Sky Detection in Images for Solar Exposure Prediction

A thesis

submitted in partial fulfilment

of the requirements for the Degree of

Master of Software and Information Technology

at

Lincoln University

by

Nuchjira Laungrungthip

Lincoln University

2008

Abstract of a thesis submitted in partial fulfilment of the requirements for the Degree of Master of Applied Computing

Sky Detection in Images for Solar Exposure Prediction

By Nuchjira Laungrungthip

This project describes a technique for segmenting regions of sky in an image from the remainder of the image. This segmentation technique is part of a method for predicting the solar exposure at a location of interest from a set of photographs. Given the latitude and longitude of the position and the direction and field of view of the camera it is possible to calculate the position of the sun in the image at a particular time on a particular day. If that position is in a sky region of the image then the location will be exposed to the sun at that time.

Critical to the success of this method for determining solar exposure is the image processing used to separate the sky from the rest of the image. This work is concerned with finding a technique which can do this for images taken under different weather conditions. The general approach to separate the sky from the rest of the image is to use the Canny edge detector and the morphology closing algorithm to find the regions in the image. The brightness and area of each region are then used to determine which regions are sky. The FloodFill algorithm is applied to identify all pixels in each sky region.

An extensive empirical study is used to find a set of threshold values for the Canny edge detector, applied to the blue colour channel, which allow successful identification of the sky regions in a wide range of images. Tests using different camera filters show that they do not usefully increase the contrast between the sky and the rest of the image, when a standard compact camera is used. The work reported in this thesis shows that this approach of finding edges to identify possible sky regions works successfully on a wide range of images although there will always be situations, such as when the image is taken directly into the sun, where manual adjustment to the identified regions may be required.

Keywords: Solar exposure, skyline, image processing, image segmentation

Acknowledgements

I really enjoyed this work and I would like to express my gratitude to all who gave me the opportunity to complete this thesis. I would like to thank the Lincoln University Image Processing and Solar Radiation Masters Scholarship and my family which supported this project. I am deeply indebted to my supervisor, Prof. Dr. Alan McKinnon whose stimulating suggestions and encouragement helped me in all the time of research. I also would like to thank my associate supervisors, Dr. Keith Unsworth and Dr. Clare Churcher, for support, interest, and valuable hints. Also, thanks go to the academic, technical and administrative staff who happily helped when it was required. Special, thanks go to Mr. Jatuporn Sutthibunwongchai to help me carry the camera equipment around.

Table of Contents

List of Tables

List of Figures

1 Introduction

The position of the sun in the sky changes continually during the day and seasonally throughout the year. Exposure to the sun, at a particular location at different times of the year, can be affected by obstructions such as hills, trees or buildings.

For this project, "solar exposure at a location" is defined as the ability of a ray from the sun to intersect a particular point at any particular time of the day for any day of the year. Exposure may be determined for a specified time or integrated over a time period such as an hour, day, week, or month. Figure 1.1 shows an example of the solar exposure at a particular location on a specific day at 7.00 am. and 9.00 am. The person is not exposed to the sun at 7.00 am. but he is exposed at 9.00 am.

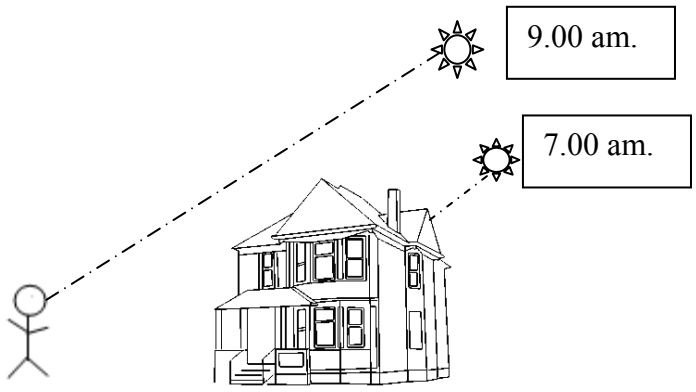


Figure 1.1: The sun can be blocked by obstructions at different times of the day or year.

The sun is very useful in different situations such as drying food for agriculture, heating domestic water or powering a solar house (Energy Information Administration, 2006). Knowing the solar exposure at a particular location would help a builder determine the position where a solar collector or photovoltaic cell would get the most sunlight. It might also assist an architect who wants to set the location of windows so they receive the most sunlight in the morning. The information might also help decision making about where to plant trees and what types of trees to plant in the most suitable places depending on their sunlight requirements.

1.1 Determining Solar Exposure from Images

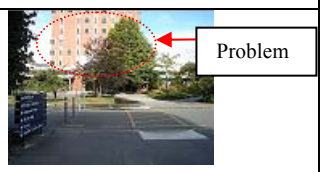
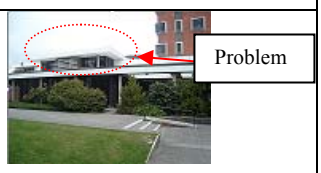
The concept for a device that predicts the solar exposure at a location was developed by McKinnon (McKinnon, 2007) although as discussed in Chapter 2, devices based on the same principles have been developed by others. The following steps are involved:

1. Take a series of digital images from the location of interest in known directions.
2. Find the sky in each image using image processing techniques.
3. Determine whether a ray from the sun can reach the location by passing through the sky part of one of the images.
4. Repeat step 3 for all times of the year of interest.
5. Display the information.

1.2 Difficult Images

There will be different situations that might affect how we separate the sky from the rest of the image. Examples are shown in Table 1.1 below.

Table 1.1: Situations that might affect the image processing techniques used to find the sky.

Description	Images
The image is taken without sky at the top of a building.	
The colour of the object is similar to the colour of the clouds.	
The effect of cloud formations in the sky which may not be uniform.	
Sun flash may obscure boundaries.	

1.3 General Objective

This project seeks to develop a robust image processing technique for identifying sky regions from a variety of images. Correctly identifying the sky regions will enable accurate calculation of the solar exposure at a location at any time of the year.

In this thesis a region is defined as an area of the image bounded by an edge where there is a significant change in contrast. Some regions will be sky but others may not.

1.4 Overall Method for Identifying Sky Regions

Image processing techniques are used to identify sky regions in images obtained from digital cameras. The image processing includes the following steps:

1. Improve the contrast between regions that are potentially sky and non-sky.
2. Determine the boundary line between contrasting regions.
3. Close gaps in the boundary lines.
4. Identify which of the regions are sky.

2 Literature Review

This chapter reviews some previous approaches to determine solar exposure and to identify the sky in digital images. They fall into the following categories:

- Mechanical inventions used to determine solar exposure.
- Computer modelling used to determine solar exposure.
    - Estimating canopy coverage for a golf green.
    - Software for architects or landscape architects, such as Ecotect, Suncast, and GIS.
    - GNS device.
    - The solmetric SunEye device.
    - The Solar Pathfinder^{TM}.
    - Wiley Electronics ASSET.
- Methods for detecting sky in images.

2.1 Mechanical Devices for Determining Solar Exposure

There are a number of patents in existence for mechanical devices that have been devised to determine solar exposure, as described in the following sections.

2.1.1 Owner-Peterson et al.'s sunlight calculator

The device by Owner-Peterson et al. (Owner-Petersen & Lund-Hansen, 1980) calculates solar exposure at various times and locations. It consists of a base portion and two sliding plates (Figure 2.1). The user can slide the sliding plate 1 to adjust for location, date and time. When the base portion and sliding plate 1 are correctly located, the position of the sun can be determined. Then, the sliding plate 2 is adjusted and slid into the base portion to determine the shading for the location. Although it can estimate the shadow caused by an obstruction such as a tree, it is not clear how it is done.

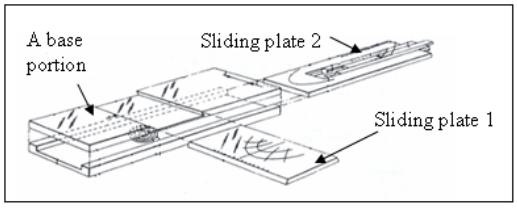


Figure 2.1: Invention by Owner-Peterson et al. to determine the sun exposure (Owner-Petersen & Lund-Hansen, 1980).

2.1.2 Griffin's device for determining sun exposure

The device by Griffin (Griffin Jr, 1980) evaluates solar exposure at different locations and times of the year by viewing through a pointer, as shown in Figure 2.2. The main components are a season adjuster, pointer, time adjuster, latitude adjuster, and base. In order to determine the solar exposure, the pointer will follow the sunpath when the latitude, the season, and the time adjusters are set and the base points north (in southern hemisphere). The pointer is set to the highest sunpath when the season adjuster is set to summer time, and the pointer is set to the lowest sunpath during winter. The different times of the year can be adjusted by moving the season adjuster up and down. The time adjuster allows the user to locate the sun for a particular time of day. The user can then see if the sun is obscured by buildings or trees. Although this device can be used in any location, it needs to be set to point north and does not provide a method for recording the solar exposure or doing any analysis of the information seen through the viewer.

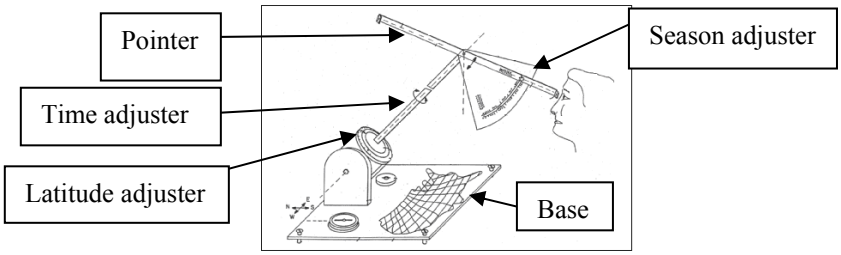


Figure 2.2: Apparatus for evaluation solar exposure due to Griffin (Griffin Jr, 1980).

2.1.3 Lewis's device to indicate solar exposure

The invention by Lewis (Lewis, 1981) uses a wide angle viewer and a transparent screen. The transparent screen has the paths of the sun at various times of the year, for a particular latitude, drawn on it, as shown in Figure 2.3. The user aligns the device with north and looks through the viewer where he or she can see whether the sky is visible at a time and date corresponding to any points of interest on the sunpath lines. If the sky is visible, the location is exposed to the sun at that time. Although this device gives a good visual portrayal of solar exposure, it does not give a quantitative measure of exposure or provide any means of recording the information. Also the orientation of the device needs to be set properly.

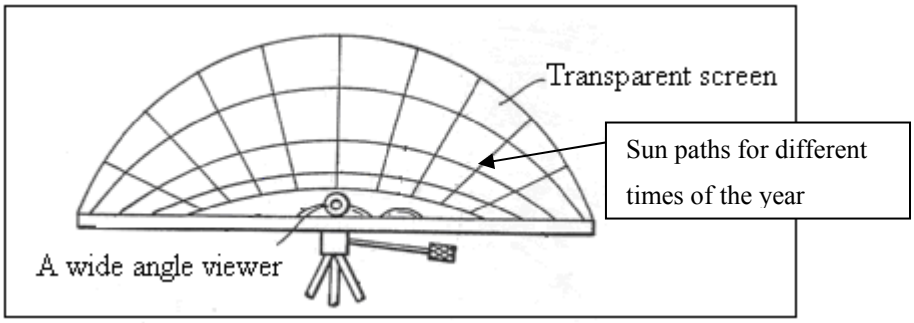


Figure 2.3: Invention by Lewis to determine the sun exposure (Lewis, 1981).

2.1.4 Dalrymple's device to indicate solar exposure

The device by Dalrymple (Dalrymple, 1987) is a variation of the device due to Lewis in which the paths of the sun are marked on a wide angle lens which is held up to the eye, as shown in Figure 2.4. When the user takes this device to a location of interest and looks through the lens towards north, the user can view the path that the sun would travel at any time of the year at that location. Although this device is more portable and more convenient to use compared with Lewis's device, it still does not give a quantitative measure of sun exposure or a method of recording the information.

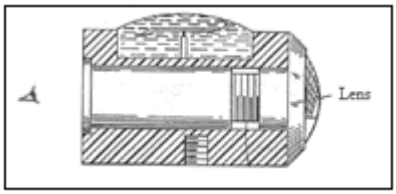


Figure 2.4: Invention by Dalrymple to determine the sun exposure (Dalrymple, 1987).

2.1.5 Moran's sunniness indicator

The device by Moran (Moran, 2006) contains a convex mirror to reflect the sky and a transparent sheet which has the path of the sun marked on it, as shown in Figure 2.5. The user of this device takes it to the place of interest, and places it facing north. The viewer looks down through the transparent sheet and can see where the sun would be exposed at a particular time of the year. Although this device is more convenient to use than the devices described in sections 2.1.3 and 2.1.4, it still does not provide quantitative information or give a method of recording solar exposure information.

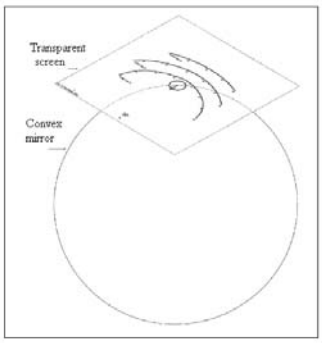


Figure 2.5: Sunniness indicator due to Moran (Moran, 2006).

2.2 Computer Modelling

The following sections describe techniques for determining solar exposure which take an approach based on computer modelling of objects in the vicinity of the location of interest.

2.2.1 Estimating canopy coverage for a golf green

The technique by Fulton (Fulton, 2002) is designed to determine canopy coverage for golf greens to ensure sufficient sunlight falls on the green. Information about the golf green and surrounding foliage and man-made features are entered into a computer. The sunpath equations are used to determine when sun or shadow would fall on the green. The result shows the number of hours that the zones in the golf green receive sunlight during the day. The simulation can be repeated with trees and other obstructions artificially added or removed. Although this computer modelling is convenient for recording information and for producing quantitative results, the information about the green, trees and the location and size of obstructions to the sunlight must be entered manually.

2.2.2 Ecotect architect software

Ecotect architect software (Autodesk, 2006) is a building analysis software package used to calculate and visualize incident solar radiation on any surface of a building. Figure 2.6 shows a colour spectrum on a grid on the building's surface which represents the radiant temperature levels for summer and winter. The result of the program is shown as a 3D view and uses complex 3D software to create the models of the building. Although this software is a highly visual architectural design and analysis tool that links with a comprehensive 3D modeller, it is nevertheless difficult and time consuming to create the 3D representation of the buildings and surroundings.

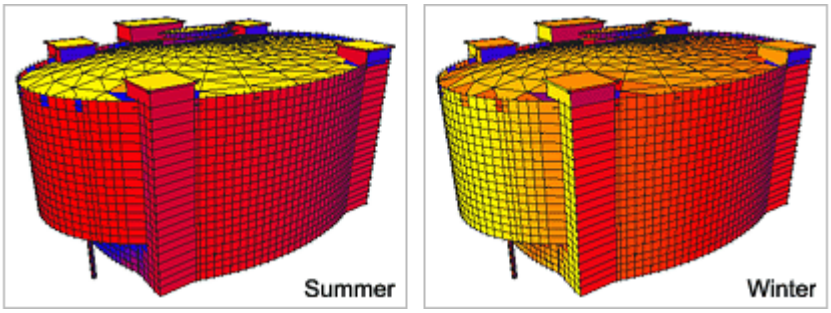


Figure 2.6: Comparing incident gains during different seasons (Autodesk, 2006).

2.2.3 SunCast architect software

SunCast™ (IES, 2007) is a building energy software tool for analyzing the solar aspect of a proposed building from the IES Company (Integrated Environmental Solutions). It is typical of a wide range of software to help architects to understand and control the geometric relationship between the sun and buildings. The inputs to this program are 3D geometry and site information. Although SunCast enables a user to perform complex solar analyses, it is still difficult to input information about complex 3D buildings and surroundings.

2.2.4 GIS software

ArcView is Geographical Information System (GIS) software and with the 3DSKYVIEW Extension (Souza, 2003) can be used to determine the exposed area of sky or buildings from a viewpoint on the earth. ArcView will import all of the structural information about buildings such as height and elevation from a CAD program that is used to represent the buildings (polygons) in a specific area. Then, the 3DSkyView Extension application computes the limit of the exposed area of sky, building area, and total sky area automatically. The result of the process can be presented on both 2D and 3D scenes. Although GIS software quickly displays the result of complex analyses to estimate the exposed area of sky from a viewpoint on the earth, it is still difficult to create complex 3D representations of buildings and surroundings.

2.2.5 GNS device

The GNS device (GNS, 2007) is manufactured by GNS Science Ltd (Institute of Geological and Nuclear Sciences Limited) to determine the solar exposure for a property. The solar exposure is determined by using digital terrain information which does not account for other obstructions such as trees and buildings.

2.2.6 The Solmetric SunEye device

The Solmetric SunEye (Macdonald, 2007; Solmetric, 2007) determines the solar exposure and calculates solar shading at different locations and times by using a digital camera with a fish eye lens to capture an image of the sky vertically above the device. The device automatically determines the skyline. It is then possible to identify sky and object regions and to calculate the position of the sun by mapping the sunpaths on the image, as shown in Figure 2.7. If the position of the sun is in a sky region at a particular date and time, then the sun is visible. The paper briefly describes two methods for determining the skyline which separates sky from non-sky regions. The first method is to scan from the top of the image down columns of pixels until the intensity of the sky is much greater than of the obstacles. This assumes that the sky is always at the top of the image. The second method is to use edge detection to identify the skyline. However, the implementation of these two methods is not described in detail.

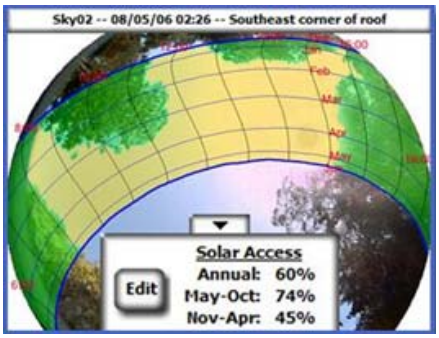
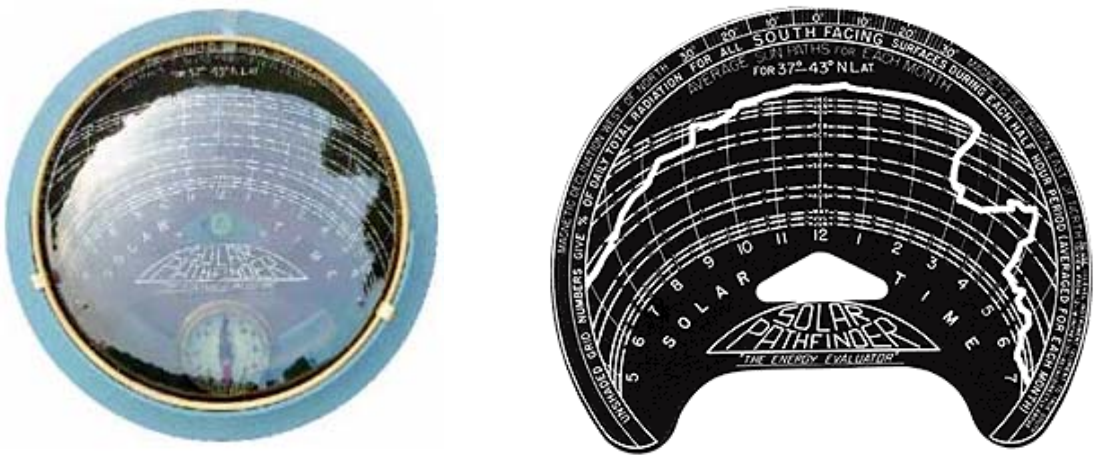


Figure 2.7: The sunpath diagram produced by the Solmetric SunEye (Solmetric, 2007).

2.2.7 The Solar Pathfinder™

The Solar Pathfinder™ (Solar Pathfinder, 2008) determines and calculates the solar exposure at any time of the year and any day of the year. It contains two main components; a dome and a sunpath diagram. The dome is placed over the sunpath diagram and there is a gap between them. The solar exposure is determined by placing the solar pathfinder at the location where trees, buildings reflect into the dome, as shown in Figure 2.8(a). The viewer then looks down through the dome and passes a white pencil through the gap to draw the

reflected shadows on the sunpath diagram, as shown in Figure 2.8(b). The Solar Pathfinder™ must be set up with the compass pointing north, and the device set level using the built-in bubble level. A different sunpath diagram is required for each latitude. The only information available is what is shown in the dome so there is no automated analysis to produce statistics such as the solar exposure by month. This is because the device does not use image processing to identify the sky region. Rather, the user identifies the edge of the sky region by drawing it on the sunpath diagram.



(a) (b)

Figure 2.8: The Solar Pathfinder device showing (a) the dome with the sunpath information superimposed and (b) the result of tracing the edge of the sky region with the white pencil (Solar Pathfinder, 2008).

2.2.8 Wiley Electronics ASSET

The Acme Solar Site Evaluation device (ASSET) (Wiley, 2005) determines the solar exposure and evaluates solar shading at different location and times. The solar exposure is estimated by taking a series of 7 or 9 pictures from east to west as shown in Figure 2.9. The pictures are then loaded into the ASSET program to calculate the shading. The latitude of the image is entered into the program. A sunpath diagram is calculated and placed at the top of the images, as shown in Figure 2.10. Image processing is used to identify the sky region and the shaded sunpath is shown with red lines in the non-sky region, as shown in Figure 2.10. Other calculations are also available such as light shaded (%), available sunlight (hours). However, the image processing methodology is not described in detail.

This device can be used at any latitude without manually changing the sunpath diagram. It is also different from the Solmetric SunEye because it does not use a fish eye lens.

Figure 2.9: A series of 7 pictures for evaluating shading by the Acme Solar Site Evaluation device (Wiley, 2005).

Figure 2.10: Sunpath diagram output by the Acme Solar Site Evaluation Device. The sunpath is shaded red in the non-sky region (Wiley, 2005).

2.3 Methods for Detecting the Sky in Images

The following reviews previous approaches to identifying the sky in digital images.

2.3.1 Sky region extraction

Luo and Etz (Luo & Etz, 2003) can distinguish blue sky pixels or regions from objects that have similar colours in an image, such as a blue wall or shirt, and determine image orientation. The colour classification uses a neural network based on hue classification to determine a belief value for all candidate blue sky pixels in the image. Next, the sky region extraction process determines regions of connected pixels where the belief values are similar to the neighbouring pixels. Finally, the row and column pixel gradient values are

computed and used to identify the orientation of the image. If the horizontal gradient value is higher than the vertical gradient value, the image indicates a landscape image; otherwise, it is a portrait image. The approach is limited to images with blue sky and is primarily intended to automatically determine the orientation of images of landscapes.

2.3.2 A model-based approach

Gallagher et al. (Gallagher, Luo, & Hao, 2008) uses a model-based approach to improve the method of Luo and Etz to detect sky in digital images. Firstly, the method of Luo and Etz is used to identify pixels that represent an initial sky region. Then, a mathematical function is fitted to the colour value of pixels in the initial sky region to estimate new pixel colour values which would have high sky belief values. Finally, the model is used to determine if additional pixels outside the initial sky region should be considered as part of the sky. The approach is still limited to images with blue sky.

2.4 Summary

Determining solar exposure can be done by using mechanical devices, computer modelling, or processing images.

The mechanical devices use indicators for good visual information of solar exposure, and most make use of mechanical adjustments for location and time in order to establish the path of the sun. The devices have become increasingly portable and convenient, but no quantitative measure of solar exposure is provided, and none of them provides a method for recording the result.

Computer modelling can make it easy to record the result. However, there is a great deal of work involved in getting an appropriate model of the area under consideration.

Another method is to take photographs of the surrounding area and use image processing to determine the sky regions and then use that information to calculate the visibility at different dates and times such as the Wiley Electronics ASSET.

We are interested in methods similar to the Wiley Electronics ASSET. Critical to its success is successfully separating the sky regions in the images. Some existing methods rely on the sky being blue or being at the top of the picture. However, it is not clear how well the current techniques cope with other situations, for example sun flash may obscure the boundary between the sky and the rest of the image.

2.5 Specific Objective for this Study

The quality of the image processing is crucial to the success of the proposed method for determining the sky region in an image. The focus of this study is to develop the image processing to make it as robust as possible to determine the boundaries of all the sky regions in the image and accurately identify the sky from the rest of the image under as wide a range of conditions as possible.

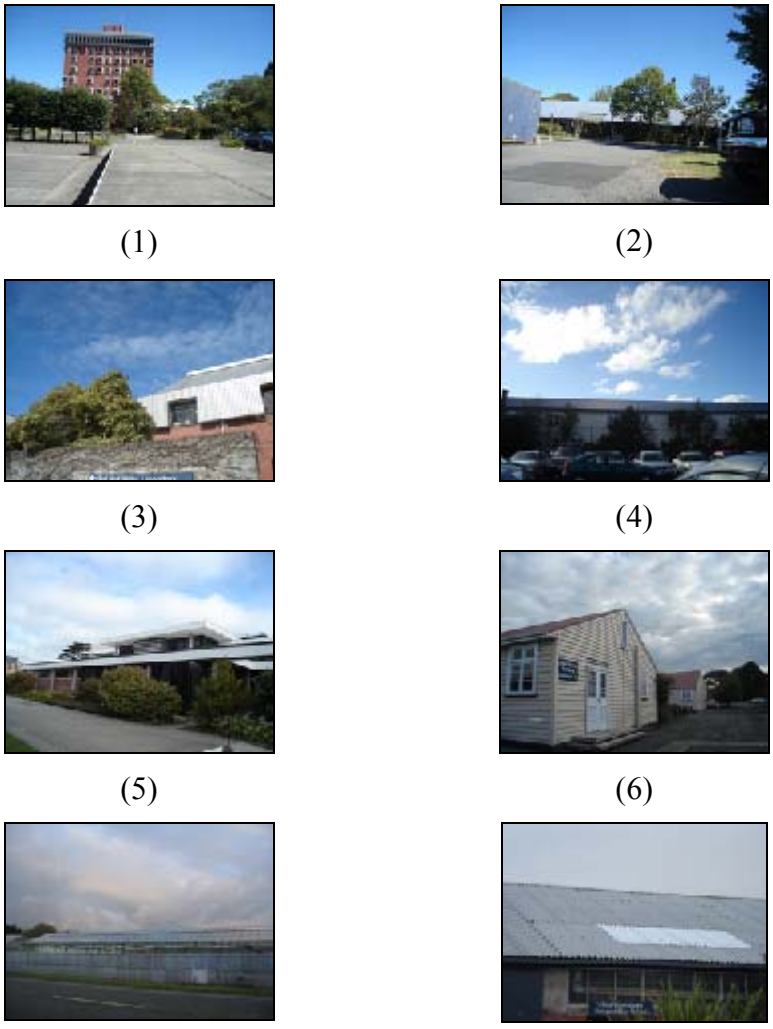
3 Method and Results

The image processing techniques used to identify the sky regions in the images need to work on a wide range of images and involve minimal user intervention. We have investigated how to make the following steps as robust as possible over a range of images:

- Selection of a camera filter and colour channel which enhance the contrast between the sky and other regions of the images.
- Finding the edges of all regions in the images including the sky regions.
- Identifying which of the regions correspond to sky.
- Providing a mechanism for the user to manually identify the sky region(s) in situations where image properties cause the automated approach to fail.

3.1 Test Images

The images used to estimate solar exposure should be taken in a range of conditions from a clear blue sky to dull overcast conditions. Accordingly the test images shown in Figure 3.1 were chosen as a representative set for subsequent testing. A number of these images include regions which are not sky but which appear to have a similar brightness and colour to the actual sky. For example, the white concrete in the building is a very similar colour to the clouds in the sky in the image shown in Figure 3.1(5). The building shown in Figure 3.1(3) contains a shiny surface. There is low contrast between the sky and glasshouses in the image foreground shown in Figures 3.1(7) and 3.1(8).



(1) (2)

(3) (4)

(5) (6)

(7) (8)

Figure 3.1: Test images used: (image 1 and 2) bright sky - no cloud; (image 3 and 4) bright sky - scattered clouds; (image 5 and 6) overcast sky - white clouds; (image 7 and 8) overcast sky - grey clouds. The black frame shown round each image is not part of the actual image.

3.1.1 Camera settings

The camera used to capture the test images was a Sony DSC-S600. There are several camera factors such as focus, white balance (dPS, 2007), flash(HowStuffWorks, 2004) and gamma (Bourke, 1998) which affect the resulting image. Although these factors can be set manually in many digital cameras, the test images were taken using the camera's automatic settings. The images captured when the solar exposure device is used in practice will vary

widely. It is therefore appropriate for the camera's automatic settings to be used rather than constrain them to optimise the test images in some way.

3.1.2 Resolution

The image resolution depends on how accurately the length of time that the site is exposed to the sun needs to be estimated (McKinnon, 2007). It is unlikely that any of the potential applications of the device discussed in section 2.2, would require the time to be estimated with an accuracy of better than 1 minute. Therefore, 1 minute was set as the accuracy requirement. The time resolution of the reports in the Solmetric SunEye is 15 minutes (Solmetric, 2007).

Because the earth rotates approximately 360 degrees in 24 hours, 1 minute corresponds to 0.25 degrees. Therefore, each pixel in the recorded image must correspond to less than 0.25 degrees.

If the number of horizontal pixels in the image is N_H and the angle of view of the camera in the horizontal direction is Θ_H degrees, then the minimum horizontal resolution requirement is given by:

$$\frac{\theta_H}{N_H} \leq 0.25 \quad \text{degrees / pixel} \tag{3.1}$$

In Appendix A, it is shown that for the Sony DSC-S600 camera, $\Theta_H = 110^\circ$. In that case from (3.1) $N_H >= 440$ which means the minimum horizontal resolution requirement is 440 pixels.

Assuming a similar vertical resolution requirement, N_V, the number of vertical pixels in the image should satisfy:

$$\frac{\theta_V}{N_V} \leq 0.25 \quad \text{degrees / pixel} \tag{3.2}$$

where Θ_V is the vertical angle of view of the camera.

In Appendix A, it is shown that for the Sony DSC-S600 camera, $\Theta_V = 75^\circ$. In that case from (3.2) $N_V >= 300$, so the minimum resolution requirement for the number pixels in the vertical direction is 300. Therefore, the resolution must be at least 400×300.

The test images used in this thesis were captured at a resolution of 2048×1536 as JPG (Scantips, 1997) images. They were sub-sampled to a resolution of 800×600 and stored as BMP (Anderson, 2007) to avoid image degradation during subsequent processes (Anderson, 2007). The resolution of 800×600 provides considerably more than the minimum resolution requirement of 400×300.

3.2 Increasing the Contrast between the Sky and the Rest of Image

This section discusses the selection of the colour channel and the use of camera filters to help to increase the contrast between the sky and the rest of the image.

3.2.1 Colour models

There are several different colour models such as RGB, HSV, YCrCb, CIE XYZ, and CIE Lab. The colour channel (Apple Computer Inc, 1996) is one component in a particular colour model. This section briefly describes each colour model.

RGB

The RGB colour model (Hearn & Baker, 2004), as shown in Figure 3.2 is an additive model in which the three primary colours (red, green, and blue) are combined to create

another colour. Typically, pixels are stored with 8 bits each for red, green and blue, respectively.

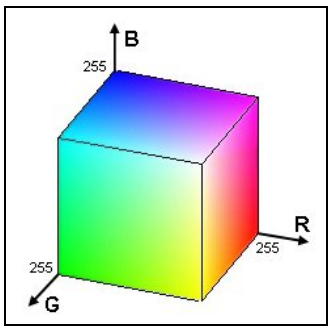


Figure 3.2: RGB colour model (Vandevenne, 2004b).

Greyscale

Greyscale (Hearn & Baker, 2004; Fisher R., 2003d) is a colour component which has 256 levels of shade between black and white. It is commonly used to represent a colour image as "white and black". When computed from an RGB colour, the greyscale is the average of the red, green and blue components.

HSV

The HSV colour model (Hearn & Baker, 2004), as shown in Figure 3.3, consists of 3 colour components which define the colour space in terms of hue (H), saturation (S), and value (V). Hue specifies the colour of an image and has a range of 0 to 360 degrees. The value of H starts from the red colour at 0 degrees, it then increases through yellow, green, and blue and returns to red again. Saturation specifies the amount of colour present and has values between 0 (white) and 1 (full colour). Value is the brightness of an image and has values between 0 (black) and 1 (light).

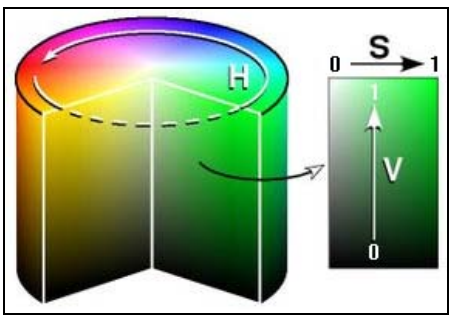


Figure 3.3: HSV colour model (adapted from Virtual Drums, 2005).

YCrCb

The YCrCb colour model (Hearn & Baker, 2004), as shown in Figure 3.4, is one of two primary colour models (the other is RGB) which is commonly used in digital images. It consists of two major components which are luminance and chrominance. “Y” corresponds to the luminance which describes the black and white components of a pixel value. “Cr” and “Cb” correspond to the chrominance which describe the colour components of the pixel separately from the luminance. Cr is the red chrominance and is proportional to the value red – Y. Cb is the blue chrominance and is proportional to the value blue –Y.

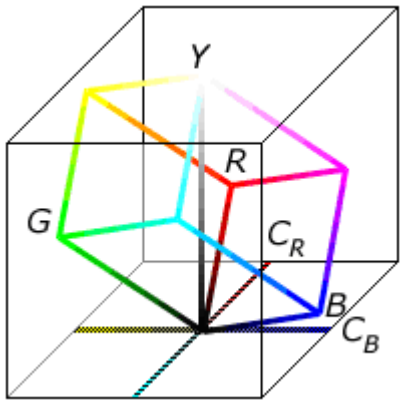


Figure 3.4: YCrCb colour model relative to the RGB model (adapted from Bit Jazz, 2004).

CIE-XYZ

The CIE-XYZ colour model (Hearn & Baker, 2004) is defined by CIE, the Commission Internationale de l’Éclairage (the International Commission on Illumination). CIE-XYZ is a transformation of the RGB colour model in the context of the CIE system which, unlike RGB or HSV, can represent all colours that can be seen by humans. CIE (Figure 3.5) is a device independent colour model which means that a colour is specified in the CIE system independently of how the colour is produced by the device such as a monitor or printer. “X”, “Y”, and “Z” correspond to red, green, and blue, respectively. In this thesis, “xYz” symbol is used to represent Y colour channel in the CIE-XYZ colour model to avoid confusion with the Y colour channel in the YCrCb colour model.

Figure 3.5 shows the CIE-XYZ in 2D. The triangle in the figure represents the RGB colour model and lower case x, y, and z refer to the normalized X, Y, and Z values.

$$x = \frac{X}{(X + Y + Z)} \tag{3.3}$$

$$y = \frac{Y}{(X + Y + Z)} \tag{3.4}$$

$$z = \frac{Z}{(X + Y + Z)} \tag{3.5}$$

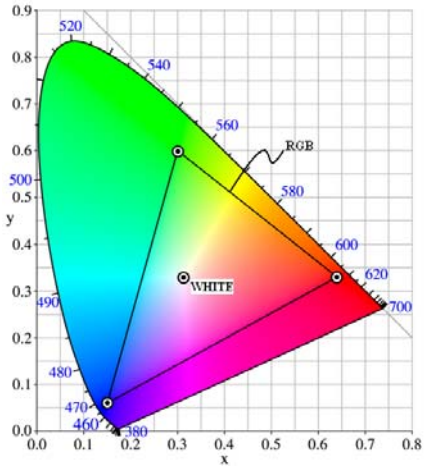


Figure 3.5: CIE XYZ colour model relative to the RGB model in the coordinate (x, y) (adapted from Wikipedia, 2007)

CIE-Lab

The CIE-Lab colour model (Colour Management Consultancy, 1980; Hunter Lab, 2008) , shown in Figure 3.6, is a transformed version of the CIE-XYZ model that describes colour using “L”, “a”, and “b” channels. The “L” colour channel represents how light a colour is within the range 0 to 100. The higher numbers are lighter. The “a” colour channel represents how green to red the colour is within the range -128 to 127. The positive values indicate more red colour and the negative values indicate more green colour. The “b” colour channel represents how blue to yellow the colour is within the range -128 to 127. The positive values indicate more yellow colour and the negative values indicate more blue colour.

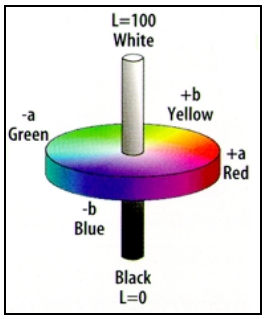


Figure 3.6: CIE Lab Colour model (Huang, 2007).

3.2.2 Testing colour channels

This section describes the method used to determine the colour channel that consistently gives the best contrast between the sky and non-sky regions and is therefore the most appropriate to use in subsequent processing to identify the sky.

The colour channels tested to determine which gives the highest contrast between sky and other regions were greyscale, (R,G,B), (H, S, V), (Y, Cr, Cb), (X, xYz, Z), and (L, a, b). The various colour channels were tested as follows:

1. On each test image, a rectangular region containing only sky and a rectangular containing no sky were manually selected, as shown for test image 4 in Figure 3.7.

(a)

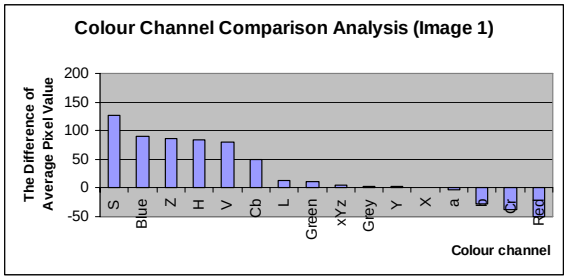
(b)

(c)

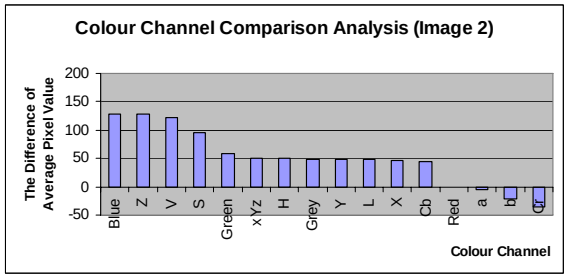
Figure 3.7: The original test image 4 (a), sky region (b) and non-sky region (c).

2. The desired colour channel was extracted from the sky and non-sky regions. All colour channels were normalized to the range 0 to 255.
3. The average pixel value of the sky and non-sky regions were calculated for the extracted colour channel.
4. The average pixel value of the non-sky region was subtracted from the average pixel value for the sky region and displayed.

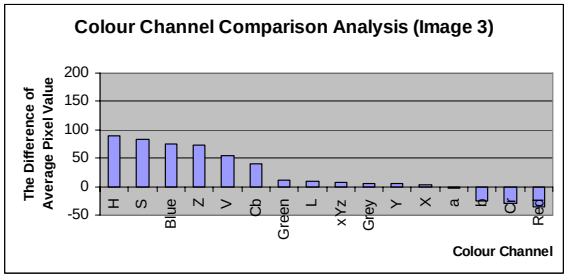
Figure 3.8 shows the results of subtracting the average pixel value of the non-sky region from the sky region for each different colour channel for each test image.



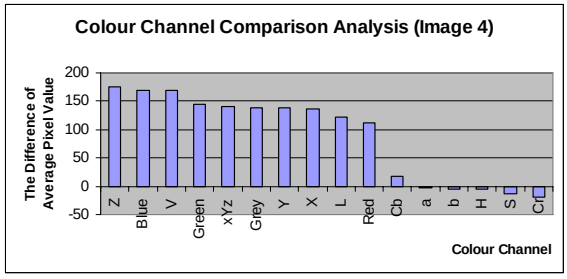
(a)



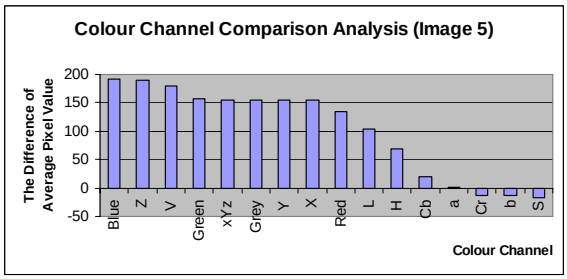
(b)



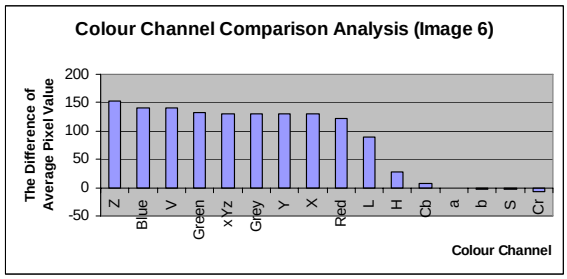
(c)



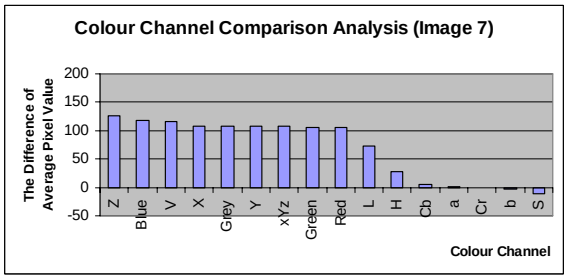
(d)



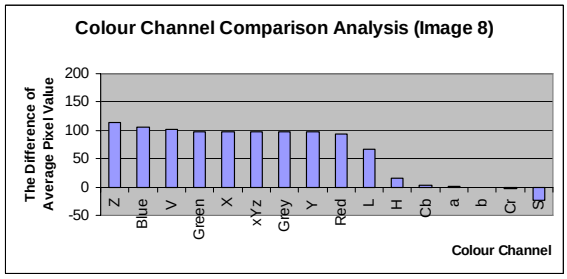
(e)



(f)



(g)



(h)

Figure 3.8: The difference of the average value of the sky and non-sky regions for each different colour channel in each test image.

The “S” channel gave the highest contrast for test image 1 Figure 3.8(a) but gave poor contrast for test image 4 (Figure 3.8(d)). Figure 3.1(1) shows the sky region in image 1 is highly saturated with a vivid blue colour which has a saturation value approaching 1. The non-sky region has low saturation in this image which means that the S component gives a

high contrast for this image. This is significantly different in image 4 (Figure 3.1(4)) where the cloud means that the blue has relatively low saturation. The S channel gives poor contrast between sky and non-sky regions in this case.

Figure 3.8(e) shows that the difference of the average contrast value between sky and non-sky regions for each different colour channel in image 5 is higher than for the other images tested. This is because the sky region in the image contains a lot of white, while the colour of the non-sky region selected for testing contains a lot of black, as shown in Figure 3.9.

(a) (b) (c)

Figure 3.9: The original test image 5 (a), sky region (b) and non-sky region (c).

Although, the blue colour channel gives a contrast that is not the best for some images such as image 1, as shown in Figure 3.8(a), it is consistently among the three channels that give the highest contrast across all test images. It was decided to use it in subsequent image processing to identify the sky regions in the images although it is recognised that the Z channel gives a contrast very similar to blue and could equally well have been used.

Another consideration was the contrast between the sky and the clouds. The blue colour channel increased the contrast between the sky and non-sky regions of an image but it also reduces the contrast between scattered clouds and blue sky itself as shown in Figure 3.10. This reduces the likelihood that scattered clouds will be identified as regions separate from the sky in later processing.

(a) (b)

Figure 3.10: The greyscale (a) and blue (b) colour channels for the bright sky – scattered clouds image (test image 4).

3.2.3 Filters

This section discusses the use of camera filters to increase the contrast between the sky and the rest of the image. We consider polarising, ultraviolet and skylight filters. A polarising filter is often used to reduce glare from light reflecting off horizontal surfaces. The ultraviolet and skylight filters can be useful if skylight contains more ultraviolet than the light from non-sky regions.

Polarising filter

The effect of a polarising filter is dependent on the rotation of the filter. If the polarising filter is oriented parallel to the reflected light, i.e. a horizontal polarising filter, it lets horizontally polarised light pass and blocks vertically polarised light. A vertical polarising filter lets vertically polarised light pass and blocks horizontally polarised light, as shown in Figure 3.11. Therefore, a vertically polarised filter has the potential to significantly reduce the intensity of light from horizontal reflecting surfaces (such as roofs) (Fuglies, 2003) and may therefore improve the contrast between sky and other regions in the image as shown in Figure 3.11.

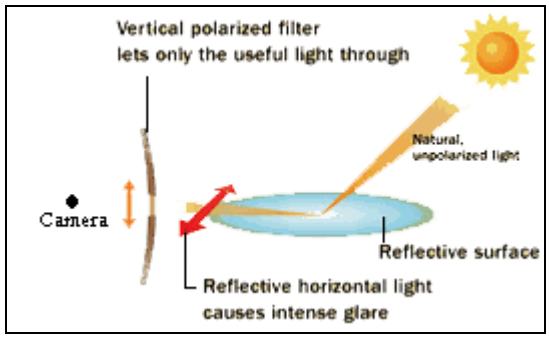


Figure 3.11: The reflection off a horizontal surface is likely to be horizontally polarised (adapted from Fuglies, 2003)

Ultraviolet and Skylight filters

When there is excessive ultraviolet light, a blue haze and loss of definition of distant objects may appear in an image. An ultraviolet or skylight filter (dpFWIW, 2004) can reduce ultraviolet light in the image. The skylight filter absorbs more ultraviolet light than the ultraviolet filter. For our purposes, if the sky region emits more ultraviolet light than the non-sky region, using an ultraviolet or skylight filter will affect the sky region of the image more than the non-sky region. However, it must be recognised that commodity digital cameras are not very sensitive to ultraviolet light (Great Landscape Photography, 2004) and therefore it may be difficult to observe the effect of these filters.

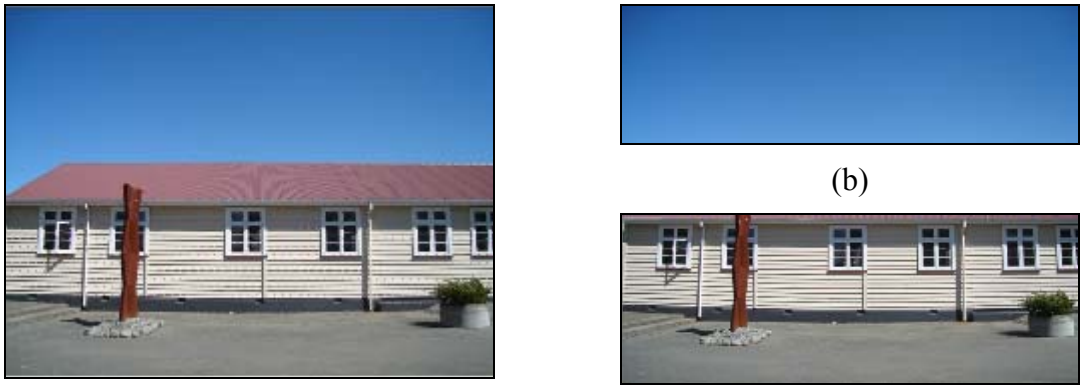
3.2.4 Testing the filters

The polarising, ultraviolet, and skylight filters were tested to see if they increased the contrast between the sky and other regions of the image. The polarising filer that was used was the “Foto Polarizer 55mm”. The Ultraviolet model was the “Vivitar 52mm UV-HAZE”. The skylight model was the “LlBc 52mm Nikon”

Testing the ultraviolet and skylight filters

The following method was used to determine if the ultraviolet filter helps to increase the contrast.

1. The image shown in Figure 3.12(a) was used for this study. It was taken with and without each filter. The ultraviolet and skylight filters were simply placed in front of the camera lens. For more details about selecting the image conditions, see the discussion at the end of section 3.2.5.
2. From each image, a rectangular region containing only sky and a rectangular region containing no sky were manually selected, as shown in the Figure 3.12 (b) and (c).



(a) (b) (c)

Figure 3.12: The original colour image (a) is separated into sky region (b) and non-sky region (c).

3. For reasons explained in section 3.2.2, the blue colour channel was extracted from the sky and non-sky regions.
4. The average pixel value for the sky and non-sky regions were calculated for the extracted colour channel.
5. For each image, the average pixel value of the non-sky region was subtracted from the average pixel value for the sky region.

Figure 3.13 shows the average contrast value between sky and non-sky regions with and without the ultraviolet and skylight filters. If the filter was blocking significant UV from the sky, the contrast should have been significantly lower. If so then it would have been worth investigating the use of a UV passing filter, or performing some subtractions to isolate the UV light. Although the use of filters gave lower contrast than without filters, the

values of the average contrast are similar. This may be due to the camera's lack of sensitivity to ultraviolet light.

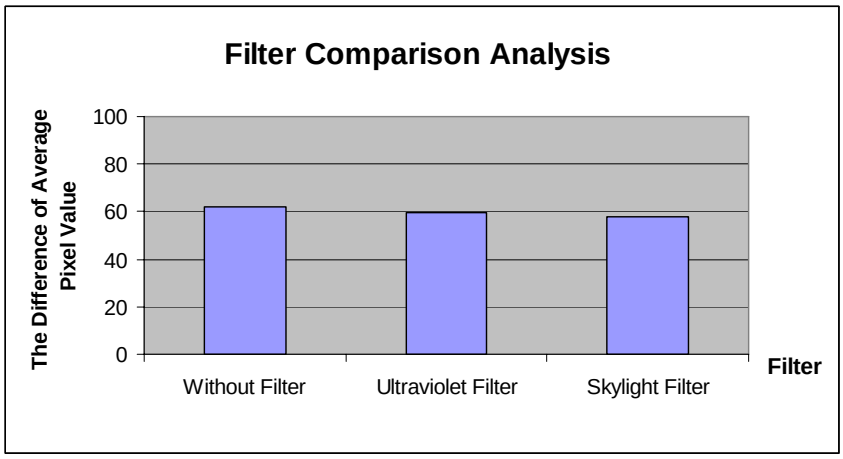


Figure 3.13: Compares the average contrast value between the sky and non-sky regions of the image with the ultraviolet and skylight filters and without filters for the image in Figure 3.12.

Testing the polarising filter

The following method was used to test whether the polarising filter enhanced the contrast between sky and other regions of the image.

1. The bright sky – no cloud image as shown in Figure 3.12(a) was taken without the filter and then two further images were taken with the polarising filter. The first was with the polarising filter orientated to visually show the highest contrast between the sky and the rest of the image. The filter was then orientated perpendicular to the first orientation and the second image taken. For more details about selecting the image conditions, see discussion at the end of section 3.2.5.
2. From each image, a rectangular region containing only sky and a rectangular region containing no sky were manually selected, as shown in the Figure 3.12.
3. For reasons explained in section 3.2.2, a blue colour channel was extracted from the sky and non-sky regions.
4. The average pixel value for the sky and non-sky regions were calculated for the extracted colour channel.

5. For each image, the average pixel value of the non-sky region was subtracted from the average pixel value for the sky region.

Figure 3.14 shows the average contrast value between sky and non-sky regions without the filter and with the polarizing filter aligned horizontally and vertically for the bright sky – no cloud images (Figure 3.12(a)). It is clear that the vertical polarising filter gave the highest contrast between sky and non-sky regions.

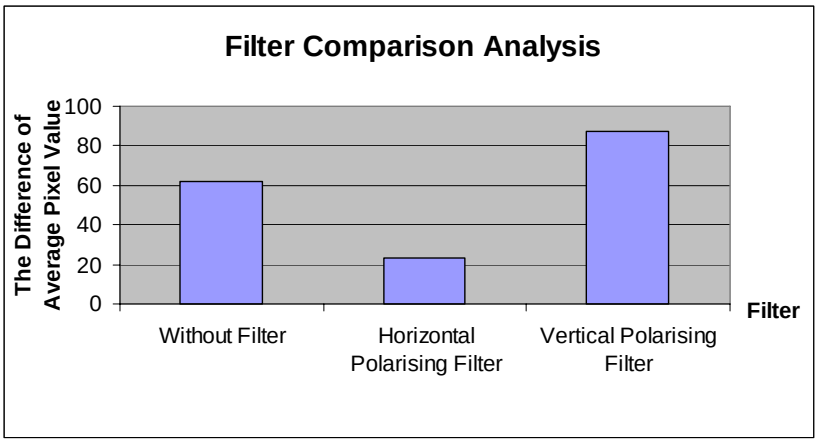


Figure 3.14: The average contrast value between sky and non-sky regions of the image with polarising filters and without the filters for the image in Figure 3.12.

3.2.5 Discussion of filter results

The contrast improvement achieved by the UV and skylight filters was relatively small. It is difficult to know whether this is because there is relatively little difference between the UV radiation from the sky and non-sky regions or whether it is because the camera's sensitivity to UV was low.

Although the vertical polarising filter gave the highest contrast between sky and non-sky regions, the difference of 20 in the contrast values between using no filter and using the polarising filter is only 8% of the dynamic range. In addition, the polarising filter is difficult to use because it needs to be oriented manually by inspecting the image as it is taken. The optimum orientation of the polarising filter depends on whether the reflecting

surface is horizontal or at some other angle. As these factors make the polarising filter difficult to use in practice it was decided not to use it further in the current project.

There is a possibility of taking an image twice, with and without a filter and to perform some subtractions. However, where the image contains objects such as scattered clouds or trees which may move, the images taken with and without the filter may be different and therefore difficult to use in subsequent processing. Given this difficulty and the relatively low contrast improvements obtained in the tests reported here, it was decided not to use the filters in subsequent work for this thesis.

3.3 Identifying the Sky Region

In order to improve and develop the image processing to segment the sky from the rest of the image, it is necessary to develop a technique that will do this successfully across a wide range of images taken under a variety of conditions. This section describes the detail of the image processing intended to achieve this.

The following steps are involved:

1. Extract a colour plane.
2. Apply the Canny edge detection algorithm (Green, 2002) to determine the boundary lines in the image.
3. Apply the morphological closing algorithm (Fisher R., 2003a) to close gaps in the boundaries identified by the Canny edge detector.
4. Identify which of the enclosed regions are sky.
5. Display the segmented image.

3.3.1 Extracting a colour channel

The blue colour channel was extracted from the original colour image because, as described in section 3.2.2, the blue colour channel was one that gave the best contrast

between the sky and the rest of the image. This also means that subsequent processing is now dealing with only an 8-bit image.

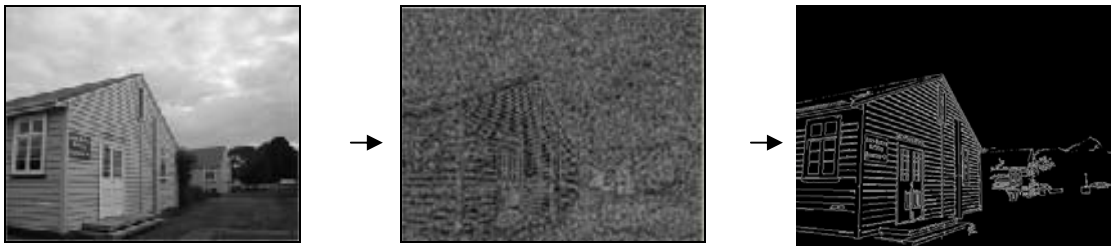
3.3.2 Finding the edges in an image

After the blue colour channel was extracted from the image, the edges of all the regions in the image were determined.

There are several methods to perform edge detection such as the Sobel, Roberts Cross, Prewitt, Laplace, and Canny edge detectors (Fisher R., 2003c). The most widely accepted of these, the Canny edge detector (Green, 2002), is used here because it reduces the possibility of missing actual edges and detecting false edges. It also attempts to minimise the distance between the detected edges and actual edges and ensures that an actual edge point in the image produces only one edge (Roushdy, 2006).

The Canny operator uses multiple processes to detect edges in an image. First, it smoothes an image with a Gaussian filter to reduce noise and unwanted texture and detail. Then it calculates the edge gradient strength which is a measure of the change in image intensity at the edge. The edge direction is then determined and non-maximal suppression is applied to make a thin line in the edge direction and suppress any pixel that is not on an edge. Finally, hysteresis thresholding eliminates the breaking up of edge contours. Any gradient values that are higher than an upper threshold represent edge pixels and any pixel values that connect to those edge pixels and are higher than a lower threshold, represent edge pixels also.

The Canny edge detector may also find edges around clouds in the sky so it is necessary to adjust the upper threshold and lower thresholds to avoid those edges, as shown in Figure 3.15. The result of the Canny edge detector is a binary image. A procedure for determining the most appropriate threshold values will be developed in section 3.4.1



(a) (b) (c)

Figure 3.15: Image 6 (a) shows the blue colour channel prior to the application of Canny edge detection. Image (b) shows the edges detected when the upper and lower thresholds are set to low values, while image (c) shows the result when the upper and lower thresholds are adjusted to avoid the edges around clouds.

3.3.3 Closing gaps in edges

The approach being taken to finding the sky is to find the edges surrounding all possible sky regions. Although the Canny edge detector helps to do that, it may leave gaps in some edges as shown in Figure 3.16. If these are not closed the area of sky may "spill" through the gap as shown in Figure 3.17. To counter this, the morphology closing operation (Fisher R., 2003a) is applied to the binary image of edges as shown in Figure 3.18.

The morphology closing algorithm uses a dilation operation followed by an erosion operation. The closing operations are performed by placing the middle pixel of a structure element, which is a uniform intensity pixel array (e.g. 3×3 or 5×5), over each pixel in the image and comparing the value of each structure element pixel with the pixel it overlays. If they are all the same value, we set the image pixel to a white binary colour for the erosion algorithm. If at least one of them is the same value, we set the image pixel to a white binary colour for the dilation algorithm. Otherwise, the image pixel is not changed. If there are large gaps, a larger structure element will be required to fill them. A procedure for determining the appropriate size of the structure element will be developed in section 3.4.1.

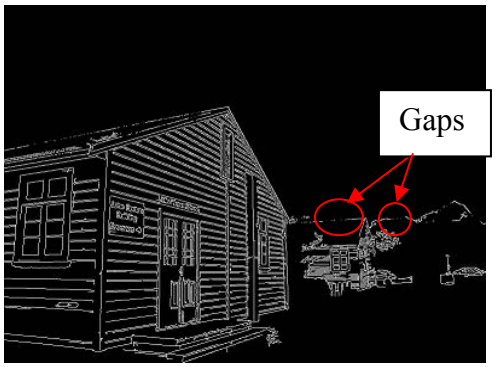


Figure 3.16: Gaps in the edges found using the Canny edge detector.

Figure 3.17: Consequences of failing to close a gap in a boundary. The grey colour represents the “sky” and it “spills” through the gaps.

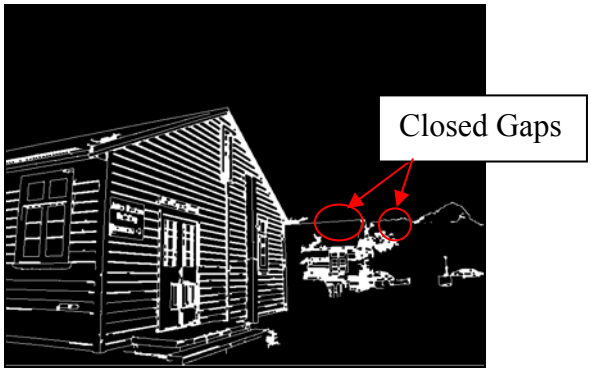


Figure 3.18: Closing the gaps in the edges by applying the morphology closing algorithm.

3.3.4 Identifying all sky regions

Having determined the boundaries of all the regions in the image we can manually find which of the regions are sky and identify all the pixels in any given region using the FloodFill algorithm (Vandevenne, 2004a). A pixel in the selected region, called the FloodFill starting pixel, is also identified by the manual selection operation and the FloodFill algorithm then finds all pixels in that region, as shown in Figure 3.19.

Section 3.4.2 will describe a procedure for automatically determining which of the regions are sky, and the FloodFill starting pixels.

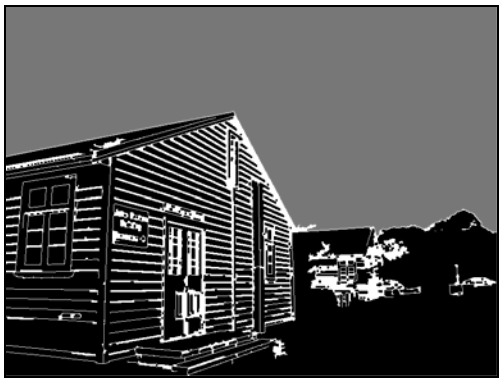


Figure 3.19: The result after applying a FloodFill algorithm to the upper region in Figure 3.18. The grey colour represents the sky.

3.3.5 Displaying and storing the segmented image

After identification of the sky region, the image is converted to binary form with a white binary colour used to represent sky pixels and a black binary colour to represent non-sky pixels as shown in Figure 3.20. This image was also stored for subsequent processing to enable estimation of solar exposure.

Figure 3.20: The final image after identifying the sky region. The white colour represents the sky. The black frame shown around it is not part of the image.

3.4 Selecting Image Processing Parameters

In order to determine the sky regions in an image using the process described in section 3.3, it is necessary to determine values for the upper threshold and the lower threshold for the Canny edge detector, the size of the structuring element for the morphological closing algorithm, a FloodFill starting pixel for the FloodFill algorithm, and which of the regions are sky.

To make the image processing as robust as possible it is necessary to use values for the threshold settings and the size of the structuring element so that they successfully identify the sky regions in a wide range of images. As there is no theoretical basis for setting these parameters, an empirical study was undertaken to find the most appropriate settings as described in the following sections.

3.4.1 Determining edge detector upper and lower thresholds and the structure element size

To determine the settings for the Canny edge detector upper and lower thresholds and the size of the structure element that ensures the best determination of the sky region across the range of test images, the approach described below was used. A discussion of the rationale for this approach is given later in this section and details of how the process was carried out are given in Chapter 4.

1. The upper threshold, lower threshold and the size of the structure element were manually adjusted for each test image using the blue colour channel, until the sky boundary was seen to coincide visually with the boundary obtained by inspecting the image. The sky region was then identified manually and its area and perimeter were determined. We refer to these as the "actual" area and perimeter in the next steps.
2. The area and perimeter of the sky were calculated for many sets of the parameters (upper threshold, lower threshold, size of structuring element). The lower thresholds ranged from 1 to 1000, the upper thresholds from 1 to 1000, and the structure element from 3×3 to 21×21 in steps of 2 in each direction.
3. The sets of parameters which gave a calculated area within various tolerances of the actual area were determined, as shown in equation (3.6). The smallest tolerance that could be achieved for all images was 0.3% of the actual area.

$$|\text{calculated area} - \text{actual area}| / \text{actual area} < \text{toleranceArea} \quad (3.6)$$

4. Step 3 was repeated to investigate the tolerance in the perimeter, as shown in equation (3.7). The tolerance in the area was kept at 0.3% for this step. The smallest tolerance in the perimeter that allowed all images to be represented was 5% of the actual perimeter.

$$|\text{calculated perimeter} - \text{actual perimeter}| / \text{actual perimeter} < \text{tolerancePerimeter} \quad (3.7)$$

5. The maximum ranges for each parameter which gave a calculated area within a tolerance of 0.3% for area and within 5% for perimeter were determined.

<u>The justification for the process</u>

As described in steps 3 and 4 above, the upper and lower thresholds and the size of the structure element are chosen to achieve the smallest tolerance between the calculated and actual perimeters and areas over all images in the test set.

Step 3 in the process described above found the range of possible values for the upper and lower thresholds and the size of the structure element which minimised the error in the area of the sky region. The area is emphasised because the accuracy of the sunpath calculations

will largely be affected by the area of the sky region. In contrast, the perimeter is more likely to affect factors such as spurious cloud edges or small deviations in the shape of the boundary of the sky region.

An indication of the accuracy with which the area of the sky region needs to be known in order to estimate the solar exposure is given by the following analysis. If we assume the sky area is half of the total area of the image, it corresponds to 240,000 (800×300) pixels. If we assume that 1 horizontal pixel corresponds to 1/2 minute of the sun's movement, as described in section 3.1.2 then, according to the accuracy criteria established in that section, a reduction in area of 1 pixel along each edge is acceptable. The area of the sky when it is reduced by 1 pixel along each edge is 237,804 (798×298) pixels, approximately 0.9% less than the original. Therefore, although this is a very simplistic analysis, it makes sense to select the parameters to minimise the error in the area of the sky region and hopefully achieve a minimum error of not more than 0.9%.

Although an error in the perimeter of the sky region does not have the same impact on the accuracy of solar exposure calculations as does error in the area, it is nevertheless desirable to minimise the error in the perimeter across all images, to reduce the possibility that spurious cloud edges may form separate regions within the sky

The results of the process

The smallest tolerance in the area of the sky region determined using step 3 was 0.3%, which is appreciably less than the indicative figure of 0.9% error established above. The smallest tolerance in the perimeter determined in step 4 was 5%. The smallest tolerance in the area and perimeter of the sky region were each determined to one significant figure.

The smallest area and perimeter tolerances both occurred only with a structure element size of 3×3. The effect of having a larger structure element size is shown in Figure 3.21 where the error in the area is increased to more than 0.3% when a 13×13 structure element is used. The perimeter error would also be affected in this example, but to less than 5%.

(a) (b)

Figure 3.21: Compares images which have the same value of the upper and lower thresholds but different structure element sizes. Image (a) uses a 3×3 structure element while image (b) uses a 13×13 structure element.

The ranges of upper and lower threshold values for the smallest area and perimeter tolerances for all images are 186 to 217 and 16 to 61 respectively, as shown by the "All Images" region in Figure 3.22. Figure 3.22 also shows the ranges for the lower and upper threshold values which met the smallest area and perimeter tolerances criteria in all but one of the test images. For details about selecting the value of the thresholds see Chapter 5.

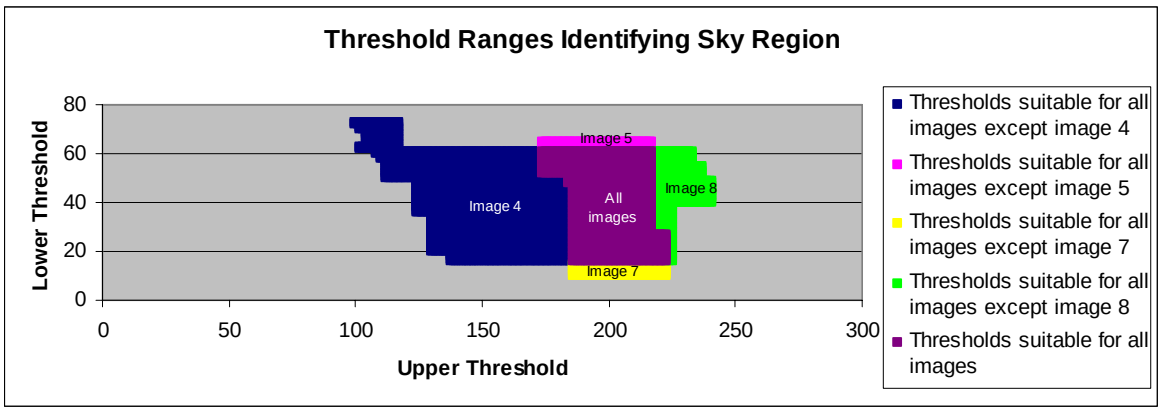
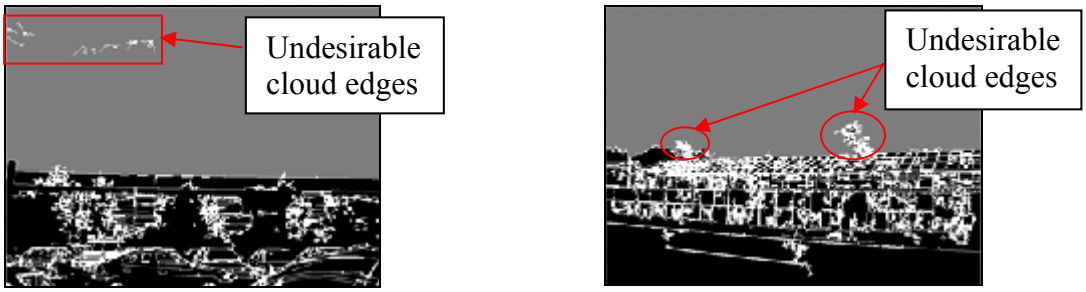


Figure 3.22: The threshold ranges which gave area and perimeter values of the sky region within 0.3% and 5% respectively of the correct values.

Discussion of the results

This empirical study has shown that it is possible to choose a size for the structuring element and ranges of the upper and lower thresholds for the Canny edge detector which give errors in the area and perimeter of the sky region which are less then would be likely to affect the accuracy of the solar exposure calculations. Nevertheless, it is instructive to see why some threshold values failed to meet the minimum criteria for images 4, 5, 7 and 8 as shown in Figure 3.22.

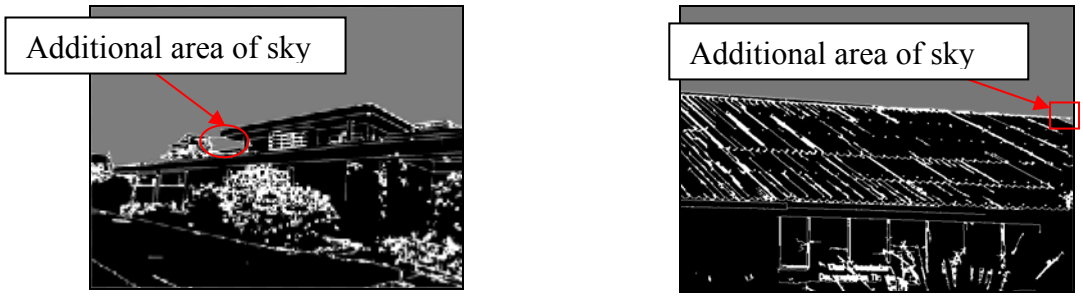
Figure 3.23(a) shows the result for image 4 when the upper threshold is 150 and lower threshold is 40 and Figure 3.23(b) shows the result for image 7 when the upper threshold is 200 and lower threshold is 10. These values produce spurious edges in the clouds which violate the perimeter criterion. Although the unwanted edges in these particular examples are not likely to have a significant effect on the solar exposure calculation, if these edges surrounded large cloud areas the resulting exposure calculation could be significantly in error.



(a) (b)

Figure 3.23: Image 4(a) and image 7(b) failed to meet the perimeter criterion when the upper threshold equals 150 and 200 and the lower threshold equals 40 and 10, respectively.

Figure 3.24(a) shows the result for image 5 when the upper threshold equals 200 and lower threshold equals 65 and Figure 3.24(b) shows the result for image 8 when the upper threshold equals 225 and lower threshold equals 40. These values give a sky area more than 0.3% larger than the correct area in both images.



(a) (b)

Figure 3.24: Image 5(a) and image 8(b) failed to meet the area criterion when the upper threshold equals 200 and 225 and the lower threshold equals 65 and 40, respectively.

3.4.2 Determining sky regions automatically

In section 3.4.1 the values of upper and lower thresholds, and the structure element size were determined to identify the boundaries of all the region in an image. However, in that analysis those regions corresponding to sky were manually identified. This section describes the initial development of an algorithm to automatically determine which of the regions identified are actually sky. The algorithm is based on the brightness of the pixels and the area of the regions.

Using image brightness to determine the FloodFill starting pixels

The following steps are involved to automatically determine FloodFill starting pixels in an image. They are based on the brightness of the pixel values in the blue colour channel.

1. A histogram of the image is created.
2. The 80^{th} percentile of the histogram is used as the brightness threshold, as shown in Figure 3.25. The end of this section will give more detail about choosing this threshold.

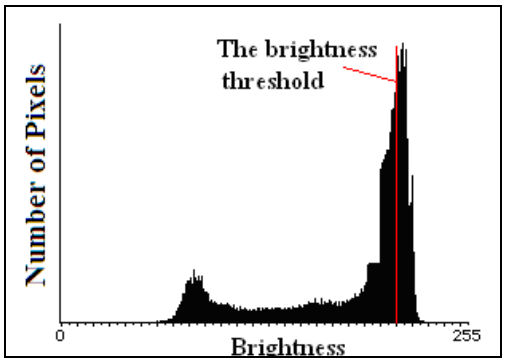


Figure 3.25: Example of determining the brightness threshold.

3. Any pixel with a value greater than or equal to the brightness threshold is classified as a FloodFill starting pixel.

Determining the actual sky regions

Every region of the image which contains at least one FloodFill starting pixel is a possible sky region. The algorithm to determine which of those regions are sky regions is based on the size of the regions. The basic idea is that possible sky regions are ranked by area and each region is considered in turn to see if it should be classified as a sky region. The criterion chosen was that to be classified as a sky region the region under consideration in ranked order should have an area at least 50% of the accumulated sky area thus far. A discussion of why 50% was used as the criterion is given later in this section. Figure 3.26 shows the pseudocode for the algorithm to identify which of the regions are classified as sky.

```
Create a temporary image in which all pixels are marked as not identified
FOR  Each pixel in the blue colour channel of the image (starting at the top left of
     image and scanning down column by column)
     IF  brightness >= The brightness threshold THEN
          IF pixel not marked as identified in temporary image
            AND
               pixel not on the edge between two regions
               THEN
                    Apply the FloodFill algorithm to the region in the
                    morphology closing image using this pixel as the seed. At
                    the same time FloodFill sets the pixels corresponding to the
                    region in the temporary image as "identified".
                    Calculate the area of the region.
                    Store the area of the region and the position of the FloodFill
                    starting pixel in an array.
          END IF
      END IF
END FOR

Initialise the starting value of the total sky area to 0.
Sort the array of areas identified in decreasing order by area.
Create an empty image that will become the final image showing sky and non sky regions.

FOR  Each element in the array of areas
     IF area of element >= 50% of the total sky area THEN
                    Classify this as a sky region by applying the FloodFill
                    algorithm to the region in the morphology closing image
                    using the FloodFill starting pixel position of the element.
                    FloodFill also sets the pixels corresponding to the region in
                    the final image indicating the sky region.
                    Add the area of the region to the total sky area.
     END IF
END FOR
```

Figure 3.26: Pseudocode for automatic identification of sky regions.

The brightness threshold and area selection criteria

As described at the beginning of this section for determining sky regions automatically, the values of 20% for the brightness threshold and 50% for the area selection criteria were used. This section will discuss why those criteria were selected.

Figure 3.27 shows the results for all test images using different values for brightness threshold and area selection criteria. The sky regions are correctly determined for most of the test images. However, images 7 and 8, as shown in Figures 3.27and are much more sensitive to these settings.

Image	Brightness Threshold / Area Selection	15%	20%
 (1) Image 1	30%	 (1.1)	 (1.2)
	50%	 (1.3)	 (1.4)
 (2) Image 2	30%	 (2.1)	 (2.2)
	50%	 (2.3)	 (2.4)
 (3) Image 3	30%	 (3.1)	 (3.2)
	50%	 (3.3)	 (3.4)

Image	Brightness Threshold / Area Selection	15%	20%
 (4) Image 4	30%	 (4.1)	 (4.2)
	50%	 (4.3)	 (4.4)
 (5) Image 5	30%	 (5.1)	 (5.2)
	50%	 (5.3)	 (5.4)
 (6) Image 6	30%	 (6.1)	 (6.2)
	50%	 (6.3)	 (6.4)

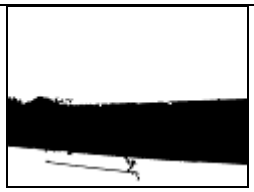
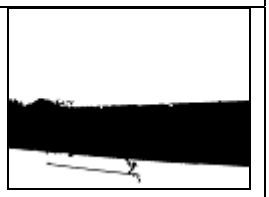
Image	Brightness Threshold / Area Selection	15%	20%
 (7) Image 7	30%	 (7.1)	 (7.2)
	50%	 (7.3)	 (7.4)
 (8) Image 8	30%	 (8.1)	 (8.2)
	50%	 (8.3)	 (8.4)

Figure 3.27: Testing criteria for FloodFill starting pixel and sky region selection. The white colour represents the sky region. The surrounding black frame is not part of the image.

In order to select the percentage of area in the histogram to determine the brightness threshold, as shown in Figure 3.25, the value of the brightness threshold needs to be large enough so that pixel values in an overcast sky with gray clouds are included, Figure 3.27 image 8. However, if the brightness threshold is too large, regions from the non-sky areas of the image may have FloodFill starting pixels.

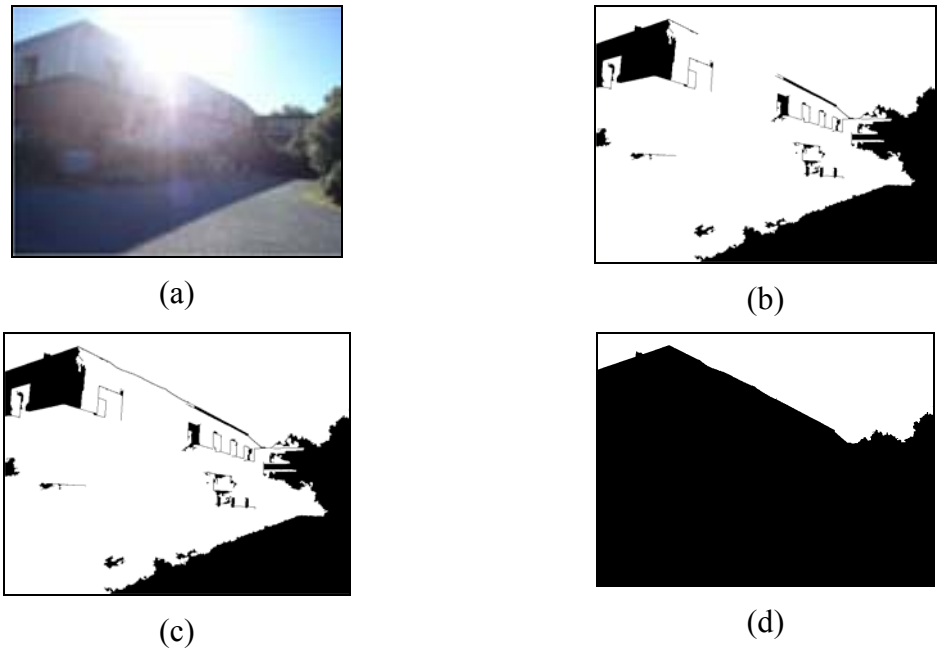
Similarly, determining a value for the area selection criteria for a region to be considered as sky is a trade-off. It needs to be small enough to include all regions that could be sky and not too small so as to include regions that are not sky, compare Figure 3.27(7.1) and (7.3).

The correct regions for these images were identified only if 20% for brightness threshold and 50% for area selection were used. Although the values of 20% for the brightness threshold and 50% for the area selection criteria selected as a result of these tests work for all the test images, they may not be successful for other images. An example which would likely fail is where the image consists of a number of small bright areas of sky such as would be obtained if the image was taken through the branches of a tree. Further testing and revision of the algorithm is required to address these cases.

3.5 User Interaction

Although this work has found the threshold values, structure element size, actual sky region in the regions that gave satisfactory results across a wide range of images, there will be some situations where image processing alone will not correctly determine sky area boundaries. For example, images which are taken directly into the sun or which involve sunlight reflection off very shiny surfaces, as shown in Figure 3.28(a), will require some user interaction to correctly identify the sky region. Possible approaches could be to draw the sky boundary manually or paint the sky region or non-sky region as shown in Figure 3.28.

When the sky boundary is identified using a line drawing tool, as shown in Figure 3.28(c), then the image could be automatically reprocessed to identify sky regions. However, as can be seen, the building region is still incorrectly classified as a sky region because it contains several bright pixels and the area is big enough to be a sky region. In this situation, the user could simply click in the region that is incorrectly classified to indicate that it is not actually sky. The result would be as shown in Figure 3.28(d).



(a) (b) (c) (d)

Figure 3.28: User interaction techniques that could be used where automatic image processing is not able to identify the sky regions correctly. The first image (a) shows the original image. Image (b) shows the sky region identified by automatic image processing. Image (c) shows how the sky boundary could be corrected using line drawing while, image (d) shows the result of using a tool to paint the non-sky region.

3.6 Summary

Image processing techniques were used to identify regions in an image including the sky region. The Canny edge detector followed by the closing morphology algorithm applied to the blue colour channel can satisfactorily find the boundaries of the sky region of an image. The FloodFill algorithm is then applied to display the image result.

The boundaries of the sky region can be accurately determined across a wide range of images with a 3×3 structure element and the upper threshold for the Canny edge detector in the range 186 to 217 and lower threshold in the range 16 to 61.

Although the structure element size and threshold values gave satisfactory results across a wide range of images, for some situations such as a sun flash image, the image processing

will not correctly determine sky region. In this situation, user interaction would be required to identify the boundary of the sky regions.

4 Implementation Issues

This chapter describes in detail how the various techniques and algorithms outlined in Chapter 3 were implemented. It includes descriptions of the programs that were used to find the ranges of upper and lower threshold values for the Canny edge detector, and the structure element size, which gave the minimum error in sky area and perimeter across the range of test images.

4.1 Image Processing Software

The image processing library used was OpenCV (Bornet, 2006) which is the Open Source Computer Vision Library for C and C++ programs. Intel released the first version of OpenCV in 1999. It contains over 500 functions including advanced algorithms for image processing and computer vision. OpenCV is supported on Windows, Linux and MaxOSX platforms.

OpenCV's functionality is contained in the modules CV, CCAUX, CXCORE, and HighGui. CV contains image processing, camera calibration, and vision algorithms. CVAUX contains experimental OpenCV functions. CXCORE contains linear algebra, statistical methods and drawing functions while HighGui contains the basic I/O interfaces and the multi-platform windowing capabilities.

OpenCV stores images as a C structure, IplImage as defined in CXCORE. In addition to raw pixel data, it contains a number of descriptive fields, collectively called the image header. The image header includes image width and height in pixels, the depth and nChannel. Depth is one of several predefined constants that indicates the number of bits per pixel per channel. For example, if depth = Ipl_DEPTH_8U, data for each pixel channel is stored as 8-bit, unsigned values. nChannel is the number of data channels (from 1 to 4). Each contains one type of pixel data. For example, RGB images have three channels- red, green, and blue intensities. (These are sometimes called BGR images, because pixel data

are stored in the order blue, green, and then red values). Greyscale images contain only one channel – pixel brightness. The OpenCV library modules that were used in this thesis are summarised in Appendix D.

4.2 Overview of the Program

Various programs were written to carry out the tasks described in Chapter 3. All of them were event-driven based around a front panel of which the screen shot in Figure 4.1 is an example. Some events, such as the adjustment of a scroll bar to change one of the Canny edge detector thresholds, caused the execution of call-back functions which were written to carry out the desired operation.

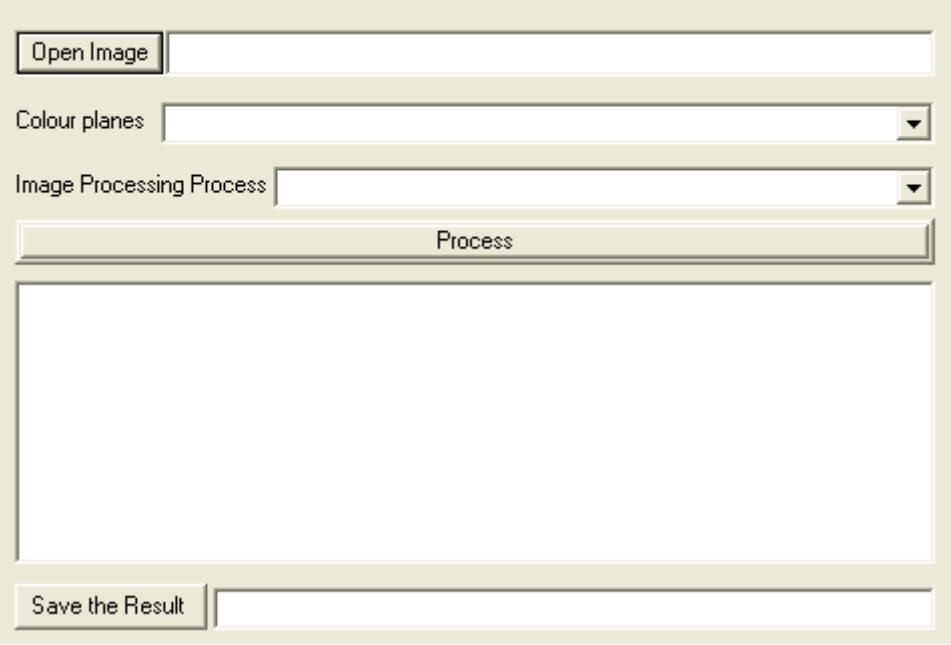


Figure 4.1: The front panel of the program.

As described in Chapter 3, a set of basic operations was used to identify sky regions in an image and to make the image processing as robust as possible. The basic operations were the following (detailed instructions for using the program are given in Appendix E):

1. Loading and creating an image.
2. Extracting a colour channel.
3. Applying the Canny edge detector.

4. Applying the morphological closing algorithm.
5. Applying the FloodFill algorithm
6. Determining the area and perimeter of a region.
7. Determining the thresholds and structure element parameters.
8. Identifying a region to be a sky region.
9. Drawing or painting boundaries of the sky region.

4.3 Basic Operations with OpenCV

This section describes how the OpenCV image processing libraries and functions were used to carry out the basic operations listed in section 4.2.

4.3.1 Load and create image

The function that is used to load an image from file is cvLoadImage. The cvCreateImage function is used to create an image, for example when a copy of an image is made.

4.3.2 Extracting a colour channel

Colour model conversions available in OpenCV are greyscale, RGB, HSV, YCrCb, XYZ, and Lab. More details about these colour models are given in section 3.2.1.

The original RGB colour model can be converted to another colour model such as greyscale, HSV, YCrCb, CIE-XYZ and CIE-Lab by using the cvCvtColor function and using the cvSplit function to separate multiple colour channels in a colour model into individual channels.

Each colour channel contains different ranges in OpenCV. Most of them are in the range 0 to 255 except the “H” and “X” colour channels. The “H” colour channel is in the range 0 to

180 and the “X” colour channel is in the range 0 to 242. A zero value in the “S” colour channel represents no colour and 255 is a fully saturated colour.

4.3.3 Finding the edges in the image

The OpenCV function that was used to implement the Canny edge detection algorithm is cvCanny. Edge pixels are represented by 255 as a white colour and the remaining pixels are represented by 0 as a black colour in a new image.

The function cvCanny requires three parameters, the upper threshold, lower threshold, and sigma value. For manual experimentation these were set by creating scroll bars attached to the specified window with appropriate callback functions which applied the changed threshold setting to the image and displayed the result. The sigma value is used when creating the Gaussian kernel which reduces the effect of noise in the image. Its default value is 3 which is the smallest value that can be used in OpenCV. If a larger value of σ was used, it was found that the edge detection was very sensitive to image noise resulting in spurious edges.

4.3.4 Closing gaps in the edges

The morphological closing algorithm was applied to close the gaps in the edges found by cvCanny to create enclosed regions. The OpenCV morphology functions take a binary image (with values 0 and 255) as input, and produce another binary image as output.

Although OpenCV does not provide a closing function directly, the close morphology operation consists of dilation followed by erosion (Fisher R., 2003a). Within OpenCV these are implemented using the cvDilate and cvErosion functions respectively. A rectangular structuring element was created using cvCreateStructuringElementEx. To set the size of the structuring element manually a scrollbar with appropriate callback function was used.

The dilation or erosion operations are performed by placing the central pixel of the structure element on top of each pixel in the image. At the edge of the image where part of the structure element lies outside the image border the image is effectively extended assuming a mirror image reflection about the relevant edge.

4.3.5 Identifying all the pixels in a region

All pixels in a region are identified by using the FloodFill algorithm, as shown in Figure 3.19 (the FloodFill image). In that example, the region is identified by pixels with a grey colour (given pixel value 128).

The OpenCV function that is used to fill a region with a given colour is cvFloodFill. The cvFloodFill function (Vandevenne, 2004a) is given a starting pixel. It then finds and fills with a specific colour, all the pixels which are connected to the starting pixel and have the same pixel value as the starting pixel. In OpenCV, the testing for connected pixels can be set to 4 or 8 directions (The MathWorks, 1994). In this thesis, connectivity in four directions was used.

The starting pixel was specified manually by a mouse click on the image with an appropriate callback function. The function used to assign callback for mouse events is cvSetMouseCallback.

4.3.6 Displaying the filled regions

The FloodFill algorithm also creates a mask image. The mask image is an 8-bit image, 2 pixels wider and 2 pixels taller than the original, which corresponds to 802×602 pixels in this project. A pixel in the mask image that corresponds to pixel (x, y) in the original image will have coordinates (x+1, y+1) in the mask. The mask image shows which pixels have been filled in the FloodFill image. It is white (pixel values of 255) in the filled region and black outside of the filled region. In this thesis, the edge of the mask image is set to black (pixel value of 0). Figure 4.2 displays the mask image created when a region (Figure 3.19)

has been identified using cvFloodFill. The mask image can also be used for other purposes such as determining the perimeter of the filled image, as described below.

Figure 4.2: The final image result after identifying a region. The white colour represents the identified region. The black frame shown is part of the mask image (the same as Figure 3.20).

4.3.7 Determining the area and perimeter of a region

Determining the perimeter of the region

Rather than using a function in the OpenCV blob library (Azpiazu, 2007) to determine the perimeter of the region, it was found to be more convenient to use the mask image produced by cvFloodFill as follows.

The dilation morphology operation (Fisher R., 2003b) with a structure element size of 3×3 was applied to the white region of the mask image using cvDilate to make the region (pixel values of 255, white) larger by one pixel. The original mask image was then subtracted from the dilated mask image. The result after subtraction shows the boundary line around the region, as shown in Figure 4.3. The perimeter is then calculated by counting the white pixels in the image resulting from the subtraction using the cvCountNonZero function.

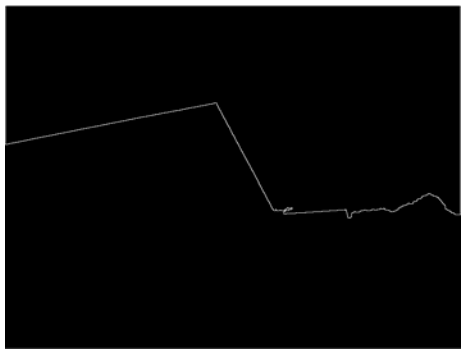


Figure 4.3: The boundary around the sky region of the image.

Determining the area of the region

The area of the region was found by using one of arguments in the FloodFill function. The argument has a property, called “area” which provides the number of pixels in the filled region.

4.4 Determining Edge Detector Upper and Lower Thresholds and the Structure Element Size

This section describes the details of the process outlined in section 3.4.1, to determine appropriate upper and lower thresholds for the Canny edge detector and the structure element size for the morphological closing algorithm. The following steps are involved:

1. The area and perimeter of the sky region in each of the sample images were determined. The region boundary of the sky was set by manually adjusting the upper and lower thresholds and the size of the structure element until the sky boundary was seen to coincide visually with the boundary obtained by inspecting the image. This information was stored in a database table, ImageInfo, with imageNumber as the primary key. In addition a binary code for each image was stored, as shown in Figure 4.4. For example, image 1 has the binary code 1, image 2 has the binary code 2, image 3 has the binary code 4 in general image n has the binary code for 2^{n-1}. Hence adding the binary codes of a set of images will result in a unique number. For example, 9 is image 1 and image 4.

ImageInfo
- ImageNumber - Area - Perimeter - BinaryCode

Figure 4.4: The ImageInfo table.

2. The area and perimeter of the sky region of each of the sample images were determined using a range of values for the upper and lower thresholds and size of structure element. The area and perimeter were determined for all combinations for the following ranges:

   - upper threshold 0 to 999 increment 1
   - lower threshold 0 to 999 increment 1
   - structure element size 3 to 21 increment 2.

   This resulted in the area and perimeter being determined 10,000,000 times for each of the 8 images. For each combination of settings the following information was written to the database table Information as shown in Figure 4.5: ImageNumber, UpperThreshold, LowerThreshold, StructureElement, Area, and Perimeter. The area and perimeter were determined as described in section 4.3.7. As the sky was at the top of all the test images, the FloodFill starting pixel for determining the area of the sky region was found by starting at the top left and scanning down until the first black pixel (i.e. the first pixel that is not an edge in the morphology closing image) was encountered.

Information
- ImageNumber - UpperThreshold - LowerThreshold - StructureElement - Area - Perimeter

Figure 4.5: The Information table.

3. The ranges of values for upper threshold, lower threshold and the structure element size were determined using the database query, as shown in Figure 4.6. The parameters were determined when the tolerance of the area in the query was reduced until at least one of the test images failed to meet the area criterion. The smallest value of the parameter toleranceArea for which all the test images met the criterion was 0.3 %.

```
SELECT      i.LowerThreshold, i.UpperThreshold, i.StructureElement,
            SUM( DISTINCT b.BinaryCode) AS sumBinary
FROM        ImageInfo AS b INNER JOIN Information AS i ON
            b.ImageNumber = i.ImageNumber AND b.Area * (1 -
            toleranceArea) < i.Area AND b.Area * (1 + toleranceArea)
            > i.Area
GROUP BY    i.LowerThreshold, i.UpperThreshold, i.StructureElement
HAVING      (SUM(DISTINCT b.BinaryCode) = 255)
```

Figure 4.6: The query used to find the upper and lower threshold values, the structure element size and the sum of the binary code, for all the images that meet the area criterion.

Note that the query in Figure 4.6 sums the binary code for all the images in the range, so that it is possible to deduce which images do not meet the criteria. For example, if the area of the sky region from step 1 is 30,000 pixels, then the acceptable area would be in the range 29,910 to 30,090. If the summed binary code is 255 then all images meet the criteria. If the total binary code is 183 all the images except images 4 and 7 meet the criteria.

4. The ranges of values for upper and lower thresholds and structure element size were further refined by testing various tolerances for the perimeter of the sky region (tolerancePerimeter) while the area tolerance was held at 0.3% as shown in Figure 4.7. The smallest value of the parameter tolerancePerimeter for which all the test images met the criterion was 5%.

```
SELECT      i.LowerThreshold, i.UpperThreshold, i.StructureElement,
            SUM( DISTINCT b.BinaryCode) AS sumBinary
FROM        ImageInfo AS b INNER JOIN Information AS i ON
            b.ImageNumber = i.ImageNumber AND b.Area * (1 -
            0.003) < i.Area AND b.Area * (1 + 0.003) > i.Area AND
            b.Perimeter * (1 - tolerancePerimeter) < i.Perimeter AND *
            b.Perimeter (1 + tolerancePerimeter) > i.Perimeter
GROUP BY    i.LowerThreshold, i.UpperThreshold, i.StructureElement
HAVING      (SUM(DISTINCT b.BinaryCode) = 255)
```

Figure 4.7: The query used to find the upper and lower threshold values, the structure element size and the sum of the binary code for all the images that meet the area and perimeter criteria.

5. The ranges of each parameter were determined which were within a tolerance of 0.3% of the correct area, and within a tolerance of 5% of the correct perimeter.

4.5 Determining Sky Regions Automatically

The approached described in section 3.4.2 to determine sky regions in an image automatically was based on the pixel brightness and area of the regions. This section describes the details of this process.

4.5.1 Determining the FloodFill starting pixels

The pixel brightness was used to determine FloodFill starting pixels in an image as follows.

1. A histogram of the blue channel values in the image was created using the cvCreateHist function in the range 0 to 255
2. An array of size 256 stores the number of pixels for each value.
3. The brightest value was determined by using cvMinMaxLoc function.

4. The number of pixels with each value starting from the brightest value in the image was added until the cumulative sum of the number of pixels was greater than or equal to 20% of the total number of pixels in the image (480000 pixels for the test images). The brightness threshold was the pixel value at which the 20% limit was reached. The choice of 20% was discussed in Chapter 3 section 3.4.2.
5. A pixel is classified as a FloodFill starting pixel if its value is greater than or equal to the brightness threshold.

4.5.2 Determining the actual sky region

The process used to determine which of the identified regions are sky regions is as follows.

1. A temporary image was created by using cvCreateImage function.
2. Each pixel in the image was scanned column by column starting at the top left. The cvGet2D function was used to get each pixel value. If a pixel's value was greater than the brightness threshold and it was not on an edge and not already identified as being part of a region, the pixel was treated as a FloodFill starting pixel.
3. The FloodFill algorithm was used to identify all pixels in the region in the temporary image. Note, one region can contain several FloodFill starting pixels but the first FloodFill starting pixel that was found in the region was stored.
4. The position (x, y) of the FloodFill starting pixel and the area of the region were stored in arrays X, Y and A respectively.
5. The arrays X, Y and A were ranked by decreasing order according to the area A
6. Successive elements of A were summed until the next element to be added was less than 50% of the accumulated sum so far. The regions included in the sum were classified as sky regions. The value 50% was discussed in Chapter 3 section 3.4.2.
7. A result image was created by using the cvCreateImage function and filled using the FloodFill starting pixels (from arrays X and Y) for the regions identified as sky regions.

4.6 User Interaction

There are some situations where user interaction techniques are required, as described in section 3.5. The first situation is when the boundary of a sky region has not been correctly identified. This is dealt with by drawing the sky boundary manually. The other situation is when a region is incorrectly identified as being sky or not sky. This is dealt with by painting the sky region or non-sky region manually.

(a) (b)

Figure 4.8: Failure of the image processing to correctly identify the sky region. (a) the original image, (b) after image processing.

Figure 4.8 shows an example where the sky region is not correctly identified because the image was taken directly into the sun. The first stage to correct this is to draw the sky boundary manually. The following describes how a line drawing is displayed in the final image result (Figure 4.8(b)).

4.6.1 Boundary not correctly identified

A line drawing technique is applied to the Canny edge detector image. The cvLine function is used to draw a line in an image between two points. A line can be drawn by clicking the mouse button at the starting position in an image and dragging the mouse to draw a line and then releasing the mouse when it is finished. The mouse events were handled using the callback function cvSetMouseCallback. After the line is drawn, the morphology closing algorithm is then used to close any small gaps that might remain in the boundaries. The brightness and area criteria are then used to determine the actual sky regions. The final image result is then updated, as shown in Figure 4.9. However, in this example, the image

still has a problem as a region that is not sky has been identified as sky. This will be addressed in the next section.

Figure 4.9: The sky boundary corrected using a line drawing technique.

4.6.2 Manually identifying the sky or non-sky regions

The building region is incorrectly identified as a sky region, as shown in Figure 4.9 because the area is big enough to be sky region and the region contains bright pixels. Therefore, the building region needs to be manually designated as a non-sky region. The following process allows the user to manually change a region from being sky to non-sky or vice versa.

1. Create a copy from the image result (Figure 4.9) which is a binary image, called the clone image result.
2. Add all the edges of the regions from the morphology closing image (Figure 4.10) to the clone image result but with the value of the edges set at 128 rather than the original 255. This helps the user to see the edges of the regions when they want to distinguish a sky region from a non-sky region as shown in Figure 4.11. In addition, it allows the user to click on sky and non-sky regions but avoid the edges of the regions in the image.

   The functions that are used to get and set the colour of the edges of the region are cvGet2D and cvSet2D.

Figure 4.10: An example of a morphology closing image.

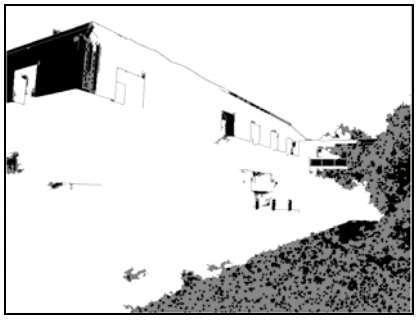


Figure 4.11: The clone image result with the edges of the image (pixel value of 128) included.

3. Users click in the region in the clone image (Figure 4.11) that they want to change from sky to non-sky or vice versa. The FloodFill algorithm is then applied to identify the sky or non-sky regions, as shown in Figure 4.12. A FloodFill starting pixel is stored. The cvSetMouseCallback function is used to execute the callback for the mouse event.

Figure 4.12: Identifying the non-sky region in the clone image result.

4. Apply the FloodFill algorithm back to the morphology closing image by using the FloodFill starting pixel identified by the user in step 3. A new mask image is then created, as shown in Figure 4.13. The new mask image represents the region that has been changed to non-sky region.

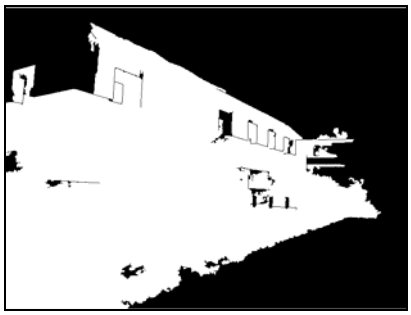


Figure 4.13: The new mask image showing the region that has been reclassified as a non-sky region.

5. Combine the new mask image with the image result (Figure 4.9) to display the non-sky regions in the image result, as shown in Figure 4.14.

Figure 4.14: The image result after applying the user interaction techniques.

5 Discussion

An image processing technique, described in Chapters 3 and 4, has been developed to identify sky regions of an image. The resulting images enable accurate prediction of the solar exposure at a location at any time of the year. The general approach is to use edge detection methods to determine the boundaries of all contrasting regions in the image and then use brightness and area criteria to determine which of the regions are sky. Various factors such as the use of light filters and the choice of the most appropriate colour channel from which to determine the regions were investigated. In particular, an extensive empirical study to determine the most appropriate threshold settings for the Canny edge detection algorithm was carried out using a set of test images.

This study has determined an approach for identifying the sky, which works successfully with the set of test images. The test images cover a variety of different sky conditions and the technique should therefore work well with other images that are used to predict solar exposure. However, there are a number of issues which require further discussion and which ultimately may affect the practical implementation of the approach to determining the sky regions of an image. These issues are discussed in this chapter.

It is important to remember that although it is desirable to be able to find the sky regions in an image automatically, it is not essential to achieve 100% success. A single determination of solar exposure at a location is likely to involve only about 7 – 10 images and it is therefore feasible for the user to inspect the results of the sky detection for each image and manually make adjustments where it has not worked ideally. This contrasts with situations where there are thousands of images to analyse, such as searching a database of images for all those which contain a significant amount of sky.

5.1 Camera Settings

In some image processing situations, it may be appropriate to control the camera's settings such as white balance, zero offset and gamma, to optimise the image so that subsequent image processing is more likely to succeed. However, images to predict solar exposure will be taken in a very wide range of conditions and it is therefore inappropriate to control individual camera settings. Rather, the approach that has been taken is to make the image processing as robust as possible by ensuring that it works successfully on a wide range of images, such as those in the test set, using automatic settings.

5.2 Test Images

The test images that were used were taken in a range of conditions including bright sky – no cloud, bright sky – scattered clouds, overcast sky – white clouds, and overcast sky – grey clouds. These images also include some non-sky regions with brightness similar to the actual sky regions, some non-sky regions containing a shiny surface and some regions where there is low contrast between the sky and the rest of the image. However, the test images used for this thesis do not include an image for an overcast sky – black clouds, as there was not one available. Also there were no test images containing regions such as snow or water which might have properties similar to sky. However, as long as there is sufficient contrast between the sky and other objects or regions in the image for the Canny edge detector to identify the edges (Green, 2002), it is expected that the threshold values and structure element size, as described in section 3.4.1 will work satisfactorily.

5.3 Colour Channel

Several colour channels were tested to determine the best contrast between the sky and the rest of an image. Although the blue colour channel gave consistently good contrast for all images, the Z colour channel was comparable and is likely to give very similar results. The blue colour channel might not be the best choice for other images such as those that include black clouds or sea or snow. This needs to be tested. A more general approach

could be to apply the Canny edge detector to all colour channels in an image and select the channel which gives the strongest edges.

5.4 Filters

A number of camera filters including polarizing, ultraviolet and skylight filters were tested to see if they increased the contrast between the sky and the rest of the image. The polarizing filter did not significantly increase the contrast and the difference in the average pixel values with and without the polarising filter was small. In addition, the polarizing filter was clumsy to use as it required manual rotation of the filter to obtain maximum contrast. However, the ultraviolet and skylight filters may still have merit. Further investigation is required, especially regarding the availability and effectiveness of cameras specifically designed to detect ultraviolet light.

5.5 Edge Detection

The Canny edge detector was used to find the boundaries of the contrasting regions in the images. This technique, described in section 3.3.2, is widely used because it reduces the possibility of missing actual edges and detecting false edges. Other possible approaches for finding the regions include the use of thresholding followed by binary mathematical morphology, or using the watershed technique (Sonka, 1998; TUDelft, 2006). However, the Canny edge detector worked well on the test images and there appears to be little to be gained by exploring alternative approaches.

5.6 The Appropriate Upper Threshold, Lower Threshold, and Size of the Structure Element

As described in section 3.4.1, the ranges of upper threshold, lower threshold and the size of the structure element that work well to identify sky regions for all the test images was 186 to 217, 16 to 61, and 3 respectively. However, the values of the upper and lower threshold, and the size of the structure element may depend on a variety of factors such as the image resolution and the particular implementation of the Canny edge detector.

The values of the upper and lower thresholds may depend on the image resolution (800 × 600 was used in the tests carried out in Section 3.1.2). A bicubic algorithm (Bouton, 2003) was used to sub-sample images to obtain that image resolution. The algorithm calculates pixel values for horizontal, vertical and diagonal directions and uses a weighted average of the total colours for any given new pixel. When an image is sub-sampled to be smaller than the original image, some detail such as the quality or sharpness of the image may be lost. If this detail was retained, it may appear as artefacts, such as undesirable cloud edges, and require higher settings for the edge detection thresholds to filter them out.

The values of the threshold parameters also depend on the particular implementation of the Canny edge detector and morphological closing algorithm used (OpenCV in this case). For example, another implementation requires the thresholds to be specified in the range 0 to 1 (IDL, 2007). Therefore, although the current work has demonstrated that it is possible to find ranges for the threshold parameters that will successfully find the sky region in the set of test images, the experimentation to find that range would have to be repeated if another implementation of the Canny edge detector was to be used.

Figure 3.22 shows that the upper and lower threshold values can each vary over quite a wide range but still identify the sky regions in all the test images with an error in area no greater than 0.3% and an error in perimeter no greater than 5%. The wide ranges in parameters occur because, when the thresholds were gradually increased from 0, the point was reached at which the error in the area of the sky region was no greater than 0.3% for any image. As the thresholds were increased further, the area in the sky region was quite stable, changing little until the upper limit of the range was encountered. In other words, the area and perimeter of the sky region were found to be insensitive to the threshold values over quite a wide range. This is encouraging as it suggests that threshold values within these ranges are likely to work well across a wide range of actual images used to determine solar exposure.

When the results of this investigation are applied to a new image, only a single value for each of the upper and lower thresholds needs to be specified. All the values in the range

remove unwanted edges in the sky area. There will be many edges remaining in the non-sky area and the different upper and lower thresholds will affect the regions that are bounded by these edges. The lower values of both thresholds result in smaller regions. These smaller areas are less likely to be mistakenly identified as sky in subsequent processing. As described in section 3.4.2, determining which areas are sky is based on the brightness and size of the region. If low values of the thresholds are used, more non-sky regions are found, but the area of each is smaller. Smaller regions are less likely to be large enough to be classified as sky regions. Therefore, it is appropriate to use the lowest values of the upper and lower threshold ranges, 186 and 16 respectively.

As discussed in section 3.4, it is the area of sky region which will have the most effect on the accuracy of the solar exposure calculations. Figure 3.22 shows that if the values of the upper or lower thresholds are lower than 186 and 16 respectively, then only the perimeter criterion for test image 4 is violated. In this case additional cloud edges are apparent. These would have virtually no effect on the area of the sky region which suggests that using threshold values at the low end of the range incurs little risk.

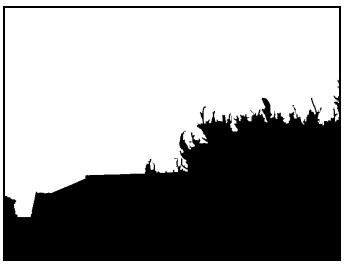
5.7 Identifying the Sky Region

As described in Chapter 3, the image processing was made as robust as possible so that the different sky regions in the test images were identified accurately. In section 3.4.2, an approach is described to determine which of the identified regions are sky and which are not. This approach is based on the brightness and area of each region.

There are some situations where some regions in an image, such as sky regions between trees branches, are not correctly identified as shown in Figure 5.1. This is because the size of the region is not big enough to meet the area selection criterion, as described in section 3.4.2. Also, the region on the edge of an image might be too small to be identified as a sky region even though it would be part of a bigger sky region when it is connected to an adjacent image.

In addition to the methods we have already described there are other aspects of the image we could consider in order to determine the sky regions. We could use information about the location and colour of pixels. For example, a region is more likely to be sky if it is at the top of the image or in a similar position to other identified sky regions. The colour information from the sky regions that have already been identified might be useful in determining whether other small regions might be sky. For example small areas of sky between tree branches as shown in Figure 5.1(a) might be able to be identified correctly by comparing their pixel colour with the larger sky area.

(a)

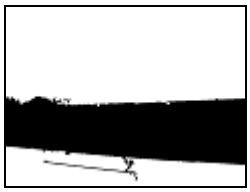


(b)

Figure 5.1: Identification of small regions of sky. Image (a) shows the original colour image. Image (b) shows the result of identifying the sky region (shown in white). The regions between the tree's branches are not classified as sky because the regions between the tree branches are too small. The edges of the tree branches in the image (b) are wider than in the original image (a) because the morphology closing algorithm tends to make them thicker. Nevertheless, the total area of the sky region for this image is still within 0.3% of the correct area.

Another situation where the region is incorrectly identified as a sky region is shown in Figure 5.2 where the bottom part of the image is identified as sky. These incorrectly identified regions may not affect the solar exposure calculation because the sunpath never passes through that area. However, it might be possible to improve the identification of sky regions by using the location of the region as a guide, i.e. regions at the bottom of an image are unlikely to be sky.

(a)



(b)

Figure 5.2: Image (a) shows the original colour image. Image (b) shows a sky region incorrectly identified in the bottom part of the image.

6 Conclusions and Future Work

6.1 Summary

For this thesis, “solar exposure at a location” is defined as the ability of a ray from the sun to intersect a particular point on the earth at any particular time of the day for any day of the year. Knowing the solar exposure at a particular location is useful for different purposes including architecture and solar panel location. For example, it may be beneficial for an architect who wants to set the orientation of windows to receive the most sunlight in the morning.

The purpose of this thesis is to develop image processing techniques for identifying the sky regions in an image. Correctly identifying the sky regions enables accurate prediction of the solar exposure at a location. The image processing should be as robust as possible so that it accurately identifies the sky from the rest of the image under a wide range of conditions, for example both clear and overcast skies, with different types of cloud.

The general approach of our suggested solution is to extract the blue colour channel from the image and then determine the boundaries of all contrasting regions, including the sky regions, using the Canny edge detector. The closing morphology algorithm is then applied to ensure that the regions are separated from each other. The next step is to automatically determine which of the regions are sky by using brightness and area criteria. The FloodFill algorithm is then applied to identify all pixels in the sky regions.

An extensive empirical investigation was undertaken to determine the values for the upper and lower thresholds for the Canny edge detector and the size of the structure element for the morphology closing algorithm so that the edges of the sky regions are correctly identified in a range of test images. It was found that the boundaries of the sky region were accurately determined across all test images when there is a 3×3 structure element and the upper threshold is in the range 186 to 217 and lower threshold is in the range 16 to 61. It is

recommended that the lowest values of each threshold, 186 and 16 respectively, are used because with these values, small areas in the non-sky region are less likely to be classified as sky regions.

Although the approach described in this thesis should enable the sky region to be identified in a wide range of images, there will be some situations where image processing alone does not correctly determine sky area boundaries, for example, if an image is taken directly into the sun or sunlight reflects off very shiny surfaces. These situations will require some user interaction to identify the boundaries of the sky area. Thus we allow the user to draw the sky boundary and manually identify the sky region, or non-sky region. This facility is also provided in some of the commercial devices, such as the Solmetric SunEye.

The techniques developed in this thesis work successfully across a wide range of images and can be readily combined with software which calculates the sunpath and traces the path of ray through the processed images to predict solar exposure at a location at any time of the day for any day of the year.

6.2 Future Work

Although the image processing developed in this thesis should work successfully to identify the sky regions in a wide range of images, there are a number of aspects which could be explored further and potentially make the image processing even more reliable. The list below outlines suggested future work.

6.2.1 Test Images

The test images used in this thesis included a wide range of sky conditions including bright sky – no cloud, bright sky – scattered clouds, overcast sky – white clouds, and overcast sky – grey clouds. However, the test images did not include an image for an overcast sky – black clouds, nor an image that contains regions such as snow or water. It would be useful

to test images taken in those conditions to see if the threshold settings are appropriate and modify them if necessary.

6.2.2 Ultraviolet camera

Ultraviolet and skylight filters were tested to see whether they increased the contrast between the sky and the rest of an image. However, the filters did not seem to increase this contrast because the digital camera used is not sensitive to ultraviolet light (Great Landscape Photography, 2004). It would be appropriate to test a camera that is designed to detect ultraviolet light and see if a significant improvement in contrast can be achieved.

6.2.3 Edge detection

The Canny edge detector was used to determine the boundaries of all the regions in an image. The upper and lower threshold values for the Canny edge detector that are used must be set to avoid identifying edges around any clouds in the sky. The appropriate choice of upper and lower threshold values is likely to depend on the image resolution and the particular implementation of the Canny edge detector being used. This would require the investigation to determine appropriate threshold values to be repeated. Other approaches to finding the regions, such as the binary mathematical morphology or watershed (Sonka, 1998; TUDelft, 2006), could be used, although it is likely that empirical testing would still be required.

6.2.4 Identifying the sky region

Once all the regions in the image have been identified, brightness and area criteria automatically determine which are the sky regions. The area of the region needs to be both big enough and contain a bright pixel to be classified as a sky region. This approach could be enhanced by using other information such as the position of each region in the image and the colour of the pixels. For example, if the region is near the top of the image or the region is in a similar position to other identified sky regions then it is more likely to be sky. Colour information might help to identify small regions of sky, such as those between tree branches, by comparing the colour of those regions with the colour of larger areas of sky

which are nearby. Another improvement would be to consider as possible sky regions only those regions that would intersect rays from the sun. It is not necessary to consider regions that are outside the sunpath, as shown in Figure 6.1.

Figure 6.1: Regions that are outside the sunpath do not need to be classified as either sky or non sky regions (Wiley, 2005).

6.2.5 User interaction

There are some situations where the image processing may not correctly identify the sky. User interaction such as painting the sky or non-sky regions can be used. In the current user interaction process, the colour of the edges of the regions was used to identify the regions which is useful when users want to identify the sky region from non-sky region. However, using grey colour may not be clear enough to identify the region. An improvement could be to make the resulting image partly transparent and overlay it onto the original coloured image.

The importance of being able to predict solar exposure is well recognized and the existence of commercial devices such as the Solmetric SunEye attests to this. The work reported in this thesis contributes to this technology by showing how images can be processed to effectively identify the sky regions.

7 References

Alexander, T. (1998). *Determining the viewing angle of a camera*. Retrieved April, 2007, from http://www.dr-lex.be/qtvr/viewangle.html

Anderson, S. (2007). *Multimedia on the Internet (Bitmap vs. Vector Images)* Retrieved March 08, 2008, from http://www.usu.edu/sanderso/multinet/bitvec.html

Apple Computer Inc. (1996). *Color Spaces*. Retrieved from http://developer.apple.com/documentation/mac/ACI/ACI-48.html

Autodesk. (2006). *Ecotect: Solar Analysis*. Retrieved March 15, 2007, from http://squ1.com/products/ecotect/features/solar

Autodesk. (2007). *SUN-PATH: SUN POSITION*. Retrieved January 23, 2007, from http://squ1.org/wiki/Sun-Path/Reading_Sun_Positions

Azpiazu, J. (2007). *Blob extraction library*. Retrieved June 20, 2008, from http://opencvlibrary.sourceforge.net/cvBlobsLib

Bit Jazz. (2004). *Synchromy, The Perfect Color Converter*. Retrieved July 30, 2008, from http://www.bitjazz.com/en/products/synchromy/

Bornet, O. (2006). *Welcome to the OpenCV Wiki*. Retrieved March 06, 2007, from http://opencvlibrary.sourceforge.net/

Bourke, P. (1998). *What is Gamma Correction? or Why do some pictures appear dark on some displays*. Retrieved July, 2008, from http://ozviz.wasp.uwa.edu.au/~pbourke/texture_colour/gamma/

Bouton, G. D. (2003). *Computer Art Lesson: Anti-Aliasing and Resampling Artwork - Part 2*. Retrieved July 28, 2008, from http://www.graphics.com/modules.php?name=Sections&op=printpage&artid=56

Cheung, K. P. (1997). *A Portable Universal Solar Tracker Sundial Attached to the Globe, A Universal Sun-Earth Direction-Time Relationship Demonstrating Device*. Retrieved March 07, 2007, from http://www.arch.hku.hk/~kpcheung/sundial/paper10-1/paper.htm

CLEAR. (2004a). *Sun Calculator*. Retrieved May 15, 2007, from http://www.learn.londonmet.ac.uk/packages/clear/visual/daylight/sun_sky/sun_calc.html

CLEAR. (2004b). *Sunpath Diagram*. Retrieved March 15, 2007, from http://www.learn.londonmet.ac.uk/packages/clear/visual/daylight/analysis/hand/images/sunpath_diagram/london_orthographic1.png

Colour Management Consultancy. (1980). *Introduction to Colour Space - CLE Lab & LCH*. Retrieved July 30, 2008, from http://www.colourphil.co.uk/lab_lch_colour_space.html

Dalrymple, M. B. (1987). Device to indicate solar exposure: US Patent 4,635,371

dpFWIW. (2004). *Filter options for digital cameras*. Retrieved August 03, 2007, from http://dpfwiw.com/filters.htm

dPS. (2007). *Introduction to White Balance*. Retrieved June, 2008, from http://digital-photography-school.com/blog/introduction-to-white-balance/

Energy Information Administration. (2006). *Solar-Energy from the sun*. Retrieved from http://www.eia.doe.gov/kids/energyfacts/sources/renewable/solar.html

Fisher R., P. S., Walker A., & Wolfart E. (2003a). *Closing* Retrieved August 05, 2007, from http://homepages.inf.ed.ac.uk/rbf/HIPR2/close.htm

Fisher R., P. S., Walker A., & Wolfart E. (2003b). *Dilation*. Retrieved February 28, 2007, from http://homepages.inf.ed.ac.uk/rbf/HIPR2/dilate.htm

Fisher R., P. S., Walker A., & Wolfart E. (2003c). *Feature Detectors*. Retrieved November 25, 2007, from http://homepages.inf.ed.ac.uk/rbf/HIPR2/featops.htm

Fisher R., P. S., Walker A., & Wolfart E. (2003d). *Grayscale Images*. Retrieved July 1, 2007, from http://homepages.inf.ed.ac.uk/rbf/HIPR2/gryimage.htm

Fuglies. (2003). *Sunglasses made to fit your fugly head*. Retrieved April 13, 2008, from http://www.fuglies.com.au/polarised_sunglasses.html

Fulton, D. (2002). Canopy modification using computer modelling: US Patent 6,338,027

Gallagher, A. C., Luo, J., & Hao, W. (2008). Detection of sky in digital color images: US Patent 7,336,819

GNS. (2007). *But does it get sun in winter?* Retrieved April 7, 2008, from http://www.gns.cri.nz/solar/

Great Landscape Photography. (2004). *The Ultraviolet Filter*. Retrieved August 20, 2008, from http://www.great-landscape-photography.com/ultraviolet-filter.html

Green, B. (2002). *Canny Edge Detection Tutorial*. Retrieved March 16, 2007, from http://www.pages.drexel.edu/~weg22/can_tut.html

Griffin Jr, R. N. (1980). Apparatus for and method of evaluating solar exposure: US Patent 4,236,313

Harkness, E. L., & Mehta, M. L. (1978). *Solar Radiation Control in Buildings*: Elsevier Science Ltd.

Hearn, D., & Baker, M. P. (2004). *Computer Graphics*: Pearson Prentice Hall. US.

Hong, H., & Yu, J. (2005). Gone Fishin': Seasonality in Trading Activity and Asset Prices http://www.mgmt.utoronto.ca/finance/seminars/051111_hong.pdf

HowStuffWorks. (2004). *How Camera Flashes Work*. Retrieved June, 2008, from http://electronics.howstuffworks.com/camera-flash.htm

Huang, J. (2007). *Color and Graphics Displays*. Retrieved July 27, 2008, from www.cs.utk.edu/~huangj/CS594S06/color.ppt

Hunter Lab. (2008). CIE L*a*b Color Scale. *2008*(7), 1-4. Retrieved from http://www.hunterlab.com/appnotes/an07_96a.pdf

IDL. (2007). *Canny*. Retrieved July 20, 2008, from http://idlastro.gsfc.nasa.gov/idl_html_help/CANNY.html#wp1362362

IES. (2007). *Solar:Suncast*. Retrieved March 12, 2007, from http://www.iesve.com/content/default.asp?page=s25_1

Indiana University. (2004). *Frequently Asked Questiions/Troubleshootings/HOWTOs: OpenCV Help*. Retrieved March 27, 2007, from http://www.cs.indiana.edu/cgi-pub/oleykin/website/OpenCVHelp/

Iqbal, M. (1983). An introduction to solar radiation.

Lewis, D. F. (1981). Device to indicate solar exposure: US Patent 4,288,922

Luo, J., & Etz, S. (2003). Method for detecting sky in images: US Patent 6,504,951

Macdonald, W. (2007). Solar Access Measurement Device: WO Patent WO/2007/089,345

McKinnon, A. E. (2007). Estimate Solar Exposure at a Particular Location - Demonstration of Concept.

MoDule. (2004). *Sun Earth Angle Relationship*. Retrieved May 28, 2007, from http://www.courses.ait.ac.th/ED06.22/course1/lecs/module1/m11o98.html

Moran, T. W. (2006). Sunniness Indicator: US Patent 20060112575

Owner-Petersen, P., & Lund-Hansen, K. B. (1980). Sunlight calculator: US Patent 4,186,297

Roushdy, M. (2006). Comparative Study of Edge Detection Algorithms Applying on the Grayscale Noisy Image Using Morphological Filter}. *ICGST International Journal on Graphics, Vision and Image Processing}, 6*, 17-23. Retrieved from http://www.icgst.com/gvip/Volume6/Issue4/P1150649005.html

Scantips. (1997). *JPEG-Joint Photographic Experts Group*. Retrieved March, 12, 2008, from http://www.scantips.com/basics9j.html

SecurityideasVAR. (1999). *Angle of View Calculation*. Retrieved April 25, 2007, from http://www.securityideasvar.com/browseproducts/Angle-of-View-Calculation.html

Solar Pathfinder. (2008). *Solar Pathfinder*. Retrieved April 14, 2008, from http://www.solarpathfinder.com/

Solmetric. (2007). *The Solmetric SunEye*. Retrieved December 12, 2007, from http://www.solmetric.com/

Sonka, M., Hlavac, V., & Boyle, R. (1998). *Image Processing, Analysis, and Machine Vision. Watershed*: PWS. Retrieved from http://www.icaen.uiowa.edu/~dip/LECTURE/Segmentation3.html#watershed

Souza, L. C. L., Rodrigues, D S., & Mendes, J F G. (2003). *Sky View Factors Estimation Using a 3D-GIS Extension*. Retrieved, March 19, 2007, from http://www.ibpsa.org/proceedings/BS2003/BS03_1227_1234.pdf

Szokolay, S. (1996). Solar Geometry. *Queensland, University of Queensland*, pp. 9-14.

The MathWorks. (1994). *Image Processing Toolbox: Pixel Connectivity*. Retrieved July, 2008, from http://www.mathworks.com/access/helpdesk_r13/help/toolbox/images/morph12.html

TUDelft. (2006). *Segmentation*. Retrieved June 12, 2008, from http://www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-Segmenta.html

Vandevenne, L. (2004a). *Lode's Computer Graphic Tutorial, Flood Fill*. Retrieved August 1, 2007, from http://student.kuleuven.be/~m0216922/CG/floodfill.html

Vandevenne, L. (2004b). *Lode's Comuter Graphics Tutorial: Light and Color*. Retrieved March 09, 2008, from http://student.kuleuven.be/~m0216922/CG/color.html

Virtual Drums. (2005). *Frozen Cameleon*. Retrieved July 30, 2008, from http://www.virtual-drums.com/frozen-cameleon.php

Wikipedia. (2007). *User:PAR/Work3*. Retrieved July 30, 2008, from http://en.wikipedia.org/wiki/User:PAR/Work3

Wiley, E. (2005). *Acme Solar Site Evaluation Tool (ASSET)* 14. Retrieved April, 2008, from http://www.we-llc.com/ASSET.html

Appendix A

The Angle of View of the Camera

After the image processing identifies the sky region in an image, the resulting image can be used to predict the solar exposure at a location. The angle of view of the camera can then be applied to determine whether the rays from the sun can reach the location by intersecting the sky or non-sky regions of an image, as described in detail in Appendix C.

There are two approaches to determining the angle of view of the camera in both the vertical and horizontal directions. The first approach (Alexander, 1998) uses measurements of the width and height of objects in the image along with the distance between the objects and camera, and the other (SecurityideasVAR, 1999) is to use the focal length of the camera lens. The following explanation describes the method of determining the angle of view of the camera in the vertical and horizontal directions using both approaches.

Determining the angle of view (x) of camera in the vertical direction

The measurement approach

Figure A.1 is used to determine the angle of view of camera in the vertical direction (x) by measuring the distance between the camera and the plane containing the object (D), and the actual height of the object captured in the image (H) then,

$$\tan(\frac{x}{2}) = \frac{(H/2)}{D} \tag{A.1}$$

hence

$$x = 2 \times \arctan(\frac{H}{2D}) \tag{A.2}$$

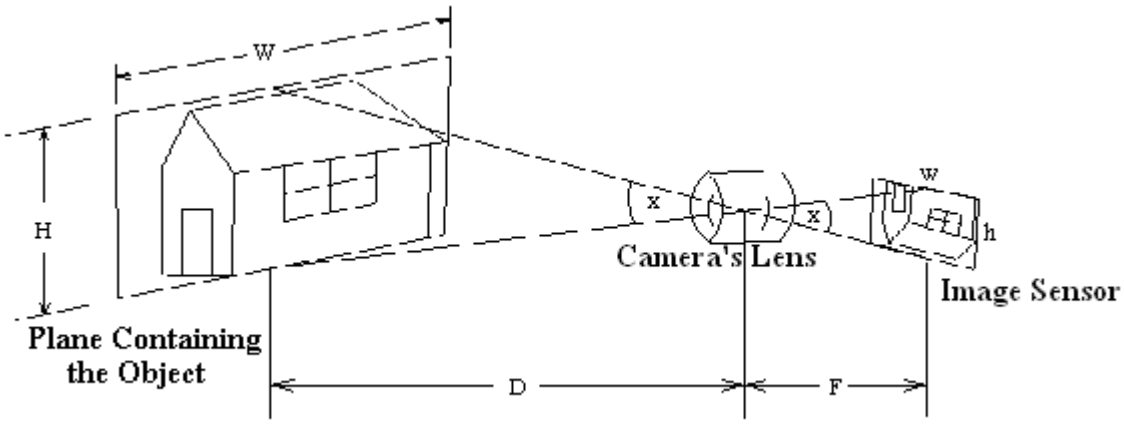


Figure A.1: Determine the angle in the vertical direction (adapted from SecurityideasVAR, 1999).

The focal length approach

The other approach to determine the angle of the camera in the vertical direction is to use the focal length of the camera lens (F) and the height of image (h), as shown in Figure A.1 then,

$$x = 2\arctan(\frac{h}{2F}) \tag{A.3}$$

Determining the angle of view (α) of camera in the horizontal direction

The measurement approach

The angle α can be calculated by using a similar approach as equation (A.1). If D is the known distance between the camera and the plane containing the object, and W is the actual width of the scene captured in the image (Figure A.2) then,

$$\tan(\frac{\alpha}{2}) = \frac{(W/2)}{D} \tag{A.4}$$

hence

$$\alpha = 2\arctan(\frac{W}{2D}) \tag{A.5}$$

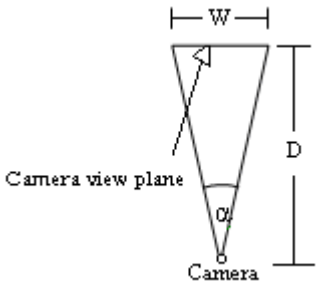


Figure A.2: Determining the angle of view (α) in the horizontal direction.

The Focal length approach

Another method to determine the angle of the camera in the vertical direction is to use the width of the image (w) and the focal length of the camera lens (F) as shown in Figure A.1 then,

$$\alpha = 2\arctan(\frac{w}{2F}) \tag{A.6}$$

Appendix B

The Position of the Sun and Reading the Position of the Sun from Sunpath Diagrams

The information about the sun's path through the sky at a particular time, latitude and date is well established and is presented in equation (B.1) on page 88.

The position of the sun at any time of the day on any day of the year can be described by two angles (Iqbal, 1983): the altitude angle (ε) and azimuth angle (β), as shown in Figure B.1. The altitude angle describes how high the sun appears in the sky. The angle is measured between a line from a point on the earth's surface to the centre of the sun and the horizontal plane that passes through the point on the surface. The azimuth angle is the angle of the sun in a horizontal plane measured from true south.

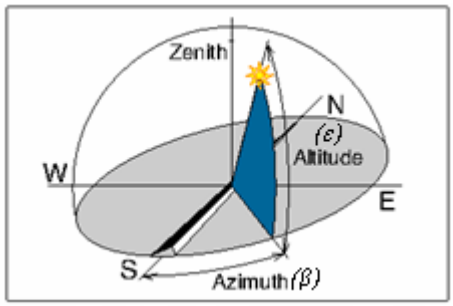


Figure B.1: The altitude (ε) and azimuth (β) angles of the sun (CLEAR, 2004a).

The position of the sun in the sky at any time of the day on any day of the year can be determined mathematically as described in the next section

Determining the position of the sun using mathematical equations

The equations that determine the movements of the sun are well established (Iqbal, 1983). We summarize some of the main points here.

- **Determining the azimuth angle of the sun (β)**

β is the angle of the sun azimuth measured in an anticlockwise direction from south, see Figure B.1. The azimuth angle is calculated as follows:

$$\sin\beta = \frac{\cos\delta \sin\omega}{\cos\varepsilon} \tag{B.1}$$

The following definitions of the angles are used:

The solar declination angle (δ)

δ is the declination angle which is between the earth-sun line and the equatorial plane. Declination is independent of the location (i.e. latitude) and changes with the date. The declination angle value is between +23.45° and -23.45°. For example, in the southern hemisphere it has the following values (Harkness & Mehta, 1978).

δ = -23.45° at the summer solstice (22nd December)

δ = +23.45° at the winter solstice (22nd June)

δ = 0 at the autumn and spring equinoxes (Approximately 22nd March and 22nd September)

The equation for calculation of the declination angle in degrees, is

$$\delta = 23.45\sin((\frac{(N-80)}{370})*360) \tag{B.2}$$

where: N is the number of the day in the year (starting from 1 on January 1st).

The hour angle (ω)

ω is the hour angle which is the angle that the earth has rotated during the day. Therefore, the earth completes one rotation of 360° in 24 hours (Harkness & Mehta, 1978). The angle increases by 15° per hour and is defined to be 0° at noon. For example, the hour angle at 1 p.m. is 15°. The negative sign of the hour angle represents the morning, and positive sign is in the afternoon. It is calculated as follows (MoDule, 2004):

$$\omega = (\frac{360}{24})(T-12) \tag{B.3}$$

where: T is the current local 24-hour time

For example, the hour angle at 10 A.M. (-2 hours from noon) is

ω = (360/24) * (10-12) = -30°.

As a further example, the hour angle at 2 P.M. (2 hours after noon) is

ω = (360/24)*(14-12) = 30°.

- **Determining the altitude angle of the sun (ε)**

ε is the solar altitude angle of the sun, which is the vertical angle between the horizontal plane and the sun. At sunset/sunrise the altitude is 0 and, at the equator, the altitude angle is 90 degrees when the sun is at its zenith. The altitude relates to the latitude of the site, the declination angle and the hour angle. The altitude angle is calculated as follows:

$$\sin\varepsilon = \cos\gamma\cos\delta\cos\omega + \sin\gamma\sin\delta \qquad \text{(B.4)}$$

where:

γ is the geographic latitude angle at the particular location. The positive or negative latitude angle value is based on the northern or southern hemisphere, respectively. For example, Christchurch, New Zealand has the latitude angle of -41° (Hong & Yu, 2005), while the latitude angle of New York City is 41° (Cheung, 1997).

Reading the position of the sun from sunpath diagrams

A sunpath diagram can be presented in several types of projection such as orthographic, stereographic, equidistant, cylindrical, and waldram projections (Szokolay, 1996). However, the orthographic diagram is the most popular for architects (Harkness & Mehta, 1978).

The position of the sun at a particular time, date and latitude can be determined as shown in the example below based on an orthographic diagram (Autodesk, 2007).

The orthographic sunpath diagram (Figure B.2) is a 2D graph of the sun's position. The azimuth angle is plotted along the horizontal axis whilst the altitude angle is plotted vertically. The sun position can be read simply by reading the two axes.

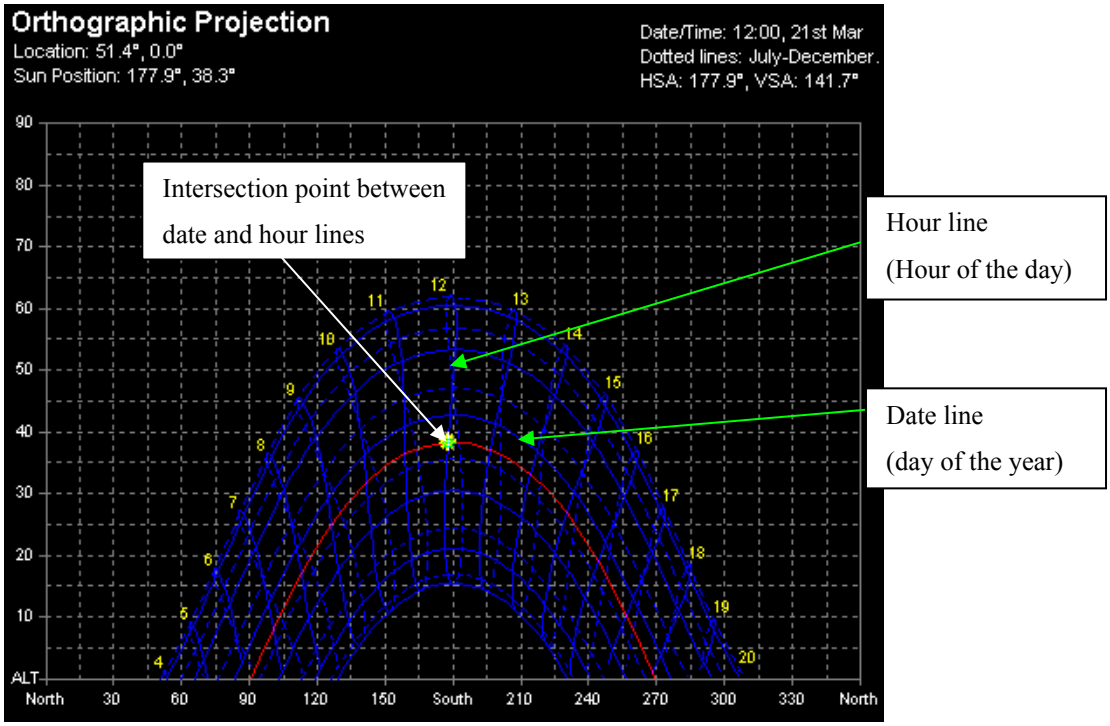


Figure B.2: The sunpath diagram at London, UK (adapted from CLEAR, 2004b).

To determine the position of the sun at 12 noon on 13th March carry out the following steps using Figure B.2:

1. Determine the hour line which represents the specific hour of the day, at which we are interested in the sun's position. In this case, the hour line is at 12 pm.

2. Determine the date line which represents day of the year of interest. In the diagram, the highest dotted date line is December, the lowest date line is July. The highest solid date

line is January, the lowest date line is June. Each line represents the 1st of the month. The highlighted date line represents 21st March.

3. Determine the intersection point between date and hour lines.

4. Read the azimuth from the horizontal axis. In this case, the value is approximately 178°.

5. Read the altitude from the vertical axis. In this case the intersection point is 38°.

Appendix C

Intersection of a Ray from the Sun with the Image

This section shows how to calculate the point where a ray from the sun intersects the image (either sky or obstruction) in terms of both azimuth and altitude angles.

Calculating the azimuth angle where a ray from the sun intersects the image

Figure C.1 shows a plan view of the relationship between the view plane normal of the camera and the azimuth angle of the sun.

In Figure C.1 β is the angle of the sun azimuth, calculated using equation (B.1) in Appendix B.

α is the angle of view of the camera in the horizontal direction. For more details, see Appendix A.

θ is the angle of the camera in the horizontal direction. This angle is between the view plane normal of the camera and south direction.

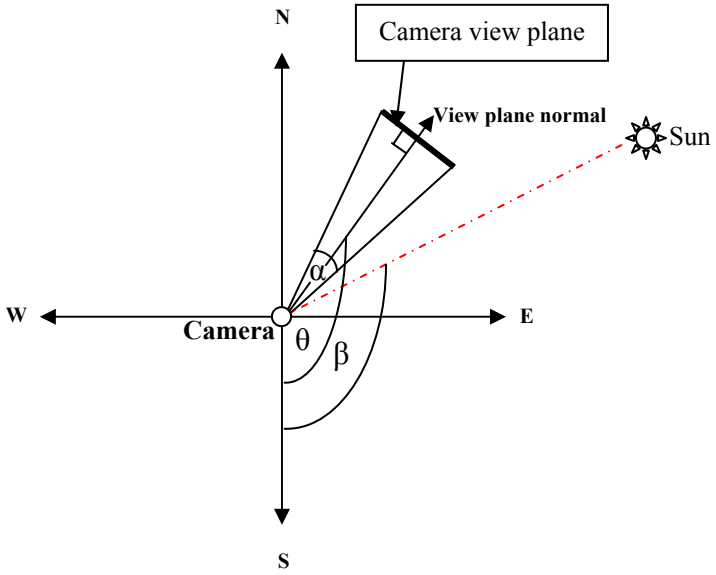


Figure C.1: Position of the sun relative to the view plane in a horizontal plane (plan view).

Figure C.1 shows the ray from the sun intersects the image when,

$$\theta - (\frac{\alpha}{2}) \leq \beta \leq \theta + (\frac{\alpha}{2}) \qquad \text{(C.1)}$$

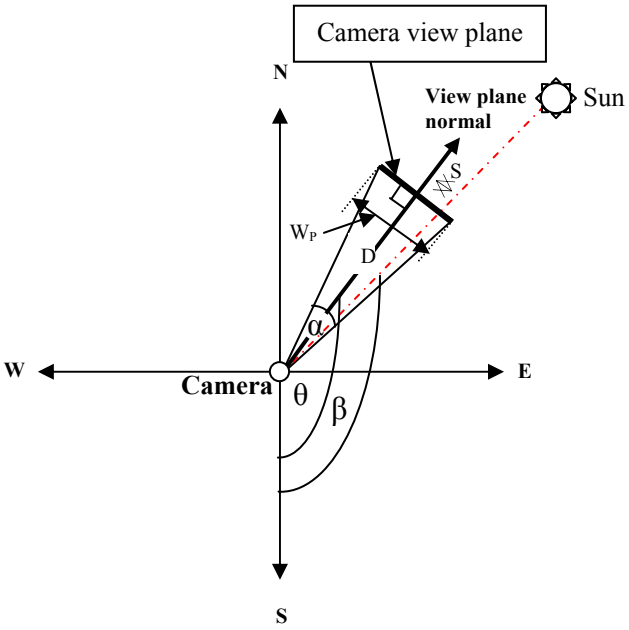


Figure C.2: The sun ray intersecting the image in a horizontal plane (plan view).

Figure C.2 shows where the sun ray intersects the image in a horizontal plane. If S is the horizontal distance (number of pixels) from the centre of the view plane to the ray from the sun and D is the distance between the camera and the view plane then,

$$\tan(\theta - \beta) = \frac{S}{D} \tag{C.2}$$

hence

$$S = D\tan(\theta - \beta) \tag{C.3}$$

The distance between the camera and the view plane (D)

If W_P is the actual number of pixels in the width of the image then,

$$\tan(\frac{\alpha}{2}) = \frac{W_P}{2D} \tag{C.4}$$

$$D = \frac{W_P}{2\tan(\frac{\alpha}{2})} \tag{C.5}$$

Therefore the distance (number of pixels) from the centre of the view plane to the ray from the sun (S) is

$$S = \frac{W_P \tan(\theta - \beta)}{2\tan(\frac{\alpha}{2})} \tag{C.6}$$

If S is a negative value, the position of the sun will be on the other side of the centre of the view plane.

Calculating the altitude where a ray from the sun intersects the image

Figure C.3 shows a view of the relationship between the view plane normal of the camera and the altitude angle of the sun. In Figure C.3,

ε is the solar altitude angle of the sun, calculated using equation (B.4) in Appendix B.

μ is the angle of the camera in the vertical direction. This angle is between the horizontal plane of the camera and view plane normal. For more details, see Appendix A.

x is the view angle of the camera in the vertical direction.

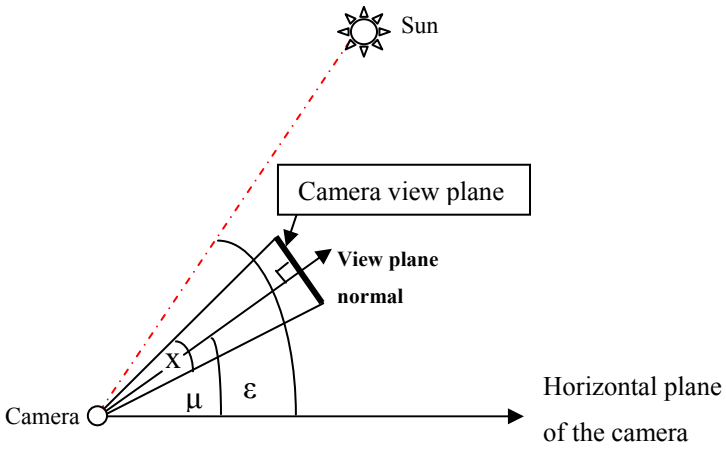


Figure C.3: Position of the sun relative to the view plane in a vertical plane (elevation facing west).

Figure C.3 shows the ray from the sun intersects the image when,

$$\mu - (\frac{x}{2}) \le \varepsilon \le \mu + (\frac{x}{2}) \tag{C.7}$$

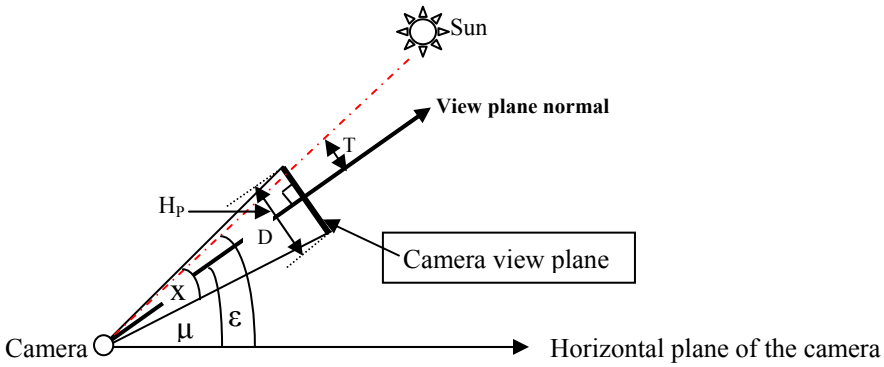


Figure C.4: The sun ray intersecting the image in a vertical plane (elevation facing west).

Figure C.4 shows the position of the sun ray intersects the image. If T is the vertical distance (number of pixels) from the centre of the view plane to the ray from the sun and D is the distance between the camera and the view plane then,

$$\tan(\varepsilon - \mu) = \frac{T}{D} \tag{C.8}$$

hence

$$T = D\tan(\varepsilon - \mu) \tag{C.9}$$

<u>The distance between the camera and the view plane (D)</u>

If H_P is the actual height of the image in pixels image then,

$$\tan(\frac{x}{2}) = \frac{H_P}{2D} \tag{C.10}$$

$$D = \frac{H_P}{2\tan(\frac{x}{2})} \tag{C.11}$$

Therefore the distance (number of pixels) from the centre of the view plane to the ray from the sun (T) is

$$T = \frac{H_P \tan(\varepsilon - \mu)}{2\tan(\frac{x}{2})} \tag{C.12}$$

If T has a negative value, the position of the sun will be under the view plane.

Appendix D

OpenCV Libraries (Indiana University, 2004)

The list below is the OpenCV libraries that were used to identify the sky regions in an image in this thesis

IplImage* cvLoadImage (const char* filename, int colour=1)

- Load an image from file.

  filename is name of file to be loaded.

  colour represents a colour of the image as follows:

  If >0, the image is displayed as a RGB colour channel image.

  If 0, the image is displayed as a greyscale colour channel image.

  If < 0, the image is displayed as a number of channels depends on the file.

IplImage* cvCreateImage (CvSize size, int depth, int channels)

- Create the image.

  size is the width and height of the image.

  depth is the bit depth of the image elements. This can be one of:

  IPL_DEPTH_8U - unsigned 8-bit integers.
  IPL_DEPTH_8S - signed 8-bit integers.
  IPL_DEPTH_16U - unsigned 16-bit integers.
  IPL_DEPTH_16S - signed 16-bit integers.
  IPL_DEPTH_32S - signed 32-bit integers.
  IPL_DEPTH_32F - single precision floating-point numbers.
  IPL_DEPTH_64F - double precision floating-point numbers.

channels is the number of channels in the image. For example, grayscale images will have a channel value of 1, while full colour will have 3 channels (1 each for red, green and blue).

void cvCvtColor (const CvArr* source, CvArr* destination, int space_code)

- Convert a colour image from the BGR colour space to another.

  source is the source 8, 16 or 24 bits colour.

  destination is the destination 8, 16 or 24 bits colour.

  space_code is a colour conversion operation. It can be CV_BGR2GRAY, CV_BGR2HSV, CV_BGR2YCrCb, CV_BGR2XYZ, CV_BGR2Lab when a colour image is converted to greyscale, HSV, YCrCb, XYZ, and Lab colour space, respectively.

void cvSplit (const CvArr* source, CvArr* destination1, CvArr* destination2, CvArr* destination3)

- Divide multiple colour channels into separate signal channels.

  destination1 is the first destination 8 bits colour.

  destination2 is the second destination 8 bits colour.

  destionation3 is the third destination 8 bits colour.

The cvSplit function can be used to split the colour image to red, green, and blue without calling the cvCvtColor to get the BGR space colour first.

void cvCanny (const CvArr* source, CvArr* destination, double upperThreshold, double lowerThreshold, int aperture_size)

- Implement Canny algorithm for edge detection.

  upperThreshold is the upper threshold value.

lowerThreshold is the lower threshold value.

aperture_size is Size of the extended Sobel kernel, must be 1, 3, 5 or 7. The default value is 3.

int cvCreateTrackbar (const char* scoreBarName, const char* windowName, int* value, int count, CvTrackbarCallBack on_change)

- Create a score bars or track bar and attach it to the specified window.

  scoreBarName is name of the created trackbar.

  windowName is name of the window which will be used as a parent for created trackbar.

  value is a pointer to the integer variable, where value is proportional to the position of the slider. Upon creation the slider position is defined by the variable.

  count is maximal position of the slider. Minimal position is always 0.

  on_change is a pointer to the function to be called every time the slider changes the position. This function should be prototyped as void Foo(int); Can be NULL if callback is not required.

void cvDilate (const CvArr* source, CvArr* destination, IplConvKernel* element = NULL, int iterations = 1)

- Dilate binary image by using arbitrary structuring element.

  element is structuring element used for dilation. If it is NULL, a3*3 rectangular structuring element is used.

  iterations is number of times dilation is applied.

void cvErosion (const CvArr* source, CvArr* destination, IplConvKernel* element = NULL, int iterations = 1)

- Erode binary image by using arbitrary structuring element.

iterations is number of times erosion is applied.

IplConvKernel* cvCreateStructuringElementEx (int columns, int rows, int anchor_x, int anchor_y, int shape, int* values=NULL)

- Create the structure element size.

  columns is number of columns in the structure element.

  rows is number of rows in the structure element.

  anchor_x is relative horizontal offset of the anchor point.

  anchor_y is relative vertical offset of the anchor point.

  shape is shape of the structure element, may have the flowing values:

    CV_SHAPE_RECT is a rectangular element.

    CV_SHAPE_CROSS is a cross-shaped element.

    CV_SHAPE_ELLIPSE is an elliptic element.

  value is pointer to the structuring element data, a plane array, representing row-by-row scanning of the element matrix. Non-zero values indicate points that belong to the element. If the pointer is NULL, then all values are considered non-zero, that is, the element is of a rectangular shape.

void cvFloodFill (CvArr* image, CvPoint seed_point, CvScalar new_val, CvScalar lo_diff=cvScalarAll(0), CvScalar up_diff=cvScalarAll(0), CvConnectedComp* comp=NULL, int flags CV_DEFAULT (4), CvArr* mask = NULL)

- Fill a connected component with given colour.

  image is input 1 or 3 channel, 8 bit or floating point image. It is modified by the function unless CV_FLOODFILL_MASK_ONLY flag is set.

  seed_point is the starting point.

  new_val is new value of repainted area pixels.

lo_diff is maximal lower brightness difference between the currently repainted area pixel and one of its neighbours.

up_diff is maximal upper brightness difference between the currently repainted area pixel and one of its neighbours.

comp is pointer to connected component structure of the repainted area. To count the pixels (area) of the region that have been filled, we use int(comp.area).

flags is if the function looks for 4-connected neighbours, otherwise it looks for 8-connected neighbours.

mask is operation mask, should be single-channel 8-bit image, 2 pixels wider and 2 pixels taller than image. If not NULL, the function uses and updates the mask, so user takes responsibility for initializing mask content. FloodFilling cannot go across non-zero pixels in the mask. For example, an edge detector output can be used as a mask to stop filling at edges. Also, it is possible to use the same mask in multiple calls to the function to make sure the filled area does not overlap. Note: because the mask is larger than the filled image, a pixel in the mask that corresponds to (x, y) pixel in image will have coordinates (x+1, y+1).

void cvSetMouseCallback (const char* window_name, CvMouseCallback on_mouse)

- Assign callback for mouse events.

  on_mouse is pointer to the function to be called every time the mouse event occurs in the specified window. This function should be prototyped as

  void mouseFcn(int event, int x, int y, int flags).

  event is one of CV_EVENT_*.

  x and y are coordinates of mouse pointer in image coordinates (not window coordinates).

  flags is combination of CV_EVENT_FLAG.

int cvNamedWindow (const char* name, int flags)

- Create window.

  name is the name of the window which is used as the window identifier and appears in the window caption.

  flag is flags for the window. Currently the only supported flag is CV_WINDOW_AUTOSIZE. If it is set, window size is automatically adjusted to fit the displayed image, while user cannot change the window size manually.

void cvShowImage (const char* name, const CvArr* image)

- Show the image in the specified window.

  image is the image to be shown.

int cvCountNonZero (const CvArr* source)

- Count all the non- zero elements in the image.

  source is the source 8 bits colour.

CvHistogram* cvCreateHist (int dimensions, int* sizes, int type, float** ranges=NULL, int uniform = 1)

- Creates histogram.

  dimensions is the number of dimensions for the histogram.

  sizes is array of histogram dimension sizes.

  type is the type of histogram. Can be either: CV_HIST_ARRAY or CV_HIST_SPARSE.

  ranges is array of ranges for histogram values. Default value is NULL.

  uniform is uniformity flag; if not 0, the histogram is evenly spaced.

void cvCalcHist (IplImage* source, CvHistogram* hist, int accumulate=0, const CvArr* mask=NULL)

- Calculate histogram of image.

  hist is pointer to the histogram.

  accumulate is accumulation flag. If it is set, the histogram is not cleared in the beginning. This feature allows user to compute a single histogram from several images, or to update the histogram online.

  mask is the operation mask, determines what pixels of the source images are counted.

int cvRound (double value)

- Converts floating-point number to integer.

  value is the input floating-point value.

void cvGetMinMaxHistValue (const CvHistogram* hist, float* min_value, float* max_value, int* min_idx = NULL, int* max_idx = NULL)

- Finds minimum and maximum histogram bins.

  hist is histogram.

  min_value is a pointer to the minimum value of the histogram.

  max_value is a pointer to the maximum value of the histogram.

  min_idx is a pointer to the array of coordinates for minimum.

  max_idx is a pointer to the array of coordinates for maximum.

void cvMinMaxLoc (const CvArr* source, double* minVal, double* maxVal, CvPoint* minLoc, CvPoint* maxLoc, const CvArr* mask=0)

- Calculate minimum and maximum in image.

  minVal is pointer to returned minimum value.

maxVal is pointer to returned maximum value.

maxLoc is pointer to returned maximum location.

mask is the option mask that is used to select a subarray.

CvScalar cvAvg (const CvArr* source, const CvArr* mask=0)

- Calculate average of an image.

CvScalar cvGet2D (const CvArr* arr, int idx0, int idx1)

- Return the particular array element.

  arr is input array.

  idx0 is the first zero-based component of the element index.

  idx1 is the second zero-based component of the element index.

void cvSet2D (const CvArr* arr, int idx0, int idx1, Cvscalar value)

- Change the particular array element.

  value is the assigned value.

void cvAdd (const CvArr* source1, CvArr* source2, CvArr* destination)

- Computes per-element sum of two arrays.

  source1 is the first source array.

  source2 is the second source array.

void cvSub (const CvArr* source1, CvArr* source2, CvArr* destination)

- Computes per-element difference between two arrays.

void cvSaveImage (const char* filename, const CvArr* image)

- Saves an image to a file on the hard disk.

  filename is the filename that the saved image will have. For example, "image01.bmp" (with the quotes) is acceptable.

  image is the image that will be saved to a file.

void cvCloneImage (const IplImage* source)

- Makes a full copy of image.

void cvLine (CvArr* source, CvPoint point1, CvPoint point2, CvScalar colour, int thickness, int line_type, int shift)

- Draws a line segment connecting two points.

  point1 is first point of the line segment.

  point2 is second point of the line segment.

  colour is line colour.

  thickness is line thickness. The default value is 1

  line_type is type of the line.

    8 (or 0) - 8-connected line.
    4 - 4-connected line.
    CV_AA - antialiased line.

  shift is number of fractional bits in the point coordinates. The default value is 0.

void cvRectangle (CvArr* img, CvPoint point1, CvPoint point2, CvScalar colour, int thickness, int line_type, int shift)

- Draws simple, thick or filled rectangle.

Appendix E

Program Instructions

The image processing programs that are used to identify the sky regions are separated into three programs. The first program allows the user to identify sky regions manually. The second program is to find the area and perimeter of the sky region for each of the test images in order to determine appropriate upper and lower thresholds and the structure element size. The third program is to identify sky regions automatically by using the appropriate value of upper and lower thresholds and the structure element size, and the brightness and area criteria.

Figure E.1 shows the overview of the program for identifying sky regions. The arrow connection between processes shows the direction of the process flow. When the program is activated, the sky detection system loads an image and stores it as the current image. The next process is to extract a colour channel. After that, one of three options can be selected. The first option is to calculate the minimum, maximum or average of the pixels. The second option is to display a histogram of the image. The last option is to use the image processing to identify sky regions. Each process for identifying sky regions requires the user to specify the relevant parameters. After finishing the image processing techniques to identify the sky region, the area and perimeter of the sky region can be calculated. User interaction to modify the regions identified as sky can also be carried out.

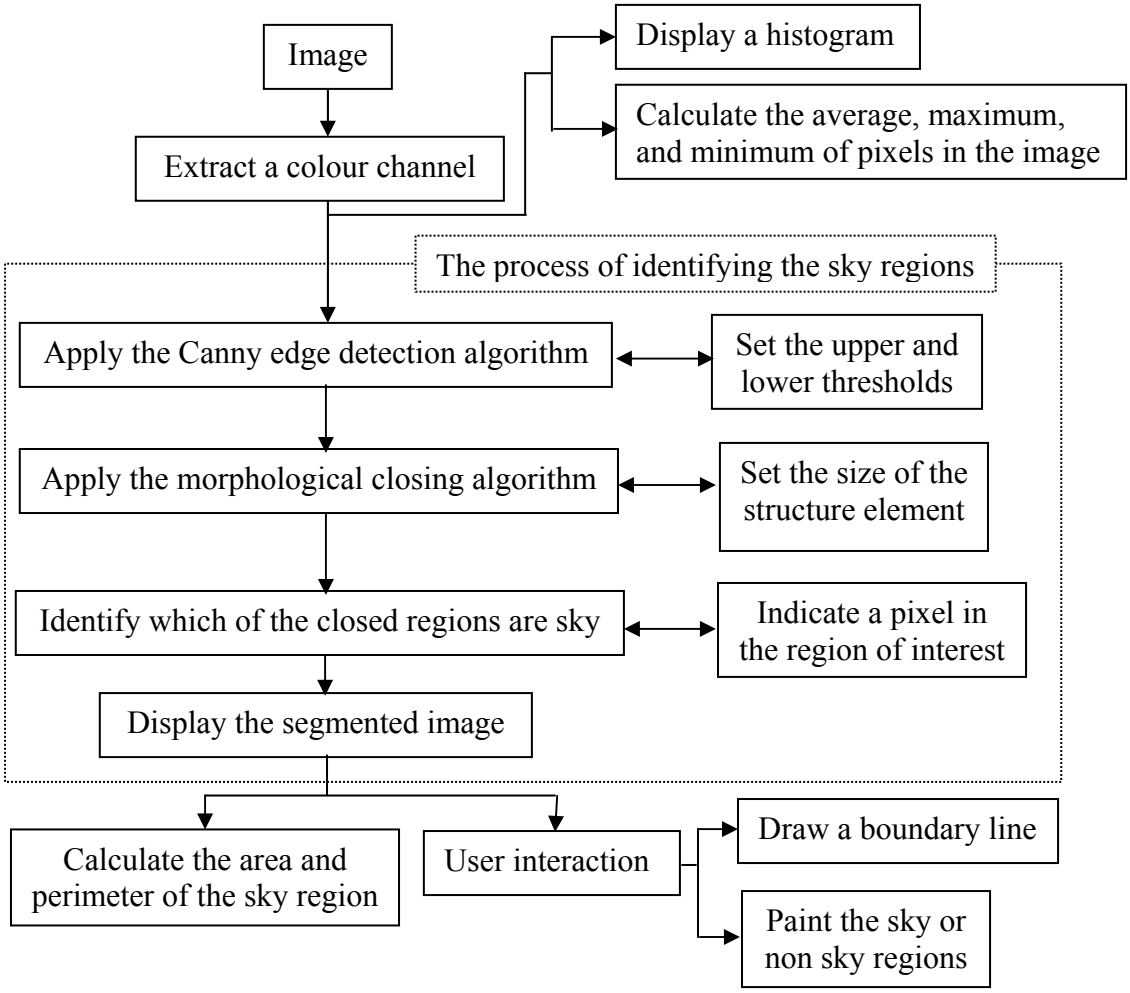


Figure E.1: The overview of the sky identification program.

The program for identifying sky regions manually

The purpose of this program is to identify sky regions in an image manually.

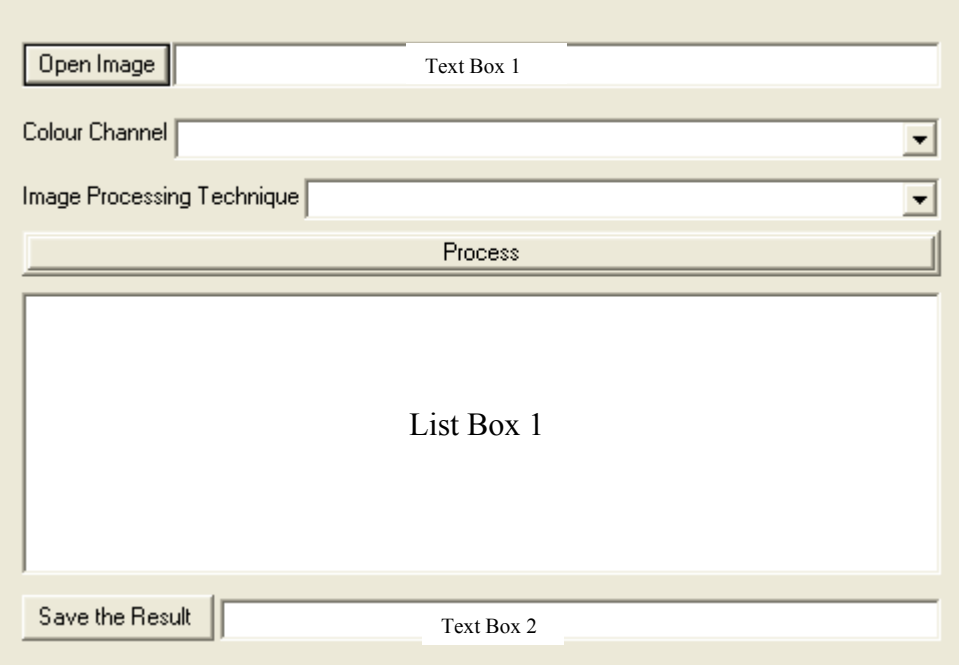


Figure E.2: The user interface for identifying sky region manually.

Instructions

1. Press “Open Image” button and browse for an image file to display a colour image. The source of an image name, such as D:\Thesis\Image1, also shows in the text box 1.
2. Select a colour channel that the user would like to extract from the drop down lists.
3. Select the image processing technique, such as Canny edge detector, Morphology closing algorithm or FloodFill algorithm, in the drop down lists.
4. Press the “Process” button to carry out the image processing. The display window showing the result of the particular image processing technique then displays. The details of the image process also displays in the List Box 1, as shown an example in Figure E.3

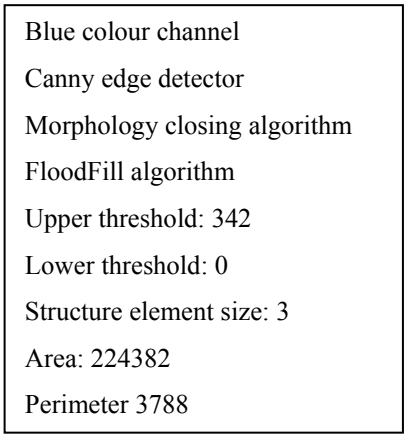


Figure E.3: An example of the details of the image processing in the List Box.

5. Repeat step 3 and 4 for each image processing technique.
6. The result can be saved in the save file by inserting the file name and pressing "Save" button. The file is stored in d:/data which is automatically created by program.

The program for determining the appropriate upper and lower threshold values, and structure element size

The purpose of this program is to determine the range of upper threshold, lower threshold and the structure element size that perform the satisfy result image of separating the sky.

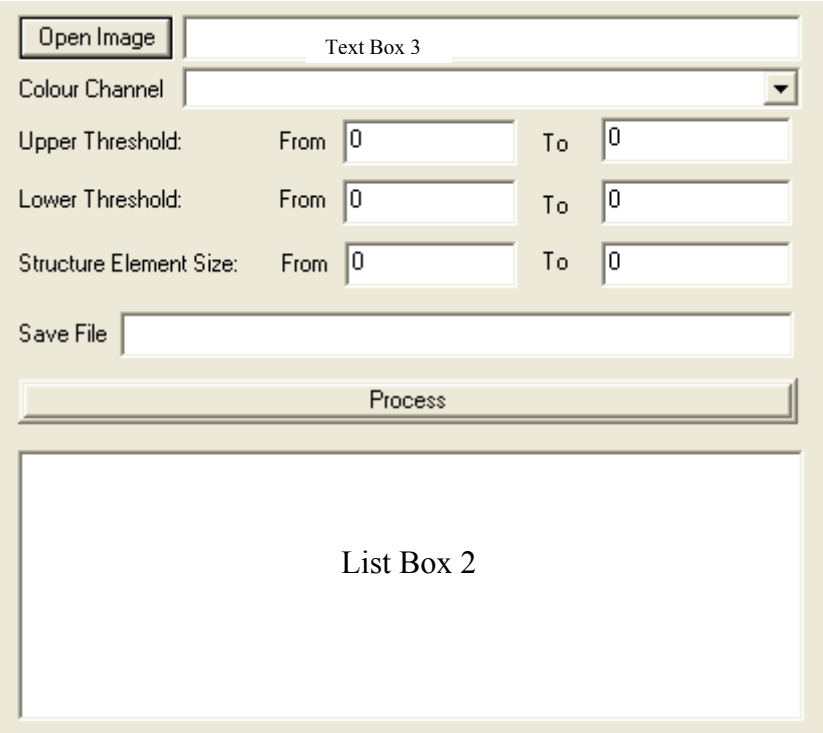


Figure E.4: The user interface of the program for determining area and perimeter of the region with in the range of the upper and lower thresholds and the structure element size.

Instructions

1. Press on the 'Open image' button to open an image. The image will be displayed as a colour image. The source of an image name also shows in the text box 3.
2. Select a colour channel in the drop down lists to perform greyscale, blue, red and green colour channel.
3. Assign the range of upper threshold, lower threshold, and the structure element values to the text boxes.
4. Assign the save file name to the text box.
5. Press the "Process" button to process the image processing.
6. The file is stored in the folder d:/data which is automatically created by the program. The format of the file is a text file. Each record of the file contains the image number, the area and perimeter of the sky region, and the upper and lower threshold values and the structure element size. The data is also shown in the List Box 2, as shown in Figure E.5.

Upper threshold: 200
Lower Threshold: 20
Structure element size: 3
Area:179810
Perimeter: 3865

Figure E.5: An example of the information stored in each record.

The programming for identifying sky regions automatically and user interaction

The purpose of this program is to determine the sky region automatically by using the appropriate upper threshold, lower threshold, and the structure element size, and the brightness and area criteria. The program is also interactive allowing the user to make changes in situations where image processing does not correctly determine the sky regions.

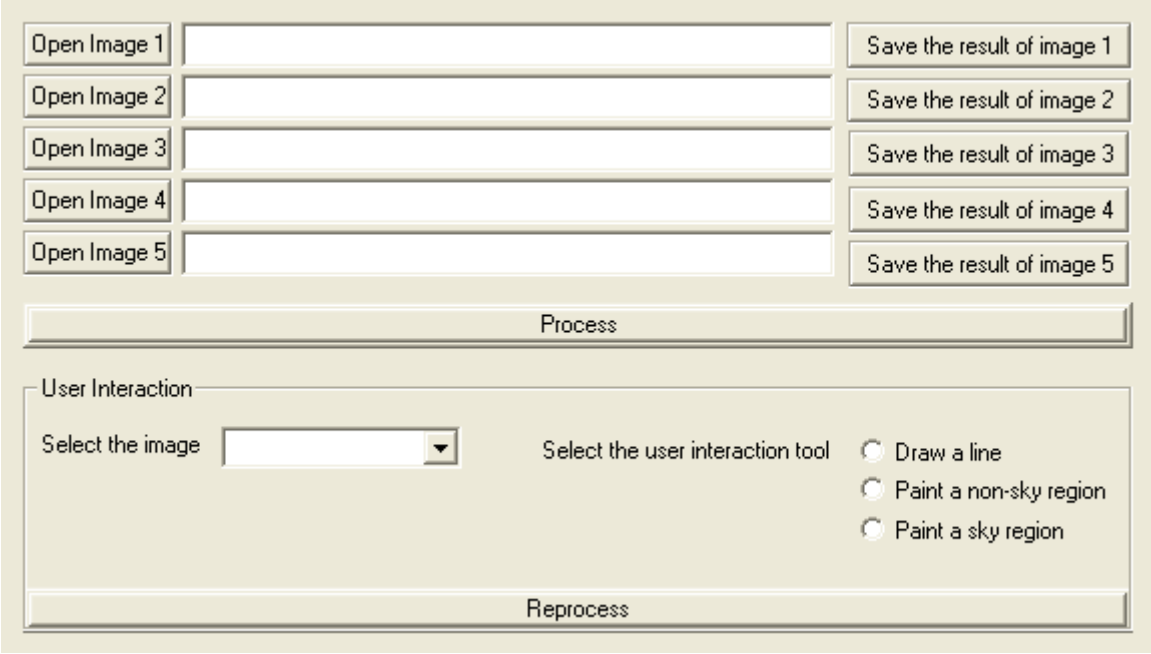


Figure E.6: The user interface for identifying sky regions automatically and allowing user interaction.

Instructions

1. Press at least one button of ‘Open Image 1 to 5’ to select the image that the user wants to process. The source of an image name also shows in the text box.
2. Press the “Process” button. The result will be then displayed automatically.
3. Press “Save the result of image 1 to 5”, if the user would like to save the image result.
4. If the image processing fails to identify the sky regions, user interaction is required on a particular image by selecting the image from the drop down lists in the “Select the image”.
5. Select the interaction tools using the radio buttons.
6. Press “Reprocess” button if the user selects a line drawing tool. If the user selects a painting tool, the program will automatically reprocess the image and display the result.