid-connected gas turbine and the turbine exhaust is then ducted into a rock bed thermal storage facility. Thermal energy from the rock bed is used as the energy source for a steam cycle during periods when there is no sunlight.

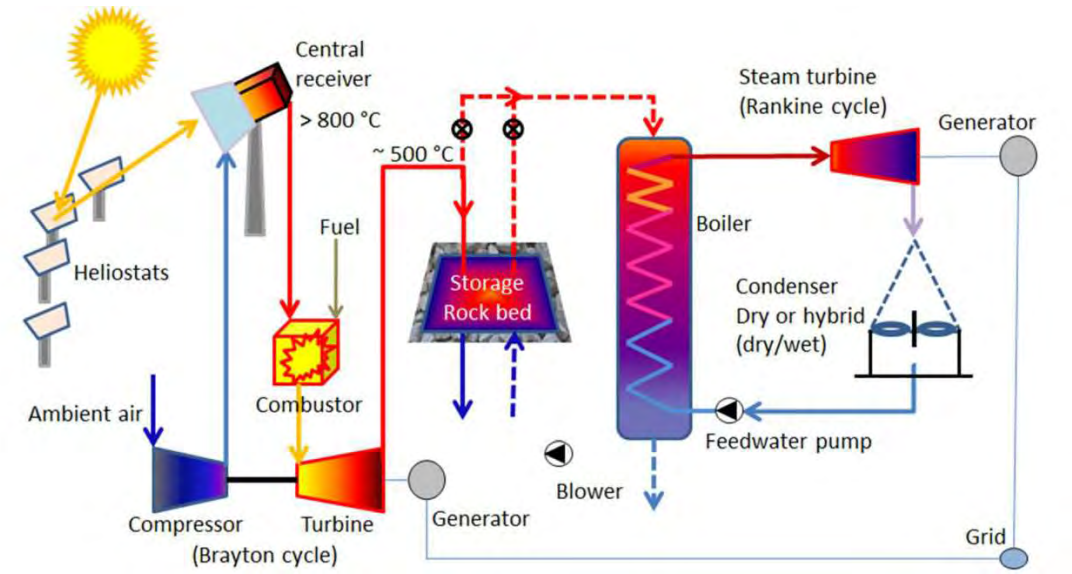


Figure 1.2: The SUNSPOT cycle (Kröger, 2012)

The transfer of energy through the SUNSPOT cycle can be simulated using numerical software tools. Numerical software tools play a critical role in CSP research and development as they can assist in design, evaluation, optimisation, and simulation of almost all aspects of a CSP system (Ho, 2008). One particular numerical method, known as Monte Carlo ray tracing, is beginning to play a predominant role in CSP optic analysis (Delatorre, Baud, Bézian, Blanco, Caliot, Cornet & et al., 2013). To aid in research and evaluation of the optics of the SUNSPOT cycle, a ray tracer has been identified as a priority research and development focus in STERG.

Monte Carlo ray tracing methods model the incoming solar radiation from collector to receiver using the laws of geometrical optics, physical optics, and radiative energy transfer (Modest, 2003). They can be used to determine the total incident power and the distribution of power on the receiver, as well as simulate various geometrical objects and the optical properties of materials (Appenidx A). Ray tracing can play a significant role in the development of the SUNSPOT cycle.

1.2. Motivation

South Africa has one of the highest solar resources in the world (Fluri, 2009) and CSP could be a good, feasible solution for power generation as well as diversify the country's energy mix (Gauché *et al.*, 2012). Through continued research and development, the LCOE can be reduced.

Several ray tracing software tools are readily available to researchers in CSP

(Section 2). However, none are known to have been developed in South Africa. In developing a ray tracer, an in-depth and comprehensive understanding of the physics and numerical procedures involved in ray tracing could be obtained. Furthermore, a ray tracer developed at STERG can be designed to be ideally suited the STERG research environment. This will accommodate future research and development of the ray tracer, which can help build a knowledge capacity of CSP optics in South Africa.

1.3. Objectives

The objective of this research project is the initial development of a comprehensive ray tracing code for CSP collector systems which can serve as an alternate to the ray tracing programs currently available. The code must be designed for improvement over time and must be capable of simulating the optical flux distribution of a CSP system, with particular focus on central receivers as part of the SUNSPOT cycle. In order to fulfil this objective, the following methodology is followed:

1. Review the optical tools and methods available to South African CSP researchers.
2. Develop the ray tracer, hereon after known as SUNRAY (Stellenbosch UNiversity RAY tracer).
3. Validate SUNRAY both theoretically (including tests with proposed and real CSP systems) and experimentally.

1.4. Delineation and Limitations

Ray tracing is a powerful method used in a range of industries. For SUNRAY to be appropriately suited for STERG, it needs to be judiciously developed specifically for CSP applications. SUNRAY therefore, needs to fulfil the following requirements:

1. Determine the coordinates of the hit points of the rays on all the objects.
2. Determine the reduction in flux as the ray is traced through the *scene* (a description of all the objects to be simulated).
3. Handle a large number of geometric objects.
4. Model the sun's position relative to any terrestrial location.
5. Model the sun's shape.
6. Track the sun (tracking algorithms must be able to model tracking errors).

7. Model a number of reflecting surfaces including ideal and real reflection.
8. Handle flux loss due to atmospheric attenuation between the heliostat and the target.

In addition to these requirements, the following criteria must be met:

1. SUNRAY needs to be as user-friendly, intuitive, and logical to use as possible.
2. SUNRAY should be developed with good programming practices to reduce computational times.
3. The ray tracing algorithm must be developed to ensure that in future it can be made to execute on parallel processors.

SUNRAY does not need fulfil the following requirements:

1. Be able to handle refraction, transparency or spit rays. However, it should be designed in such a way that these factors can be developed into SUNRAY in the future.
2. Process results internally.
3. Have a user interface.
4. Have parallel or multi-thread capabilities.
5. Be applicable beyond CSP systems.

1.5. Chapter Overview

Chapter 2 presents a comprehensive literature review on optical software tools used in CSP. Chapter 3 describes the method followed in this project. Chapters 4-7 give the details of the various features of SUNRAY. Chapter 4 describes all features with regard to the sun, such as sunshape and tracking algorithms. The details of how a ray hits an object, using a sphere as an illustrative example, are described in Chapter 5. Chapter 5 also discusses triangles, an important object in ray tracing. Ray reflections off various materials are discussed in Chapter 6. Chapter 7 describes the various acceleration techniques which are essential to reduce simulation times. In Chapter 8 all the features discussed in the preceding chapters are validated. Finally, a summary of the findings and recommendations for further work are provided in Chapter 9. The six appendices provide detailed background information and a description of the mathematics for various algorithms used in SUNRAY.

2. LITERATURE REVIEW

Ray tracing falls within a broader category of software tools known as optical codes (Ho, 2008). Optical codes are used ubiquitously throughout CSP design, optimisation, and analysis. Optical codes can be broadly divided into two categories: those used to analyse and optimise field layout and those capable of accurately simulating the flux incident on the receiver from one or more heliostats (Garcia, Ferriere & Bezian, 2008).

With the intention of fully understanding ray tracing's role in CSP systems, a comprehensive review of the optical codes, which have been specifically designed for use in central receiver systems, was performed. Similar reviews have been performed by Ho (2008), who investigated the optical tools used at Sandia National Laboratories, and Garcia *et al.* (2008), who performed a highly cited comparative overview of codes dedicated to central receivers. However, almost all the codes reviewed by Garcia *et al.* (2008) and Ho (2008) have either been upgraded or are no longer available.

A review of the optical software tools used in CSP are presented and discussed in this chapter in chronological order of development. The functionality and the availability of the optical codes are addressed throughout the discussion. A substantial portion of this chapter is based on a peer-reviewed paper, *A Review of Optical Software Used in Concentrating Solar Power*, written by the author (Bode & Gauché, 2012). Only the key findings of this paper have been presented here, but a more detailed review can be found in the paper itself.

2.1. Optical Software Tools Used In CSP

The first codes developed for central receiver systems originated from studies carried out on Solar One in the late 1970s (Falcone, 1986). Among the earlier codes are HELIOS and the well-known code, DELSOL. HELIOS uses cone optics to calculate flux density from heliostat fields (Biggs & Vittltoe, 1976) and is still used at SANDIA, although it is unsupported and difficult to use (Ho, 2012).

DELSOL, or its windows adaptation winDELSOL, is capable of predicting the optical performance of a heliostat field, optimising field layout based on energy costs, and has an economic model of central receiver components (Kistler, 1986). Delsol 3 has recently been made available for download on the SANDIA website, but is no longer supported (SANDIA, 2012).

The first generation codes were constrained by the computing power at the time and attempts were made to minimise computational resources (Falcone, 1986). With modern multi-core computer systems, there has been a paradigm change in software development toward parallel and multi-threaded programming (Fox, Williams *&* Messina, 1994). Despite this, the well-founded algorithms and routines of the first generation codes are still applicable today. Thus, the first generation codes MIRVAL, UHC, and HFCAL have been incorporated into modern software packages.

SPRAY (MIRVAL) Development of MIRVAL began in the 1970s by Sandia National Laboratories (Leary *&* Hankins, 1979) and was one of the first Monte Carlo ray tracing programs written for heliostat optical performance simulation (Falcone, 1986). Development of MIRVAL has been taken over by the German Aerospace Centre (DLR) where it forms the core of a FORTRAN-based (The Fortan Company [online]) code called SPRAY (Buck, Pfahl *&* Roos, 2012). SPRAY can be used to calculate field efficiencies as well as flux maps from individual heliostats or fixed heliostat fields. One of its main advantages is that it has a large number of built-in features and geometries due to its long history of usage (Schwarzbözl, 2012). SPRAY is not user-friendly as it does not have a user interface and is operated via ASCII files (Schwarzbözl, 2012). SPRAY is commercially available through the DLR's Institute of Solar Research and may also be made available through academic collaborations (Schwarzbözl, 2012).

University of Houston field Codes (TieSOL) The University of Houston Field Codes (UHC), or RCELL codes, are a suite of four codes which deal mainly with the optical design and optimisation of heliostat fields and receivers (Falcone, 1986). These codes are no longer in a deliverable condition, but they are being updated by the software company Tietronix who are developing them into a commercial package known as TieSOL (Vant-Hull, 2012). The date of release of TieSOL with field layout capabilities is currently unknown. Although the UHC are no longer available, their concepts and procedures have been extensively documented in several publications, for example (Vant-Hull *&* Izygon, 2003, 1998)

HFLCAL The HFLCAL code was developed to perform two main tasks: the calculation of the annual plant output at a given configuration and the layout and optimisation of a total system (Kiera, 1989). Today it continues to be used and developed by the DLR for the layout and optimisation of heliostat fields. The software uses a simplified mathematical model of concentrator optics, modelling the reflected image of each heliostat by a circular normal distribution (Schwarzbözl, Schmitz *&* Pitz-paal, 2009). This has a tremendous computational speed advan-

tage over traditional ray tracing, although at the cost of accuracy (none of the codes reviewed employed Monte Carlo methods for field optimisation). HFLCAL features include automatic multi-aiming, secondary concentrator optics, tower reflector systems, various receiver models, and the ability of least-cost optimisation (Schwarzbözl *et al.*, 2009). HFLCAL is also commercially available through the DLR and academic collaborations (Schwarzbözl, 2012).

ISOS ISOS was developed to improve the durability of receivers. It uses a numerical algorithm to calculate regions of homogeneous flux (isosurfaces). A three dimensional flux map is generated, which allows the user to assess the flux at any height above the focusing heliostat (Riveros-Rosas, Sánchez-González *&* Estrada, 2008). The code is available for academic use but requires input data from a separate ray tracing program (Riveros, 2012).

SolTrace SolTrace is a Monte Carlo ray tracing program developed at the National Renewable Energy Laboratory (NREL) United States of America (USA). In 2011 SolTrace was completely rewritten. The latest C++ version uses parallel processing techniques which, in theory, for a computer with n processors will experience a speed of nx over a single processor (NREL, 2012*b*). SolTrace can model and characterise the optics of a single heliostat or, with the aid of a built-in scripting language, be used to model large optical systems. SolTrace is downloaded as an executable file and the inner workings of the software are unknown to the user. Without knowledge or an understanding of how the code works, it is difficult to identify sources of possible errors in simulated results. SolTrace is freely from the NREL website (NREL, 2012*a*).

Tonatiuh Tonatiuh is another freely available CSP Monte Carlo ray tracer (Blanco, 2013). Similar to SolTrace, Tonatiuh has several geometric heliostat and receiver shapes (Blanco, Mutuberria, Monreal *&* Albert, 2011). Furthermore, it has a built-in visualiser to view the placement of objects within a scene. However, flux maps need to be generated with an external program. Tonatiuh, unlike SolTrace, has a GNU General Public License which allows free access to its source code for anyone interested in using it or contributing to its development (The GNU General Public License v2 [online]). However, the source code was found to be poorly documented with few comments.

STRAL STRAL is a ray tracer which generates rays on the surface of the heliostats. Because rays are generated on the heliostats, no rays are wasted, which makes the process computationally efficient. STRAL enables the setup of heliostat

field models in great detail using highly resolved heliostat mirror surface and geometry data as well as real sunshapes, blocking and shading (Pitz-Paal, Schwarzbözl *&* Ulmer, 2009). STRAL is commercially available through the DLR as well as through academic collaborations (Schwarzbözl, 2012).

TieSol The TieSol suite uses the parallel processing power of Graphic Processing Unit (GPU) to implement extremely fast Monte Carlo ray tracing (Izygon, Armstrong, Nilsson *&* Vu, 2011). Tietronix has developed an advanced visualisation tool for TieSol capable of displaying heliostat tracking in real-time. TieSOL is commercially available from Tietronix and is sold as individual modules (Izygon, 2012).

Heliostat Field Layout Design (HFLD) HFLD is a field-layout design code based on the edge-ray principle of non-imaging optics (Wei, Lu, Wang, Yu, Zhang *&* Yao, 2010*a*). The edge-ray principle states that if the limiting rays (rays coming from the edges of the source) hit the receiver, then all rays coming from the inner points will also hit the receiver (Ries *&* Rabl, 1994). When compared with other codes, such as winDELSOL, HFLD has a shorter computational time during design and optimisation of the heliostat field. The code also calculates the magnitude and duration of annual sunshine on the land surface between heliostats, which can be used to evaluate the feasibility of crop growth (Wei, Lu, Yu *&* Wang, 2010*b*). The HFLD code can be purchased from the authors for commercial or academic use (Wei, 2012).

CRS4-2 CRS4-2 is a FORTRAN-based code used for the simulation of optical performance of central receiver systems. For an arbitrarily arranged field it can calculate the effects of blocking, shading, and cosine loss through tessellation of the heliostats (Leonardi *&* D'guanno, 2011). CRS4-2 is currently not available but discussions are under way to allow sharing or collaborations with external institutes (Leonardi, 2012).

'Biomimetic' In an altogether different approach, the 'biomimetic' code uses a biomimetic pattern for heliostat field layout optimisation (Noone, Torrilhon *&* Mitsos, 2012) (biomimicry is the emulation of nature in man-made structures (Vincent, Bogatyreva, Bogatyrev, Bowyer *&* Pahl [2006]). Using the PS10 plant as a demonstration application of this code, the biomimetic model showed a 0.36% improvement in efficiency on existing configurations with a 15.8% reduction in land usage (Noone *et al.*, 2012). Currently the code is under pending patent and unavailable. However, the developers are willing to licence the code under conditions

decided by the Massachusetts Institute of Technology (MIT) Technology Licensing Office (Mitsos, 2012).

Solfast Solfast is a Monte Carlo ray tracer developed in partnership with the High Performance Computing - Simulation Acceleration (HPC-SA) and Processes, Materials and Solar Energy - National Centre for Scientific Research (PROMES-CNRS). Solfast uses a Monte Carlo algorithm which is computationally faster than Tonatiuh and SolTRACE (Roccia, Piaud, Coustet, Caliot, Guillot, Flamant *&* Delatorre, 2012). Solfast can be made available for research and collaboration purposes (Coustet, 2013).

EDStar The most recent code to have been developed is EDStar. EDStar has only been publicised recently and, therefore, was not reviewed in the original paper by Bode *&* Gauché (2012). EDStar facilitates the task of analysing complex systems by using corpuscular transport models. Coupled with a number of additional programming libraries and data banks, it is capable of resolving most CSP research problems with regard to radiative transfer simulation. Their website is under construction but, when completed, will allow the EDStar to be downloaded with accompanying CSP-specific simulation examples (Delatorre *et al.*, 2013).

Other Codes Other codes investigated by Garcia *et al.* (2008), namely Fiat Lux, OPTEC, SOLVER, SENSOL, and SCT, could not be obtained, nor could the authors of these codes be reached. Furthermore, as no recent publications could be found on these codes they were excluded from this review.

2.2. Conclusions

While there are a number of methods for flux prediction, Monte Carlo ray tracing methods remain the preferred numerical simulation techniques due to their flexibility and their ability to deal with complex geometries (Delatorre *et al.*, 2013). Two Monte Carlo ray tracers are freely available, however these have their limitations. This suggests that there is scope for new or improved well documented tools to be developed. This will also allow the codes to be developed further.

In the next section an overview of the the procedures and algorithms used in SUNRAY are discussed.

3. METHOD

Ray tracing is a field of optical physics known as geometrical optics or ray optics (Goodman, 2004). This chapter introduces the basic fundamentals of ray tracing and Monte Carlo methods as well as the basic ray tracing algorithm used in SUNRAY. Finally, programming and the programming methodology, which formed a major component of this thesis, is then introduced.

3.1. Ray Tracing Methods for Flux Prediction

Ray tracing is defined as the process of calculating the geometric paths of rays and the flow of radiant energy through a scene (Whitted, 1980). The principles of ray tracing are based in electromagnetic radiation theory and have been used in a wide range of applications, ranging from radiative heat transfer (Siegel & Howell, 1992) to the evaluation of wireless antenna systems (Dandekar, Arredondo, Xu & Ling, 2002). Pioneering work by Spencer & Murty (1961) provided one of the first unified ray tracing procedures applicable to systems of a general type.

One of the greatest advances in the development of ray tracing has been for use in computer graphics (Heckbert, 1994). The goal of graphical ray tracers is to render a photo-realistic image on a computer screen (Glassner, 1989). Graphical ray tracers are viewer-dependant and rays are propagated from pixels in the computer screen toward a number of objects (Heckbert, 1994). If a ray hits an object, information about that object, such as colour, is sent back along the ray and the pixel is coloured accordingly. This simple ray casting technique was developed by Appel (1968) in the 1960s.

Extending on the ray casting technique, Whitted (1980) developed a global illumination model which recursively traces a ray as it is reflected through a scene. Information is sent back along a ray-*tree* to the pixel. Whitted's algorithm is limited in that it only accounts for perfect specular reflection. Therefore, building on Whitted's algorithm, Kajiya (1986) developed the renowned rendering equation (Section D.5.1), which is able to compute diffuse reflection.

The methods and the fundamental theories of light transport used in graphical ray tracing, such as Kajiya's rendering equation, are the same for CSP systems. Moreover, because a significant amount of research has been done on graphic-based ray tracing, many of the routines and algorithms of SUNRAY have been based on those developed for computer graphics.

3.1.1. The Ray

A ray is a geometrical description of a physical situation. It is a convenient means of describing light as both a wave and a photon. In wave theory rays can be defined as normals to a wavefront and in photon theory a ray refers to the path of the photons (Jenkins & White, 1957). There are, however, shortcomings in describing light in terms of rays. For example, there are situations where describing light as wavefront normals can complicate calculations or, in photon theory, it is possible that the energy density of photons can become infinite (Jenkins & White, 1957).

With these limitations in mind, the approach of SUNRAY is to model a ray as a semi-infinite vector pointing in the direction of energy (photon) flow. Although, even with this approach, it is not possible to describe light as a pure mathematical entity and concepts from wave or photon theory cannot be excluded altogether. For the most part the simplicity of rays will, however, compensate for its inaccuracies.

A ray is defined by its origin, **o**, a point in 3D-space, and direction, **d**, a 3D-vector (Suffern, 2007), as illustrated in Figure 3.1

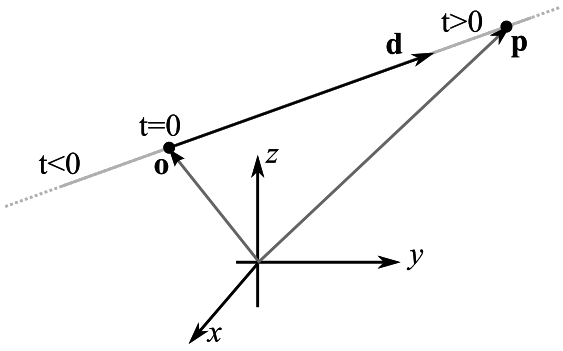


Figure 3.1: Definition of a ray

A ray is parameterised with the ray parameter t, where the value of t increases in the direction of the ray. The parametric form gives a set of points that the ray passes (Pharr & Humphreys, 2010).

$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d} \qquad 0 \leq t \leq \infty \tag{3.1}$$

An arbitrary point on the ray can be expressed as

$$\mathbf{p} = \mathbf{o} + t\mathbf{d} \tag{3.2}$$

3.1.2. Ray-Object Interaction

Mathematically, surfaces can be described by implicit equations or parametric equations. Rays can interact with each type of surface.

Implicit Equation

Implicit equations have the form (Stewart, 2003)

$$f(x, y, z) = 0 \tag{3.3}$$

Any point, $\mathbf{p} = (x, y, z)$, when given as an argument to f, will return a value of zero if it is on a surface and another value if it is not. Given a ray, $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, and an implicit surface, $f(\mathbf{p}) = 0$, an intersection occurs when the points on the ray satisfies the implicit equation

$$f(\mathbf{r}(t)) = f(\mathbf{o} + t\mathbf{d}) = 0 \tag{3.4}$$

An example of this is the intersection with an infinite plane. For a plane with normal, $\mathbf{N}$, which passes through a point, $\mathbf{a}$, the implicit equation is given in (John, 2005) as

$$(\mathbf{p} - \mathbf{a}) \cdot \mathbf{N} = 0 \tag{3.5}$$

Where $\mathbf{p}$ is any unknown point that satisfies the plane's implicit equation. Inserting Equation (3.1) into Equation (3.5) and solving for the only unknown t

$$t = \frac{(\mathbf{a} - \mathbf{o}) \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}} \tag{3.6}$$

The solution to this equation is illustrated in Figure 3.2. There is only one solution for t which satisfies Equation (3.6), surface C. A denominator with a value of zero corresponds with a ray which is parallel to the plane and perpendicular to the normal, surface B, and a negative solution corresponds to an intersection behind the ray's origin, surface A.

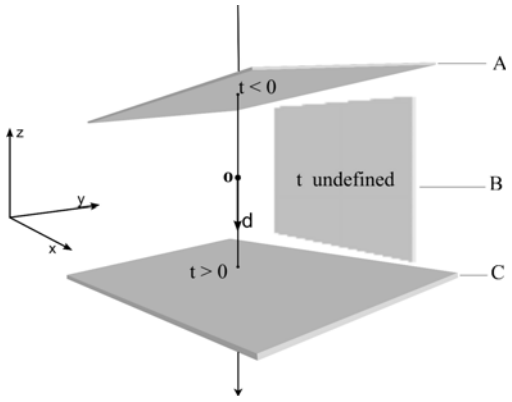


Figure 3.2: Ray-plane intersection

Parametric Surface

Another way to represent a 3D surface is with a parametric equation. A parametric equation is a function which maps the two-dimensional 2-dimensional (2D) parameters to 3D points on the surface. These surfaces have the form (Stewart, 2003)

$$\begin{aligned} x &= f(u,v) \\ y &= g(u,v) \\ z &= h(u,v) \end{aligned} \tag{3.7}$$

The parametric form is particularly useful for calculating surface normal. The normal vector at point $\mathbf{p} = (x, y, z)$ for a parametric surface is computed from the cross product (John, 2005)

$$\mathbf{n}(u,v) = \left(\frac{\partial f}{\partial u}, \frac{\partial g}{\partial u}, \frac{\partial h}{\partial u}\right) \times \left(\frac{\partial f}{\partial v}, \frac{\partial g}{\partial v}, \frac{\partial h}{\partial v}\right) \tag{3.8}$$

Where $\frac{\partial f}{\partial u}$ is the partial derivative of $f(u, v)$ with respect to u. Similar calculation is performed for v.

3.1.3. The Ray Tracing Algorithm

Conceptually SUNRAY is divided into three phases of execution: the build phase, the main ray tracing algorithm, and post processing. In the build phase SUNRAY reads a scene description file. This is a text file, provided by the user, describing all the geometric objects, the location of the objects, and the materials of the objects to be simulated. The file also contains a description of the sun's position, the sunshape as well as any tracking algorithms chosen by the user.

After the scene has been built, SUNRAY executes the main ray tracing algorithm. Development of this algorithm has been the main focus of this thesis. Illustrated in Figure 3.3, the algorithm firstly propagates rays into the scene. Each ray is then tested for intersection with objects in a scene. If a ray hits an object, it continues through the scene until it is either absorbed by the target or exits the scene. The algorithm continues to run until a stopping criterion is met: either a desired number of rays have been traced or a tolerable error has been achieved. The output of a the ray tracing loop is a Comma-Separated Value (CSV) file with all the results from the trace. The final stage of execution is the post processing of these results.

All objects in a scene have a material property. The material property will affect the direction of the reflected ray as well as the magnitude of energy transported by the ray. To describe random light scattering in SUNRAY, Monte Carlo methods are used.

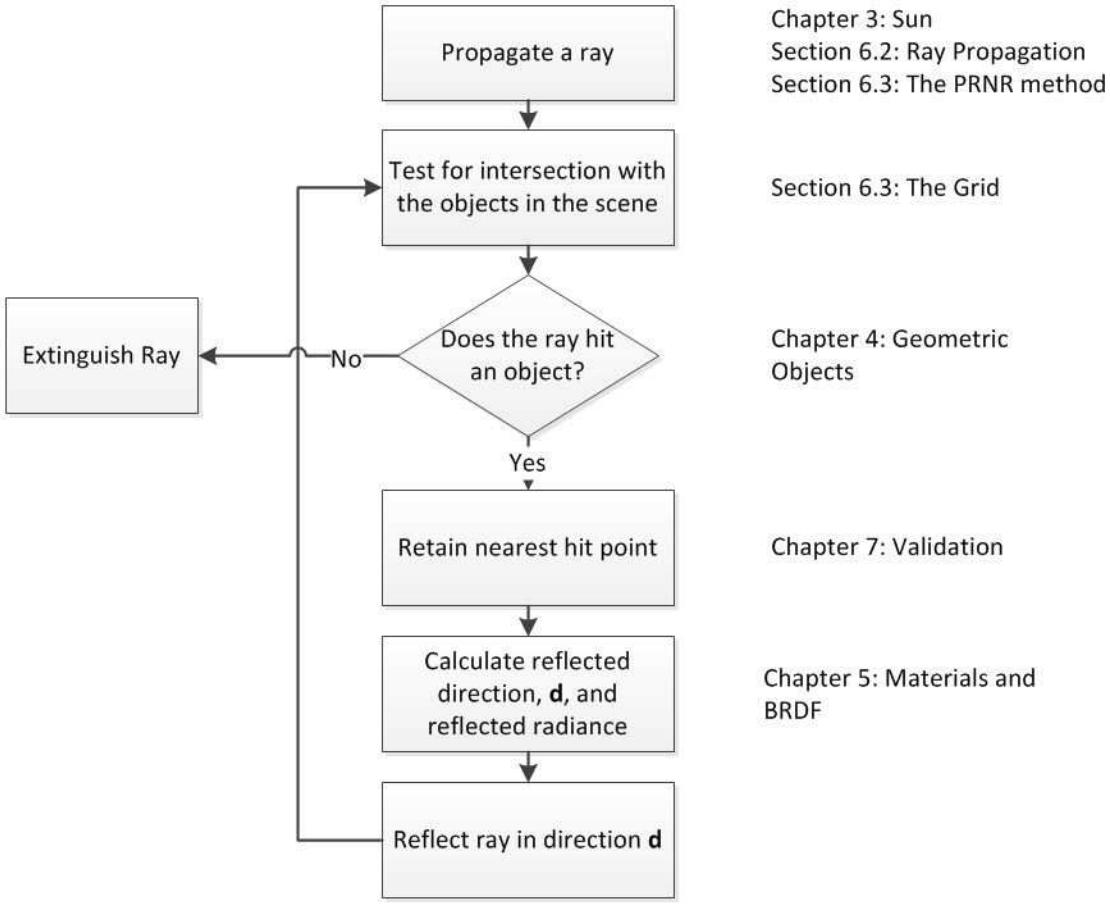


Figure 3.3: The main ray tracing algorithm

3.2. Monte Carlo Methods

Monte Carlo methods are a broad class of computational algorithms which use random sampling to solve numerical problems. They are used in many fields of science, engineering, statistics, and finance (Kroese, Taimre & Zdravko, 2011). Supporting theory to Monte Carlo integration and random numbers is given in Appendix A. Monte Carlo methods are used in ray tracing to solve the following two problems:

Problem 1: Generate samples from a given probability distribution $p(x)$.

Problem 2: To numerically estimate integrals.

Monte Carlo integration solves the integrals which have no analytic solution and are independent of the dimensionality of the integrand. Therefore they are used in ray tracing where high dimensional and discontinuous integrals are often found (Shirley & Marschner, 2009).

3.3. Programming and Development Methodology

When programming SUNRAY, the focus was on reducing computational times and ensuring that SUNRAY can be easily developed in future research. In order to achieve this, methods for efficient memory usage, effective design of algorithms, and logical abstraction of the object-orientated class structure was incorporated into the design. It is beyond the scope of this thesis to discuss the low-level details of the program and instead the validation cases presented in Chapter 8 verify that correct programming practices were followed. Furthermore, the code has been extensively commented and is intended to be made publically available pending the approval of Stellenbosch University's Technology Transfer office.

3.3.1. Programming Languages

SUNRAY was written in C++, which is currently one of the most popular programming languages (Welton, 2011). It is an intermediate language with both low-level and high-level components (Stroustrup, 1995). C++ is an object-orientated[1] programing languages which offers a more versatile environment to improve the performance of the code when compared with higher level languages such as MATLAB. It does however require considerably more development time compared to higher level languages (Schildt, 2002). Post-processing of the code was, however, conducted in MATLAB due to its visualisation capabilities.

3.3.2. Parallel Programming

In ray tracing, every ray can be traced independently from the next (Fox *et al.*, 1994). This makes ray tracing very suitable for parallel programming. It was not within the objectives of this study to write SUNRAY with parallel capabilities, but SUNRAY was developed in such a way that it could be made parallel in future research. Freisleben, Hartmann & Kielmann (1997) gives a description of various methods of parallelisation in ray tracing. The design approach adopted by SUNRAY is to distribute rays among multiple processors. Each processor traces a group of rays and the results are accumulated at the end of the simulation. This approach has a higher memory usage than other methods described in (Freisleben *et al.*, 1997), but it simplifies programming.

[1] Object-orientated programming allows for use classes (Stroustrup, 1995). In this thesis, classes and data members are refered to by proper nouns, for example the View Plane introduced in Chapter 4

3.3.3. Method and Algorithm Sources

SUNRAY was developed using a number of sources. The initial framework of SUNRAY was based on a basic ray tracer described by Suffern (2007). For programming convenience, some sections of Suffern's code, in particular some math routines, have been used directly in SUNRAY. Similarly, some small segments of the Tonatiuh source code have also been used in SUNRAY. Under the GNU licence the original author of any section of code is accredited and any modifications to the code are conspicuously and appropriately noted.

Several other documents and publications, for example (Pharr & Humphreys, 2010), provided methods and algorithms that were not used directly, but were adapted and re-programmed into SUNRAY in order to maintain a programming standard throughout code. In some instances, entire code sections that can be downloaded have been seamlessly integrated into SUNRAY, for example, the sun position algorithm (Appendix B.1). However, to prevent bugs into the program this has been avoided as far as possible.

3.3.4. Coordinate System

A crucial component of a good ray tracer is the description of the coordinate systems (Spencer & Murty, 1961). SUNRAY uses a global right-hand Cartesian coordinate system with z pointing vertically up. Within the world or global coordinate system, there are several additional local coordinate systems of arbitrary orientation. A number of transformations are required to correctly describe a ray or object within the world coordinate system.

Unlike most ray tracers, SUNRAY calculates all local coordinates and the user only needs to be concerned with the world coordinates. This is to improve user-friendliness and to avoid confusion.

3.4. Conclusions

The choice of programming SUNRAY in C++ was based on the speed of execution and versatility of this programming language and was thought to be more valuable than the quick prototyping approach of higher level codes. Furthermore, using C++ ensured that SUNRAY could be developed with well-structured abstract classes and a robust foundation to ensure extensibility.

The ray tracing procedure and recursive algorithm described in this chapter forms the core engine of SUNRAY. The following chapters address, in more detail, how rays are traced from the sun through a scene.

4. THE SUN

The sun plays a central role in CSP and in SUNRAY it defines how rays are propagated into a scene. In ray propagation, rays are generated on a plane set above the scene. This plane, known as the View Plane, is set perpendicular to a vector, known as the Sun Vector, pointing from the centre of the world coordinate system toward the sun. Rays are generated almost parallel to the Sun Vector but are perturbed by a certain amount to account for the sunshape. In this chapter the various sunshapes are described as well as the methods SUNRAY uses to track the sun. Methods for determining the sun's position and the Sun Vector are discussed in Appendix B.1 and a full description of ray propagation is provided in Section 7.2.

4.1. Sunshape

Due to it finite size, the sun, as observed from earth, is seen as a disc. Using about 9 000 observations Puliaev, Penna, Jilinski & Andrei (2000) determined the mean semi-diameter of the sun to be 959".13 ± 0".01 or 0.0465mrad. However, local atmospheric conditions have the effect of transferring some of the solar energy from within the solar disc to the circumsolar aureole, which can cause a broader distribution of solar energy (Buie & Monger, 2004). The consequence of this is an overestimation of power if all the power is assumed to fall within the solar disc (Buie, Monger & Dey, 2003*b*).

The Circumsolar Ratio (CSR) is defined as the ratio of the radiant flux contained within the circumsolar region of the sky, Φ_{cs}, to the incident flux from the direct beam and aureole.

$$\chi = \frac{\Phi_{cs}}{\Phi_i} \tag{4.1}$$

Even without the influence of the terrestrial atmosphere, the solar radiance decreases with angular distance from the centre of the sun. This is referred to as limb darkening (Wilbert, Reinhardt, Devore, Röger & Gueymard, 2011). Therefore, a number of sunshape models have been included in SUNRAY to emulate the various sunshapes which could be found in real systems. There are currently three sunshapes in SUNRAY: the pillbox, the Gaussian sunshape, and the user-defined sunshape. The Buie, which is an inherited class of the user-defined sunshape model, is also included.

For simplicity all these sunshapes assume rotationally symmetric distributions, which, for most situations, is a feasible assumption (Schubnell, 1992). However, it should be noted that Neumann & Witzke (1999) found that for high CSR values, which are sometimes caused by irregular clouds, the brightness distribution is not rotationally symmetrical.

4.1.1. Pillbox

The most simple and ideal sunshape is one that models a constant intensity across the extent of the solar disc and zero elsewhere (CSR=0%). This is known as a pillbox sunshape. Using a pillbox sunshape can lead to an overestimation of the total power on the receiver, although Schubnell (1992) and Johnston (1998) showed that for some simulations it can be used with little error.

The input into the pillbox function is half the sun's subtend angle β (by default $\beta = 0.0465$ mrad). The pillbox function uses Monte Carlo methods to perturb the Sun Vector by uniformly sampling random points within the solid angle formed by this subtend angle, as shown in Figure 4.1. The derivation of the multi-dimensional transformation for the pillbox sunshape is given in Appendix B.1.3 and the results are given in Equation (4.2)-(4.4).

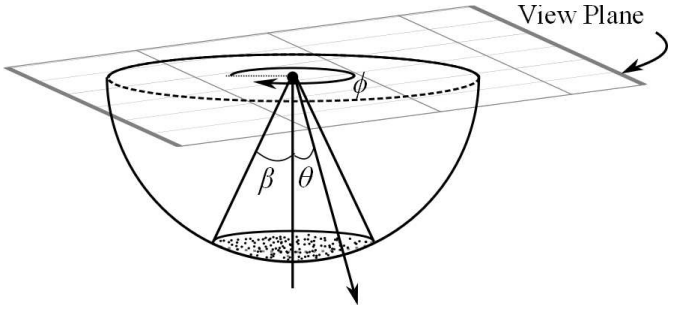


Figure 4.1: Angles used in the pillbox sunshape

The direction of the ray, $\mathbf{d}$, propagated into the scene is given in terms of Cartesian coordinates as a point on the unit hemisphere where

$$\begin{aligned} x &= \sin\beta\cos\phi \\ y &= \sin\beta\sin\phi \\ z &= \cos\beta \end{aligned} \tag{4.2}$$

For a given solar subtend angle, β, the angles, ϕ, θ are found from

$$\phi = 2\pi\xi_1 \tag{4.3}$$

$$\theta = \arccos(1 - (1 - \cos(\beta))\xi_2) \tag{4.4}$$

Where ξ is a uniform random number.

4.1.2. Gaussian Sunshape

In the Gaussian sunshape the radial drop in intensity across the solar disc is described by a Gaussian distribution. The Gaussian sunshape has been included in solar ray tracers such as SolTrace (NREL, 2012*b*) and Solfast (Roccia *et al.*, 2012) and therefore it has been included in SUNRAY for completeness. However, Neumann, Witzke, Jones & Schmitt (2002) show that modelling the sunshape as a Gaussian distribution is a poor representation of reality and should only be used when other errors within the system are substantial.

In the Gaussian sunshape the solar subtend angle is represented by three standard deviations, $\beta = 3\sigma_{sun}$. Thus, approximately 99% of the rays will be contained within the solar half angle and about 68% will be contained within one third of the solar half angle (Lyman Ott, 1988).

In order to generate random samples between the interval (0-3σ_{sun}) the Box-Muller function is used (Appendix A.2.2). This function generates a set of normally distributed random numbers, Z, between the interval $(0, 1)$. Thus, before Z can be used, it needs to be transformed between Gaussian curves. This is done through (Lyman Ott, 1988)

$$Z' = Z\sigma_{sun} + \mu_{sun} \tag{4.5}$$

Assuming an axisymmetric sunshape, μ_{sun} is always zero. The Cartesian coordinates are found in the same manner as in the pillbox sunshape (Section 4.1.1) except that the normally distributed random number Z' replaces ξ_2 in Equation (4.4).

4.1.3. User-Defined Sunshape

The third sunshape profile is the user-defined sunshape. In this profile the sun's intensity is modelled using a series of discrete, regular data points, which are provided by the user in CSV file format. The user defines the angular displacement from the centre of the sun and the corresponding flux intensity at that displacement. In order to draw a random number from the user-defined distribution, the Monte Carlo inversion method is used (Kroese *et al.*, 2011). In this procedure the user-defined CSV file is scaled so that the domain is defined over the interval $[0, 1]$. The User-Defined function then generates random numbers between the interval $[0, 1]$ which have the same distribution as the user-defined profile. The complete procedure is described in Appendix B.1.4. An example of a user-defined sunshape is the evidence-based sunshape model developed by Buie *et al.* (2003*b*).

Buie Sunshape Model

The Buie model is the most realistic sunshape model in SUNRAY. Using a vast number of measured sunshape profiles Buie *et al.* (2003*b*); Buie, Dey & Bosi

(2003*a*); and Buie & Monger (2004) developed an accurate sunshape model which is independent of geographical location. The formula for the Buie sunshape model is provided in Appendix B.1.5.

The inputs to the Buie model are the circumsolar ratio, the number of data points and the maximum angular displacement away from the centre of the sun. For convenience, SUNRAY automatically generates the CSV file which is read by the user-defined sunshape.

4.2. Tracking

In order to compensate for the apparent rotation of the sun around the earth and to ensure the sun's rays are reflected onto the target at all times, both point focus and line focus collectors need to track the sun. Rotation can be about a single-axis or about two axes and tracking systems are classified by their motions (Duffie & Beckman, 2006). In SUNRAY three methods of tracking have been incorporated; two dual-axis tracking methods and one single-axis tracking method. Only dual-axis tracking is discussed in this section and the single-axis tracking formulas can be found in Appendix B.2.3. The two dual-axis sun tracking methods are the conventional Azimuth-Elevation (AE) and the revolutionary Spin-Elevation (SE) methods (Chong & Tan, 2011).

4.2.1. Azimuth-Elevation Tracking

The most common dual-axis tracking method used in central receiver plants is AE tracking (Guo, Wang, Zhang, Sun & Zhang, 2011). In AE tracking one of the tracking axes of the heliostat, the azimuth axis, points toward the zenith while the other axis, the elevation axis, is perpendicular to the first (Chong & Tan, 2011). In AE tracking the normal vector of the heliostat, $\mathbf{N}$, must bisect the vector pointing toward the target, the reflection vector, $\mathbf{R}$, and the Sun Vector $\mathbf{S}$. This is illustrated in Figure 4.2.

The formulas of sun-tracking angles for AE methods applied in SUNRAY are given below. These formulas have been re-derived to fit within SUNRAY's coordinate system. From Figure 4.2, the elevation, γ_{AE}, and azimuth, ρ_{AE}, angles are expressed as

$$\gamma_{AE} = \arcsin\left(\frac{R_z + \sin\alpha}{2\cos\theta}\right) \tag{4.6}$$

Given that

$$\cos\rho\prime = \frac{\cos\alpha\cos A - R_x}{2\cos\theta\cos\gamma_{AE}} \tag{4.7}$$

For $\cos \rho\prime > 0$

$$\rho_{AE} = \arcsin\left(\frac{\cos\alpha \sin A - R_y}{2\cos\theta\cos\gamma_{AE}}\right) \tag{4.8}$$

For the case of $\cos \rho\prime \leq 0$

$$\rho_{AE} = \pi - \arcsin\left(\frac{R_y - \cos\alpha \sin A}{-2\cos\theta\cos\gamma_{AE}}\right) \tag{4.9}$$

Where

$$\theta = \frac{1}{2}\arccos(-R_x \cos\alpha \cos A - R_y \cos\alpha \sin A + R_z \sin\alpha) \tag{4.10}$$

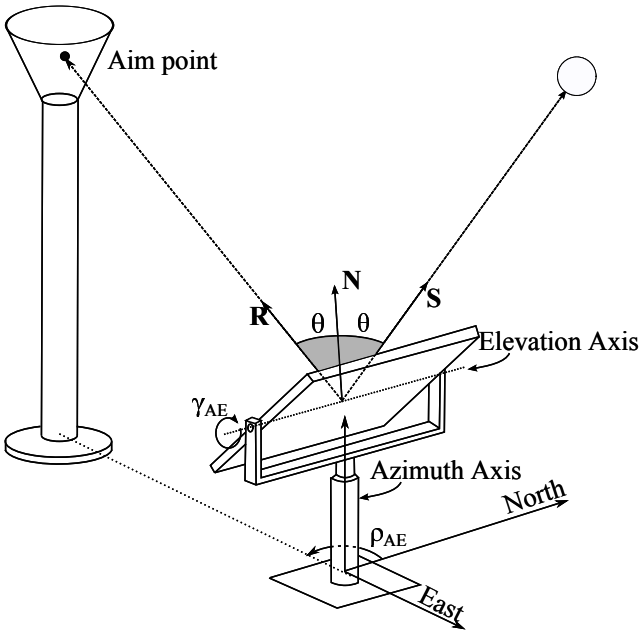


Figure 4.2: Azimuth-elevation tracking adapted from (Chong & Tan, 2011)

In AE tracking the incidence angle on the heliostat changes considerably with time and the heliostat suffers strong astigmatic aberration losses whenever the reflection is off-axis (Kribus & Ries, 2003). This leads to an increase of the sun's image size on the target, which results in considerable spillage losses and a reduction in the average flux on the receiver. This then ultimately leads to a reduction in receiver efficiency (Karni, Buck, Pfahl, Bligh & Chen, 2004).

In a widely cited theoretical study conducted by Igel & Hughes (1979), it was found that the amount of aberration depends on the incidence angle measured in the plane made by the sun, the centre of the heliostat and the tower, called the tangential plane, as illustrated in Figure 4.3. These authors found that, in order to correct for the astigmatism, the heliostat should have a different radius of curvature

along the tangential plane and the plane perpendicular to the tangential plane, the sagittal plane. By reducing the linear dimension of the heliostat, the size of the image produced is compatible with the receiver aperture. For AE tracking, the sagittal and tangential planes rotate throughout the day and therefore astigmatic correction is only possible for a short moment per day. Several studies have been conducted in an attempt to reduce this loss which lead to the development of the Spin-Elevation tracking method.

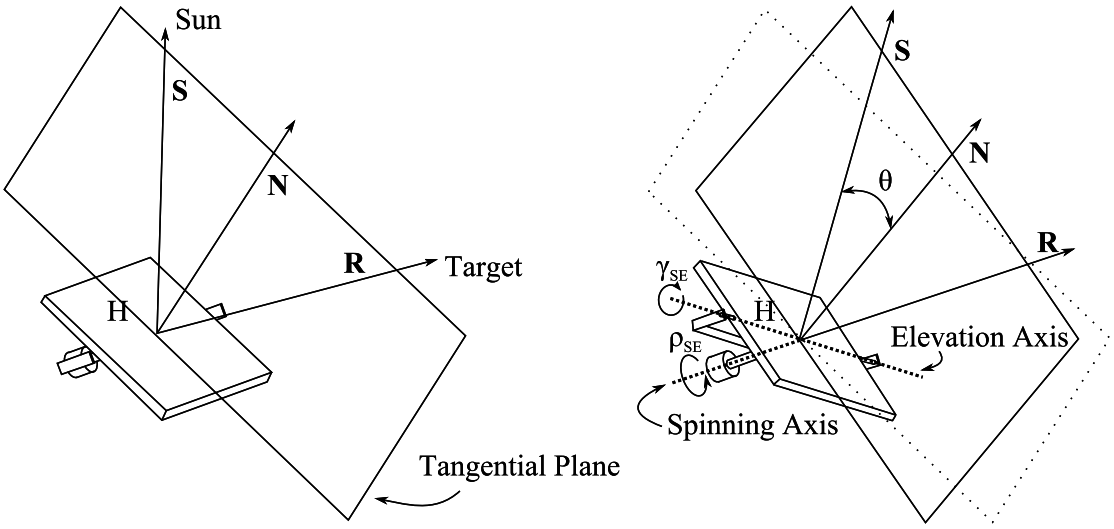


Figure 4.3: Spin-Elevation optics adapted from (Tam & Tan, 2001)

4.2.2. Spin-Elevation Tracking

Ries & Schubnell (1990) and Zaibel, Dagan, Karni & Ries (1995) describe the mounting system for a non-symmetric heliostat with two different radii of curvature. In this system the sagittal and tangential directions remain stationary with respect to the heliostat frame. This requires that the first rotational axis, the spinning axis, points towards the target in order to maintain the heliostat normal in the tangential plane, as shown in Figure 4.3. The second axis, the elevation axis, is set perpendicular to the first and parallel to the heliostat frame. This makes it possible to adjust the heliostat normal within the tangential plane until it bisects **S** and **R**. No information describing the implementation of SE tracking in any of the CSP ray tracer could be found.

Using the sign convention in Figure 4.4, the elevation, γ_{SE}, and spinning,ρ_{SE}, tracking angles for Spin-Elevation (SE) tracking have been derived as

$$\gamma_{SE} = \frac{\pi}{4} - \frac{1}{2}\arcsin\left(R_z \sin\alpha - R_y \cos\alpha \sin A - R_x \cos\alpha \cos A\right) \tag{4.11}$$

Given that

$$\cos\rho\prime = \frac{\cos\lambda\sin\alpha + R_z\sin\phi\cos\alpha\sin A - R_z\cos\phi\cos\alpha\cos A}{\cos\left(\frac{\pi}{2} - 2\gamma_{SE}\right)} \tag{4.12}$$

For the case of $\cos\rho\prime > 0$

$$\rho_{SE} = \arcsin\left(\frac{\sin\phi\cos\alpha\cos A - \cos\phi\cos\alpha\sin A}{\cos\left(\frac{\pi}{2} - 2\gamma_{SE}\right)}\right) \tag{4.13}$$

For the case of $\cos\rho\prime \leq 0$

$$\rho_{SE} = \pi - arcsin\left(\frac{\sin\phi\cos\alpha\cos A - \cos\phi\cos\alpha\sin A}{\cos\left(\frac{\pi}{2} - 2\gamma_{SE}\right)}\right) \tag{4.14}$$

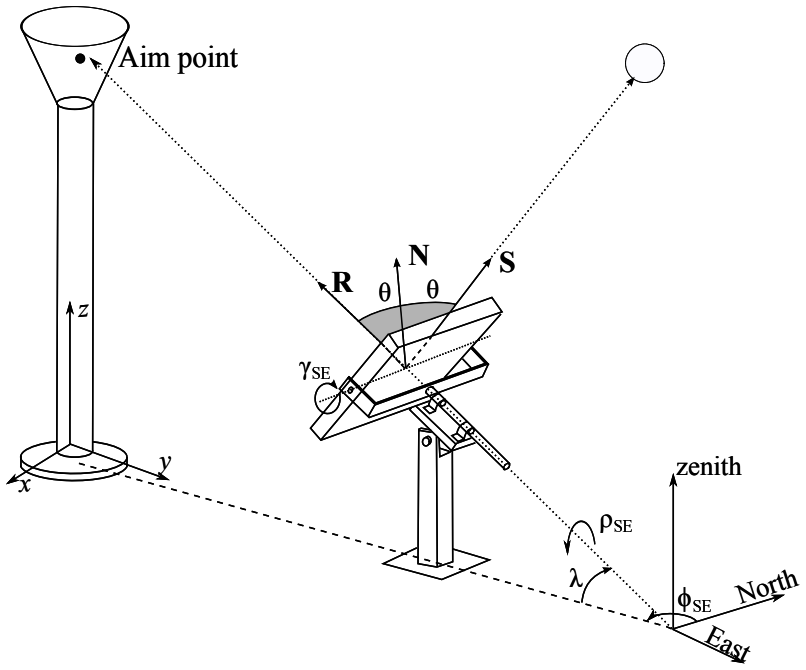


Figure 4.4: Spin-Elevation tracking adapted from (Chong & Tan, 2011)

The various tracking methods in SUNRAY allow for SUNRAY to simulate the passing of time. SUNRAY is capable of running a number of simulations at incremental times of the day (at a resolution of per second increments, if required) and the tracking algorithms ensure light is directed toward the target. SUNRAY is also able to read a CSV file of actual Direct Normal Irradiance (DNI) data to simulate the changing DNI values throughout the day.

To keep the tracking user-friendly, the user only needs to specify the desired aim point and the centroid of the collector. As SUNRAY knows what the position of the sun is, it automatically calculates the required rotation angles and creates a transformation matrix to correctly position and rotate the heliostat. The order of these transformations are critical to align the heliostat correctly. The complete transformation matrices for all tracking methods are provided in Appendix B.2.

4.3. Tracking Errors

All real heliostat systems suffer from tracking errors introduced by geometrical and mechanism imperfections (Kribus, Vishnevetsky, Yogev & Rubinov, 2004). It is possible that these tracking errors can lead to overall energy collection losses of $10-20\%$ (Jones & Stone, 1999*b*). In a study conducted on Solar Two in the USA, Jones & Stone (1999*a*) identified three dominant tracking error sources namely: incorrect tilting of the heliostat, incorrect alignment of the heliostat, and encoder reference errors. SUNRAY provides the framework to model these error sources, for example, alignment errors can be introduced in the build phase however, only the collective resultant effect of these errors are modelled in the Sun Tracker class. This is achieved by introducing random errors into the tracking algorithms in order to produce image misalignment.

Most errors can be modelled using a Normal Gaussian distribution, which has been set as the default distribution. However, if the errors follow a known or empirical distribution curve, Monte Carlo methods can be used to draw samples from this distribution. To implement tracking errors the Sun Tracker class generates random numbers from the given distribution, which are added to the γ and ρ angles.

4.4. Conclusions

The sun is an important factor in CSP systems. Features such as sunshape can strongly impact how flux is distributed on a receiver. With the various sunshapes models in SUNRAY it is possible to simulate a CSP system under a range of atmospheric conditions. Similarly, tracking mechanism can influence the performance of a CSP system. Therefore three important tracking methods have been implemented in CSP.

The tracking methods discussed in this chapter can be applied to any of the geometric objects in SUNRAY. These objects are discussed in the next chapter.

5. GEOMETRIC OBJECTS

The feature which will most likely vary among simulations is the number, position, and shape of collectors. A number of geometric objects have been included in SUNRAY, which form part of a broad collection of various objects that can be found in a CSP system. This chapter first describes basic primitive shapes using a sphere as an illustrative example. It then introduces triangles and demonstrates how they can be meshed to form almost any other shape. Finally, instancing is discussed and how it is used to simulate very large CSP systems.

5.1. Spheres

Quadratics is the general classification for the group of shapes which includes parabolic dishes, parabolic troughs, cones, and spheres. Spheres and partial spheres are common in CSP systems as they can form the basic shape of an imaging collector. They also have the simplest intersection algorithm of the quadratics (Suffern, 2007) and are thus a good illustration of the general concepts used to ray trace quadratic shapes. This section details how a sphere is constructed, bound and intersected in SUNRAY.

5.1.1. Ray-Sphere Intersection

Spheres are described by their well-known implicit equation.

$$x^2 + y^2 + z^2 - r = 0 \tag{5.1}$$

The equation for a ray intersecting a full sphere is found by substituting the parametric equation of a ray, Equation (3.1), $r(t) = \mathbf{o} + t\mathbf{d}$, into the implicit equation of a sphere, Equation (5.1).

$$(o_x + td_x)^2 + (o_y + td_y)^2 + (o_z + td_z)^2 = r^2 \tag{5.2}$$

Where o and d are the components of the ray's origin and direction, respectively. The only unknown in Equation (5.2) is t. Expanding Equation (5.2) and gathering coefficients gives the general quadratic form

$$at^2 + bt + c = 0$$

Where

$$\begin{aligned} a &= d_x^2 + d_y^2 + d_z^2 \\ b &= 2(d_x o_x + d_y o_y + d_x o_z) \\ c &= o_x^2 + o_y^2 + o_z^2 \end{aligned} \tag{5.3}$$

The solution to a quadratic equation is given by the standard quadratic formula and can have zero, one or two real roots, depending on the value of the discriminant, $d = b^2 - 4ac$. This is illustrated in Figure 5.1. Ray 1 does not hit the sphere ($d < 0$), Ray 2 only hits the sphere at one point ($d = 0$) and Ray 3 hits the sphere at two points ($d > 0$).

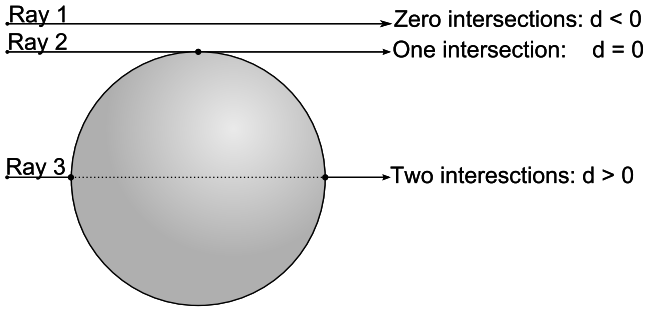


Figure 5.1: Ray sphere intersections

Once a value for t has been found, it is substituted back into Equation (3.1) to obtain the x-,y-, and z-coordinate of the hit point.

Partial Sphere

There is no closed-form equation describing only partial objects. For partial objects, rays are intersected with the full object and hit points falling outside the valid region are treated as missed rays. For the partial sphere in Figure 5.2, Ray 1 would miss the sphere at $\mathbf{p}_1$ as its hit point's z-value is greater than z_{max}.

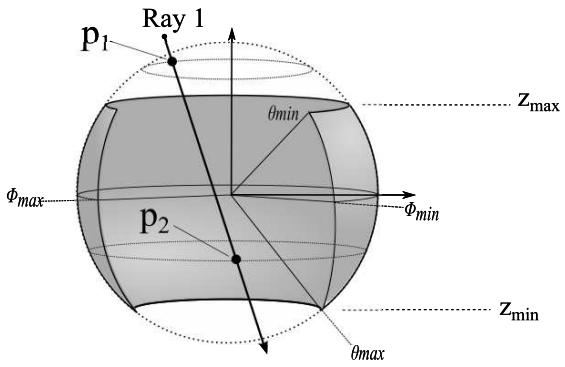


Figure 5.2: Ray intersections with a partial sphere

Similarly, the ray would also miss the partial sphere at $\mathbf{p}_2$, as $\phi_{hit} > \phi_{max}$, where ϕ_{hit} is calculated from the x and y hit points, (x_{hit}, y_{hit}), as follows

$$\tan \phi_{hit} = \frac{x_{hit}}{y_{hit}} \tag{5.4}$$

Surface Normal

For reflection calculations (Appendix D) the surface normal at the hit point $\mathbf{p} = (x, y, z)$ is required. The normal vector for a parametric surface is calculated using Equation (3.8). For a unit sphere centred on the origin, the partial derivatives are given in (Pharr & Humphreys, 2010) as

$$\begin{aligned} \frac{\partial p_x}{\partial u} &= \cos\theta \sin\phi = -y \\ \frac{\partial p_y}{\partial u} &= -\cos\theta \cos\phi = -x \\ \frac{\partial p_z}{\partial u} &= 0 \end{aligned} \tag{5.5}$$

Similar calculations can be found for $\partial\mathbf{p}/v$.

5.1.2. Construction

There are several ways in which a user can describe each object. For each object the number of parameters have been specifically limited to make it as simple as possible to construct an object and yet allow the user complete control over the description of the object. This is in line with the best practice C++ programming guidelines stipulated in (Meyers, 1995).

In SUNRAY there are four parameters which define a sphere: z_{max}, z_{min}, ϕ_{max}, and the radius, r. These are illustrated in Figure 5.3. There are three ways in which a user can construct a sphere. The simplest method is to use the default constructor. The default constructor takes no inputs and is a unit sphere centred on the origin. The second way is to define the radius and centre of the sphere. Alternatively, the sphere can be truncated to form a partial sphere. This can be done by specifying the z_{min}, z_{min}, and ϕ_{max}.

To assist in debugging, error messages (a violation which will prevent SUNRAY from executing) and warning messages (contradictions in logic, which still allow for a simulation to run) have been included in most functions in SUNRAY. For the sphere, an error is issued if $z_{min} > z_{max}$ or if $z_{max} > r$.

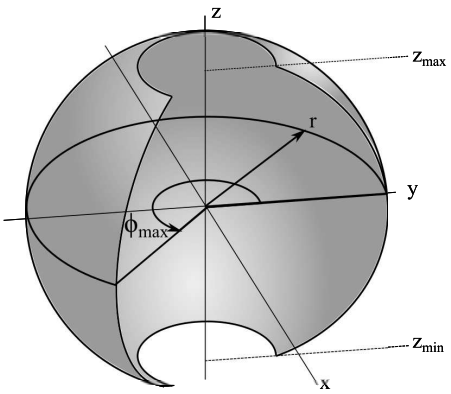


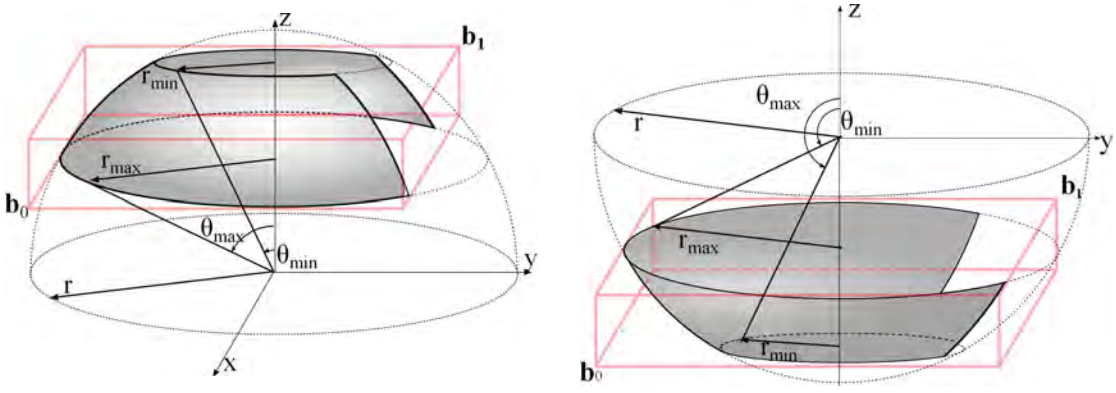
Figure 5.3: Sphere construction parameters

5.1.3. Bounding

All objects in SUNRAY are surrounded by a bounding box, which is an axis-aligned, box-shaped volume. Bounding volumes reduce the overall computational time of a ray tracer. This is further discussed in Section 7.1.

The bounding box is described by the opposite two corners of the box, $\mathbf{b}_0 = (x_0, y_0, z_0)$ and $\mathbf{b}_1 = (x_1, y_1, z_1)$ where $x_0 < x_1$, $y_0 < y_1$ and $z_0 < z_1$. For computational efficiency all bounding boxes in SUNRAY are computed to fit as tightly as possible to objects. For a complete sphere, the tightest fitting bounding box is simply a cube with the length of the sides equal to the diameter of the sphere. However, computing the values for $\mathbf{b}_0$ and $\mathbf{b}_1$ for a partial object is not as trivial.

Figure 5.4 illustrates how two partial spheres are bound by a bounding box. Each partial sphere, one located above the x-y plane and one below, has a radius, r, and is centred on the origin.



(a) Upper Hemisphere (b) Lower Hemisphere

Figure 5.4: Two partial spheres bound by a bounding box

In Figure 5.4 the spheres are truncated such that they do not cross the x-y plane. The longest length in the x-y plane, r_{max}, is thus not equal to r but is calculated from

$$r_{max} = \begin{cases} max \begin{cases} r\sin\theta_{max} \\ r\sin\theta_{min} \end{cases} & \text{if } z_{max} \times z_{min} \geq 0 \\ r & \text{if } z_{max} \times z_{min} < 0 \end{cases} \tag{5.6}$$

Similarly, the shortest length in the x-y plane, r_{min}, is calculated from

$$r_{min} = min \begin{cases} r\sin\theta_{min} \\ r\sin\theta_{max} \end{cases} \tag{5.7}$$

The values for θ_{min} and θ_{max} are calculated from

$$\theta_{min} = \arccos(z_{min}/r) \tag{5.8}$$

$$\theta_{max} = \arccos(z_{max}/r) \tag{5.9}$$

The value for $\mathbf{b}_0 = (x_0, y_0)$ is calculated from

$$y_0 = \begin{cases} 0 & \text{if } \phi_{max} \leq \dfrac{\pi}{2} \\ -r_{max}\cos\phi_{max} & \text{if } \frac{\pi}{2} < \phi_{max} \leq \pi \\ -r_{max} & \text{if } \phi_{max} \geq \pi \end{cases} \tag{5.10}$$

$$x_0 = \begin{cases} -r_{max}\sin\phi_{max} & \phi_{max} < \frac{\pi}{2} \\ -r_{max} & \phi_{max} \geq \dfrac{\pi}{2} \end{cases} \tag{5.11}$$

The value for $\mathbf{b}_1 = (x_1, y_1)$ is calculated from

$$y_1 = r_{max} \tag{5.12}$$

$$x_1 = \begin{cases} 0 & \text{if } \phi_{max} \leq \dfrac{\pi}{2} \\ -r_{max}\sin\phi_{max} & \text{if } \frac{\pi}{2} < \phi_{max} \leq \pi \\ -r_{max} & \text{if } \phi_{max} \geq \pi \end{cases} \tag{5.13}$$

In addition to quadratics, SUNRAY has a number of other shapes such as a plane, a disc (which is a truncated plane described by its radius), and triangles.

5.2. Triangles

Triangles are the most common shape used in graphical ray tracers (Heckbert, 1994). They are also an important shape in SUNRAY as they are used as the building blocks for more complex shapes and are also used to represent real mirror surfaces.

The intersection of a ray and a triangle is conceptually similar, but mathematically different to that of a ray and a quadratic and the procedure is discussed in Appendix C. In this section triangular meshes are discussed. Triangular meshes are important as they are used to describe the surface profile of real mirror surfaces which have been scanned into SUNRAY. This allows for the simulation of experiments (Appendix F).

Triangular meshes, also known as Triangular Irregular Networks (TIN), are tessellated triangles which are meshed to approximate surfaces with complex mathematical descriptions (Amanatides & Choi, 1997). Shapes are built using a network of triangles, which share vertices and edges to form a continuous surface, such as the sphere in Figure 5.5. For a closed shape Haines (2001) showed that, on average, six triangles will share one common vertex. To save on memory usage, instead of each triangle storing its own vertex, the coordinates of the common vertex can stored in a separate data structure and each triangle is given access to this auxiliary data structure.

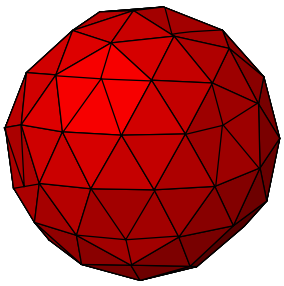


Figure 5.5: A sphere approximated by tessellated triangles

Because triangular meshes play such an essential role in ray tracing, they require an efficient data structure. There are several data formats available and a good review of some of these formats is given by Bourke (2003). From the available formats, the Polygon File Format (PLY) was chosen for SUNRAY.

The PLY file format, developed by Turk & Levoy (1994) and Turk (1998), was chosen for its simplicity and because it was developed specifically to handle 3D scanned data files. Turk's code has been adapted in SUNRAY to automatically generate a PLY file from the output of a coordinate measuring machine (CMM) (Appendix F).

If required, PLY files can be converted from other data structures. Programs, such as 3D Object Converter [online], which is freely available, can convert PLY files to and from multiple data structures, for example (.obj) file format, which is the output file of Google's 3D modelling program, Sketchup (Trimble, 2013).

5.3. Compound Objects

Paraboloids and triangles fall within a subset of shapes often referred to as primitives or generic objects. Primitives are simple shapes with closed-form solutions (John, 2005). A compound object is an object made up of a number of these primitives.

There are a number of advantages in using a compound object. Firstly, a collection of objects can be treated as a single object. For example, the compound object Multi-Facet is a rectangular heliostat made up of multiple rectangular facets, which can be placed anywhere in a scene and is able to track the sun as if it were a single object. Secondly, new compound objects can be created without having to write new intersection routines because the intersection tests are simplified to test only the primitives which make up the object.

Compound objects can be constructed from other compound objects, which in themselves can be constructed from compound objects. In SUNRAY there is no limit to the depth of compound objects, but increasing the number of objects has a negative effect on memory usage. To reduce memory usage, SUNRAY applies a technique known as instancing.

5.4. Instancing

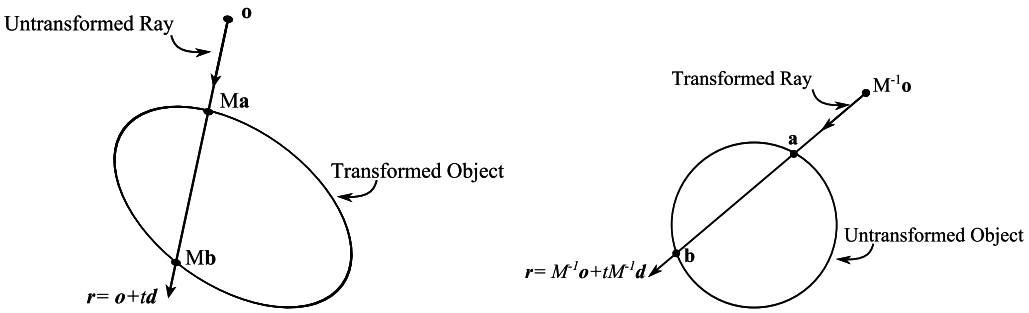
Instancing is used to save memory when there are a number of repeated objects (such as multiple heliostats) in SUNRAY. It does this by storing just a single version of an object and creating a number of instance objects, which each store a pointer to that object. To allow for different materials, each instance object also stores a pointer to its material.

Each instance object is placed on the origin of the world coordinate system. Then, through a series of transformations (described by the transformation matrix M), the object is rotated and translated to its location within the scene. Because only one coordinate system is used in SUNRAY, transformations need to be conducted in a strict order. This is discussed in Appedix C.2.

Instancing is particularly useful when designing a large field layout. To reduce the time it takes for a user to set up a heliostat field, SUNRAY has been developed to read a CSV file describing the coordinate locations of multiple heliostats. It

then uses instancing to translate the heliostats to their respective positions. After that, the tracking algorithms from Section 4.2 automatically rotate each heliostat to ensure that its image strikes the target.

The procedure for intersecting a ray with a transformed object is slightly different to what has been discussed. Because ray-object intersection can be performed in any space and because untransformed objects often have simpler intersection tests than transformed objects, the transformation is not performed on the object, but instead the ray is transformed. This is done by applying the inverse-transformation matrix to the ray, as illustrated in Figure 5.6. In the figure **a** and **b** represent the ray hit points on the object and M represents the transformation matrix.



(a) Object transformation (b) Ray transformation

Figure 5.6: A ray intersecting a transformed object

SUNRAY uses the algorithm described in (Suffern, 2007) for intersecting transformed objects. The steps for intersecting a ray with an object are as follows:

1. Apply the inverse-transformation matrix (M^{-1}) to the ray.
2. Intersect the transformed ray with the untransformed object and calculate the normal and hit point of the untransformed object.
3. Apply the transformation matrix, M, to the normal and hit point (the scalar t is not affected by the transformation).

5.5. Conclusions

Most common shapes have reasonably closed-form solutions and most other complex objects can be described using compound objects or TINs. TINs also allow for highly resolved surface shapes and real mirror surfaces to be simulated. This has the valuable advantage of being able to simulate actual experiments.

Once a ray has intersected an object, SUNRAY calls the object's Material and BRDF classes in order to determine how the ray is reflected off the object. This is discussed in the next chapter.

6. MATERIALS AND BRDF

Reflection from a surface can be split into four broad categories: diffuse, perfect specular, imperfect specular (glossy), and retro-reflection. Most real surfaces exhibit a reflection that is a mixture of these four types (Siegel & Howell, 1992). Diffuse reflectors, also known as Lambertian surfaces, scatter light equally in all directions. In perfect specular reflection the incident light is reflected in a single, outgoing direction. Imperfect specular surfaces scatter light preferentially in a set of directions and retro-reflectors, such as velvet, scatter light back along the incident direction. Reflection models can come from numerous sources such as measured data, phenomenological models, simulations, wave optics or geometrical optics (Modest, 2003). The reflection models in SUNRAY are based on a combination of geometric optics and empirical models. This chapter discusses the various reflection models that are incorporated into SUNRAY. The supporting radiometric theory behind the reflection models is provided in Appendix D.

6.1. Materials and BRDF Implementation

Surface reflection is implemented through two classes, namely the Bidirectional Reflection Distribution Function (BRDF) class and the Material class. The Material class calculates the direction of the reflected ray and the BRDF class calculates the magnitude of reflected radiance.

From the linearity properties of BRDF (see point 3, Appendix D.3) a material can have multiple BRDFs, for example, polished steel can exhibit a mixture between diffuse and specular reflection. To define a new material, a user can combine several existing (or new) BRDFs.

6.2. Material

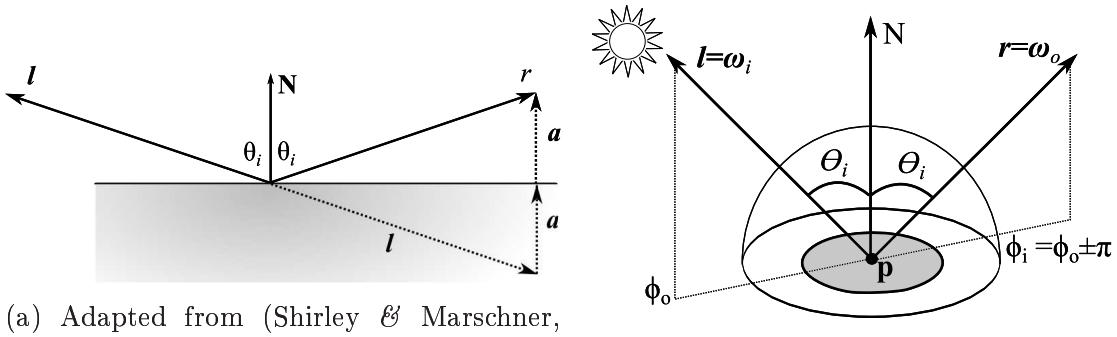
In SUNRAY there are two reflection models: perfect specular and imperfect specular. In perfect specular reflection the direction of the reflected ray is calculated using the ideal Law of Reflection. The Law of Reflection states that the angle of incident light relative to the surface normal is equal to the angle of reflection, $\theta_i = \theta_o$. In imperfect specular reflection, the direction of the reflected ray is calculated using Monte Carlo methods.

6.2.1. Perfect Specular

A perfect specular reflector, is a surface which is perfectly smooth and is completely flat (has an infinite radius of curvature [Modest, 20013)]). The direction of a ray, reflected off a perfectly specular surface, is given by The Law of Reflection. The reflected ray's directions can be derived from Figure 6.1a.

The reflected direction, $\mathbf{r}$, is given by $\mathbf{r} = -\mathbf{l} + 2\mathbf{a}$, where $\mathbf{l}$ is the incident direction. The vector $\mathbf{a}$ is parallel to the normal, $\mathbf{N}$, but has a length of $\mathbf{l}$, scaled by the cosine of the angle between $\mathbf{l}$ and $\mathbf{N}$. If $\mathbf{N}$ is a unit vector then

$$\mathbf{r} = -\mathbf{l} + 2(\mathbf{N} \cdot \mathbf{l})\mathbf{N} \tag{6.1}$$



(a) Adapted from (Shirley & Marschner, 2009)

(b)

Figure 6.1: Specular reflection off a smooth surface

6.2.2. Imperfect Specular Reflection

On an atomic level, a perfectly smooth surface does not exist. Therefore, specular reflection is always combined with a certain amount of scattering (Siegel & Howell, 1992). Scattering results in a widening of the beam of reflected radiance.

Many models have been developed to represent imperfect specular reflection. Some of these models have no physical basis and have only been developed to render photorealistic images. An example of such a model is the Phong reflection model (Phong, 1975), which formed the cornerstone of early graphical ray tracers (Shirley & Marschner, 2009). Other models are more physically based as they have been developed using empirical data or on concepts such as geometric optics and microfacets.

Microfacets are tiny, flat reflectors of random size and orientation. The reflection off a microfacet can be modelled as Lambertian (equal probability of scattering in any direction), such as the Oren and Nayer model described by Pharr & Humphreys (2010) or perfect reflector, such as by the Torrence and Sparrow model (Torrence & Sparrow, 1967).

Imperfect specular reflection has been implemented in SUNRAY by adapting the parameters in the SolarPACES guidelines: *Parameters and Method to Evaluate the Solar Reflectance Properties of Reflector Materials CSP Technology* (Meyen, García & Kennedy, 2011; Meyen, Montecchi, Kennedy, Zhu, Gray, Crawford & et al., 2013). These guidelines are recommendations for the standardisation of reflectance characterisation of CSP collectors and are therefore well-suited for SUNRAY. These guidelines have been based, in part, on the work of Butler & Pettit (1977); Pettit (1977); and Gee, Brost, Zhu & Jorgensen (2010).

Through experiments on solar concentrators, Pettit (1977) showed that the scattering distribution function of light off solar reflective materials can be characterised by the sum of two normal distributions: one with a high amplitude and narrow standard deviation, where most of the reflected light is contained; the other with a low amplitude and broad standard deviation. These distributions represent the specular and diffuse spread of light, respectively. In a more recent study, Gee *et al.* (2010) used a weighting term K, to weigh the two normal distributions. The specularity error, σ_{spec}, illustrated in Figure 6.2 is given by the following

$$\sigma_{spec}^2 = K\sigma_{spec1}^2 + (1-K)\sigma_{spec2}^2 \tag{6.2}$$

Where σ_{spec1} and σ_{spec2} are the higher intensity and lower intensity Gaussian distributions respectively. The default value for K in SUNRAY is set as 1, which reduces Equation (6.2) to just one term. Alternate values for K can be found in (Gee *et al.*, 2010).

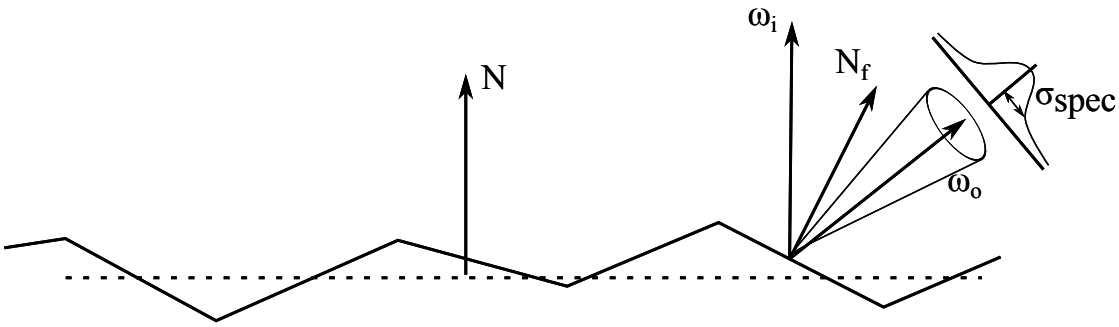


Figure 6.2: The reflected vector off a surface of microfacets

In addition to specularity errors, the Imperfect Specular class also models slope errors, σ_{slope}. Slope errors are described by the distribution of the surface normals, $\mathbf{N}_f$. From the laws of reflection, a slope error of θ will result in the reflection vector $\boldsymbol{\omega}_o$ to be reflected by 2θ, as illustrated in Figure 6.3.

From Gee *et al.* (2010), a collector surface is modelled as a sum of squares of the two surface errors: specularity (micro-errors) and slope (macro-errors) slope.

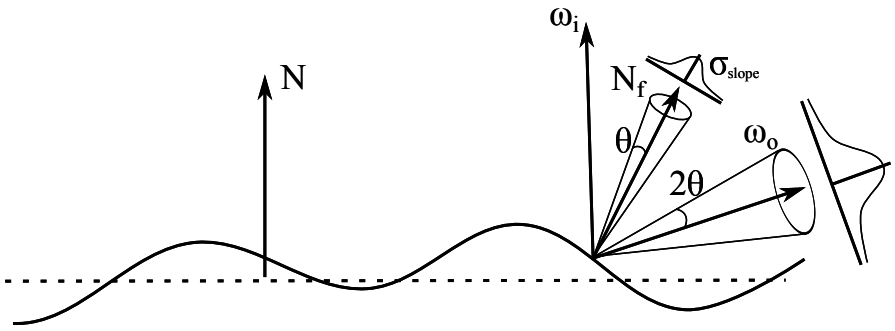


Figure 6.3: Slope error adapted from (NREL, 2012*b*)

The total beam spread is then calculated as the root mean square (rms) value of the two optical errors.

$$\sigma_{combined} = \sqrt{\sigma_{micro}^2 + 4\sigma_{macro}^2} \tag{6.3}$$

6.2.3. Imperfect Specular Implementation

The Imperfect Specular class generates the sample points (x_{sp}, y_{sp}, z_{sp}) on the unit hemisphere above hit point $\mathbf{p}$, using similar Monte Carlo methods as described in the Gaussian sunshape in Section 4.1.2.

Imperfect Specular class sets up a new coordinate system centred on $\mathbf{p}$. This new coordinate system is constructed by

$$\begin{aligned} \mathbf{w} &= \boldsymbol{\omega}_o \\ \mathbf{u} &= \mathbf{y} \times \mathbf{w} \\ \mathbf{v} &= \mathbf{u} \times \mathbf{w} \end{aligned} \tag{6.4}$$

Where $\mathbf{y}$ is the y-axis in the world coordinate system and $\boldsymbol{\omega}_o$ is the perfect specular reflected ray direction calculated from Equation (6.1). The reflected ray's direction can then be written, as a linear combination of this orthonormal coordinate system (Suffern, 2007)

$$\mathbf{d} = x_{sp}\mathbf{u} + y_{sp}\mathbf{v} + z_{sp}\mathbf{w} \tag{6.5}$$

An issue which arose when developing the Imperfect Specular class is illustrated in Figure 6.4. For simulations with very rough surfaces (a large amount of scattering) or with rays arriving at high-incident angles, the reflected ray direction can fall below the surface of the mirror. In these situations the reflected direction is simply reflected through $\boldsymbol{\omega}_0$ by reversing the signs of x_{sp} and y_{sp}. Thus Equation (6.5) becomes

$$\mathbf{d} = -x_{sp}\mathbf{u} - y_{sp}\mathbf{v} + z_{sp}\mathbf{w} \tag{6.6}$$

Unfortunately, this introduces a bias in the distribution function, resulting in a higher probability for rays to fall within the upper periphery. This issue has not been addressed in this study as it was assumed that the majority of simulations would be conducted with near specular reflection. Further work may have to address this problem.

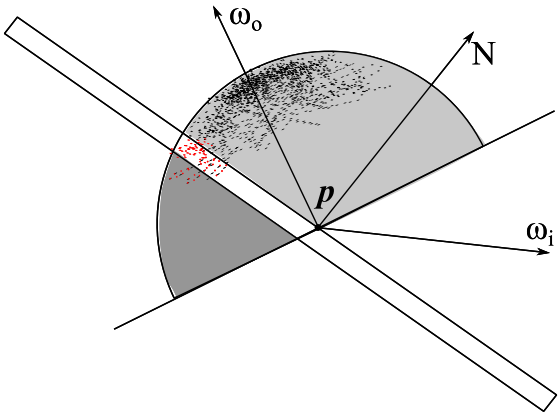


Figure 6.4: Rays generated below the surface of the reflector

6.3. BRDF Models

In SUNRAY the fraction of reflected[1] light is given by the reflection coefficient $k_r \epsilon [0, 1]$. There are two BRDF models in SUNRAY that can be used to determine k_r: Constant Specular BRDF and Fresnel Reflection. Of the two, Fresnel Reflection is the more accurate as it is based on actual physical laws.

In addition to the fraction of reflected light, SUNRAY also models the spectral distribution of the reflected ray. This is a unique feature of SUNRAY which has not been implemented in either Tonatiuh or SolTrace. Modelling the spectral distribution of reflection is important for simulations of systems with, for example, selective absorbers (Kennedy, 2002). To model the spectral distribution, SUNRAY uses a 3D base function $\mathbf{c}_r$. As an example, low-quality solar glass has a high iron content, therefore there will be higher reflection in the green wavelength (Alanod Solar, 2012) and the base function could be $(R, G, B) = (0.8, 1.0, 0.8)$. This is discussed further in Appendix D.1.

[1]The percentage of light not reflected can either be absorbed or transferred. However, for energy calculations SUNRAY assumes that all energy not reflected is absorbed.

6.3.1. Constant Specular BRDF

The simplest approximation to model the reflection of light off a surface is to approximate a constant BRDF over the entire surface. Thus, the amount of flux in the single direction, $\boldsymbol{\omega}_o$, is independent of the incoming direction, θ_i, and is scaled by a constant reflection coefficient, $f_{r,s} = \rho_s = k_r \mathbf{c}_r$.

In order to ensure that the BRDF is only valid for the reflected direction and is zero for all values of $\theta_r \neq \theta_1$ and $\phi_i \neq \phi_r$ (Figure 6.1b), a singularity term needs to be included in the BRDF. Nicodemus, Richmond & Hsia (1977) gives the constant specular reflection BRDF as

$$f_{r,s}(\mathbf{p}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = 2k_r \ {}_r\delta\sin^2\theta_r - \sin^2\theta_i\delta\phi_o \pm \pi - \phi_i / \cos\theta_i \tag{6.7}$$

The delta term is a singularity term which ensures that light is only reflected in a single (specular) direction. The cosine term has been included in Equation (6.7) to cancel out with the cosine term in Equation (6.10).

For quick, simple solutions, Equation (6.7) is a reasonable approximation for a specular BRDF and that is why it is used in many ray tracers. However, physically based reflection is directionally dependent and cannot be captured by a constant value of reflectivity. The Fresnel Equations, derived by Augustin-Jean Fresnel (1788-1827), specify how light is reflected and transmitted at the boundary of optically smooth surfaces (Querry, 1969).

6.3.2. Fresnel Reflection

There are two sets of equations for Fresnel reflection, which are derived from the Maxwell Equations. One is for conductors, i.e., materials such as metals which conduct electricity, and one for dielectric materials, or non-conductors, such as clear quartz (Querry, 1969). Since transparency is not modelled in SUNRAY only conductors have been considered.

Conductors do not transmit light but some of the incident light is absorbed by the material and turned into heat. The amount of light reflected is a function of the refractive index, η, the absorption coefficient, k, and the wavelength of light, λ. An example of a reflective conductor is silver. The refractive index and absorption coefficient for silver is plotted in Figure 6.5, along with the spectral wavelengths of light.

The Fresnel Equations can take on a further two forms depending on the polarisation of incoming light (Querry, 1969).

$$r_{\parallel}^2 = \frac{(\eta^2 + k^2)\cos\theta_i^2 - 2\eta\cos\theta_i + 1}{(\eta^2 + k^2)\cos\theta_i^2 + 2\eta\cos\theta_i + 1} \tag{6.8a}$$

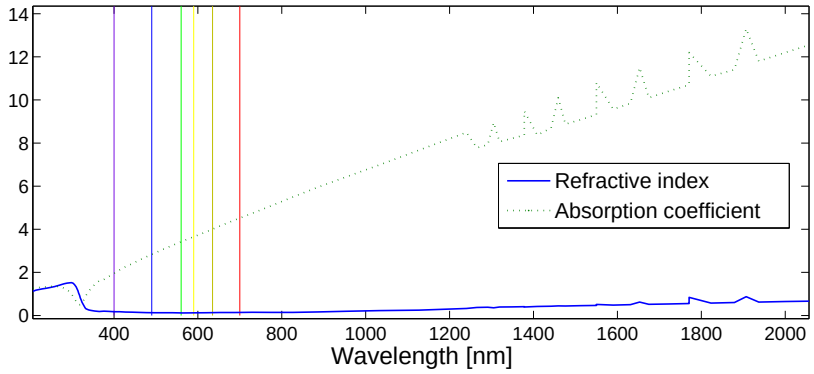


Figure 6.5: Absorption oefficient and refractive index of silver from Palik (1985)

$$r_{\perp}^2 = \frac{(\eta^2 + k^2) - 2\eta\cos\theta_i + \cos\theta_i^2}{(\eta^2 + k^2) + 2\eta\cos\theta_i + \cos\theta_i^2} \tag{6.8b}$$

Where $r_{\parallel}$ is the Fresnel reflection for parallel polarised light and $r_{\perp}$ is the reflectance for perpendicular polarised light. However, SUNRAY does not take polarisation into account as sunlight is unpolarised (Meyen *et al.*, 2013). Instead, SUNRAY assumes that light is randomly orientated with respect to wavelength. The Fresnel Equations then simplify to the average of the squares of parallel and perpendicular terms (Pharr & Humphreys, 2010). The reflection coefficient, F_r, for Fresnel reflection is given by

$$F_r = \frac{1}{2}(r_{\parallel}^2 + r_{\perp}^2) \tag{6.9}$$

A visualisation of F_r is plotted in Figure 6.6. The values for η and k are obtained from Figure 6.5 for the red, blue, and green wavelengths. It can be seen from the figure how the value of F_r varies with incidence angle. This demonstrates how approximating reflectivity with a constant value (as is done in many ray tracers) can introduce errors in the calculation.

6.3.3. BRDF Implementation

A ray propagated into a scene will carry a certain amount of energy (flux). As the ray interacts with an object in the scene it loses energy (a ray can also be attenuated through the atmosphere, see Appendix D.6). SUNRAY continually keeps track of this energy decrease via Kajiya (1986) rendering equation. Kajiya gives the reflected radiance at point $\mathbf{p}$ in the direction $\boldsymbol{\omega_o}$, due to the incident illumination from the whole hemisphere above $\mathbf{p}$ as

$$L_o(\mathbf{p}, \boldsymbol{\omega}_o) = \int_{2\pi^+} L_i(\mathbf{p}, \boldsymbol{\omega}_i) f_r(\mathbf{p}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \cos\theta_i d\omega_i \tag{6.10}$$

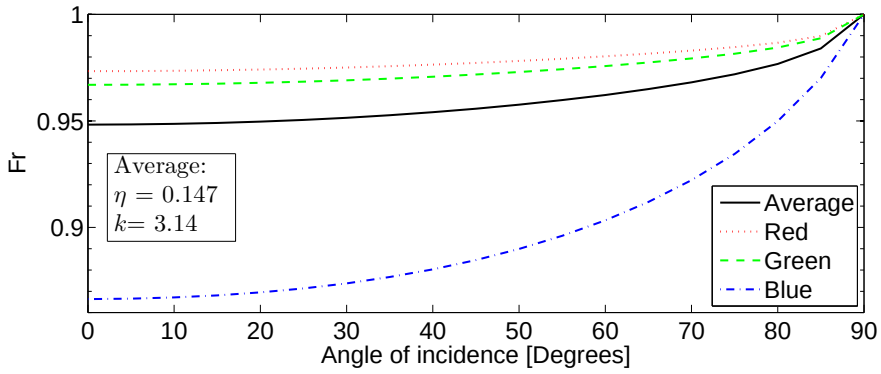


Figure 6.6: Fresnel reflection for various wavelengths and incident angles

This fundamental equation in ray tracing is used to describe how light at a point is transformed into an outgoing distribution. It is also often referred to as the scattering equation (Pharr & Humphreys, 2010) and reflection equation (if the domain is just the upper hemisphere Ω_i [Suffern, 2007]). A key task of SUNRAY is to compute the solution to the rendering equation at every hit point.

The reflected radiance in direction $\boldsymbol{\omega}_o$ for a constant BRDF is found by substituting Equation (6.7) into Equation (6.10). Thus, specular reflection is implemented by

$$L_o(\mathbf{p}, \boldsymbol{\omega}_o) = \frac{k_r \mathbf{c}_r}{N \cdot \boldsymbol{\omega}_i} L_i(\mathbf{p}, \boldsymbol{\omega}_i) \tag{6.11}$$

Where the direction of the reflected ray, $\boldsymbol{\omega}_o$, is calculated from Equation (6.1). For Fresnel reflection, F_r from Equation (6.9) replaces k_r in Equation (6.11).

6.4. Conclusions

Correct description of various material properties is important in ray tracing and requires effective use of Monte Carlo methods. Therefore to assist the user the Material and BRDF classes have been written to ensure that a very wide range of solar materials can be simulated in SUNRAY. To achieve this, the reflection models were based on the physically accurate Fresnel Equations as well as the SoarPaces guidelines. Moreover, describing the spectral distribution of light as a 3D (R, G, B)-vector allows for greater control over the simulation of a CSP plant.

In order to obtain an accurate solution to a simulation a description of the objects material and BRDF must be provided. Acceleration techniques, which are discussed in the next chapter, do not necessarily affect the accuracy of a simulation, but they are essential for ensuring reasonable simulation times.

7. ACCELERATION TECHNIQUES

The ray tracing method of testing the intersection of each ray with each and every object and retaining the nearest hit point is adequate to determine a solution to any simulation. However, this exhaustive method of ray tracing has simulation times linearly proportional to the number of objects in the scene, causing the computational times for very large simulations to become prohibitively high. Most rays will only possibly intersect very few objects and miss the others by a large distance. If whole groups of objects can be rejected and not even tested, there will be a substantial increase in performance.

Due to the essential role they play in ray tracing, a number of acceleration techniques have been developed. An overview of the popular acceleration techniques used in ray tracing is provided in Appendix E. This chapter discusses the four main acceleration techniques used in SUNRAY. These are: bounding boxes, the Grid, which reduce the cost of ray intersections; the View Grid, which is a method for ray propagation, which reduces the required number of rays; and the method to Predict the Required Number of Rays (PRNR), which is a novel statistical method which ensures that the minimum number of rays are traced in a simulation.

7.1. Bounding Boxes

A bounding volume is a shape which surrounds an object in the scene and permits simpler ray intersection calculations (Goldsmith & Salmon, 1987). Only if the ray intersects the bounding volume is the actual object tested. Although this actually increases the computational cost of rays which pass through the volume, it simplifies calculations for the vast majority of rays which miss the object.

In SUNRAY the bounding volume is an axis-aligned box, called a bounding box (Appendix E.2.1). As discussed in Section 5.1.3, all objects in SUNRAY have a bounding box and a user defining a new shape needs to include a routine to calculate the shape's bounding box. All bounding boxes in SUNRAY have been computed to fit as tightly as possible onto their shapes. However, a loose-fitting bounding box will not affect the accuracy of a simulation, only the running time.

For a bounding box to be effective it needs an inexpensive intersection test. The algorithm for ray-object intersection is best described in the 2D case, as illustrated

in Figure 7.1. The ray parameter where the ray hits the line $x = x_0$ is given as

$$t_{xmin} = \frac{x_0 - o_x}{d_x} \tag{7.1}$$

Where o_x and d_x are the x-components of the ray's origin and direction, respectively. Similar calculations are made for t_{xmax}, t_{ymin}, and t_{ymax}.

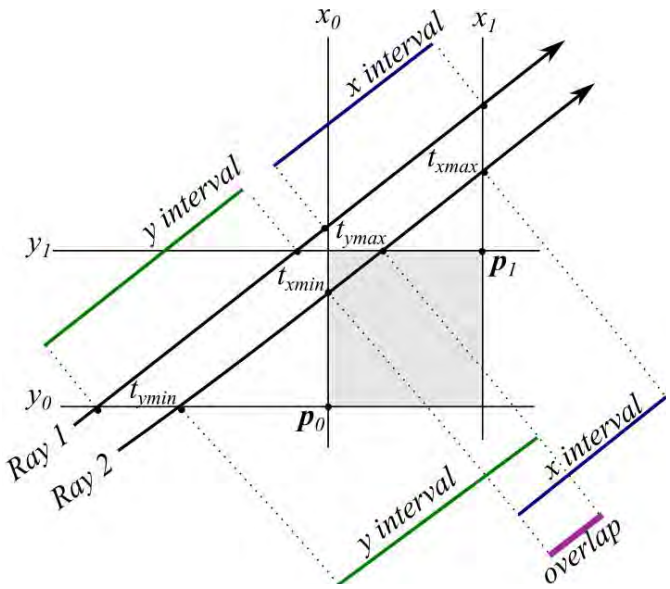


Figure 7.1: Ray-bounding box intersection adapted from (Suffern, 2007)

A ray only hits the box if the intervals $[t_{xmin}, t_{xmax}]$ and $[t_{ymin}, t_{ymax}]$ overlap. Therefore, a ray will only hit if one of the following conditions are met

$$\begin{aligned} t_{xmin} &< t_{ymax} \\ t_{ymin} &< t_{xmax} \end{aligned} \tag{7.2}$$

As seen in Figure 7.1, the upper ray misses the box, whereas the lower ray hits. The complete algorithm is given in Appendix E.2.1.

Bounding boxes also play an important role in ray propagation. In the initial build stage (Section 3.1.3), once all the objects have been added to a scene, a search algorithm finds the minimum and maximum coordinates of all the bounding boxes in the scene. These coordinates form the corners of a *global* bounding box which encompasses the entire scene. Unlike graphical ray tracers, where the user defines the area from where rays are propagated, this global bounding box describes the size of the scene completely and limits the area from where rays are propagated in SUNRAY.

7.2. Ray Propagation

There are numerous ways to propagate rays into a scene. An investigation into the different ray propagation techniques is given in Appendix E.3, whereas this section discusses a novel method of ray propagation which has been developed for SUNRAY. This proposed propagation method reduces the computational time by only tracing rays which have the highest probability of intersecting an object.

Various methods for ray propagation were considered and explored for SUNRAY, including generating rays on the objects themselves or on the target (backward ray tracing [Arvo,1986]). These techniques, however, do not allow for easy extendibility. Therefore, it was decided to propagate rays from the sun toward the scene. This approach utilises the intuitive, natural path of light and can be used for any scene regardless of the number or type of collectors or targets.

In SUNRAY all rays are generated from the same plane, which has been called the View Plane, which is located above the scene. This plane is perpendicular to a vector pointing from the origin of the world coordinate system to the sun. To reduce computational time, rays are only generated above each object in the scene. The method for ray propogation is illustrated in Figure 7.2. In this method the View Plane is divided into a number of cells, forming the View Grid. Rays are only generated from cells which contain a projection of the object's bounding box as this ensures that fewer rays miss the collectors.

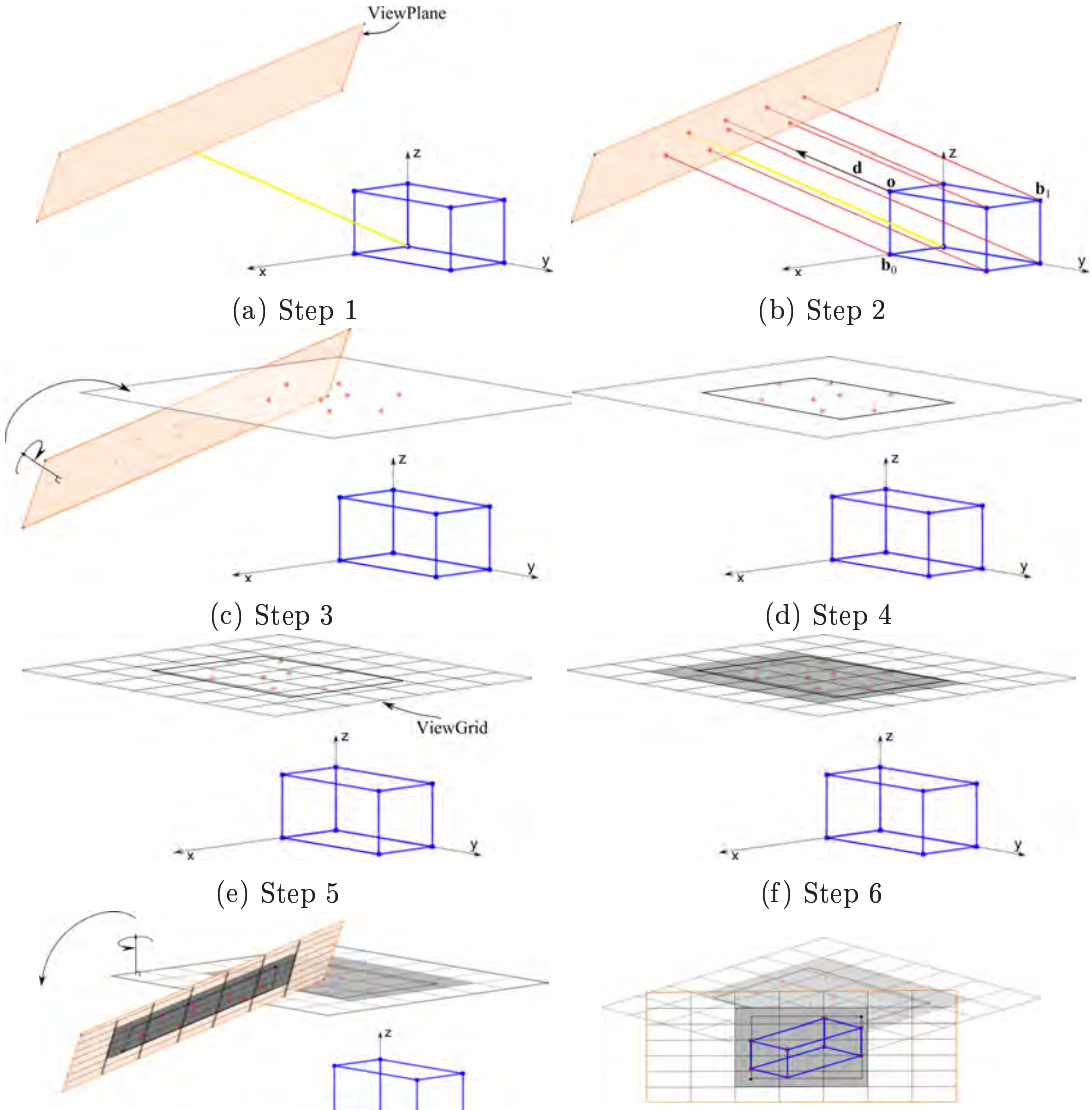
The following steps for ray propagaion are performed only once in the initial build stage. After the rays have been generated they can be traced using multiple processors (Section 3.3.2).

Step 1: Generate the View Plane Regardless of how the user defines the position of the sun (Appendix B.1), a vector is generated which points from the origin of the world coordinate system towards the sun's position, called the Sun Vector. This vector forms the normal to the plane from where rays are generated.

Step 2: Project the bounding box Principles of ray tracing are used to project each object's bounding box onto the View Plane. From each corner of the bounding box a ray is shot toward the View Plane. The intersection points of the rays and the plane are calculated using Equation (3.6).

$$t = \frac{(\mathbf{a} - \mathbf{o})\mathbf{n}}{(\mathbf{d} \cdot \mathbf{n})}$$

Where $\mathbf{o}$ is the corner of the bounding box, $\mathbf{d}$ is parallel to the Sun Vector, $\mathbf{n}$ is in the opposite direction to $\mathbf{d}$ and $\mathbf{a}$ is a point on the plane which determines how far the View Plane is located from the scene. The View Plane needs to be generated



(a) Step 1 (b) Step 2

(c) Step 3 (d) Step 4

(e) Step 5 (f) Step 6

(g) Step 8 (h) Bounding box as viewed from the View Grid

Figure 7.2: Ray propagation steps

sufficiently far away from the scene to ensure that no part of the View Grid is below the horizon. Therefore, point **a** is found from the scalar multiplication of **d** by the maximum length of the bounding box.

$$\mathbf{a} = \mathbf{d}|\mathbf{b}_o - \mathbf{b}_1| \tag{7.3}$$

Step 3: Transform the View Plane In order to simplify the alculations in Step 4, the z-ordinate is removed. This is done by transforming the View Plane and all the projected points (from Step 2) so that they sit on a plane that is par-

allel to the x-y plane. The transformation is done by multiplying all the projected points by the inverse transformation matrix M_{sun}^{-1}, see Appendix B.1.

Step 4: Compute the encompassing rectangle Once the projected corners of the bounding boxes are all on a horizontal plane, finding the minimum and maximum x- and y-coordinates of a rectangle which contains all the projected points, simplifies to a straightforward minimum/maximum search algorithm. When this rectangle is transformed back to the original sun position, it will completely enclose the global bounding box. Figure 7.2h is a view of the bounding box as seen from the View Grid once it has been transformed back. The figure shows that the bounding box is completely enclosed the rectangle.

To account for the solar subtend angle, the rays are not propagated perpendicular to the View Plane but within a cone defined by the sunshape (Section 4.1). The dimensions of the rectangle need to be extended to account for this. Each corner of the rectangle is extended by an *extra* amount defined by

$$extra = |z_1 - z_0 + offset| \tan \beta \tag{7.4}$$

Where z_1 and z_0 are the maximum and minimum z-ordinates of the global bounding box. The $offset$ is a small number (by default 0.001) which ensures that the View Plane is not positioned on the global bounding box, as illustrated in Figure 7.3.

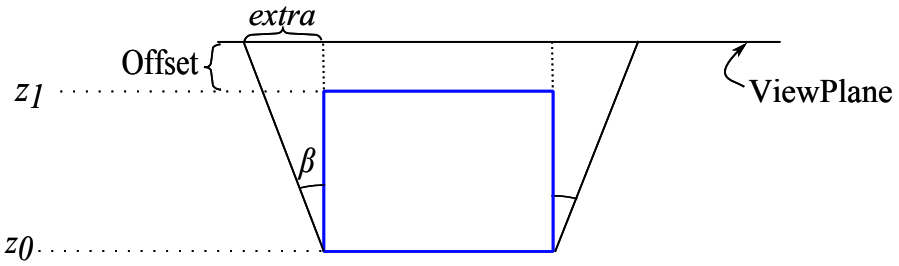


Figure 7.3: Increasing the projected area to account for the sunshape

Step 5: Setting up the View Grid The following expression, which has been adapted from (Suffern, 2007), is used to calculate n_x and n_y, i.e., the number of cells in the x and y directions, respectively. This expression ensures that the cells of the View Grid are roughly square.

$$\begin{aligned} s &= w_x w_y / n^{1/3} \\ n_x &= m_{vg} w_x / s + 1 \\ n_y &= m_{vg} w_y / s + 1 \end{aligned} \tag{7.5}$$

To prevent partial cells, SUNRAY rounds n_x and n_y off to the nearest whole number. The values for w_x and w_y are the x and y dimensions of the View Grid, illustrated in Figure 7.4. n is the number of objects in the scene and m_{vg} is a multiplying factor which varies the number of cells in the grid.

It is more favourable to have a large value for m_{vg} as this reduces the size of the cells of the View Grid. The multiplying factor is automatically selected by SUNRAY but can also be varied by the user. A multiplying factor of zero turns the View Grid off. The +1 to the right of Equation (7.5) ensures that even with a multiplying factor of zero, there will be at least one active cell.

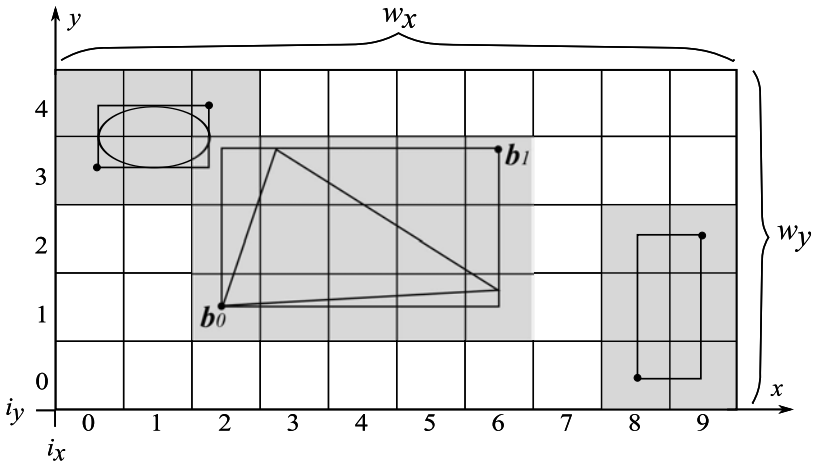


Figure 7.4: View Grid with encompassed objects

Step 6: Activate cells All cells which contain a projection of the object's bounding box, the shaded-grey cells in Figure 7.4, need to be activated. To do this, the cells which contain the upper $\mathbf{b}_1$ and lower $\mathbf{b}_0$ corners of the box are first found. Each cell is then given an index value, where $index = (i_x, i_y)$, and SUNRAY stores the indices in a standard array. The index of a cell is calculated from

$$index = i_x + n_x i_y \tag{7.6}$$

Figure 7.5 shows how SUNRAY computes which cells contain the corners of the bounding box.

In the figure the grid extends from $\mathbf{p}_0$ to $\mathbf{p}_1$. The length of the grid in the x direction is $p_{1x} - p_{0x}$. The following expression is used to find the index of the cell in which point $\mathbf{b}$ lies

$$\begin{aligned} f(b_x) &= (b_x - p_{0x})/(p_{1x} - p_{0x}) \\ i_x &= n_x f(b_x) \end{aligned} \tag{7.7}$$

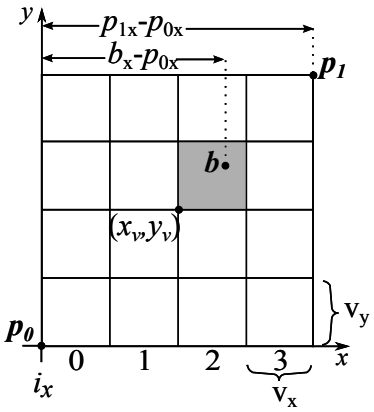


Figure 7.5: View Grid with a point **b**

An issue with Equation (7.7) occurs when **b** is located on the boarder of the grid, for example when $b_x = p_{0x}$. In such a case, it was found that computer rounding errors returned values of $f(\mathbf{b}) > 1$. This would have caused SUNRAY to crash because it would have been trying to access an index value which was out of range. The clamp function, described in (Suffern, 2007), was used to fix this problem.

Once the index values of the cells which contain the corners of the bounding box ($\mathbf{b}_0$, $\mathbf{b}_1$) have been found, SUNRAY sequentially activates all cells between the corners.

Step 7: Populate View Grid cells When generating the View Grid, the total number of active cells and the size of each cell are stored. These are used to calculate the total area from where the rays are propagated. To ensure no bias is introduced, rays are evenly distributed among active cells. If it is not possible to divide the rays equally among the cells, the remaining rays are not generated.

In addition to the total number and size of the active cells, the lower left corner, (x_v, y_v), of each active cell is also stored. The origin, **o**, of each ray can then be found from

$$\begin{aligned} o_x &= (w_x \xi_x + x_v) \\ o_y &= (w_y \xi_y + y_v) \end{aligned} \tag{7.8}$$

Where ξ is a canonical random variable drawn from a Monte Carlo sampling structure.

Step 8: Transform the ray origins back to the sun's position The final step in ray propagation is to transform the rays back to the original space. This is done by applying the transformation matrix, M_{sun}, to the rays.

An alternate approach to the eight steps described here, is to generate a plane using two orthogonal vectors normal to the sun vector. Together, these vectors will form a system of coordinates centred on the plane. Rays can be described as a linear combination of these base vectors and all calculations involved with setting up the View Grid can be handled in the plane-centred coordinate system. However, this approach was not used for SUNRAY as it requires transformations between multiple coordinate systems. From Section 3.3.4, working within one coordinate system is more intuitive and simplifies debugging for future development.

7.3. The Grid

The Grid is a method for the spatial subdivision of the area of a scene. A discussion of spatial subdivision schemes and a justification of using a grid is provided in Appendix E.2. This section introduces the main concepts behind the Grid and discusses how a ray transverses it.

The Grid is conceptually similar to the View Grid and can be considered as a 3D extension of the techniques already discussed, with the key exception of what is stored in the cells. The cells of the View Grid only store a Boolean value of 1 for active and 0 for inactive cells, whereas the cells of the Grid store a list of all the objects that fall within each cell. If an object spans several cells then it must be included in more than one list (to reduce memory usage, a pointer to the object is stored). Due to their similarities with the View Grid, the techniques for setting up the Grid and storing objects (akin to activating cells) are not dicussed here. Instead, the following section discusses a more important aspect of how a ray moves through the Grid.

A ray transverses the Grid in a strict order so that once the ray hits an object, the process stops. This reduces the number of unnecessary intersection tests. Figure 7.6 illustrates how a ray can move through a 2D grid. The ray first hits the cell labelled *a* and, since there are no objects in that cell, it moves on to cell *b*. The ray then tests for intersection with all the objects in cell *b*, in this case only *object 1*. Because the ray misses *object 1*, it moves on to the next cell. The ray then tests for intersection with all the objects in cell *c*. The ray hits both *object 2* and *object 3*, but only the nearest hit point of the two is recorded. Since the ray has hit an object, the process stops.

Due to the sequential transversal of the Grid, there is no need to test for intersection with *object 4*. *Object 5* is also not tested as it is not in any of the cells that the ray transverses. Without a grid, SUNRAY will test all objects in the scene.

The transversal algorithm used in SUNRAY, illustrated in Figure 7.7, was first proposed by Amanatides & Woo (1987). The algorithm begins by first determining which cell contains the origin of the ray. For secondary rays this is found using a

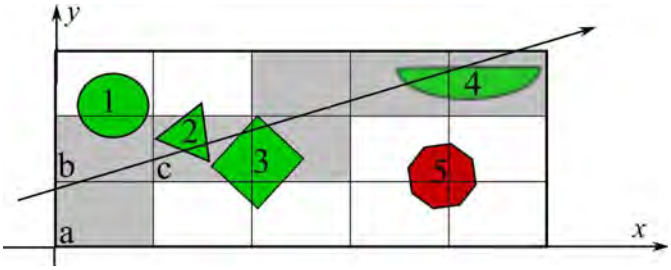


Figure 7.6: Object intersections as a ray transverses the Grid

similar process to step 6 in Section 7.2. For primary rays this is found where the primary ray enters the Grid.

Next, the value of δt is determined. This value represents the length of the ray for the horizontal component of the ray to equal one cell width and is calculated by

$$\delta t_x = (t_{xmax} - t_{xmin})/n_x \tag{7.9}$$

Where n_x is the number of cells in the x-direction and t_{xmax} and t_{xmin} are found using Equation (7.1). Similar expressions are used to determine δt_y and δt_z.

The next step is to determine how far along the ray can move before it enters the next cell and the final step is to determine which cell it enters. To do this, the value of t at which the ray crosses the first vertical cell boundary is determined from

$$t_{xnext} = t_{xmin} + (i_x + 1)\delta t_x \tag{7.10}$$

Where i_x is the same indexing system used in Equation (7.6). A similar computation is used to find t_{ynext} and t_{znext}. From Figure 7.7, if t_o is the point where the ray enters the cell, then t_{xnext} is the value of t where the ray hits the next x face. Similarly, t_{ynext} is the t value where the ray hits the next y face. If $t_{xnext} < t_{ynext}$ the ray enters the next horizontal cell (in the x direction), otherwise the ray moves to the next vertical cell (in the y direction). SUNRAY continues trace a ray through the Grid until an either an object has been hit or the ray exits the Grid.

(a) Ray moving to horizontal cell (b) Ray moving to vertical cell

Figure 7.7: Grid transversal

7.4. The PRNR method

A ray tracer can be an invaluable tool for any CSP researcher. Most ray tracers run until a predefined number of rays, chosen by the user, have been traced. The accuracy of a ray tracing simulation is dependent on the number of rays traced and differs for every scene, which leads to the question of how many rays are enough. For a non-optics specialist the number of rays is an abstract concept and a higher number is usually seen as better. However, this can lead to long computational times as a needlessly large number of rays are traced. On the contrary, too few rays can produce erroneous results. Even for the optics specialist, the choice of how many rays to trace is so seemingly arbitrary that often a simulation is run several times, with a various number of rays, in order to obtain an idea of how many rays are required.

In this section a new method which predicts the minimum number of rays required for a simulation is presented. The proposed method, namely the Predict the Required Number of Rays (PRNR) method, was published in the proceedings of the peer reviewed 2013 SolarPACES paper (Bode, Gauché & Griffith, 2013). The PRNR method was developed in order to help a user, who is unfamiliar with ray tracing, to use SUNRAY. However, the algorithm is universal in that it can be implemented in most ray tracing codes.

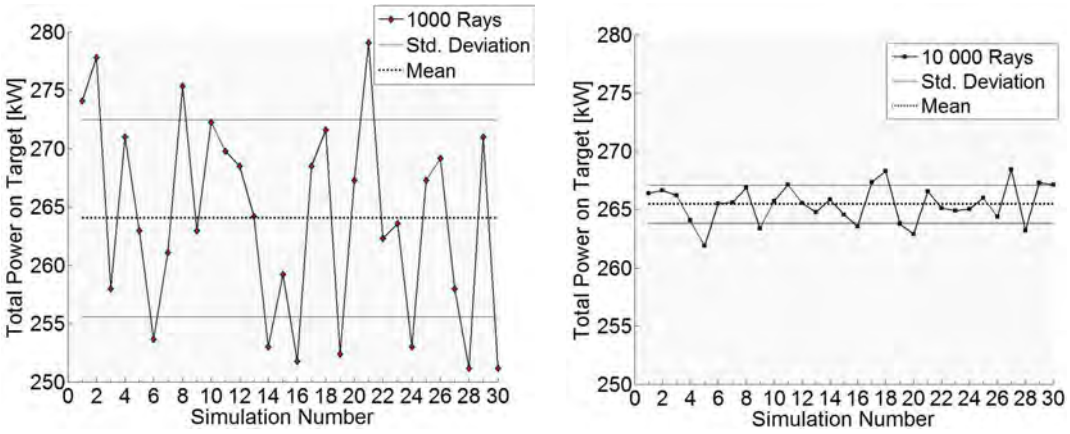
The value of the PRNR method is that, instead of defining a number of rays, a user only has to define an acceptable (tolerable) error in the result. The PRNR method then computes how many rays are required to ensure that the results fall within this error limit.

7.4.1. Development of the PRNR Method

The PRNR method uses statistical tools to predict the minimum number of rays required to obtain an adequate solution to a simulation. The solution is defined here as both the total power and the spatial resolution of radiant flux on a target if an infinite number of rays were to be traced. It should be noted that even if an infinite number of rays could be traced, a ray tracer can never fully simulate a real CSP system, as every input into a ray tracer is a potential source of error, for example, an imprecise description of the reflecting surfaces. This method models only one type of error, i.e., shot noise.

First discovered by Walter Schottky in 1918 (Perepelitsa, 2006), shot noise arises in ray tracing due to the arrival of a discrete number of independent and randomly generated rays. Shot noise is observed as fluctuations in the results of a repeated simulation. In ray tracing the Signal-to-Noise Ratio (SNR) of a result is proportional to the square route of the number of rays which hit the target.

Therefore, a large SNR requires a large number of rays. Figure 7.8 shows the fluctuations in the results of repeating the same simulation on a random field of 200 flat heliostats.



(a) 1 000 rays (b) 10 000 rays

Figure 7.8: A simulation of 200 random heliostats

The development of the PRNR method is based on the observation that, because each ray can be traced independently from the next, running a simulation with an infinite number of rays, $N = \infty$, is equivalent to repeating a simulation with a finite number of rays, $N \geq 1$, an infinite number of times, $n = \infty$.

From Figure 7.8, if a simulation with a finite number of rays is repeated n times, where $n > 0$, the results of each simulation will fluctuate about a certain mean value, μ_N. As n is increased ($n \to \infty$), the mean will tend toward the correct solution ($\mu_N \to \mu_\infty$) with a variance of σ_N. Thus, the solution to a simulation can be found if an adequate inference of μ_∞ can be made. This concept is illustrated in Figure 7.9.

From Figure 7.9, for each ray increment, the simulation was repeated $n = 10^5$ times. The mean value of each repeated simulation has been plotted with the vertical lines representing the variance in each of the the results. The figure shows that even for as few as 50 rays, the long-run mean solution is within 0.3% of the analytical (deterministic) solution, although with a large degree of variance.

In practice, a simulation can only be repeated a finite number of times. However, if n is large enough, a point estimate (Lyman Ott, 1988) of the mean, μ_∞, can be found from the standard arithmetic mean.

$$\bar{y} = \frac{1}{n}\sum_{i}^{n} y_i \tag{7.11}$$

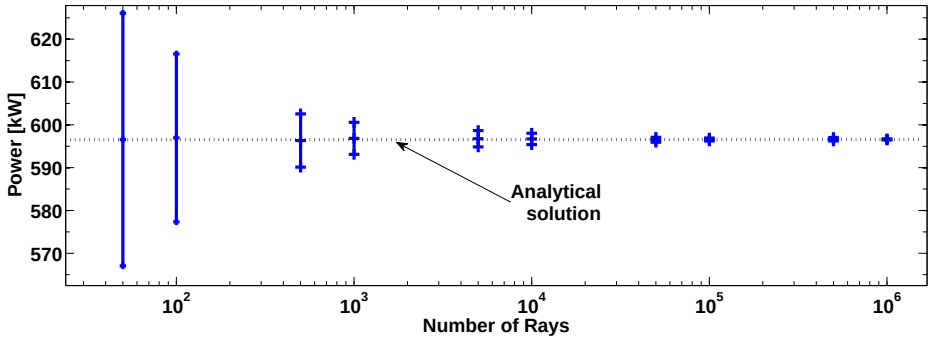


Figure 7.9: Results of a repeated simulation for various ray increments

The variance, σ_N, can be approximated from the standard deviation, s_n.

$$s_n = \sqrt{\frac{1}{n-1}\left[\sum_i y_i^2 - \frac{(\sum_i y_i)^2}{n}\right]} \tag{7.12}$$

Where y_i is the solution to the ith repeated simulation.

In Figure 7.10, the frequency of the results using 50 rays for the same simulation in Figure 7.9 is plotted. For a large set of simulations the Central Limit Theorem holds (Kroese *et al.*, 2011) and the results of each trace are normally distributed about μ_∞. Therefore, it is possible to use Gaussian distribution to set up an interval estimate of the mean, μ_∞.

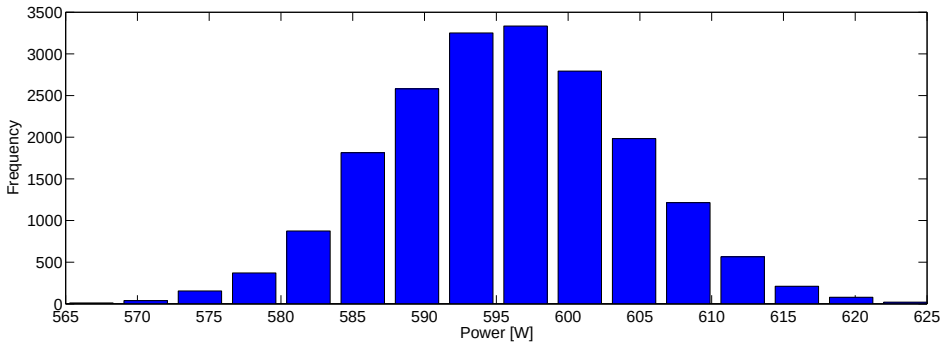


Figure 7.10: Frequency of results for multiple runs using a simple heliostat

For a Normal Gaussian distribution, approximately 99% of the area under the curve lies within $Z = 3$ standard deviations of μ_∞. This means that 99% of the $\bar{y}$ results will fall within the interval $\mu_\infty \pm Z\sigma_N$. However, it is unlikely that σ_N will be known and is it impractical to run a large number of simulations for s_n to be a reasonable estimate of σ_N. Therefore, the more variable Student's t-distribution with $\nu = n - 1$ degrees of freedom (Lyman Ott, 1988) is conservatively used.

The confidence interval estimate for μ_∞ is then

$$\bar{y} \pm t_{\alpha/2}\frac{s_n}{\sqrt{n}} \tag{7.13}$$

Where $(1-\alpha)$ is the confidence coefficient which determines a $100(1-\alpha)\%$ confidence interval for μ_N. The term on the right-hand side of the plus or minus in Equation (7.13) is the error, E, in the estimate.

$$E = t_{\alpha/2}\frac{s_n}{\sqrt{n}} \tag{7.14}$$

Three quantities influence the error and thus the width of the interval: the desired confidence level (which determines the t-value used), the standard deviation and the number of repeated simulations.

The confidence level, which is chosen by the user, needs to be carefully selected, as a low value will mean that, within the interval, there will likely be an error and, in ontrast, a high value will have a very large interval. For a fixed confidence level, the error can be reduced by increasing the number of repeated simulations. By rearranging Equation (7.14), an estimate of the number of simulations required can be obtained.

$$m = \frac{t^2_{\alpha/2}s^2_n}{E^2_T} \tag{7.15}$$

Where m is the number of times a simulation using N rays needs to be repeated in order to resolve the solution within a tolerable error, E_T, with a confidence of $100(1-\alpha)\%$. This is equivalent to running only one simulation with mN rays. Equation (7.15) is used to predict the number of rays and is the cornerstone of the PRNR algorithm.

7.4.2. The PRNR Algorithm

In the PRNR algorithm the user is required to define E_T, the tolerable error (the level of error which a user is willing to accept for a given spatial resolution) and α, the confidence level. Then, the only unknown in Equation (7.15) is the standard deviation. The first step in the algorithm is thus to obtain an estimate of s_n.

In order to obtain an estimation of s_n, a simulation is run twice, $n = 2$, with either $N_{initial} = 50$ rays per object or $N_{initial} = 50$ rays per division on the target, whichever is larger. The value of 50 has been empirically chosen as it was found to be a good balance of the least number of rays which can still provide a reasonable estimate of σ_n. The standard deviation between the two simulations is calculated using Equation (7.12).

The next step is to compute the error in the result using Equation (7.14). For a target subdivided into a number of cells, the error for each cell is calculated. If

this error is equal or below the tolerable error, the simulation will stop. However, in the more likely case where the error is greater than the tolerable error, Equation (7.15) is used to predict the number of simulations required. In the final step, the simulation is run with a total of mN rays. To further save on computational time, the results of the initial simulations are stored so that the final trace only requires $N(m-n)$ rays. The algorithm is summarised below:

SET Tolerable Error (E_T) and Confidence (α)
Run n simulations with $N = N_{initial} \times max$(Number of objects, Spatial Resolution)
Calculate s_n using Equation (7.12) and the Error (E) using Equation (7.14)
IF $E_T \leq E$
STOP
ELSE
Calculate the required number of simulations (m) using Equation (7.15)
Run one simulation using $N(m-n)$ rays

The default setting of the PRNR method is to run two initial simulations with 50 rays per object. With so few rays, the results between the two simulations can vary greatly. If, for example, the difference between the first two simulations is very large (a large s_n), the PRNR method could over predict the required number of rays. Conversely, if the results are similar, the PRNR method could under predict the required number of rays. To avoid this, the t-value is used. The t-value is a multiplying factor which ensures a large interval estimate that encompasses most results (depending on the confidence). However, it was found that these two situations are rare and for most validation cases the method suitably predicts the required number of rays. Two of these validation cases are presented in the next chapter.

7.5. Conclusions

In the absence of a quantitative comparison of all the available acceleration techniques, it is difficult to make an absolute decision on which ones will be best suited for SUNRAY. There is no standard technique and no fundamental reason why one technique should be preferred above all others. Keeping within the criteria of this thesis, the techniques incorporated into SUNRAY were based on performance, extensibility, and simplicity. In the next chapter various features of SUNRAY discussed thus far, including the PRNR method, are validated.

8. VALIDATION

This hapter presents a select number of validation test cases. These tests serve to validate the fundamental routines and algorithms of SUNRAY. Since it is not possible to present the entirety of the actual code in this thesis, this chapter also serves as validation of correct programming procedures. Some features, for example azimuth-elevation and single-axis tracking, are not presented as including every validation case is beyond the scope of this thesis.

8.1. Validation Methodology

The validation tests begin with simple behavioural tests and progressively increase in complexity. Once the fundamentals of SUNRAY have been validated, various aspects of the sun and material are tested in detail. Following this, more complex scenes which do not have simple analytical solutions are tested. These are comparative tests with the same simulation run in codes which have already been validated, namely SolTrace and Tonatiuh. The acceleration schemes are then tested, including a thorough investigation into the ray propagation method and the PRNR method. Finally, SUNRAY is experimentally validated by comparing the results with actual experimental tests.

8.2. Behavioural Tests

Behavioural tests illustrate the fundamental capabilities of SUNRAY. Some behavioural tests do not have any quantifiable solutions, but the ray traced images serve as sufficient validation. The behavioural tests presented here are for ray-object intersection, object transformation, shading, basic image formation, multiple reflection, and energy balance tests.

8.2.1. Ray-Object Intersection and Object Transformation

The first test is the basic, but important, ray-object intersection test, illustrated in Figure 8.1. Here, four heliostats are simulated and the (x, y, z) coordinate of the ray-object intersection points, the hit points, are plotted. The first three heliostats are square 1 m×1 m heliostats (the scale of the images has been distorted in MATLAB). The fourth heliostat is scaled to 1 m×2 m.

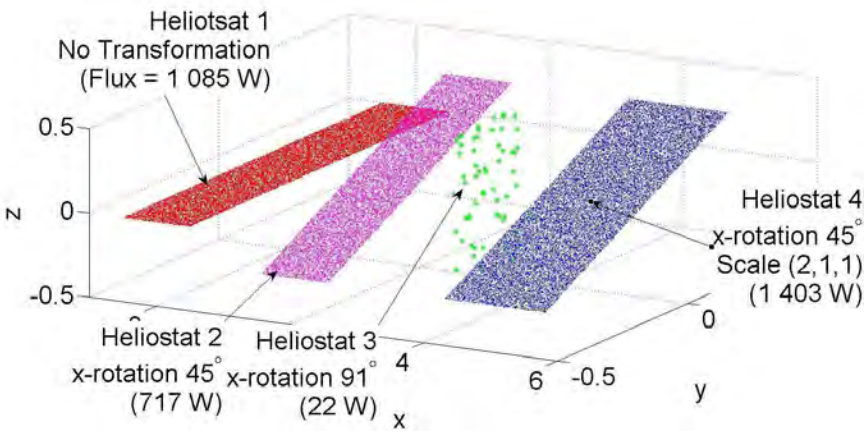
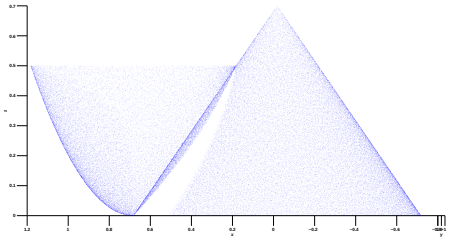


Figure 8.1: Intersection and transformation of four heliostats

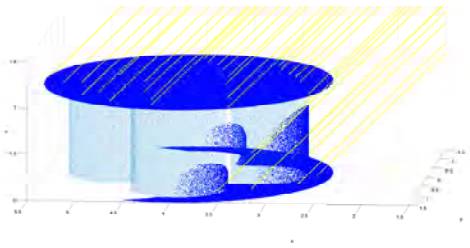
This test illustrates one of the most central aspects of SUNRAY: the ability to intersect objects. From the figure, it can be seen that less rays have struck Heliostat 3, which also has the lowest flux. This is because Heliostat 3 is orientated near parallel to the incoming rays, causing many to miss. Figure 8.1 also depicts a validation case of the transformation algorithms rotation, translation, and scaling.

8.2.2. Shading

Two shapes, a cone and paraboloid, are illustrated in Figure 8.2a. The compound object, a cylindrical receiver, which is composed of two discs and four cylinders (representing boiler tubes), is illustrated in Figure 8.2b.



(a) Shading of primitive objects



(b) Shading of compound objects

Figure 8.2: Primitive objects, compound objects and shading

In Figure 8.2a rays are generated vertically above the objects. In the figure it can be seen that there are no hit points on the lower left-hand side of the cone. This is because the rays have hit the paraboloid first. This illustrates SUNRAY's ability to simulate various primitives as well as its ability to resolve shading. It also illustrates what happens when two objects intersect.

SUNRAY has no collision tests and therefore allows objects to overlap. It then returns the hit point of only the closest object. This is observed as part of the cone is protruding through the paraboloid.

In Figure 8.2b the sun is given an elevation of 45°. It can be observed that the upper disk is preventing most of the rays from striking the boiler tubes and only a small section of the tubes are consequently receiving light (the lighter points have been included to aid in visualisation of the cylinders). The figure demostrates how an object can be self-shaded in SUNRAY.

8.2.3. Basic Image Formation

The next validation test demonstrates basic reflection as well as beam spread due to a pillbox sunshape. To quantify the beam spread, cone optics and edge ray principles are used (Appendix E.1.3). A program was written in MATLAB to trace the cones from the edge of a flat heliostat. For the validation case presented here, the superposition of the images from three heliostats cast onto an arbitrarily orientated target is illustrated in Figure 8.3. The heliostats are located at $(120, 5, 0), (100, 20, 10)$, and $(100, 10, 10)$ and the target normal is set at $(-0.7, 0.7, -0.2)$. The ellipsoids in Figure 8.3 represent the analytical boundary of the images formed by the heliostats.

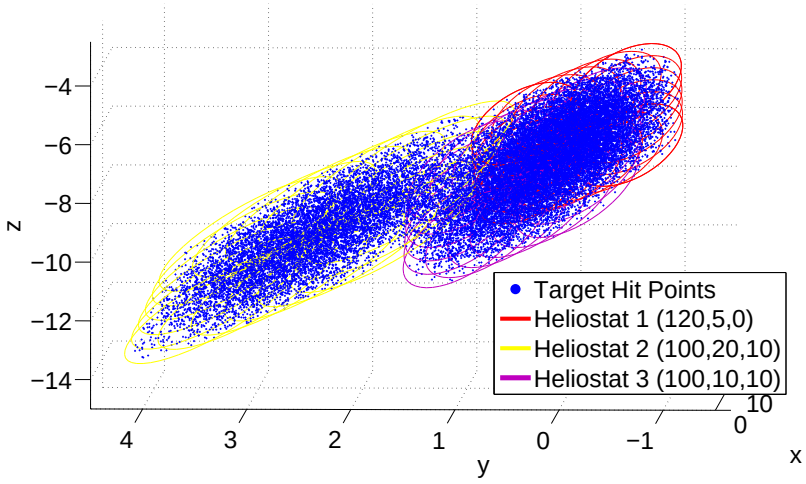


Figure 8.3: Superposition of three heliostat images on a target

In Figure 8.3 all the hit points are contained within the boundary of the ellipsoids. This shows that SUNRAY is able to correctly resolve the outer shape of the flux on the heliostat (Section 8.3.1 quantifies the flux distribution within the ellipse).

8.2.4. Multiple Reflection

To illustrate multiple reflections, a Compound Parabolic Concentrator (CPC) is traced with the sun at two different angles, as illustrated in Figure 8.4. When the incoming rays are perpendicular to the aperture of the CPC, illustrated by the green rays, all the rays are reflected to the focal point of the CPC. However, if the source is not aligned normal to the aperture, the magenta rays, the rays are re-reflected internally and many eventually leave back through the aperture.

This is also an example of how SUNRAY can assist in CPC design and highlights the versatility and use of ray tracing in many fields of CSP research.

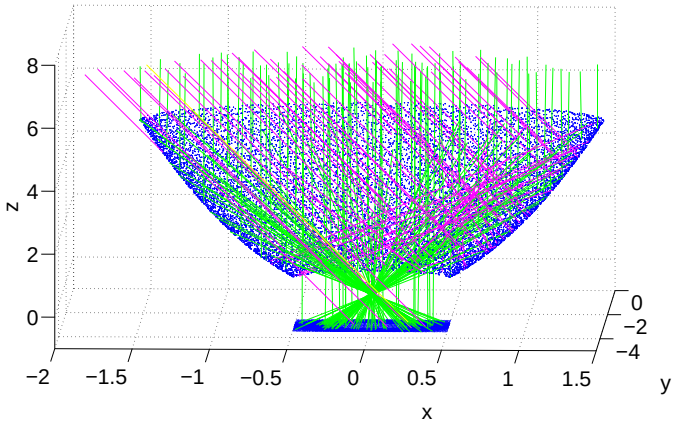


Figure 8.4: Multiple reflections in a CPC

8.2.5. Energy Balance

The final behavioural test is an energy balance test. A scene with two simple reflective (k_r=0.89) 1 m×1 m mirrors and a perfect absorber (a non-reflecting target) is illustrated in Figure 8.5. Two virtual surfaces have also been included. A virtual surface is a surface which does not interact with the ray but records ray intersections. Virtual surfaces were developed in SUNRAY to determine power and flux passing through an area. The total number of rays and power on each surface is tabulated in Table 8.1. The effects of atmospheric attenuation have also been included using the Vittoe-Biggs model. To simplify the test rays have only been generated above the first reflecting surface.

The energy balance test shows that, for a scene, no rays are created or destroyed and every ray that enters the scene can be accounted for. Likewise, the total energy carried by the rays as they are reflected through the scene is balanced. This is observed as a reduction of 0.89 of the power after the rays have been reflected off each surface.

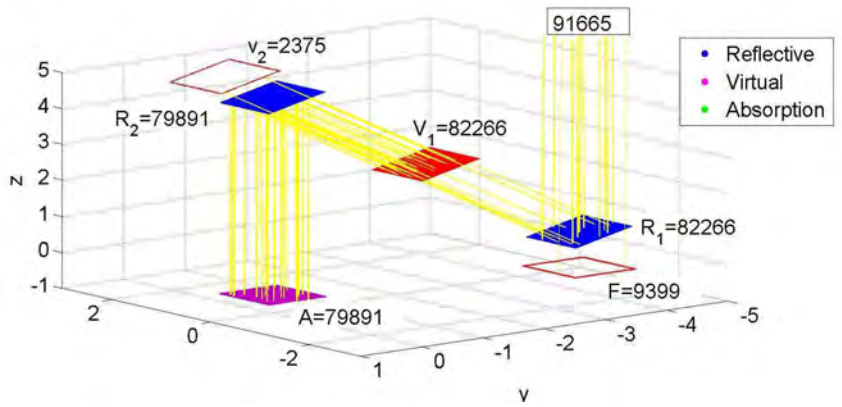


Figure 8.5: Energy balance for multiple reflections

	Number of Hits	Total Power [W]	Atmospheric Attenuation [W]
Reflective 1	82 266	923.64	923.64
Virtual 1	82 266	822.04	816.20
Virtual 2	2 375	23.73	23.51
Reflective 2	79 891	798.31	792.64
Absorber	79 891	710.50	705.30
Floor	9399	105.53	105.53
Total Rays	91 665	950.31	-

Table 8.1: Energy balance for multiple reflections

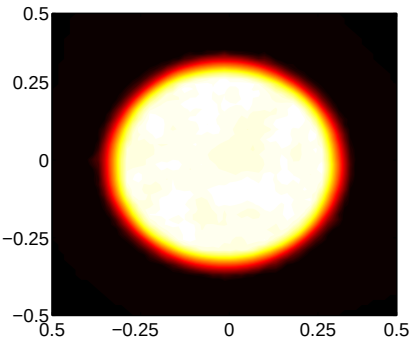
8.3. The Sun

This section validates various features of SUNRAY regarding the sun. This section begins with a formal validation of the sunshape, followed by validation of the position and tracking algorithms. The validation of blocking also is included in this section.

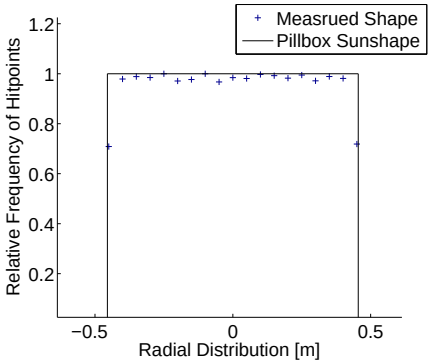
8.3.1. The Sunshape

To validate the various sunshapes, an imaging heliostat (paraboloid with a focal length of 100 m) is used to cast an image of the sun onto a target positioned at the heliostat's focal length. The four images below are for a pillbox, a Gaussian, a Buie, and a random user-defined sunshape.

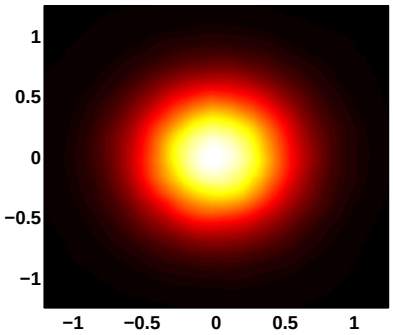
The flux maps of the different sunshapes were developed by discretizing the target into a fine grid. The number of rays which hits in each cell as well as the contribution of flux of each ray is tallied.



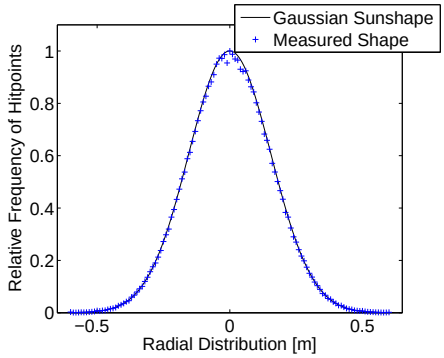
(a) Pillbox fluxmap



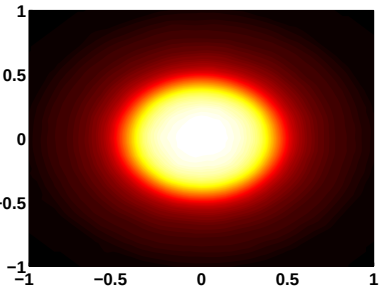
(b) Pillbox sunshape



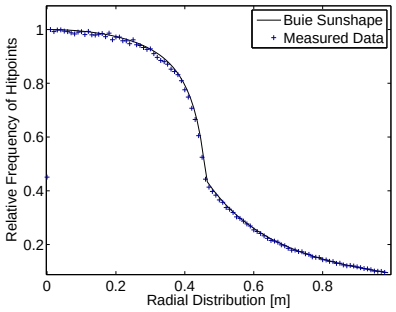
(c) Gaussian fluxmap



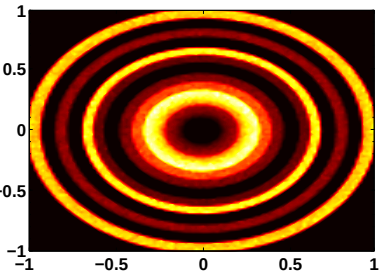
(d) Gaussian sunshape



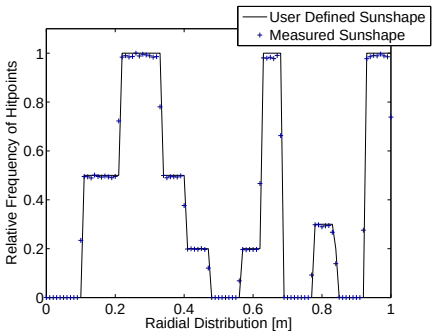
(e) Buie fluxmap CSR = 40%



(f) Buie Sunshape CSR = 40%



(g) Random Sunshape Fluxmap



(h) Random Sunshape

Figure 8.6: Fluxmap of various sunshapes

The total power incident on each target in Figure 8.6 is the same, although the distribution of power is different. The Gaussian sunshape in Figure 8.6c shows a higher flux value at the centre, but reduces significantly further away from the centre when compared to Figure 8.6a. Although Figure 8.6g does not represent any real sunshape, it has been included to demonstrate the ability of SUNRAY to parse any user-defined sunshape file. The band of lower flux in Figure 8.6e is due to the 40% circumsolar radiation.

The Buie sunshape is defined using a discrete number of points. It is advisable to use a larger number of points when defining a user-defined sunshape as this will ensure a better curve fitting. In Figure 8.6e a resolution of 0.1 mrad is used and the results show that this produces the correct sunshape. However, an indepth investigation between simulation time and resolution is out of scope of this thesis.

The results in Figure 8.6 show that each sunshape in SUNRAY follows the correct distribution.

8.3.2. Tracking and Solar Position

For the validation of dual-axis tracking, two heliostats, placed at random positions around a target, are simulated. The test evaluates whether the heliostat reflects sunlight correctly onto the target for various hours of the day. The test presented here is for SE tracking at two-hour increments over a 14 hour period, as shown in Figure 8.7.

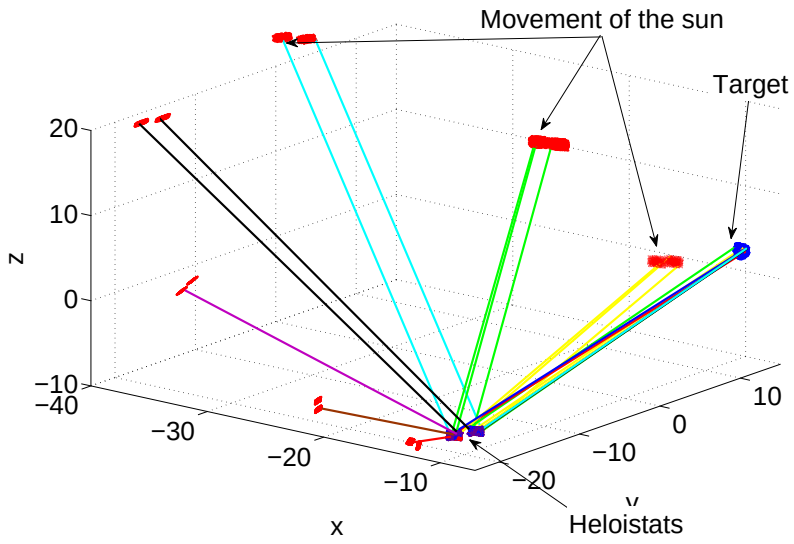


Figure 8.7: Tracking of the sun over 14 hours

Figure 8.8 shows the superposition of the hit points on the target due to the two heliostats at midday. Similar images have been captured for the other time intervals and it was found that the reflected images of each heliostat remain on the target for the entire day. This shows correct implementation of the SE tracking algorithm.

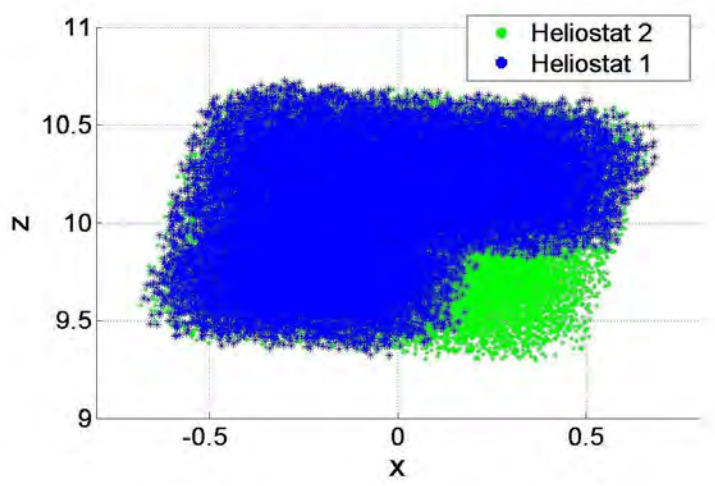


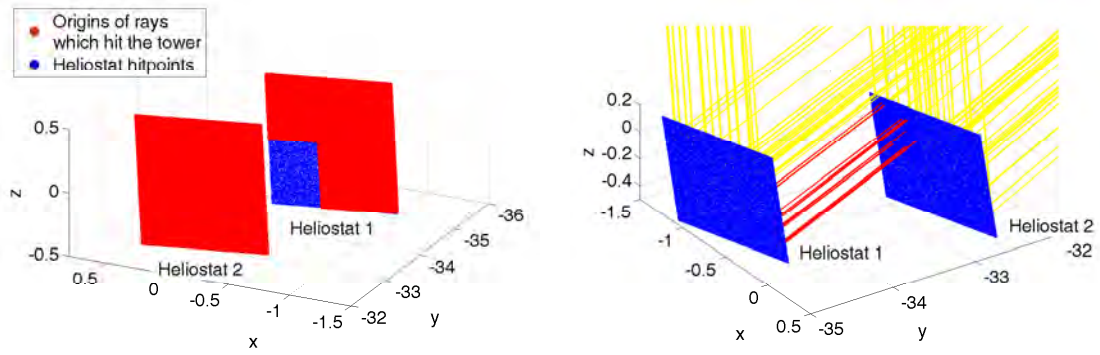
Figure 8.8: Hit points on target

In addition to tracking, this test also validates the correct implementation of the SOLPOS algorithm into SUNRAY. Using a third-party code can introduce possible bugs into SUNRAY, but the SOLPOS code has been developed by a reputable source (NREL, 2000) and it was felt that the C source code could be safely implemented without affecting the integrity of SUNRAY. This test has also been used to validate blocking, an important feature in ray tracing.

8.3.3. Blocking

In Figure 8.8, there are no hit points for Heliostat 1 on the lower right corner of the target. This is because rays originating from Heliostat 1 were blocked by Heliostat 2. Figure 8.9a provides a view of the two heliostats from just next to the target. The red points are the origins of the rays which hit the target and the blue points are the hit points on the heliostats. Due to blocking, no rays which hit the target originate from the lower left corner of Heliostat 1. Figure 8.9b is provided as a visualisation of the blocked rays hitting the back of Heliostat 2.

This test also validates SUNRAY's ability to correctly determine which side of the object the rays have intersected. This can be observed by the fact that the rays which hit the back of Heliostat 2 are not re-reflected.



(a) Origins of rays which hit the target (b) Rays being blocked by second heliostat

Figure 8.9: Blocking of two heliostats

8.4. Material

This section tests various materials available in SUNRAY.

8.4.1. Specularity

The effects of various specularity errors are illustrated with four flat heliostats in Figure 8.10. Once again, cone optics have been used to determine the maximum size of the image cast onto the target. The mirrors are orientated at 45° relative to the horizontal plane and located 100 m from the target. The sun is modelled as a pillbox sunshape with a 0.00465 rad subtend angle and is directly overhead.

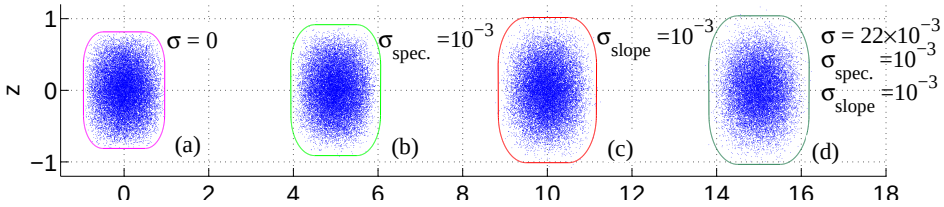


Figure 8.10: Specularity errors

In Figure 8.10 a larger beam-spread is observed for mirrors with larger σ errors. Comparing Figures 8.10b and 8.10c shows how slope errors make a greater contribution to beam-spread compared to specularity errors. Interestingly, there are some rays which are not contained within the borders. This is due to the underlying Gaussian distribution. There is a 99.7% probability that a normally distributed random number, ξ, will have a value between 0-3. To clamp the value between the range 0-1, it is scaled by $\xi/3$. The outliers represent approximately 0.3% of the rays which have a value greater than $3\sigma_{slope}$.

To quantify the number of outliers, the same simulation is repeated 50 times. It was found that the average number of rays which do not fall within the borders is in the order of 0.3%.

8.4.2. Reflectance

The reflectance of various materials is illustrated in the flux maps in Figure 8.11. In the figure four imaging heliostats with a focal length of 100 m each and a constant specular BRDF are simulated.

Figure 8.11 demonstrates that SUNRAY adequately simulates the reflected flux. For each material the relative difference in the simulated and analytical value of reflected flux is no more than of no more than 0.53%. For example using a reflectance of 0.5, the total power on the target is just over half that for a perfect reflector (the image at the upper left corner).

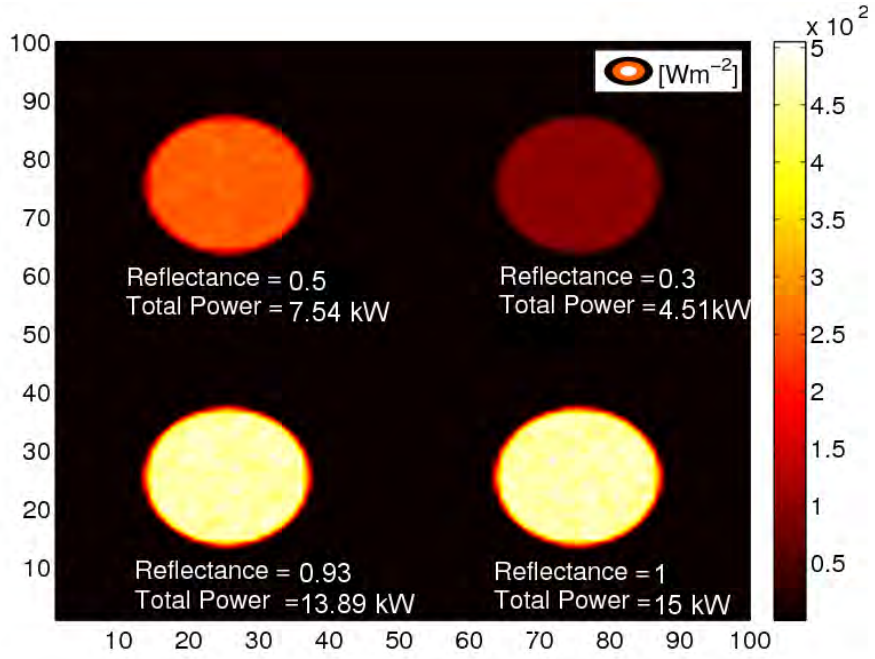


Figure 8.11: The reflectance of four heliostats using a constant specular BRDF

8.4.3. Fresnel Reflection and Real Materials

Three polished (containing no specularity errors) materials have been simulated in Figure 8.12. These are silver (Ag), copper (Cu), and duralumin, an aluminium-copper alloy. The average value for the refractive index and absorption coefficient

between the red and blue wavelengths has been used for this simulation. The heliostats have been positioned at 22.5° to the incident rays.

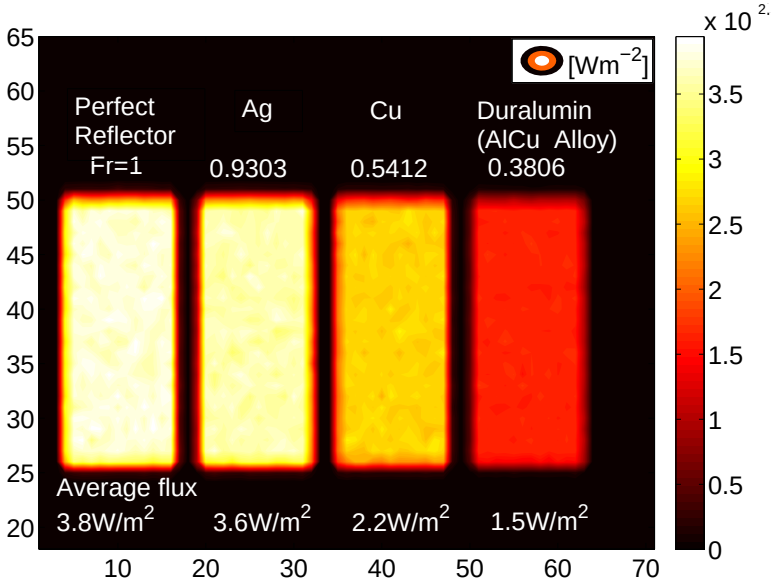


Figure 8.12: Fresnel reflection from four non-imaging heliostats rotated at 10°

The results in Figure 8.12 illustrate how SUNRAY can be used to simulate reflection off real surfaces.

8.5. Comparative Tests

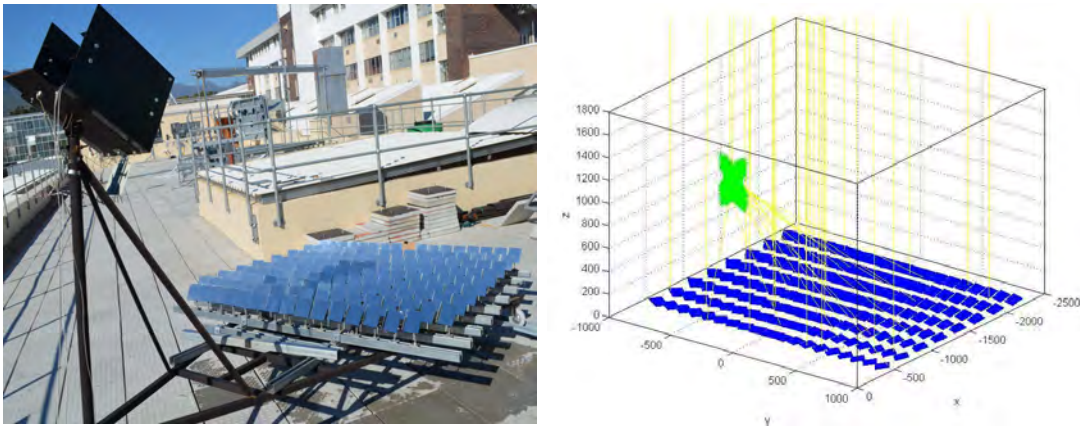
The validation of various features of SUNRAY has, thus far, been tested in isolation. This section tests the combination of the various features of SUNRAY. For these tests there is no closed-form analytic solution and thus, to quantify the results the two test cases are compared with the validated ray tracers, Tonatiuh and SolTrace.

The following test is an example of how SUNRAY can be applied in research projects at STERG.

8.5.1. SolTrace: Total Power and Simulation Time

In a study on CSP receivers, Kretzschmar (2013) constructed a scaled-down heliostat field on the solar roof laboratory at STERG, depicted in Figure 8.13a. Using the same parameters, the simulation is run in both SolTrace and SUNRAY,

illustrated in Figure 8.13b. Atmospheric attenuation, tracking errors, and Fresnel reflection are not included in the simulation as these cannot be modelled in SolTrace.



(a) Scaled down heliostat field (b) SUNRAY simulation

Figure 8.13: Simulation of a mini-heliostat field

Kretzchmar's study is primarily focused on the total incident power on the receiver aperture and therefore in this validation test, only the total power is recorded and not the flux distribution. The results as well as the time of the simulations are tabulated in Table 8.2.

	SolTrace	SUNRAY	Difference
Total Power on Target	880.453 kW	879.662 kW	0.09%
Total Rays Traced	2 700 137	2 700 000	-
Total Time	79.309 s	68.82 s	×1.152
		(28.79s for trace)	(×2.7547)

Table 8.2: Comparison with SolTrace and SUNRAY

Table 8.2 shows that, on average, SUNRAY and SolTrace have very similar results, with a relative difference of less than 0.1% with regard to total power.

Comparing the times of the simulation, it is observed that using a single processor and a similar number of rays, the execution time of SUNRAY is less than that of SolTrace. For SUNRAY, this time includes the initial build phase as well as the time take to write the results to a CSV file.

Currently the function which writes the results to CSV is inefficient and makes the greatest contribution to the simulation time. The time it takes for just the ray tracing loop to execute in SUNRAY, is only 28.79 s, which is almost a third of the time for the same simulation run in SolTrace. This suggests that SUNRAY has simulation times comparable to, if not better than, SolTrace.

8.5.2. Tonatiuh: Image formation

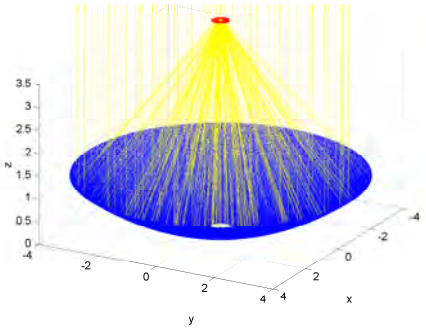
In a validation study of Tonaituh against SolTrace, Blanco (2009) simulated the Eurodish. The Eurodish was again used by Roccia *et al.* (2012) to validate and compare Solfast to Tonatiuh and SolTrace. It is possible to perform a direct comparison between SUNRAY and Tonatiuh as the model of the Eurodish is available for download (Mutuberria, 2009). This is an important validation test as a comparison can be performed between both the total power and the distribution of the hit points. Using the same parameters as in (Mutuberria, 2009), the Eurodish is simulated in SUNRAY, Tonatiuh (Version 1.2.6) and SolTrace (Version 2012.3.28). Figure 8.14a is a visualisation of the Eurodish in SUNRAY.

The results for the total power on the receiver are tabulated in Table 8.3 and show that SUNRAY and Tonatiuh compare well, with a relative difference in magnitude of less than 0.5%.

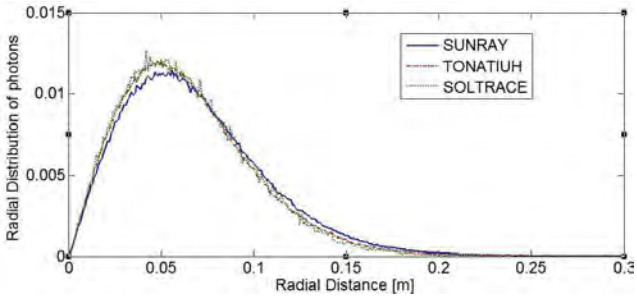
	Total Power on the Target
Tonatiuh	56 685.7 kW
SUNRAY	56 459.0 kW
Difference	0.4%

Table 8.3: Total power on Eurodish receiver

A plot of the radial distribution of rays on the receiver is provided in Figure 8.14b. Again, SUNRAY compares well to Tonatiuh and SolTrace, with a maximum relative difference of no more than 3%. Solfast could not be obtained an so a direct comparison has not been performed. However, since the results of the simulation with Tonatiuh and SolTrace are similar to SUNRAY and the results of the Solfast for the Eurodish simulation are quoted to be no more than 5% (Roccia *et al.*, 2012), it is likely that SUNRAY will have results similar to Solfast in such a simulation.



(a) Eurodish visualization



(b) Radial distribution of hit points on the Eurodish receiver

Figure 8.14: Eurodish simulation

8.6. Acceleration Techniques

The preceding tests have successfully validated the core functions of SUNRAY. Expanding on those, this section tests three acceleration techniques in SUNRAY.

8.6.1. View Grid

The simulation discussed in this section illustrates the efficiency of the method for ray propagation. In the first simulation 1500 flat 1 m×1 m rectangular heliostats, placed in a golden-ratio spiral, are simulated on a arbitary summer morning, as depicted in Figure 8.15. Two simulations with 2×10^6 rays are traced. In the first simulation a multiplying factor of $m_{vg} = 0$ (i.e., no View Grid) is used. In the second simulation a multiplying factor of $m_{vg} = 19$ is used. The simulation is run with the Grid switched off. The results are provided in Table 8.4.

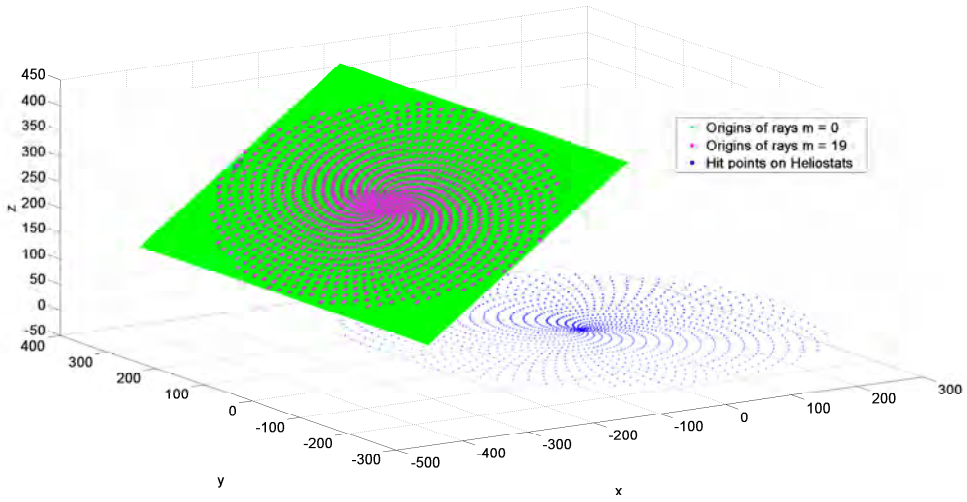


Figure 8.15: Goldenratio field layout

Table 8.4 indicates that for $m_{vg} = 19$ not all the rays have been traced. This is because SUNRAY has distributed the rays evenly among the View Grid cells and the remaining rays have not been generated. What the table shows is that even though less rays have been traced, 3.954 times more rays have intersected objects compared to $m_{vg} = 0$.

When comparing simulation times, the total time for $m_{vg} = 19$ is longer than for $m_{vg} = 0$. This is because rays which intersect objects have a higher computational overhead compared to missed rays. Moreover, the computational cost of testing

ray-object intersection typically overshadows all other operations. In comparison, the time spent setting up the View Grid is negligible compared to the total running time.

m_{vg}	Total rays traced	Total intersected rays	Total simulation time [s]	Time spent setting up View Grid
0	2 000 000	33 953	10.341	-
19	1 771 652	134 250	13.614	0.438
Equiv.	7 907 990	134 250	**40.89**	-

Table 8.4: Simulation results for goldenratio field layout

The final row of Table 8.4 shows how many rays need to be propogated, as well as the simulation time if an equivalent number of rays (134 250 rays) intersect objects if the View Grid is switched off. These results indicate that the View Grid substantially decreases computational time (by a factor of four in this simulation).

It was found that it is favourable to choose a large value for m_{vg}. With a larger multiplying factor, the cells of the View Grid are finer, which ensures that fewer rays are wasted. The extra computational cost of subdividing the View Grid into more cells is negligible when compared to the reduction in computational costs of missed rays.

Because the method for ray propagation is unique to SUNRAY, it had to be comprehensively tested. The test case presented here has been chosen as a fair representation of a typical simulation, in that the heliostats are not too densely or sparsely packed. However, in all the test simulations run with the View Grid a reduction in computation time is always observed. This highlights the value of the proposed method for ray propagation.

One possible issue, which was discovered when developing the View Grid, is illustrated in Figure 8.16. In the figure a simple flat heliostat has been simulated with 10^6 rays and with various values for m_{vg}. The number of rays traced, the number of rays which have hit objects, and the fraction of generated rays which hit the heliostat have been plotted.

In the figure two troughs are observed at $m_{vg} = 300$ and at $m_{vg} = 400$. These troughs occur because at these values the total number of rays to be traced cannot be equally distributed among all the active cells. This affects the number of rays which actually hit the target. However, this situation is still better than when no View Grid is used (from Figure 8.16, when $m_{vg} = 0$, only about 34% of the generated rays intersect the heliostat). To ensure that all rays are traced SUNRAY selects the largest possible value for m_{vg}, while still ensuring an even distribution of rays.

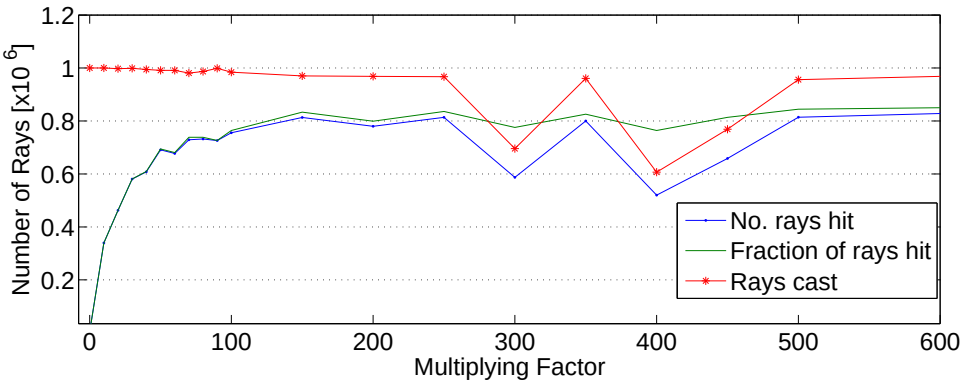


Figure 8.16: Simple scene with various values for the View Grid multiplying factor

An inherent advantage of using the View Grid is that rays are evenly distributed across the View Plane, as indicated in Figure 8.17. This is similar to the Jittered sampling pattern described by Shirley & Marschner (2009). Figure 8.17 demonstrates two potential problems if a purely randomly distribution is used ($m_{vg} = 0$). When purely random distribution is used no rays are generated in the upper left corner and a cluster of rays are generated in the lower right corner. In comparison when $m_{vg} = 50$ a much more even distribution is observed. These clusters and open spaces can introduce errors in the calculation. For this reason pure random sampling is often avoided in graphical ray tracers.

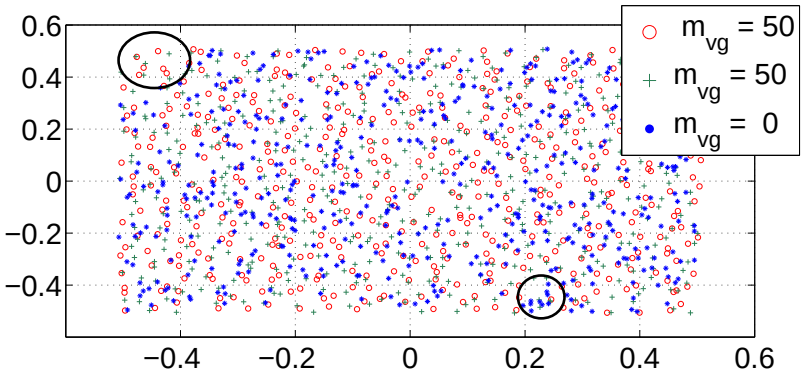


Figure 8.17: Distribution of rays on the View Plane for $m_{vg} = 50$ and $m_{vg} = 0$

8.6.2. Grid

The validation case for the Grid presented here is of a field of heliostats placed in a random, nominally circular arrangement around a tower. The total area has been fixed and therefore the size of the Grid remains constant. The View Grid has

been switched off and a Grid multiplying factor of $m = 2$ is used. The results are presented in Table 8.5.

Number of Heliostats	With Grid $[s]$	Without Grid $[s]$	Speed up
40	0.72100	1.49258	2.07015
180	0.96634	13.99987	14.487406
400	1.3300	32.432	24.330
800	2.347	75.862	32.310183
1 600	2.923	151.318	51.768
10 000	6.359	1 730.61	272.151

Table 8.5: Performance increase using the Grid with 10^6 rays and $m = 2$

Table 8.5 shows that a substantial speed increase is obtained with the Grid. Without using the Grid, the time for the simulation is linearly proportional to the number of objects. In stark contrast, the Grid provides approximately a $O\log(n)$ increase in speed in most cases.

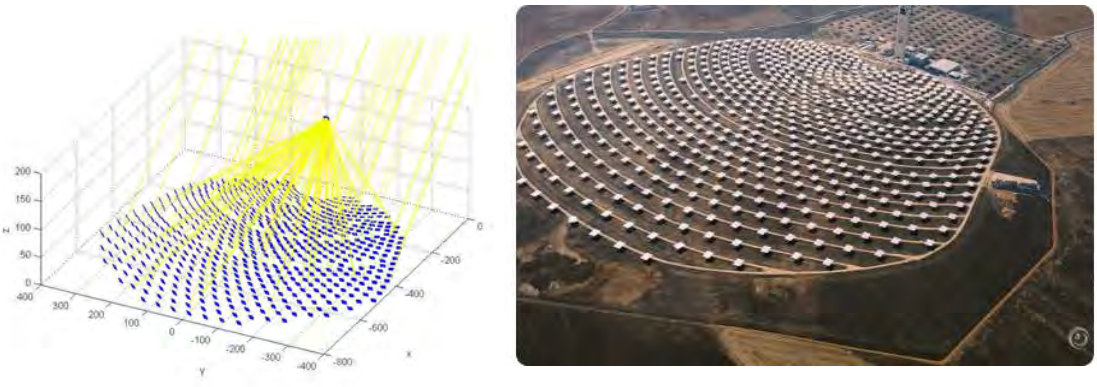
8.6.3. PRNR

The PRNR method was validated using a wide variety of test cases. The validation discussed in this section was published in a paper at the 2013 SolarPaces conference (Lutchman, Groenwald & Bode, 2013), and is for a simulation performed on a heliostat field layout designed by Lutchman (2013). This validation test case also serves as a second example of the applications and benefits of SUNRAY for research projects.

For Lutchman's research an optimisation code for a central receiver field layout was developed. SUNRAY was used to help validate the accuracy of Lutchman's code by running a number of detailed simulations at specific times. One of these time intervals is presented here.

The simulation is for an optimised Planta Solar 10 (PS10) field layout (Mills, 2004), illustrated in Figure 8.18. In the simulation 624 rectangular imaging heliostats (with a focal length equal to the distance to the receiver) were simulated on an arbitrary winter day with a DNI value of 1 000 W/m^2, a Buie sunshape with CR=20%, a Vittitoe-Biggs atmospheric attenuation model, a perfect specular reflection coefficient of $k_r = 0.89$ and a $\sigma_{combined} = 0.95$.

To test the PRNR method, the target was discretized into a grid of 20×20 cells (each cell equal to 0.7 m×0.7 m). The simulation was run twice with 20 000 rays. A tolerable error of 100 kW per cell or about 0.2% total intersected power of the PS10 plant at design point, was chosen.

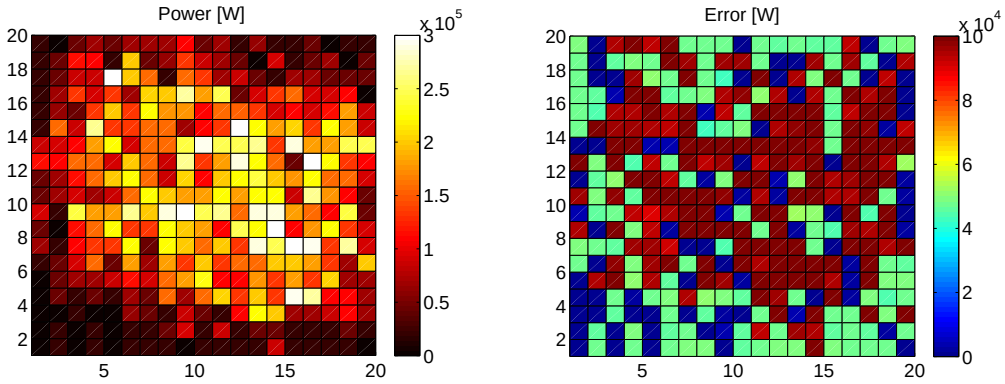


(a) PS10 optimised layout

(b) PS10 layout

Figure 8.18: The PS10 plant, Spain (Abeinsa, 2013)

The power, which is calculated from the number of ray hits per cell, and error, which is calculated from Equation (7.14), are illustrated in Figure 8.19. Numerous cells in Figure 8.19b have an error which exceeds 100 kW with maximum error being 953.98 kW.



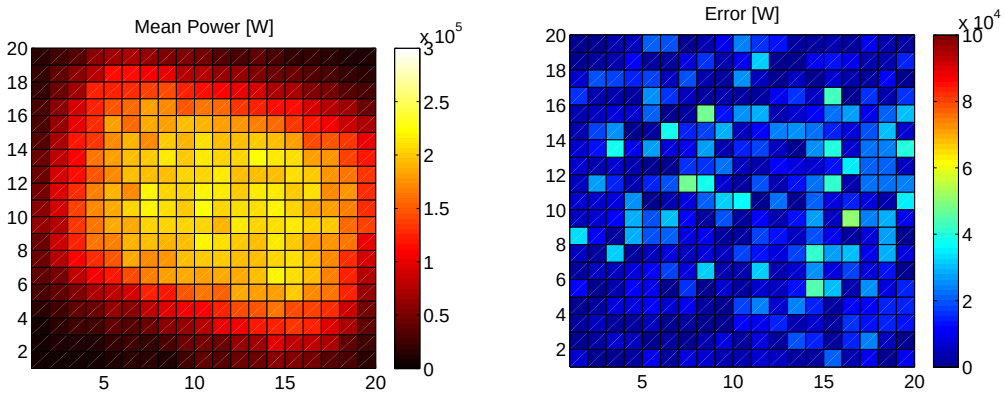
(a) Power

(b) Error

Figure 8.19: The power on the target running the simulation twice with 20 000 rays

Because the maximum error exceeds the tolerable error, SUNRAY automatically computes the required number of rays using Equation (7.15). The total number of rays required to reduce the maximum error below the tolerable error, with a 98% confidence, was calculated to be m =320 850 rays. To substantiate this, the results of a simulation run with $m/2$ rays is presented in Figure 8.20.

In the figure the error in all cells is below the required 100 kW with a maximum error of only 50.737 kW. This validates that the PRNR method does indeed predict the required number of rays.



(a) Power (b) Error

Figure 8.20: The power on the target running the simulation twice with 160 425 rays

A breakdown of the simulation times is provided in Table 8.6. The table shows that the time it takes to calculate the required number of rays is a fraction of the total simulation time. This shows that the simulation time is still proportional to the number of rays and, therefore, by tracing the minimum required number of rays, the simulation runs in the shortest possible time.

Simulation number	Simulation time [s]	Percentage
First initial simulation	3.221	10.66%
Second initial simulation	2.873	9.15%
Calculate error in each cell and m	0.002	0.01%
Final simulation	24.120	79.83%
Total simulation time	30.216	100%

Table 8.6: Simulation times for the PS10 validation case

8.7. Experimental Validation

The final stage in the validation procedure is to validate SUNRAY against real data. Several experimental validation tests were conducted over a number of days and in various weather conditions. The experiment is for three different heliostat profiles designed by Landman (2013). A description of the experimental design, procedure, and apparatus is provided in Appendix F. Only three of the most illustrative tests are presented here.

8.7.1. Image and Flux Sensor Correlation

The first test was conducted in order to correlate the sensor reading to that of the images captured. This was done by focusing the centroid of the image, which has been reflected from the heliostat, onto the sensor. Then, the image was allowed to drift off the sensor due to the movement of the sun. Images were captured at two second intervals. The images were calibrated to flux sensor readings by scaling them according to the maximum and the minimum (null) measured flux value. The flux values from the sensor and those derived from the calibrated images were then plotted on the same set of axes, as shown in Figure 8.21.

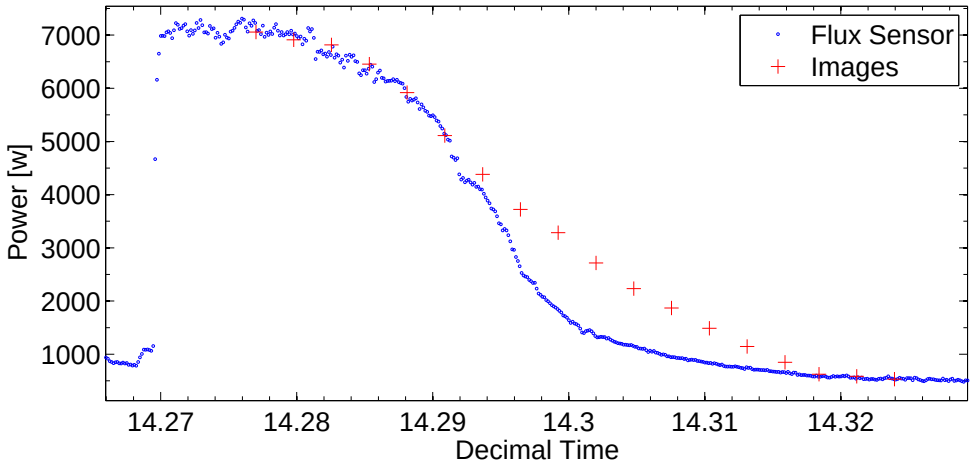


Figure 8.21: Flux sensor readings and calibrated image readings

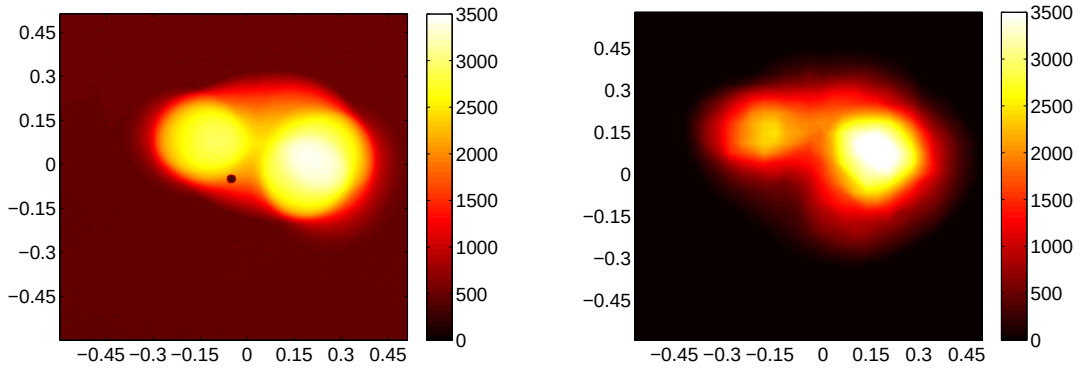
Figure 8.21 shows that the flux values obtained from the sensor and images correlate well at the image centroid. This is true until approximately 4000 W/m^2, where the results begin to deviate. Below 4000 W/m^2 a discrepancy in the results of up to 50% is observed. The results correlate again at the null value.

The discrepancy in the results indicates that the light reflected off the target is not linearly scalable with the camera images at low flux values. This is most likely due to the target not being a perfect Lambertian surface.

In order to correct the discrepancies a corrective function could be calculated. However, it was found that discrepancies at low flux levels are too irregular and no corrective function could be found. The experiment is still adequate to determine flux and total power on the target at higher flux values.

8.7.2. Image Formation

The second test case presented here is of a mirror which had been purposefully deformed in order to test whether SUNRAY is capable of resolving the correct flux map. The deformed mirror was scanned using a Coordinate Measurement Machine (CMM). SUNRAY has a function which converts scanned data from the CMM to PLY file format. The results of both the ray tracing simulation and experiment are illustrated in Figure 8.22



(a) Experimental image formation (b) SUNRAY image formation

Figure 8.22: Shape test flux map

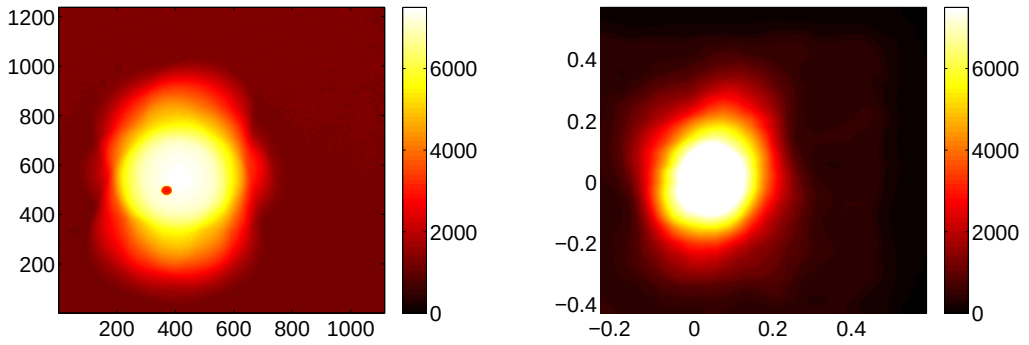
Comparing the two images in Figure 8.22, it can be seen that SUNRAY captures the apparent shape and orientation well. Two distinct areas of higher flux can be observed in both images although the area of flux on the left-hand side of the ray traced image is less than that of the experimental image. Moreover, the area of higher flux is slightly more on the right-hand side for both images. This may be due to incorrect measurement of the orientation of the target and heliostat in the experiment. Additional factors such as the deformation of the mirror caused by gravity or deformation during transit from the CMM could have an influence on the result.

8.7.3. Total Power on Target

The final test determines the total reflected flux on the target. For the experiment the total power on the target is calculated by integrating the the flux map obtained from the camera image depicted in Figure 8.23a.

The simulation was run in SUNRAY with 2×10^6 rays, a Buie sunshape (CRS=20%), a reflection of coefficent of 0.89, the Vittoe-Biggs clear sky model for atmospheric attenuation, and a DNI value of 920 W/m^2. The total power of the target was calculated as 528.80 W. From the experiment the total power on

the target calculated from the camera image to be 538.19 W. This value has already been adjusted to account for a Diffuse Horizontal Irradiance (DHI) value of 123.63 W/m^2 (University, 2013). This is a relative difference of only 1.74% betwen experiment and SUNRAY which indicates a good correlation between the two.



(a) Camera Fluxmap (b) SUNRAY

Figure 8.23: Reflection test

The peak flux from the camera image was calculate to be 7 334 W/m^2. The peak flux from SUNRAY (discretizing the target into 1 cm×1 cm bins) was 7 790.91 W/m^2. Here, the relative difference is 6.23%, which is higher than that of the total power but still shows very good correlation. The higher difference may again be due to incorrect measurement of the heliostat and target alignment.

8.8. Conclusions

The fundamental features of SUNRAY have been tested in this chapter. In each validation case SUNRAY fulfilled all of the objectives with satisfactory results, thus demonstrating correct algorithmic and programming implementation of these features.

The case presented in this chapter for the ray propagation technique showed a factor of four increase in time, compared to not using a View Grid. The validating case for the PRNR method demonstrates the application of SUNRAY in an actual research project conducted at STERG. It showed that, for the optimised PS10 field layout, the PRNR method is able to correctly predict the required number of rays for the scene with the simulation time reduced to less than a minute. In the experimental validation, the simulated results showed good correlation with the experimental results.

The following, concluding chapter discusses some of the main findings from this thesis.

9. CONCLUSIONS

This thesis contributes to the field of CSP research, by developing a Monte Carlo ray tracer for solar thermal systems. SUNRAY has been developed for use within STERG to aid in research and evaluation of the optics of CSP systems, in particular, central receiver systems and the SUNSPOT cycle. To demonstrate the operational range of SUNRAY a selection of the most relevant algorithms and features of SUNRAY have been presented in this thesis. These have been extensively validated through numerous simulations. This chapter begins with a brief summary of SUNRAY's capabilites followed by the concusions a key findings of this thesis. Finally some recommendations for future work are provided.

9.1. Summary of SUNRAY

SUNRAY is capable of simulating a variety of CSP systems due to its accurate description of the sun, numerous geometric objects, various materials, and acceleration techniques which have been incorporated. These components are briefly summarised below.

The Sun. The correct description of the sun is important for a ray tracer used in CSP research. There are four sunshapes programmed into SUNRAY. Among these are the Buie model, which was developed from actual solar observations, and the user-defined model, which simplifies defining a new sunshape for a user unfamiliar with Monte Carlo methods. Three tracking methods have also been included in SUNRAY. These simplify setting up a scene as a user does not need to input the orientation of the collectors.

Geometric Objects. A number of geometric objects have been programmed into SUNRAY. These represent the majority of the basic shapes which can be found in CSP systems. Real mirror surface profiles are simulated using scanned mirror data and TINS.

Materials and BRDF. There are two ways in which a ray can be reflected in SUNRAY. The first is the ideal perfect specular reflection model. The second, more accurate model, has been based on the SolarPACES guidelines for solar materials and the physically accurate Fresnel equations. SUNRAY also models the spectral distribution of the reflected flux.

Acceleration Techniques Four acceleration techniques have been implemented in SUNRAY, namely the Grid, bounding boxes, the method for ray propagation, and the Predict the Required Number of Rays algorithm. Acceleration techniques are vital for rapid simulations and without them simulation times for SUNRAY would have been prohibitively long.

9.2. Summary of the Key Findings and Contributions

Reviewing the available optical codes afforded the opportunity to identify features of these codes which could be improved on. Incorporating these into SUNRAY differentiates it from the other codes and allows it to be used in a greater number of research areas. Furthermore, the literature review conducted for this study is currently one of the most up to date reviews of CSP optical tools. The extended literature review on optical CSP codes was presented at the 2012 SASEC conference (Bode & Gauché, 2012).

By developing a new, unique ray tracing program, it was possible to create a program which is flexible enough to assist in a range of CSP simulations while remaining robust enough to be extended in future research. Future development of SUNRAY can contribute to the capacity of understanding in ray tracing at STERG.

Writing the program in C++ allowed for efficient memory management and good structuring of the algorithms and routines. This provided significant improvement in the execution time of the code. Simulating a scaled-down heliostat field test rig at STERG it was found that the execution time of SUNRAY had a 15% increase in speed compared to that of the proven solar ray tracer, SolTrace, with a difference in results of only 0.09%. Similarily, the relative difference for total power on the Eurodish receiver between SUNRAY and another proven ray tracer, Tonatiuh, was only 0.4% with a difference of no more than 3% for the flux distribution on the receiver.

It was found that the increase in speed and the reduction in simulation time gained from the various acceleration methods were far greater than the time it took for SUNRAY to implement these methods. For example View Grid provided a factor of four increase in simulation time whereas the setting up the View Grid represented only 1% of the total simulation time.

Incorporating physically accurate models in SUNRAY, for example the Buie sunshape and Fresnel reflection, allows for more accurate simulations and for the simulation of real CSP systems.

TINS and PLY files enabled SUNRAY to handle real 3D scanned data, and thus making it possible to simulate real CSP systems. This was demonstrated through the experimental validation of SUNRAY using heliostat profiles designed at STERG.

Two novel algorithms have been proposed in this project. The first is a method for ray propagation where the rays are only generated above objects in the scene and not over the entire scene. This reduces the number of ineffective rays, which increases the accuracy and reduces the simulation time of SUNRAY. The second algorithm predicts the required number of rays for a simulation. The algorithm can enhance the usability of ray tracing, allowing it to be accessible to a wider field of research, which may ultimately increase the research output for CSP. This algorithm was published in the proceedings of the double peer reviewed 2013 SolarPACES paper (Bode *et al.*, 2013).

SUNRAY has been used to help validate a field optimization code developed at STERG. Results from the optimised PS10 field layout were published in a second paper at the 2013 SolarPACES conference (Lutchman *et al.*, 2013).

9.3. Recommendations for Implementation and Future Work

Ray tracing is an active field of research and futher development of the SUNRAY can, for example, investigate alternate acceleration techniques, advanced shape handling for novel heliostat designs, new tracking algorithms or the effects of spectral distributions. Continued use and development of SUNRAY will increase the operational range of SUNRAY and help ensure that it remains up to standard. The following have been identified as key features which will encourage use and implementation of SUNRAY.

1. Design a user-friendly interface.
2. Implement parallel programming. This will ensure faster running time of the code and may also allow for optimisation algorithms to be incorperated.
3. Implement Transparency. It is suggested that transparency be implemented through BRDF class. Similar to a surface's BRDF, which describes the amount of flux reflected off a surface, the amount of flux transmitted through a material is given by the Bidirectional Transmittance Distribution Function (BTDF) (Shirley & Marschner, 2009)
4. Develop an internal visualisation routine for SUNRAY. This will ensure that CSV files do not have to be exported and will also have a significant improvement of the speed of the simulations.

Appendices

A. MONTE CARLO METHODS

Monte Carlo methods and random numbers are used throughout SUNRAY. This appendix provides some background theory to Monte Carlo integration and random numbers.

A.1. Background Theory

One-Dimensional Continuous Probability Density Functions

The probability density function is a function which describes the relative likelihood for a random variable, x, to take on a given value. The probability for a random variable to fall within a particular interval, $[a, b]$, is given by the integral of that variable's Probability Density Function (pdf) over the interval (Fleet & Hertzmann, 2009)

$$P[x_0 \leq x \leq x_1] = \int_{x_0}^{x_1} p(x)dx \tag{A.1}$$

Any real-valued function, $p(x)$, that satisfies the following equation is a valid pdf (Lyman Ott, 1988).

$$p(x) \leq 0 \quad \text{for all} \quad x \tag{A.2a}$$

$$\int_{-\infty}^{\infty} p(x)dx = 1 \tag{A.2b}$$

As an example, the canonical random variable's pdf is constant (Section A.2). Thus, the probability density fun tion for ξ is

$$p(\xi) = \begin{cases} 1 \text{ if } 0 \leq \xi \leq 1 \\ 0 \textit{ otherwise} \end{cases} \tag{A.3}$$

One-Dimensional Cumulative Density Function

The Cumulative Density Function (cdf) describes the probability that a real-valued random variable with a given probability distribution will be found at a value less than or equal to x (Shirley & Marschner, 2009). For a continuous function, the cdf is the area under the probability density function.

$$F_c(x) = \int_{-\infty}^{x} p(u)du \tag{A.4}$$

One-Dimensional Expected Value

The average value that a real function, f, of a random variable with underlying pdf, p, will take on is the expected value (Shirley & Wang, 1992). For a one-dimensional random variable, this is defined as

$$E(f(x)) = \int f(x)p(x)dx \tag{A.5}$$

The mean, μ, of a distribution, $p(x)$, is the expected value of that distribution. For example, given a set of N independent and identically distributed (iid) random variables, which share a common density, the mean is the sum of these variables, divided by the number of the variables (Shirley, 2003).

$$\mu \approx Ex \approx \frac{1}{N}\sum_{i=1}^{N} x_i \tag{A.6}$$

The estimate becomes more accurate as N increases.

A.1.1. Monte Carlo Integration

Because the expected value can be expressed as an integral, Equation (A.5), it is thus possible to express the integral as a sum, Equation (A.6). For the one-dimensional integral $\int f(x)dx$, given set of uniform random variables x_i, the Monte Carlo Estimator is given as (Shirley & Morley, 2003)

$$E(f(x)) = \int f(x)p(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i) \tag{A.7}$$

To approximate a single function, $g(x)$, rather than a product, it is possible to substitute $g(x) = f(x)p(x)$ into the integrand.

$$\int g(x)dx \approx \frac{1}{N}\sum_{i=1}^{N} \frac{f(x_i)}{p(x_i)} \tag{A.8}$$

The advantage of Monte Carlo integration is that the number of samples, N, can be arbitrarily chosen and is independent of the dimensionality of the integral. The only restriction is that $p(x)$ must be positive when $f(x)$ is not zero (Kroese *et al.*, 2011). This is an extremely important step, because it allows the integral in Equation (A.8) to be solved by drawing random numbers from an arbitrary pdf. This is known as importance sampling (Kroese *et al.*, 2011).

Importance sampling exploits the fact that the estimator in Equation (A.8) has a higher rate of convergence, as the samples are taken from the distribution, $p(x)$, that is similar to the function $f(x)$. Therefore, it is important to be able to generate random variables from a given arbitrary distribution Chen (2005).

A.2. Random Variable

Random numbers are used in many algorithms throughout SUNRAY. A random number generator produces an infinite set of random variables that are independent and identically distributed according to some distribution (Kroese *et al.*, 2011). When this distribution is on the interval $(0, 1)$, the generator is said to be a uniform random number generator. A uniform random number, called a canonical random variable, ξ , takes on all the values in the domain [0,1) with equal probability(Pharr & Humphreys, 2010).

Canonical random variables are important in ray tracing as they are easy to generate and are used to generate samples from arbitrary distributions (Problem 1, Section 3.2). However, an infinite sequence of random numbers is only a mathematical concept. The best that can be achieved in reality is a sequence of pseudo-random numbers with statistical properties that are almost indistinguishable from a true sequence (Robert & Casella, 2004). Many pseudo-random number generator (PRNG) algorithms have been developed with varying degrees of speed, efficiency, and theoretical support. The majority of these algorithms require an initial input (seed) value. The mark of a good random number generator is that it is reproducible, in that it is able to produce the same sequence of random numbers, given the same seed value (Kroese *et al.*, 2011). The random number generator used in SUNRAY is the Mersenne Twister[1] (Matsumoto & Nishimura, 1998), which has been specifically optimised for Monte Carlo methods.

A.2.1. Random Number Generation

Generating a random vector, $\mathbf{x}$, from an arbitrary distribution in the real space, $\mathbb{R}$, commonly involves two steps (Kroese *et al.*, 2011):

1. First draw uniform random variables $\xi_1, \ldots, \xi_k$ for some $k = 1, 2, \ldots$;and
2. Then $\mathbf{x} = F(x_1, \ldots, x_k)$, where F is some function

There are a number of methods for generating random variables from a given probability. Some of these methods are exact, in that each generated random variable has exactly the required distributions (Kroese *et al.*, 2011), and others are only approximate generation methods, such as a Markov chain (Winkler, 2003). Approximate methods are used when it is difficult or impossible to achieve exact

[1]The Mersenne Twister is the default random number generator used in programs such as MATLAB and Python. However, it has only relatively recently (2011) been incorporated in C++. Compilers developed before 2011 may need library extensions such as the TR1 (Austern, 2005) to run SUNRAY.

solutions. Methods have even been developed to generate random variables from a discrete distribution.

One of the procedures for exact random variable generation is known as the inverse transform method or function inversion (Kroese *et al.*, 2011). The algorithm is given in (Pharr & Humphreys, 2010):

1. Compute the cdf $F_c(x)$
2. Compute the inverse $F_c^{-1}(x)$
3. Draw uniform random variables ξ_1
4. Compute $x_i = F_c^{-1}(\xi_1)$

Function inversion has been used throughout SUNRAY. An example of how it has been used to generate random numbers according to a given sunshape is given in Appendix B.1.3.

A very important statistical distribution is the a Gaussian Normal distribution. However, there is no closed-form inversion for the Gaussian distribution function (Goodman, 2005). The celebrated Box-Muller transformation overcomes this by using a polar coordinate transformation (Box & Muller, 1958).

A.2.2. Box-Muller

The Box-Muller approach takes two uniform random variables and returns two independent standard normal variables.

$$\begin{aligned} Z_1 &= \sqrt{-2\ln\xi_1}\cos(2\pi\xi_2) \\ Z_2 &= \sqrt{-2\ln\xi_1}\sin(2\pi\xi_2) \end{aligned} \tag{A.9}$$

A different approach for Gaussian sampling is based on a Monte Carlo method called acceptance-rejection. Marsaglia & Bray (1964) developed this method to avoid using trigonometric functions which can be computationally expensive. However, because most modern computer systems have trigonometric functions in their instruction sets, the Box-Muller equations can be efficiently executed in SUNRAY (Ganssie, 2004).

B. THE SUN

In this appendix, the three options for defining the sun's position are described. Then, the derivation of the pillbox sunshape is provided, as well as the three tracking transformation matrices (the order of transformations) and the formulas for single-axis tracking.

B.1. Sun Position

The simplest way for a user to define the sun's position is to explicitly define the (x, y, z) components of a vector pointing towards the sun, the Sun Vector (SUNRAY converts it to a unit vector). However, a user may not always know the coordinates of the sun's position. Therefore, there are two other methods for determining the sun vector, either by defining the azimuth and altitude angles or by calculating the sun's position from the geographical position of the heliostats and the time of the simulation.

B.1.1. Sun's Azimuth and Altitude Angles

The geometric relationship between a collector at any arbitrary orientation relative to the earth and the sun can be defined in terms of several angles as described in (Duffie & Beckman, 2006). However, defining the sun's position from a coordinate system based at the point of observation can be done with just two angles: the solar altitude angle, α, and the solar azimuth angle, A (Stine & Harrigan, 1986). Figure B.1 illustrates the sign convention for these angles.

$$\begin{aligned} S_x &= -\cos\alpha\cos A \\ S_y &= -\cos\alpha\sin A \\ S_z &= \sin\alpha \end{aligned} \tag{B.1}$$

To maintain consistency with the affine transformational matrices (Appendix C.2), SUNRAY defines the azimuth angle using the right-hand rule: due North is zero and is positive in the counter-clockwise direction. This is in contrast to the conventional clockwise rotation of the cardinal points.

For ray propagation (Section 7.2) a transformation matrix is required, M_{sun}, which tranforms a vector pointing in the z direction (of the world coordinate

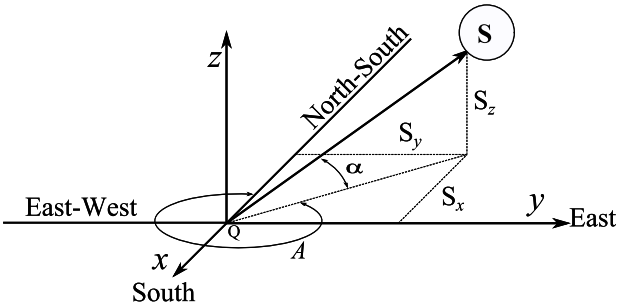


Figure B.1: The earth-surface coordinate system for an observer at Q

system) to be parallel with the Sun Vector. When defining the Sun Vector using azimuth and altitude angles, the transformation matrix is found from

$$M_{sun} = [R_z(A)][R_y(\alpha)] \tag{B.2}$$

Where $R_y(\alpha)$ is rotation about the y-axis by α. Similarly $R_z(A)$ is rotation about the z-axis by A

If a user defines the sun's position by explicitly defining the Sun Vector then SUNRAY obtains the azimuth and altitude angles by rearranging Equation (B.1).

B.1.2. Sun Position Calculator

The third option for determining the sun vector is to calculate it using the input parameters of time and the geographical position of the collector. Several algorithms have been developed which are capable of determining the sun's position. Blanco-muriel, Alarcon-padilla & Lopez (2001) performed a review of the main algorithms developed since the 1960s and also proposed a new algorithm, namely the PSA algorithm.

Sun position algorithms only work for a certain time period, for example 2003-2023 for the Solar Position and Intensity Algorithm 2.0 (SPA) (Grena, 2008), and with various degrees of precision; for example, the Michalsky (1988) algorithm has an error of $\pm 0.01°$.

In SUNRAY only one solar position algorithm has been implemented. The algorithm chosen is the NREL Solar Position Algorithm (SOLPOS) algorithm (NREL, 2000), which has a maximum error of $\pm 0.001°$ and is valid from 1950 to 2050. The SOLPOS algorithm was implemented in SUNRAY by adapting the C source code available on the NREL website (NREL). The SOLPOS algorithm is one of two algorithms available on the NREL website. The second algorithm is the SPA algorithm, which has an uncertainty of $\pm 0.0003°$ between the years -2000 to 6000 (Reda & Andreas, 2008).

The SOLPOS algorithm was chosen for reasons of simplicity. To calculate the position of the sun, the SOLPOS algorithm only takes the parameters of time, time

zone, and the latitudinal and longitudinal position of the collector. In addition to these, the SPA algorithm also takes the parameter ΔT. This is the difference between the earth's rotational time and the terrestrial time and is only derived from observations (Reda & Andreas, 2008).

B.1.3. Pillbox

The following section provides an example of how Monte Carlo methods are used to derive the pillbox sunshape and the next section derives the Bui sun shape. The first step to deriving the pillbox sunshape is to find the solid angle cone with apex angle β, equal to the subtend angle of the pillbox sunshape. This is equal to the area of the spherical cap of the cone (see Figure 4.1)

$$\begin{aligned} A &= \int_0^{2\pi}\int_0^{\beta} \sin\theta d\theta d\phi \\ &= \int_0^{2\pi} \cos\theta \,|_0^{\beta}\, d\phi \\ &= \int_0^{2\pi} (1-\cos\beta) d\phi \quad = 2\pi(1-\cos\beta) \end{aligned} \tag{B.3}$$

The pillbox has uniform sampling within the angle, β, which means that its density function is constant, $p(\omega) = c$. Furthermore, a density function must integrate to one over its domain by using

$$\int_{\mathcal{S}^{\in}} p(\omega) d\omega = 1 \tag{B.4}$$

Thus

$$\begin{aligned} c\int_{\mathcal{S}^{\in}} A d\omega &= 1 \\ c = p(\omega) &= \frac{1}{2\pi(1-\cos\beta)} \end{aligned} \tag{B.5}$$

The density function of a sphere in polar coordinates is found in the same procedure described in (Pharr & Humphreys, 2010). The density function with respect to θ and ϕ is

$$p(\theta,\phi) = \sin\theta p(\omega) \tag{B.6}$$

Next, the marginal density function is required (see Pharr and Humphreys [2010] for a definition of marginal density function). This is found by integrating out ϕ

$$\begin{aligned} p(\theta) = \int_0^{2\pi} p(\theta,\phi)d\phi &= \int_0^{2\pi} \frac{\sin\theta}{2\pi(u)}d\phi \qquad\qquad u = 1-\cos\beta \\ &= \frac{\sin\theta}{2\pi(u)}\phi\,|_0^{2\pi} \\ &= \frac{\sin\theta}{u} \end{aligned} \tag{B.7}$$

The conditional density ϕ is thus

$$p(\phi|\theta) = \frac{p(\theta,\phi)}{p\theta} = \frac{1}{2\pi} \tag{B.8}$$

Using 1D inversion to sample each pdf

$$\begin{aligned} p(\theta) &= \int_0^{\alpha} \frac{\sin\theta}{u}d\theta \\ &= \frac{1}{u}(1-\cos\theta) \end{aligned} \tag{B.9}$$

And

$$\begin{aligned} p(\phi|\theta) &= \int_0^{\phi} \frac{1}{2\pi}d\phi \\ &= \frac{\phi}{2\pi} \end{aligned} \tag{B.10}$$

And, finally, inverting Equation (B.9) and (B.10)

$$\begin{aligned} &\frac{1}{u}(1-\cos\theta) = \xi_1 \\ &\theta = \arccos(1-\xi_1(u)) \end{aligned} \tag{B.11}$$

and

$$\begin{aligned} &\frac{\phi}{2\pi} = \xi_2 \\ &\phi = 2\pi\xi_2 \end{aligned} \tag{B.12}$$

B.1.4. User-Defined

A user-defined profile is defined for N values of radial displacement of equal size, $\Delta = 1/N$. The region of intensity starts and ends at points $x_i = i\Delta$, where $i \in [0,1]$. (SUNRAY scales the distribution to fit between the domain $[0,1]$, as shown in Figure B.2).

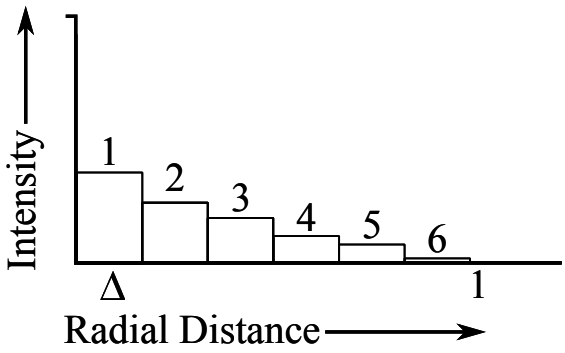


Figure B.2: Probability density function for a 1D discrete user-defined profile

Within each region, the user-defined value function $f(x)$ (intesity) is a constant.

$$f(x) = \begin{cases} \upsilon_0 & x_0 \leq x \leq x_1 \\ \upsilon_0 & x_1 \leq x \leq x_2 \\ \vdots \end{cases} \tag{B.13}$$

The integral of $f(x)$ is the summation of the areas formed by each step

$$c = \int_0^1 f(x)dx = \sum_{i=0}^{N-1} \Delta\upsilon_i = \sum_{i=0}^{N-1} \frac{\upsilon_i}{N} \tag{B.14}$$

The pdf of $f(x)$ is $f(x)/c$. The cdf $P(x)$ is a defined at points x_i by

$$\begin{aligned} P(x_0)) &= 0 \\ P(x_1) &= \int_{x_0}^{x_i} p(x)dx = \frac{\upsilon_0}{Nc} = P(x_0) + \frac{\upsilon_0}{Nc} \\ &\vdots \\ P(x_i) &= P(x_{i-1}) + \frac{\upsilon_{i-1}}{Nc} \end{aligned} \tag{B.15}$$

In order to sample the function, $f(x)$, the cdf needs to be inverted, as shown in Figure B.3, such that

$$\begin{aligned} \xi &= \int_0^x p(x')dx' \\ &= P(x) \end{aligned} \tag{B.16}$$

Finally, because the cdf is monotonically increasing, the value for x must be between x_i and x_{i+1}, such that $P(x_i) \leq \xi$ and $\xi \leq P(x_{i+1})$. Thus, with an array of cdf values (calculated in Equation (B.15)), the pair of $P(x_i)$ values can be found efficiently with a binary search.

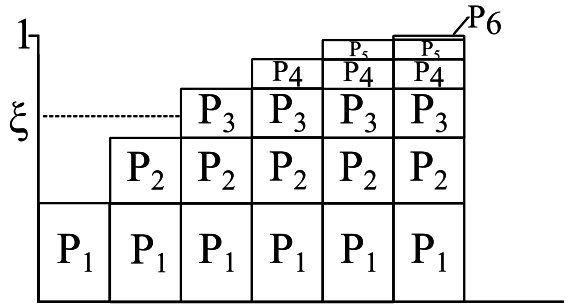


Figure B.3: Inversion of the user-defined profile

B.1.5. Buie

Buie *et al.* (2003*b*) show that the terrestrial solar flux distribution can be simulated using

$$\phi(\theta)) = \begin{cases} \dfrac{\cos(0.326\theta)}{\cos(.308\theta)} & \text{for } \theta \in \Re | 0 \leq \theta \leq 4.65\ mrad \\ e^{\kappa}\theta^{\gamma} & \text{for } \theta \in \Re | \theta > 4.65 \text{ mrad} \end{cases} \tag{B.17}$$

Where γ and κ are defined in terms of the CSR $\chi = \Phi_{cs}/\Phi_i$.

$$\gamma = 2.2\ln(0.52\chi)\chi^{0.43} - 0.1 \tag{B.18}$$

$$\kappa = 0.9\ln(13.5\chi)\chi^{-0.3} \tag{B.19}$$

This equation is used in the User-Defined function in order to generate random samples according to the Buie sunshape profile.

B.2. Tracking Transformations

For collectors to reflect light onto the target, they have to be correctly aligned. Therefore, the order for transformation is extremely important. The transformation matrices for the two dual-axis tracking methods have been derived for SUNRAY and are provided below. This section also discusses single-axis tracking.

B.2.1. AE Transformation

For AE tracking only two angular movements of the heliostat are required. First, the object is rotated about the y-axis by γ_{AE} and then about the z-axis by ρ_{AE}, see Figure 4.2. The transformation matrix M_{AE} is thus

$$M_{AE} = [T][R_z(\rho_{AE})][R_y(\gamma_{AE})][M\prime] \tag{B.20}$$

Where $[M\prime]$ is the transformation matrix due to any scaling and/or shearing and T is the translation matrix.

B.2.2. SE Transformation

The coordinate transformation for SE tracking is more complex than for AE tracking as there are two additional orientation transformations required to ensure that the heliostat faces the target.

The order of transformation is first the elevation movement of γ_{SE} about the y-axis, then spinning of ρ_{SE} about the x-axis in the negative direction, followed by rotation of λ_{SE} about the y-axis and, finally, rotation about the z-axis by ϕ, as illustrated in Figure 4.4. The complete transformation matrix is given as

$$M_{AE} = [T][R_z(\phi_{SE})][R_y(\lambda_{SE})][-R_x(\rho_{SE})][R_y(\gamma_{SE})][M\prime] \tag{B.21}$$

B.2.3. Single-Axis Tracking

A single-axis collector rotates about an axis of rotation, r, until the sun vector and the aperture normal are coplanar. If r is arbitrarily orientated and not collinear with an axis of the world coordinate system, then SUNRAY needs to perform a series of coordinate transformations.

The steps for the derivation of single-axis tracking in SUNRAY are similar to those described in (Stine & Harrigan, 1986). The single-axis tracking angle, ρ_{SA}, about r is

$$\rho_{SA} = \arctan\left(\frac{\cos\alpha\sin(\phi_{SA} - A)}{\sin(\alpha - \lambda_{SA}) + \sin\lambda_{SA}\cos\alpha[1 - \cos A - \phi_{SA}]}\right) \tag{B.22}$$

Where ϕ_{SA} is the horizontal (x-y-plane) rotation of the collector and λ_{SA} is the tilt angle. The complete transformation matrix is

$$M_{SA} = [T][R_z(\lambda_{SA})][R_x(\phi_{SA})][M\prime] \tag{B.23}$$

For the single-axis concentrators SUNRAY allows the rotation axis of the collector axis to be rotated about any axis and at any orientation. However, in practice the rotation axis is usually aligned with a horizontal cardinal direction (Stine & Harrigan, 1986).

C. GEOMETRIC OBJECTS

It is beyond the scope of this thesis to include the construction, intersection tests, and bounding routines for all objects in SUNRAY. For the quadratics, such as the cone, cylinder, trough, and paraboloid the intersection and bounding routines are similar to the sphere as described in Chapter 5. The parametric and implicit equations for these shapes can be found in ray tracing books such as (Heckbert, 1994) or (John, 2005). Similarly, for objects such as discs, rectangles, and boxes, the intersection and bounding routines have also been derived with the aid of various ray tracing literature.

In SUNRAY objects such as the cylindrical receiver and the multifaceted heliostat were constructed as proof-of-concept of compound objects. It should be noted that these shapes are only indicative and do not represent any real receiver or heliostat design.

More complex shapes, such as the focused heliostats used in the PS10 simulation (Section 7.4) or a toroidial facet, were created using the same intersection routines of other objects, except they were truncated to form the required shape. The focused heliostat was created by 'cutting' a rectangle out of a parabolic dish. The Torodial Facet is similar to a Focused Heliostat, except with the constraint that it has uneven scaling in one direction, thus forming two axes of curvature. These are the sagittal and tangential planes for target-aligned (SE Tracking) heliostats.

This appendix discusses one important shape, the triangle, which has a different intersection routine than other shapes. In this appendix, affine transformations are also discussed.

C.1. Triangles

In Section 5.2, triangular meshes and PLY files are discussed. This section discusses the construction, bounding, and intersection of triangles.

C.1.1. Triangle Construction and Bounding

A triangle is defined by three points, $\mathbf{a}, \mathbf{b}$, and $\mathbf{c}$. To construct a triangle in SUNRAY the user simply defines the (x, y, z) coordinates of these corners. By convention the points are defined in a counter-clockwise order. The lower and

upper coordinates of the bounding box are the minimum and maximum values of the corner coordinates of the triangle, respectively.

C.1.2. Triangle Intersection

If the corners of the triangle are not collinear[1] they also define a plane. A common way to describe a plane is to use barycentric coordinates (John, 2005).

$$\mathbf{p}(\alpha, \beta, \gamma) = \alpha\mathbf{a} + \beta\mathbf{b} + \gamma\mathbf{c} \tag{C.1}$$

constrained with

$$\alpha + \beta + \gamma = 1 \tag{C.2}$$

For a point, $\mathbf{p}$, inside the triangle, the coordinates (α, β, γ) satisfy the inequalities

$$\begin{aligned} 0 < \alpha < 1 \\ 0 < \beta < 1 \\ 0 < \gamma < 1 \end{aligned} \tag{C.3}$$

By substituting $\alpha = 1 - \beta - \gamma$ into Equation (C.1), one variable can be eliminated. Thus, a triangle is parameterised by two variables β andγ.

$$\mathbf{p}(\beta, \gamma) = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}) \tag{C.4}$$

The inequalities in Equation (C.3) then become

$$0 < \beta < 1 \tag{C.5a}$$

$$0 < \gamma < 1 \tag{C.5b}$$

$$0 < \beta + \gamma < 1 \tag{C.5c}$$

For example when $\beta = 0$ Equation (C.4) becomes

$$\mathbf{p} = \mathbf{a} + \gamma(\mathbf{c} - \mathbf{a}) \tag{C.6}$$

This is a straight line through $\mathbf{a}$ and $\mathbf{b}$ which defines the γ-axis in Figure C.1. Then, with the inequality $0 < \beta + \gamma < 1$, Equation (C.6) defines the edge $\mathbf{c}$-$\mathbf{a}$.

To test whether a ray hits a triangle, the equation of a ray, Equation (3.2), is substituted into Equation (C.4).

$$\mathbf{o} + t\mathbf{d} = \mathbf{a} + \beta(\mathbf{b} - \mathbf{a}) + \gamma(\mathbf{c} - \mathbf{a}) \tag{C.7}$$

[1] SUNRAY tests if the points are collinear and issues a warning if they are not

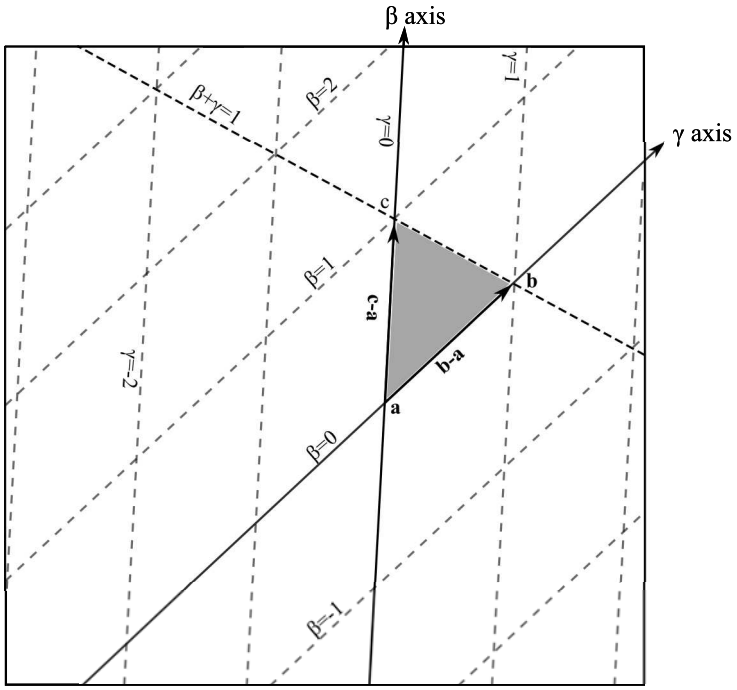


Figure C.1: A triangle with origin at **a** within the (β,γ) coordinate system

There are three unknowns in Equation (C.7). To solve for them, Equation (C.7) is broken up into its vector form and rewritten as a standard linear equation

$$\begin{bmatrix} a_x - b_x & a_x - c_x & d_x \\ a_y - b_y & a_y - c_y & d_y \\ a_z - b_z & a_z - c_z & d_z \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} a_x - o_x \\ a_y - o_y \\ a_z - o_z \end{bmatrix} \tag{C.8}$$

Using dummy variables for simplicity, Equation (C.8) can be represented by

$$\begin{bmatrix} a & b & c \\ e & f & g \\ i & j & k \end{bmatrix} \begin{bmatrix} \beta \\ \gamma \\ t \end{bmatrix} = \begin{bmatrix} d \\ h \\ l \end{bmatrix} \tag{C.9}$$

The solution to Equation (C.9) is found using Cramer's Rule described in (Butt, 2009)

$$\beta = \frac{d(fk - gj) + b(gl - hk) + c(hj - fl)}{a(fk - gj) + b(gi - ek) + c(ej - fi)} \tag{C.10a}$$

$$\gamma = \frac{a(hk - gl) + d(gi - ek) + c(el - hi)}{a(fk - gj) + b(gi - ek) + c(ej - fi)} \tag{C.10b}$$

$$t = \frac{a(fl - hj) + b(hi - el) + d(ej - fi)}{a(fk - gj) + b(gi - ek) + c(ej - fi)} \tag{C.10c}$$

For computational efficiency early-out tests are programmed into all objects. Early-outs stop a ray-object intersection function if the ray has missed an object. For example, after β is calculated in Equation (C.10a), it is tested using the equality in Equation (C.5a). If $\beta < 0$, the triangle hit function immediately exists, ensuring no unnecessary calculations.

C.2. Affine Transformations

Affine transformations preserve straight lines and ratios of points lying on straight lines (Shirley *&* Marschner, 2009). The theory behind affine transformations is covered in books on linear algebra, for example (Anton *&* Rorres, 2008). This section only covers details relevant to SUNRAY.

C.2.1. Transformations

All objects can be explicitly sized and placed anywhere within the world coordinate system. However, it is recommended, for simplicity and computational speed, to use instancing (Section 5.4). When a scene is built all instance objects are placed at the origin of the world coordinate system. The user can then transform the object by scaling, $S_{x,y,z}$; rotating, $R_{x,y,z}(\theta)$; shearing, $Sh_{x,y,z}$; and translating, $T_{x,y,z}$ the object. For rotation, the axis of rotation is indicated by the subscript and the positive direction is taken as counter-clockwise about the axis (right-hand rule).

For ease of use, SUNRAY was written so that all transformations occur within the world coordinate system. This, however, introduces an unavoidable complexity, which is maintaining the correct order of the transformations.

C.2.2. Order of Transformations

An example of the ordering of transformations is illustrated in Figure C.2.

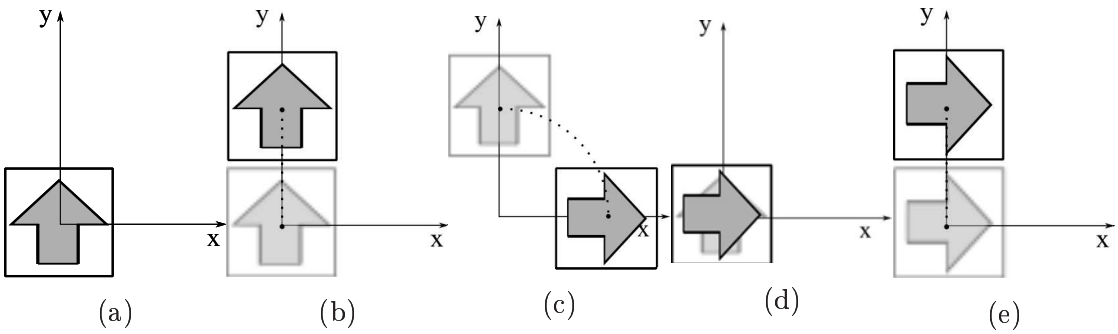


Figure C.2: Ordering of transformations

The object in Figure C.2a undergoes one rotation and one translation. In the first scenario, the object is first translated along the y-axis, Figure C.2b, and then rotated about the z-axis, Figure C.2c. In the second scenario the transformations are applied in the opposite order, that is, rotation about the z-axis, Figure C.2d, and translation along the y-axis, Figure C.2e. It can be seen from the figure that a completely different result is obtained. This is due to the fact that generally transformations do not commutate, that is $M_1 M_2 \neq M_2 M_1$ (John, 2005). It is important that the user is aware of this when placing objects in a simulation.

It is suggested that the general procedure when placing objects in a scene is to first perform any scaling, shearing, and rotation to the object before translating it. Following this order will ensure that the objects are always transformed correctly. The tracking formula in Section 4.2 have been derived to compensate for the fact that transformations do not commute.

D. RADIOMETRY

This appendix focuses on the mathematical description of the propagation and reflection of radiation throughout the electromagnetic spectrum, known as radiometry (Modest, 2003). The following radiometric theory applies to electromagnetic radiation across the entire wavelength, but for brevity all concepts are discussed with regard to light.

D.1. Spectral Distribution of Light

The amount of energy, q, carried by one photon with a wavelength, λ, is given by $q = 6.63 \times 10^{-34} c/\lambda$. For a large collection of photons grouped into an interval $\Delta\lambda$, the spectral power, Q_λ, is computed by summing the energy of each photon (Modest, 2003). The Spectral Power Distribution (SPD) for the extraterrestrial solar radiation is given in Figure D.1.

From the viewpoint of terrestrial applications of solar energy, Duffie & Beckman (2006) state that only the radiation of wavelengths between 290 nm and 2500 nm need to be considered.

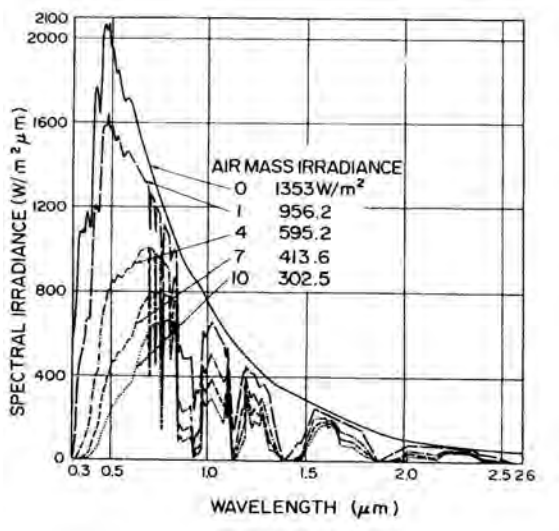


Figure D.1: Solar Spectral irradiance for different air mass values (Thekaekara, 1976)

All real-world objects also have their own SPDs. Figure D.2 shows the spectral distribution of reflectance off the Alanod MIRO-SUN surface. Representing a SPD, like in Figure D.2, is often done with a basis function. Basis functions map the infinite-dimensional space of SPD functions to a low-dimensional space of coefficients, $c_i \in \mathrm{R}$ (Pharr & Humphreys, 2010). The value of c_i could range from 1, representing the average value of the entire SPD, to intervals in the order of nano-meters (or smaller).

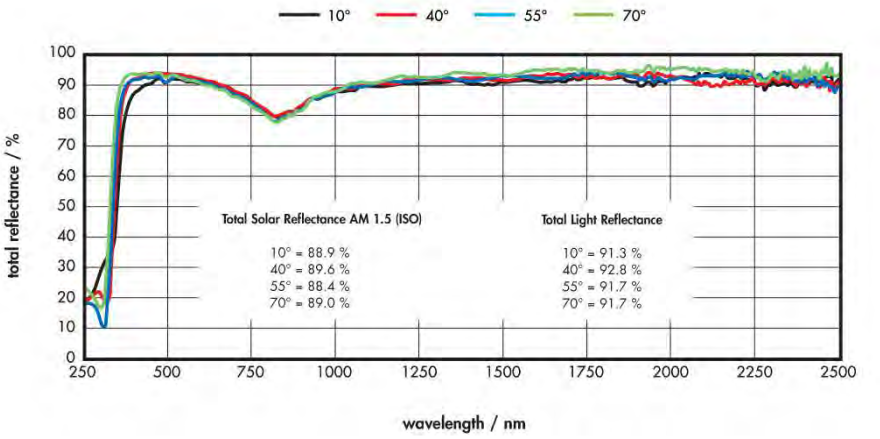


Figure D.2: Total spectral reflectance of MIRO-SUN ® (Alanod Solar, 2012)

For many solar ray tracers, such as in SolTrace and Tonatiuh, sunlight is modelled with a single coefficient. As SUNRAY is to be used in a research institute, it is possible that a user might require higher order spectral representation. Therefore, SUNRAY has been written so that any basis function can be included. Currently, light is represented by a triplet of floating point numbers. This represents the traditional red, green, and blue spectrum, $c_3 = (R, G, B)$, but can equally be considered as the ultraviolet, visible, and infrared spectrum.

D.2. Radiometric Quantities

There are four quantities that are central to radiometry: radiant flux, irradiance/radiant exitance, radiant intensity, and radiance.

Radiant Flux (Φ): Also known as power, radiant flux is the amount of energy that passes through a surface or region of space per second. It is measured in W (Modest, 2003).

Irradiance (E) and Radiant Exitance (M): Radiant flux density is the radiant flux per unit surface area measured in $\mathrm{W.m^{-2}}$ (Modest, 2003)

$$E = d\Phi/dA \tag{D.1}$$

Irradiance is the flux density of flux arriving at a surface and radiant exitance flux density leaving a surface (Siegel & Howell, 1992).

Radiant Intensity (I): Radiant intensity is the flux density per solid angle and has the units W.m^{-2}.sr^{-1} (Modest, 2003).

Radiance (L): Radiance, an important radiometric property, is the flux density per unit area, per unit solid angle (Suffern, 2007)

$$L = \frac{d\Phi}{d\boldsymbol{\omega} dA^{\perp}} \tag{D.2}$$

Where $dA^{\perp}$ is the projected area of dA on a hypothetical surface perpendicular to the direction of the (normalised) vector $\boldsymbol{\omega}$. Radiance measures the radiant flux from photons travelling down the cone of incident's direction of interest, $d\boldsymbol{\omega}$, as illustrated in Figure D.3. In the limit, where both $d\boldsymbol{\omega}$ and $dA^{\perp}$ approach zero, the photons are confined to a ray in a single direction.

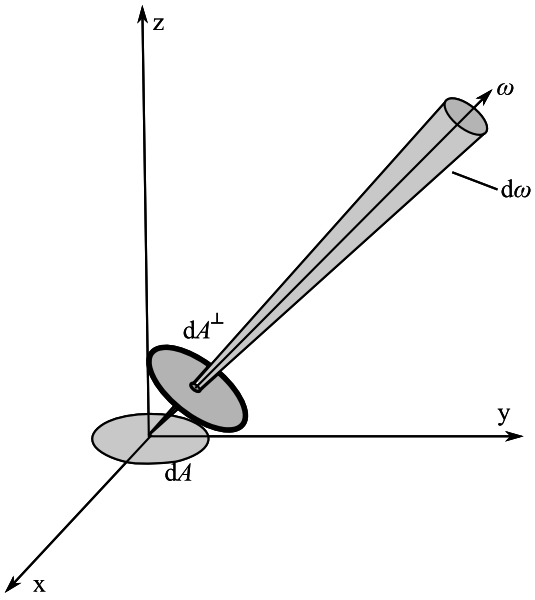


Figure D.3: Radiant flux L in a cone of incident angles $d\boldsymbol{\omega}$ passing through a surface element $dA^{\perp}$

The radiance hitting a surface is proportional to the cosine angle between the light direction and the surface normal, $dA^{\perp} = \cos\theta dA$. This in known as Lamberts' Law (Pharr & Humphreys, 2010). In solar applications this *cosine effect* is extremely important in optimising the orientation of solar collectors (Stine & Harrigan, 1986).

It is possible to describe all radiometric properties by radiance in terms of integrals over areas and directions (Suffern, 2007). For example, the incident

radiance is related to irradiance (represented by subscript i) at point $\mathbf{p}$, through Equation (D.1) and Equation (D.2).

$$dE_i(\mathbf{p}, \boldsymbol{\omega}_i) = L_i(\mathbf{p}, \boldsymbol{\omega}_i) \cos\theta_i d\omega_i \tag{D.3}$$

Therefore, the total irradiance at $\mathbf{p}$ can be found by integrating over the solid angle $\Omega_i = 2\pi^+$

$$E(\mathbf{p}) = \int_{\Omega_i} L_i(\mathbf{p}, \boldsymbol{\omega}_i) \cos\theta_i d\omega_i \tag{D.4}$$

A similar expression can be found for reflected radiance $L_o(\mathbf{p}, \boldsymbol{\omega_o})$. The notation in the SUNRAY is to define the vector which points away from $\mathbf{p}$, as illustrated in Figure D.4.

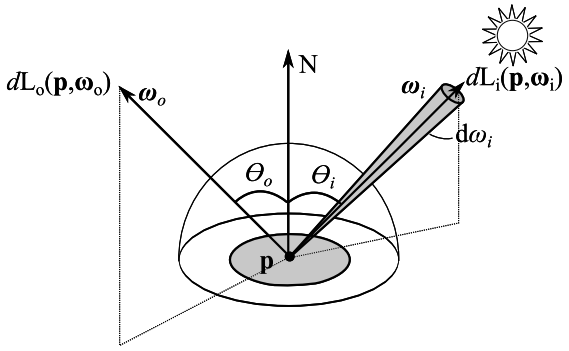


Figure D.4: Irradiance at point $\mathbf{p}$ from incident direction i

The spectral radiance at $\mathbf{p}$ on a surface can be found by integrating over wavelength λ (Siegel & Howell, 1992).

$$L(\mathbf{p}, \boldsymbol{\omega}_i) = \int_0^\infty L_\lambda(\boldsymbol{p}, \boldsymbol{\omega}_i, \lambda) d\lambda \tag{D.5}$$

From Section D.1, the radiance at a point is represented by three wavelengths of light. The reflection of light off a surface is defined by its BRDF.

D.3. Bidirectional Reflectance Distribution Function

In general, for light impinging on a surface of finite thickness, some of the irradiance will be absorbed by the medium, some will be transmitted through the medium and the rest will be reflected or scattered back into the environment (Modest, 2003). The reflective properties of the material are precisely described by the

surface BRDF (Nicodemus *et al.*, 1977). In Figure D.4, the differential amount of radiance, $dL_o(\mathbf{p}, \boldsymbol{\omega}_o)$, leaving a surface in direction $\boldsymbol{\omega}_o$ is a result of the irradiance, $dL_i(\mathbf{p}, \boldsymbol{\omega}_i)$, along differential cones of directions, $\boldsymbol{\omega}_i$. From the linearity assumption of geometric optics (Goodman, 2004), the differential reflected radiance is proportional to the irradiance

$$dL_o(\mathbf{p}, \boldsymbol{\omega}) \propto dE(\mathbf{p}, \boldsymbol{\omega}_i) \tag{D.6}$$

The BRDFs is simply the constant of proportionality defined for particular directions $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$

$$f_r(\mathbf{p}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{dL_o(\mathbf{p}, \boldsymbol{\omega}_o)}{dE(\mathbf{p}, \boldsymbol{\omega}_i)} = \frac{dL_o(\mathbf{p}, \boldsymbol{\omega}_o)}{L_i(\mathbf{p}, \omega_i) \cos\theta_i d\omega_i} \tag{D.7}$$

Physically based BRDFs have three important properties for ray tracing (Dutré, 2003)

1. **Reciprocity**: The value for the BRDF remains the same in both directions $f_r(\mathbf{p}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = f_r(\mathbf{p}, \boldsymbol{\omega}_o, \boldsymbol{\omega}_i)$.
2. **Energy Conservation**: The total energy of reflected light is less than or equal to the incident light for all $\boldsymbol{\omega}_o$.
3. **Linearity**: Materials can be modelled using multiple BRDFs. The total reflected radiance at a surface point is the sum of the reflected radiance from each BRDF. An example of this is a surface with both specular and diffuse properties.

D.4. Reflectance

Reflectance, ρ, is defined as the ratio of reflected flux to incident flux. It reduces the 4D BRDF function (if wavelength is taken into account) to a 2D function over a single direction or even a constant value which describes the overall scattering behaviour (Pharr & Humphreys, 2010). The directional-hemispherical reflectance is defined as the energy reflected into all solid angle from one direction.

$$\rho_{dh} = \int_{\Omega_i} f_r(\mathbf{p}), \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \cos\theta_i d\omega_i \tag{D.8}$$

For example, for the theoretical ideal diffuse surface, called a Lambertian Surface, light is scattered equally in all directions (Siegel & Howell, 1992). Thus, the reflected radiance is independent of $\boldsymbol{\omega}_o$. This is only possible if BDRF is independent of $\boldsymbol{\omega}_o$ and $\boldsymbol{\omega}_i$. In this case, Equation (D.8) simplifies to

$$\rho_d(\mathbf{p}) = f_r(\mathbf{p}) \int_{\boldsymbol{\Omega}_i} \cos\theta_i d\omega_i = f_r(\mathbf{p})\pi \tag{D.9}$$

Thus, the BRDF for a perfectly diffused reflector can be defined as

$$f_{r,d} = \rho_d(\mathbf{p})/\pi \tag{D.10}$$

D.5. Light Transport Equation

The light transport equation is the governing energy equation in the SUNRAY. The principal behind the LTE and thus the SUNRAY, is energy balance (Section 6.3.3). The LTE is derived from the rendering equation, Equation (6.10).

D.5.1. Rendering Equation

From Equation (6.10), the reflected radiance at a point is a function of irradiance, which, in turn, could have been reflected from another surface. From the conservation of energy assumption, the radiance is constant along each ray (attenuation due to participating media is included in Section D.6). Thus, the incident radiance at point $\mathbf{p}$ is equal to the exitance radiance at $\mathbf{p}_1$, where $\mathbf{p}_1$ is the nearest hit point along the ray in direction $\boldsymbol{\omega}_i$, as shown in Figure D.5.

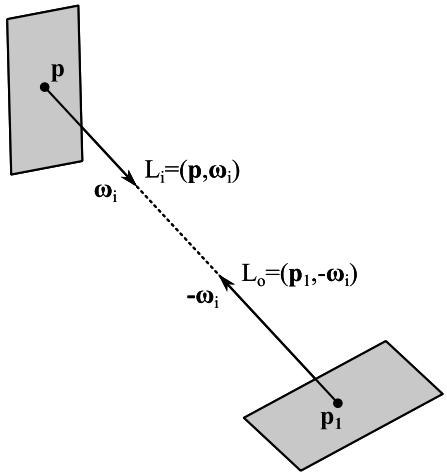


Figure D.5: The incident radiance at $\mathbf{p}$ is equal to the exitant radiance at $\mathbf{p}_1$

An alternative formation of the rendering equation, derived in (Suffern, 2007), is the area form

$$L_o(\mathbf{p}, \boldsymbol{\omega}_o) = \int_A L_o(\mathbf{p}_1, -\boldsymbol{\omega}_i) f_r(\mathbf{p}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) G(\mathbf{p}, \mathbf{p}_1) dA \tag{D.11}$$

With

$$G(\mathbf{p}, \mathbf{p}_1) = \cos\theta_i \cos\theta_1 / ||\mathbf{p}_1 - \mathbf{p}|| \tag{D.12}$$

Furthermore, it is possible that the irradiance at point $\mathbf{p}_1$ is due to the radiance reflected from another surface point, $\mathbf{p}_2$. Thus, Equation (D.11) is repeatedly called for multiple reflections.

D.5.2. Light Transport Equation Over Path

If a ray has multiple reflections through a scene, the rendering equation is called recursively. Using the notation illustrated in Figure D.6, the light transport equation for multiple paths is given in (Pharr & Humphreys, 2010).

$$\begin{aligned} L(\mathbf{p}_2 \to \mathbf{p}_3) &= L_e(\mathbf{p}_2 \to \mathbf{p}_3) \\ &\quad + \int_A L_e(\mathbf{p}_1 \to \mathbf{p}_2) f_r(\mathbf{p}_1 \to \mathbf{p}_2 \to \mathbf{p}_3) G(\mathbf{p}_1 \leftrightarrow \mathbf{p}_2) dA(\mathbf{p}_1) \\ + \int_A \int_A L_e(\mathbf{p}_0 \to \mathbf{p}_1) f_r(\mathbf{p}_0 \to \mathbf{p}_1 \to \mathbf{p}_2) G(\mathbf{p}_0 \leftrightarrow \mathbf{p}_1) & \\ \times f_r(\mathbf{p}_1 \to \mathbf{p}_2 \to \mathbf{p}_3) G(\mathbf{p}_1 \leftrightarrow \mathbf{p}_2) dA(\mathbf{p}_0) dA(\mathbf{p}_1 + \dots & \end{aligned} \tag{D.13}$$

Here $L_e(\mathbf{p}, \boldsymbol{\omega}_o)$ is radiance emitted from the sun. This can be represented by the sum over the paths

$$L(\mathbf{p}_0 \to \mathbf{p}_3) = \sum_{n=1}^{\infty} P(\bar{p}_n) \tag{D.14}$$

Where $P(\bar{p})$ is the radiance scattered over a path $\bar{p}_n$ with $n+1$ vertices. $\bar{p} = , \mathbf{p}_1 \dots, \mathbf{p}_n$. Where $\mathbf{p}_0$ is on the View Plane and $\mathbf{p}_n$ is the target.

$$\begin{aligned} P(\bar{p}) = & \underbrace{\int_A \int_A \dots \int_A}_{n-1} L_e(\mathbf{p}_n \to \mathbf{p}_{n-1}) \\ & \times \left(\prod_{i=1}^{n-1} f_r(\mathbf{p}_{i+1} \to \mathbf{p}_i \to \mathbf{p}_{i-1} G(\mathbf{p}_{i+1} \leftrightarrow \mathbf{p}_i) \right) dA(p_2) dA \dots dA(p_n) \end{aligned} \tag{D.15}$$

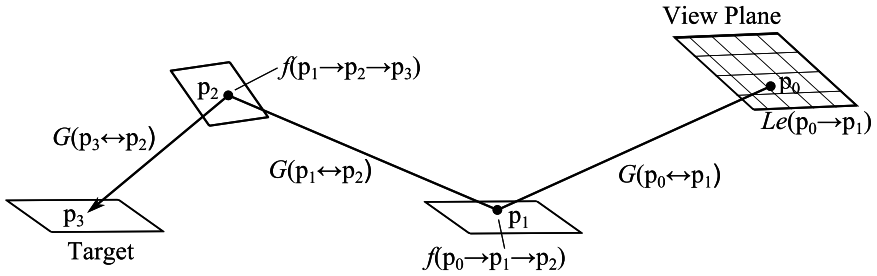


Figure D.6: Path tracing

To account for multiple BRDF, Equation (D.15) can be partitioned. For example, $f_{r\Delta}$ could be the perfect specular component and $f_{r\Delta'}$ could be a diffuse component.

$$\begin{aligned} P(\bar{p}) = & \int_{A^{n-1}} L_e(\mathbf{p}_n \rightarrow \mathbf{p}_{n-1}) \\ & \times \prod_{i=1}^{n-1} (f_{r\Delta}(\mathbf{p}_{i+1} \rightarrow \mathbf{p}_i \rightarrow \mathbf{p}_{i-1} + f_{r\Delta'}(\mathbf{p}_{i+1} \rightarrow \mathbf{p}_i \rightarrow \mathbf{p}_{i-1}) \\ & \times G(\mathbf{p}_{i+1} \leftrightarrow \mathbf{p}_i) dA(p_2) dA \ldots dA(p_n) \end{aligned} \tag{D.16}$$

Equation (D.16) is a very high-dimensional integral which involves complicated integrands and complex reflection functions. It is these complexities that make Monte Carlo methods the ideal choice for solving rendering problems (Hanraham, 2003).

Implementation of Equation (D.15) in SUNRAY is performed through a main ray tracing loop. The object-oriented design of SUNRAY ensures that different classes, such as the BRDF class, can be appropriately called to determine the flow of energy through a scene.

There are some theoretical challenges with tracing the path of a ray. From Equation (D.15), a scattered reflected ray can take on an infinite number of paths. Thus, one challenge is to estimate the value of an infinite sum. To prevent a ray from being reflected an infinite number of times, SUNRAY extinguishes the ray after the flux carried by the ray has dropped below a certain predefined percentage. This, however, introduces a small bias (consistent error) into the simulation. Fortunately, in most CSP simulations the number of secondary reflections is usually small.

A second challenge is to generate low-variance random path directions (Shirley *&* Morley, 2003). The method by which a direction is chosen is called importance sampling (Kroese *et al.*, 2011). The approach employed by SUNRAY is to construct incremental paths by sampling a new direction at each hit point. Monte Carlo procedures provide efficient ways of generating directions from a given multi-dimensional distribution, for example the Imperfect Specular class uses Equation (6.3) to generate random paths using Box-Muller sampling (Appendix A.2.2).

D.6. Atmospheric Attenuation

The loss associated with atmospheric attenuation between the heliostat field and the receiver is a significant source of energy loss and can reach over 10% in central

receiver plants (Sengupta & Wagner, 2012). A discussion on the effect of atmospheric attenuation, as well as a discussion on various models used to describe atmospheric attenuation in a CSP system, is provided by Ballestrín & Marzo (2012). Three models have been incorporated into SUNRAY to simulate atmospheric attenuation, namely Vittitoe-Biggs, MIRVAL, and NREL.

On a very clear day, the energy loss per kilometre between a heliostat and a tower can be small. However, the loss increases considerably as the content of aerosols and water vapour in the air increase. These air particles cause two main processes which affect the attenuation of a ray: absorption and scattering (Siegel & Howell, 1992). The amount of absorption or scattering is usually expressed in terms of the absorption cross-section, σ_a, and scattering cross-section, σ_s, respectively. The cross-section is the probability density function that light (or a photon of light) will interact with particles per unit distance travelled (Modest, 2003). The cross-section may vary with both position, $\mathbf{p}$, and direction, $\boldsymbol{\omega}$.

The total attenuation or extinction of radiance is due to a combination of both scattering and absorption. The extinction cross-section is thus

$$\sigma_e = \sigma_a + \sigma_s \tag{D.17}$$

The effect of attenuation along a differential portion of a ray is illustrated in Figure D.7. For an amount of radiance, $L_i(\mathbf{p}, -\boldsymbol{\omega})$, arriving at $\mathbf{p}$, it is required to compute the exit radiance, $L_o(\mathbf{p}, \boldsymbol{\omega})$, after the attenuation in the differential volume. The change in radiance along a portion of a ray, t, is described by

$$L_o(\mathbf{p}, \boldsymbol{\omega}) - L_i(\mathbf{p}, -\boldsymbol{\omega}) = dL_o(\mathbf{p}, \boldsymbol{\omega}) = -\sigma_e(\mathbf{p}, \boldsymbol{\omega}) L_i(\mathbf{p}, -\boldsymbol{\omega}) dt \tag{D.18}$$

The differential equation describing the attenuation is thus

$$\frac{dL_o(\mathbf{p}, \boldsymbol{\omega})}{dt} = -\sigma_e(\mathbf{p}, \boldsymbol{\omega}) L_i(\mathbf{p}, -\boldsymbol{\omega}) \tag{D.19}$$

Solving Equation (D.19) gives expression for the ray transmittance, τ_a, between two points separated by distance d.

$$\tau_a = \exp(-\int_0^d \sigma_e(\mathbf{p} + t\boldsymbol{\omega}, \boldsymbol{\omega}) dt) \tag{D.20}$$

Thus, if the exitant radiance from a point on a collector surface has a reflected direction, $\boldsymbol{\omega}$, and radiance, $L_o(\mathbf{p}, \boldsymbol{\omega})$, the incident radiance at the next hit point (for example the receiver) is $\tau_a L_o(p, \boldsymbol{\omega})$. The three attenuation models in SUNRAY were developed to model τ_a in central receiver systems.

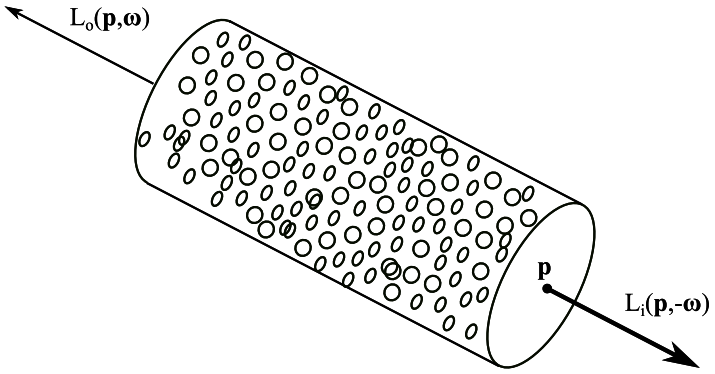


Figure D.7: Absorption of a ray through participating media

Vittitoe *&* Biggs (1978) modelled atmospheric transmittance on a clear day (visibility 25 km) and for a hazy day (visibility 5 km). The attenuation loss for a clear day is given as

$$\begin{aligned} Loss(\%) &= 100(1-\tau_a) \\ &= 0.6739 + 10.46S - 1.70S^2 + 0.2845S^3 \end{aligned} \tag{D.21a}$$

and for a hazy day

$$Loss(\%) = 1.293 + 27.48S - 3.394S^2 \tag{D.21b}$$

Where S is the slant range in kilometres between the first and second hit point. These expressions are used in the DELSOL codes (Falcone, 1986), see Section 2.

An alternative model, used in the MIRVAL code (Leary *&* Hankins, 1979), models attenuation on a clear day and is given as

$$\begin{aligned} Loss(\%) &= 0.679 + 0.0117610^3 S - 1.97S^2 && (S \leq 1km) \\ Loss(\%) &= 100(1 - e^{-0.1106S}) && (S > 1km) \end{aligned} \tag{D.22}$$

The third model, developed by Hanraham (2003), only requires knowledge of the DNI at surface level and is given as

$$DNI_2 = DNI_1 \exp \frac{-Y \times d}{250} \tag{D.23}$$

Where DNI_2 is the DNI at the second hit point. DNI_1 is the DNI measured at the same level as the collectors and Y is the optical depth of the bottom 250 m atmospheric layer.

Implementation of attenuation in SUNRAY is simplified by the fact that the hit point of each ray is recorded (primarily for post processing). The distance between two consecutive hit points is calculated and the flux carried by the ray is adjusted according to the attenuation model selected by the user using Equations (D.21)-(D.23).

E. ACCELERATION TECHNIQUES

Ray-object intersections can account for the bulk of the computational resources of a ray tracer. Therefore, substantial research has been devoted to acceleration techniques for ray tracers. Modern computing hardware has further reduced computational time by means of parallelization. However, hardware acceleration methods are not addressed here. Instead, this appendix discusses the algorithms and routines which can reduce SUNRAY's simulation time.

This appendix begins with a brief overview of the various acceleration techniques available. Three of these techniques are then discussed in more detail.

E.1. Overview of Acceleration Techniques

An extensive survey by Arvo and Kirk in (Glassner, 1989) reviews the most common acceleration techniques for graphical ray traces. Broadly speaking, acceleration techniques can be divided into three categories:

1. Those reducing the computational cost of intersecting rays;
2. Those reducing the total number of rays to be traced; and
3. Those replacing individual rays with a more general entity.

Two of the four acceleration techniques used in SUNRAY fall within category 1, that is, they simplify ray-object interactions. For this, bounding boxes have been used as well as a uniform spatial subdivision of the scene. The third technique is the method for ray propagation which reduces the number of rays to be traced. Finally, the Predict the Required Number of Rays (PRNR) method also reduces the number of rays to be traced using statistical techniques.

E.1.1. Reducing computational cost of intersections

There are two main techniques used to reduce the computational cost of ray-object intersection: object subdivision and spatial subdivision (Glassner, 1989).

In object subdivision faster object-ray intersections are achieved. This is done with one of the fundamental tools in ray tracing, namely the bounding volume. This is a volume or shape which surrounds an object in the scene and permits simpler ray intersection calculations. Only if the ray intersects the bounding volume is the actual object tested.

Spherical shaped bounding volumes were first introduced by Whitted (1980) as spheres are one of the simplest shapes to intersect. Modern ray tracers, such as STRAL, still use spherical bounding volumes (Pitz-Paal *et al.*, 2009). Later, Rubin *&* Whitted (1980) used rectangular axis-aligned boxes known as bounding boxes. SUNRAY uses a similar method based on the work of Kay *&* Kajiya (1986), which is an axis-aligned rectangle.

In spatial subdivision the space around an object is decomposed into regions. One method of doing this is to divide the space into a grid of axis-aligned boxes. This ensures fewer ray-object intersections.

E.1.2. Reducing the Number of Rays

Techniques for reducing the number of rays include both primary and secondary (reflected) rays. These techniques allow for the minimum amount of rays possible to be intersected, while still maintaining a level of accuracy. An example of a simple technique to reduce secondary rays is to extinguish a reflected ray after its flux has dropped below a predefined percentage and its contribution to the final result is negligible. A method to reduce primary rays is to use statistical tools. In certain situations a relatively small number of rays are required to produce statistically reliable results. This is the fundamental basis of the PRNR method described in Section 7.4.

Most literature on statistical techniques is focused on graphical ray tracers. Here, the objective is to render images for human vision and, therefore, techniques such as anti-aliasing (reducing the jagged edges found in computer generated images [Ashdown 2000]) have been developed (Lee, Redner *&* Uselton, 1985; Davies, 1992). Monte Carlo methods, which are used in SUNRAY, are also sometimes included in this category (Pharr *&* Humphreys, 2010).

No literature could be found which automatically predicts the required number of rays, as is done in the PRNR method (Section 7.4). Furthermore, no literature could be found which uses the same method as SUNRAY to propagate rays into a scene.

E.1.3. Using Generalised Rays

A ray is infinitely long and infinitesimally thin. These properties ensure easy representation, high accuracy, and efficient ray-object intersection. However, sacrificing one or more of these advantages by replacing a ray with a more general bundle of rays to form a cone or beam, can lead to faster execution times (Glassner, 1989).

General ray tracing procedures can be applied to a generalised ray. Early optical tools, such as those developed by Athavaley, Lipps *&* Vant-Hull (1979), used beam optics for the analysis of a central receiver plant. Lipps *&* Walzel

(1978) and Gomez & Ganot (1986) developed analytical methods to analyse central receivers and codes such as Helios used cone optics for central receiver simulations (Falcone, 1986).

Simulating reflection with cones, which are represented by an apex, centre line, and spread angle, requires using standard ray tracing techniques to compute a new centre line (Amanatides, 1984; Besant, 1895). However, intersection calculations require not only the point of intersection but also an indication of how much of the cone is blocked by the object. Cone optics have been used to help validate certain aspects of SUNRAY against simple scenes (Section 8.2.3).

Ray tracing with generalised rays requires imposed constraints, either restricting the type of objects or relaxing the degree of accuracy and accepting approximations instead (Glassner, 1989). Examples of modern tools which use generalised rays are DLR's HFCAL (Schwarzbözl *et al.*, 2009) and the HFDL (Wei *et al.*, 2010*a*).

E.2. Reducing the Cost of Intersections

Two methods are used in SURAY to reduce the computational cost of ray intersections, namely bounding boxes and the Grid.

E.2.1. Bounding Boxes

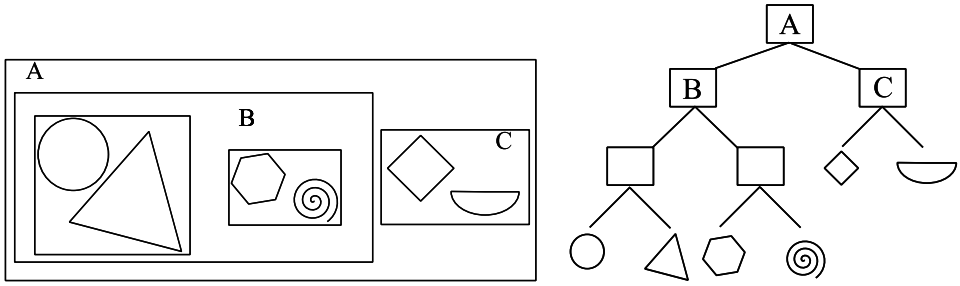
Every object in SUNRAY has a bounding box. Therefore, an efficient bounding box intersection test is required. The Bounding Box Hit function is based on the code given in (Suffern, 2007) and the pseudo-code given in (Shirley & Marschner, 2009) as well as the techniques in (Shirley & Morley, 2003), which use the Institute of Electrical and Electronics Engineers (IEEE) floating-point arithmetic to handle cases where rays have a negative value for $\mathbf{d}$. For example, $\mathbf{d} = (0, -1, 0)$ which, from equation (7.1), will cause division by zero. The pseudo-code for the Bounding Box Hit function is given as

$t_{xmin} = (x_0 - \mathbf{o}_x)/\mathbf{d}_x$
$t_{xmax} = (x_0 - \mathbf{o}_x)/\mathbf{d}_x$
$t_{ymin} = (y_0 - \mathbf{o}_y)/\mathbf{d}_y$
$t_{ymax} = (y_0 - \mathbf{o}_y)/\mathbf{d}_y$

IF $(t_{xmin} > t_{ymax})$ or $(t_{ymin} > t_{tmax})$
return FALSE
else
return TRUE

E.2.2. Grid

An extension on bounding volumes is to use what Rubin *&* Whitted (1980) coined Bounding Volume Hierarchies (BVH). The BVH technique encloses a number of objects within a larger bounding volume. If a ray does not hit the parent volume, there is no need to test the rays against the volumes contained within it. Shirley *&* Marschner (2009) use a *tree* of overlapping bounding boxes, as depicted in Figure E.1.

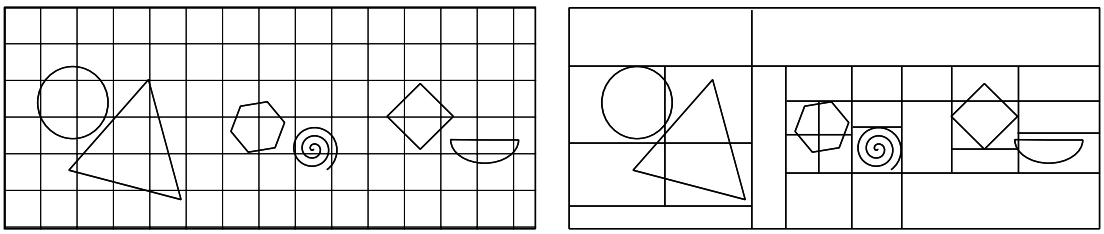


(a) Bounding volumes (b) BVH Tree

Figure E.1: Bounding Volume Hierarchies

In contrast to BVH, which divides objects into volumes, spatial subdivision divides the environment into volumes. This approach has been chosen for SUNRAY as emphasis is placed on the space rather than the object. This is important for SUNRAY because a size of the scene is defined by the objects in the scene, unlike graphical ray traces where a user chooses the size of the scene.

Spatial subdivision can be further broken down into two categories: non-uniform and uniform spatial subdivision (Glassner, 1989), as illustrated in Figure E.2. In non-uniform subdivision the space is broken down into regions of various sizes, based on the number of objects in those regions. Many data structures have been developed for non-uniform spatial sub-division, such as octrees (Knoll, 2006) and k-d trees (Wald *&* Havran, 2006).



(a) Uniform subdivision (b) Non-uniform subdivision

Figure E.2: Spatial subdivision

Uniform subdivision is non-adaptive. The total volume is broken down into a uniform regular 3D grid, also referred to as Spatially Enumerated Auxiliary Data Structure (SEADS) (Fujimoto, Tanaka *&* Iwata, 1986). Fujimoto *et al.* has shown that significant speeds can be gained using uniform subdivisions, because ray tracing non-uniform subdivisions involves complex processing for the traversal of divisions. However, there is no clear set advantage for using either uniform or non-uniform divisions and arguments can be made for and against both. For reasons of programming simplicity it was decided to implement uniform subdivision in SUNRAY.

E.3. Propagation Techniques: Reducing the Number of Rays

The method for ray propagation in SUNRAY has been specifically developed to reduce the number of rays. Ray propagation is not a trivial task. Therefore, various ray propagation techniques were investigated before one was chosen. This section describes techniques investigated for SUNRAY.

In the basic graphical ray tracer described in (Suffern, 2007), the author generates rays on a fixed plane positioned on the scene's z-y plane. These rays are generated in a uniform grip pattern. The user is able to vary the spacing between the rays as well as the horizontal and vertical length of the plane. Each ray is generated with a direction that is perpendicular to the plane spanning the computer monitor.

In more complex graphical ray tracers, this plane can be orientated at any angle. In these ray tracers rays all have a single origin (the eye of the viewer) and the direction is a vector from the eye to a uniform grid point on the plane (Shirley *&* Morley, 2003). Generating rays in this manner is the standard practice for most graphical ray tracers.

In the solar ray tracers Tonatiuh (Blanco, 2013) and SolTrace (NREL, 2012*b*) rays are on a plane positioned above the scene. The details of the ray generation in SolTrace are the intellectual property of NREL and are therefore not accessible, but the basic steps are as follows. First, the sun's direction is determined by a vector connecting the first stage and the source (NREL, 2012*b*). A plane, normal to this vector, is then populated with randomly generated points. At each point a ray is generated which is equal to the previously calculated sun direction vector. All primary rays are parallel to one another and the influence of the sunshape on the ray direction is implemented by perturbing the rays only after they have been reflected off an object. In Tonatiuh, the rays are not generated parallel to the plane, but are perturbed by a certain angle to simulate the sunshape.

In the ray tracer, STRAL, rays are not generated above the field but on a uniform raster on the surface of the heliostat. As the rays are generated on the primary surface, there are no wasted rays, although, large increases in speed do introduce new complications (Pitz-Paal *et al.*, 2009). For example, shading and blocking have to be handled using collision analysis.

Solfast also generates rays on the surface of the primary reflectors. Sampling points are uniformly distributed on the primary reflecting surface and on a disc representing the sun (Roccia *et al.*, 2012). The Solfast algorithm then determines if there is any shadowing between a sample point on the heliostat and a sample point on the sun. If no shadowing occurs, rays are reflected from the heliostat's sample point toward the receiver.

F. EXPERIMENTAL SET UP

To validate SUNRAY with an actual heliostat and target, an experiment was conducted which is able to measure both the magnitude of flux and the distribution of flux (fluxmap) on a receiver due to a number of heliostat profiles. The experiment was performed in association with Landman (2013).

F.1. Solar Flux Measurement Techniques

A number of flux measuring techniques have been developed and a summary of the techniques used in large-scale systems is provided in (Röger, Herrmann, Ebert, Prahl, Ulmer & Göhring, 2011). However, factors such as temperature, magnitude of flux, and target area differ between large-scale and small-scale experiments. For this thesis the experimental was designed specifically for low-flux ($<$10 000 W/m^2), small-scale experiments.

Flux measurement techniques can be divided into two categories: direct measurement and indirect measurement. In direct flux measurement, instruments are used to measure the incident flux directly as opposed to indirect methods, which measure the flux reflected off a target.

Flux sensors are one method commonly used for direct flux measurement. These sensors are based on the thermocouple principal and deliver a measurement signal proportional to the irradiance flux striking them (Diller, 1999). The majority of direct flux measurement techniques can only be used to measure flux at discrete points.

A common technique for indirect flux measurement in solar thermal systems is to use a digital camera. In a digital camera, images are captured on a light sensitive sensor made up of millions of photosensitive pixels. Each pixel outputs a voltage directly proportional to the amount of photons which strike it (Kelby, 2009). For a camera aimed at a target, the amount of photons hitting each pixel is proportional to the flux incident on the target. Digital cameras are able to produce a continuous flux map with very high spatial resolution, however, they cannot quantify the flux magnitude. Cameras therefore, they need to be calibrated.

F.2. Experimental Apparatus and Error Analysis

The experiment uses a combination of indirect and direct methods. A Complimentary Metal-Oxide Semi-Conductor (CMOS) digital camera (Kelby, 2009) is used for the indirect measurement and a Gardon circular foil flux sensor (Diller, 1999) is used for the direct flux measurement. The camera is positioned near normal to a flat target coated with matte white paint. The flux sensor is positioned at the centre of the target. By obtaining a direct flux measurement, the images captured with the camera can be calibrated and the entire flux map can be produced. The following section describes the major components of the experiment, the error introduced by each component, and the procedure followed to reduce the error. The specifications are provided in Table F.1.

Instrument	Specifications
	Indirect Measurement
Digital Camera	Nikon D5100
Lens	AF_SDX Zoom-Nikokor 55-200 mm f/4-5.6G IF-ED
Neutral Density Filter	KenkoPro1D8(W) 52 mm
Target	1 m^2 4 mm mild steel
Target Coating	Prominent Paint Wall Primer
	Direct Measurement
Flux Sensor	VatellCorpTG100-0 with AMP-15 S/N :9627
Data Logger	National Instruments C-DAQ 9181 S/N:16615BD

Table F.1: Experimental instrumentation

F.2.1. Camera

There are two main sensors in digital cameras: Charged Couple Device (CCD) and CMOS (Litwiller, 2001). The sensors differ in their charge to voltage conversion and are both comparable in image quality (Janesick & Putnam, 2003).

In a digital camera, limitations in camera hardware cause imperfections (noise) in the image captured (Kelby, 2009). Two types of noise exist: random and correlated. Random noise cannot be predicted from one image to another. Correlated noise, however, can be predicted and can be compensated for if a correlation factor is identified. The major sources of noise are bias (Mansouri, Marzani & Gouton, 2005), vignetting (Kelcey & Lucieer, 2012), and imperfect sensors (Mansouri *et al.*, 2005). To account for these errors, each image in the experiment was calibrated using the calibration procedure described by Mansouri *et al.* (2005).

An additional source of error was introduced, because the camera could not be positioned normal to the target and thus the image was captured from a non-zero incident angle. This error was reduced through corrective image warping using a MATLAB code. Details of the code are given in Landman (2013).

F.2.2. Flux Sensor

The flux sensor is an essential part of the experiment. A Gardon-type gauge, which is the most common gauge used in solar applications (Ballastrin, Estrada, Rodriguez-Alonso, Perez-Rabago, Langley & Barnes, 2004), was chosen because of its large range of measurement (0-50k W/m^2) and mechanical robustness (Barnes, 2012). The flux sensor was calibrated by the manufacturer Vatell Corporation (Vatell Corporation, 2012) and, therefore, no further error reduction was performed. The flux sensor gave a standard deviation of 4.7 mV under steady state conditions and a response time of approximately 3 seconds was observed.

F.2.3. Lambertian Surface of Target

A Lambertian surface will ensure that the incident flux is diffusely reflected in all directions allowing a camera to be placed at any location. Based on the recommendation of Griffith (2011), who performed experiments at the Council for Scientific and Industrial Research (CSIR) on the optical properties of South African paints, three near Lambertian paints were identified. An experiment was conducted where a qualitative measurement of the BRDF was determined using a digital camera. The paint which exhibited the most Lambertian properties was the Prominent Wall Primer.

F.3. Mirror Surface Shape Characterisation

A number of mirror characterisation techniques have been developed for CSP systems. Xiao, Wei, Lu, Yu & Wu (2012) provide a description of the common techniques. For the experiment, a Coordinate Measurement Machine (CMM) was used to measure the surface shape. The CMM specifications are provided in Table F.2 for the specifications.

A CMM uses a touch probe to measure the mirror surface at set intervals with a high volumetric accuracy of 6 μm. The output of the CMM machine is the 3D coordinate at each point. A sub-routine was written in SUNRAY which connected each coordinate point to form a triangular mesh. This mesh is stored as a PLY file (Section 5.2).

Measurements could not be done on the heliostats themselves and the mirrors had to be transported to the machine. Errors could have been introduced through the variation of the gravity vector, but their influence was assumed to be minimal due to the stressed glass and rigid supporting structure.

The location of the heliostat relative to the target was measured by hand with an assumed accuracy of 0.1 m.

CMM	Mitutoyo Bright Apex 710
Probe	RenshawPH10M
Probe Head	TP2 Touch Trigger
Working Volume	$700 \times 1000 \times 900$ mm

Table F.2: Coordinate measurement machine specifications

List of References

3D Object Converter [online] (2012). Available at: http://3dconverter.clanteam.com/index.html, [June 2012].

Abeinsa (2013). Infrastructure for a sustainable world. Available at: http://www.abeinsa.com/, [2013 August].

Alanod Solar (2012). Solar Reflection Products: Technical Information. Available at: http://www.alanod-solar.us/reflection-technical-information.php, [2012 November].

Amanatides, J. (1984). Ray tracing with cones. *Computer Graphics*, vol. 18, no. 3, pp. 129–135.

Amanatides, J. and Choi, K. (1997). Ray Tracing Triangular Meshes. In: *Proceedings of the Eighth Western Computer Graphics Symposium*.

Amanatides, J. and Woo, A. (1987). A Fast Voxel Traversal Algorithm for Ray Tracing. *Dept. of Computer Science University of Toronto*.
Available at: http://www.cse.yorku.ca/~amana/research/grid.pdf

Anton, H. and Rorres, C. (2008). *Elementary Linear Algebra Applications*. Wiley. ISBN 9788126518692.

Appel, A. (1968). Some techniques for shading machine renderings of solids. In: *Spring Joint Computer Conference*, pp. 37–47.

Arvo, J. (1986). Backward ray tracing. In: *Developments in Ray Tracing, Computer Graphics, Proc. of ACM SIGGRAPH 86 Course Notes*, pp. 259–263.

Ashdown, I. (1994). *Radiosity: a programmers perspective*. John Wiley & Sons, Inc.

Athavaley, K., Lipps, F. and Vant-Hull, L. (1979). An analysis of the terminal concentrator concept for solar central receiver systems. *Solar Energy*, vol. 22, no. 6, pp. 493–504.

Austern, M. (2005). Draft Technical Report on C ++ Library Extensions.

Ballastrin, J., Estrada, C.A., Rodriguez-Alonso, M., Perez-Rabago, C., Langley, L.W. and Barnes, A. (2004). High-heat-flux sensor calibration using calorimetry. *Metrologia*, vol. 41, pp. 314–318.

Ballestrín, J. and Marzo, A. (2012). Solar radiation attenuation in solar tower plants. *Solar Energy*, vol. 86, no. 1, pp. 388–392.

Barnes, A. (2012). Flux Sensors. Email to: Adam Barnes [online]. Available email eng@vatell.com.

Behar, O., Khellaf, A. and Mohammedi, K. (2013 July). A review of studies on central receiver solar thermal power plants. *Renewable and Sustainable Energy Reviews*, vol. 23, pp. 12–39. ISSN 13640321.
Available at: http://linkinghub.elsevier.com/retrieve/pii/S1364032113001184

Besant, W. (1895). *Conic sections, treated geometrically*. George Bell and Sons. Made available in Ebook #29913 butenberg.org.

Biggs, F. and Vittltoe, C.N. (1976). The Helios Model for the Optical Behavior of Reflecting Solar Concetrators. Tech. Rep., SANDIA.

Blanco, M. (2009). Preliminary validation of Tonatiuh. In: *International SolarPACES Symposium*.

Blanco, M. (2013). Downloads -Tonatiuh - A Monte Carlo ray tracer for the optical simulation of solar concentrating systems.
Available at: `https://code.google.com/p/tonatiuh/downloads/list`

Blanco, M., Mutuberria, A., Monreal, A. and Albert, R. (2011). Results of the empirical validation of Tonatiuh at Mini-Pegase CNRS- PROMES facility. In: *SolarPACES*. Seville.

Blanco-muriel, M., Alarcon-padilla, D.C. and Lopez, T. (2001). Computing the solar vector. *Solar Energy*, vol. 70, no. 5, pp. 431–441.

Bode, S.-J. and Gauché, P. (2012). Review of optical software for use in concentrating. In: *1st South African Solar Energy Conference*, pp. 1–8.

Bode, S.-J., Gauché, P. and Griffith, D. (2013). A novel approach to reduce ray tracing simulation times by predicting number of rays. In: *SolarPACES*. Las Vegas, United States of America.

Bourke, P. (2003). Data Formats.
Available at: `http://paulbourke.net/dataformats/`

Box, G.E.P. and Muller, M. (1958). A note on the generation of random normal deviates. *The Annals of Mathematical Statistics*, vol. 29, no. 2, pp. 610–611.
Available at: `http://projecteuclid.org/DPubS/Repository/1.0/Disseminate?view=body\&id=pdf\_1\&handle=euclid.aoms/1177706645`

Boyle, G. (2004). *Renewable Energy*. 3rd edn. Oxford. ISBN 978-0-19-954533-9.

Buck, R., Pfahl, A. and Roos, T.H. (2012). Target aligned heliostat field layout for non-flat terrain. In: *South Africa Solar Energy Conference*, pp. 1–9.

Buie, D., Dey, C. and Bosi, S. (2003*a*). The effective size of the solar cone for solar concentrating systems. *Solar energy*, vol. 74, no. 5, pp. 417–427.

Buie, D., Monger, A. and Dey, C. (2003*b*). Sunshape distributions for terrestrial solar simulations. *Solar Energy*, vol. 74, no. 2, pp. 113–122.

Buie, D. and Monger, A.G. (2004 January). The effect of circumsolar radiation on a solar concentrating system. *Solar Energy*, vol. 76, no. 1-3, pp. 181–185. ISSN 0038092X.
Available at: `http://linkinghub.elsevier.com/retrieve/pii/S0038092X03003451`

Butler, L. and Pettit, B. (1977). Optical evaluation techniques for reflecting solar concentrators. In: *Optics Applied to Solar Energy Conversion*, vol. V, pp. 43–49.

Butt, R. (2009). *Introduction To Numerical Analysis Using MATLAB*. Jones & Bartlett Learning. ISBN 9780763773762.

Chen, H. (2005). *Tutorial on Monte Carlo Sampling*, vol. 1. Dept. of Chemical & Biomoleculare Eng. Ohio State University.

Chong, K.-K. and Tan, M.H. (2011). Comparison study of two different sun-tracking methods in optical efficiency of heliostat field. *International Journal of Photoenergy*, vol. 2012, pp. 1–10. ISSN 1110-662X.
Available at: `http://www.hindawi.com/journals/ijp/2012/908364/`

Coustet, C. (2013). SolFast. Email to: Mr. Coustet [online]. Available email christophe.coustet@hpc-sa.com.

Dandekar, K.R., Arredondo, A., Xu, G. and Ling, H. (2002). Using ray tracing to evaluate smart antenna system performance for wireless communications. *Wireless Communications*, vol. 8, pp. 480–487.

Davies, P.a. (1992 December). Methods of choosing sample rays in ray-tracing computations. *Applied optics*, vol. 31, no. 34, pp. 7277–82. ISSN 0003-6935.
Available at: `http://www.ncbi.nlm.nih.gov/pubmed/20802594`

Delatorre, J., Baud, G., Bézian, J., Blanco, S., Caliot, C., Cornet, J. and et al. (2013 May). Monte Carlo advances and concentrated solar applications. *Solar Energy*. ISSN 0038092X.
Available at: `http://linkinghub.elsevier.com/retrieve/pii/S0038092X13001448`

Department of Environmental Affairs (2012). Strategic Plan: 1 April 2012 to 31 March 2017. Tech. Rep., DEA.
Available at: `http://www.info.gov.za/view/DownloadFileAction?id=175475`

Diller, T. (1999). Heat Flux. In: *The Measurement, Instrumentation and Sensors Handbook*, chap. 34. CRC Press.

Duffie, J.A. and Beckman, W.A. (2006). *Solar Engineering of Thermal Processes*. John Wiley & Sons. ISBN 978-0-471-69867-8.

Dutré, P. (2003 April). Global Illumination Compendium.
Available at: `http://www.ncbi.nlm.nih.gov/pubmed/22451589`

Energy, T. (2010). Photo Library.
Available at: `http://www.torresolenergy.com/TORRESOL/image-library/en?initdate=\&enddate=\&categoria=cw4e4122c7d262d`

Falcone, P.K. (1986). A handbook for solar central receiver design. Tech. Rep., Sandia National Labs., Livermore, CA (USA).

Fleet, D. and Hertzmann, A. (2009). Probability Density Functions (PDFs).

Fluri, T.P. (2009). The potential of concentrating solar power in South Africa. *Energy Policy*, vol. 37, no. 12, pp. 5075–5080.

Fox, G.C., Williams, R.D. and Messina, P.C. (1994). *Parallel Computing Works*. Morgan Kaufmann Publishers. ISBN 1-55860-253-4.
Available at: `http://www.netlib.org/utk/lsi/pcwLSI/text/BOOK.html`

Freisleben, B., Hartmann, D. and Kielmann, T. (1997). Parallel raytracing: a case study on partitioning and scheduling on workstation clusters. *Proceedings of the Thirtieth Hawaii International Conference on System Sciences*, vol. 1, pp. 596–605.
Available at: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=667407`

Fujimoto, A., Tanaka, T. and Iwata, K. (1986). ARTS: Accelerated Ray-Tracing System.

IEEE Computer Graphics and Applications, vol. 6, no. 4, pp. 16–26. ISSN 0272-1716. Available at: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4056861`

Ganssie, J. (2004). *The Firmware Handbook*. Linacre House. ISBN 0-7506-7606.

Garcia, P., Ferriere, A. and Bezian, J. (2008 March). Codes for solar flux calculation dedicated to central receiver system applications: A comparative review. *Solar Energy*, vol. 82, no. 3, pp. 189–197. ISSN 0038092X. Available at: `http://linkinghub.elsevier.com/retrieve/pii/S0038092X07001727`

Gauché, P., Backström, T.W.V. and Brent, A.C. (2012). A value proposition of CSP for South Africa. In: *Southern African Solar Energy Conference*.

Gee, R., Brost, R., Zhu, G. and Jorgensen, G. (2010). An improved method for characterizing reflector. In: *Solar Paces Conference*.

Glassner, A.S. (1989). *An Introduction to Ray Tracing*. 3rd edn. Academic Press. ISBN 0-12-286760-4.

Goldsmith, J. and Salmon, J. (1987). Automatic Creation of Object Hierachies for Ray Tracing.

Gomez, A. and Ganot, J.A.T. (1986). An analytic function for the flux density. *Solar Energy*, vol. 37, no. 3, pp. 215–234.

Goodman, D.S. (2004). *General principles of geometrical optics*, vol. I. 3rd edn. McGraw-Hill.

Goodman, J. (2005). Monte Carlo Methods: Box Muller. Available at: `Availablehttp://www.math.nyu.edu/faculty/goodman/teaching/MonteCarlo2005/notes/GaussianSampling.pdf`

Grena, R. (2008). An algorithm for the computation of the solar position. *Solar Energy*, vol. 82, no. 5, pp. 462–470.

Griffith, D. (2011). BRDF paint and Experiments. Email to: Derek Griffith [online]. Available email dgriffith@csir.co.za.

Guo, M., Wang, Z., Zhang, J., Sun, F. and Zhang, X. (2011 May). Accurate altitude-azimuth tracking angle formulas for a heliostat with mirror-pivot offset and other fixed geometrical errors. *Solar Energy*, vol. 85, no. 5, pp. 1091–1100. ISSN 0038092X. Available at: `http://linkinghub.elsevier.com/retrieve/pii/S0038092X11000867`

Haines, E. (2001). Triangles per Vertex. *Ray Tracing News*, vol. 14, no. 1. Available at: `http://tog.acm.org/resources/RTNews/html/rtnv14n1.html`

Hanraham, P. (2003). Monte Carlo Path Tracing. In: *Monte Carlo Ray Tracing: Siggraph 2003 Course 44*, pp. 89–108.

Hartnady, C.J. (2010). South Africa's diminishing coal reserves. *South African Journal of Science*, vol. 106, no. 9-10, pp. 1–5.

Heckbert, P.S. (1994). *Graphics Gems IV*. 4th edn. Morgan Kaufmann, San Diago.

Heun, M.K., van Niekerk, J.L., Swilling, M., Meyer, A.J., Brent, A. and Fluri, T.P. (2010). Learnable Lessons on Sustainability From the Provision of Electricity in South Africa. *ASME 2010 4th International Conference on Energy Sustainability,*

Volume 1, , no. 13, pp. 13–23.
Available at: http://proceedings.asmedigitalcollection.asme.org/proceeding.aspx?articleid=1607110

Ho, C.K. (2008). Software and Codes for Analysis of Concentrating Solar Power Technologies. Tech. Rep. December, Sandia National Laboratories, Albuquerque.

Ho, C.S. (2012). HELIOS. Email to: Clifford Ho [online]. Available email ckho@sandia.gov.

Igel, E.A. and Hughes, R.L. (1979). Optical analysis of solar facility heliostats. *Solar Energy*, vol. 22, pp. 283–295.

IPCC (2013). The Physical Science Basis. Contribution of Working Group I to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change Intergovernmental Panel on climate Change [Stocker, T.F. and D. Qin and G.K. Plattner and M. Tignor and S. K. Allen and J. Boschung and A. Nauels and Y. Xia and V. Bex and P.M. Midgley (eds.)]. Tech. Rep., Intergovomental Panel on Climate Change, Cambridge, United Kingdom and New York, NY, USA.

Izygon, M. (2012). TieSOL. Email to: Michel Izygon [online]. Available email Michel.Izygon@tietronix.com.

Izygon, M., Armstrong, P., Nilsson, C. and Vu, N. (2011). TieSOL - A GPU-Based Suite of Software for Central Receiver Solar Power Plants. In: *SolarPACES*. Seville.

Janesick, J. and Putnam, G. (2003 December). Developments and applications of high-performance CCD and CMOS imaging arrays. *Annual Review of Nuclear and Particle Science*, vol. 53, no. 1, pp. 263–300. ISSN 0163-8998.
Available at: http://www.annualreviews.org/doi/abs/10.1146/annurev.nucl.53.041002.110431

Jenkins, F.A. and White, H.E. (1957). *Fundamentals Of Optics*. 3rd edn. McGraw-Hill.
Available at: http://ia600704.us.archive.org/19/items/FundamentalsOfOptics/JenkinsWhite-FundamentalsOfOptics.pdf

John, V. (2005). *Geometry for Computer Graphics: Formulae, Examples & Proofs*. Springer. ISBN 1852338342.

Johnston, G. (1998). Focal region measurements of the 20 m2 tiled dish at the australian national university. *Solar Energy*, vol. 63, no. 2, pp. 117–124.

Jones, S.A. and Stone, K. (1999*a*). Analysis of solar two heliostat tracking error sources. Tech. Rep., Sandia National Laboratories, Albuquerque, NM, and Livermore, CA.

Jones, S.A. and Stone, K.W. (1999*b*). Analysis of strategies to improve heliostat tracking at solar two. Tech. Rep., SANDIA.

Kajiya, J.T. (1986). The Rendering Equation. *SIGGRAPH*, vol. 20, no. 4, pp. 143–150.

Karni, J., Buck, R., Pfahl, A., Bligh, T. and Chen, Y.T. (2004). Comparison of two sun tracking methods in the application of a heliostat field. *Journal of Solar Energy Engineering*, vol. 126, no. 17, pp. 638–644.

Kay, T.L. and Kajiya, J.T. (1986 August). Ray tracing complex scenes. *ACM SIGGRAPH Computer Graphics*, vol. 20, no. 4, pp. 269–278. ISSN 00978930.
Available at: http://portal.acm.org/citation.cfm?doid=15886.15916

Kelby, S. (2009). *Digital Photography*. 2nd edn. Pearson Education. ISBN 9780133443479.

Kelcey, J. and Lucieer, A. (2012). Sensor correction of a 6-band multispectral imaging sensor for uav remote sensing. *Remote Sens*, vol. 4, no. 5, pp. 1462–1493.

Kennedy, C.E. (2002). *Review of mid-to high-temperature solar selective absorber materials*. National Renewable Energy Laboratory Golden Colorado.

Kiera, M. (1989). Heliostat field: computer codes, requirements, comparison of methods. *GAST–The Gas-Cooled Solar Tower Technology Program. Proceedings of the Final Presentation. Springer, Berlin*.

Kistler, B.L. (1986). A user's manual for Delsol3: A computer code for calculating the optical performance and optimal system design for solar thermal central receiver plants. Tech. Rep., SANDIA.

Knoll, A. (2006). A Survey of Octree Volume Rendering Methods. *Scientific Computing and Imaging Institute*.

Kretzschmar, H. (2013). *Concept Development of the Hybrid Pressurized Air Receiver (HPAR) for Combined Cycle Solar Thermal Power Plants*. MSc Mechanical Engineering Thesis, University of Stellenbosch.

Kribus, A. and Ries, H. (2003). LiMoNAED : A limited motion , non-shading , asymmetric , ecliptic-tracking dish. *Solar Energy*, vol. 73, no. 5, pp. 337–344.

Kribus, A., Vishnevetsky, I., Yogev, A. and Rubinov, T. (2004). Closed loop control of heliostats. *Energy*, vol. 29, pp. 905–913.

Kroese, D.P., Taimre, T. and Zdravko, B.I. (2011). *Handbook of Monte Carlo Methods*. Wiley. ISBN 978-0-470-17793-8.

Kröger, D.G. (2012). The Stellenbosch UNiversity Solar POwer Thermodynamic cycle. Tech. Rep..

Landman, W.A. (2013). *Optical Performance of the Reflective Surface Profile of a Heliostat*. MS Mechanical Engineering, University of Stellenbosch.

Leary, P. and Hankins, J.D. (1979). A user's guide for MIRVAL - a computer code for comparing designs of heliostat-receiver optics for central receiver solar power plants. Tech. Rep., Sandia Laboratories SAND77-8280.

Lee, M.E., Redner, R.A. and Uselton, S.P. (1985). Statistically optimized sampling for distributed ray tracing. In: *ACM SIGGRAPH Computer Graphics*, vol. 19, pp. 61–68. ACM.

Leonardi, E. (2012). Availability of CRS4-2. Email to: E. Keonardi [online]. Available email ermy@crs4.it.

Leonardi, E. and D'guanno, B. (2011 August). CRS4-2: A numerical code for the calculation of the solar power collected in a central receiver system. *Energy*, vol. 36, no. 8, pp. 4828–4837. ISSN 03605442.
Available at: `http://linkinghub.elsevier.com/retrieve/pii/S0360544211003422`

Lipps, F. and Walzel, M. (1978). An analytic evaluation of the flux density due to sunlight reflected from a flat mirror having a polygonal boundary. *Solar Energy*, vol. 21, no. 2, pp. 113–121.

Litwiller, D. (2001). CCD vs. CMOS. *Photonics Spectra*, vol. 35, no. 1, pp. 154–158.

Lutchman, S. (2013). *Heliostat Field Layout Optimisation for a Central Receiver*. MSc

Mechanical Engineering, University of Stellenbosch.

Lutchman, S., Groenwald, A. and Bode, S. (2013). On using a gradient-based method for heliostat field layout optimization. In: *SolarPACES*. Las Vegas, USA.

Lyman Ott, R. (1988). *An Introduction to Statistical Methods and Data Analyis*. 4th edn. Wadsworth Inc. ISBN 0-534-93150-2.

Mansouri, A., Marzani, F. and Gouton, P. (2005). Development of a protocol for ccd calibration: Application to a multispectral imaging system. *International Journal of Robotics and Automation*, vol. 20, no. 2. ISSN 1925-7090.
Available at: `http://www.actapress.com/PaperInfo.aspx?paperId=20490`

Marland, G., Boden, T. and Andres, R. (2011). Global, regional, and national fossil fuel CO2 emissions. *Trends: A compendium of data on global change*.
Available at: `http://cdiac.ornl.gov/trends/emis/overview`

Marsaglia, G. and Bray, T.A. (1964). A convenient method for generating normal variables. *SIAM Review*, vol. 6, no. 4, pp. 260–264.

Matsumoto, M. and Nishimura, T. (1998 January). Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, vol. 8, no. 1, pp. 3–30. ISSN 10493301.
Available at: `http://portal.acm.org/citation.cfm?doid=272991.272995`

Meyen, S., García, A.F. and Kennedy, C. (2011). Measurement of solar weighted reflectance of mirror materials for concentrating solar power technology with commercially available contents. Tech. Rep. May.

Meyen, S., Montecchi, M., Kennedy, C., Zhu, G., Gray, M., Crawford, J. and et al. (2013). Parameters and method to evaluate the solar reflectance properties of reflector materials for concentrating solar power technology Draft Version 2.4. Tech. Rep. March, SolarPACES.

Meyers, S. (1995). *More Effective C++: 35 new ways to improve your programs and designs*. Pearson Education.

Michalsky, J.J. (1988). The astronomical almanac's algorithm for approximate solar position (1950–2050). *Solar Energy*, vol. 40, no. 3, pp. 227–235.

Mills, D. (2004 January). Advances in solar thermal electricity technology. *Solar Energy*, vol. 76, no. 1-3, pp. 19–31. ISSN 0038092X.
Available at: `http://linkinghub.elsevier.com/retrieve/pii/S0038092X03001026`

Mitsos, A. (2012). Biomimetic Code. Email to: Alexander Mitsos [online]. Available email mitsos@mit.edu.

Modest, M.F. (2003). *Radiative Heat Transfer*. 2nd edn. Academic Press. ISBN 0-12-503163-7.

Mutuberria, A. (2009). SolTrace projects to comparisson with Toantiuh.
Available at: `https://code.google.com/p/tonatiuh/downloads/detail?name=archivosSolTrace.zip\&can=4\&q=`

Neumann, a. and Witzke, a. (1999 September). The influence of sunshape on the DLR Solar Furnace beam. *Solar Energy*, vol. 66, no. 6, pp. 447–457. ISSN 0038092X.
Available at: `http://linkinghub.elsevier.com/retrieve/pii/`

S0038092X99000481

Neumann, A., Witzke, A., Jones, S.A. and Schmitt, G. (2002). Representative Terrestrial Solar Brightness Profiles. *Journal of Solar Energy Engineering*, vol. 124, no. 2, p. 198. ISSN 01996231.
Available at: http://solarenergyengineering.asmedigitalcollection.asme.org/article.aspx?articleid=1456476

Nicodemus, F., Richmond, J. and Hsia, J. (1977). Geometrical Considerations and Nomenclature for Reflectance. Tech. Rep. October, Institute for Basic Standards National Bureau of Standards, Wahington.

Noone, C.J., Torrilhon, M. and Mitsos, A. (2012 February). Heliostat field optimization: A new computationally efficient model and biomimetic layout. *Solar Energy*, vol. 86, no. 2, pp. 792–803. ISSN 0038092X.
Available at: http://linkinghub.elsevier.com/retrieve/pii/S0038092X11004373

NREL (). NREL: Solar and Lunar position calculators.
Available at: http://www.nrel.gov/midc/solpos/

NREL (2000). Solpos: Documentation.
Available at: http://rredc.nrel.gov/solar/codesandalgorithms/solpos/aboutsolpos.html

NREL (2012*a*). NREL SolTrace - Download.
Available at: http://www.nrel.gov/csp/soltrace/download.html

NREL (2012*b*). SolTrace Help File.

Palik, E.D. (1985). *Handbook of Optical Constants of Solids*. ISBN 9780125444200.

Perepelitsa, D.V. (2006). Johnson Noise and Shot Noise. Tech. Rep. 2, MIT Department of Physics.

Pettit, R.B. (1977). Characterization of the reflected beam profile of solar mirror materials. *Solar Energy*, vol. 19, no. 6, pp. 733–741.

Pharr, M. and Humphreys, G. (2010). *Physically Based Rendering: From Theory to Implementation*. 2nd edn. Morgan Kaufmann. ISBN 9780123750792.

Phong, B.T. (1975 June). Illumination for computer generated pictures. *Communications of the ACM*, vol. 18, no. 6, pp. 311–317. ISSN 00010782.
Available at: http://portal.acm.org/citation.cfm?doid=360825.360839

Pitz-Paal, R., Schwarzbözl, P. and Ulmer, S. (2009). A new fast ray tracing tool for high-precision simulation of heliostat fields. *Journal of Solar Energy Engineering*, vol. 131, pp. 031002–1.

Puliaev, S., Penna, J.L., Jilinski, E.G. and Andrei, A.H. (2000). Solar diameter observations at Observatorio Nacional in 1998-1999. *Astronomy & Astrophysics Supplement Series*, vol. 267, pp. 265–267.

Querry, M.R. (1969). Direct solution of the generalized fresnel reflectance equations. *JOSA*, vol. 59, no. 7, pp. 876–877.

Reda, I. and Andreas, A. (2008). Solar Position Algorithm for Solar Radiation Applications. Tech. Rep. January, NREL.

Ries, H. and Rabl, A. (1994 October). Edge-ray principle of nonimaging optics. *Journal*

of the Optical Society of America A. ISSN 1084-7529.
Available at: `http://www.opticsinfobase.org/abstract.cfm?URI=josaa-11-10-2627`

Ries, H. and Schubnell, M. (1990). The optics of a two-stage solar furnace. *Solar Energy Materials*, vol. 21, pp. 213–217.

Riveros, D. (2012). ISOS availability. Email to: Mr. Riveros [online]. Available email driveros@geofisica.unam.mx.

Riveros-Rosas, D., Sánchez-González, M. and Estrada, C.A. (2008). Three-dimensional analysis of a concentrated solar flux. *Journal of Solar Energy Engineering*, vol. 130, no. 1, p. 014503. ISSN 01996231.
Available at: `http://link.aip.org/link/JSEEDO/v130/i1/p014503/s1\&Agg=doi`

Robert, C.P. and Casella, G. (2004). *Monte Carlo Statistical Methods*. Springer.

Roccia, J., Piaud, B., Coustet, C., Caliot, C., Guillot, E., Flamant, G. and Delatorre, J. (2012). SOLFAST, a Ray-Tracing Monte-Carlo software for solar concentrating facilities. vol. 369, no. 1, p. 012029.

Röger, M., Herrmann, P., Ebert, M., Prahl, C., Ulmer, S. and Göhring, F. (2011). Flux density measurement on large-scale receivers. In: *SolarPACES*. Granada, Spain.

Rubin, S.M. and Whitted, T. (1980). A 3-Dimensional Representation for Fast Rendering of Complex Scenes. *Proc. SIGGRAPH'80, Computer Graphics*, vol. 14, no. 3, pp. 110–116.

Rutledge, D. (2011 January). Estimating long-term world coal production with logit and probit transforms. *International Journal of Coal Geology*, vol. 85, no. 1, pp. 23–33. ISSN 01665162.
Available at: `http://linkinghub.elsevier.com/retrieve/pii/S0166516210002144`

SANDIA (2012). Sandia National Laboratories: CSP Software and Tools.
Available at: `http://energy.sandia.gov/?page\_id=6530`

Schildt, H. (2002). *The Complete Reference: C++*. 4th edn. McGraw Hill Professiona. ISBN 978-0071634809.

Schubnell, M. (1992 August). Influence of circumsolar radiation on aperture, operating temperature and efficiency of a solar cavity receiver. *Solar Energy Materials and Solar Cells*, vol. 27, no. 3, pp. 233–242. ISSN 09270248.
Available at: `http://linkinghub.elsevier.com/retrieve/pii/0927024892900854`

Schwarzbözl, P. (2012). Stral and HFCAL. Email to: Peter Schwarzbözl [online]. Available email peter.schwarzboezl@dlr.de.

Schwarzbözl, P., Schmitz, M. and Pitz-paal, R. (2009). Visual HFLCAL: A software tool for layout and optimisation of heliostat fields. In: *SolarPACES*. Berlin.

Sengupta, M. and Wagner, M. (2012). Atmospheric attenuation in central receiver systems from DNI measurements. Tech. Rep. 1, National Renewable Energy Laboratory (NREL), Golden CA.
Available at: `http://ases.conference-services.net/resources/252/2859/pdf/SOLAR2012\_0765\_fullpaper.pdf`

Shirley, P. (2003). Fundamentals of Monte Carlo Intergration. In: *Monte Carlo: Siggraph*

Course, chap. 2.

Shirley, P. and Marschner, S. (2009). *Fundametals of Computer Graphics*. CRC Press. ISBN 9781439865521.

Shirley, P. and Morley, R.K. (2003). *Realistic Ray Tracing*. A K Peters, Ltd.

Shirley, P. and Wang, C. (1992). Distribution Ray Tracing : Theory and Practice. In: *Third Eurographics Workshop on Rendering*, pp. 1–18.

Siegel, R. and Howell, J.R. (1992). *Thermal Radiation Heat Transfer*. 3rd edn. Taylor & Francis. ISBN 0-89116-271-2.

Solomon, S., Qin, D., Manning, M., Chen, Z., Marquis, M., Averyt, K., Tignor, M. and Miller (eds.) (2007). IPCC, 2007: Summary for Policymakers. In: Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change. Tech. Rep., Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA.

Spencer, G. and Murty, M.V.R.K. (1961). General Ray-Tracing Procedure. *Journal of the Optical Society of America*, vol. 52, no. 6, pp. 672–678.

Stewart, J. (2003). *Calculus*. 5th edn. Thomson. ISBN 0-534-27408-0.

Stine, W. and Harrigan, R. (1986). *Power From the Sun*. John Wiley & Sons.
Available at: `Republishedonlineas"Powerfromthesun"http://www.powerfromthesun.net/`

Stroustrup, B. (1995). *The C++ Programming Language*. 2nd edn. Pearson Education. ISBN 9788131705216.

Suffern, K. (2007). *Ray Tracing from the Ground Up*. A K Peters, Ltd. ISBN 978-1-56881-272-4.
Available at: `http://www.raytracegroundup.com/`

Tam, C.M. and Tan, K.K. (2001). Non-imaging , focusing heliostat. *Solar Energy*, vol. 71, no. 3, pp. 155–164.

The Fortan Company [online] (2003).

The GNU General Public License v2 [online] (1991).

Thekaekara, M. (1976). Solar radiation measurement: techniques and instrumentation. *Solar Energy*, vol. 18, no. 4, pp. 309–325.

Torrence, K.B.o.S. and Sparrow, M. (1967). Theory for off-specular reflection from roughed surfaces. *Journal of Optical Society of Ameria*, vol. 57, no. 9, pp. 1105–1114.

Trimble (2013). Google Sketchup.
Available at: `http://www.sketchup.com/intl/en/index.html`

Turk, G. (1998). The PLY File Format.
Available at: `http://www.cc.gatech.edu/projects/large\_models/ply.html`

Turk, G. and Levoy, M. (1994). Zippered polygon meshes from range images. *Pro eedings of the 21st annual conference on Computer graphics and interactive techniques - SIGGRAPH '94*, pp. 311–318.
Available at: `http://portal.acm.org/citation.cfm?doid=192161.192241`

University, S. (2013). Stellenbosch Weather.
Available at: `weather.sun.ac.za`

van Niekerk, L. (2011). South Africa Yearbook 2011/12.

Available at: http://www.gcis.gov.za/sites/www.gcis.gov.za/files/docs/resourcecentre/yearbook/2011/13_Energy.pdf

Vant-Hull, L.L. (2012). RCELL codes. Email to: Mr. Vant-Hull [online]. Available email -.

Vant-Hull, L.L. and Izygon, M. (1998). Optimization of Central Receiver Fields to Interface with Applications Requiring High Flux Densities. In: *9th International Symposium on Solar Thermal Concentrating Technologies*. Font-Romeu, France.

Vant-Hull, L.L. and Izygon, M.E. (2003). Guideline to central receiver system heliostat field optimization. *Advances in solar energy*, vol. 15, pp. 1–42.

Vatell Corporation (2012). Heat Flux Specialists Inc.
Available at: http://www.vatell.com/

Vincent, J.F.V., Bogatyreva, O.a., Bogatyrev, N.R., Bowyer, A. and Pahl, A.-K. (2006 August). Biomimetics: its practice and theory. *Journal of the Royal Society, Interface / the Royal Society*, vol. 3, no. 9, pp. 471–82. ISSN 1742-5689.
Available at: http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1664643\&tool=pmcentrez\&rendertype=abstract

Vittitoe, C.N. and Biggs, F. (1978). Terrestrial Propogation Loss. In: *Amer. Sec. ISES meeting*. Denver.

Wald, I. and Havran, V. (2006 September). On building fast kd-Trees for Ray Tracing, and on doing that in O(N log N). *2006 IEEE Symposium on Interactive Ray Tracing*, pp. 61–69.
Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4061547

Wei, X. (2012). HFLD code. Email to: Xiundon Wei [online]. Available email wei.xiudong@yahoo.com.cn.

Wei, X., Lu, Z., Wang, Z., Yu, W., Zhang, H. and Yao, Z. (2010 September*a*). A new method for the design of the heliostat field layout for solar tower power plant. *Renewable Energy*, vol. 35, no. 9, pp. 1970–1975. ISSN 09601481.
Available at: http://linkinghub.elsevier.com/retrieve/pii/S0960148110000431

Wei, X., Lu, Z., Yu, W. and Wang, Z. (2010 April*b*). A new code for the design and analysis of the heliostat field layout for power tower system. *Solar Energy*, vol. 84, no. 4, pp. 685–690. ISSN 0038092X.
Available at: http://linkinghub.elsevier.com/retrieve/pii/S0038092X10000332

Welton, D. (2011). Programming Language Popularity.
Available at: http://www.langpop.com/

Whitted, T. (1980). An improved illumination model for shaded display. *Communications of the ACM*, vol. 23, no. 6, pp. 343–349.

Wilbert, S., Reinhardt, B., Devore, J., Röger, M. and Gueymard, C. (2011). Measurement of solar radiance profiles with the sun and aureole measurement system (SAM). In: *SolarPACES*. Granada, Spain.

Winkler, G. (2003). *Image analysis, random fields and Markov chain Monte Carlo meth-*

ods: a mathematical introduction. 2nd edn. Springer. ISBN 3-540-44213-8.

Xiao, J., Wei, X., Lu, Z., Yu, W. and Wu, H. (2012 June). A review of available methods for surface shape measurement of solar concentrator in solar thermal power applications. *Renewable and Sustainable Energy Reviews*, vol. 16, no. 5, pp. 2539–2544. ISSN 13640321.
Available at: `http://linkinghub.elsevier.com/retrieve/pii/S1364032112000755`

Zaibel, R., Dagan, E., Karni, J. and Ries, H. (1995). An astigmatic corrected target-aligned concentration heliostat for high. *Solar Energy Materials and Solar Cells*, vol. 37, pp. 191–202.