ARC Project A89131897

The Development of Modelling Strategies for Whole Sky Spectrums under Real Conditions for International Use

Geoffrey G. Roy
Nancy Ruck
Graeme Reid
Frederick C. Winkelmann
Warren Julian

University of Sydney
Murdoch University
September 1995

ARC Project A89131897

Final Report

The Development of Modelling Strategies for Whole Sky Spectrums under Real Conditions for International Use

Geoffrey G. Roy
Computer Science, PSET,
Murdoch University,
Murdoch WA 6150

Nancy C. Ruck
Department of Architectural & Design Science
University of Sydney,
Sydney NSW 2006

Graeme Reid
Computer Science, PSET,
Murdoch University,
Murdoch WA 6150

Frederick C. Winkelmann
Lawrence Berkeley Laboratory
Berkeley, California 94720
United States of America

Warren Julian
Department of Architectural & Design Science
University of Sydney,
Sydney NSW 2006

The University of Sydney

Murdoch University

September 1995

Acknowledgments

This is the final report from a research project aimed at the development of a new universal method for the development of sky luminance models for use in daylighting studies and the simulation of interior lighting conditions in buildings.

Financial support for the project came from the Australian Research Council (Grant No. A89131897), and is gratefully acknowledged.

Contents

SUMMARY

1.0 Project Objectives

The aim of this project is to widen the research of the recent ARC research project entitled 'The development and verification of sky models which represent the predominant skies in Australia for the prediction of daylight in buildings' to include sky conditions worldwide. The data from luminance measurements from the International Daylight Measurement Program (IDMP) are used to develop modelling strategies which have been applied and tested in a proven internationally used computer program. The results are to be used in the development of software packages in the International Energy Agency's Task 21: Daylight in Buildings.

The significance of the research lies in the development of sky modelling strategies for the whole sky spectrum under real conditions and their applicability world-wide by providing a process in which local sky models can be created in computerised terms.

Specific objectives to attain this aim are:

(a) The development of new ideas for a common and consistent representation of existing sky luminance models and their definition in computational terms for sky model comparisons and the creation of new sky luminance models

(b) The development of a methodology for the inclusion of an efficient representational format for use with existing computer software and/or software packages.

(c) The designing and implementation of an interactive computer system to demonstrate the use of the models.

(d) The comparison and validation of existing analytical sky models using measured data and the representational format.

(e) The displaying and testing of the representational format in proven internationally used computer software such as DOE-2 and evaluating the output.

2.0 Main Results

A new approach to modelling sky luminance is defined which can be used to accurately represent the luminance characteristics of the sky under all sky conditions. The approach consists of a Standard Digital Form (SDF) describing a sky's luminance distribution in an efficient form with little loss of information content. The SDF can be accessed through a set of functions which allows comparisons of the luminance distributions of various sky models and their manipulation and integration into simulation packages.

The purpose of the comparison studies are to prove the feasibility of integrating these functions representing sky luminance models into an internationally used simulation package such as DOE-2. The International Energy Agency Task 21: Daylight in Buildings will be incorporating this Standard Digital Form (SDF) in their work program to improve and develop new computer software packages calculating daylight input into buildings to predict building energy performance such as the ADELINE package.

The physical results of the project are also embodied in a series of C-language functions which provide the tools for the implementation of the ideas and their integration into third party software systems. These libraries of functions are available to researchers for further development and/or testing and possible integration into other software packages. For research and academic purposes no fees are payable. Commercial use of the software is possible under agreed commercial terms.

3.0 Work Program Description

To carry out the project objectives the work program consisted of the following:

(a) Determining a range of new modelling strategies for the luminance distributions across the sky dome to cater for local conditions worldwide by surveying existing sky luminance models.

(b) Determining an efficient representational format and defining the semantics for representation both as a data structure and as a permanent file.

(c) Providing a set of functions to allow a user to read, create and manipulate the format to represent local sky luminance distributions.

(d) The development of an interactive computer package (SKYMAP) to demonstrate the use of the underlying modelling concepts and to facilitate the construction of sky models.

(e) Validating the models by comparing the SDF of the standard CIE clear and overcast sky models with empirical representations of these CIE skies in existing proven computer software and examining and evaluating the differences.

4.0 Determination of Future Research Directions

The feasibility of integrating Standard Digital Form functions into a simulation package has been proven. However the definition of edges to define discontinuities in the luminance distribution due to clouds and other obstructions to the sky dome is a problem yet to be solved. This can be accomplished manually by tracing the edges of clouds or obstructions on a screen from displayed sky models or photographs and placing the results into an SDF model before contours are computed.

There is the need for generating more precise methods for the assessment of cloud characteristics, their spatial distribution and the depiction of cloud edges which are better carried out automatically by the development of new automated instrumentation and procedures for sky imaging, for example as demonstrated by Davies, Griggs and Sullivan [Davies et al, 1992]. Various filtering methods to isolate cloud components, particularly edges from clear sky and direct solar images are now being investigated in an ARC research project 'Automated sky luminance and cloud cover estimation for improved sky modelling in different climate zones'.

TECHNICAL REPORT

1.0 Introduction and Overview

1.1 Introduction

It was concluded in the ARC project 'The development and verification of sky models which represent the predominant skies in Australia for the prediction of daylight in buildings' that appropriate models representing local sky conditions are required if daylighting is to become a predictable quantity in the design of energy efficient buildings or if sky models are to be developed for specific purposes such as simple models that can be used at sketch design stage to assess daylight quantity and quality in interiors; or more detailed models giving half-hourly descriptions of the sky luminances for research purposes or for the development of electric lighting control systems in buildings.

Research class measurement stations in the International Daylight Measurement Program are beginning to provide data on real sky luminance distributions and as a result sky luminance distributions such as the CIE Overcast Sky can be re-assessed against measured results. The quantity of data available is producing a profusion of sky models with claimed high accuracy for particular locations. However it is necessary to take into account the statistical nature of the daylight climate and the end purposes for which the sky models are being measured or developed before appropriate models can be determined for daylight design in buildings.

This project's objective is to develop new ideas for a consistent format for sky model representation and its definition in computational terms to encompass all possible sky models to enable comparison and manipulation. Daylighting designs can then be tested against a standardised sky luminance format rather than the current criterion of illuminance. The determination of daylight availability based on sky luminances can be used to produce luminance patterns in daylit interiors. The use of luminance rather than illuminance will produce a means to assess glare and the quality of the visual environment which is currently impossible with current illuminance design techniques.

1.2 Overview

To attain the objectives the project consists of several tasks:

(a) A survey of existing sky luminance models (Section 2)

(b) A common representation of existing sky luminance models which will allow comparison, manipulation and the construction of new and appropriate sky luminance models for local conditions (Section 3)

(c) A method to access a standard representation through a set of functions (Section 4)

(d) The transferring of empirical and measured sky luminance models to the representational format by contouring and interpolation (Section 5)

(e) Defining modelling operators to manipulate and transform sky luminance models (Section 6)

(f) Displaying sky models using the interactive SKYMAP system (Section 7)

(g) Using SDF operators to build localised sky models (Section 8)

(h) Macro programming to define new sky luminance models and to extract luminance properties of sky models. (Section 9)

(i) Validation of the new modelling methods by integration of the SDF functions into the DOE-2 program and undertaking a series of daylighting computations (Section 10).

The Appendices contain the range of basic formulae used in the project and the formal definitions (manual pages) for the SDF functions.

2.0Sky Models and Modelling

.c2.2.1 Introduction

The sun provides the primary source of radiant energy for most of our human activity. The diffusing influence of the earth's atmosphere not only influences the quantity of this radiant energy, but also its distribution as it impacts the surface of the earth. This project is concerned with a particular facet of the sun's radiated energy - the diffuse sky component which appears as visible light ie. the energy spectrum falling between 400 and 700 nanometres wavelength. This visible component is of particular interest to the designers of buildings in a number of ways, for example:

- The study of the quantity of natural daylight penetrating buildings through windows and skylights for the purpose of ensuring adequate levels of interior lighting. The required levels of interior lighting will vary with the tasks being performed, and often must be designed to meet health and safety requirements.

- The design of artificial lighting, and associated control mechanisms to reduce the use of artificial lighting by accounting for the availability and dynamics of natural light to conserve energy and cost.

- The simulation of the quality of interior lighting to assist in the design of interior spaces (colours and luminance contrasts).

Daylight consists of the visible direct radiation from the sun or sunlight and the diffuse radiation from the sky or skylight. As the sunlight penetrates the earth's atmosphere a proportion is diffused by the atmosphere via a process of refraction and absorption due to molecules and particulate matter. In heavily overcast days, little direct radiation arrives at the surface of the earth - most is diffuse radiation. Even with clear sky conditions, the diffuse component contributes to the natural light that penetrates windows or skylights. While direct beam penetration will have a significant effect, it is often limited by sun screening devices as it can cause glare (ie. high levels of contrast) which is generally considered undesirable in interior lighting.

The intensity of the skylight varies with the movement of the sun during the day and year and with variations in atmospheric turbidity which are due to the climate and factors such as cloud type and cover. For the calculation of sky luminances, the sky is envisaged as an imaginary hemispherical dome placed directly above the location being studied, with the

surface having a luminance distribution equivalent to the diffuse light being filtered through the atmosphere. This project focuses on the determination of sky luminance distributions and examines ways of modelling these luminance distributions over the sky dome excluding the direct beam component.

The impact of the various atmospheric elements on the sky luminance distribution can be complex. Global composition effects such as turbidity can be modelled reasonably well but the impact of non-uniform cloud conditions (which is the norm), as well as other obstructions on the horizon (eg. adjoining buildings and trees) are problems which need to be solved in the daylighting design of buildings. Clouds of various types (and altitudes) can cause both increased and decreased luminance levels from patches of the sky dome. Some cloud patches can be brighter or duller than the background sky depending on cloud type and location relative to solar position. This means that the daylight available at a window is influenced significantly by the orientation of the window relative to the position of the patches of cloud. Even fully overcast sky conditions have their own characteristic variations and are rarely uniform.

Another major practical influence on the availability of daylight is the effect of physical obstructions. Most buildings in urban areas are located adjacent to others. This means that some parts of the sky dome are hidden from view. In addition, there will be reflected light from the surfaces of the adjoining buildings which, depending on the surface characteristics, may also impact on the light penetrating a window. For windows near ground level, local vegetation can also reduce the amount of light available from the sky dome.

An approach to estimating daylight illumination and luminance distributions within an interior space taking into account obstructions can involve the use of scale model photometry. While modelling the window and room are not difficult, modelling the sky is more problematical. An innovative sky simulator recently constructed in Switzerland is comprised of a computer controlled heliodon rotating about a sky simulator consisting of a partial hemisphere of specified variable luminance [Scartezzini, 1994]. A video camera with probe is used to determine interior luminance patterns within the scale model. While this approach has potential, it is first necessary to know the sky luminance distributions for a particular location before these can be programmed on the simulator.

In the computation of the quantity and quality of daylight in an interior, knowledge is also required on the optical properties of the window glazing and the interior surface reflectances, as the total light received at a reference point within a room is a combination of the internal and external reflections as well as the sky component seen through the window. The calculation processes of the daylight components reflected from external

ground surfaces and internal room surfaces are not included in this project which is concerned solely with the source of the daylight ie the sky luminance (excluding the direct beam component) and its distribution over the sky dome. There are specialized computer software for this purpose eg. DOE-2 [Winkelmann and Selkowitz, 1985, Winkelmann et al, 1993] and SUPERLITE.

2.2 Standard sky models

To enable analytical studies of the daylighting in buildings it is necessary to have a means of representing the luminance distribution of the sky dome. The modelling of sky luminance distributions has been carried out for several years with most attempts assuming standard sky conditions which can be modelled using an analytical function. In practice, however, sky luminance distributions are far more complex than models represented by empirical formulae.

The empirical sky models most commonly used are the Commission Internationale de L'Eclairage (CIE) Standard Clear and Overcast Skies representing the extremes ie. clear and fully overcast sky conditions. However they are only very loosely related to measured sky luminance models and unrelated to any meteorological data of cloud cover or type, or sunshine duration recorded at a particular site. Although these standards are not advanced sufficiently to suit the future needs for computer-aided design or expert systems involving energy conscious design, they are considered adequate for some clear and overcast conditions, particularly in Europe, although they are not always appropriate in other locations in tropical and subtropical climate zones. However these CIE models provide a reference for this project as they can be mathematically defined.

The Standard Clear Sky [CIE, 1973] represents the luminance distribution for a perfectly clear (cloudless) sky based on known (or assumed) values for the atmospheric moisture content and turbidity. The model is based on the theoretical diffusion characteristics of the earth's atmosphere.

The Standard Overcast Sky [CIE, 1990] defines the luminance distribution for a fully, and uniformly, overcast sky with stratus type clouds. In this case the luminance peaks at the zenith. The relative luminance distribution has a 1 to 3 horizon to zenith ratio and only varies with the altitude of the part of sky of concern. The CIE has also standardised a modification (1 to 2 ratio) for areas with high ground reflection such as snow.

A full description of these skies is given in the Appendix.

2.3 Other sky models

The CIE standard skies refer to the extremes of cloudiness conditions. Problems when using the CIE Standard Overcast as a design tool have generated the need to consider other intermediate skies or partly cloudy skies between these extremes. A number of approaches have been taken. Either the two extremes of overcast and clear have been weighted by some meteorological or luminous factor eg. a cloud ratio [Harrison, 1991], a diffuse ratio [Matsuura, 1987] or a simplified theoretical approach has been attempted. Some examples, which rely on empirically determined constants are:

The BRE Average Sky [Littlefair, 1981] assumes a uniform diffuse background with a direct solar and a circumsolar component making use of luminance data measured by Wegner [Wegner, 1975]. It has been modified to fit south-east England and was developed to be suitable for indoor daylight calculation. It is based on measurements of a wide range of real skies.

The Mean sky [Nakamura and Oki, 1986] is composed of three basic skies including an intermediate sky [Nakamura, Oki and Iwata, 1987] with absolute illuminance values. The all weather distribution is obtained by linear combinations of these three skies

The BRE average sky and the mean sky by Nakamura, Oki et al. are useful for some limited purposes such as the estimation of energy savings or prediction of the average interior luminous environment. However these skies can be used only in restricted parts of the world.

The Perez All-Weather Model [Perez et al, 1992] is derived from the CIE Clear Sky, but includes the facility to control the luminance distribution through a set of three parameters to reflect the local insolation conditions (solar elevation, sky clearness, and brightness). These are influenced by the ratio of normal to diffuse incident radiation.

The Harrison Model [Harrison, 1991] includes two luminance distribution profiles, overcast and clear. The general profile is a linear combination based on opaque cloud cover and the solar zenith angle. The opaque cloud cover was measured by the University of Calgary's cloud detector.

The Kittler Homogeneous Sky [Kittler, 1987] requires an assessment of the illuminance turbidity coefficient which can be derived from direct illuminance data [Navvab et al, 1984]. It simulates the diffusion characteristics of a real sky by an equivalent homogeneous sky and

has had reported success in modelling resultant horizontal illuminances particularly for the extreme conditions.

All these models have reported some success in modelling daylight in particular locations in the world [Marland, 1991]. Recently, a database of irradiance measurements has been established to compare various luminance models [Perez et al, 1991] but with varying results. In the Perez comparison studies the input consisted of global and direct (diffuse) irradiance. Those models using illuminance eg. the Kittler and Harrison models were linked to irradiance by a luminous efficacy ratio creating a possible source of error. The relative failure of a number of partly cloudy and "all weather" models to accurately predict sky luminance conditions is due to the assumption of the form of linear interpolation between extreme conditions. These cannot be expected to model accurately the circumsolar region nor the dynamic luminances of a partly cloudy sky. In addition, a more consistent and flexible format is required to compare one model against another.

2.4 Sky models from measured luminance data

Actual knowledge of sky luminances comes from measured data recorded by sky scanners at a fixed set of points over a short duration of time. For example the PRC Krochmann scanner [Krochmann, 1991] completes the task of scanning 145 data points in about 20 secs and at each data point records the average luminance over a patch of sky subtending an angle of 10 degrees. Measured data from this type of scanner have indicated that sky luminance patterns vary considerably depending on location, solar position and atmospheric conditions [Hayman and Stevens, 1992]. Results have also placed some doubt on the use of the CIE standard models in some parts of the world. But long term records from scanners, such as at Sydney Airport [Hayman el al, 1993] allow statistical analyses which can provide information of the effects of different types of sky conditions on sky luminances and their distributional properties. Such data are essential as an aid to the development of design models for a particular location.

While scanning devices give reasonably good records of sky luminance patterns, the resolution is insufficient to define and isolate the discontinuities often present (eg. edges of clouds or other obstructions) and these devices are expensive to purchase and difficult to maintain in operation. This project proposes ways in which sky models can be developed for daylighting design with minimal calls on expensive recording devices.

2.5 The sky modelling problem

For a specific location a means of modelling the local sky conditions is needed. Since measured data are not available for every location, effective methods for using existing knowledge need to be determined such as the use of existing empirical models and any available measured data which are relevant to the target location. The problem is how to devise suitable models for a particular sky condition.

In addition, a design model may be needed for different purposes. It may be needed to represent a set of conditions which can be used for further analysis to achieve design specifications eg. to devise bright or dull skies for the analysis of interior daylight levels and/or the qualities of interior light or rapidly varying skies to test the performance of control devices which adjust the levels of artificial lighting levels to meet design standards for work place illuminance.

A desirable characteristic of a sky luminance model is the ability to provide estimates of the luminance at some position (ie. altitude and azimuth) on the sky dome relative to the orientation of the window at some time of day and location on the earth's surface. New methods are proposed in this project aimed at solving these problems and to assist in the design of sky models which are suitable for locations where the standard models are not directly applicable.

The methods include:

- Designing a modelling strategy which allows models to be specified to suit local conditions taking into account the varying properties of time, solar altitude, latitude, and longitude.

- Allowing any data source (eg. any of the empirical models, or data from a scanner) to be integrated into a single modelling strategy.

- Providing a vehicle which will produce the required results of luminance at a specified azimuth and altitude for a given time of day.

- Allowing for the output from the process to be fed directly into simulation packages which can use these data.

Assuming that the formal output from a sky model is only of practical use when integrated into a simulation package no attempt has been made to realise the results in a traditional

way, but rather to define them in terms of a set of functions which are intended to be integrated into other software packages.

The proposed modelling framework is therefore based on the definition and realization of a set of programmed functions which provide the basic facilities for a user (programmer) to access them, use them, and provide the results directly into other simulation packages. This approach has the distinct advantage of ensuring that the implementations are correct. Most empirical models defined in the literature are poorly described, or they often contain typographical errors which make implementation difficult if not unreliable. Our approach better assures correctness without the problems of transcribing complex formulae.

2.6 Conclusions

The sky modelling problem has been approached as a design problem. The process has aimed at the development of a format which accommodates the constraints of a limited knowledge of local sky conditions. The task of developing a sky model for any location world-wide requires an integration and synthesis of all available knowledge about that locality and hence a single sky model with an input of fixed parameters and an output of luminance values is not possible. Rather, the aim has been to provide a set of tools which can be used to assist in the development of a more accurate interpretation from known data.

3.0 New Modelling Strategies

3.1 Model Types

Models of the luminance distribution of the sky have to date been defined in the form of analogue expressions which describe the luminance at any point on the sky dome. These have been derived from various theoretical assumptions pertaining to the diffusion characteristics of the atmosphere validated with limited measured data. The resulting empirical relationships provide an estimate of the luminance distribution with some level of reliability or accuracy. Some examples are outlined in Appendix A. While such models may be defined for a given location their use at other locations is, arguably, unreliable.

Models of this kind are classified as *Analogue Empirical Models* (AEMs). An AEM is an empirically derived expression composed of a set of parameters and/or coefficients, which allow the luminance of any point on the sky dome to be computed. This type of model is very convenient for computational purposes as input comprises only the necessary parameters (including location, time, solar altitude and atmospheric conditions) and calibration coefficients.

Given the renewed interest in modelling sky conditions through the International Daylight Measurement Program, we also must include the *Measured Data Model* (MDM). MDMs are defined by sets of data resulting from various types of measurements which allow the luminance levels from various parts of the sky dome to be described. Typically these measurements are made with a scanning device which can record the luminance at a set of (well defined) points on the sky dome. From these data the luminance distribution on the sky dome can be determined.

One of the available scanner devices is manufactured by PRC Krochmann GMBH of Berlin. This device will measure luminance values on a 12° grid on the sky dome with a 10° acceptance angle in a period of about 20 seconds. This gives a grid of 145 "points". Various filters can also be used to sample selected parts of the visible spectrum.

MDMs naturally provide "good" quality data for the location of the measuring device and for the time at which the measurements were recorded. In addition these data may be used to validate, calibrate or design AEMs which may be suitable for the location. It is also possible to use MDMs directly in simulation programs, where the "real" luminance data can be used to study the lighting conditions, or the behaviour of lighting control devices, over periods of time. In this case the MDM can naturally account for turbidity and cloudiness

during a "real" period of time for the locality. MDMs may also be used to define the luminance characteristics for sky simulators used with physical scale models.

Given that both these forms of model (AEM and MDM) have their own advantages a modelling approach which can use information from both these model forms may be considered to be rather useful. We will adopt this approach.

A third modelling strategy can also be proposed. The *Hybrid Data Model* (HDM) will facilitate the construction of (new) artificial models which can be derived from other models, especially AEMs and MDMs. For example a sky model could be constructed from a selected AEM, scaled using data from local measurements, with cloud cover and type included to represent local sky conditions. The resulting model is thus a hybrid of the MDM and AEM models, combined to represent the local conditions.

Each of these model types has its own particular value and area of application. AEMs are very convenient for computation purposes, but generally lack the realism necessary to represent real local conditions. MDMs can provide accurate estimates for specific locations and times, but are more expensive to obtain and somewhat complex to process. HDMs offer the promise of building locally specific models based on more general models, or on models derived from other locations.

3.2 Design Requirements

As a way of facilitating the representation and manipulation of each of these model forms in an integrated and consistent way, a common representation scheme is proposed which will allow:

- A common description scheme to cover all model forms and origins.
- Models derived from different data sources to be directly compared.
- Models to be manipulated using a common set of tools.
- The definition of a set of operators to take data from models of any type and to construct new models.
- Any form of model to be used for the common analysis tasks (daylight analysis, simulation studies, etc).

The luminance distribution across the sky dome is a complex phenomenon and not appropriately represented by uni-dimensional statistics such as mean, standard deviation and other measures of dispersion or variance. The representation scheme must therefore include the spatial component in a formal way.

For relatively uniform sky conditions (eg CIE clear and overcast skies) various AEMs have been shown to represent luminance distributions reasonably well [Hayman et al, 1993]. For partly cloudy sky conditions the problem is more complex as the luminance distribution becomes much less uniform, and even discontinuous due to the effects of clouds and other horizon obstructions to the sky dome.

A representation format is therefore proposed which has the following properties:

- Allows the distribution of luminance to be described at varying levels of detail and resolution.

- Facilitates the representation of discontinuities due to effects of clouds and other obstructions to the sky dome.

- Minimises the loss of information in the representation so that the original spatially variant data can be recovered with acceptable precision.

We now have a set of design goals for our new approach to sky luminance modelling.

3.3 Model Forms

The simplest way of achieving these goals may be through a grid-based model, by recording the luminance values at a set of grid points located over the sky dome. There, however, are several major problems with this approach.

The accuracy of a grid based model depends on the resolution of the grid (spacing of grid points). To increase the accuracy of the model, the size of the grid spacing must be decreased. This means that the total number of grid points will grow with the square of the grid size. Secondly, a grid generally has to be uniform[1], which means that regions of rapidly varying luminance with be represented with less accuracy than regions with little variation. Grid-based schemes cannot take direct account of discontinuities in the luminance

[1]This is not strictly true, but it would add considerably to the complexity of the resulting computations.

distribution caused by clouds and other obstructions to the sky dome. Discontinuities can only be inferred from rapidly changing data values.

In theory increasing the grid resolution will solve most of these problems but at the expense of rapidly increasing data complexity.

In an attempt to avoid these problems we have devised a new approach which records the luminance data as sets of *iso-luminance contours* in the sky dome, combined with *edges* to represent discontinuities in luminance. The iso-luminance contours define a simple contour map of the luminance distribution with the contour spacing chosen to suit the required resolution. Each contour is taken as a polyline which can be defined by a value, and a set of co-ordinate pairs. The iso-luminance contour can be open or closed.

To represent discontinuities in the luminance distribution edges are introduced. Edges are assumed to be closed polygons. Edges will provide information for the interpolation scheme which is designed to account for the presence of edges. To enable reasonable interpolations to be made near the edge of the sky dome (the horizon) an edge is also located around the perimeter. This will allow a consistent interpolation scheme to be proposed across the full sky dome.

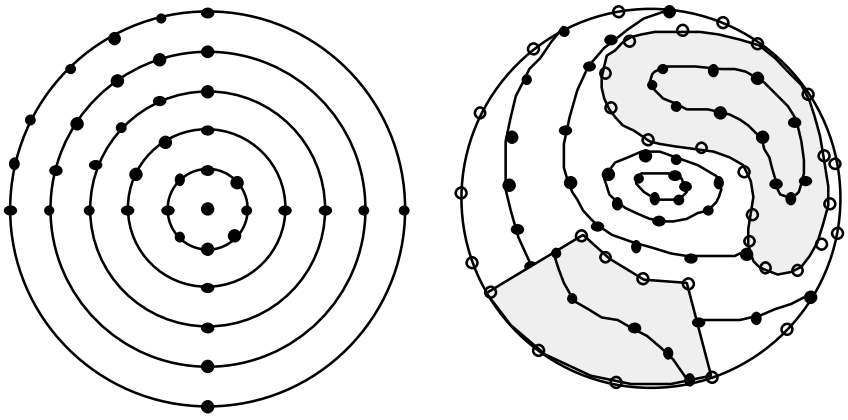

Figure 3.1: (a) Grid-based representation, (b) Contour-based representation

In Figure 3.1(b) the shaded areas represent areas of obstruction due to cloud and local building development. The • denote vertex locations on iso-luminance contours (note that in Figure 3.1(a) not all the grid is shown). The ◦ marks denote vertex locations on the edges in Figure 3.1(b).

A regular grid scheme shown in Figure 3.1(a) might be used to represent this luminance distribution, this will require approximately 84 vertices (ie azimuth and altitude values).

The contour scheme in Figure 3(b) requires about 66 vertices (made up from 33 vertices on iso-luminance contours, and 33 vertices on lines of discontinuity, or edges).

In the case of the grid scheme, to improve the accuracy of the model we must reduce the size of the grid interval. If we reduce it to half then the number of vertices will quadruple (ie $O(n^2)$). In this case 84 vertices will increase to about 336.

In the case of the contour scheme, we can increase the resolution by halving the contour interval. We could also consider decreasing the interval of vertices along the contour, but the value of this will depend on how the original intervals were chosen. If we also halved the interval then we would increase the total number of vertices by $O(n^2)$. This would not offer the advantages we are looking for.

We will see later in this report that the process by which the iso-luminance contours are computed (and thus the vertex locations) is such that it ensures that the interval between vertices is related to the curvature of the contour. Iso-luminance contours that are relatively straight will have more widely spaced vertices than more highly curved contours. This is exactly what we need. We will argue, therefore, that with such a scheme the vertex spacing along the iso-luminance contour is optimized and thus little improvement in resolution will be achieved by further reduction of interval. The number of contour intervals will, however, have a significant effect. It would appear that beyond some number (eg 16) contour intervals the improvements in accuracy are small. If we keep the same curvature limits on the contours and edges, then the number of vertices will therefore, at most, double (ie. O(n)) if we halve the contour interval.

From experience we know that a model for a single sky requires 2-3,000 bytes of data to give an acceptably accurate representation, with typically 16 contour intervals.

Within an SDF we thus have two classes of polylines:

Iso-luminance contours.

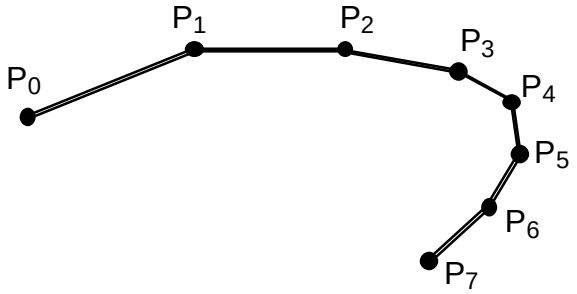

Figure 3.2: Iso-luminance Polylines

To fully define a contour we need to have its value (luminance level in Lux), and a set of co-ordinate pairs, one for each of the vertices (labelled P_0 to P_7 in Figure 3.2). These co-ordinates will be taken to be in polar co-ordinates, by azimuth and altitude (θ and ϕ), which define a unique position on the sky dome. The conventions used are as follows:

Azimuth:	0°	- true north
	90°	- east
	180°	- south
	range of valid values 0 to 180°	
Altitude:	0°	- at horizon
	90°	- zenith
	180°	- Diametric horizon.

These conventions allow a position to be defined by four 8-bit bytes. Both azimuth and altitude being defined by a degree and minute value in the ranges 0 to 180° and 0 to 59' respectively.

Edges:

Edges describe lines of discontinuity and are assumed to be closed polygons. This means that an edge has a well define inside and outside as shown in Figure 3.3.

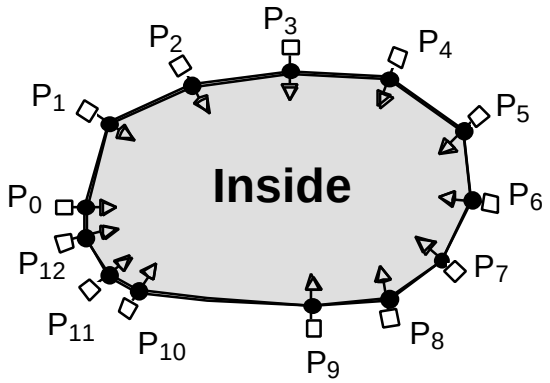

Figure 3.3: Edge Polygons

In this case each vertex is defined by a set of (θ, ϕ, I_i, I_o) quadruples. At each vertex we record the luminance value on the inside of the edge (I_i), as indicated by the symbol in Figure 3.3, as well as the one on the outside (I_o) as indicated by the symbol.

The combinations of iso-luminance contours and edges with inside-outside luminance values allows us considerable flexibility to interpolate with some confidence over the whole sky dome, including right up to the horizon edge as well as other edges in the model. The interpolation problem is shown schematically in Figure 3.4. Without the knowledge of a discontinuity, the contouring interpolation scheme can only choose those neighbourhood points which appear relevant (eg closest) as in Figure 3.4(a). When the edge is included, the interpolation can be controlled to contain only those points on the same side of the edge. We would thus expect to see results like as shown in Figure 3.4(b), with surface a discontinuity along the edge.

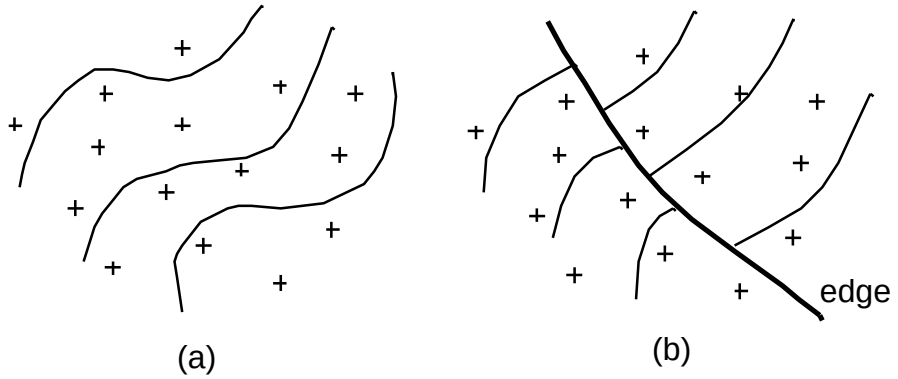

Figure 3.4: (a) Contour interpolation without the edge,
(b) Contour interpolation with the edge.

The proposed modelling framework is thus a digital representation in the sense that the continuous variation of luminance is represented by a set of discrete data values, and we will refer to this as the *Standard Digital Form* (SDF).

3.4 Modelling Data Structures for SDF

A complete sky model requires further information in addition to the iso-luminance and edge information if it is to be useful. For example we must record the location and time information for the sky model. We may also wish to record other data (equipment details, global irradiation data, data provider, etc) which might be available from the original source (whether AEM, MDM or HDM). We may also wish to perform some analyses and retain the results within the model itself (eg average luminance, cloud cover). The SDF therefore facilitates a general scheme for storing *parameters* and their *values*. A parameter is a named object which can be defined and inserted into a SDF model, along with its associated data items.

It would be usual to expect parameters like "site" (to include data items of longitude°, longitude', latitude°, latitude' and reference longitude°) and "time" (to include data items of

year, month, day, hour and minutes) to be present since these are necessary to fully define a sky model. Other parameters can be user-defined and added as required. These might be, for example, cloud cover, radiation data, average luminance data, sun position and so on.

The iso-luminance contours and edges are also included in the SDF model as parameters. These parameters are naturally used to compute user-requested analyses (eg luminance at a point, average luminance in a region of sky) which are obtained through specially prepared functions.

The SDF model therefore is composed of sets of parameters. Each one is defined by its name and a set of data items. In general we can visualise the parameter description in the form:

parameter name
$item_1$
$item_2$
:
:
$item_n$

There are no limitations on the number of data items for a parameter. Parameter items can be one two or mutli-byte data types, depending on the range of values required. Some typical examples of parameter descriptions are:

site
longitude
latitude
ref. longitude

time
year
month
day
hours
minutes

sun
azimuth degrees
azimuth minutes
altitude degrees
altitude minutes

Parameters of this type enable us to define the context of a particular sky model. As described above, iso-luminance contours and edges are also defined in terms of parameters, as follows:

iso	
size	
r	level value
e	size
p	$(azimuth, altitude)_1$
e	$(azimuth, altitude)_2$
a	:
t	$(azimuth, altitude)_n$

edge	
size	
r	type code
e	size
p	$(internal\ value, external\ value, azimuth, altitude)_1$
e	$(internal\ value, external\ value, azimuth, altitude)_2$
a	:
t	$(internal\ value, external\ value, azimuth, altitude)_n$

These definitions for iso-luminance contours and the edges allow multiple contours/edges to be defined within a single parameter list.

A more complete list, and description, of the commonly used and built-in parameters is given in Appendix B.

We can now therefore define an SDF model as a collection of parameters of these and other user-defined data types. The user of an SDF model does not need to be concerned with the details as the realization of the model is typically hidden from the user. The user will interface to the model through a set of pre-defined functions which will handle all the detail and ensure the correctness of the data and the resulting luminance computations.

User defined parameters are definable using similar types of data structures. The user is provided with the tools to define functions of the type exemplified by the*site*, *sun* and *time* parameters shown above. Facilities for this will be described later.

As defined, an SDF model can (should) contain at least:

- a site parameter
- a time parameter
- a non-empty set of iso parameters
- a non-empty set of edge parameters (note that we will always insert an edge around the horizon to facilitate interpolation near this boundary).

This information is sufficient to uniquely define a single sky model. Typically, however, we will require a sequence of models to represent the sky conditions over a day, several days or even a longer period to be accessed by appropriate simulation packages. We can easily extend our modelling concepts to include multiple models within a single data structure as shown below:

sdf
$model_1$
$model_2$
:
:
$model_n$

The complete data structure is intended to be realized in a database, and initially in the form of a simple sequential file. While this format is less efficient (than a random access file structure) for locating a particular model from the set of models, most interest lies in accessing models in a sequential fashion. This is certainly the case for simulation purposes where the models for a particular location are required in time sequence. Assuming that the file of models is created in the correct sequence then little is lost in accessing them sequentially. An SDF model file may, if required, contain just a single model

4.0 The SDF Access Functions

4.1 Accessing SDF Models

Access to SDF models is exclusively through a set of formally defined functions that can be used within a computer program. This allows a well defined (and hopefully correct) set of tools to be provided to the user at a level which can be directly linked into a simulation package or some other user-written software package.

The user, therefore, does not need to have very much concern about the details of the data structures and how they are implemented and managed. Our functions have been written in the C-language, but can readily be linked to FORTRAN coded programs.

The relationships between the SDF data, the SDF access functions and the user's program is shown in Figure 4.1.

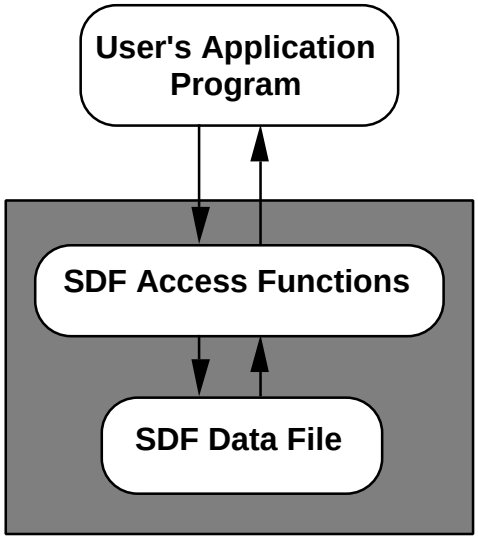

Figure 4.1: Relationships of SDF Date Files, Access Functions and the User's Program

We have implemented several classes (and levels) of SDF functions to facilitate the needs of users. These functions are summarized below.

4.2 Model Manipulation Functions

These functions provide high level access to SDF files, SDF models within files and conversions to SDF from other standard models which have been provided. In the

following brief descriptions, the semantics of the function parameters can be inferred from their names. A more complete description of the data types in included in Appendix C.

ƒ1. To open an existing file containing SDF models, or to create a new file intended to contain SFD models:

```
FileDesc = sdfOpenFile(Filename,Mode)
```

ƒ2. To close the file:

```
sdfCloseFile(FileDesc)
```

ƒ3. To create a new model:

```
ModelDesc = sdfCreateModel()
```

ƒ4. To load the next model from an open file into memory:

```
ModelDesc = sdfLoadModel(FileDesc)
```

ƒ5. To save a model from memory and append to a currently open file:

```
sdfStoreModel(ModelDesc,FileDesc)
```

ƒ6. To delete a model from memory:

```
sdfFreeModel(ModelDesc)
```

ƒ7. To import a model of a particular kind (AEM or MDM) from a file, or a hard coded definition, and convert into SDF:

```
ModelDesc = sdfImport(ModelType,ModelDesc,MatchFunc,FileDesc)
```

where the ModelType parameter will be the predefined name of one of the available model types.

4.3 Model Parameter Functions

These functions provide access to individual model parameters, to create new parameters and to insert them into models. Note that iso-luminance and edge type parameters are not directly accessible in this way. They are only available via the analysis functions, see below.

ƒ8. To add a parameter to a model:

```
Exists = sdfSetParameter(ModelDesc,ParDesc,Format,P1,P2,...)
```

ƒ9. To extract a parameter from a model:

```
Exists = sdfGetParameter(ModelDesc,ParDesc,Format,P1,P2,...)
```

ƒ10. To delete a parameter from a model:

```
Exists = sdfDeleteParameter(ModelDesc,ParDesc)
```

4.4 Analysis Functions

The analysis functions provide the basic capabilities for computing luminance levels at some point, or region, in the sky dome from the SDF model.

ƒ11. To calculate the luminance at a point:

```
Lum = sdfPointLuminance(ModelDesc,&Point)
```

ƒ12. To calculate the average luminance of a patch of sky defined by an arbitrary polygon patch:

```
TotalLum = sdfAreaLuminance(ModelDesc,PointList,&Patch_area)
```

This suite of functions thus provides most, if not all, of the tools that a user would require to establish, manipulate and use the SDF modelling approach.

4.5 User Libraries

These functions are formally provided in two libraries which can be linked with a user's program to provide the complete functionality required. The user's program can thus be quite simple since the library functions handle all the detailed conversions and memory

management operations. The data type definitions are contained in appropriate header files which must also be included by the user.

To demonstrate the steps required we now present some simple programs which demonstrate the use of the library functions. To begin with, the Makefile (below) sets out the linking process using the two libraries *sdf* and *sdfops*. These contain all the pre-defined SDF functions that the programmer needs to access and are fully described in Appendix C.

```
CC = /opt/local/bin/gcc -fpcc-struct-return
CDEBUGFLAGS = -O2
SDFINCDIR = /home/skymap/include
SDFLIBDIR = /home/skymap/lib
CFLAGS = $(CDEBUGFLAGS) -I$(SDFINCDIR)
FFLAGS = $(CDEBUGFLAGS)
LDFLAGS = -L$(SDFLIBDIR)
LDLIBS =  -lsdf -lsdfops -lm

OBJS = test1.o

test1:  $(OBJS)
        $(RM) $@
        $(CC) -o $@ $(OBJS) $(CDEBUGFLAGS) $(LDFLAGS) $(LDLIBS)

clean:
        $(RM) test1 *.o core make.log
```

Note that the name of the file containing the program is *test1.c* in this Makefile.

Program 1: To access an AEM and extract the luminance at a specificed point on the sky dome.

The basic steps are:

(i) create a sky model (*template*) in which we will specify the time and location required.
(ii) set the time and site parameters in the template.
(iii) import a CIE_CLEAR sky model, assigning it to a new model (*model*).
(iv) specify the required point as an azimuth and altitude in the *point* variable.
(v) compute the luminance in the model (*model*) and print out the result.
(vi) collect the garbage.

```
#include <sdf.h>
#include <sdfops.h>

struct _point point;
double lumin;

int match ( Model template, Model model)
{
   return(1);
}

void main()
{
    template = sdfCreateModel();
    (void) sdfSetParameter(template,"site","%d%c%d%c%d",
                                  151,12,-33,54, 150);
    (void) sdfSetParameter(template,"time","%c%c%c%c%c",
                                       92, 7, 7, 12, 45);

    model = sdfImport(CIE_CLEAR, template, match, (FILE *)NULL);
    point.azimuth = RADIANS(0);
    point.altitude = RADIANS(45);
    lumin = sdfPointLuminance(model,&point);
    printf ("Luminance value = %f \n", lumin);

    sdfFreeModel(model);
    sdfFreeModel(template);
}
```

The *match* function, in this case, always returns true value since the required model is always found by the *sdfImport* function. We will present an example later where the real purpose of this function is demonstrated.

Program 2: To access an AEM and extract the average luminance over a patch of the sky dome.

This program is based on the above example, just extending it to compute the average luminance over a patch of sky. The patch is defined by a linked-list of points (each containing the appropriate *azimuth* and *altitude* values). The function *sdfAreaLuminance* also computes the area of the patch which is given as a proportion of the surface area of a unit sphere, ie surface area of the sky dome (a hemisphere) is 2p.

```
#include <sdf.h>
#include <sdfops.h>

Model model, template;
struct _point point;
Point  pointlist, pointt;
double lumin, area, totallum;
int i, ret_average;
double azlist[13] = { 0, 30, 60, 90, 120, 150, 180, 30, 60, 90,
                                    120, 150, 0 },
       altlist[13] = { 60, 60, 60, 60, 60, 60, 60, 120, 120,
                                    120, 120, 120, 60};

int match ( Model template, Model model)
{
   return(1);
}


void main()
{
    template = sdfCreateModel();
    (void) sdfSetParameter(template,"site","%d%c%d%c%d",
                                          151,12,-33,54, 150);
    (void) sdfSetParameter(template,"time","%c%c%c%c%c",
                                          92, 7, 7, 12, 45);

    model = sdfImport(CIE_CLEAR, template, match, (FILE *)NULL);
     pointlist = (Point) Malloc(sizeof(struct _point));
     pointt = pointlist;
     pointt->azimuth = RADIANS(azlist[0]);
     pointt->altitude = RADIANS(altlist[0]);
   for (i =1; i<13; i++)
    {
     pointt->next = (Point) Malloc(sizeof(struct _point));
     pointt = pointt->next;
     pointt->azimuth = RADIANS(azlist[i]);
     pointt->altitude = RADIANS(altlist[i]);
     pointt->next = NULL;
    }
    totallum = sdfAreaLuminance(model, pointlist, &area);
    printf("Average lumin = %f , area = %f \n", totallum/area,area);

    sdfFreeModel(model);
    sdfFreeModel(template);
}
```

Program 3: To access an AEM, and save it in SDF together with a new parameter containing the average luminance over the whole sky.

The third example extends the above example by inserting a new parameter ("avlum") into the newly created sky model (*model*) and then saves the complete model in SDF as a file *test1.sdf*. This file thus contains an SDF model with the luminance data of the selected CIE_CLEAR sky as well as a parameter which can be retrieved (see code) as required.

```
#include <sdf.h>
#include <sdfops.h>

Model model, template;
struct _point point;
Point  pointlist, pointt;
double lumin, area, totallum;
int i, ret_average;
FILE   *output;
double azlist[13] = { 0, 30, 60, 90, 120, 150, 180, 30, 60, 90,
                              120, 150, 0 },
       altlist[13] = { 60, 60, 60, 60, 60, 60, 60, 120, 120,
                              120, 120, 120, 60};

int match ( Model template, Model model)
{
   return(1);
}


void main()
{
    template = sdfCreateModel();
    (void)  sdfSetParameter(template,"site","%d%c%d%c%d",  151,12,-33,54,
150);
    (void)  sdfSetParameter(template,"time","%c%c%c%c%c",  92,  7,  7,  12,
45);

    model = sdfImport(CIE_CLEAR, template, match, (FILE *)NULL);
     pointlist = (Point) Malloc(sizeof(struct _point));
     pointt = pointlist;
     pointt->azimuth = RADIANS(azlist[0]);
     pointt->altitude = RADIANS(altlist[0]);
   for (i =1; i<13; i++)
    {
     pointt->next = (Point) Malloc(sizeof(struct _point));
     pointt = pointt->next;
     pointt->azimuth = RADIANS(azlist[i]);
     pointt->altitude = RADIANS(altlist[i]);
     pointt->next = NULL;
    }
    totallum = sdfAreaLuminance(model, pointlist, &area);
    printf("Average lumin = %f , area = %f \n", totallum/area,area);

    (void) sdfSetParameter(model,"avlum","%d",(int)totallum/area);
    (void) sdfGetParameter(model,"avlum","%d", &ret_average);
```

```
    printf("returned average = %d \n",ret_average);

    output = sdfOpenFile("test1.sdf","w");
    sdfStoreModel(model,output);
    sdfCloseFile(output);
    sdfFreeModel(model);
    sdfFreeModel(template);

}
```

To enable development and testing, as well as providing an operational tool to assist in sky model development we have implemented these functions in an interactive modelling package (SKYMAP). This package provides a full set of capabilities to manipulate and visualize SDF models as well as generating SDF versions of a range of common AEMs. Facilities are also provided to import data from a PRC scanner data file, as well as a text formatted file. and create SDF models. A full description of the SKYMAP package is described later in this report.

5.0 Contouring and Interpolation

5.1 The Contouring Task

Our modelling framework assumes that all forms of sky models are converted into SDF from where we can establish, and use, a common set of manipulation and analysis functions. Models of types AEM and MDM must therefore be first converted to SDF before they are accessible. This conversion process requires the necessary iso-luminance contours and edges to be defined.

AEM models provide a continuous description of the luminance distribution across the sky dome. The transformation of an AEM therefore requires a suitable contouring process to be applied to a sufficiently large set of points where the luminance values are calculated from the appropriate algebraic expression. Since there are (generally) no discontinuities in AEMs no internal edges are added, only that one around the horizon is included to facilitate later interpolation in regions near the horizon.

The transformation of an MDM also requires a contouring analysis from the set of data points defined in the MDM. PRC data sets contain measured luminance values at 145 uniformly spaced points across the sky dome. Other MDMs may contain less points. For example it is perfectly feasible to have MDMs defined from a much smaller number of recordings (perhaps from a hand held luminance meter). It is also possible to have MDMs derived from digital CCD images which may have up to 1000x1000 pixel values. MDMs may also contain edge information (in addition to the horizon edge). If cloud edges can be identified in the original data, or if the horizon obstructions are in well defined locations, then this information can be included in the contouring computation.

The contouring problem is essentially the same for both AEMs and MDMs. It is really all to do with choosing the best set of points for contour interpolation and the associated interpolation processes.

5.2 Constructing the Surface Model

A complete set of contours is intended to provide an approximation to the complete and continuous surface (of luminance intensity in our case). The surface model is constructed from a set of points at which the luminance values are known. Clearly the accuracy of the resulting surface model will depend on the number of points. For an AEM we must choose a sufficient number of points to provide the required accuracy. For a MDM we are limited

to the data supplied (eg 145 points from the PRC Krochmann scanner). It is our experience, however, that it is not necessary to use a very large number of points from AEMs as little increase in representational accuracy is achieved once we go beyond this number of points (ie 145). We have thus chosen to use a regular grid of about 180 points for representing the AEM models in SDF. This grid is based on the 145 point grid proposed by Tregenza [Tregenza, 1987], plus a set of edge points defining the horizon edge.

The first step in constructing a surface model is to group the discrete sample points into neighbourhoods. This is done to determine the area of influence of each point. The definition of a neighbourhood will be determined in part by the type of model chosen to represent the surface. Various surface models are possible, from flat surface patches to curved surfaces which allow higher order surface continuities to be maintained. While these latter models may offer increased accuracy we have not considered it worthwhile at this stage to add the extra complexity. We have thus chosen to stay with planar patches.

For the case of planar patches, the most suitable grouping is in triplets, since only three non-linear points are needed to define a unique plane. The process of grouping the sample points into these triplets is known as triangulation, as it results in a tessellation of triangles that completely span the sample space. While there are any number of ways in which this triangulation can be performed, the resultant properties of the tessellation will affect the accuracy of the surface model. The further apart the vertices of any triangle, the less accurate the surface patch will be. So, in order to ensure the minimum error, we need a tessellation consisting as nearly as possible, of small equilateral triangles. This can be achieved by using the Delaunay triangulation. The Delaunay triangulation has the constraint that the circumcircle of every triangle is empty. That is, no circle passing through the three vertices of a triangle will contain any other sample points. This constraint imposes several desirable properties on the triangulation. Firstly, and most importantly, it avoids the formation of long, thin triangles, ensuring that each datum will be close to its neighbourhood. It fact, it has been shown by Lawson [Lawson, 1977] that the Delaunay triangulation is locally equiangular. Also, except for a few isolated cases, the Delaunay triangulation for a given set of points is unique.

There exists many algorithms for constructing a Delaunay triangulation for an arbitrary set of points [eg Cline and Renka, 1984, Lawson, 1977 and Watson 1981]. Also, algorithms which calculated the Voronoi diagram [Fortune, 1987, Green and Sibson, 1978], which is a geometric transformation of the Delaunay triangulation, may be considered. However most of these algorithms require that the convex hull for the set of points is calculated first. This can be computationally expensive for large data sets and has been shown by Sloan [Sloan,

1991] to be unnecessary. The algorithm proposed by Sloan can also be easily extended to include constrained edges.

The initial step requires the definition of a super-equilateraltriangle which is guaranteed to enclose all data points, like as shown in Figure 5.1. In this case the data space is shown normalized into a unit circular space.

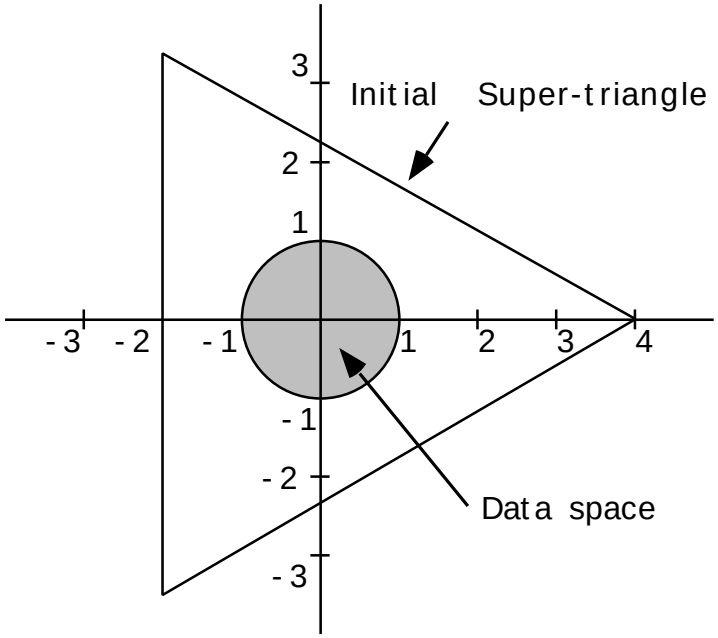

Figure 5.1: The Super-triangle surrounding the normalised data space

A variation of the Sloan algorithm has been implemented, and is given here in point form:

1. Construct a "super-triangle" guaranteed to contain all sample points (as in Figure 5.1).

2. Add a point to the triangulation (P).*

(a) Determine which existing triangle contains the point (as in Figure 5.2).

(b) Remove this triangle and create three new triangles by joining the new point to each of the vertices of the old triangle.

(c) Add these new triangles to a stack.

3. Restore the Delaunay constraint.

(a) Remove a triangle from the stack.

(b) Check that the circumcircle of this triangle is empty.

(c) If not, swap the diagonals of the quadrilateral formed by this triangle and the

point within the circumcircle, and add the two newly formed triangles to the stack.

(d) Repeat (a) - (c) until the stack is empty.

4. *Repeat (2) - (3) until all points have been added.*
5. *Remove any external triangle, that is, any triangle which shares a vertex with the super-triangle.*

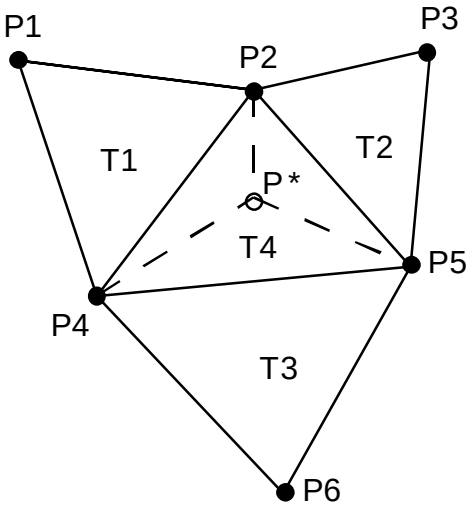

Figure 5.2: Adding New Triangles

Each step of the algorithm deserves consideration. The super-triangle is most easily formed by normalising the input data. This ensures that a standard triangle can be used. For our implementation, the super-triangle is defined by the three vertices (-2, 2√3), (-2, -2 √3) and (4, 0). This is an equilateral triangle with its circumcentre at the origin (see Figure 5.1).

To add a new point to the triangulation, we need first to find which existing triangle contains it. For a given point P, and an arbitrary triangle (ABC) from the triangulation, we can determine whether the point lies inside the triangle or, if outside, in which direction we need to move. This is done by calculating the barycentric co-ordinates for the point P with respect to the triangle (ABC) as shown by the following equations.

$$a_0 = (y_A - y_P)(x_B - x_P) - (x_A - x_P)(y_B - y_P)$$
$$a_1 = (y_B - y_P)(x_C - x_P) - (x_B - x_P)(y_C - y_P)$$
$$a_2 = (y_C - y_P)(x_A - x_P) - (x_C - x_P)(y_A - y_P)$$

Using this information, we can determine the surrounding triangle with the following algorithm.

1. *Choose an arbitrary triangle from the triangulation.*
2. *Calculate the barycentric coordinates for the point relative to the current triangle.*
3. *If all the coordinates are negative, then the point lies inside the current triangle.*
4. *Otherwise, the largest positive value indicates the direction to P (a_0 -> AB, a_1 -> BC, a_2 -> AC).*
5. *Repeat from step 2, using the adjacent triangle which shares the edge calculated above.*

Three new triangles are formed by joining the point P to each of the vertices of the existing triangle ΔABC. These triangles, labelled ΔPAB, ΔPBC, and ΔPCA, are then tested against their neighbours to ensure that their relative circumcircles are empty. If not, then the diagonal of the quadrilateral formed by the offending triangle and its neighbour are swapped. This swapping process continues until the Delaunay constraint is restored. When all of the points have been included in the triangulation, the outer triangles, that is any triangle which shares a vertex with the super-triangle, are removed from the list of triangles.

The information required by this algorithm is a list of points and triangles. The points are given as inputs to the triangulation routine and provide 3D Cartesian coordinates, that is, a 2D location and an associated value. For each triangle we need to store the three vertices which define it, as well as pointers to the three adjacent triangles. By this, we mean the triangles that share a common edge and, by implication, two vertices with the current triangle. This information is stored in the following C data structures.

```
typedef struct _point {
    struct _point       *next;
    double              value,
                        external,
                        azimuth,
                        altitude,
                        x,y;
} *Point;

typedef struct _triangle {
    struct _triangle    *next;
    struct _triangle    *adj[3];
    Point               vertex[3];
} *Triangle;
```

This information can easily be updated as each new point is added to the triangulation, provided it is done in a consistent manner. The vertices of a triangle are defined from an arbitrary starting point and proceed counter-clockwise around the perimeter. The adjacent

triangles are numbered from the first edge, which lies between the first and second vertices, and also proceed counter-clockwise.

The most significant step in the triangulation process is the swapping test, as this ensures the Delaunay constraint is maintained. To achieve this, we use Lawson's Swapping Algorithm [Lawson, 1977] together with Cline and Renka's circumcircle test [Cline and Renka, 1984] which provides better numerical stability under some conditions.

The triangulation process has also been modified to facilitate the inclusion of edges. In the above data structure we see that the vertex (luminance) values are contained in each triangular surface patch. This allows the values of adjacent patches to be different for the same vertex point. We thus can represent the discontinuity within the triangulation. All that is required is to ensure that each constrained edge corresponds to a triangle edge. This is done by applying the swapping algorithm to remove any triangles which intersect the edge, and then restoring the Delaunay constraint on the newly formed triangles.

5.2 Surface Approximation

Once we have constructed the Delaunay triangulation for a set of input points, we have an approximation of the surface. The problem with this form of surface model is that it is difficult and expensive to store in a meaningful way. To overcome this problem, we construct a set of linear[2] spaced contours from the triangulation. There are several advantages to this approach. Firstly, it is efficient in terms of storage, as for each contour we need only to store the single value associated with it, and a list of control points which define its shape. The number of contours can be varied to achieve the desired level of accuracy with an minimum amount of additional information. Another advantage to generating these contours is that there will naturally be more information where the surface is changing rapidly, and less where it is flat.

We can outline the algorithm as follows:

1. *Find the maximum value from the sample points.*
2. *Determine the contour interval by dividing the maximum value by the desired number of contours.*

[2]Nonlinearly spaced contours have the potential to better represent surfaces with large peak values, but also make "reading" the surface more complex. We have not taken this step.

3. *For each contour level, starting with the lowest, determine which triangles lie below, across or above the current level.*
   (a) *All of the triangles that lie completely below the current level are removed from the triangulation.*
   (b) *All of the triangles that lie across the current level are added to a stack.*
   (c) *All of the triangles that lie completely above the current level are ignored (ie neither deleted nor added to the stack).*
4. *Determine one point (of the two) of intersection between an edge of the triangle on the top of the stack and the current level.*
5. *If the triangle adjacent to the current triangle, which shares the intersecting edge, is on the stack, remove it and determine the other point of intersection between this triangle and the current contour level. Append this point to the current contour.*
6. *Repeat step (5) until the next triangle is not in the stack.*
7. *Repeat steps (5) & (6) in the opposite direction.*
8. *Repeat steps (4), (5) & (6) until the stack is empty.*
9. *Repeat step (3) for each contour level.*

It can be seen from this algorithm that all contour control points will lie on the edge of a triangle from the tessellation. This highlights another advantage of the Delaunay triangulation, as the production of near equiangular triangles will reduce the number of control points in the resulting contours. These points will generally tend closer to their neighbours only as the curvature of the contour increases. The benefit of this result is similar to that of using contours to store the surface, in that it reduces the amount of information to be stored without effecting the accuracy of the data.

5.4 Reconstructing the Surface

The first stage of this process is to define what we mean by a continuous surface. Obviously, we are not going to produce the physical surface itself, but a model with sufficient detail to include information about every point on the surface. What we require is the height (luminance) of the surface for any given location (azimuth and altitude). Rather than using a single function to define the surface, we will use a series of different functions to describe different regions of the surface.

Since our input data is in the form of a set of isoluminance contours, we will use these contours to produce a set of planar patches to describe the surface. Since each contour is defined by a polyline of (vertex,value) pairs it is possible to construct a triangular patched surface to approximate the original (whether AEM or MDM). We use the same Delaunay

triangulation process to ensure that we end up with well conditioned (close to equilateral) triangles as shown in Figure 5.3.

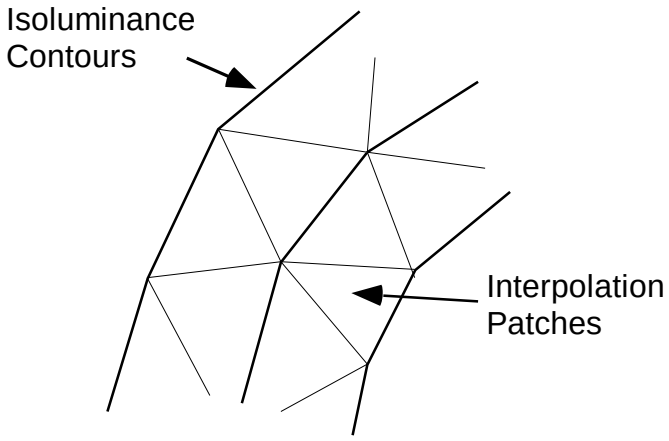

Figure 5.3: Re-triangulation for Interpolation

Now that we are able to produce a surface composed of triangular planar patches, we need to determine the coefficients of each planar equation. Given that on each triangle the three points A, B and C with coordinates (x_A,y_A), (x_B,y_B) and (x_C,y_C), and luminance values f_A, f_B and f_C respectively, define a triangular patch, we can determine the unknown coefficients of triangular plane by the solving the following set of simultaneous equations:

$$f_A = a_0 + x_A a_1 + y_A a_2$$
$$f_B = a_0 + x_B a_1 + y_B a_2$$
$$f_C = a_0 + x_C a_1 + y_C a_2$$

for the coefficients, a_0, a_1 and a_2. Once the triangular surface patch in which our point is contained is found, then it is straight forward to compute its value (luminance) for the actual azimuth (x) and altitude (y) using a function like those shown above.

6.0 Modelling Operators

6.1 Using Sky Models

Once in SDF we can now define a set of operators which will allow us to manipulate and transform sky models. It is through these functions that a user can devise and build new sky models as well as accessing data from existing models. There are obvious questions that can be answered by comparing two sky models, or designing a new one based on two, or more, standard models.

Comparison of sky models can be done statistically (average, median, standard deviations, etc of the luminance distributions), but these statistics explain little of the spatial distribution effects of differences. For example, one model may provide higher levels of luminance in certain regions of the sky dome, lower in other regions, but a similar level of average luminance.

New (hybrid) models are often used in simulation computations, based on linear combinations of clear and overcast sky models. It would seem rather useful to know what type of sky these new models actually define. It is suspected that this analysis is not often done due to the complications of visualising the result.

The modelling operators to be introduced here are intended to provide a suite of tools that will allow these types (and others) of operations to be easily performed and, given an appropriate visualisation system (eg. SKYMAP, to be discussed later), can be easily displayed and compared.

The transformation processes included in the modelling operators are generally speaking quite straight forward, only complicated by the fact that we are dealing with a spatially defined data sets. It is appropriate, therefore, to formally define the desired operators so that their meaning is clear.

6.2 Operating on SDF Models

Operations on two models are defined to be carried out on a union of the two SDF models. In practical terms this means that for any two models (say A and B) we need to ensure that the transformation operators are applied at a spatial resolution that reflects the information content of the models separately and collectively. This has been achieved by creating a new set of vertices as the union of the vertices of the polylines from both models.

If M^A and M^B represent the two models with each containing a set of polylines:

$$M^A = \{P^A\}$$

$$M^B = \{P^B\}$$

where $\{P^A\}$ is a set of polylines (each of which is a vector of vertices),

then union is then defined by:

$$\{P^{A \approx B}\} = \{P^A\} \approx \{P^B\}$$

The union set will thus contain a total number of vertices (at which the transformation operator is applied) equal to the sum of the vertices in M^A and M^B, as shown in Figure 6.1.

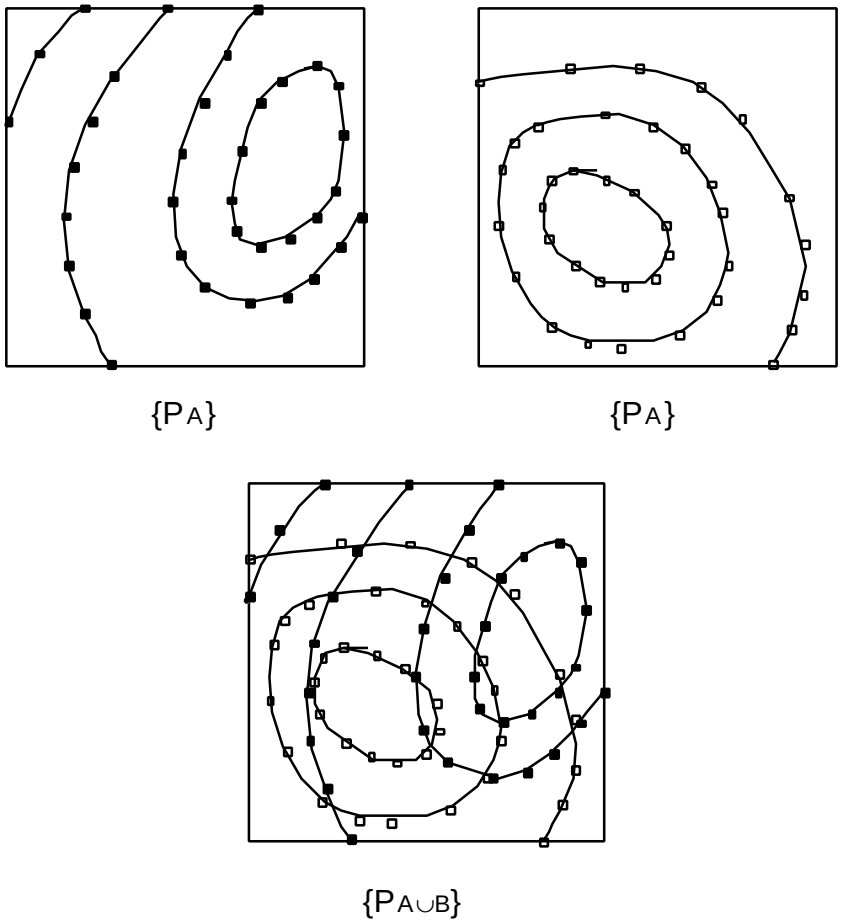

Figure 6.1: The Union Operation to define the set of vertices for binary operators.

For binary operators, $\{P^{A\approx B}\}$ defines the set of vertices at which the operator is applied. For unary operators applied to M^A , then $\{P^A\}$ defines the set of appropriate points. The transformation will thus not involve any significant loss of information.

The set of edges that may exist in a model is defined by {E}. Edges define regions of the sky dome as closed polylines. Edges are used to define discontinuities in the luminance distribution. Here we assume that {E} excludes E^o (the edge inserted around the horizon for the purposes of interpolating luminance values near the horizon.

6.3 Standard SDF Operators

The following family of operators has been defined for use on SDF models:

(a) Addition ($\overset{\smile}{+}$):

The addition operator simply adds together the luminance values at each vertex in $\{P^{A\approx B}\}$. Therefore:

$$M^A \overset{\smile}{+} M^B = I^A_{f,q} + I^B_{f,q}, \quad \forall\, f,q \in \{P^{A\cup B}\}$$

Where θ and ϕ refer to the azimuth and altitude angles, respectively for each vertex in $\{P^{A\approx B}\}$, and *I* is the luminance value at the respective vertex.

The addition operator is used to construct a new model by adding together two model components as is often done in forming a composite model, for example to build an average sky by combining CIE clear and overcast models in some predefined proportions.

(b) Subtraction ($\overset{\smile}{-}$):

The subtraction operator is complementary to the addition operator and is defined by:

$$M^A \overset{\smile}{-} M^B = I^A_{f,q} - I^B_{f,q}, \quad \forall\, f,q \in \{P^{A\cup B}\}$$

This operator is useful for comparing two models where we are interested in examining the absolute difference between two models.

(c) Ratio ($\dot{\div}$):

The ratio operator allows the ratio of two models to be evaluated, again a useful tool for comparing two models. It is defined by:

$$M^A \dot{\div} M^B = \frac{I^A_{f,q}}{I^B_{f,q}}, \quad \forall f,q \in \{P^{A\cup B}\}$$

(d) Product ($\dot{\times}$):

The product operator allows us to modify the luminance distribution according to some distribution function (which is in effect another model). It is defined by:

$$M^A \dot{\times} M^B = I^A_{f,q} \times I^B_{f,q}, \quad \forall f,q \in \{P^{A\cup B}\}$$

This operator allows us to build a model (say M^A) which contains a function which we wish to use to scale the other model (M^B). The scaling function may be derived from other model operations, for example by taking the ratios of two models or from measured data.

(e) Normalise ($\tilde{n}\bullet$):

Normalised sky models are often used for reference and comparison purposes. They are created by scaling all luminance values so that the value at the zenith is unity. The normalise operator is defined by:

$$\tilde{n} \bullet M^A = \frac{I^A_{f,q}}{I^A_{90,0}}, \quad \forall f,q \in \{P^A\}$$

(f) Scale ($f\bullet$):

The scale operator is a convenience function to enable models to be scaled by constant factors. It is defined by;

$$f \bullet M^A = f \times I^A_{f,q}, \quad \forall f,q \in \{P^A\}$$

A typical use of this function would be to assist in building a model from some linear combination of CIE clear and overcast skies.

(g) ExtractPatch (∅):

The presence of cloud, or other obstructions, in the sky dome naturally suggests that we would like to remove or add patches representing these obstructions. A patch is described by an *edge*. Once an edge is defined (computed in some way or drawn interactively by the user), then we can extract that bit of the sky inside the edge with the ExtractPatch operator. If we define a set of edges by *{E}* each of which is actually a closed polyline across the sky dome, then the ExtractPatch operator is define by:

$$M^A \rightarrow \{E^A\} = I^A_{f,q}, \quad \forall\, f,q \in \left[\{P^A\} \text{✋} \{E^A\}\right]$$

(h) ExtractRest ( ):

The ExtractRest operator is complementary to ExtractPatch in that it extracts that part of the sky outside the edge, $\{E\}$ It is defined by:

$$M^A \Rightarrow \{E^A\} = I^A_{f,q}, \quad \forall\, f,q \in \left[\{P^A\} \text{✋} \neg\{E^A\}\right]$$

(i) Overlay ( )

The overlay operation allows the contents of one model to replace the contents of another in an overlay operation. If the first model contains no edges then the result will simply be the first model as it will cover the second model completely. If the First model contains an edge and no luminance values exist either inside or outside the edged, then the result will be a combination of the first model and the second model. The luminance values of the second model will show through where no values exist in the first model. Formally therefore:

$$M^A \Downarrow M^B = M^A \quad if\ \{E^A\} = \varnothing$$

$$M^A \Downarrow M^B = M^A \rightarrow \{E^A\} \text{ ✞ } M^B \Rightarrow \{E^A\} \quad if\ M^A \Rightarrow \{E^A\} = \varnothing$$

$$M^A \Downarrow M^B = M^A \Rightarrow \{E^A\} \text{ ✞ } M^B \rightarrow \{E^A\} \quad if\ M^A \rightarrow \{E^A\} = \varnothing$$

6.4 Operator Function Descriptions

Each of the above operators have been implemented as a C-language function which are summarized as follows. More complete descriptions are given in Appendix C.

f 13. To construct a new model which is the sum ($\dot{+}$) of two models:

```
ModelDesc = sdfSum(ModelDesc1,ModelDesc2)
```

f 14. To construct a new model which is the difference ($\dot{-}$) of two models:

```
ModelDesc = sdfDifference(ModelDesc1,ModelDesc2)
```

f 15. To construct a new model which is the ratio ($\dot{\div}$) of two models:

```
ModelDesc = sdfRatio(ModelDesc1,ModelDesc2)
```

f 16. To construct a new model which is the product ($\dot{\times}$) of two models:

```
ModelDesc = sdfProduct(ModelDesc1,ModelDesc2)
```

f 17. To normalize (*ñ*•) a given model so that the zenith luminance is unity (1.0):

```
ModelDesc = sdfNormalize(ModelDesc)
```

f 18. To scale (*f*•) a given model by a constant factor:

```
ModelDesc = sdfScale(ModelDesc,ScaleFactor)
```

f 19. To construct a new model which contains only that data *inside* (∅) the defined edges:

```
ModelDesc = sdfExtractPatch(ModelDesc)
```

f 20. To construct a new model which contains only that data *outside* ( ) the defined edges:

```
ModelDesc = sdfExtractRest(ModelDesc)
```

f 21. To perform an opaque overlay of one model on top of another ( ). Wherever the luminance is defined in the first model, it will appear in the result, otherwise the information from the second model will be used.

```
ModelDesc = sdfOverlay(ModelDesc1,ModelDesc2)
```

A programming example using these operators is given in Section 7. We will first look at the interactive environment of SKYMAP which includes these operators and makes most of them available for interactive use.

7.0 The SKYMAP Interactive Modelling Package

7.1 Functional Overview

We have implemented a complete interactive package as a vehicle for the development and testing of the above SDF functions and operators as well as providing an operational tool to actually view and manipulate sky models to assist in designing and building new hybrid models. The SKYMAP system is an interactive application for the manipulation and display of sky models of various forms.

The basic functionality of SKYMAP includes:

(a) The ability to access a number of the standard empirical models (AEMs) and to convert these into SDF. Currently those implemented include: CIE Clear Sky, Overcast Sky , CIE Intermediate Sky, Perez All-weather Model, Harrison Sky, the BRE Average Sky, and the Nakamura Intermediate sky.

Since AEM models are defined from a set of algebraic equations and appropriate coefficients, they must be hard coded. Other empirical models can be readily added, once their respective expressions and parameters are fully defined. See an example in Appendix E.

(b) The ability to access data (MDMs) from scanning devices (currently the PRC at Mascot Airport, Sydney, [Hayman et al, 1993], and similarly formatted data from other recording sites) and to convert this to SDF.

(c) The ability to input data from a simply formatted text file containing luminance data for a set of azimuth-altitude points and convert the result into SDF.

(d) The ability to access hybrid models (HDMs) which have been previously constructed using the system. A hybrid model is one constructed from either a AEM or MDM, or some combination of both, with the intention of representing some particular local conditions.

(e) The ability to display these models in either contour or fully shaded presentaton forms.

(f) The ability to perform the range of operations on these models and to save the results as new HDMs for later use, or for use in other software packages.

The general structure of the complete system is shown in Figure 7.1. SKYMAP has been built using the standard SDF libraries as described earlier. The complete SKYMAP system provides a fully functional and highly interactive tool for the analysis and construction of sky models. It operates within an X-windows environment running under UNIX.

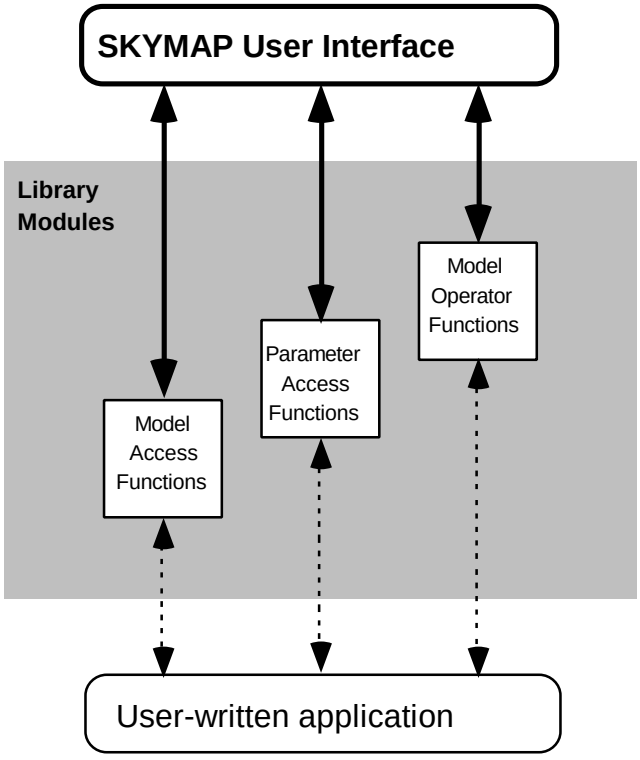

Figure 7.1: Structure of SKYMAP System

7.2 Operation of the SKYMAP System

SKYMAP has been developed for the UNIX/Xwindows environment and is thus relatively transportable within this software domain. The user interface adopts the Xwindows style using readily available widgets to facilitate various user interface operations.

The system operates from a control panel as shown in Figure 7.2. The options included in this control panel allow the user to:

- select the type of model to be displayed
- specify the location, date and time for the displayed model
- display the model(s)

- perform various operations on the displayed model(s)

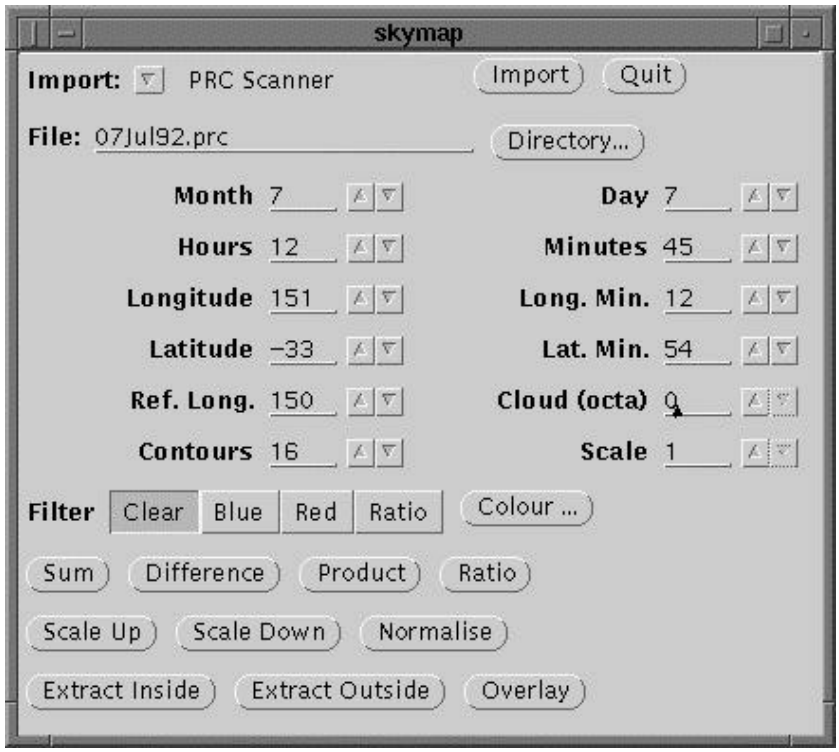

Figure 7.2: SKYMAP Conrol Panel

In detail, the various items in the control panel are:

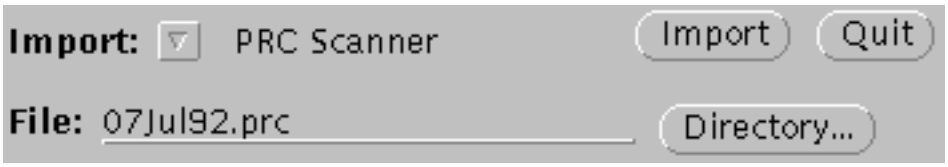

Figure 7.3: Model Selection Options

(i) *Import menu:* Enable the selection of the type of model to be imported to be specified. Options include:

PRC data files (MDM)
SDF files
CIE Clear sky (AEM)
CIE Overcast sky (AEM)
CIE Intermediate (AEM)
Perez All Weather model (AEM)
Harrison model (AEM)
Nakamura Intermediate sky model (AEM)

Text file for simple MDMs
Photograph (digitized in PPM format)

In each case (except the photograph) the importing process converts the source-model to SDF for display and any further manipulation.

The photograph is just displayed as a bitmap and not converted at this stage. Its prime use is to be able to directly compare the actual sky to the results from the scanner, or one of the AEM models. If the photographic image was callibrated for luminance then it could also be converted to SDF.

Photographs are static images and cannot be edited in SKYMAP, other standard Xwindows utilities (eg. *xv*) can be used for this purpose.

The text file option is intended for use by users who need to create their own data formats for MDMs. The format for this model type is given in Appendix D.

(ii) *File field:* This field specifies the name (and location) of the file to be read. The name is typed directly into this field. By clicking on the *Directory* button the user can select the required file from a directory listing and, as necessary, change to the required directory location.

(iii) *Import button:* Initiates the import process. Depending on model type various fields in the model parameter section (see below) may also have to be correctly filled out. If the import is successful, the sky model appears in a model window, as seen in Figure 7.4 The details and operations of these model windows is discussed later.

(iv) *Quit button:* Terminates SKYMAP.

The parameter section contains a number of editable fields which should be set before importing a new model. In each case the required value can be typed directly into the field, or by clicking on the *up* an *down* arrow buttons to increase or decrease the value by a unit value (see Figure 7.5).

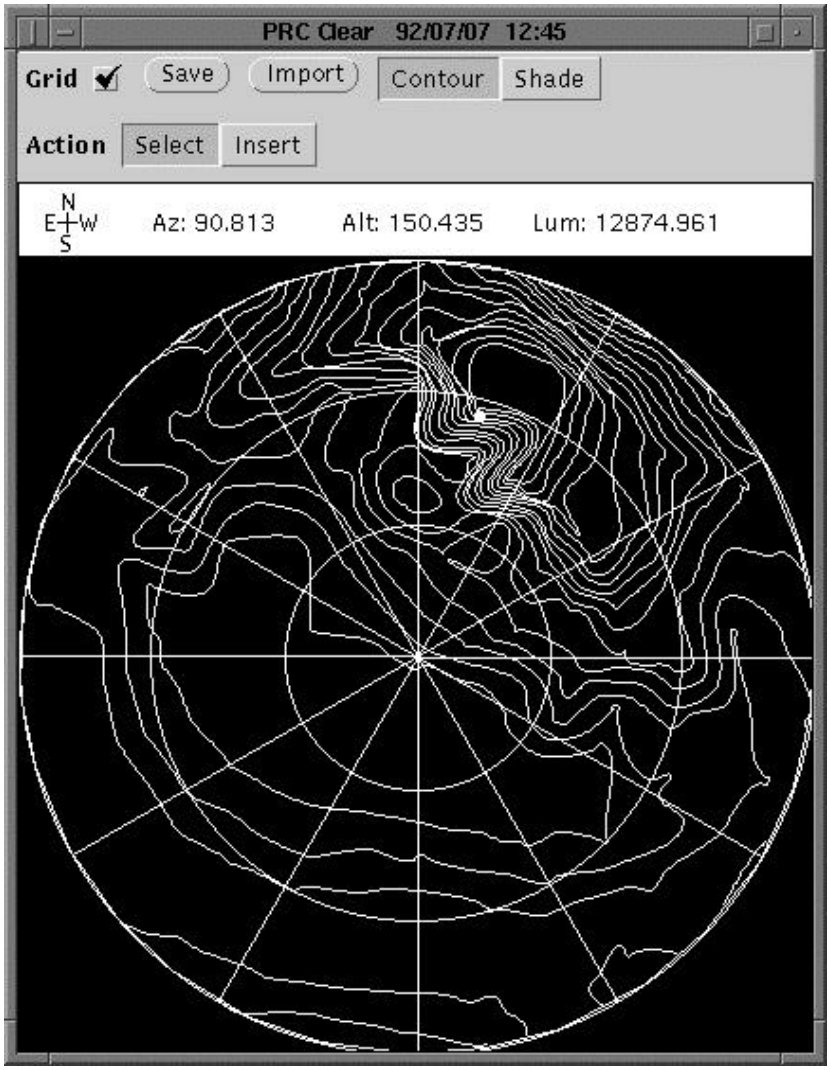

Figure 7.4: Example Model Window Display

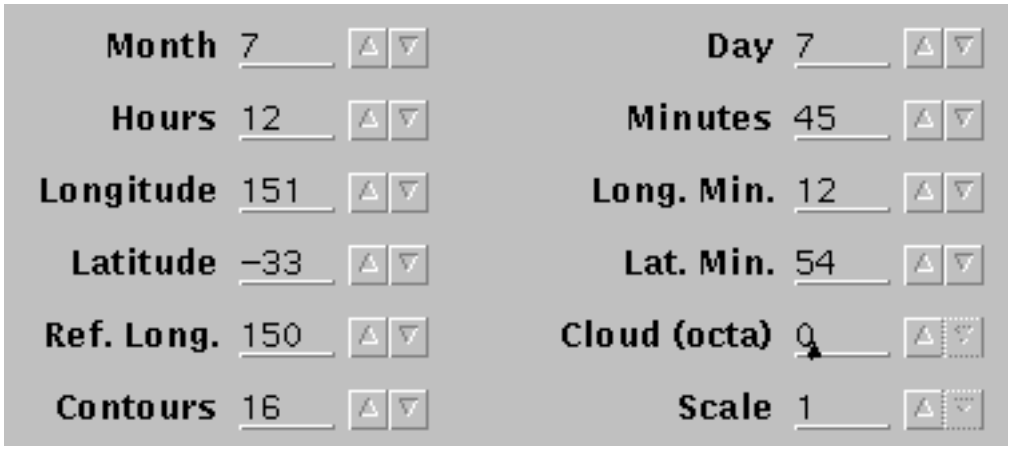

Figure 7.5: Parameter Specification Options

(v) *Month field:* Sets the month as a number in the range 1 to 12

(vi) *Day field:* Sets the day number in the month (1 to 28, 29, 30 or 31 depending on the month).

(vii) *Hours field:* Sets the hour value for the time (24 hour clock)

(viii) *Minutes:* Sets the minutes value for the time (0 to 59).

(ix) *Longitude:* Sets the longitude (degrees) of the location. East of Greenwich is positive.

(x) *Long. Min:* Sets the longitude minutes value (0 to 59).

(xi) *Latitude:* Sets the latitude (degrees) of the location. South of the equator is negative.

(xii) *Lat. Min:* Sets the latitude minutes value (0 to 59)

(xiii) *Ref. Long:* Sets the reference longitude for the local time zone for normal clock time. No allowance for summer time is included.

(xiv) *Cloud (octa):* Cloud cover in octas. This required for Harrison model only

(xv) *Contours:* Number of contour intervals to be defined for SDF models. The default is 16. Experience has shown that little improvement in accuracy is obtained by increasing this number. The contour intervals are equally spaced over the expected range of luminance values (see shading profile below)

(xvi) *Scale:* This is a scale factor which is used by the *Scale Up* and *Scale Down* operations described below.

(xvii) *Filter switch:* These buttons (Figure 7.6) are used when importing PRC data files where the PRC scanner has recorded luminance values through three filters. *Clear* button refers to a neutral filter with V(λ) matching characteristics. The *Blue* and *Red* buttons refer to filters with blue and red transmission characteristics which are normally recorded

immediately following the initial scan. Choosing the relevant button selects the data with the required filter.

Figure 7.6: Filter Selection and Shading Controls

The *Ratio* button is not currently in operation.

(xviii) *Colour* This button enable the user to set (and modify) the shading characteristics that will be used for shading sky models. Clicking this button brings up the shading control panel as in Figure 7.7)

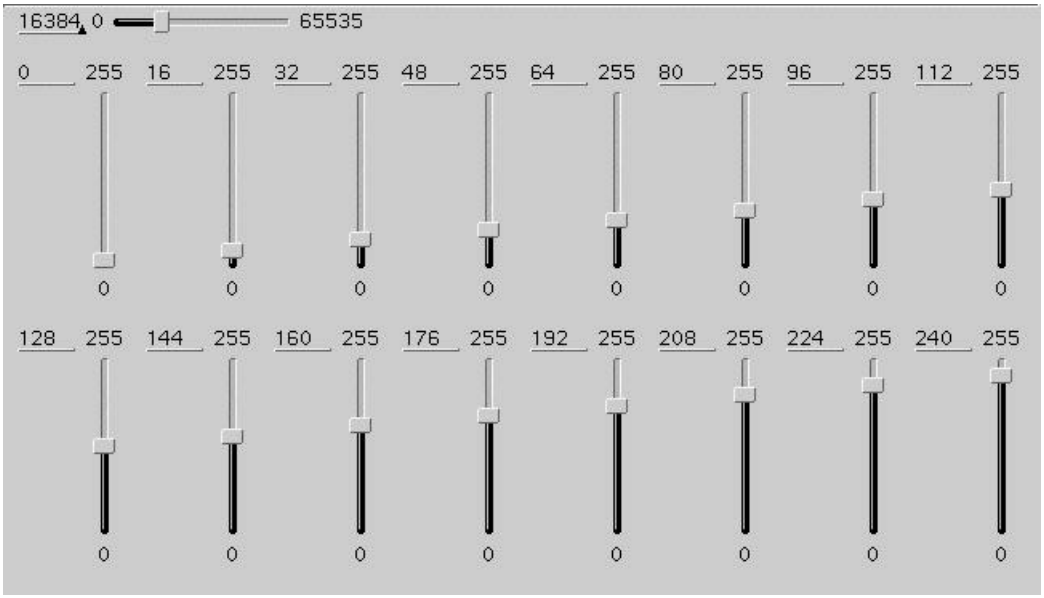

Figure 7.7: The Shading Control Panel

This control panel sets the range and profile of luminance values for shading purposes. This is necessary since the visual perception of a shaded luminance distribution may not be best represented by a simple linear density map.

The slider (top left) sets the range for the shading. Any luminance values above this range will be shaded at the highest level (typically white).

Within this range the relationship between luminance and display density (grey level) is defined by the set of vertical sliders (16 in number). The default (as displayed above) has a linear profile set in the range 0 to 255 (ie black to white). The user is free to adjust the slider levels to give some required profile which might be used to give a higher discrimination in certain ranges of luminance and a lower discrimination in other ranges.

If the shading control panel is adjusted after displaying a model with shading, the model must be re-shaded to reflect the new profile.

The operator section (Figure 7.8) of the control panel allows most of the SDF operators to be performed interactively.

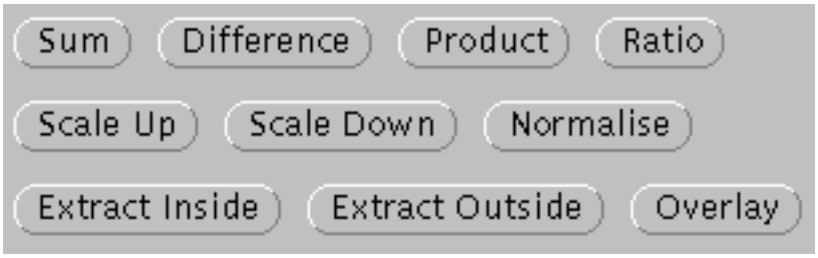

Figure 7.8: SDF Operations Buttons

(xix) *Sum button:*
(xx) *Difference button*
(xxi) *Product button:*
(xxii) *Ratio button:* These buttons perform the respective operations on two models. Both models must first be displayed on the screen. Following selecting the required button, the two models are selected by clicking in each of their windows (an audible click is given with each model selection). The resulting new model is displayed in a new window.

In case where the order of the operation is significant, it is determined by the order of selection of the two models.

(xxiii) *Scale Up:*
(xxiv) *Scale Down:* These buttons scale a single model (up or down) by the Scale factor set in the parameter section of the control panel. The model to be

scaled must be first displayed, then selected by clicking in its window after selecting the required button. A new model is created.

(xxv) *Normalize:* This button produces a new model by normalizing the luminance in a selected model so that the luminance value at the zenith is 1.0.

(xxvi) *Extract Inside:*
(xxvii) *Extract Outside:*
These two options are used when there is one or more edges (in addition to the horizon edge) include in the model. The operation results in a new model which contains only that part of the model inside, or outside, the edges in the model.

(xxviii) *Overlay:* This button allows one model to be overlayed on top of another. The top model (first one selected) masks out the bottom model where the top model contains data. If the top model has empty areas, then the bottom model shows through.

7.3 Model Displays

The display of luminance distribution in sky models is by means of a contour map, or a fully shaded display. Figure 7.9 shows a typical display. Several displays of this type can be up concurrently, only limited by computer memory. The projection used is equiangular.

There are two basic display options, shaded as in Figure 7.9, and the contour map as shown earlier in Figure 7.4. The small control panel at the top of the window contains a number of options as shown in Figures 7.10 and 7.11.

(i) *Grid:* This button toggles on and off the display of the 30° x 30° polar grid.

(ii) *Save:* The Save button saves this model to a file: A file name dialogue is presented to allow the user to enter a suitable file name.

(iii) *Import:* Import provides the function to import a model using the edges defined in the current window. These edges are then built into a new SDF model and thus impacts on the contouring process. The new model appears in a new window.

(iv) *Contour:* Requests the display to be given as a contour map.

(v) *Shade:* Requests the display to be given as a shaded display.

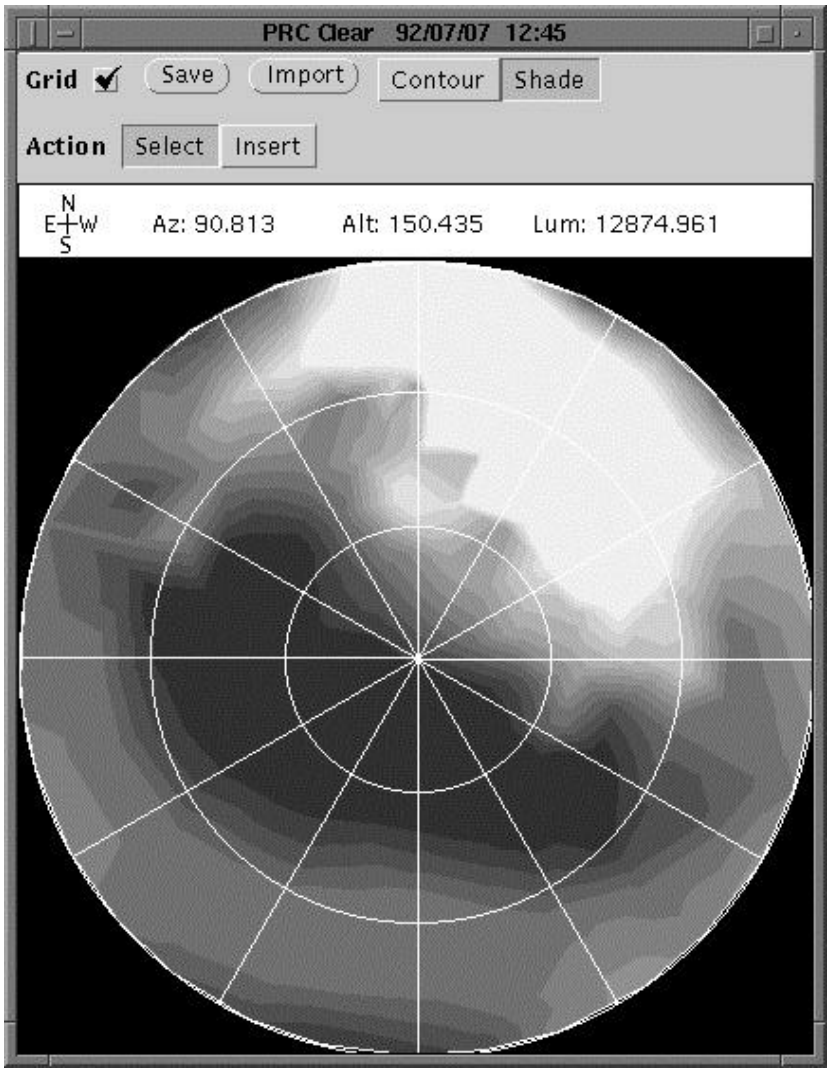

Figure 7.9: Typical Model Display Window

Figure 7.10: Model Control Options

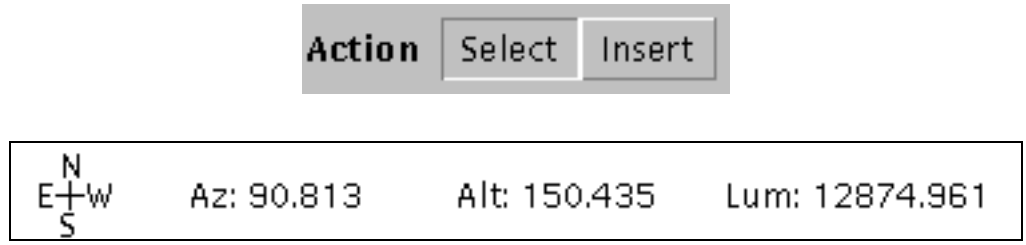

Figure 7.11: Window Actions

The Select and Insert operations in Figure 7.10 are designed to allow the user to draw an edge in the current window or to sample points for luminance from within the displayed model.

(vi) *Insert button:* Selecting Insert enables the mouse buttons as follows:

left button: mark the vertices of an edge in sequence.
middle button: terminate, and close, the edge.
right button: delete the edge (before closing the edge).

Several edges may be drawn and included in the model. Once drawn, there is currently no way of editing an edge

(vii) *Select button:* Choosing this option enables the mouse to select and sample any point in the model to display its position (azimuth and altitude) and luminance value.

This display (as shown in Figure 7.11) includes a compass orientation showing that the display is orientated with north upwards, and east to the left. This orientation is therefore what is seen by lying on one's back, looking upwards with the head pointing towards the north.

SKYMAP allows the user to create and display several models for concurrent display and manipulation. Each model is displayed in its own window which can be moved and sized to suit the users requirements. An example screen display is shown in Figure 7.12.

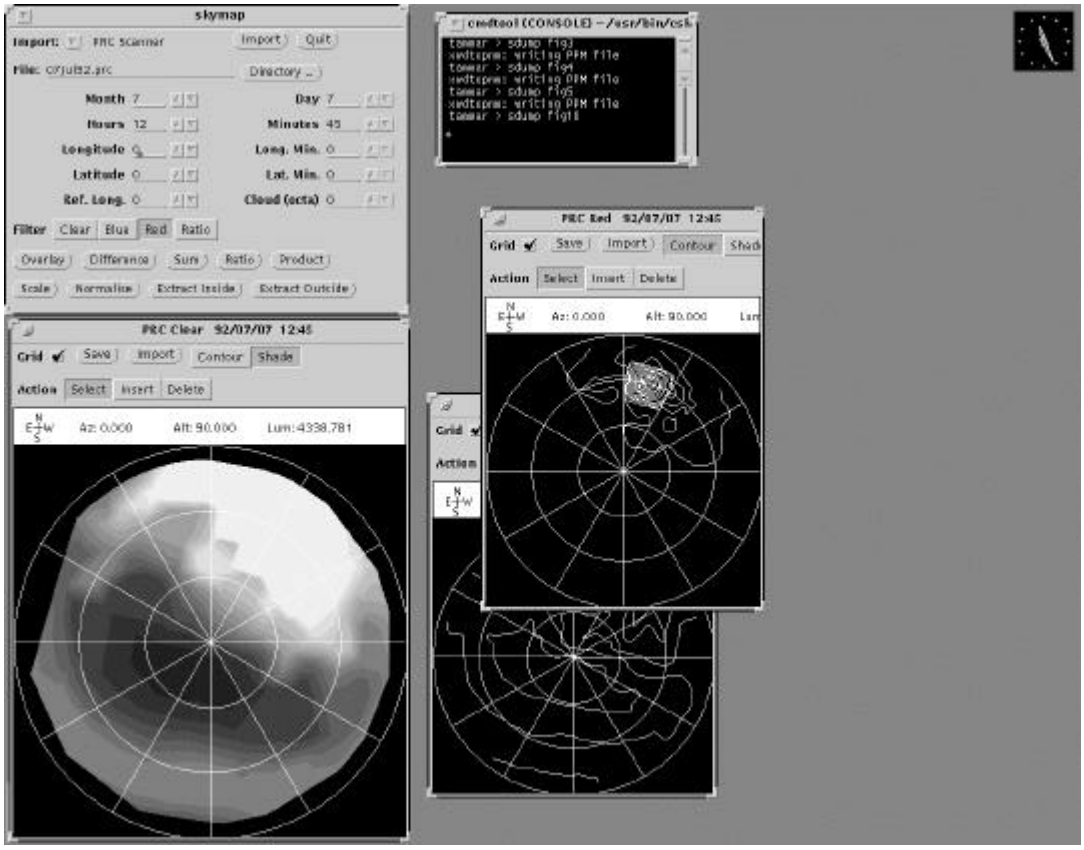

Figure 7.12 Example Screen Display

7.4 Example Displays

To demonstrate some of the flexibility of the SKYMAP package we include here some example displays. In each case the model type is selected and the necessary location and time parameters defined. The SDF model is then constructed and displayed. Figures 7.13 to 7.16 show a small selection of results.

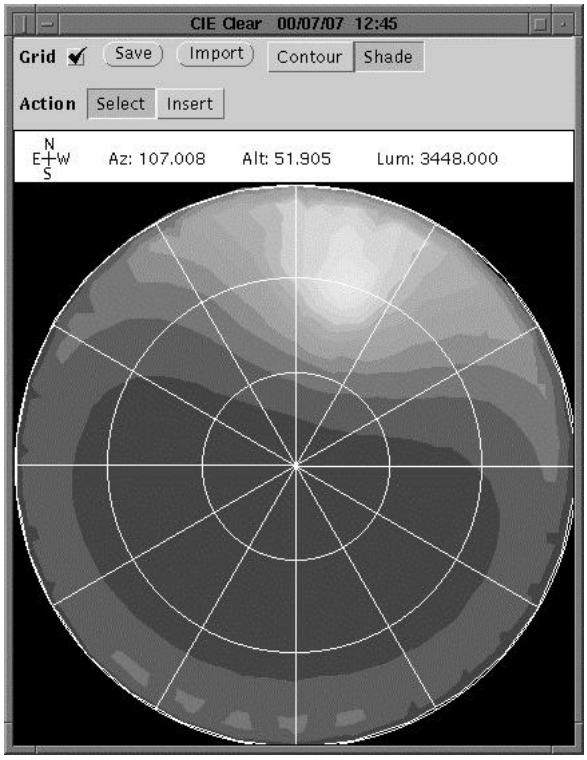

Figure 7.13: CIE Clear Sky (Sydney, 7th July, 12.45 hours):

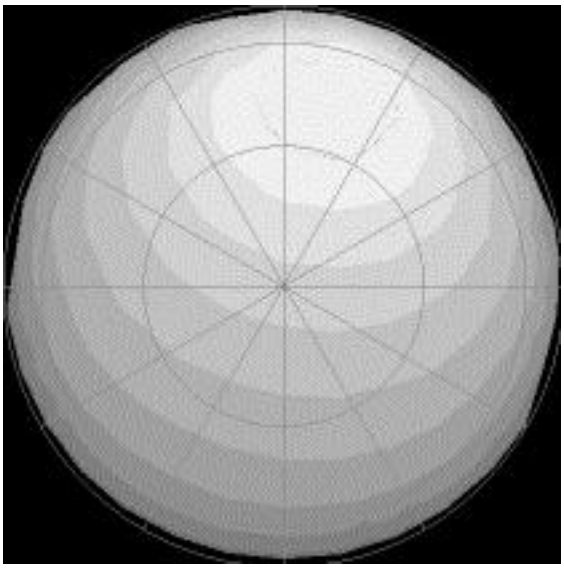

Figure 7.14: Perez All Weather Sky (Sydney, 7th July, 12.45 hours):

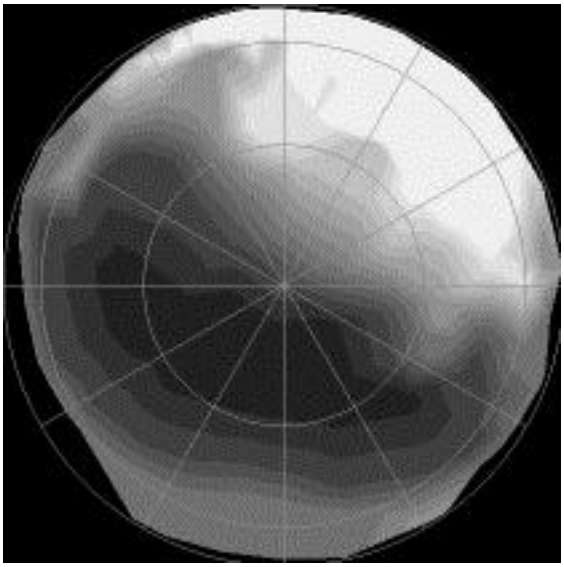

Figure 7.15: PRC scanner data sky (Sydney, 7th July 1992, 12.45 hours):

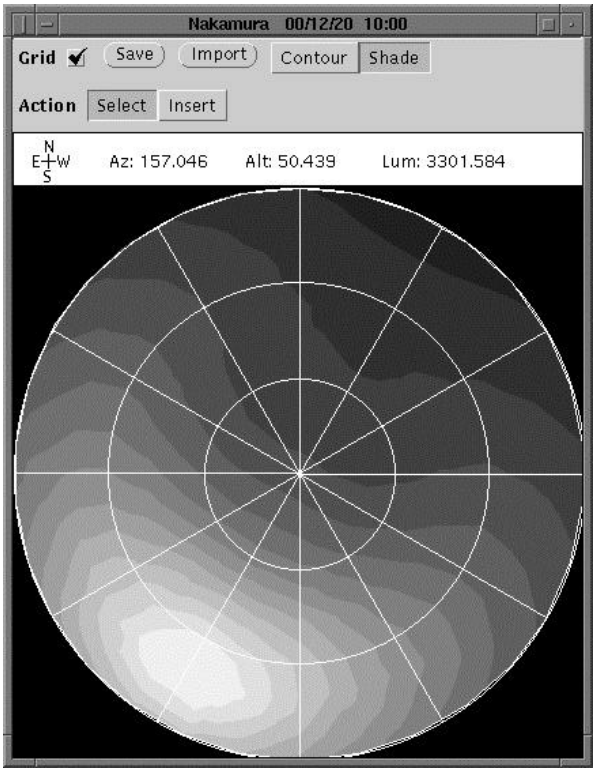

Figure 7.16: Nakamura Intermediate Sky (Fukuoka, Japan, 20th Dec, 10.00 hours)

8.0 Example Use of SDF Operators

8.1 Operator Use

In this section we will demonstrate a selection of the SDF operators using the SKYMAP package. All these operators are available in the SDF libraries and can be readily built into user-written software. The implementations within SKYMAP allow us to easily perform the operations and visualize the results. It is thus a convenient way of understanding the results obtained.

The whole purpose of these operators is to define a set of operations on skies that allow a range of manipulation tasks to be defined and made available to the user. While the concepts are quite straight forward, the characterization of the results is often rather simplistic - perhaps relying on simple uni-dimensional statistical measures (means, standard deviations, and so on). The value of some operation (ie the information content) is very much tied up with the 2-D spatial characteristics, ie the distribution over the sky dome. These characteristics are most easily seen by suitable graphical displays as provided by SKYMAP.

8.2 An Intermediate Sky

Intermediate skies are commonly used in daylighting simulations and computations. Their use is justified on the basis that partially cloudy skies are neither clear nor fully overcast and the intermediate sky provide some sort of average conditions. The intermediate sky is typically a simple weighted sum of CIE Clear and CIE Overcast skies.

The proportion of each depends on the cloud coverage if known, or estimated from measured solar radiation conditions. If we assume that we wish to build an intermediate sky from a composition of 30% clear sky and 70% overcast sky then we can define an intermediate sky from to following sequence of operations:

Firstly, we take the actual sky models and scale them:

$$M^A = 0.7 \cdot M^{Overcast}$$
$$M^B = 0.3 \cdot M^{Clear}$$

to produce two scaled models M^A and M^B as shown in Figures 8.1 and 8.2 as displayed by SKYMAP.

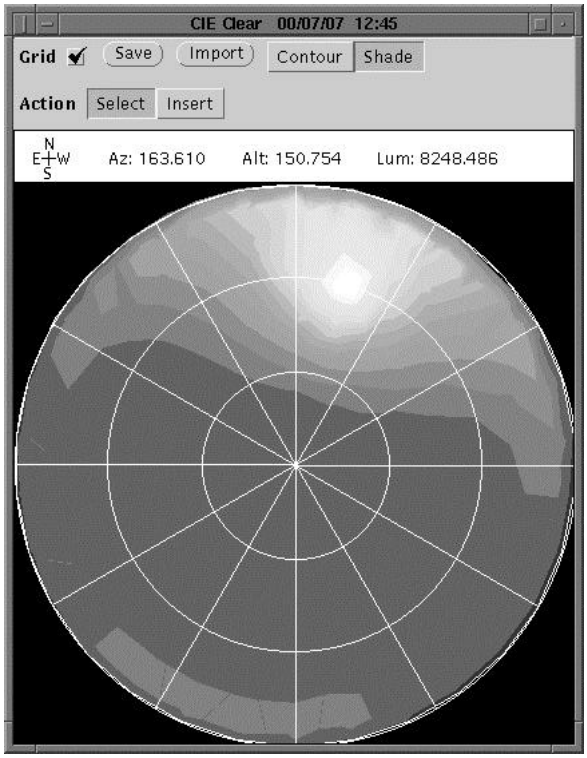

Figure 8.1: The Scaled CIE Clear Sky

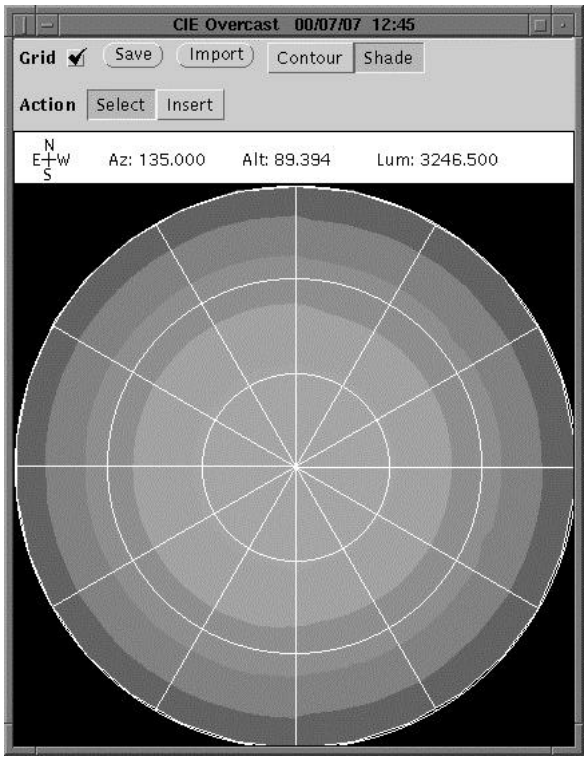

Figure 8.2: The Scaled CIE Overcast Sky

The next step is to sum the two scaled models together to produce the intermediate sky:

$$M^{inter} = M^{A} \overset{\smile}{+} M^{B}$$

Figure 8.3 shows the result of this operation for location: latitude -33°54', longitude 151°12' and reference longitude 150° (ie Sydney), on date 7th July at time 12:45.

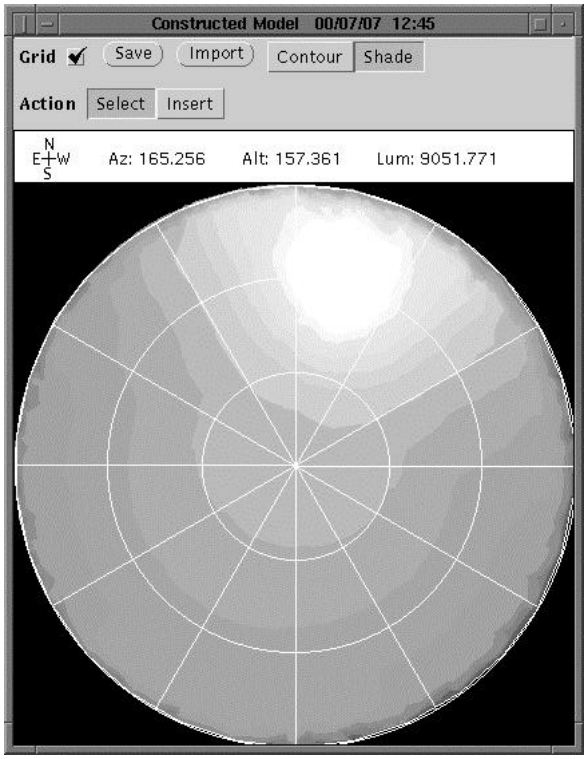

Figure 8.3: Example Intermediate Sky

Note that in Figures 8.1, 8.2 and 8.3 the same shading profile has been used, so they are directly comparable.

8.3 A Localised Clear Sky

The localised sky is a derived model based on some local data, and the presumption that the general character will be something like the CIE Clear model for clear sky conditions.

A simple approach might be to simply measure the zenith luminance at the site and assume that (for clear sky conditions) the CIE Clear Sky model provides the best estimate of the luminance distribution after it has been scaled to match the measured zenith luminance values. Formally this could be defined by:

$$M^L = f \bullet M^{Clear}$$

where f is the ratio of zenith luminances of the CIE Clear and locally measured values for the appropriate time and day.

If we have available more measured data, say measurements every 45° of azimuth and altitude, then we can construct a much better local model. For each of these points we can compute the ratio of the measured value to that of the CIE Clear sky at the corresponding points. An example set of ratios is given in Figure 8.4.

Azimuth	Altitude	Ratio: $\frac{Measured}{CIE\ \ Clear}$
0	0	1.2
0	45	1.3
0	90	1.1
0	135	0.9
0	180	0.8
45	0	1.1
45	45	1.3
45	135	0.7
45	180	0.6
90	0	0.5
90	45	0.6
90	135	0.9
90	180	0.8
135	0	1.1
135	45	1.2
135	135	1.3
135	180	1.4

Figure 8.4: Example ratios of CIE Clear to Local Measurements

This ratio data can now be input to build a "model" (M^{Ratio}), ie a model of the distribution of ratios over the sky dome, as shown in Figure 8.5. This model defines a surface matching (with interpolations) the measured data from Figure 8.4. If we multiply this model with the CIE Clear model (like as shown in Figure 8.1, but unscaled) then we will have a new model where the luminance values at the measured points match, but in between we have a surface like the CIE Clear model. Formally we have:

$$M^L = M^{Clear} \overset{\circ}{\times} M^{Ratio}$$

where M^{Clear} is the CIE Clear sky model. Note that M^{Ratio} is a complete surface as defined by the SDF interpolation scheme. The results of this operation are shown in Figure 8.6.

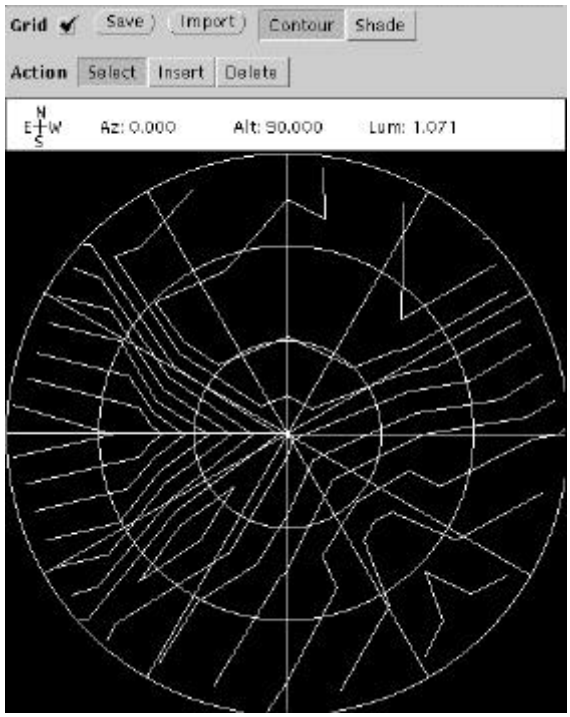

Figure 8.5: Example model of ratios

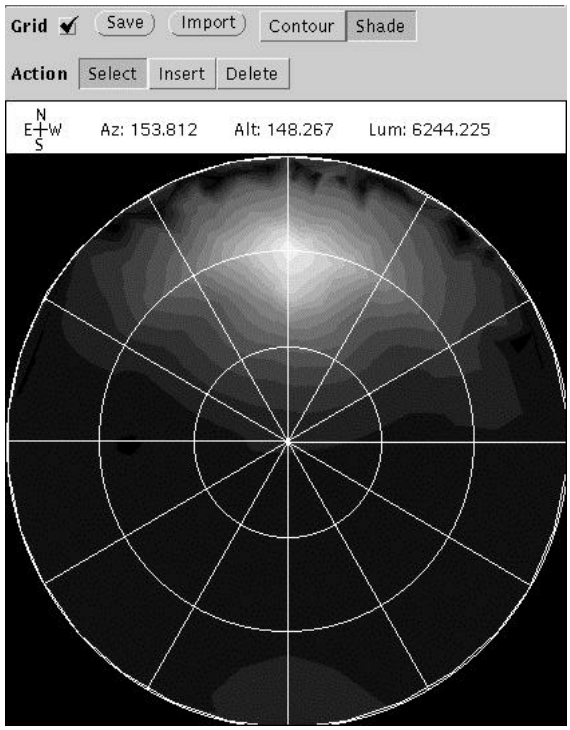

Figure 8.6: The Localised Sky

8.4 Ratio of Two Skies

To compare two sky models the ratio operator provides a useful tool. Here we can easily see the spatial variance characteristics. In this example we will take the ratio of the Harrison and CIE Clear skies. This operation is defined by:

$$M^{ratio} = M^{Harrison} \overset{\cup}{\div} M^{Clear}$$

The results are shown in Figure 8.7.

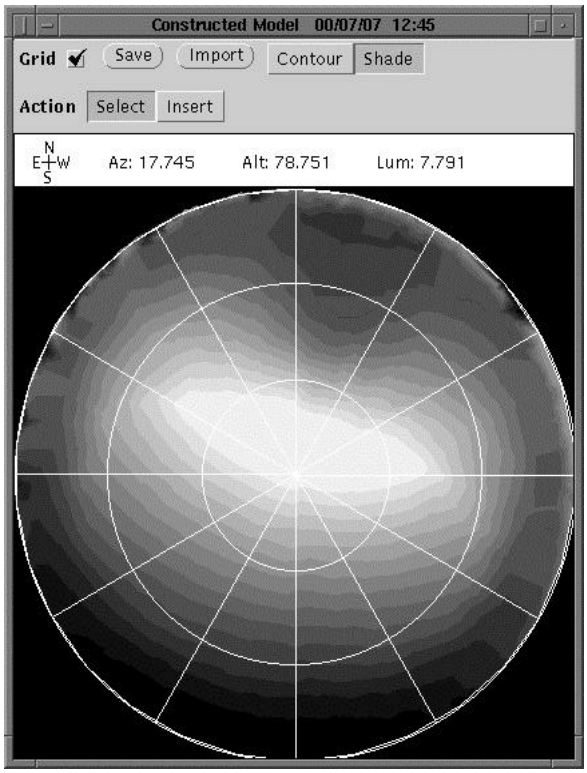

Figure 8.7: Ratio of Harrison and CIE Clear Skies

9.0 Macro Programming

Macro programming involves writing a program in the C-language, using the various functions (model access, parameter access and model operator) described earlier in this report and in detail in the Appendix C. The program which must be written is quite simple, providing the correct header files are included and libraries linked. We have already given some simple examples, but in this section we will provide a more complete example with a fuller explanation of its various components.

The example task will be to combine a series of CIE Clear and overcast skies for the 7th July 1992 at 15 minute intervals between 0800 and 1645 hours for the location: longitude 151°12' and latitude -33°54', with a reference Longitude of 150° (ie. Sydney).

The following program uses the standard functions defined for the SDF. Note that the line numbers have been added here to aid the explanation and do not form part of the actual program.

In this example, the new sky is assumed to be the summation of 30% of a clear sky and 70% of an overcast sky.

```
[1] #include <sdf.h>
[2] #include <sdfops.h>

[3] /*------------------------------------------------------------ *
[4]  *   match checks a model (model) against the given template   *
[5]  *   model (template) and returns whether or not they match.   *
[6]  *------------------------------------------------------------ */

[7]  static int match(
[8]    Model       template,
[9]    Model       model)
[10] {
[11]    int         year_t,month_t,day_t,
[12]                hours_t,minutes_t,
[13]                year_m,month_m,day_m,
[14]                hours_m,minutes_m;

[15]    (void)sdfGetParameter(template,"time","%c%c%c%c%c",
                    &year_t,&month_t,&day_t,&hours_t,&minutes_t);
[16]    (void)sdfGetParameter(model,"time","%c%c%c%c%c",
                    &year_m,&month_m,&day_m,&hours_m,&minutes_m);
[17]    if (year_t && year_t != year_m)
            return(0);
[18]    if (month_t && month_t != month_m)
            return(0);
[19]    if (day_t && day_t != day_m)
            return(0);
[20]    if (hours_t && hours_t != hours_m)
            return(0);
[21]    if (minutes_t && minutes_t != minutes_m)
```

```
            return(0);

[22]    (void)sdfSetParameter(model,"time","%c%c%c%c%c",
                    year_m,month_m,day_m,hours_m,minutes_m);
[23]    return(1);
[24]  }

[25]  int main(
[26]    int         argc,
[27]    char        **argv)
[28]  {
[29]    FILE        *fp;
[30]    Model       clear,
[31]                overcast,
[32]                model, temp1, temp2;
[33]    int         hours,
[34]                minutes;

[35]    fp = sdfOpenFile("07Jul92.sdf","w");
[36]    for (hours = 8 , minutes = 0 ; hours < 17 ; minutes += 15) {
[37]        if (minutes == 60) {
[38]            hours ++;
[39]            minutes = 0;
[40]        }
[41]        model = sdfCreateModel();
[42]        (void)sdfSetParameter(model,"site","%d%c%d%c%d",151,12,
                                        -33,54,150);
[43]        (void)sdfSetParameter(model,"time","%c%c%c%c%c",92,7,7,
                                          hours,minutes);
[44]        clear = sdfImport(CIE_CLEAR,model,match,fp);
[45]        overcast = sdfImport(CIE_OVERCAST,model,match,fp);
[46]        temp1 = sdfScale(clear,0.3);
[47]        temp2 = sdfScale(overcast,0.7);
[48]        model = sdfSum(temp1,temp2);
[49]        sdfStoreModel(model,fp);
[50]        sdfFreeModel(clear);
[51]        sdfFreeModel(overcast);
[52]        sdfFreeModel(model);
[53]        sdfFreeModel(temp1);
[54]        sdfFreeModel(temp2);
[55]    }
[56]    sdfCloseFile(fp);
[57]    return(0);
[58]  }
```

Some notes of explanation about this small program follow:

Lines 1 and 2: these include the header files for the SDF library of functions.

Line 7: the *match* function facilitates matching the required model to the specified template of parameters, typically time and location parameters.

Note how the match function operates. Since we assume (generally) that the required model is to be imported from a file, which may

contain several SDF models in sequence, we need to ensure that the correct model is loaded. The match function sets the required matching criteria. The import function then keeps searching forward in the input file until the correct (matching) model is found.

Lines 15 and 16: these get the parameter values from the template, and the model.

Lines 17 to 21: these perform the match, if the respective parameters were defined in the template.

Line 22: establishes the full set of parameters in new model.

Line 25: this is the start of the *main* program.

Lines 29 to 34: various declarations, note the models are of the type *Model*..

Line 35: opens file for saving results in write enabled mode.

Lines 36 to 40: sets up a loop to create a new model for each required time.

Line 41: creates a new model called *model*.

Lines 42 and 43: sets the template parameters into *model*.

Line 44: import a CIE clear model to be named *clear*.

Line 45: import a CIE overcast model to be named *overcast*.

Line 46: scales the clear sky by 0.3.

Line 47: scales the overcast sky by 0.7

Line 48: adds together the resulting two CIE models.

Line 49: saves the new model.

Lines 50 to 54: removes the models and frees associated data storage in memory.

Line 55: end of *for* loop.

Line 56: close output file will all models saved.

Line 57: terminates main program.

The *match* function in this example does little since the *sdfImport* function (lines 44 and 45) will always import the correct CIE models. Where the import operation is extracting models from an SDF file, then we may have to keep reading forward in the file until the correct model is found. The *match* function ensures the match and will return a *true* (1) value if the correct model has been loaded, other wise it returns a *false* (0) value. In such a case the *sdfImport* function will continue to search forward in the file until the correct model is found. Providing the match function is set up correctly the user does not have to be concerned with this process , it happens automatically.

10.0 Practical Validation of SDF Models

10.1 Daylight Study Comparisons

To evaluate the efficacy of the SDF models we have undertaken testing and evaluation of some sample models along side the standard CIE models. We have used the DOE-2 building energy simulation package as developed by the Simulation Research Group at Lawrence Berkeley Laboratory [Winkelmann and Selkowitz, 1985, Winkelmann et al, 1993]. This package has a capability to compute interior daylighting levels. The intention of this study was to demonstrate the viability of the SDF models and to examine the differences between the CIE models and the real sky condition as measured on two days at the Sydney Airport.

We have established a test environment for this work, based on a rectangular room with a single window, and the computation of the illuminance levels at a fixed control point. The room is orientated with the window facing north, east, south and west for the different sky conditions. The proportions and dimensions of the test room are shown in Figure 10.1.

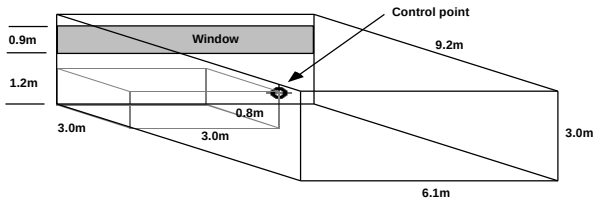

Figure 10.1 The test room

The room is empty and has no internal obstructions. The interior surfaces have reflectances of: 0.3 - floor, 0.5 - walls and 0.7 - ceiling. The window is single glazed with a transmittance at normal incidence of 100% (a rather ideal assumption, as a typical clear glass has a transmittance of around 85% at normal incidence).

Our interest is to study the behaviour of SDF models, so we have eliminated the direct beam component from the analysis. We have also assumed no reflectance from the ground. While these assumptions may not make the computed luminance values very meaningful, they give us a good deal of control over the various influencing factors, by eliminating those that would otherwise be assumed the same for both conventional DOE-2 modelling and the SDF approach.

The computations give us the average luminance values (lux) at the control point for each hour period, 0800-0900 hours, 0900-1000 hours, up to 1700 hours (labelled as 9, 10... 17 hours in the results in Figures containing the results).

Two days in July 1992 were chosen for study. The first day (20th July) is considered to a fully overcast day with heavy cloud (though not of uniform density) present all day. The second day (23rd July) is considered to be a fully clear day, with no detectable cloud present all day.

These two extreme days were chosen as they gave us the best chance to make sensible comparisons with the standard CIE Clear and Overcast models.

Four sets of SDF models were developed for the tests:

(i) SDF models derived from the CIE Overcast sky model for 20th July.

(ii) SDF models derived from the measured sky conditions for 20th July.

(iii) SDF models derived from the CIE Clear sky model for 23rd July.

(iv) SDF models derived from measured sky conditions for 23rd July.

In each case a sequence of models was developed for the hours 9, 10 to 17 hours. The measured data was derived from a sky scanner located at Mascot Airport, Sydney [Hayman et al, 1993].

The analyses were therefore aimed at:

(i) Verifying that the results from DOE-2 using its own standard sky models were comparable with the results using SDF models based on CIE Clear and Overcast sky models. We would expect a good match.

(ii) Examining the difference between the results obtained using an SDF model of the actual sky conditions as compared with the standard models in DOE-2. DOE-2 uses a linear combination of CIE Clear and Overcast models based on the measured radiance data. In this case the differences may be explained by the "non-standard" characteristics of the skies on the two chosen days. The two days were chosen specifically as examples of virtually fully clear and overcast skies.

The above test environment was used to build a room model for DOE-2, and the DOE-2 program was modified to accept SDF sky models through use of the SDF library functions. These provide access to each sky model, as well as the necessary interpolations to give the sky luminance at a specified azimuth and altitude angles. In all other respects the DOE-2 computations are identical for each experimental run.

10.2 Adding SDF Functionality to DOE-2

The DOE-2 program was modified to accept the SDF models. This is a relatively straight forward step even though DOE-2 is written primarily in FORTRAN. First we defined a small set of C-FORTRAN interface functions which are callable as FORTRAN procedures or functions:

sdfopen_ to open the files containing SDF models.

sdfload_ to load CIE Clear and Overcast models in SDF from the file.

sdflum_ to return the luminance at the specified azimuth and altitude of the average sky
according to the two coefficients.

sdfclose_ to close the SDF files.

sdffree_ to free the memory allocated to the loaded models

The implementation is given as follows:

```
#include <sdf.h>

/* global declarations */

Model modelclear,
      modelocast;

FILE  *fpclear,
      *fpocast;

/* opens files */

void sdfopen_()
{
  fpclear = sdfOpenFile("23Jul92.clear","r");
  fpocast = sdfOpenFile("23Jul92.overcast","r");
  if ((fpclear == (FILE *)NULL) || (fpocast == (FILE *)NULL)) {
    fprintf(stderr,"sdfdoe: Unable to open data files\n");
```

```
    exit(-1);
  }
}

static int match(model,yearW,monthW,dayW,hourW,minW)
Model model;
int yearW, monthW, dayW, hourW, minW;
{
  int year, month, day, hour, min;

  sdfGetParameter(model,"time","%c%c%c%c%c", &year, &month, &day, &hour,
&min);
  if (year != yearW) return(0);
  if (month != monthW) return(0);
  if (day != dayW) return(0);
  if (hour !=  hourW) return(0);
  if (min != minW) return(0);
  return (1);
}

/* get next matching model */

void sdfload_(yearW,monthW,dayW,hourW,minW)
int *yearW, *monthW, *dayW, *hourW, *minW;
{
  modelclear = sdfCreateModel();
  while (sdfLoadModel(modelclear,fpclear)) {
    if (match(modelclear,*yearW,*monthW,*dayW,*hourW,*minW))
      break;
    sdfFreeModel(modelclear);
    modelclear = sdfCreateModel();
  }
  modelocast = sdfCreateModel();
  while (sdfLoadModel(modelocast,fpocast)) {
    if (match(modelocast,*yearW,*monthW,*dayW,*hourW,*minW))
      break;
    sdfFreeModel(modelocast);
    modelocast = sdfCreateModel();
  }
  if ((modelclear == (Model)NULL) || (modelocast == (Model)NULL)) {
    fprintf(stderr,"sdfdoe: Unable to load models\n");
    exit(-1);
  }
}

/* computes average luminance at ax, alt */

float sdflum_(az, alt, c1, c2)
float *az, *alt, *c1, *c2;
{
  float lumclear, lumocast, average;
  struct _point point;

  point.azimuth = *az;
  point.altitude = *alt;
  lumclear = sdfPointLuminance(modelclear,&point);
  lumocast = sdfPointLuminance(modelocast,&point);
  average = *c1 * lumclear + *c2 * lumocast;
  return(average);
}

/* closes files */

void sdfclose_()
```

```
{
  sdfCloseFile(fpclear);
  sdfCloseFile(fpocast);
}

/* free a model */

void sdffree_()
{
  sdfFreeModel(modelocast);
  sdfFreeModel(modelclear);
}
```

We can construct a simple sample FORTRAN program which uses these functions and displays some results.

```
      PROGRAM Example

      INTEGER year,month,day,hours,minutes
      REAL phi,theta,c1,c2,val

      CALL sdfopen

      year = 92
      month = 7
      day = 23
      hours = 12
      minutes = 0
      CALL sdfload(year,month,day,hours,minutes)

      phi = 0.0
      theta = 1.57
      c1 = 0.5
      c2 = 0.5
      val = sdflum(phi,theta,c1,c2)

      write(*,"(3F8.2)") phi,theta,val

      CALL sdfclose
      CALL sdffree

      END
```

DOE-2 was modified to use these FORTRAN routines and to link with the SDF library functions. This enabled the required SDF models to be loaded and used in the DOE-2 simulations, just replacing the standard DOE-2 sky modelling strategy.

10.3 The Results - Overcast Sky

The 20th July 1992 is considered to be a fully overcast sky for all the day. Figure 10.2 is a photograph of the sky at 1515 hours and is typical of the sky throughout the day.

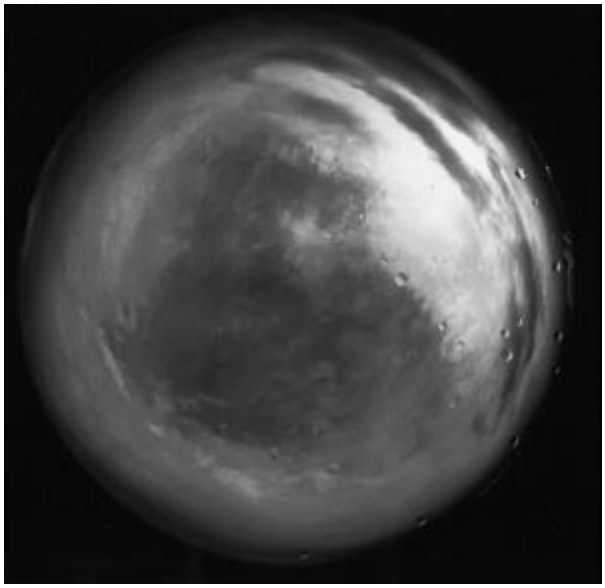

Figure 10.2: The Overcast Sky at 1515 hours on 20/7/92

Figures 10.3(a)-(d) show the results for this day (20/7/92) for each orientation of the test room. The average illuminance (lux) at the control point in the room is plotted against time. The value at 9 is the average over the hour 0800 to 0900 hours.

The first thing to note is the close match between the DOE-2 Overcast sky prediction and the SDF Overcast sky models. This was expected and confirms that the SDF function primitives are providing essentially the same results as those produced from the values computed directly for the CIE Overcast model equations as integrated into DOE-2. Remember that DOE-2 also uses the CIE Overcast model, but computes the luminance values directly from the empirical expression. The SDF luminance values are obtained from the appropriate SDF function which estimates the luminance from a computation (interpolation) on the SDF iso-luminance contour map. This was created originally from the CIE Overcast model expression.

It is clear from these graphs that the actual sky conditions vary considerably from the idealised CIE model. This is also expected as we know that the CIE Overcast sky represents dark overcast skies with stratus type clouds. Such skies are not typical of the bright overcast skies experienced in Australia. In these skies the influence of the solar component can be seen.

The actual sky conditions are reflected in the SDF measured model results. The MDMs for the days studies were converted into SDF and then read directly into DOE-2 for the simulation.

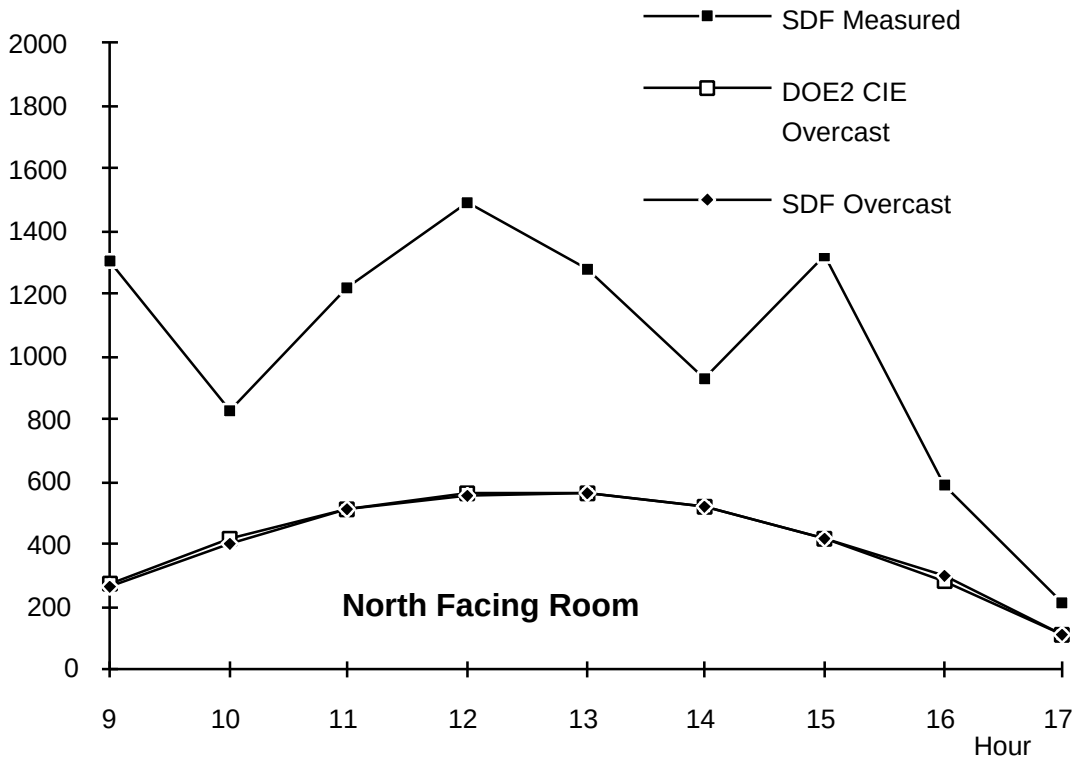

Figure 10.3(a): Overcast Sky - North Facing Window

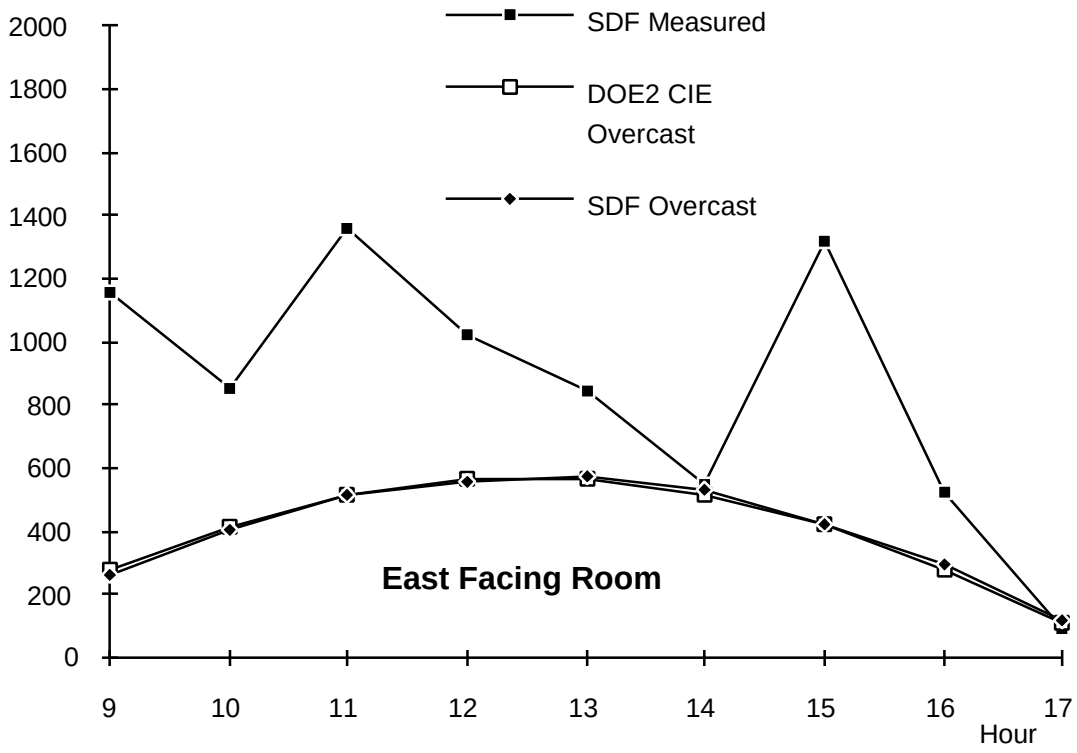

Figure 10.3(b): Overcast Sky - East Facing Window

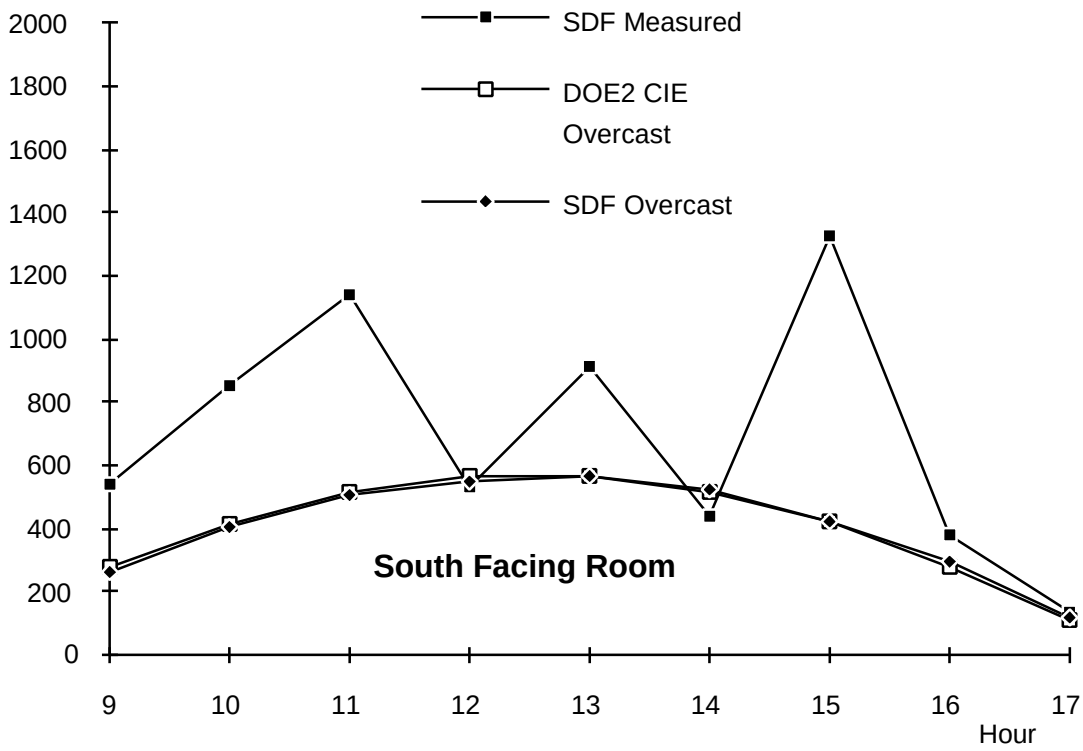

Figure 10.3(c): Overcast Sky - South Facing Window

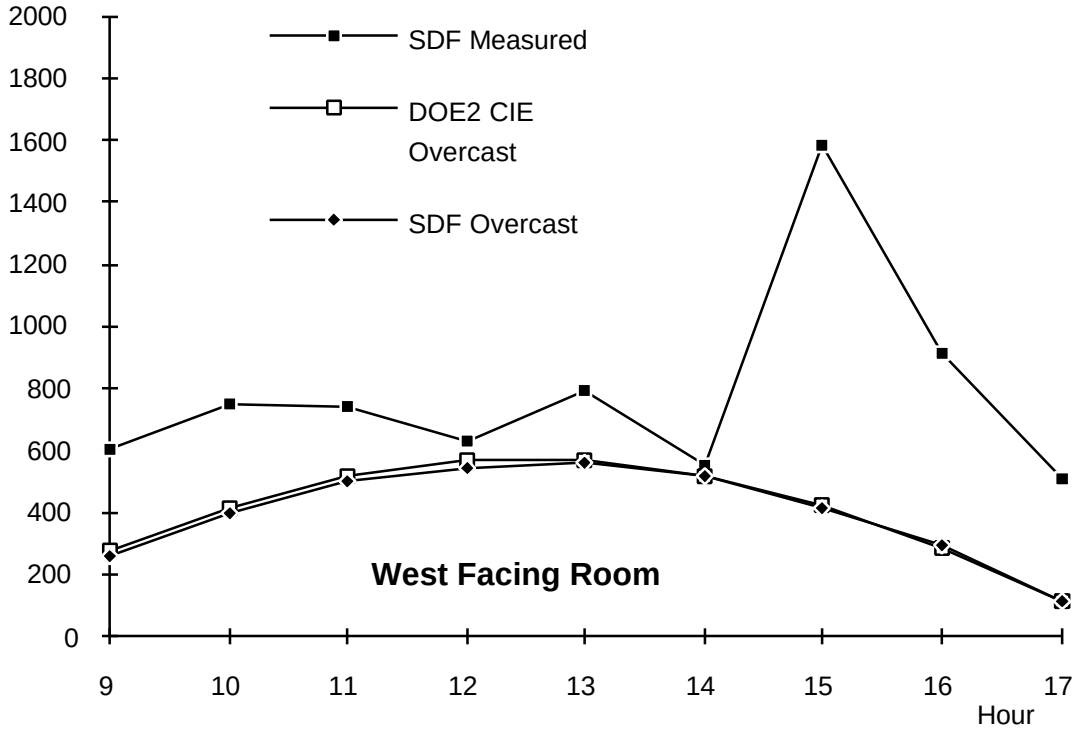

Figure 10.3(d): Overcast Sky - West Facing Window

10.4 The Results - Clear Sky

The 23rd July, in contrast, had virtually totally clear conditions all day. Figure 10.4 shows a photograph at 1115 hours and is typical of the conditions during the day.

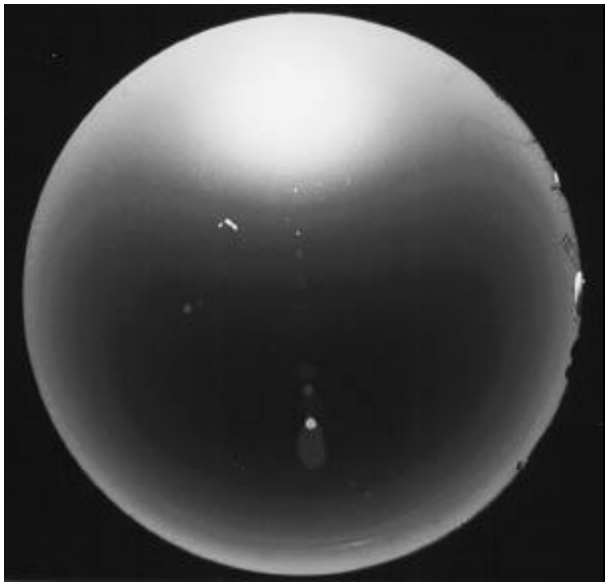

Figure 10.4: The Clear Sky at 1115 hours on 23/7/92

The clear sky conditions provide more variance than noted in the Overcast sky comparisons. The DOE-2 and SDF clear sky models are relatively close but not identical as shown in Figures 10.5(a) to (d). This is probably explained by the fact that the variation of sky luminance is much greater depending on the solar altitude. This places much more significance on the interpolation schemes for estimating intermediate luminance values. DOE-2 computes the luminance for 20 different sun positions and intermediate points are linearly interpolated. The SDF models use a more complex interpolation scheme in which the number of (effective) points use depends on the rates of change of the luminance in each part of the sky dome.

In both cases the same atmospheric conditions were assumed (atmospheric moisture of 1.21cm, and a turbidity of 0.12). These are typical for Sydney clear skies.

It is also of note that the illuminance results for actual sky conditions are not too far from these standard models. The differences are not uniform, which probably indicates that the actual luminance distribution across the sky dome does not match the standard CIE models. It might still be concluded that the CIE Clear model seems to have a reasonable predictive

value, at least for this particular day. Some of the differences might also be explained by the assumed, and possibly, varying atmospheric conditions (turbidity and moisture content).

10.5 Summary

The purpose of these comparison studies was to prove the feasibility of integrating the SDF functions into a standard simulation package, like DOE-2. This has been successful in the sense that the results obtained above indicate that both the SDF model and the normal DOE-2 sky models give essentially the same results.

One issue which surfaced during the experiment concerned that fact that DOE-2 prepares its sky data one day in advance before beginning the hourly simulation for each day. To enable an efficient use of the SDF models, all models for a day should be loaded concurrently so that repeated file accesses are avoided. At the end the day all models should then be removed and the memory recovered, ready for the next day's simulation.

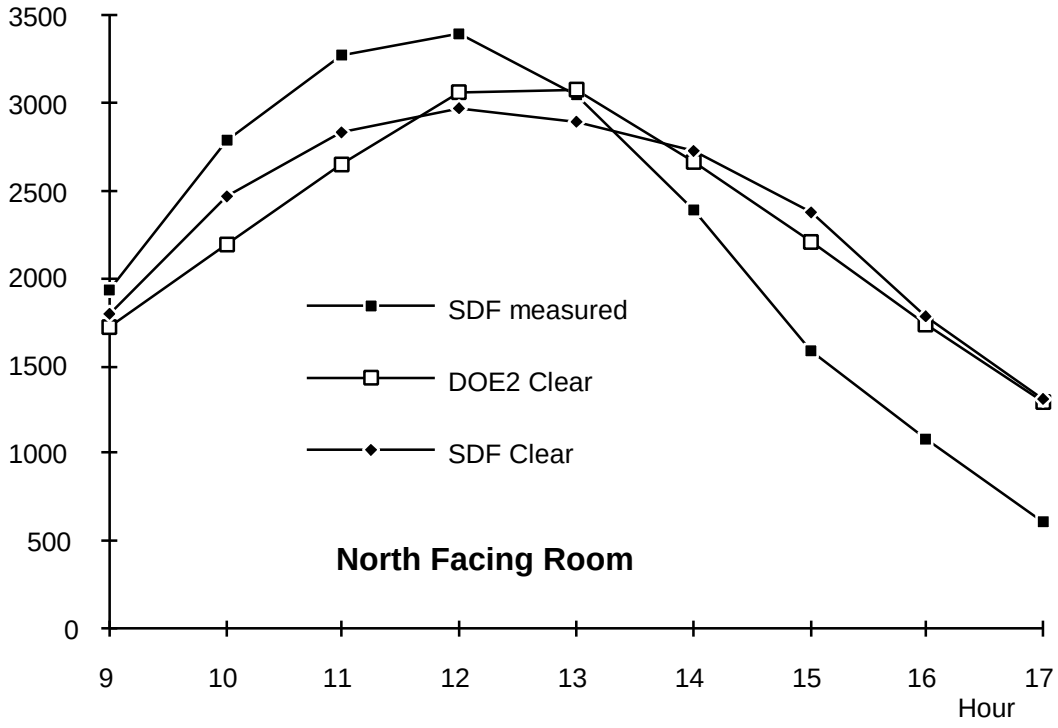

Figure 10.5(a): Clear Sky - North Facing Window

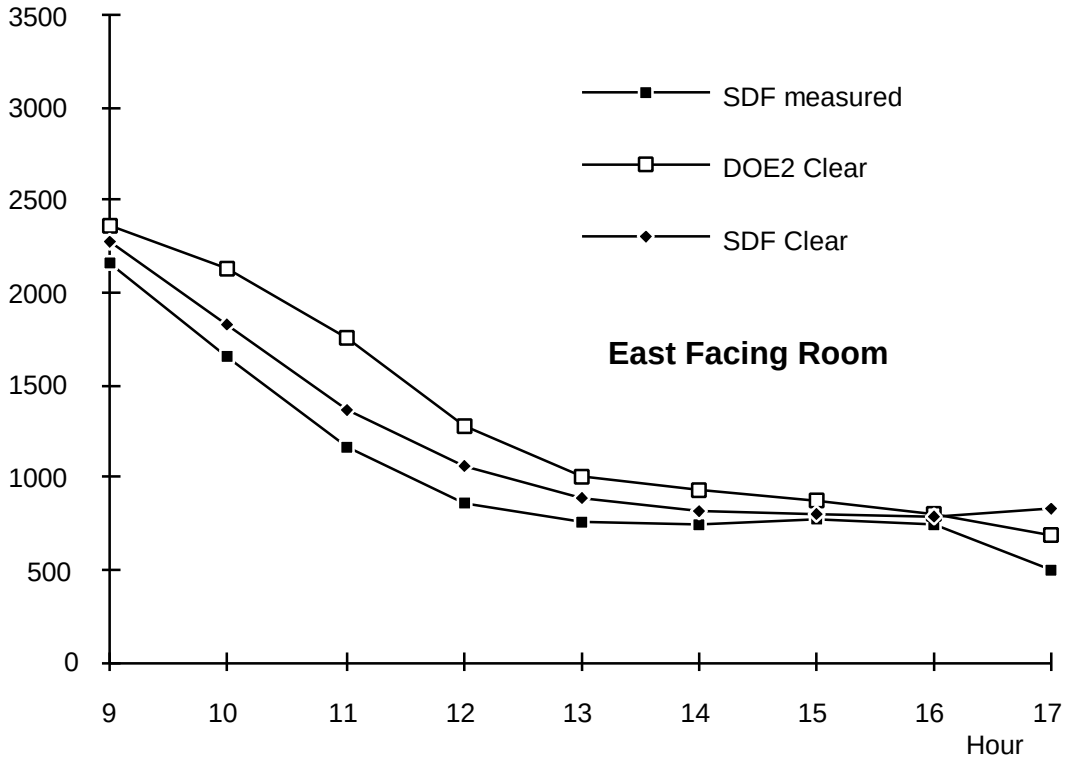

Figure 10.5(b): Clear Sky - East Facing Window

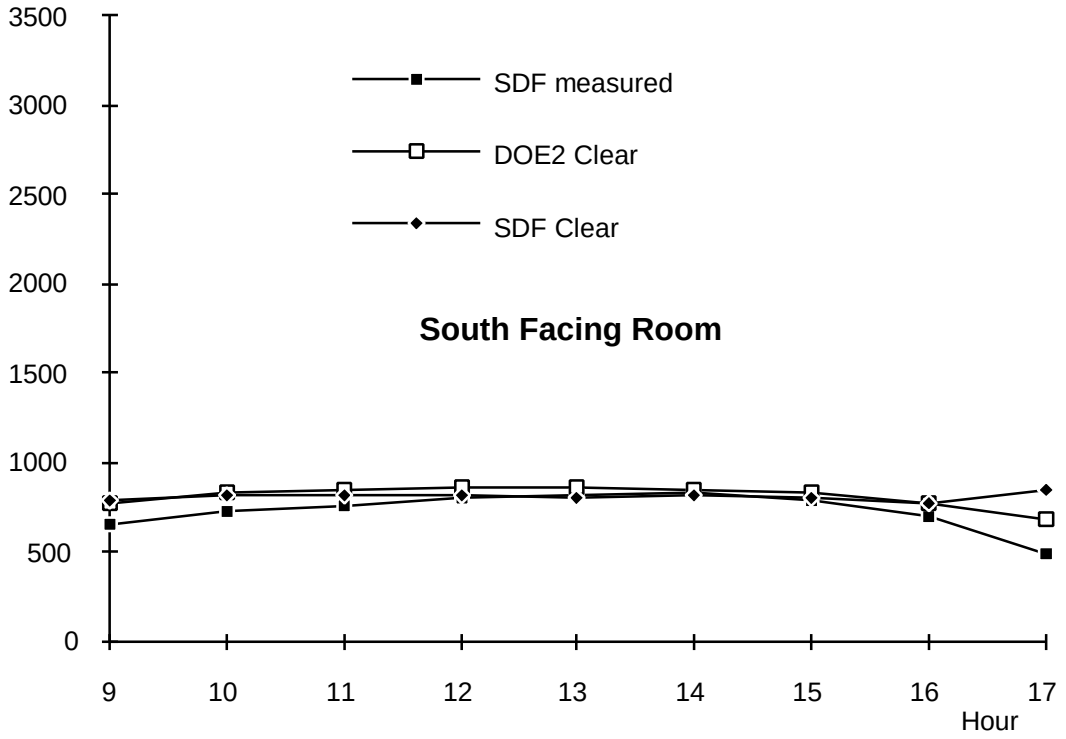

Figure 10.5(c): Clear Sky - South Facing Window

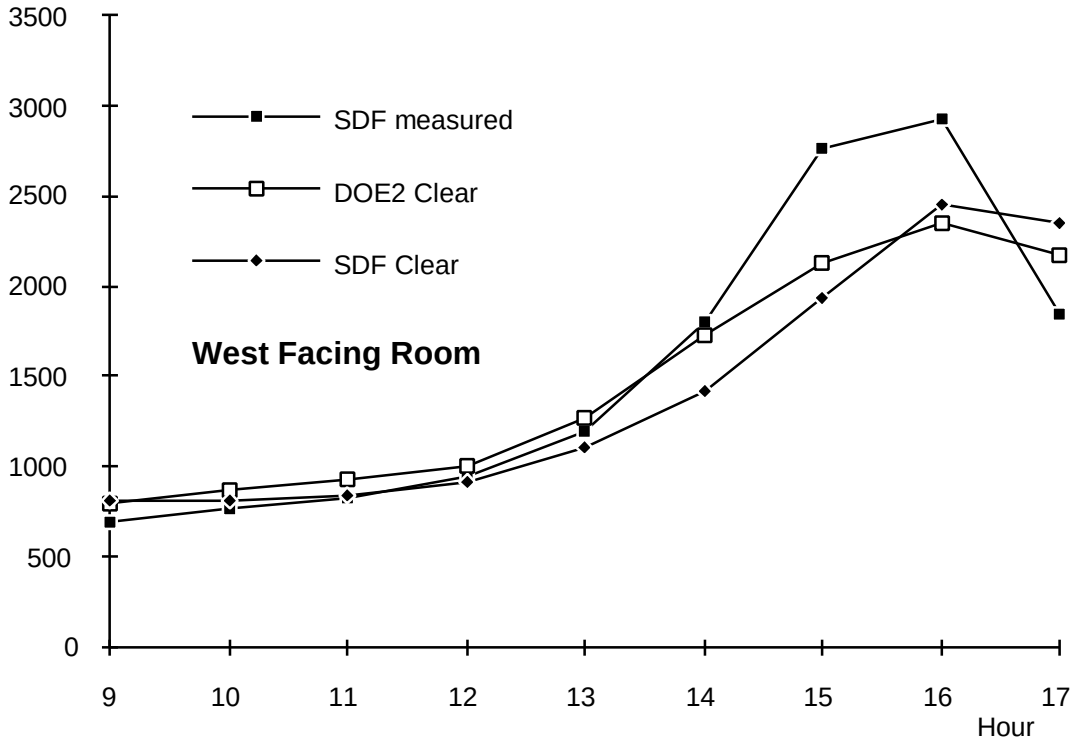

Figure 10.5(d): Clear Sky - West Facing Window

11.0 Conclusions

The research of the ARC project "The development and verification of sky models which represent the predominant skies in Australia for the prediction of daylight in buildings" has been widened to include sky conditions world-wide. Measured data from Australia, Japan, the UK and the USA have been used to develop universal luminance models demonstrating that the adopted sky modelling strategies can be applied world-wide. These measured data have been used to develop the Standard Digital Form (SDF) for representing sky luminance (and the SKYMAP system) which has been applied and tested in internationally used software. The results will be utilized in the development of integrated software packages such as ADELINE in the International Energy Agency's new Task 21: Daylight in Buildings.

Modelling the luminance distribution of the sky is a complex problem. While there have been a wide range of proposals for empirical models, none are universally reliable. The extremes of clear and fully overcast skies are better modelled, but partly cloudy skies are more problematical. Modern scanners allow good quality data to be collected to capture real sky conditions which can be used for further modelling or analysis tasks. Such data are, however, expensive to obtain and a lot of data must be collected to capture the variety of conditions necessary to obtain the required representative skies for analysis.

This project has been concerned with proposing some new modelling strategies which can facilitate further development of sky models. The SDF models provide a number of new features which will help in the development of luminance models. In particular there is now a modelling framework which:

(a) Provides a scheme for integrating data from different sources (eg empirical models and measured data) into a consistent framework.

(b) Allows different models of the sky luminance to be compared and manipulated using a common set of basic operators which are accessible from user written software.

(c) Allows an explicit inclusion of luminance discontinuities and other edge effects.

(d) Allows sequences of SDF models to be built and accessed by standard simulation software packages.

Models in SDF can be manipulated with a standard set of operators which allow model builders to design and construct models for new locations based on known reference models, local data and sky conditions. We are now better able to consider the development of models of real skies in localities where there is no detailed scanner data, or where it is known that the reference models do not well represent the local conditions.

The outcomes from the project are embodied in a set of C-language functions which define the data structures and basic operations that can be performed on SDF models. These are built into libraries which can be accessed by user-written software or integrated into other simulation packages which can access SDF sky models. The operation of these facilities has been demonstrated in this report. To show the capabilities of these SDF operators and functions the SKYMAP system has been implemented. This system provides a demonstration of an interactive program incorporating most of the SDF operators and functions.

A series of C-language functions are available for inclusion in user-written software systems which will allow a user to readily adopt the proposed SDF modelling standard and to effectively use it with a minimum of programming effort. The libraries of functions are available to researchers for testing, validation and use in non-commercial software free of cost. For commercial use there may well be the need to negotiate a royalty arrangement.

The methodology used has pin-pointed the following:

(i) The differences in sky luminance patterns at various latitudes and particularly the inadequacy of the CIE fully overcast sky to represent the fully overcast sky condition in latitudes between 50° South and 50° North where most of Australia is located. This research has validated the conclusions in the previous ARC project.

(ii) Highlighted the importance of determining cloud cover as a description. Filtered (blue and red) sky scanner measurements to indicate total cloud amount have been inconclusive and manually tracing edges of clouds from photographs has proved onerous. It is evident that for research purposes as well as for meteorological observations, automatic filtering methods to isolate cloud components need investigating. These are now being assessed in a new ARC project "Automated sky luminance and cloud cover estimation for improved sky modelling in different climate zones".

References

Cline, A. K. and Renka, R. L., 1984, *A Storage Efficient Methods for Constructions of a Theissen Triangulation*, Rocky Mountain Journal of Mathematics, vol 14, pp119-139.

Commission Internationale de L'Eclairage, (CIE) 1973, *Standardisation of luminance distribution on clear skies*, CIE Publ. 22, Paris, Bureau Central CIE.

Commission Internationale de L'Eclairage, (CIE) 1990, *ISO/CIE: Standard overcast and clear sky*, 3rd. draft, Div 3, CIE.

Commission Internationale de L'Eclairage, (CIE), *Standardisation of luminance distribution on clear skies*, CIE Publ. 22, Paris, Bureau Central CIE, 1973.

Carruthers D, Uloth C and Roy G. G., 1990, *An Evaluation of Formulae for Solar Declination and the Equation of Time*, Research Report No RR17, School of Architecture, the University of Western Australia, Jan.

Davies, G. B., Griggs, D. J. and Sullivan, G. D., 1992, *Automated Estimation of Cloud Amount Using Computer Vision*, Journ. of Atmospheric and Oceanic Technology, Vol 9, Feb, pp81-85.

Fortune, S., 1987, *A Sweepline Algorithm for Voronoi Diagrams*, Algorithmica, Vol 2, pp153-174.

Green, P. J. and Sibson, R., 1978, *Computing Dirichlet Tessellations in the Plane*, The Computer Journal, vol 21, pp167-173.

Harrison, A., 1991, *Directional Luminance versus cloud cover and solar position*, Solar Energy, vol 46, pp13-20.

Hayman, S. and Stevens G.,1992, *Sky Luminance Models*, ANZAAS, Perth.

Hayman, S., 1994, *Appropriate Sky Models*, Proceedings of the International Workshop on Daylight in Buildings, University of Sydney, Nov 18-19.

Hayman, S., Prasad, T., Ruck, N. and Julian. W., 1993, *International Daylight Measurement Programmes in Australia: Sky Luminance Research*, Proc. 7th European Lighting Conference, Edinburgh, pp795-798.

Kittler, R., 1985, *Luminance distribution characteristics of homogeneous skies: a measurement and prediction stratagy*, Lighting Research & Technology, vol 17. No. 4, pp183-188.

Kittler, R., 1987, *Homogeneous intermediate skies as simulating models for unsteady state daylight climates*, Proceedings 21st CIE Session, Venice, La Scientifica.

Krockmann, E, 1991, *Sky Scanner: photometer for sky luminance distribution measurement manual*, PRC Krockmann GMBH, Berlin., PRC Krockmann GMBH, Berlin.

Lawson, C., 1977, *Software for C^1 Surface Interpolation*, in Rice J. R. (ed), Mathematical Software III, Academic press, NY, pp161-194.

Littlefair, P, 1981, *The Luminance Distribution of an Average Sky*, Lighting Research and Technology, vol 13, no 4, pp192-198.

Marland, B., 1991, *The daylighting climate of Zambia: a bias for daylighting design in tropical regions*, Proceedings 22nd CIE Session, Melbourne, Australia.

Matsuura, K., 1987, *Luminance distributions of various reference skies*, CIE Committee TC 3-09, Draft Report.

Nakamura, H. and Oki, M. and Iwata, T., 1987, *Mathematical Description of the Intermediate Sky*, Proceedings 21st CIE Session, Venice, La Scientifica, pp230-31.

Nakamura, H. and Oki, M., 1986, *The mean sky composed taking into account the relative sunshine duration*, International Daylighting Conference, Long Beach, Ca, ASHRAE, Atlanta.

Navvab, M., Karayel., Ne'eman, E. and Selkowitz, S., 1984, *Analysis of turbidity for daylight calculations*, Energy and Buildings, vol 6, pp293-303.

Perez, R., Michalsky, J. J. and Seals, R., 1991, *Evaluation of algorithms for sky luminance distribution: prospects for performance improvements*, proceedings ISES World Congress, Denver.

Perez, R., Seals, R. and Michalasky, J., 1992, *An All-weather model for sky luminance distribution*, Proc. of ASES Annual Meeting, Cocoa Beach Florida, and Solar Energy (in press).

Perez, R., Seals, R., Michalasky, J. and Ineichen P., 1992, *Geostatistical properties and Modelling of Random Cloud Patterns for Real Skies*, (Submitted to Solar Energy).

Reid, G. B., Roy, G. G. and Ruck, N., 1992, *Modelling the Sky: A Proposed Standard Form*, Proceedings Annual Research Conference, Department of Computer Science, The University of Western Australia.

Ruck, N., Roy G. G. and Reid G. 1993, *Modelling the Sky - A Standard Digital Form, Proceedings*, 3rd International Conference, International Building Performance Association, Adelaide, 16th-18th Aug, pp525-531

Scartezzini J-L, 1994, *Recent Progress in Daylighting*, Proceedings of the International Workshop on Daylight in Buildings, University of Sydney, Nov 18-19.

Sloan, S. W.,, 1991, *A Fast Algorithm for Generating Constrained Delaunay Triangulations*, Research Report, No 065.07.1991, University of Newcastle, July.

Tregenza, P., 1987, *Subdivision of the Sky Hemisphere for Luminance Measurements*, Lighting Research and Technology, Vol 19, pp13-14.

Watson, D. F., 1981, *Computing the n-Dimensional Delaunay Triangulation with Application to Voronoi Polytypes*, The Computer Journal, vol 24, pp167-172.

Wegner, J., 1975, *Berechnung der mittleren Beleuch- tungsstarke durch Tageslicht in Innenraumen auf der Grundlage der mittleren Leuchtdichteverteilung des Himmels*, Dissertaion Technical University, Berlin.

Winkelmann, F. C. and Selkowitz, S., 1985, *Daylighing Simulation in the DOE-2 Building Energy Analysis Program*, Energy and Buildings, vol 8, pp271-286.

Winkelmann, F. C., Birdsall, B. E., Buhl, W. F., Ellington, K. L., Erdam, A. E., Hirsch, J. J. and Gates S., 1993, *DOE-2 Supplement, Version 2.1E*, Lawrence Berkeley Laboratory, Report No. LBL-34947.

Publications from the Project

Ruck, N., Roy G. G. and Reid G., 1993, *Modelling the Sky - A Standard Digital Form, Proceedings*, 3rd International Conference, International Building Performance Association, Adelaide, 16th-18th Aug, pp525-531.

Roy, G. G., Ruck, N., Reid, G. and Julian, W., 1995, *Modelling the Sky for Luminance*, Lighting Research and Technology (forthcoming).

Roy, G. G., Ruck, N. and Winkelmann, F, 1995, *New Modelling Technologies for Sky Luminance*, Proceedings 23rd CIE Session, New Delhi (forthcoming).

Roy, G. G., Ruck, N. and Winkelmann, F., 1994, *A Digital Modelling Strategy for Sky Luminance*, Proceedings International Workshop on Daylighting in Buildings, University of Sydney, Nov.

Reid, G., 1995, *Contouring and Representation of Discontinuous Surface Models*, MSc Thesis, The University of Western Australia (forthcoming).

Appendix A. Various Formulae

This section contains a summary of the various key formulae used in the SDF functions. Please note that not all computational situations are reflected here, for example where altitude angles are in the range $\pi/2$ to π. Also, the distinction between angles measured in degrees and radians is not always made explicit.

Sun Position

In order to calculate the position of the sun, we need the time and date, as well as the latitude, longitude and reference longitude of the location.

Solar declination is the altitude angle between the sun position and the equatorial plane. the equation of time represents the difference (in seconds) between clock time and solar time due to purtibations in the earth's orbit around the sun. The formulae for solar declination and the equation of time is as proposed by Carruthers, Uloth and Roy [Carruthers et al, 1990] are given by:

$$\begin{aligned} d_s = {} & 0.322003 - 22.9711\cos t + 3.94638\sin t \\ & -0.357898\cos 2t + 0.019334\sin 2t \\ & -0.143980\cos 3t + 0.059280\sin 3t \end{aligned}$$

where:

$$t = \frac{2p \cdot J}{366}$$

and

$$\begin{aligned} ET = {} & 5.0323 - 430.847\cos t + 12.5024\cos 2t + 18.25\cos 3t \\ & -100.976\sin t + 595.275\sin 2t + 3.6858\sin 3t \\ & -12.47\sin 4t \end{aligned}$$

where:

$$t = \frac{2p \cdot J}{366} + 4.8718$$

The hour angle (x) is the azimuth angle between the sun position and the solar noon.

$$TST = (LT_h - 12) \cdot 60 + LT_m + \frac{ET}{60}$$

$$x = \frac{TST}{4} + l - l_{TZ}$$

The position of the sun can now be calculated from:

$$\sin(\gamma_s) = \sin(\delta_s) \cdot \sin(\varphi) + \cos(\delta_s) \cdot \cos(\varphi) \cdot \cos(x)$$

$$\cos(\alpha_s) = \frac{\sin(\delta_s) \cdot \cos(\varphi) - \cos(\delta_s) \cdot \sin(\varphi) \cdot \cos(x)}{\cos(\delta_s)}$$

where α is the azimuth angle (measured from north towards the east), and γ is the altitude angle (measured up from the horizon).

The angle between the sun and any point on the sky dome is given by:

$$q = \sin\gamma \cdot \sin\gamma_s + \cos\alpha \cdot \cos\alpha_s \cdot \cos(\alpha - \alpha_s)$$

Normal Incident Extraterrestrial Irradiance

$$E_{esh} = 1367(1 + 0.0334\cos(n - 3.5) + 0.000721\cos(2n - 6.9) - 0.000023\cos(3n - 10.15))$$

where

$$n = \frac{360J}{365}$$

CIE Standard Overcast Sky (CIE, 1990)

The luminance at the point (α,γ) by the formula:

$$L_g = \frac{L_Z(1 + 2\sin\gamma)}{3}$$

and with

$$L_Z = 123 + 8600\sin\gamma$$

CIE Standard Clear Sky (CIE, 1973)

The luminance at the point (α,γ) by the formula:

$$L_{(a,g)} = \frac{L_Z(1-e^{-0.32\sec z})\cdot(0.91+10e^{-3q}+0.45\cos^2 q)}{(1-e^{-0.32})\cdot(0.91+10e^{-3e_s}+0.45\cos^2 e_s)}$$

where:

$L_{(\alpha,\gamma)}$ is the luminance at (α,γ)

θ is the angle between the point at (α,γ) and the sun

ε_S is the zenith angle of the sun

and with

$$b = \frac{g+85}{39.5e^{-w}+47.4} + 0.1 + (16+0.22w)\cdot T$$

$$L_Z = (1340b-3460)\cdot\tan g + 100b + 900$$

where:

ω is the atmospheric moisture content (taken as 1.21 cm)

T is the turbidity (taken as 0.12)

CIE Intermediate Sky

The luminance at the point (θ,ϕ) is given by the formula:

$$L_{(a,g)} = \frac{Ae^{Bq}}{L_Z}$$

with

$$A = 0.43(\gamma + 4.799 + 1.35(\sin(3.59\,\gamma - 0.009) + 2.31)\cdot\sin(2.6\,\gamma + 0.316))$$
$$B = -0.563((\gamma + 1.059)\cdot(\gamma - 0.008) + 0.812)$$

and

$$L_Z = 0.988(\sin(2.60\,\gamma + 0.316) + 2.772))\cdot e^{1.481(\gamma + 0.301)\cdot(\gamma - \frac{\pi}{2})}$$

Perez All-Weather Model (Perez, et al, 1992)

The luminance at the point (α,γ) by the formula:

$$L_{(\alpha,\gamma)} = \frac{L_Z(1 + Ae^{B\sec\varsigma})\cdot(1 + Ce^{D\theta} + E\cos^2\theta)}{(1 + Ae^{B})\cdot(1 + Ce^{D\varepsilon_s} + E\cos^2\varepsilon_s)}$$

where:

$L_{(\alpha,\gamma)}$ is the luminance at (α,γ)

θ is the angle between the point at (α,γ) and the sun

ε_S is the zenith angle of the sun

A,B,C,
D & E are the insolation coefficients as described below:

The Perez All-Weather model consists of a generalization of the CIE Standard Clear Sky with several adjustable coefficients, whose values represent the insolation conditions of the sky. The effect of each coefficient on the sky luminance distribution is outlined below.

Coefficient *a* determines the amount of darkening (>0) or brightening (<0) of the horizon with respect to the zenith. The magnitude of the horizon-zenith gradient is proportional to the value of *A*

Coefficient *B* controls the luminance gradient near the horizon. The larger the negative value of *B* the thicker the area of the horizon effected by *E* will be.

The relative intensity of the circumsolar region is proportional to the magnitude of*A*

The width of the circumsolar region is determined by D The larger the negative value of D the thinner the circumsolar region affected by C will be.

Coefficient e accounts for the relative intensity of backscattered light received.

The CIE Standard Clear Sky can be obtained by assigning:

A = -1
B = -0.32
C = 10
D = -3
E = 0.45

In order to calculate the insolation coefficients we need to know the sky clearness (η) and brightness (Δ) and the solar zenith angle (Z).

$$h = \frac{\left(\frac{E_{ed} + E_{es}}{E_{ed}} + 1.041 e_s^3 \right)}{\left(1 + 1.041 e_s^3 \right)}$$

$$\Delta = \frac{m E_{ed}}{E_{es_0}}$$

where:

E_{ed} is the horizontal diffuse irradiance
E_{es} is the normal incident direct irradiance
E_{es0} is the normal incident extraterrestrial irradiance
m is the optical air mass

The coefficients can then be calculated from the following tables using the following formulae:

$$X = X_1(h) + X_2(h) e_s + \Delta\left(X_3(h) + X_4(h) e_s \right)$$

There are two exceptions to this formula for coefficients C and D in the first ε interval:

$$C = e^{(\Delta(C_1 e_s + C_2))^{C_3}} - C_4$$

$$D = -e^{\Delta(D_1 e_s + D_2)} + D_3 + \Delta D_4$$

D	1.000	1.065	1.230	1.500	1.950	2.800	4.500	6.200
A_1	1.3525	-1.2219	-1.1000	-0.5484	-0.6000	-1.0156	-1.0000	-1.0500
A_2	-0.2576	-0.7730	-0.2515	-0.6654	-0.3566	-0.3670	0.0211	0.0289
A_3	-0.2690	1.4148	0.8952	-0.2672	-2.5000	1.0078	0.5025	0.4260
A_4	-1.4366	1.1016	0.0156	0.7117	2.3250	1.4051	-0.5119	0.3590
B_1	-0.7670	-0.2054	0.2782	0.7234	0.2937	0.2875	-0.3000	-0.3250
B_2	0.0007	0.0367	-0.1812	-0.6219	0.0496	-0.5328	0.1922	0.1156
B_3	1.2734	-3.9128	-4.5000	-5.6812	-5.6812	-3.8500	0.7023	0.7781
B_4	-0.1233	0.9156	1.1766	2.6297	1.8415	3.3750	-1.6317	0.0025
C_1	2.8000	6.9750	24.7219	33.3389	21.0000	14.0000	19.0000	31.0625
C_2	0.6004	0.1774	-13.0812	-18.3000	-4.7656	-0.9999	-5.0000	-14.5000
C_3	1.2375	6.4477	-37.7000	-62.2500	-21.5906	-7.1406	1.2438	-46.1148
C_4	1.0000	-0.1239	34.8438	52.0781	7.2492	7.5469	-1.9094	55.3750
D_1	1.8734	-1.5798	-5.0000	-3.5000	-3.5000	-3.4000	-4.0000	-7.2312
D_2	0.6297	-0.5081	1.5218	0.0016	-0.1554	-0.1078	0.0250	0.4050
D_3	0.9738	-1.7812	3.9229	1.1477	1.4062	-1.0750	0.3844	13.3500
D_4	0.2809	0.1080	-2.6204	0.1062	0.3988	1.5702	0.2656	0.6234
E_1	0.0356	0.2624	-0.0156	0.4659	0.0032	-0.0672	1.0468	1.5000
E_2	-0.1246	0.0672	0.1597	-0.3296	0.0766	0.4016	-0.3788	-0.6426
E_3	-0.5718	-0.2190	0.4199	-0.0876	-0.0656	0.3017	-2.4517	1.8564
E_4	0.9938	-0.4285	-0.5562	-0.0329	-0.1294	-0.4844	1.4656	0.5636

Harrison Sky Model (Harrison, 1991)

The luminance at the point (α,γ) by the formula:

$$L_{(a,g)} = L_Z \cdot C\left(0.40 + 0.21(90 - g) + 0.27\cos g_c + 1.45e^{-2.41q}\right)$$
$$+(1-C)\left((1.28 + 147e^{-11.1q} + 4.28\cos^2 q \cdot \cos(90 - g)) \cdot \left(1 - e^{\frac{-0.42}{\cos g}}\right) \cdot \left(1 - e^{\frac{-0.67}{\cos g}}\right)\right)$$

where

C is the cloud cover (octas)

T is the turbidity (= 3.2)

and

$$L_Z = (1340T - 3460) \cdot \tan g + 100T + 900$$

$$g_s = \frac{p}{2} - g$$

BRE Average Sky Model (Littlefair, 1981)

The luminance at the point (θ,ϕ) by the formula:

$$L_{(a,g)} = Ae^{\frac{-2qp}{9}} + B\frac{5 - 2\sin g}{3}$$

with

$$A = 100 + 420\, g - 700\sin(7.2\, g)$$

$$B = \frac{9}{11p}(300 + 434\, g - 4.2\, g^2)$$

Nakamura Intermediate Sky Model (Nakamura et al, 1987)

The relative luminance at the point (α,γ) is given by:

$$Lrin_{(g_s,g,q)} = \frac{L_{(g_s,g,q)}}{L_{(g_s,\frac{p}{2},\frac{p}{2}-g_s)}}$$

where

$$L_{(g,gq)} = Ae^{Bq}$$

$$A = 0.43(g + 1.4799 + 1.35(\sin(3.59\, g - 0.009) + 2.31) \cdot \sin(2.60\, g + 0.316))$$

$$B = -0.563((g + 1.059)(g_s - 0.008) + 0.812)$$

$$L_{(g_s,\frac{p}{2},\frac{p}{2}-g_s)} = (0.988(\sin(2.60\, g_s + 3.16) + 2.772)) \cdot e^{(1.481 g_s + 0.301)(g_s - 1.571)}$$

where γ_s is the solar altitude.

Appendix B. Commonly Used SDF Parameter Types

Parameter Description	Items included	Name	Format
Source of data	text description	"src"	"%s"
Site description	long°, long', lat°, lat',ref_long°	"site"	"%d%c%d%c%d"
Time specification	year, month, day, hour, minute	"time"	"%c%c%c%c%c"
Number of contour intervals	number of contours	"num"	"%c"
Cloud coverage	cloud coverage in octas	"octa"	"%c"
Sun position	az°, az', alt°, alt'	"sun"	"%c%c%c%c"
Iso-luminance contour	set of polylines	"iso"	"%p"
Discontinutity edge	set of polygons	"edge"	"%e"

The format symbols are:

%c a single byte value

%d a two byte value

%s a null terminated string of characters

%p iso-luminance contour (internal use only)

%e edge (internal use only)

Appendix C. Descriptions of SDF Functions

Included in Library *sdf*

```
SDF(3)                  C Library Functions                  SDF(3)


NAME
     sdf - standard digital form file format

DESCRIPTION
     The standard digital form is used as a means of representing
     models of the luminance of the sky from a variety of
     sources.  A file is a (possibly empty) collection of sky
     models, where each model consists of a 2 byte quantity
     specifying the number of parameters in the model, followed
     by the parameters themselves.

     Each parameter is defined as follows:

     - A 2 byte quantity specifying the size of the parameter.

     - size bytes containing a unique key used to identify the
       parameter, and the parameter data.

     It is the responsibility of the user to know the exact for-
     mat of each parameter.  However, a number of standard param-
     eters are already defined:

     <site> ::= site <longitude> <latitude> <reference longitude>

          <longitude> ::= <degrees> <minutes>

          <latitude> ::= <degrees> <minutes>

     <source> ::= src <source>

     <time stamp> ::= time <date> <time>

          <date> ::= <year> <month> <day>

          <time> ::= <hours> <minutes>

     <sun position> ::= sun <azmiuth> <altitude>

          <azimuth> ::= <degrees> <minutes>

          <altitude> ::= <degrees> <minutes>

     <contour list> ::= iso <size> { <contour> }

          <contour> ::= <level> <size> { <point> }

          <point> ::= <azimuth> <altitude>

     <edge list> ::= edge <size> { <edge> }

          <edge> ::= <level> <size> { <point> }
```

```
          <point> ::= <internal> <external> <azimuth> <altitude>

SEE ALSO
     sdfOpenFile(3),      sdfCloseFile(3),       sdfLoadModel(3),
     sdfStoreModel(3)

AUTHOR
     Graeme Reid <graemer@cs.murdoch.edu.au>
```

NAME

sdfCreateModel, sdfFreeModel, sdfFreeIsoline - create, or free a standard digital model

SYNOPSIS

```
#include <sdf.h>

cc [ flag ... ] file ...  -lsdf -lm [ library ... ]

Model sdfCreateModel()

void sdfFreeModel(model)
Model model;

void sdfFreeIsoline(isoline)
Isoline isoline;
```

DESCRIPTION

sdfCreateModel() creates a new (empty) model and returns a pointer to be used to identify the model in subsequent operations.

sdfFreeModel() releases the memory occupied by each of the parameters in the named model using free(). The contents of this model are then lost.

sdfFreeIsoline() releases the memory occupied by an isoline parameter using free().

SEE ALSO

free(3X), sdfGetParameter(3)

AUTHOR

Graeme Reid <graemer@cs.murdoch.edu.au>

NAME

sdfLoadModel, sdfStoreModel - load or store a standard digital model

SYNOPSIS

```
#include <sdf.h>

cc [ flag ... ] file ...  -lsdf -lm [ library ... ]

int sdfLoadModel(model,stream)
Model model;
FILE *stream;

int sdfStoreModel(model,stream)
Model model;
FILE *stream;
```

DESCRIPTION

sdfLoadModel() loads the parameters of the next standard digital model from the named stream in the named model.

sdfStoreModel() stores a standard digital model to the named stream.

Both sdfLoadModel() and sdfStoreModel() return 1 on success and 0 on failure.

SEE ALSO

free(3X), sdfOpenFile(3), sdfCloseFile(3)

AUTHOR

Graeme Reid <graemer@cs.murdoch.edu.au>

NAME

sdfOpenFile, sdfCloseFile - open or close a standard digital form data file

SYNOPSIS

```
#include <sdf.h>

cc [ flags ... ] file ...  -lsdf -lm [ library ... ]

FILE *sdfOpenFile(filename,mode)
char *filename,*mode;

void sdfCloseFile(stream)
FILE *stream;
```

DESCRIPTION

sdfOpenFile() opens the file named by filename using fopen() and associates a stream with it. If the open succeeds, sdfOpenFile() returns a pointer to be used to identify the stream in subsequent operations.

sdfCloseFile() closes the named stream using fclose().

SEE ALSO

fopen(3S), fclose(3S), sdfLoadModel(3), sdfStoreModel(3)

AUTHOR

Graeme Reid <graemer@cs.murdoch.edu.au>

NAME

sdfPointLuminance, sdfAreaLuminance - calculate the sky luminance for a point or region on the sky dome

SYNOPSIS

```
#include <sdf.h>

cc [ flags ... ] file ...  -lsdf -lm [ library ... ]

int sdfPointLuminance(model,point)
Model model;
Point point;

double sdfAreaLuminance(model,region,area)
Model model;
Point region;
double *area;
```

DESCRIPTION

sdfPointLuminance() returns the interpolated luminance at the specified point, for the given model.

sdfAreaLuminance() returns the interpolated total luminance of the specified region, for the given model. The area parameter is set to the area of the specified region.

AUTHOR

Graeme Reid <graemer@cs.murdoch.edu.au>

NAME

sdfSetParameter, sdfGetParameter, sdfDeleteParameter - set, get or delete a parameter from a standard digital model

SYNOPSIS

```
#include <sdf.h>

cc [ flags ... ] file ... -lsdf -lm [ library ... ]

sdfSetParameter(model,key,format,...)
Model model;
char *key,*format;

sdfGetParameter(model,key,format,...)
Model model;
char *key,*format;

int sdfDeleteParameter(model,key)
Model model;
char *key;
```

DESCRIPTION

sdfSetParameter() adds a parameter, identified by the key, to the named model. sdfSetParameter() converts and saves its args under control of the format. The format is a character string which contains two types of objects: plain characters, which are simply saved to the parameter, and conversion specifications, each of which causes conversion and saving of zero of more args. The results are undefined if there are insufficient args for the format. If the format is exhausted while args remain, the excess args are simply ignored.

Each conversion specification is introduced by a % character, followed by a conversion character. The meanings of the various conversion characters are defined as follows:

c - Treat the argument as a (1 byte) character.

d - Treat the argument as an (2 byte) integer.

s - Treat the argument as a NULL terminated string.

p - Treat the argument as a point list.

e - Treat the argument as an edge list.

% - Save a % character: no argument is converted.

sdfGetParameter() retrieves a parameter, identified by the key, from the named model. The contents of the parameter are converted under the control of the format string, and the variables in the args list set accordingly.

sdfDeleteParameter() deletes a parameter, identified by the key, from the named model.

sdfSetParameter(), sdfGetParameter() and sdfDeleteParameter() return 1 if the named parameter existed before the operation, and 0 otherwise.

```
SEE ALSO
     sdfCreateModel(3), sdfLoadModel(3), sdfStoreModel(3)

AUTHOR
     Graeme Reid <graemer@cs.murdoch.edu.au>
```

NAME

sdfExtractInside, sdfExtractOutside - extract regions from a model

SYNOPSIS

```
cc [ flag ... ] file ...  -lsdfops -lsdf -lm [ library ... ]

#include <sdf.h>
#include <sdfops.h>

Model sdfExtractInside(model)
Model model;

Model sdfExtractOutside(model)
Model model;
```

DESCRIPTION

sdfExtractInside() creates a new model from the given model containing only the regions inside a defined edge.

sdfExtractOutside() creates a new model from the given model containing only the regions outside a defined edge.

A region is defined to be any area of the sky enclosed by an edge. If the sky contains no edge information (other than the edge which defines the boundary of the sky dome) it is considered to be one region. Note that edges must be simple closed polygons.

SEE ALSO

sdfOverlay(3)

AUTHOR

Graeme Reid <graemer@cs.murdoch.edu.au>

NAME

sdfImport - import a measured or empirical model

SYNOPSIS

```
cc [ flag ... ] file ...  -lsdfops -lsdf -lm [ library ... ]

#include <sdf.h>
#include <sdfops.h>

Model sdfImport(type,template,match,fp)
int type;
Model template;
int *match();
FILE *fp;
```

DESCRIPTION

sdfImport() converts a measured or empirical model to standard digital form and returns a pointer to the new model. type determines how the conversion is done. Currently supported types are : Currently supported types are:

STANDARD_DIGITAL_FORM - use sdfLoadModel to load a model in standard digital form.

TEXT_FILE - convert a generic text file as defined below.

PRC_SCANNER - convert a text output file from the PRC Krochman Scanner in Sydney.

PEREZ_ALL_WEATHER - calculate the Perez All Weather sky based on the given weather file.

CIE_CLEAR - calculate the CIE Clear sky.

CIE_OVERCAST - calculate the CIE Overcast sky.

HARRISON - calculate the Harrison sky.

NAKAMURA - caluculate the Namkamura Intermediate sky.

SEE ALSO

sdfCreateModel(3) sdfLoadModel(3)

AUTHOR

Graeme Reid <graemer@cs.murdoch.edu.au>

Included in Library *sdfops*

```
SDF(3)                  C Library Functions                  SDF(3)


NAME
     sdfNormalise, sdfScale - scale the  luminance  values  of  a
     model

SYNOPSIS
     cc [ flag ... ] file ...  -lsdfops -lsdf -lm [ library ... ]

     #include <sdf.h>
     #include <sdfops.h>

     Model sdfNormalise(model)
     Model model;

     Model sdfScale(model,scale)
     Model model;
     double scale;

DESCRIPTION
     sdfNormalise() creates a new  model  by  scaling  the  given
     model  in  such a way that the zenith luminance is one.  The
     zenith luminance is calculated  by  sdfPointLuminance()  and
     then  inverted and passed to sdfScale() along with the given
     model.

     sdfScale() creates a new model from the given model by scal-
     ing  the  luminance  by the specified scale factor.  This is
     equivalent to calling sdfProduct() with the given model  and
     a uniform sky.

SEE ALSO
     sdfPointLuminance(3) sdfProduct(3)

AUTHOR
     Graeme Reid <graemer@cs.murdoch.edu.au>
```

NAME

sdfOverlay - place one model on top of another

SYNOPSIS

```
cc [ flag ... ] file ...  -lsdfops -lsdf -lm [ library ... ]

#include <sdf.h>
#include <sdfops.h>

Model sdfOverlay(model1,model2)
Model model1,model2;
```

DESCRIPTION

sdfOverlay() combines two models by super-imposing the first model on top of the second in an opaque fashion. Any parameters which occur in both the first and second models take on the value of the first model, with the exception of luminance data. Where a region defined in the first model has luminance information, it will be used, but if there are regions which have no luminance data, they will be treated as transparent and replaced with data from the second model. This function is primarily designed to facilitate adding a region of cloud to an otherwise clear sky.

A region is defined to be any area of the sky enclosed by an edge. If the sky contains no edge information (other than the edge which defines the boundary of the sky dome) it is considered to be one region. Note that edges must be simple closed polygons.

SEE ALSO

sdfSum(3) sdfDifference(3) sdfProduct(3) sdfRatio(3)

AUTHOR

Graeme Reid <graemer@cs.murdoch.edu.au>

NAME

sdfSum, sdfDifference, sdfProduct, sdfRatio - combine two models arithmetically

SYNOPSIS

```
cc [ flag ... ] file ...  -lsdfops -lsdf -lm [ library ... ]

#include <sdf.h>
#include <sdfops.h>

Model sdfSum(model1,model2)
Model model1,model2;

Model sdfDifference(model1,model2)
Model model1,model2;

Model sdfProduct(model1,model2)
Model model1,model2;

Model sdfRatio(model1,model2)
Model model1,model2;
```

DESCRIPTION

These functions combine two models by performing an arithmetic calculation on the luminance data. As for the sdfOverlay() function, any parameters which occur in both the first and second models take on the value of the first model, with the exception of luminance data. This is combined arithmetically to produce the new luminance information. The operations used should be obvious from the name of each function. sdfRatio() avoids dividing by zero by first checking the luminance values. If no luminance value is defined for a region in the second model, the resulting luminance will also be undefined for this region.

SEE ALSO

sdfOverlay(3)

AUTHOR

Graeme Reid <graemer@cs.murdoch.edu.au>

Appendix D. Format for Text File Model Input

The Krockmann PRC sky scanner is a relatively common instrument so we have encoded its particular data format into the SDF function library. Other scanners will undoubtably have different data formats and would have to be encoded to suit each case. There is, however, a general purpose formatting process which allows data from any source to be directly input and then converted to SDF.

This is achieved via the text file input option which has the following format:

```
site 151 12 -33 54 150
time 92 7 7 12 0
iso 17
  0    0 1200
  0   45 1300
  0   90 1100
  0  135  900
 45    0 1100
 45   45 1300
 45  180 2500
 90    0  500
 90   45  600
 90  135 2500
 90  180 3000
135    0  700
135   45  800
135   45  800
135  135 2300
135  180 2800
```

The text file should contain at least:

- a site parameter description, ie longitude (deg), longitude (min), latitude (deg) latitude (min) and reference longitude (deg).

- a time parameter description, ie year, month, day, hour, minute

- a isoline parameter, with the number of following lines. Each isolinepoint is defined by azimuth and altitude angles (deg), and luminance value.

Optionally the text file can also include an edge which is used to either better define the horizon boundary or to include a cloud/obstruction edge. In this case the edge must be a closed polygon defined by pairs of azimuth-altitude values together with the inside and outside luminance values. For the horizon edge the outside values are set as zero, eg.

```
site 151 12 -33 54 150
time 92 7 7 12 0
iso 17
  0    0 1200
  0   45 1300
  0   90 1100
  0  135  900
 45    0 1100
 45   45 1300
 45  180 2500
 90    0  500
 90   45  600
 90  135 2500
 90  180 3000
135    0  700
135   45  800
135   45  800
135  135 2300
135  180 2800
edge 0 9
  0    0 1200 0
 45    0 1100 0
 90    0  900 0
135    0  700 0
180    0 1000 0
 45  180 2500 0
 90  180 3000 0
135  180 2800 0
  0    0 1200 0
```

In this case the edge is of type 0 (ie the horizon edge) and contains 9 vertices (counting first and last - which are the same point). An edge to define a discontinuity in the sky luminance would be declared as a type 1 edge. Several type 1 edges may be included, but only one type 0 edge.

To demonstrate this type of model development, we have been able to take data provided the Nakamura Laboratory (Kyushu University) from scanner records on the 9th Jan 1994 at Fukuoka, Japan (Latitude: 33° 31' N, Lat: 130° 29' E) at 0900 hours. The data was prepared in the format described above, then imported into SKYMAP using the text file option. The resulting SDF model is then built, and displayed as in Figure D.1.

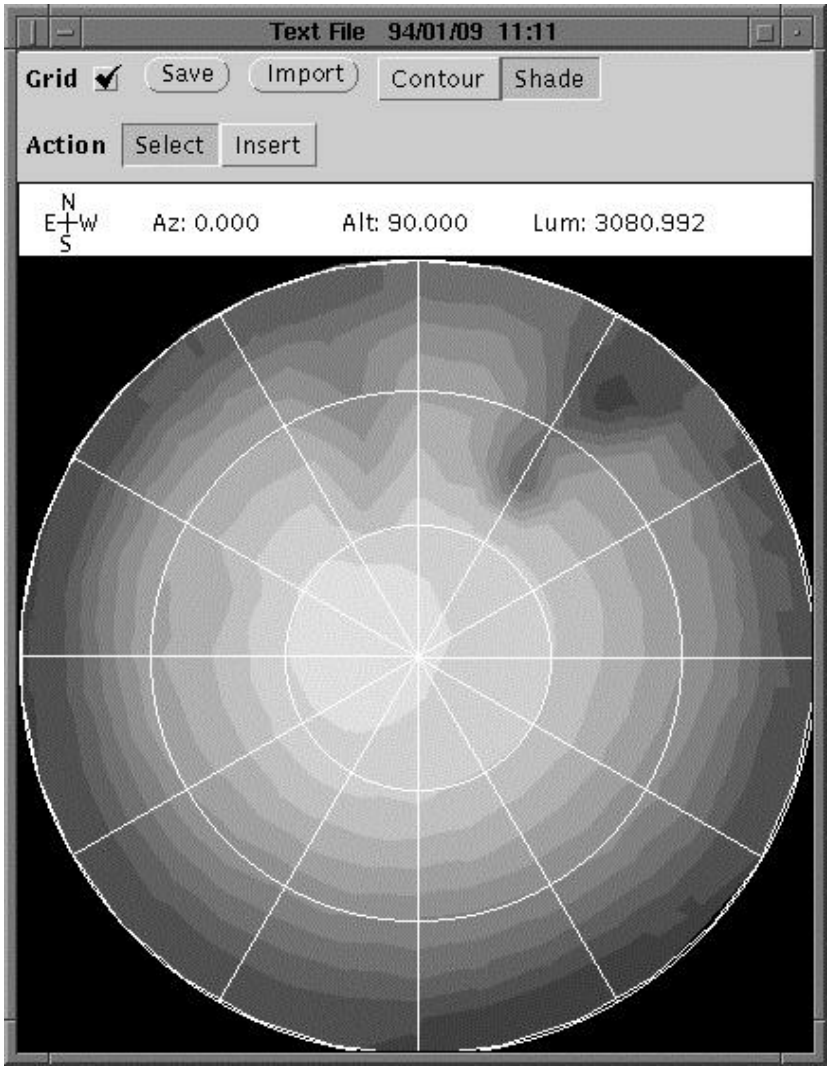

Figure D.1: The SDF model formed from data from the Nakamura Laboratory

Appendix E. Programming AEM Models

Below is an example of the function required to implement a new AEM in the SDF framework. The function must include an explicit description of the luminance model (Harrison in this case). The program constructs the iso-luminance contours based on the Tregenza grid of 145 points. The program also adds an horizon edge into the model to facilitate interpolation near the horizon.

The required parameters (site, time, sun, and octa) are set before this function is called and are passed in the sky model (*model*) included as the argument to the function. The new parameters (iso and edge) are set in this function before the updates model is returned.

```
#include "sdfopsP.h"
#include "import.h"

/* ------------------------------------------------------------    *
 *    harrison calculates a Harrison Sky for the site and    *
 *    time specified in the given model (template).          *
 * ------------------------------------------------------------    */

Model harrison(
    Model   template)
{
    Model         model;
    Isoline       isoline,
                  edge,
                  iso;
    Point         point;
    int                 azi,azi_min,
                  alt,alt_min,
                  octa;
    double        cloud,
                  alpha,
                  turbidity,
                  zenith,
                  cospsi,
                  vo,vc;

    isoline = grid();
    edge    = (Isoline)Malloc(sizeof(struct _isoline));
    edge->value   = (double)0;
    edge->point   = border();
    edge->next    = (Isoline)NULL;
    (void)sdfGetParameter(template,"edge","%e",&edge->next);
    model = sdfCreateModel();
    _combine_models(template,model);
    _sun_position(model);
    (void)sdfSetParameter(model,"src","%s","Harrison");

(void)sdfGetParameter(model,"sun","%c%c%c%c",&azi,&azi_min,&alt,&alt_min)
;
    (void)sdfGetParameter(model,"octa","%c",&octa);
    cloud   = (double)octa / 8;
    turbidity     = 3.2;
    zenith  = (1340 * turbidity - 3460) * abs(tan(RADIANS(alt)))
```

```
            + 100 * turbidity + 900;
    for (iso = isoline ; iso != (Isoline)NULL ; iso = next(iso)) {
      cospsi            = sin(altitude(iso->point)) * sin(RADIANS(alt))
                  + cos(altitude(iso->point)) * cos(RADIANS(alt))
                  * cos(azimuth(iso->point) - RADIANS(azi));
      alpha       = abs(M_PI / 2 - altitude(iso->point));
      vo          = 0.40 + 0.21 * RADIANS(abs(90 - alt))
                  + 0.27 * cos(alpha) + 1.45 * exp(-2.41 * acos(cospsi));
      vc          = (1.28 + 147 * exp(-11.1 * acos(cospsi))
                  + 4.28 * cospsi * cospsi * cos(RADIANS(abs(90 - alt))))
                  * (1 - exp(-0.42 / cos(alpha)))
                  * (1 - exp(-0.67 / cos(RADIANS(abs(90 - alt)))));
      iso->value  = zenith * (cloud * vo + (1 - cloud) * vc);
    }
    (void)sdfSetParameter(template,"iso","%p",isoline);
    sdfFreeIsoline(isoline);
    for (point = edge->point ; point != (Point)NULL ; point =
next(point)) {
      cospsi            = sin(altitude(point)) * sin(RADIANS(alt))
                  + cos(altitude(point)) * cos(RADIANS(alt))
                  * cos(azimuth(point) - RADIANS(azi));
      alpha       = abs(M_PI / 2 - altitude(point));
      vo          = 0.40 + 0.21 * RADIANS(abs(90 - alt))
                  + 0.27 * cos(alpha) + 1.45 * exp(-2.41 * acos(cospsi));
      vc          = (1.28 + 147 * exp(-11.1 * acos(cospsi))
                  + 4.28 * cospsi * cospsi * cos(RADIANS(abs(90 - alt))))
                  * (1 - exp(-0.42 / cos(alpha)))
                  * (1 - exp(-0.67 / cos(RADIANS(abs(90 - alt)))));
      point->value      = zenith * (cloud * vo + (1 - cloud) * vc);
      point->external   = (double)0;
    }
    for (iso = edge ; next(iso) != (Isoline)NULL ; iso = next(iso)) {
      if (iso->next->value == 0) {
          isoline = next(iso);
          iso->next = next(isoline);
          isoline->next = (Isoline)NULL;
          sdfFreeIsoline(isoline);
          break;
      }
    }
    (void)sdfSetParameter(template,"edge","%e",edge);
    sdfFreeIsoline(edge);
    return(model);
}
```

Appendix F. CIE Terminology

Definitions of all terms and symbols commonly used in the study and practice of illumination can be found in the International Lighting Vocabulary, 4th Edition., C.I.E., Pub. No. 17.4, 1987.

Solar Radiation

Electromagnetic radiation from the Sun.

Extraterrestrial Solar Radiation

Solar radiation incident at the outer limit of the Earth's atmosphere.

Solar Constant

Irradiance of extraterrestrial solar radiation perpendicular to the Sun's rays at mean Sun-Earth distance: E_{eo} = 1367 W/m^2

Direct Solar Radiation

That part of extraterrestrial solar radiation which as a single collimated beam reaches the Earth's surface after selective attenuation by the atmosphere. If not stated otherwise, direct beam radiation refers to radiation incident on a plane normal to the direction of incidence.

Diffuse Sky Radiation

Solar radiation which reaches the Earth as a result of being scattered by the air molecules, aerosol particles, cloud and other particles of the atmosphere. If not stated otherwise, diffuse sky radiation refers to radiation received on a horizontal plane from the whole hemisphere.

Global Solar Radiation

Sum of direct beam radiation and diffuse sky radiation. If not stated otherwise global radiation refers to radiation incident on a horizontal plane.

Reflected Solar Radiation

Global solar radiation reflected from the surface of the Earth and by any surface intercepting that radiation.

Turbidity Factor (according to Linke)

Ratio of the vertical optical thickness of a turbid atmosphere to the vertical optical thickness of a clean and dry atmosphere (Rayleigh atmosphere), as related to the whole solar spectrum:

$$T_L = \frac{d}{d_R} = \frac{d_R + d_A + d_Z + d_W}{d_R}$$

where

δ_R = optical thickness with respect to Rayleigh scattering and absorption of the aerosol particles.

δ_A = optical thickness with respect to Mie scattering and absorption of the aerosol particles.

δ_Z = optical thickness with respect to ozone absorption.

δ_W = optical thickness of the atmosphere with respect to water vapour absorption..

Optical thickness of the atmosphere

Quantity defined by the formula

$$\delta(\varepsilon) = -\ln (\phi'_e/\phi_e)$$

where ϕ_e is the radiant flux of a collimated beam entering the upper limit layers of the atmosphere at an angle ε to the vertical and ϕ'_e the attenuated radiant flux of that beam reaching the surface of the Earth.

Relative Optical Air Mass

Ratio of the slant optical thickness to the vertical optical thickness of the atmosphere:

$$m = \frac{d(e)}{d(o)}$$

where: ε = zenith angle, and o = in zenith

Daylight

Visible part of global solar radiation. Daylight is the sum of sunlight and skylight.

Sunlight

Visible part of direct solar radiation.

Skylight

Visible part of diffuse sky radiation.

Global Illuminance

Illuminance from the sun and sky received on a horizontal plane from the whole hemisphere unless stated otherwise.

Diffuse Illuminance

Illuminance from the sky received on a horizontal plane from the whole hemisphere unless stated otherwise.

CIE Standard Overcast Sky

Completely overcast sky for which the ratio of its luminance L_γ at an angle of elevation γ above the horizon to the luminance L_Z at the zenith is assumed to be given by the relation :

$$L_g = \frac{L_Z(1 + 2\sin \gamma)}{3}$$

CIE Standard Clear Sky

Cloudless sky for which the relative luminance distribution is described in Publication CIE No. 22 (TC.4.2) 1973.

Total Cloud Amount

Ratio of the summated solid angles subtended by clouds to the solid angle 2π steradians of the whole sky.

Sunshine Duration

The sum of time intervals within a given time period (hour, day, month, year) during which the irradiance from direct solar radiation on a plane normal to the sun direction is equal to or greater than 120 W/m^2 (about 18,000 lx).

Note: This irradiance is the minimum required to produce a burned trace on the standardized card of International Reference Sunshine Recorder (IRSR) of the World Meteorological Organization (WMO).

Astronomical Sunshine Duration

The sum of time intervals within a given time period during which the sun is above an even, unobscured horizon.

Possible Sunshine Duration

The sum of time intervals within a given time period during which the sun is above the real horizon which may be obscured by mountains, buildings, trees etc.

Relative Sunshine Duration

Ratio of sunshine duration to possible sunshine duration within the same time period.

Window Factor

Ratio of the average vertical illuminance on the window plane from the sky and the ground to the total outdoor illuminance on a horizontal plane.

Daylight Factor

Ratio of the illuminance at a point on a given plane due to the light received directly or indirectly from a sky of assumed or known luminance distribution, to the illuminance on a horizontal plane due to an unobstructed hemisphere of this sky. Direct sunlight is excluded in both illuminances.

Note: When calculating the lighting of interiors, the contribution of direct sunlight must be considered separately.

Sky Component of the Daylight Factor

Ratio of that part of the illuminance at a point on a given plane which is received directly from a sky of assumed or known luminance distribution, to the illuminance on a horizontal

plane due to an unobstructed hemisphere of this sky. Direct sunlight to both illuminances is excluded.

Externally Reflected Component of the Daylight Factor

Ratio of that part of the illuminance at a point on a given plane in an interior which is received directly from external reflecting surfaces, illuminated directly or indirectly by a sky of assumed or known luminance distribution, to the illuminance on a horizontal plane due to an unobstructed hemisphere of this sky. Direct sunlight to both illuminances is excluded.

Internally Reflected Component of the Daylight Factor

Ratio of that part of the illuminance at a point on a given plane which is received directly from internal reflecting surfaces, illuminated directly or indirectly by a sky of assumed or known luminance distribution, to the illuminance on a horizontal plane due to an unobstructed hemisphere of this sky. Direct sunlight to both illuminances is excluded.

Obstruction

Surfaces outside the building which obstruct the direct view of the sky from the reference point.

Daylight Opening

Area (glazed or unglazed) which is capable of admitting daylight to an interior.

Window

Daylight opening on a vertical or nearly vertical area of a room envelope.

Rooflight

Daylight opening on a roof or on a horizontal surface of the building.

Shading device

Device to obstruct, reduce or diffuse the penetration of solar radiation.

Total Transmittance of Glazing Material

Sum of radiant transmittance (τ_e) of the glazing and secondary heat gain (q_i) of the glazing to the interior by long wave IR radiation of the absorbed energy and by convection.

$$g = \tau_e + q_i$$

Appendix G. CIE Standard Symbols

The following subscripts should be used for sun, skylight or radiation

Radiant quantities	-	e
Luminous quantitites	-	v
Direct sun	-	s
Extraterrestrial	-	o
Global	-	g
Diffuse	-	d
Clear sky	-	cl
Overcast sky	-	oc
Average sky	-	av
Partly cloudy sky	-	pc
A tilted surface	-	(α_k, γ_k)
A horizontal surface ($\gamma_k = 0^o$)	-	h
A vertical surface ($\gamma_k = 90^o$)	-	v

$\bar{a}_R$	-	Mean extinction coefficient for dry and clean air (Rayleigh scattering)
a_D	-	Extinction coefficient for aerosol
a'_D	-	Decadic extinction coefficient for aerosol
α	-	Azimuth angle (deg) (Note: specify how it is measured)
α_k	-	Azimuth angle of the normal to a surface (deg) (Note: specify how it is measured)
B	-	Decadic turbidity coefficient (according to Schuepp)
β	-	Turbidity coefficient (according to Angstrom)
γ	-	Altitude above the horizon (deg)
γ_k	-	Angle of slope of a surface from the horizon (deg)
γ_s	-	Altitude of the sun above the horizon (deg)
δ	-	Optical thickness of the atmosphere
δ_s	-	Angle of the solar declination (deg)
D	-	Daylight factor
D_s	-	Sky component of the daylight factor
D_e	-	Externally reflected component of the daylight factor
D_i	-	Internally reflected component of the daylight factor
E_e	-	Irradiance (W/m^2)
E_{eav}	-	Irradiance from an unobstructed average sky on a given surface (W/m^2)

E_{ecl}	-	Irradiance from an unobstructed clear sky on a given surface (W/m^2)
E_{esh}	-	Irradiance from direct sun radiation on a horizontal surface (W/m^2)
E_{esv}	-	Irradiance from direct sun radiation on a vertical surface (W/m^2)
E_{eg}	-	Global irradiance (W/m^2)
E_{eo}	-	Solar constant for the mean distance of the earth from the sun (1367 W/m^2)
E_{eo}	-	Extraterrestrial irradiance $(W/m)^2$
E_{eoc}	-	Irradiance from an unobstructed overcast sky on a given surface (W/m^2)
E_{es}	-	Irradiance from direct sun radiation on a given surface (W/m^2)
$E_{es}(\alpha_k,\gamma_k)$	-	Irradiance from direct sun radiation on a tilted surface (W/m^2)
E_v	-	Illuminance (lx)
E_{vav}	-	Illuminance from an unobstructed average sky on a given surface (lx)
E_{vcl}	-	Illuminance from an unobstructed clear sky on a given surface (lx)
E_{vd}	-	Diffuse illuminance on a given surface (lx)
E_{vdh}	-	Diffuse illuminance on a horizontal surface (1x)
E_{vg}	-	Global illuminance on a given surface (lx)
E_{vo}	-	Sunlight constant (klx)
E_{voc}	-	Illuminance from an unobstructed overcast sky on a given surface (lx)
E_{vs}	-	Illuminance from direct sunlight on a given surface (lx)
E_{vsh}	-	Illuminance from direct sunlight on a horizontal surface (lx)
$E_{vs}(\alpha_k,\gamma_k)$	-	Illuminance from direct sunlight on a tilted surface (lx)
E_{vr}	-	Reflected illuminance (lx)
ET	-	Equation of time
$\boldsymbol{e}$	-	Angular distance from the zenith (deg)
$\boldsymbol{e}_s$	-	Angular distance of the sun from the zenith (deg)
f	-	Window factor
φ	-	Geographical latitude of a station (deg) (positive north of the equator, negative south of the equator)

λ	-	Geographical longitude of a station (deg(h)) (negative value east of Greenwich, positive value west of Greenwich)
λ_{TZ}	-	Geographical longitude of standard meridian for time zone (TZ)(deg(h))(negative value east of Greenwich, positive value west of Greenwich)
GMT	-	Greenwich mean time (TU) (h)
z	-	Height of station above sea level (m)
H_e	-	Radiant exposure (Wh/m^2)
J	-	Day of year (1=January 1. 365 = December 31)
K_o	-	Luminous efficacy of extraterrestrial radiation (lm/W)
K_{cl}	-	Luminous efficacy of clear-sky radiation (lm/W)
K_g	-	Luminous efficacy of global radiation (lm/W)
K_{oc}	-	Luminous efficacy of overcast sky radiation (lm/W)
K_S	-	Luminous efficacy of direct solar radiation at ground level (lm/W)
LCT	-	Local clock time (h), LCT = LT + summer time correction
L_e	-	Radiance ($Wsr^{-1}\ m^{-2}$)
$L_e\ (\varepsilon,\alpha)$	-	Radiance of a sky element at angle ε and azimuth α ($W\ sr^{-1},m^{-2}$)
L_{ez}	-	Radiance of the sky at the zenith ($W/sr,m^2$)
L_v	-	Luminance of the sky (cd/m^2)
$L_v(\varepsilon,\alpha)$	-	Luminance of a sky element at angle ε from the zenith and azimuth α (cd/m^2)
L_{vz}	-	Luminance of the sky at the zenith (cd/m^2)
LT	-	Local time (h,min) LT = GMT + TZ
LT_h	-	Local time (hours)
LT_m	-	Local time (minutes)
m	-	Relative optical air mass
m_d	-	Relative aerosol mass
ρ	-	Reflectance (of the ground) (%)
R	-	Actual distance of the earth from the sun (km)
$\bar{R}$	-	Annual average distance of the earth from the sun (km)
R_{ecl}	-	Irradiance ratio for a clear sky (%) $R_{ecl} = E_{ecl}(\alpha_k,\gamma_k)/E_{eclh}$

R_{eoc}	-	Irradiance ratio for an overcast sky (%), $R_{eoc} = E_{eoc}(\alpha_k,\gamma_k)/E_{eoch}$
R_{vcl}	-	Illuminance ratio for a clear sky (%), $R_{vcl} = E_{vcl}(\alpha_k,\gamma_k)/E_{vclh}$
R_{voc}	-	Illuminance ratio for an overcast sky (%) $R_{voc} = E_{voc}(\alpha_k,\gamma_k)/E_{voch}$
σ_{rel}	-	Relative sunshine duration (%), $s = S/S_o$
σ_d	-	Monthly mean daily relative sunshine duration (%)
σ_h	-	Monthly mean hourly relative sunshine duration (%)
σ_m	-	Monthly mean relative sunshine duration (%)
S	-	Actual sunshine duration (h,min)
S_d	-	Monthly mean daily sunshine duration (h,min)
S_h	-	Monthly mean hourly sunshine duration (min)
S_{dm}	-	Monthly mean sunshine duration (h,min)
S_o	-	Length of the day (Astronomical sunshine duration (h,min)
T_c	-	Correlated color temperature (K)
T_L	-	Turbidity factor (according to Linke)
T_{il}	-	Illuminance turbidity factor
t_{sr}	-	Time of sunrise (h,min)
t_{ss}	-	Time of sunset (h,min)
TLT	-	True local time (h,min)
TST	-	True solar time (h,min) $TST = LT + \lambda + ET$
TZ	-	Time zone (h,min)
ξ	-	Solar hour angle
τ	-	Transmittance (%)
θ	-	Angle between a sky element and the sun (deg)
w	-	Water vapor content of the atmosphere (cm)