Image-Based Relighting

by

Jingyuan Huang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis proposes a method for changing the lighting in some types of images. The method requires only a single input image, either a studio photograph or a synthetic image, consisting of several simple objects placed on a uniformly coloured background. Based on 2D information (contours, shadows, specular areas) extracted from the input image, the method reconstructs a 3D model of the original lighting and and $2\frac{1}{2}$D models of objects in the image. It then modifies the appearance of shading and shadows to achieve relighting. It can produce visually satisfactory results without a full 3D description of the scene geometry, and requires minimal user assistance.

While developing this method, the importance of different cues for understanding 3D geometry, such as contours or shadows, were considered. Constraints like symmetry that help determine surface shapes were also explored. The method has potential application in improving the appearance of existing photographs. It can also be used in image compositing to achieve consistent lighting.

Acknowledgements

First of all, I would like to thank Bill Cowan, my supervisor, for his guidance and support. As a pretty typical Type I student, who wants to maximize the value of qualifications earned while minimizing time, I am grateful that my somewhat stubborn and hasty nature did not stop him from teaching me the right attitude of scientific research.

I want to thank Justin Wan for taking me as his student at the beginning of my study, and letting me go free to pursue my own research topic.

I'd also like to thank Stephen Mann, who gave me valuable suggestions when I was unsure of what was lying ahead. Plus, who can say no to the delicious ice cream on Tuesday afternoons?

Many thanks to Elodie Fourquet for the encouragement and inspiration she gave me. I will miss our conversations during those late nights in the lab.

Also thanks to all other members in CGL. It was a pleasure to work with them.

Last but not least, I owe the deepest gratitude to my mom, who brought me up and taught me how to be a good person. I hope I haven't disappointed her so far.

to Mom

Contents

Appendices

List of Tables

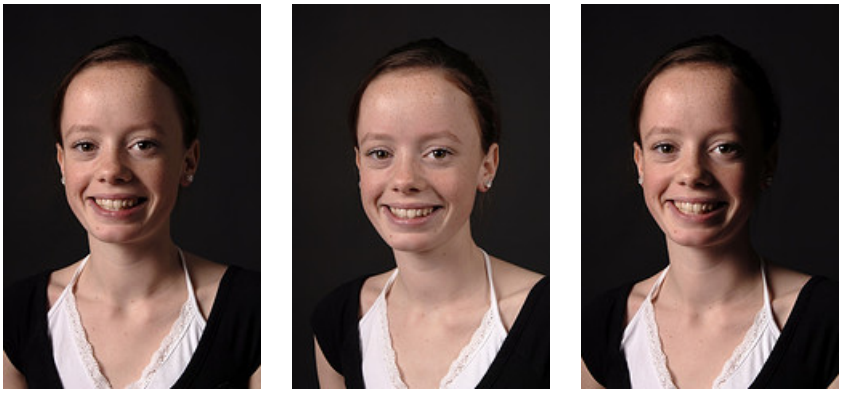
List of Figures

Chapter 1

Introduction

A photograph records a 2D image of the illuminated world. The word "photograph", which was coined in 1839 by Sir John Herschel, is based on the Greek word $\varphi\tilde{\omega}\varsigma$ ($ph\bar{o}s$), meaning light, and $\gamma\rho\alpha\varphi\omega$ ($graph\bar{o}$), referring to the act of writing or drawing pictures [60]. Taking these definitions together, a photograph is a "drawing with light". Photography is indeed drawing with light: its essence is light falling on a light-sensitive surface, usually photographic film or a charge-coupled detector.

Photography was once a complicated procedure: photographers were burdened with heavy equipment and complex chemical post-processing. With the emergence of digital imaging, however, photography is now an inescapable part of everyday life, appearing in advertisements, scientific research, forensic studies, to name only a few of many applications. But the ease and ubiquity of photography means that many photographs fall short of perfection. Making a good photograph entails exposing film or a digital sensor to just the right amount and distribution of light. Good lighting is the essence of good photography. When photographs are composed and captured in studios, professional photographers carefully control the lighting, adjusting the direction, intensity, and colour of the lights. Moreover, different lighting configurations, usually comprising two or more lights, help add emotion to photographs. In the simple example shown in Figure 1.1, a fill light brings out the facial details, or a deflector darkens the shadow to add depth and feeling. Amateur photographers not possessing studio equipment or unaware of the importance of light rarely obtain comparably subtle lighting effects in their photographs.



(a) 45° Key Light (b) Add 45° Fill Light (c) Add Deflector

Figure 1.1: Studio Lighting Examples [39]

Such oversights can sometimes be corrected by reposing and relighting the subjects, then recapturing the image. However, this is not always feasible, especially if the photograph was taken in the distant past or at a remote location. Another solution is to use a computer-based method to rectify such faults in the image. In computer graphics, algorithms create photorealistic images by modeling the transport of physical light that occurs when a picture is taken. They operate on a scene with a full specification of object geometry, surface reflectance, light sources, and camera. Given complete information, relighting a scene using rendering techniques is as simple as repositioning the lights and rendering the scene again. However, such a relighting procedure does not apply to the problem of relighting an existing photograph. Even if the input image is in digital format, it gives only a confounded description of 3D geometry, surface properties, and light sources. Relighting an isolated 2D image requires a method that is based on the 2D input rather than assumed 3D representations.

A common 2D solution to the problem takes advantage of the ability of the human visual system to assess lighting and 3D geometry from a 2D image: trained artists change the appearance of the image using digital editing software such as Photoshop or GIMP. However, changing illumination and the corresponding shading and shadow effects is not as straightforward as it may appear. Usually, the modification process requires much layering, colour matching, and shadow modification, demanding great skill and patience [1]. Useful tools, such as the shadow/highlight tool of Photoshop and the graduated filter

[1]With Photoshop or GIMP, one can mimic the relighting effects by following tutorial like

tool of Lightroom, for example, bring out the details in underexposed areas or darken overexposed ones. The result is similar to adding a fill light or a deflector. However these tools cannot be used to add secondary light sources or to generate shadows because their algorithms are based on individual pixel values.

Problems like these ones motivated the work presented in this thesis. Relighting a single image is a difficult task because the image provides no explicit 3D structure. Because every individual image can be derived from any one of many 3D scenes, there is no single solution to all relighting problems. Therefore, the method described here restricts its inputs to images consisting of several simple objects placed on a uniformly coloured background. With this restriction, we hope to find the key 2D elements for creating perceptually acceptable images and to provide insights into the relighting problem. The method is able to modify lighting and produce visually satisfactory results for simple inputs with minimal user interaction (Figure 1.2). It can be extended or combined with other tools to accomplish more complex image editing tasks.

(a) The Input Image (b) The Relit Image

Figure 1.2: Sample Results of the New Method

This thesis examines the following questions.

- Is it possible to relight a 2D image without additional input?

http://www.3datadesign.com/gallery/eng/tutorial_3dlight.php

- How much 2D information can be extracted from the input image? What is the minimal 3D information required for perceptually adequate image relighting?
- What is the most difficult aspect of relighting?

The first question is addressed in Chapter 2, where a positive conclusion is reached by examining research in image-based computer vision and graphics that influenced the formulation of the method. Chapter 2 also describes assumptions required by the method. Chapter 3 summarizes the current state of research in relevant areas. Chapter 4 then provides the details of the algorithm, addressing the second question above: how much 2D and 3D information from the inputs can be extracted and how it is done? Chapter 5 shows some results and examines the drawbacks of the method, concluding that the most difficult aspect of relighting is building precise representations of the object surfaces. Finally, the last chapter summarizes the research and suggests possible future work.

Chapter 2

Background and Assumptions

When the detailed geometry of a 3D scene is available, relighting is easily done by changing the lights and re-rendering the scene using traditional rendering techniques. Relighting a 2D image, however, is much more difficult because geometry and lighting are intermingled in the image. The main data structure used in such a relighting process is a 2D image, so this chapter walks through the existing studies in image-based areas to discover what has been successfully extracted from 2D images, thereby addressing the question: is relighting a single 2D image is possible?

The chapter also describes how an answer to this question provides the basic outline of the method, and defines a list of assumptions used in later implementation. Based on this formulation of the problem and its solution, Chapter 3 then reviews the relevant research that has influenced the implementation described in Chapter 4.

2.1 Image-Based Methods

Researchers have been motivated to work directly with 2D images because traditional computer graphics rendering from full 3D scene descriptions comes with a cost: the more sophisticated the rendering model, the more expensive the computation. True photorealism also makes high demands on image creators. 3D modeling and texture tuning require a high level of skill and a great deal of effort to attain a realistic appearance. Image-based methods can resolve both shortcomings, offering inexpensive and realistic rendering results.

There are three main kinds of image-based methods in contemporary computer graphics: rendering, lighting, and modelling. By definition image-based rendering is the one most closely related to relighting 2D images. Many methods in this area aim to create new renderings using novel camera views or modified illumination, an idea similar to the goal of this thesis. Therefore research in image-based rendering is reviewed first in Section 2.1.1. Image-based lighting and modelling, on the other hand, try to separate the confounded scene elements to compute the unknown light and/or object geometry. Research in these areas provides many useful techniques for extracting light and geometry; it is examined in Section 2.1.2 and 2.1.3.

2.1.1 Image-Based Rendering

Image-based rendering uses a set of input images to achieve new renderings from different viewpoints or with modified illumination. Existing image-based rendering techniques are generally classified into three categories: no geometry, implicit geometry, and explicit geometry [58].

At one extreme is rendering without geometry. Its algorithms ignore scene geometry, obtaining their results, such as extracting a plenoptic function[1], directly from the input images. A complete encoding of light transport is captured by the plenoptic function. Lights with known plenoptic functions can be added to or subtracted from the scene with ease, and camera views can be freely updated using light field interpolation. The biggest drawback of using the plenoptic function, however, is that its construction requires many images, usually acquired by a calibrated camera array, a device that is not generally available.

Because reconstructing the plenoptic function is often impossible, other methods that require estimation of geometry have been considered. An implicit geometry of the scene can be inferred from the input images by identifying positional correspondences in different images, the key idea in a variety of view-interpolation methods. With enough input images, a dense optical flow can be used to reconstruct arbitrary viewpoints [6]. Alternatively, linear combinations of a sparse set of 2D views allow construction of novel views [67]. In these methods, the explicit 3D representation of the scene is replaced by images and an implicit

[1]The plenoptic function is the 5D function representing the intensity or chromacity of the light observed from every position and direction in 3D space. It can be simplified to a 4D function called the light field [29] or the lumigraph [17].

3D geometry is encoded in the mappings among them. Compared to the construction of the plenoptic function, fewer input images are required, but accurate image registration or additional data is needed for high-quality image synthesis.

A third category of methods is based on explicit 3D information for each pixel in the form of depth or 3D coordinates, which is most often obtained from external sources such as a 3D scanner data or stereoscopic depth reconstruction. These methods, which take advantage of explicit 3D geometry, build a full 3D description of the scene, ignoring illumination. For example, given a set of depth inputs from a 3D scanner, warping techniques [40, 36] can create images from most nearby viewpoints, limited only by occlusion. Improvements such as Layered Depth Image method (LDI) [56] better solve the occluded surface problem. The drawback of these methods are similar to those encountered in the previous two categories of methods. Either complex and expensive equipment such as 3D scanners, or multiple spatially-corresponding, well-registered images are required.

For general purpose photography it is rare to have multiple photographs or additional data like optic flow or scanned object depths. Thus the methods mentioned above can not be used for relighting. However, they contain some important ideas. First, the intractability of approximating the plenoptic function demonstrates the extreme difficulty of retrieving a full description of the illumination that was present when an image was captured. To circumvent such difficulty, assumptions must be made to constrain light abstraction. Second, methods using implicit geometry suggest that relighting can be achieved without a complete 3D representation of the scene. Finally, explicit geometry shows that pixel depth is a partial but effective representation of the scene geometry.

Studies of image-based rendering have demonstrated that the relighting problem is more likely to be solved if approached from two separate directions, illumination and geometry. Therefore the next two sections examine research in image-based lighting and modeling, looking for ideas in these areas.

2.1.2 Image-Based Lighting

Image-based lighting is a technique of using an image, often a high dynamic range (HDR) image, as the light source for rendering. Its direct use lies in traditional 3D rendering, but other applications, such as rendering synthetic objects into real scenes [9], have extended

its domain. Such work, known as inverse rendering, estimates the object reflectance and scene illumination given the rendering results of the scene.

An important aspect of inverse rendering is recovering the surface properties of objects in an input image. A common assumption of such research is that at least the scene geometry is known [34, 50], and sometimes both geometry and lighting [33, 4, 31, 70]. In the problem domain of this thesis, neither lighting nor geometry is given initially, and estimating surface reflectance is impossible. A workaround might be replacing reflectance estimation with assumptions. The other aspect of inverse rendering, inverse lighting, is more promising.

Inverse lighting tries to estimate the lighting of input images. Much research in this area assumes known geometry. Some of it extracts the original lighting using geometry and a basis of lights [38, 55]. Other research focuses on special features of object surfaces, such as the critical points of a sphere, to compute the light directions [72, 65]. There is also research on recovering lighting without geometry. A sequence of work originating in Pentland [46] focuses on light estimation for extending shape from shading (SFS) algorithms [20, 73]. These studies recover the initial light directions using a single input with no knowledge of geometry. They usually assume a simple lighting model and Lambertian surfaces. Doing so these methods require the least input information of all inverse lighting methods and reinforce the expectation that the original lighting can be recovered from a single image, if reasonable assumptions can be made for the illumination model and surface reflectance.

2.1.3 Image-Based Modeling

Image-based modeling is a set of techniques for creating a 3D representation of a scene from 2D images. Many image-based modeling methods use motion cues [13] from a stream of inputs to make 3D scenes for traffic surveillance and robotics, or stereoscopic cues [18] to create multi-view reconstructions. There is also abundant research in modeling using monocular depth cues such as foreshortening, occlusion, and shadows [12] to build 3D models. Using the convergence of parallel lines and foreshortening, collectively known as perspective, algorithms can rebuild 3D structures or create $2\frac{1}{2}$D depth maps from a single input image [21, 7], especially when the image mainly consists of indoor main-made structures [22]. In the absence of strong perspective cues, neural network [53, 19] may be used to estimate depth. Aiming to generate a depth map for all types of input images,

most neural network methods fail to deliver good surface detail. On the other hand, with shading available as a depth cue, traditional SFS [20] methods can determine pixel depths of smooth curved surfaces from a single input images. These methods prove that, if the lighting of the image is known, a partial form of the scene geometry, either represented in a 3D or $2\frac{1}{2}$D format, can be recovered from the input image. Therefore with known lighting, it is possible to recover object shapes from a single image.

2.2 Formation and Assumptions

The conclusions from research in image-based rendering are the following:

- Assumptions of simple lighting must be made. A point light or a directional light is a useful approximate of the initial lighting conditions.
- Rendering can be achieved without a complete 3D representation of the scene. A depth map or normal map is a good candidate for such a partial representation.

Deeper examination of studies in image-based lighting and modeling gives the further conclusions:

- It is possible to recover the original lighting from a single image without other prior information, provided that the image satisfies prerequisites listed later in this section.
- It is also possible to recover surface shapes from a single image after the lighting has been estimated if the image satisfies the assumption listed later in this section.

If both lighting and geometry are available, relighting can be done. Therefore, relighting a single 2D image is possible, but only when constrained to a small subset of input images. The relighting problem is thus undertaken in from two stages: light recovery and shape recovery. Lighting is estimated first; the results make it possible to evaluate and recover shapes in the input image and to store shape information as a normal map. The two stages are combined to achieve relighting.

It is necessary to impose constraints to ensure the success of the method. The inputs images are studio photographs or synthetic images similar to studio photographs. A typical

studio setting usually consists of a uniformly coloured surface as a ground plane and several simple objects as the main subject. Adding in the constraints assumed in previous research, the following assumptions are then used throughout the research of this thesis:

- There is a single input image.
- The initial lighting is provided by one light source.
- The ground plane is uniformly coloured.
- The ground plane occupies all the space in an image except the objects of interest.
- All objects are placed on the ground plane.
- Objects in the image are uniformly coloured as well. The object surface reflectance can be well approximated by the Phong shading model.
- The objects and their shadows are within the image boundaries.
- The objects do not occlude one another.

Chapter 3

Related Work

The relighting problem is decomposed into two subproblems: light recovery and shape recovery. Both have been extensively investigated in computer graphics and computer vision. Rather than list all the heterogeneous research in both areas, Section 3.1 and 3.2 review the most relevant research in each aspect satisfying the assumptions imposed in previous chapter or providing particular insights used in this research.

3.1 Light Recovery

As described in Chapter 2, the method first recovers the lighting in the input image before undertaking geometry estimation. Light recovery is also known as light source estimation or detection. Table 3.1 shows a simple taxonomy of typical research that falls into two categories. Papers in the first estimate lighting in the absence of scene geometry: they are the forerunners of the method developed here. Papers in the second category use shadows and specular highlights for light recovery. Theses methods, because they assume geometry to be known, cannot be used as is in the present work, but they prove that shadows can be an important additional cue.

Table 3.1: Light Recovery Methods Comparison

	Unknown Geometry	Shadow & Specular Cues	Detect Multiple Light Sources
Pentland	√		
Zheng & Chellappa	√		
Yang & Yuille	√		√
Poulin & Fournier		√	
Sato & Ikeuchi		√	
Li et. al.		√	√

3.1.1 Unknown Geometry

Pentland [46] estimates the illumination direction from the derivative of image intensity, which depends on the variation of surface normals relative to the illumination direction. Given a distribution of intensity values of a convex object, he provides a maximum-likelihood method to deduce the illumination direction causing such intensity changes. Pentland's paper also includes a study showing that human subjects can estimate the light direction with less than 6° of inconsistency. This result provides an accuracy threshold for our method. There is no need for light recovery method to outperform human beings. For relighting purposes, a method generating errors below this threshold is acceptable.

Following Pentland's idea, Zheng and Chellappa [73] also use the image intensity change with respect to the light direction. More specifically, they examine occluding boundaries, which provide normal values to locate the xy direction of light sources, and propose a contour-based method for light recovery. Their approach assumes uniform albedo and Lambertian surfaces, starting with the estimation of light direction and moving to shape reconstruction, an idea similar to our work. Thus it asserts the feasibility of using lighting to discover shape and to reconstruct the scene geometry.

Zheng and Chellappa's light detection method improved upon Pentland's approach, and inspired Yang and Yuille [69]. Yang and Yuille's method also uses occluding boundaries, but it is capable of estimating two unknown light sources, assuming that the lights are from sufficiently different directions. The z directions of the light sources are recovered by comparing the brightest shading value on the Lambertian surface to the brightest value on

the boundary. Yang and Yuille's method is selected as the base method for light recovery in our work and is modified to accommodate only one light source as specified in the assumptions.

3.1.2 Multiple Cues

Poulin and Fournier [48] look at visual cues other than shading for understanding the illumination in a scene. They acknowledge work that uses highlights and surface normals to detect light direction, and notice importance of shadows for indicating the direction and shape of light sources. Using 3D models of objects, they calculate the convex hulls of shadow volumes. New light positions are calculated when users drag shadows to new locations. Sato and Ikeuchi [52] analyze the relationships between objects and and shadow brightness to estimate the illumination distribution in a scene. Li et al. [32] further extend light estimation to images with textured surfaces by integrating different cues including shading, shadows, and specular reflection. They use the scene geometry to minimize differences in shading, highlights, and shadow appearance given a light source with the actual image.

From the above work one learns the effectiveness of shadows and specular highlights for determining light direction. However, since the specular highlights are useful for light recovery only when the surface normals are known, our method can only make use of shadows. Rather than adopt any of the above methods directly, which are inappropriate since they are designed to work with known geometry, our algorithm uses the geometry of the object and its shadows as additional correctness check during illumination estimation.

3.2 Shape Recovery

Shape recovery is also called image-based reconstruction or modeling, as described in Section 2.1.3. Much research in shape recovery requires multiple input images. The number of required images varies, from a dense set of inputs from all angles in space carving approaches, to two or three images with spatial correspondence for stereoscopic reconstructions. However, for general purpose photography, we cannot expect more than one

input image. Thus, methods requiring multiple inputs are not reviewed because they break our first assumption.

Table 3.2 shows a simple taxonomy of some research that uses a single input image. Common techniques used in such research include: user interaction, perspective, and shape from shading. The reviewed work is classified by the techniques it uses. Each category is discussed in details in the following sections. Note that methods aided by user specification to discover the perspective construction are reviewed under perspective (Section 3.2.2) rather than under user interaction (Section 3.2.1).

Table 3.2: Shape Recovery Methods Comparison

	User Interaction	Perspective	Shading
Oh et al.	√		
Okabe et al.	√		
Horry et al.	√	√	
Criminisi et al.	√	√	
Boulanger et al.		√	
Huang & Cowan		√	
Zheng & Chellappa			√
Danial & Durou			√
Pentland			√
Lee & Rosefeld			√
Tsai & Shah			√

3.2.1 User Interaction

Oh et al. [44] create an interactive system to build a collection of layered depth images for the input, an idea similar to LDI in image-based rendering. They also provide a variety of tools that assist users to manipulate depth and textures. Their method can create high quality models, but the segmentation and depth specification take from several hours up to a couple of days to complete, as revealed in Chen's thesis [5]. While a user's ability at recognizing objects and shapes are far superior to that of computers, such tedious user

interaction is avoided as much as possible in our method, which aims to give users quick feedback and achieve relighting with a short response time.

Inspired by Oh et al., Okabe et al. [45] develop a method to paint normals. With a pen-tablet input device, pen orientation marks the normal direction pixel by pixel. The marked normals are then propagated to the entire image for relighting. The propagation method allows users to mark only a subset of pixels in the image, requiring less user effort than Oh et al.'s method. However, many pixels must be hand marked if the propagation is to work correctly. Without such an input device, normal marking does not meet the assumptions. But Okabe et al. do show that steps such as separating specular highlights and shadows and refining normals based on chromaticity are useful.

3.2.2 Perspective

User interaction was also a big component of early work in 3D reconstruction from perspective. Horry et al. [21] use the concepts in projective geometry to build a coarse 3D structure. In their method users specify one vanishing point and an inner rectangle so that the program can compute the perspective projection. Users also separate foreground objects from background ones, each of which are represented by simple 2D billboards, to build a coarse model of the scene by placing these billboards at their corresponding 3D depths. Our constraints on the ground plane and objects in the scene enable automatic separation of the background and the foreground objects. The idea of using 2D billboards placed at the right depths is promising since it is a partial yet effective way to represent the scene geometry. This idea is later used in our method for geometry representation.

In his classic work Criminisi et al. [7] recovered the camera position and depth without camera calibration, by having users specify anchor points and lines. Based on ratios of lengths of parallel lines and ratios of areas of parallel planes, the camera location can be calculated and the perspective projection recovered. More recently automatic methods have been developed to use perspective for 3D reconstruction. Boulanger et al. [27] automate Horry et al.'s method. They find the vanishing points by classifying straight lines into groups using the Hough transform. Huang and Cowan [22] use similar techniques but focus on indoor scenes consisting of orthogonal planes. Both methods are able to recover 3D models without user intervention by limiting the geometric shapes present in the image. Because 3D reconstruction from a single 2D image is impossible without prior knowledge,

imposing such constraints is one important way to realize automation. A similar approach based on symmetry is developed in this thesis to build accurate surface shapes.

Perspective works well for images with architectural structures or shapes with straight lines and orthogonal planes. Indoor studio photographs, whose subjects are usually objects with curved surfaces, are not good candidates for reconstruction using perspective. Other methods, not depending on the perspective cues, must be found.

3.2.3 Shape from Shading

Another technique of shape recovery relies on shading. Shape from shading is a class of research that uses shading for recovering the shape of surfaces. It works well for input images containing simple objects with Lambertian surfaces, and its results, depth values of each pixels, are the usual target of shape recovery. Therefore, work in SFS offers many potential solutions for shape recovery. SFS methods are classified based on the approaches they take [71, 11]. The three main classes are: global optimization, local approximation, and linear methods.

There are many examples of solving the SFS problem as a global optimization problem. Zheng and Chellappa improve the method of Frankot and Chellappa [15] and formulate a constrained optimization problem guaranteeing that the solution is an integral surface. Danial and Durou [8] propose a gradient iterative scheme, demonstrating that their method can work with both synthetic and real images.

Pentland noted the importance of local analysis of the image [20]. His method and Lee and Rosefeld's work [28], as well as many other local SFS approaches, compute the normal at each point independently of other points. Most local methods make an assumption that the surface is locally spherical. When this assumption breaks, the results are far from their true values. Due to this limitation, local methods receive less attention than the global ones.

Linear methods are a combination of global optimization and local approximation. These methods make assumptions about the reflectance function, rather than the surface shape. Pentland [47] develop a linear approximations of the true reflectance function, which provides an efficient closed-form solution for the surface shape. Tsai and Shah [59] believe that linearizing the reflectance map in depth is more appropriate and devised a corresponding linear solution.

It is difficult to decide which class of methods gives the best results under the assumptions of the thesis. A quick glance suggests that both global and linear methods are possible candidates. In Section 4.2.2 several SFS methods are tested and their results are compared. The best performer is then modified to suit the problem at hand.

Chapter 4

Proposed Algorithm

The algorithm separates the problem into two discrete aspects: light recovery and shape recovery. Figure 4.1 shows the overall work flow, which has three stages: pre-processing, processing, and relighting. During pre-processing basic 2D features are extracted from the input image, such as object contour, shadow, and specular highlights. These features are used to recover lighting and shape in the processing stage, where lighting is estimated and a $2\frac{1}{2}$D normal map is built to represent the scene surfaces. To achieve relighting, these aspects of the scene are merged to construct images with different illumination.

The details of each stage are discussed in sections 4.1, 4.2, and 4.3. Section 4.4 summarizes the important elements in the algorithm and answers the second question we have: how much 2D information can be extracted from the input? What is the minimal 3D information for a perceptually adequate relit image?

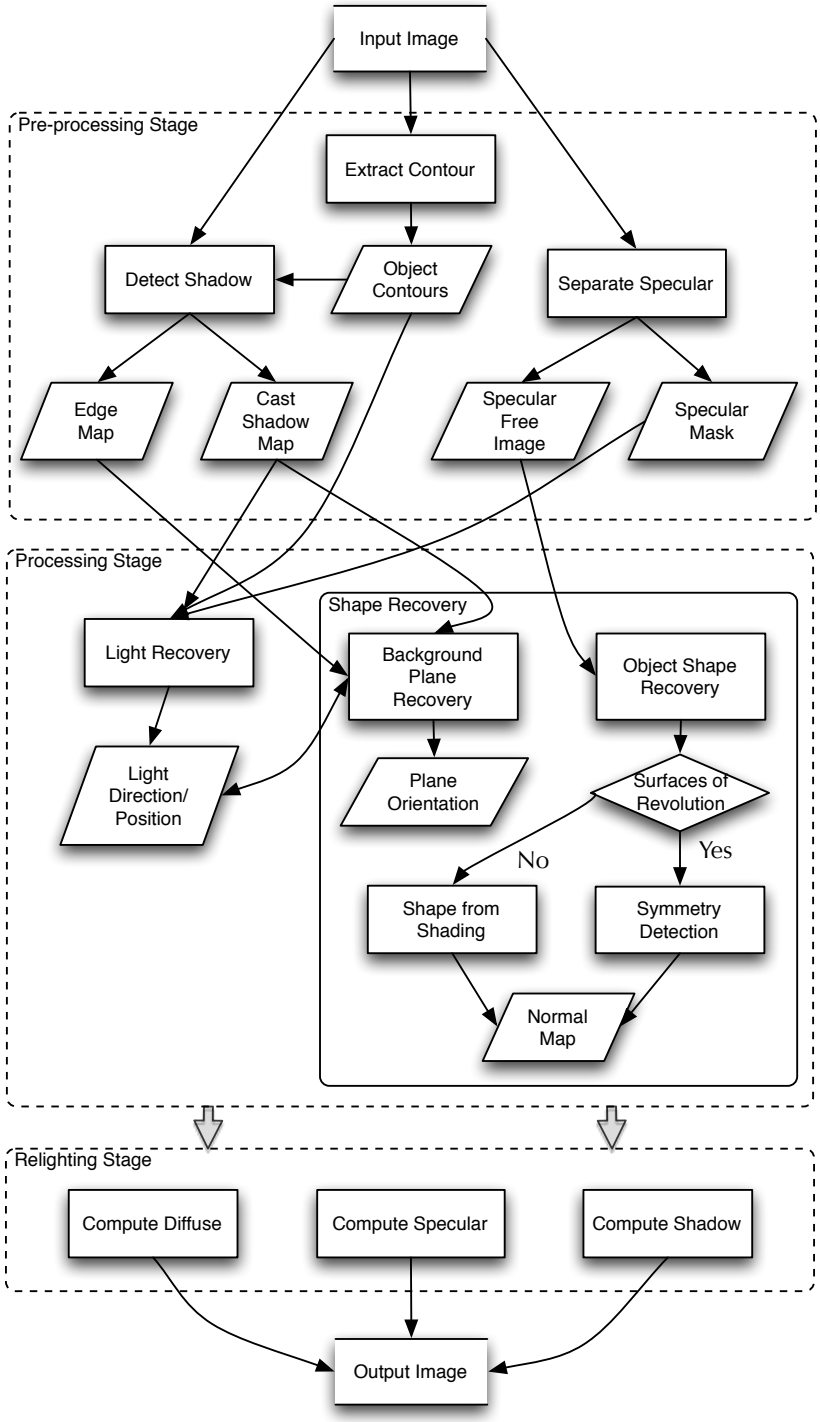


Figure 4.1: Shape Recovery Work Flow

4.1 Pre-processing Stage

The pre-processing stage extracts 2D features including object contours, cast shadows, and specular highlights from the input image. Human vision uses such 2D features to identify the shape and position of objects. Using staged processing [66], David Marr identifies the first stage of vision as creation of a 2D primal sketch, followed by a $2\frac{1}{2}$D sketch and a full 3D shape representation [37]. The new method forms a primal sketch – edges and contours in the image – to get the basic structure of the scene, then proceeds to a $2\frac{1}{2}$D representation in later stages. In addition to the primal sketch, pre-processing extracts shadow and specular areas because they also provide useful cues for light and shape recovery [32, 52].

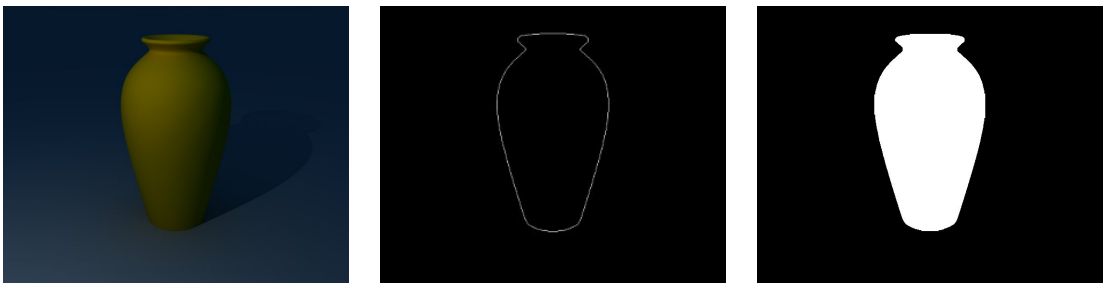
4.1.1 Object Contour Extraction

The first step in pre-processing extracts object contours from the input. Four conditions create edges in an image: discontinuities in depth, illumination effects like shadows and highlights, discontinuities in surface orientation, and changes in surface reflectance [37]. The contour extraction algorithm detects only edges caused by discontinuities in depth, since they mark the positions and boundaries of objects. To obtain them it is necessary to eliminate edges caused by other factors.

First, edges caused by illumination effects are eliminated by using an illumination-invariant colour space, which is a colour space transformation that is robust against illumination changes. The most effective colour spaces are ones designed to be perceptually uniform, like $CIELuv$ or $CIELab$ because they discount illumination by normalizing with the white point. Commonly used colour spaces for this purpose includes normalized rgb, the H component of HSV, $l_1l_2l_3$ [16], and $m_1m_2m_3$ [16]. As suggested by Gevers et al., two colour models, $l_1l_2l_3$ and H, are best for object recognition when there are highlights in the image [16]. H, together with $CIELuv$ and $CIELab$, which were examined in Khan and Reinhard's study [25], were picked as the candidates and tested on a set of images. Both $CIELuv$ and $CIELab$ outperformed H in most cases, with the results from $CIELab$ and $CIELuv$ being similar. $CIELab$ is used for rejecting illumination edges because Khan and Reinhard suggests that overall it generates more consistent results [25] than $CIELuv$.

Edges caused by changes in surface orientation or reflectance must also be removed. Fortunately, both types of edges are located within object boundaries. The outermost

contour of each object is obtained by running Canny edge detection on the image in the illumination-invariant colour space. Flood-fill then removes all edges within the object. These operations give us the object contours (silhouettes) and the object mask (Figure 4.2).



(a) Original Image (b) Object Contour (c) Object Mask

Figure 4.2: Object Contour Results

4.1.2 Shadow Detection

Artists consider shadows to be an important aspect of realistic images. Leonardo da Vinci made the first thorough analysis of shadows in 1490, describing how artists can use light and shade to evoke the perception of three-dimensional relief in paintings [3]. There are two types of shadows: self shadows and cast shadows. Self shadows occur when an object occludes itself; cast shadows occur when an object is occluded by another object (Figure 4.3).

Knill et al. studied shadow geometry and its implications [26]. Their conclusions are disappointing: shadows are only minimally useful for inferring shape. Mamassian et al. [35] also show that cast shadow provide few surface shape cues. On the other hand, cast shadows strongly help the perception of light source direction [32], object depths [26], and 3D shape of the object that shadow is cast on [43]. Therefore, shadow detection serves two main functions in the method: cast shadows are used for recovering light direction and for reconstructing the geometry of the ground plane on which shadows are cast.

Shadow classification is largely based on the results of Salvador et al. [51]. Their method both detects shadows in static images and classifies them into either self shadows or cast shadows. It uses an illumination-invariant colour space to obtain object contours, as in the

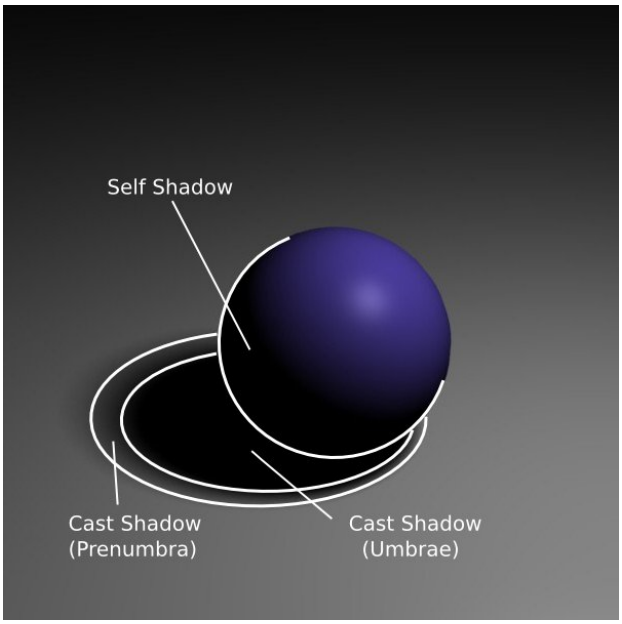


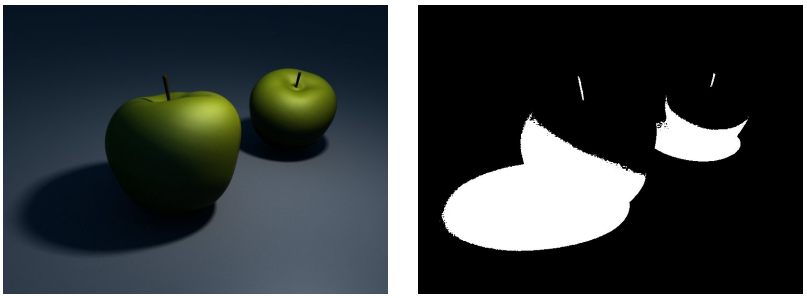
Figure 4.3: Shadow Geometry

previous step (Section 4.1.1). The remaining work builds shadow candidates and separates self shadows from cast shadows. Salvador et al. use Sobel edge detection to obtain their edge map. They scan the edge map horizontally and vertically marking points as shadow candidates. This method works well for cast shadows with sharp edges, but fails when shadow boundaries are blurry. The failure is averted by the earlier technique of Jiang and Ward [23], which requires a threshold parameter to compute the shadow candidates. It scans the image in the x direction for each line. A line scanned at $y = y_0$ $I(x, y_0)$ is an intensity function for $0 \leqslant x < n_x$. A linear intensity function that interpolates between the pixels at the ends of the scanned line is defined as

$$I_L(x, y_0) = \frac{(I_b - I_a)}{n_x - 1} x + I_a,$$

where $I_a = I(0, y_0)$ and $I_b = I(n_x - 1, y_0)$. If the intensity of a pixel on the scanned line is lower than the threshold $I_t(x) = I_L(x, y_0) - \Delta$, then the pixel is classified as a shadow candidate. It is not easy to determine a value for Δ that gives good results for all inputs. Testing this algorithm on a dozen images provided an empirical value of $0.05 I_{max}$ as the default value for Δ, where I_{max} is the maximum pixel intensity in the image. Users can, if necessary, adjust this value to improve the shadow map.

Figure 4.4 shows the results of running this algorithm on a synthetic image. The shadow candidates are first marked using Jiang and Ward's method. It is then separated to cast shadow map and self shadow map using the object mask generated in contour detection step.



(a) Original Image (b) Shadow Candidates

(c) Cast Shadow Map (d) Self Shadow Map

Figure 4.4: Shadow Classification Results

4.1.3 Specular Separation

The object surface reflectance is assumed to follow the Phong shading model, thus the surface material contains three components: ambient, diffuse, and specular. The ambient value is small compared to the other two and is roughly constant over the entire scene, so it is treated as input noise. It is necessary to separate only the other two components, specular and diffuse, since most shape recovery methods assume diffuse surfaces [11, 71].

Much research in computer vision has explored highlight removal. Nayar et al. [42] used a polarization filter to estimate the colour of the specular highlight. Tan and Ikeuchi establish the Specular Free (SF) image [61] and successfully remove highlights from some

images with textures [63]. The SF image was later extended to the Modified Specular Free (MSF) image by Shen et al. [57].

Specular separation in this thesis is based on Tan et al. [62] and on Shen et al. [57]. The input image is first normalized using the method of Tan et al. [62] to ensure that the specular colour is white. It is then processed as suggested by Shen et al. [57] to generate the corresponding SF image:

$$V_{sf,i}(p) = V_i(p) - \min(V_i(p)) \tag{4.1}$$

$$= V_i(p) - V_{min}(p), \tag{4.2}$$

where $V_i(p)$ is the intensity value of each pixel p in channel i where $i = (R, G, B)$. Contrary to the practice of Shen et al., it is unnecessary to generate an SF image for the entire input. Instead, a separate SF image is computed for each object using the object masks. In Shen et al., a threshold value is added to the SF images to stabilize noise and form the MSF images:

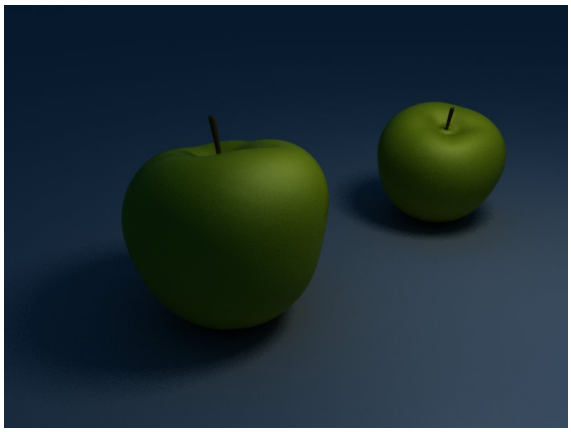
$$V_{msf,i}(p) = V_{sf,i}(p) + \bar{V}_{min},$$

where

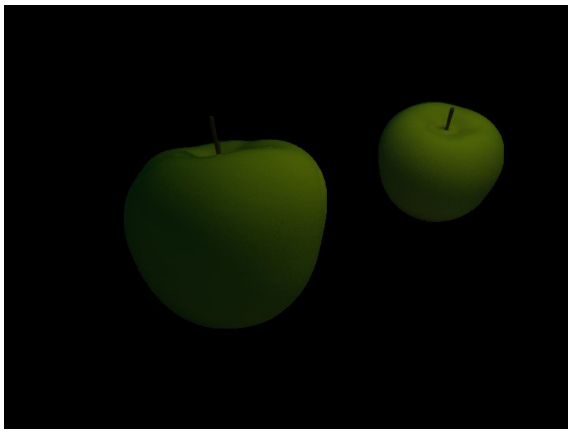
$$\bar{V}_{min} = \frac{\sum_p V_{min}(p)}{\#pixels}.$$

The present method improves on Shen et al. by iteratively locating a threshold value that minimizes the difference between the original image and the MSF image $\sum_{i,p} |V_i(p) - V_{msf,i}(p)|$. This produces the MSF image that most closely matches the original image, but with the specular component removed. After the MSF image for each object is computed, a specular mask is generated by thresholding the difference between the original image and the MSF image. Figure 4.5 shows the result of running this algorithm on a synthetic image.

Our experiments using the specular removal algorithms of Tan et al. [61] and Shen et al. [57] showed that the resulting diffuse-only images do not always contain smoothly varying pixel values in specular areas. Sometimes, uniform patches of colour remains and worsen relighting results in later stages. Therefore such methods are inappropriate for generating a diffuse-only image; using the MSF image described above gives better results.



(a) Original Image



(b) MSF Image

(c) Specular Mask

Figure 4.5: Specular Separation Results

4.2 Processing Stage

The processing stage converts the 2D primal sketch to a $2\frac{1}{2}$D sketch representation of the image. The 2D structures retrieved in the previous stage are used to estimate lighting and shape. The light direction and/or position is evaluated first, then the object surface shapes are approximated and represented as normal maps.

4.2.1 Light Recovery

Initial Estimation of Light Direction

Initial estimation of light direction uses the diffuse component of the image to compute light direction. Specular pixels are excluded from consideration by the specular mask so the remaining image consists only of diffuse pixels. Since each object is uniformly coloured, the surface exhibits Lambertian reflectance:

$$I = k_a + \sum_{m \in lights} k_d \max(0, \hat{n} \cdot \hat{l}_m).$$

As stated in Section 2.2, the input image is assumed to contain only one light source, either a point light or a directional light. Then the above equation simplifies to

$$I = k_a + k_d \max(0, \hat{n} \cdot \hat{l}), \tag{4.3}$$

where $\hat{n}$ is the surface normal, $\hat{l}$ is the light direction, k_d is the diffuse coefficient, and k_a is the ambient factor.

Equation (4.3) is used to find the light direction $\hat{l}$. Yang and Yuille noted that under orthogonal projection all normals at the silhouette are unchanged when projected to the image plane [69]. However, the assumed photographic input images are captured under perspective projection, and surface normals along the object contours change when projected to the image plane. Figure 4.6 shows the unit vector $\hat{n}$ and the angle of view γ. The y component of $\hat{n}$ is projected onto the image plane $n_y = \cos\alpha$ and it is the value that can be measured. The γ value for a standard 35mm camera is around 40°. Suppose the image is well-composed and the subject is placed close to the center of the image, α is expected to be smaller than 10°, leading to a value larger than 0.98 for $\cos\alpha$. Therefore the projection of $\hat{n}$ is close enough to its actual value and the error can be safely replaced by a small constant ϵ for a rough estimation.

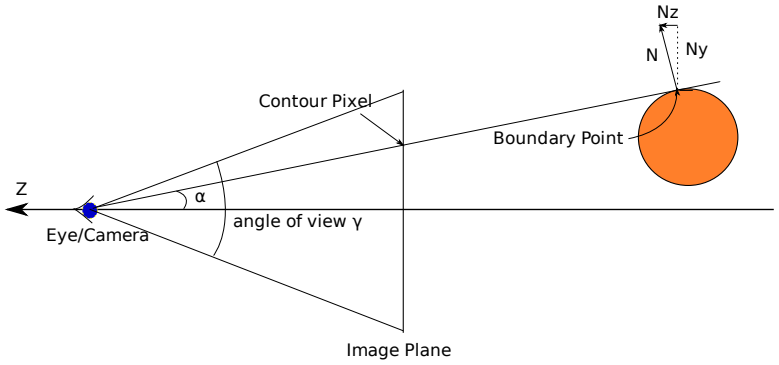


Figure 4.6: Normals of Boundary Points under Perspective Projection

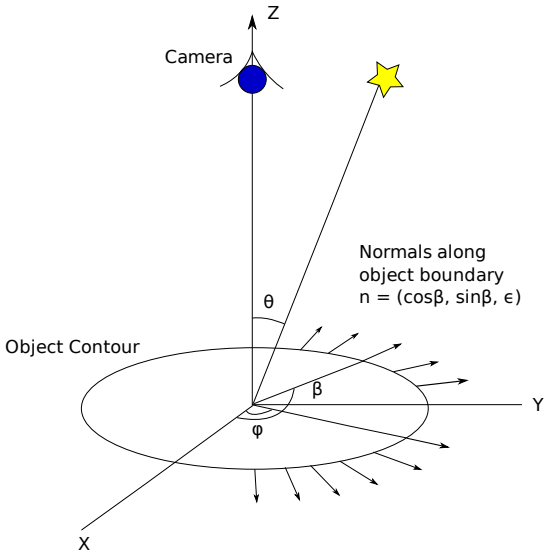


Figure 4.7: Various Normal Values for Light Estimation

A normal at the silhouette then can be expressed as $\hat{n} = (x, y, \epsilon)$. Equation (4.3) is then written as

$$I = k_a + k_d \max(0, n_x l_x + n_y l_y + \epsilon l_z). \tag{4.4}$$

Both $\hat{l}$ and $\hat{n}$ are unit vectors: $\hat{l} = (\cos\phi \sin\theta, \sin\phi \sin\theta, \cos\theta)$ and $\hat{n} = (\cos\beta, \sin\beta, \epsilon)$; equation (4.4) is then

$$I = k_a + k_d \max(0, \sin\theta(\cos\phi \cos\beta + \sin\phi \sin\beta) + \epsilon \cos\theta),$$

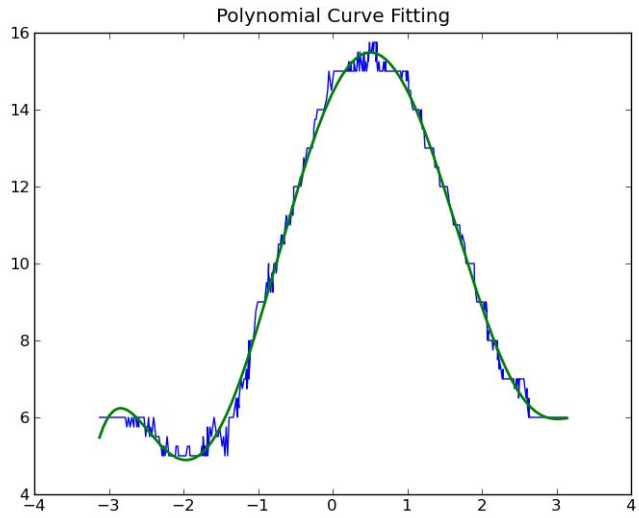
$$I = k_a + k_d \sin\theta \max(0, \cos\phi \cos\beta + \sin\phi \sin\beta + \epsilon \cot\theta). \tag{4.5}$$

Since ϵ is small, $\epsilon \cot\theta$ is set to a small constant value δ. Equation 4.5 then becomes

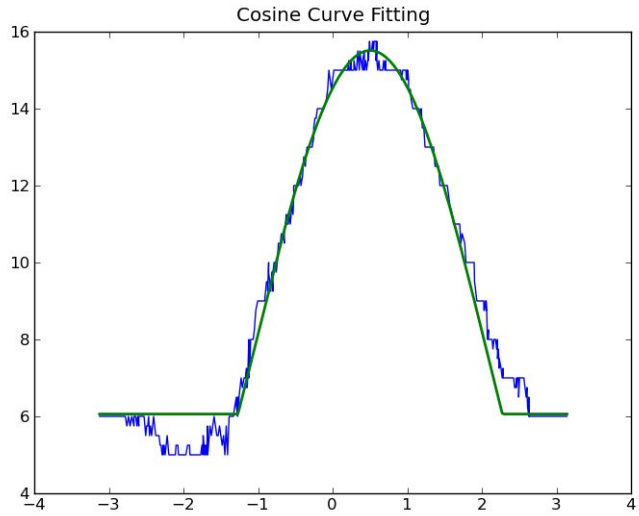
$$I = \gamma + \lambda \max(-\delta, \cos(\phi - \beta)) \tag{4.6}$$

where $\lambda = k_d \sin\theta$, and $\gamma = k_a + \epsilon k_d \cos\theta \approx k_a$.

A good distribution of normal directions along the contour, which provides various β angles throughout the range $(-\pi, \pi]$, allows the algorithm to compute ϕ using least square fitting methods (Figure 4.7). It is not simple to fit a cosine to I because the negative values are clamped to zero. However, I reaches a maximum at $\phi = \beta$, so it is easy to determine ϕ, after which the unknowns λ and γ are easily calculated by curve-fitting. The values of I are obtained by sampling the illumination values along the object contours. Since the object contours obtained in the pre-processing stage are not perfectly accurate, the sample values are noisy. Polynomial curve fitting (Figure 4.8 (a)) smooths the measured intensities so that the maximum is not affected by outliers in the data samples. After determining the value of ϕ from the polynomial fitting, a least squares fit gives estimates for λ and γ. The final fit is excellent (Figure 4.8 (b)).



(a) Polynomial fitting (normal direction from $-\pi$ to π)



(b) Cosine fitting (normal direction from $-\pi$ to π)

Figure 4.8: Plots for Light Direction $\phi = 30°$

The fitted value ϕ is the xy component of the light direction. The next step is to find the z component. The intensity of brightest point on the silhouette I_{smax} occurs when $\cos(\phi - \beta) = 1$. In terms of γ and λ it is

$$I_{smax} = \gamma + \lambda \approx k_a + k_d \sin\theta.$$

If the object's surface normals are well distributed over the image, then the pixel intensity in the MSF image $I = k_a + k_d \hat{n} \cdot \hat{l}$ reaches its maximum when $\hat{n} \cdot \hat{l} = 1$:

$$I_{max} = k_a + k_d.$$

By locating the maximum diffuse value on the object surface, we have

$$k_d \sin\theta = I_{smax} - k_a = I_{smax} - \gamma, \tag{4.7}$$

$$k_d = I_{max} - k_a = I_{max} - \gamma. \tag{4.8}$$

Substituting equation (4.8) into (4.7), the value of θ is estimated by

$$\theta = \sin^{-1}\left(\frac{I_{smax} - \gamma}{I_{max} - \gamma}\right).$$

Correction Based on Shadow

Shadows can be used as a second cue for light direction to refine the initial estimates. The location of an object's cast shadow verifies the correctness of the estimate or adjusts it if necessary. Figure 4.9 illustrates the relationships between light, object, shadow, and their orthogonal projections on the image plane. The light direction is defined by $\hat{l} = (\sin\phi\sin\theta, \cos\phi\sin\theta, \cos\theta)$. In the figure the line formed by point A and its corresponding shadow A_s indicates the value of α, the projection of the light direction on the image plane. This is used to verify if the previously estimated ϕ is correct. Points A and A_s are located by finding the tangent plane that contains both the light source and the camera. This plane is tangent to the object at point A. Point A lies on the object's silhouette on the image plane; its shadow A_s lies on the boundary of the cast shadow (Figure 4.10).

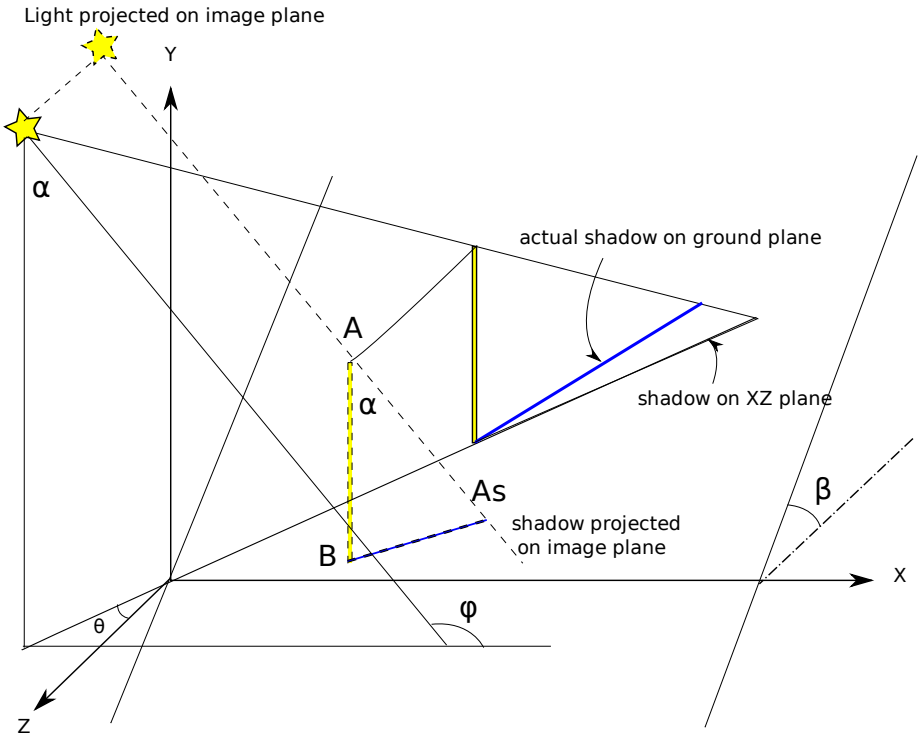


Figure 4.9: Geometry of a light, a stick object, and its shadow

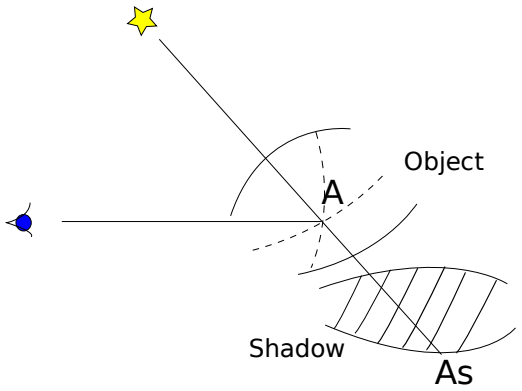


Figure 4.10: Point A on both boundaries and its shadow

Verification and adjustment using cast shadows is simple, consisting of the following steps:

1. The estimated light direction is $\hat{l} = (\sin\phi\sin\theta, \cos\phi\sin\theta, \cos\theta)$. On the image draw a line with slope $\tan\phi$ that is tangent to the object contour. The point of tangency is A.

2. Move the line along the contour near A, rotating it to retain tangency until it is also tangent to the cast shadow contour with intersection point B. Geometrically B is not guaranteed to be the projection of A's shadow position, but the tangency ensures proximity to the real position.

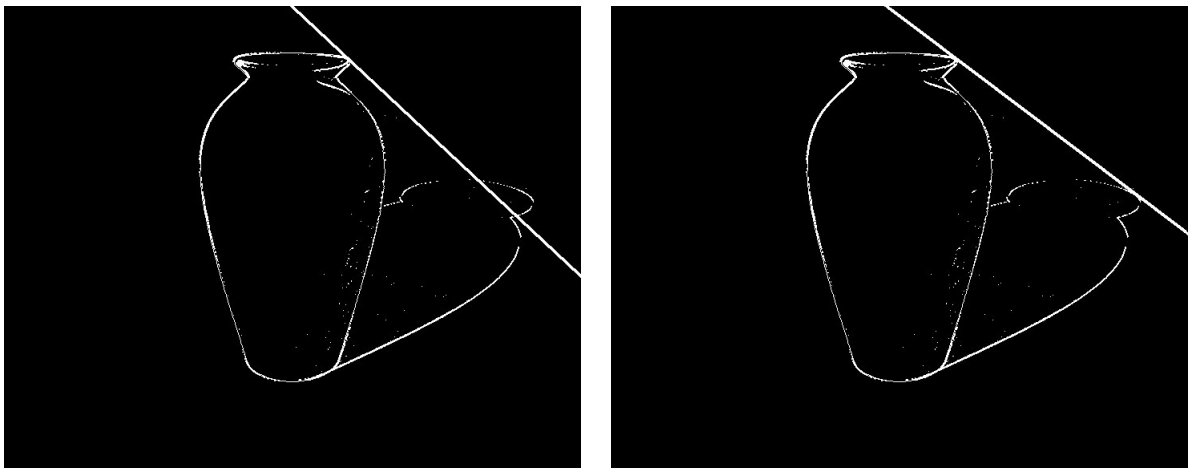
3. Since ine AB represents the projection of the light direction in the image plane, its slope can be used to compute the new ϕ.

A sample correction is shown in Figure 4.11. Note that to compute the correct position of A and B, step 1 and 2 need to be computed iteratively. However, the algorithm chooses not to do so for two reasons. First of all, B is not guaranteed to be geometrically correct even when the calculation is done iteratively. Secondly, such iteration, which is not guaranteed to create a more accurate estimate, is costly and the algorithm is expected to give real-time responses to users. It should also be noted that because this step does not take perspective projection into account, the adjustment does not always lead to perfect estimates.

Light Position

When there is a single object in the scene, only the light direction can be determined. When there are two or more objects, the light position can be triangulated from the two estimated directions. Assume that $P_1 = (x_1, y_1, z_1)$ is the brightest point on one object's contour and $P_2 = (x_2, y_2, z_2)$ is the brightest point on the second object's contour. Their orthogonal projections to the image plane are $P_1' = (x_1, y_1)$ and $P_2' = (x_2, y_2)$. The projections of the light vectors $\hat{l_1} = (l_{1x}, l_{1y}, l_{1z})$ and $\hat{l_2} = (l_{2x}, l_{2y}, l_{2z})$ on the image plane are $\hat{l_1'} = (l_{1x}, l_{1y})$ and $\hat{l_2'} = (l_{2x}, l_{2y})$. If $\hat{l_1'} \neq \hat{l_2'}$, there exists a point S in the image such that

$$S = P_1' + t_1\hat{l_1'} = P_2' + t_2\hat{l_2'}.$$



(a) Initial Estimation: $\phi = 136°$ (b) Final Estimation: $\phi = 143°$

Figure 4.11: Light Direction Estimation (Ground Truth: $\phi = 140°$)

From the previously estimated values of x_1, y_1, x_2, y_2, $\hat{l_1'}$, and $\hat{l_2'}$, t_1 and t_2 are determined by

$$x_1 + t_1 l_{1x} = x_2 + t_2 l_{2x},$$
$$y_1 + t_1 l_{1y} = y_2 + t_2 l_{2y}.$$

The solutions for t_1 and t_2 are

$$t_1 = \frac{1}{l_{1x} l_{2y} - l_{1y} l_{2x}} [(x_2 - x_1) l_{2y} - (y_2 - y_1) l_{2x}], \tag{4.9}$$

and

$$t_2 = \frac{1}{l_{1x} l_{2y} - l_{1y} l_{2x}} [(x_2 - x_1) l_{1y} - (y_2 - y_1) l_{1x}]. \tag{4.10}$$

Note that equation (4.9) and (4.10) are ill-conditioned when $\hat{l_1'} \approx \hat{l_2'}$. In such cases the algorithm aborts the light position calculation and uses only the estimated directions. When $\hat{l_1'}$ and $\hat{l_2'}$ have adequately different values, t_1 and t_2 are computed first. The xy values of the light position are then

$$x = x_1 + t_1 l_{1x}, \quad y = y_1 + t_1 l_{1y}.$$

The algorithm then computes values in the z direction using t_1 and t_2. The relative distance between the two objects is estimated as:

$$z = z_1 + t_1 l_{1z} = z_2 + t_2 l_{2z},$$

$$\Delta z = z_1 - z_2 = t_2 l_{2z} - t_1 l_{1z}.$$

Δz lets the algorithm keep track of the relative distances between the objects. To get an absolute value for each object's depth, however, approximation using camera lens geometry is required.

Depth Estimation

Figure 4.12 shows the optical geometry of a camera lens in the thin lens approximation. The subject at distance S_1 is in focus on the image plane at distance S_2, where $S_1 \gg S_2$. The camera focal length is f. From the thin lens equation,

$$\frac{1}{S_1} + \frac{1}{S_2} = \frac{1}{f}.$$

Since $S_1 \gg S_2$, $\frac{1}{S_1}$ is a small value. As S_1 goes to realistic distances, $f \to S_2$.

$$\frac{1}{S_2} \approx \frac{1}{f}.$$

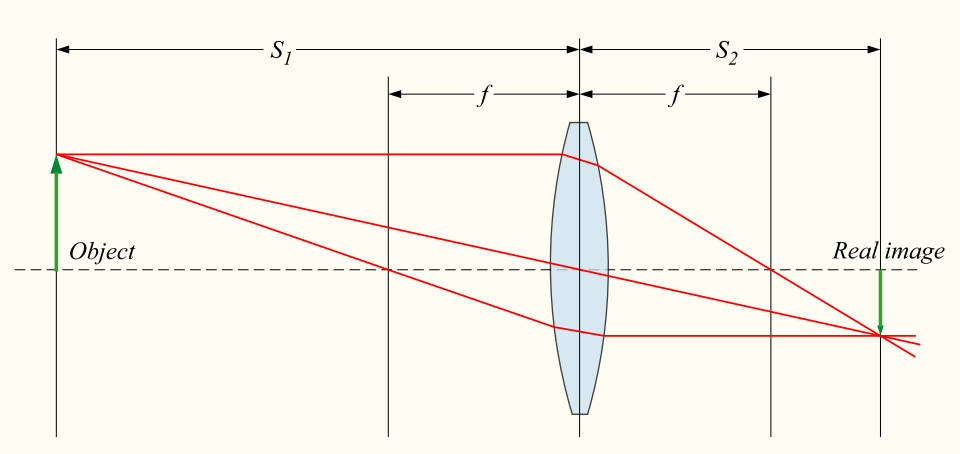


Figure 4.12: Thin Lens Geometry [1]

For most modern cameras, the focal length f is included in a digital photograph's metadata, usually an EXIF specification [1]. If no metadata is available for the input, it is

[1]Exchangeable image file format (EXIF) is a specification for the image file format used by digital cameras.

computed [22]. The distance between the camera and image plane is approximated by f. f is also the base depth, in terms of which all object depths are calculated. There is no mathematical proof for f being the base depth, but it is the only depth value that is easily estimated. Depth f is assigned to the object closest to the camera. The depth values of the rest of the objects are calculated based upon the depth difference between each other. Finally, the calculation of the light position is completed using one of the object depths:

$$z = z_1 + t_1 l_{1z} \tag{4.11}$$

$$= f + \Delta z + t_1 l_{1z} \tag{4.12}$$

Results

To test the correctness of illumination estimation, several tests were conducted using images with known light and object positions. Table 4.1 shows estimated directions compared with true values. The light positions are listed using a unit defined by the rendering software. The estimated ϕ values are generally close to the ground truth. The differences fall within the criteria proposed in Pentland's user study [46]. The estimated θ values, however, have larger errors owing to errors introduced by perspective projection.

Table 4.1: Light Direction Estimation Results

Actual Light Position	Direction to (0, 0, 0)	Estimated Direction
(2.96, 1.71, 9.40)	$\phi = 30°$, $\theta = 20°$	$\phi = 31.96°$, $\theta = 15.42°$
(2.83, 3.21, 8.66)	$\phi = 40°$, $\theta = 30°$	$\phi = 38.89°$, $\theta = 38.29°$
(4.13, 4.92, 7.66)	$\phi = 50°$, $\theta = 40°$	$\phi = 48.53°$, $\theta = 43.70°$
(2.50, 4.33, 8.66)	$\phi = 60°$, $\theta = 30°$	$\phi = 60.20°$, $\theta = 38.28°$
(1.12, 6.33, 7.66)	$\phi = 80°$, $\theta = 40°$	$\phi = 81.44°$, $\theta = 45.62°$
(0.00, 6.43, 7.66)	$\phi = 90°$, $\theta = 40°$	$\phi = 90.06°$, $\theta = 42.18°$
(-2.62, 7.20, 6.43)	$\phi = 110°$, $\theta = 50°$	$\phi = 111.25°$, $\theta = 50.42°$
(-3.83, 6.63, 6.43)	$\phi = 120°$, $\theta = 50°$	$\phi = 119.81°$, $\theta = 50.09°$
(-2.20, 2.62, 9.40)	$\phi = 130°$, $\theta = 20°$	$\phi = 123.97°$, $\theta = 31.01°$
(-7.20, 6.04, 3.42)	$\phi = 140°$, $\theta = 70°$	$\phi = 143.83°$, $\theta = 61.45°$
(-7.50, 4.33, 5.00)	$\phi = 150°$, $\theta = 60°$	$\phi = 153.44°$, $\theta = 57.66°$

Table 4.2 shows the estimated positions in image space, also using a unit defined by the

used rendering software. In this case both ϕ and θ can be quite different from their true values. The error can be as large as 10°. Overall the estimations are not very accurate, but they provide us with sufficient information about the light source.

Table 4.2: Light Position Estimation Results

Distance	Direction	Estimated Distance	Estimated Direction
10	$\phi = 30°$, $\theta = 20°$	8.832	$\phi = 35.12°$, $\theta = 26.06°$
10	$\phi = 70°$, $\theta = 50°$	10.386	$\phi = 72.65°$, $\theta = 52.50°$
10	$\phi = 130°$, $\theta = 30°$	10.737	$\phi = 122.65°$, $\theta = 29.02°$
10	$\phi = 150°$, $\theta = 40°$	12.765	$\phi = 144.64°$, $\theta = 30.73°$

4.2.2 Shape Recovery

Knowing the light source, it is now easy to reconstruct the geometry of objects in the scene. There are two important types of shape: the ground plane on which shadows are cast, and objects placed on the plane. Different techniques are used for recovering the different types, with details discussed in the following subsections.

Ground Plane Orientation

Objects are assumed to be placed on a large ground plane that is uniform in colour. The plane's geometry is effectively its orientation, and the main visual cue for the orientation is shadows cast on the plane. Cast shadows are of little use for recovering object shapes, but they facilitate perception of the object on which the shadows are cast.

Figure 4.13 shows the side view of a scene. Shadow points S_1 and S_2 are the projection of T_1 and T_2 from the light source. The projections of S_1 and S_2 on the image plane are Q_1 and Q_2, and the projections of T_1 and T_2 are P_1 and P_2. The ground plane is described as $G = (G_x, G_y, G_z, G_w)$, where $G_x^2 + G_y^2 + G_z^2 = 1$. Almost all photographs are taken with the photographer holding the camera parallel to the horizon, so it is rare to have the ground plane tilted sideways. The orientation of the ground plane thus can be simplified to $G = (0, G_y, G_z, G_w)$, where the slant in x direction is 0. This assumption changes the number of unknown variables to three. To construct G, it is necessary to locate

the projections of (P_1, P_2) on the object silhouette and of (Q_1, Q_2) in the cast shadow to establish the additional two equations to solve G.

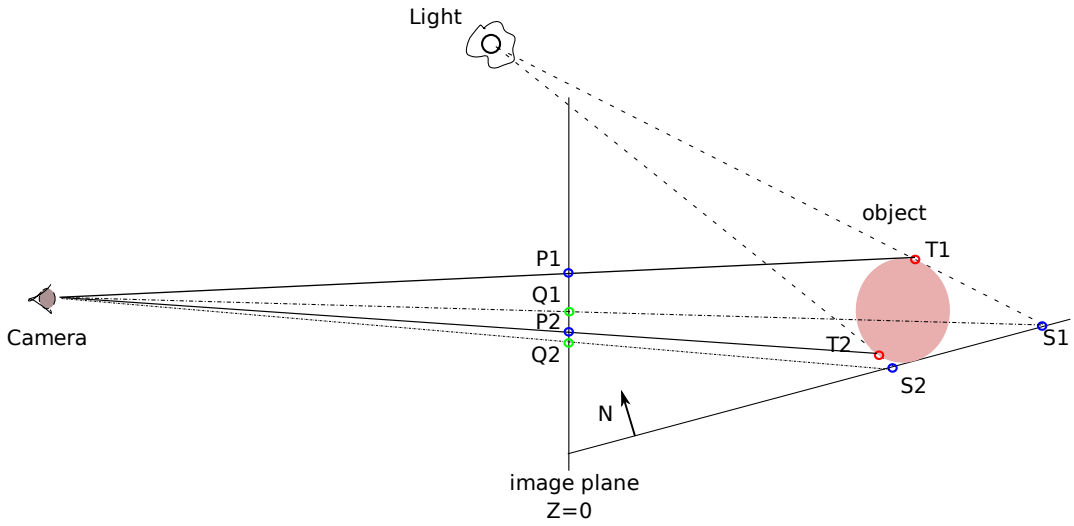


Figure 4.13: Side view of a scene. S_1 and S_2 are visible shadow points on the side of the object.

The first set of corresponding points is easy to find, since a line that is tangent to both the object contour and cast shadow has been found in the light recovery step. The two intersection points form the first corresponding pair P_1 and Q_1. The second pair, P_2 and Q_2, can be located by finding the second line that is tangent to both the object contour and cast shadow using the light direction (Figure 4.14).

The algorithm now has all three equations of corresponding points required for the calculation. Orthogonal projection is used again to simplify the computation. The view direction is $(0, 0, -1)$, and the following holds true:

$$T_{1x} = P_{1x}, \quad T_{1y} = P_{1y}, \quad T_{1z} = z_o, \tag{4.13}$$

where z_o is the object depth obtained in the light recovery step. For pair P_1 and Q_1 we know that

$$S_1 = Q_1 + (0, 0, -1)t = T_1 + \hat{l_1} t_1, \tag{4.14}$$

$$S_1 \cdot G = 0, \tag{4.15}$$

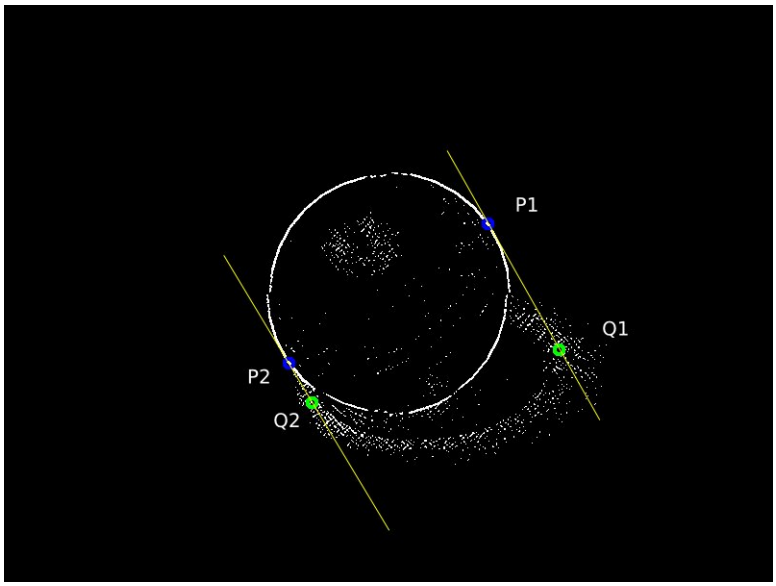


Figure 4.14: First Two Point Pairs

where $S_1 = (S_{1x}, S_{1y}, S_{1z}, 1)$ and $\hat{l_1}$ is the light direction for T_1. The light direction is calculated for each point if the light position is known, otherwise the same light direction would be used for both pairs. Expanding Equation (4.14) gives

$$Q_{1x} = T_{1x} + l_{1x}t_1,$$

$$Q_{1y} = T_{1y} + l_{1y}t_1,$$

$$Q_{1z} - t = T_{1z} + l_{1z}t_1.$$

T_{1x} and T_{1y} are replaced by P_{1x} and P_{1y} to get

$$t_1 = (P_{1x} - Q_{1x})/l_{1x} = (P_{1y} - Q_{1y})/l_{1y}.$$

Substitute equation (4.13) and t_2 into equation (4.15):

$$(P_{1x} + t_1 l_{1x})G_x + (P_{1y} + t_1 l_{1y})G_y + (z_o + t_1 l_{1z})G_z + G_w = 0,$$

which is also true for the other pair,

$$(P_{2x} + t_2 l_{2x})G_x + (P_{2y} + t_2 l_{2y})G_y + (z_o + t_2 l_{2z})G_z + G_w = 0.$$

Having these two equations together with the normal constraint $G_x^2 + G_y^2 + G_z^2 = 1$, the algorithm now can solve G.

Table 4.3 shows some estimated orientation using this method. Using the actual light directions as input, the method performs well, with slightly larger errors when the light directions are more oblique. However, since the algorithm heavily depends on the light direction: results can be quite wrong if the estimated light directions are not correct. For example, in Table 4.4 the first case shows that the estimated θ value is 29.6° while the actual θ is 20°. The estimated plane orientation thus also has a large error of 9.6°. Similar errors appear in the last two cases in Table 4.4. Regardless of the possible errors, the estimates are still close enough to the ground truth and can be used to generate visually convincing relit images.

Table 4.3: Plane Orientation Estimation Using True Light Directions

Light Direction	Actual Orientation	Estimated Orientation
$\phi = 20°$, $\theta = 20°$	(0, 0.94, 0.34) $\alpha = 20°$	(0, 0.87, 0.49) $\alpha = 18.34°$
$\phi = 30°$, $\theta = 40°$	(0, 0.94, 0.34) $\alpha = 20°$	(0, 0.94, 0.33) $\alpha = 19.52°$
$\phi = 60°$, $\theta = 30°$	(0, 0.87, 0.50) $\alpha = 30°$	(0, 0.85, 0.52) $\alpha = 31.37°$
$\phi = 90°$, $\theta = 40°$	(0, 0.77, 0.64) $\alpha = 40°$	(0, 0.76, 0.65) $\alpha = 40.84°$
$\phi = 120°$, $\theta = 50°$	(0, 0.87, 0.50) $\alpha = 30°$	(0, 0.88, 0.47) $\alpha = 27.90°$
$\phi = 150°$, $\theta = 60°$	(0, 0.91, 0.41) $\alpha = 20°$	(0, 0.91, 0.41) $\alpha = 24.29°$

Table 4.4: Plane Orientation Estimation Using Estimated Light Directions

Light Direction	Actual Orientation	Estimated Direction	Estimated Orientation
$\phi = 20°$, $\theta = 20°$	(0, 0.94, 0.34) $\alpha = 20°$	$\phi = 19.5°$, $\theta = 29.6°$	(0, 0.87, 0.49) $\alpha = 29.6°$
$\phi = 30°$, $\theta = 40°$	(0, 0.94, 0.34) $\alpha = 20°$	$\phi = 27.3°$, $\theta = 41.3°$	(0, 0.94, 0.33) $\alpha = 20.2°$
$\phi = 60°$, $\theta = 30°$	(0, 0.87, 0.50) $\alpha = 30°$	$\phi = 59.8°$, $\theta = 35.2°$	(0, 0.91, 0.42) $\alpha = 25.1°$
$\phi = 90°$, $\theta = 40°$	(0, 0.77, 0.64) $\alpha = 40°$	$\phi = 96.0°$, $\theta = 37.9°$	(0, 0.74, 0.67) $\alpha = 42.4°$
$\phi = 120°$, $\theta = 50°$	(0, 0.87, 0.50) $\alpha = 30°$	$\phi = 122.4°$, $\theta = 41.8°$	(0, 0.92, 0.38) $\alpha = 22.5°$
$\phi = 150°$, $\theta = 60°$	(0, 0.94, 0.34) $\alpha = 20°$	$\phi = 147.8°$, $\theta = 48.8°$	(0, 0.93, 0.36) $\alpha = 20.9°$

Shape from Shading

Shape from shading (SFS) algorithms are chosen to recover object shapes since, for the most part, they require only light position as the prior information. Many SFS algorithms exist. To determine the most effective algorithm, several algorithms from [71] and [11] were tested. The linear algorithm developed by Tsai and Shah (TS) [59] is extremely fast but can be affected by self shadows and specular highlights (Figure 4.15 (a)(b)). As a result the relighting quality is poor because the surface estimate is quite different from the real shape. The global algorithm developed by Daniel and Durou (DD) [8] generates the most smooth surfaces overall, but has a major disadvantage: it restricts the light direction to the z axis. When given an image that is not lit from the right direction, the algorithm produces a result that has a high central peak and is also affected by illumination effects such as shadows and highlights (Figure 4.15 (c)(d)).

The algorithm of Daniel and Durou was modified to incorporate the light direction. It starts with the basic equations for the SFS problem. The height of each pixel $h(x, y)$ with respect to the corresponding object 2D panel is the unknown to be determined. (p, q) defines the pixel tangent plane:

$$p = \frac{\partial h}{\partial x}, \quad q = \frac{\partial h}{\partial y}.$$

The normal $\hat{n}$ is perpendicular to both tangent vectors

$$n_x + p(x, y)n_z = 0,$$

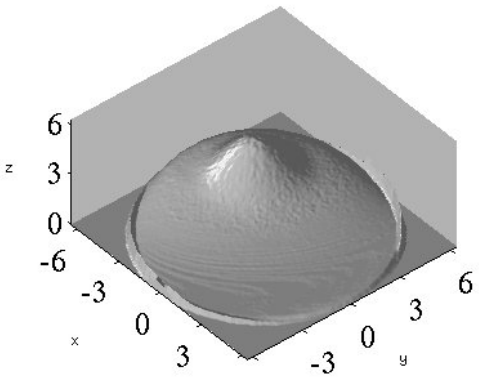
$$n_y + q(x, y)n_z = 0.$$

It is therefore written as a function of p and q:

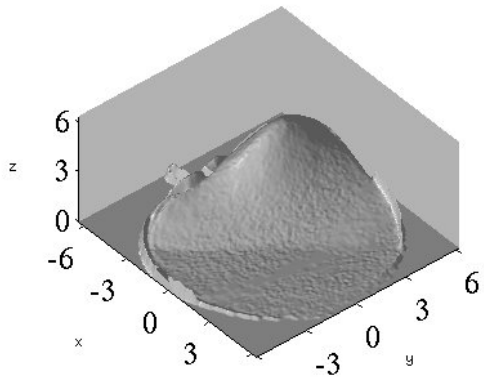
$$\hat{n} = \frac{1}{\sqrt{1 + p^2 + q^2}}(-p, -q, 1)^T.$$

Ideally, the reflectance value for each $\hat{n}$ should be the same as the image intensity at that position $R(\hat{n}) \approx I(x, y)$. The reflectance function can be expressed as follows if we assume diffuse surfaces:

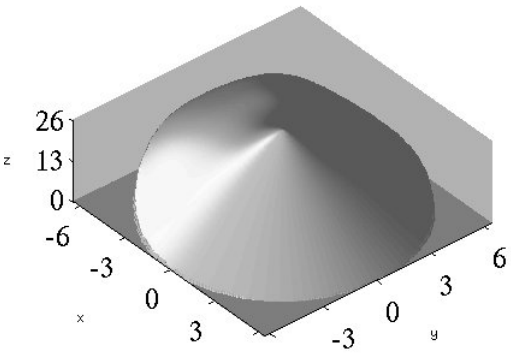
$$R(\hat{n}) = k_d \max(0, \hat{l} \cdot \hat{n}),$$



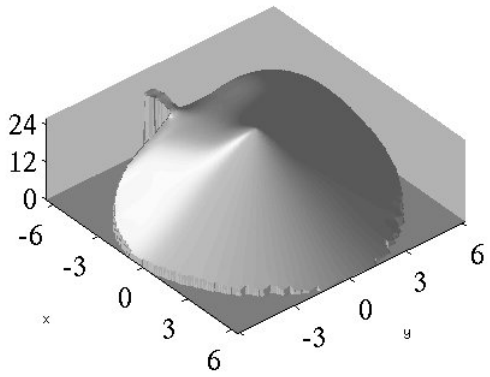
(a) Reconstructed Sphere Using TS



(b) Reconstructed Apple Using TS



(c) Reconstructed Sphere Using DD



(d) Reconstructed Apple Using DD

Figure 4.15: Shape from Shading Results

where k is a constant that depends on the characteristics of the observer and on the albedo of the surface [8] and $\hat{l}$ is the light direction. Substituting for $\hat{n}$ the reflectance is

$$R(\hat{n}) = \frac{k}{\sqrt{1+p^2+q^2}} |\hat{l} \cdot (-p, -q, 1)^T|.$$

When $\hat{l}$ is parallel to as $\hat{n}$, $k = I_{max}$.

Daniel and Durou's method is an optimization method that minimizes the objective function $F(p, q)$ with respect to p and q:

$$F(p, q) = E(p, q) + T(p, q) + S(p, q),$$

where $E(p,q)$ is the error term for all pixel (x,y)

$$E(p,q) = \int_{(x,y)} \left[R(\hat{n}) - I(x,y)\right]^2 dxdy,$$

$T(p,q)$ is the integrability constraint with μ as its Lagrange multiplier

$$T(p,q) = \mu \int_{x\in\Omega} \left[\frac{\partial p(x)}{\partial y} - \frac{\partial q(x)}{\partial x}\right]^2 dxdy,$$

and $S(p,q)$ is the smoothness constraint with the smoothness factor λ

$$S(p,q) = \lambda \int_{x\in\Omega} \left[|\nabla p(x,y)|^2 + |\nabla q(x,y)|^2\right] dxdy.$$

The MSF image generated in the preprocessing stage is $I(x,y)$. Instead of using the reflectance function derived by Daniel and Durou [8]

$$R(\hat{n}) = \frac{k}{\sqrt{1+p^2+q^2}},$$

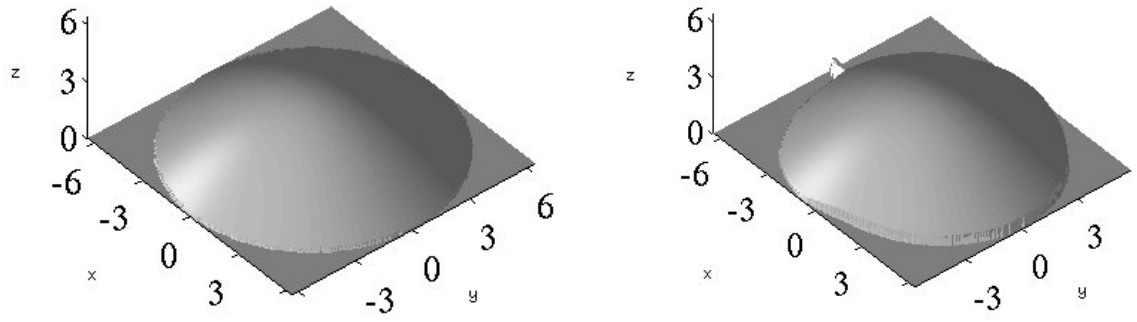
the algorithm uses the version that contains the light direction:

$$R(\hat{n}) = \frac{k}{\sqrt{1+p^2+q^2}} |\hat{l} \cdot (-p,-q,1)^T|.$$

The gradient of $F(p,q)$ is computed using the new equation accordingly, for gradient descent optimization.

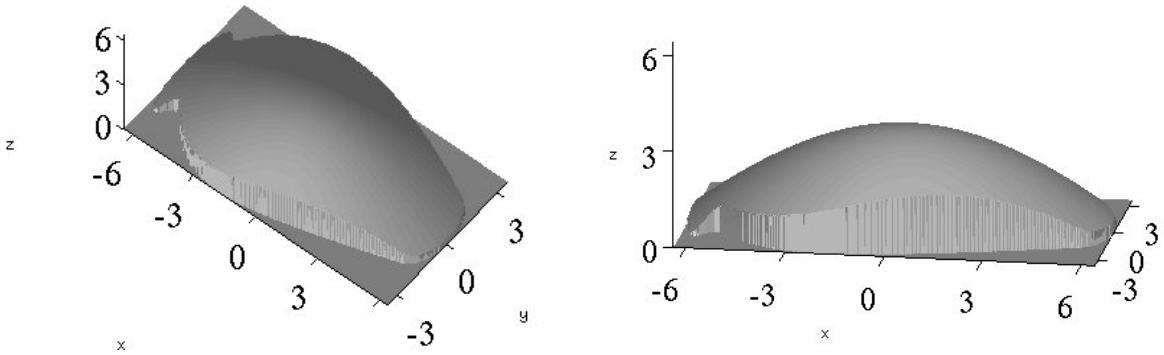
Figure 4.16 shows the reconstructed height values of a sphere and an apple using the same input as was used to create in Figure 4.15. By introducing the light position, the surface height no longer has a spurious central peak. The spurious shape changes caused by illumination effects also disappear.

However, the modified algorithm has remaining problems. It works well for smooth and closed surfaces like spheres or apples, but it fails to follow detailed curvatures in the shape (Figure 4.17). There are several reasons for this weakness, including the initialization of (p,q), the integrability and smoothness constraints of the surface, and boundary conditions. Other methods, that can build more precise reconstructions, are needed.



(a) Sphere (b) Apple

Figure 4.16: Modified DD Results



(a) Top View (b) Side View

Figure 4.17: Modified DD Result for a Jar

Symmetry and Surfaces of Revolution

In addition to the shading cues used by SFS algorithms, the image has another cue that is useful for reconstructing the surface: symmetry. Daily life is full of symmetric objects, or near symmetric objects including plants, flowers, architecture, and furniture. Hence, it is common that photographs contain symmetric objects. If an object is symmetric, the symmetry can be exploited to build its surface shape unaffected by shadows or highlights.

Given an object contour extracted in the preprocessing stage, it is straightforward to test whether the object is symmetric and to locate its symmetry axis. A simple and

fast algorithm developed by Li et al. [30] is adopted. Li et al.'s method processes the entire image and locates the top N most salient symmetry axes. Since the contour of each individual object is available separately, the method is modified to work with one object at a time. Figure 4.18 illustrates the basic idea of the algorithm. An empty histogram $H[x][\theta]$ is created first. It accumulates votes in terms of rotation angle θ and distances x from the axis. First, a bounding box is constructed for each object contour and the centroid of the object is computed. The contour pixels are then rotated around the centroid by θ where $\theta \in [-90°, 90°]$. After rotation every pixel is paired with every other pixels in the same row. The pairs then vote for their center position as the symmetry line. The voting is done for all discrete θ values and the maximum value in H indicates the symmetry axis.

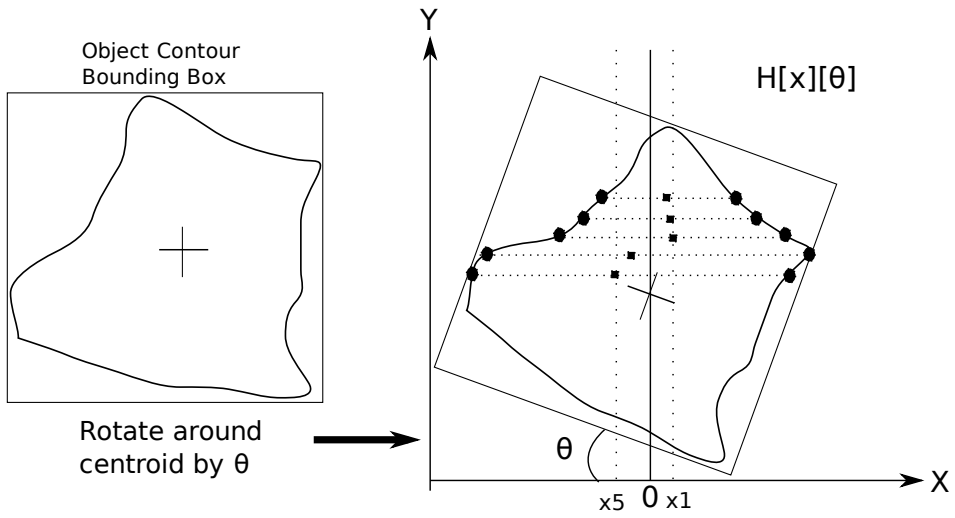


Figure 4.18: Symmetry Detection

A special case of symmetry, surfaces of revolution, gives us a full description of the object surface. A surface of revolution is a surface in Euclidean space created by rotating a curve around a straight line, the axis [41]. Once the object contour and the symmetry axis are discovered during symmetry detection, an analytical representation for the object surface is calculated.

There exists no simple algorithm to determine which objects are surfaces of revolution. The symmetry detection method finds a symmetry axis even if the shape is not really symmetric. Without a simple algorithm to verify symmetry and to search for surfaces of

revolution, the program asks the user to indicate if the shape recovery process should use surfaces of revolution as the reconstruction method.

If the object is indicated as a surface of revolution by the user, the symmetry detection mentioned above provides the symmetry axis. The axis is rotated to the x axis, making half of the object contour a function. The contour pixels are fit to a piece-wise Bézier spline using the fitting method developed by Schneider [54] which provides a function form for the contour. Instead of taking all the contour points as inputs, which could be noisy, several samples are used for curve-fitting, sampled sparsely in regions of low curvature, and densely in regions of high curvature. The sampled contour points are then passed to the Bézier fitting algorithm to generate a set of Bézier segments, each with four control points. This method is able to fit both simple and complex curves. Figure 4.19 shows two plots with the original contour values (blue), sample points (green), Bézier control points (red) and fitted Bézier curves (cyan).

After the Bézier curves are computed, they are rotated about the symmetry axis to build a height map. A normal map is then computed by evaluating the tangent vectors of the height map (Figure 4.20).

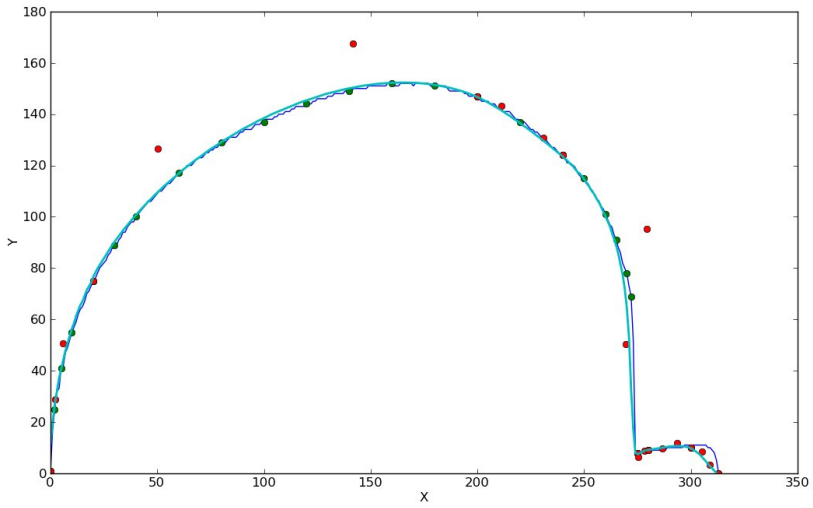
Normal Map Adjustment

The final step of shape recovery generates a composite normal map comprising both the ground plane and all objects. This straightforward process is done in two steps.

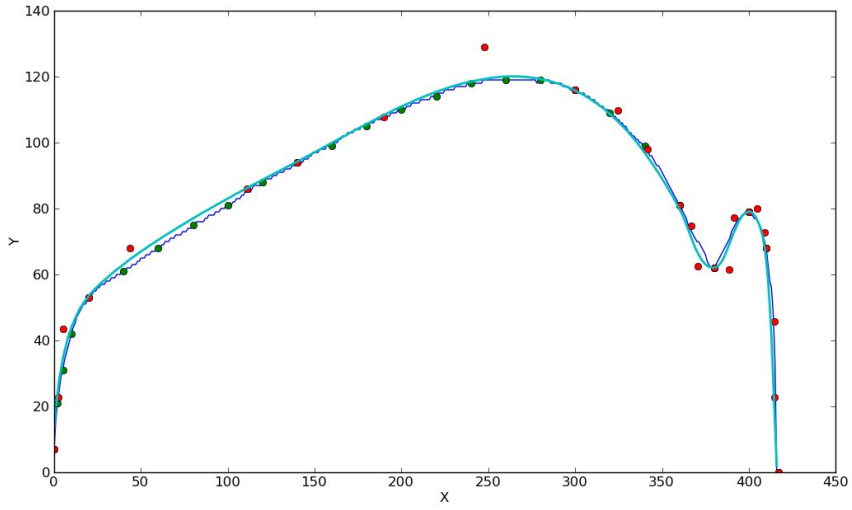
1. Start with the ground plane parallel to the xz plane and initialize all normals to $(0, 1, 0)$. Overwrite the ground plane normal map with the normal maps of the objects where appropriate.

2. Rotate the normal vectors by the ground plane orientation.

Figure 4.21 shows two examples of the generated normal map. The xyz values of the normals, ranging from $[-1, 1]$, are shifted and then scaled to rgb channels of range $[0, 1]$ in the image.

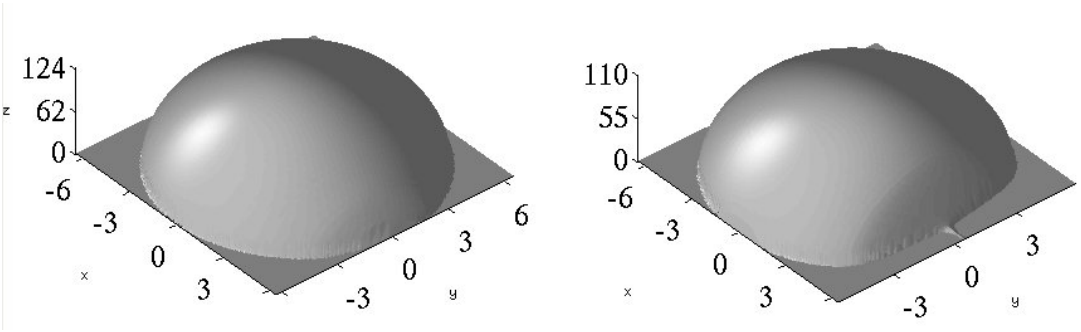


(a) Contour for an Apple



(b) Contour for a Vase

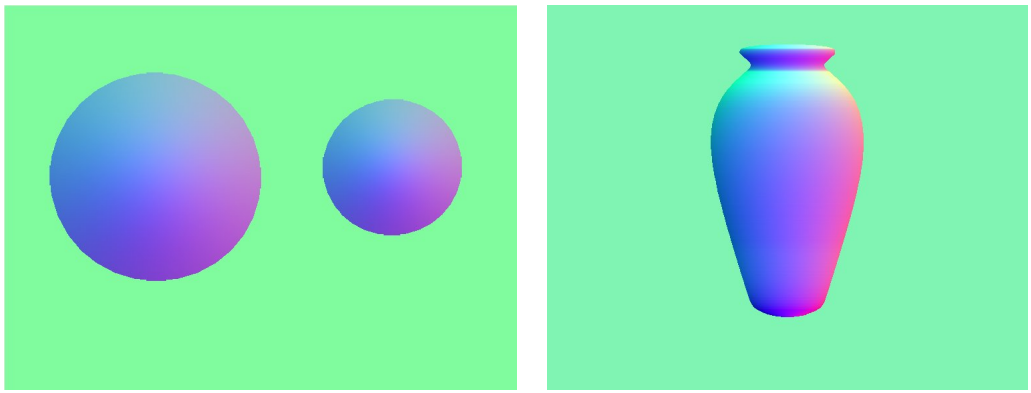
Figure 4.19: Bézier Curve Fitting



(a) Sphere (b) Apple

(c) Cylinder (d) Jar

Figure 4.20: Reconstructed Surfaces



(a) Spheres Normal Map using SFS (b) Jar Normal Map using Revolution

Figure 4.21: Normal Map Results

4.3 Relighting Stage

With the necessary 2D and $2\frac{1}{2}$D information available, it is now possible to modify the object apperances with new light directions. Relighting is divided into three steps, which change the diffuse, specular, and shadow apperances. The following sections present the algorithmic details of each part.

4.3.1 Relighting the Diffuse Component

The normal map of the object and the light source position have been obtained in the processing stage. To change the diffuse appearance, one last piece of information is necessary – the surface reflectance. In the specular separation step, the MSF image is generated to represent the diffuse components of each object. Using the estimated light source, the normal map, and the MSF images, the diffuse albedo for the objects is approximated. Pseudo-code below shows how to calculate the diffuse albedo for each object.

ObjectMaterial($MSF, \hat{l}, \hat{n}, objmask$)

```
1  Split each n̂ to n_x, n_y, n_z in the normal map
2  Split l̂ to l_x, l_y, l_z
3  nl = l_x n_x + l_y n_y + l_z n_z
4  for each pixel p
5        if nl(p) ⩽ 0
6              nl(p) = 1
7        albedo(p) = MSF(p) / nl(p)
8  albedo = albedo ∩ objmask
```

There is no MSF image for the ground plane, so the same calculation cannot be applied to it. Fortunately, because it is uniform in colour, it is necessary to retrieve only a single colour value. After discarding the pixels in the cast shadows, the algorithm uses the original pixel values, as well as the product of the ground plane normal map and the original light direction, to compute an array of colour candidates. The average value of this array is the ground plane colour. The code block below outlines the algorithm.

BACKGROUNDMATERIAL($image, objmask, castshadow, l, n$)

```
1  bgmask = ¬objmask
2  bgmask = bgmask - castshadow
3  split each n̂ to (n_x, n_y, n_z) in the normal map
4  split l̂ to (l_x, l_y, l_z)
5  nl = l_x n_x + l_y n_y + l_z n_z
6  nl = nl ∩ bgmask
7  for each pixel p
8        sum += image(p) / nl(p)
9  rgb = sum / #pixels
10 bgmask = ¬objmask
11 for each pixel p
12       if bgmask(p)
13             C(p) = rgb
```

Now we have everything we need to generate the new diffuse appearances for the objects. Given a new array of light directions $\hat{l}_{new}$ and material colour C for each pixel, the new diffuse value D_{new} is calculated as

$$D_{new} = C \cdot (\hat{l}_{new} \cdot \hat{n})$$

4.3.2 Relighting the Specular Component

For the specular component, user input is obtained to get the characteristics of the specular component, its specular hardness, α, and brightness, k_s. These might be determined from the image, but it seems likely that users would desire to modify these properties as a part of relighting, so they are treated as user-controlled, as are the new light positions.

The specular component is calculated as described in the Phong shading model. The reflection direction $\hat{r}$ is computed using the normal map, the light source, and view direction $\hat{v}$. The highlight value is then computed as $k_s(\hat{r} \cdot \hat{v})^{\alpha}$. Pseudo-code of such calculation is shown below.

ObjectSpecular(k_s, α, n, l, v)

---

1 **for** each pixel p
2 $\quad r(p) = 2n(p) - l$ // compute reflected direction
3 split each $\hat{r}$ to (r_x, r_y, r_z) for all pixels
4 split each $\hat{v}$ to (r_x, r_y, r_z) for view vectors
5 $spec = \max(0, r_x v_x + r_y v_y + r_z v_z)$
6 Normalize($spec$)
7 $spec = spec^{alpha}$
8 $spec = k_s spec$

4.3.3 Relighting the Shadows

Changing the shadows is more complex, consisting of a series of transformations. The algorithm starts with the object bounding box as a 2D panel in the image plane ($z = 0$) [14], and carries out the following steps:

1. M_T – Translate the panel to put the image center at the origin.
2. M_s – Move the panel to the object's estimated depth Z.
3. M_p – Project the panel onto the ground plane using the light as the centre of projection to form the shadow.
4. M_v – Project the shadow to the image plane orthogonally.

Among those transformations, M_T and M_s are simple translations. M_v is a matrix based on the positions of the camera, image plane, and 2D panel depth. Since the distance between the camera and image plane as well as object depths are both estimated values, there is no guarantee of the correctness of M_v if it is a perspective projection matrix. Orthogonal projection therefore is used and M_v has a simple orthogonal projection form. M_p, however, is more complex. A point $\vec{P} = (P_x, P_y, P_z, 1)$ on the panel is projected through the light source $\vec{L} = (L_x, L_y, L_z, 1)$ onto the ground plane as $\vec{S} = (S_x, S_y, S_z, 1)$. $G = (G_x, G_y, G_z, G_w)$ describes the plane. Since $\vec{S}$, $\vec{P}$ and $\vec{L}$ form a straight line,

$$\vec{S} = \vec{P} + t(\vec{L} - \vec{P}),$$
$$\vec{S} \cdot G = 0,$$

which expands to

$$S_x = (1-t)P_x + tL_x,$$
$$S_y = (1-t)P_y + tL_y,$$
$$S_z = (1-t)P_z + tL_z,$$
$$S_w = (1-t)P_w + tL_w,$$

and

$$G_x S_x + G_y S_y + G_z S_z + G_w S_w = 0.$$

It is convenient to use the abbreviations $G_a S_a = G_x S_x + G_y S_y + G_z S_z + G_w S_w$ and $G_a P_a = G_x P_x + G_y P_y + G_z P_z + G_w P_w$, a variant of the Einstein summation convention. Use the equations above to solve for t:

$$t = \frac{G_a P_a}{G_a P_a - G_a L_a}.$$

Now rewrite $\vec{S}$ as

$$S_x = \frac{1}{G_a P_a - G_a L_a}(G_a P_a L_x - G_a L_a P_x)$$
$$S_y = \frac{1}{G_a P_a - G_a L_a}(G_a P_a L_y - G_a L_a P_y)$$
$$S_z = \frac{1}{G_a P_a - G_a L_a}(G_a P_a L_z - G_a L_a P_z)$$

These equations can be written as a projection matrix M_p so that

$$\begin{bmatrix} S_x \\ S_y \\ S_z \\ S_w \end{bmatrix} = M_p \begin{bmatrix} P_x \\ P_y \\ P_z \\ P_w \end{bmatrix} \tag{4.16}$$

where

$$M_p = \begin{bmatrix} G_x L_x - G_a L_a & G_y L_x & G_z L_x & G_w L_x \\ G_x L_y & G_y L_y - G_a L_a & G_z L_y & G_w L_y \\ G_x L_z & G_y L_z & G_z L_z - G_a L_a & G_w L_z \\ G_x L_w & G_y L_w & G_z L_w & G_s L_w - G_a L_a \end{bmatrix}. \tag{4.17}$$

The denominator $G_aP_a - G_aL_a$ is successfully embedded in the matrix as $G_aP_aL_w - G_aL_aP_w$ where $L_w = 1$ and $P_w = 1$. A similar derivation can be made knowing only the light direction $\vec{D} = (D_x, D_y, D_z, 0)$:

$$M_p = \begin{bmatrix} G_xD_x - G_aD_a & G_yD_x & G_zD_x & G_wD_x \\ G_xD_y & G_yD_y - G_aD_a & G_zD_y & G_wD_y \\ G_xD_z & G_yD_z & G_zD_z - G_aD_a & G_wD_z \\ 0 & 0 & 0 & -G_aD_a \end{bmatrix} \tag{4.18}$$

The final transformation M is computed as $M_vM_pM_sM_T$, where M_sM_T can be written as

$$M_sM_T = \begin{bmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

and $M_pM_sM_T$ is computed as

$$M_pM_sM_T = \begin{bmatrix} G_xL_x - G_aL_a & G_yL_x & G_zL_x & G_ad_aL_x - G_aL_ad_x \\ G_xL_y & G_yL_y - G_aL_a & G_zL_y & G_ad_aL_y - G_aL_ad_y \\ G_xL_z & G_yL_z & G_zL_z - G_aL_a & G_ad_aL_z - G_aL_ad_z \\ G_xL_w & G_yL_w & G_zL_w & G_ad_aL_w - G_aL_ad_w \end{bmatrix}.$$

It is a 4×4 matrix, but transforming the 2D panel to a new 2D shadow position only needs x and y values. Since the algorithm starts from the image plane where every point P can be expressed as $(P_x, P_y, 0, 1)$, the final matrix M can be expressed as a 3×3 projective matrix with one row and one column removed:

$$M = M_vM_pM_sM_T = \begin{bmatrix} G_xL_x - G_aL_a & G_yL_x & G_ad_aL_x - G_aL_ad_x \\ G_xL_y & G_yL_y - G_aL_a & G_ad_aL_y - G_aL_ad_y \\ G_xL_w & G_yL_w & G_ad_aL_w - G_aL_ad_w \end{bmatrix}. \tag{4.19}$$

The corresponding shadow position is then calculated by applying M to the object mask.

4.4 Summary

The three stages of the algorithm provide a partial answer to the second question asked at the beginning of this thesis: how much 2D information can be extracted from the input? What is the minimal 3D information for relighting?

Three types of 2D information are extracted in the pre-processing stage: object contours, cast shadow maps, and specular highlights. Object contours are the most important, and are used throughout the process. Most importantly, they provide an initial estimate of the light direction. The contours also mark the boundaries for the SFS method, or act as the profile for surfaces of revolution in shape recovery. Cast shadow maps are also important for verifying the correctness of light estimation and for providing the means of recovering the ground plane orientation.

To relight an image, two essential things are needed: the light position in 3D space, and the object geometry. Using the method described here, only the light position is obtained in 3D. The objects are flat 2D panels with associated depth values and normal maps, a $2\frac{1}{2}$D representation of the scene. The diffuse and specular appearances can be successfully updated with the normal maps. The position of the light and the depths of the object panels are enough to generate shadows. Based on the algorithm in processing stage, we can say that there is no need for a full 3D reconstruction to achieve relighting. A $2\frac{1}{2}$D representation suffices for such tasks.

Chapter 5

Results

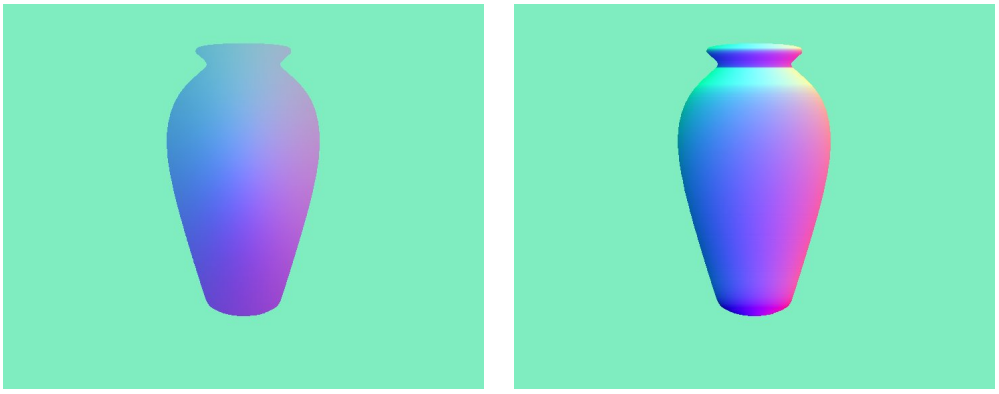
This chapter presents some results generated using the new method. The algorithm was tested on six input images, three of which are synthetic images, three of which are real photographs. A user interface was built to display the results of different stages. It allows users to adjust the 2D structures from the preprocessing stage before proceeding to the processing stage, in which computation is finished within a few seconds, so that users quick feedback of the relighting effects. The relighted images were analyzed to discern its strong and weak points. The analyses, summarized in Section 5.7, provide the answer to the third question: what is the most difficult aspect of the relighting problem?

5.1 One Jar

The first test image is an image consisting of one jar (Figure 5.1). Both SFS and surfaces of revolution (SoR) methods were run to relight the image.

We first compare the normal maps that each method generates. As shown in Figure 5.2, the normal map generated using SFS is very smooth. However, the xy components of the normal vectors are small along the boundary. Nor is curvature around the neck of the jar correct. The normal map generated using SoR is closer to the actual shape of the jar, but the surface is not as smooth. There is a sudden change of normal at the neck position.

Figure 5.1: Original Image – Jar



(a) Normal Map by SFS (b) Normal Map by SoR

Figure 5.2: Normal Map Comparison – Jar

The results of changing the existing light are shown in Figure 5.3. A ground truth image was generated to examine which method produces more accurate results. Since relighting only uses a directional light source, global illumination effects, which are present in the ground truth image, cannot be found in the relit images. The normal map generated by SoR is a more accurate description of the object surface, so its resulting image is closer to the actual image, although it has a bright spot on the top of the jar due to the large y component in the normal map. The result of SFS is not visually convincing.

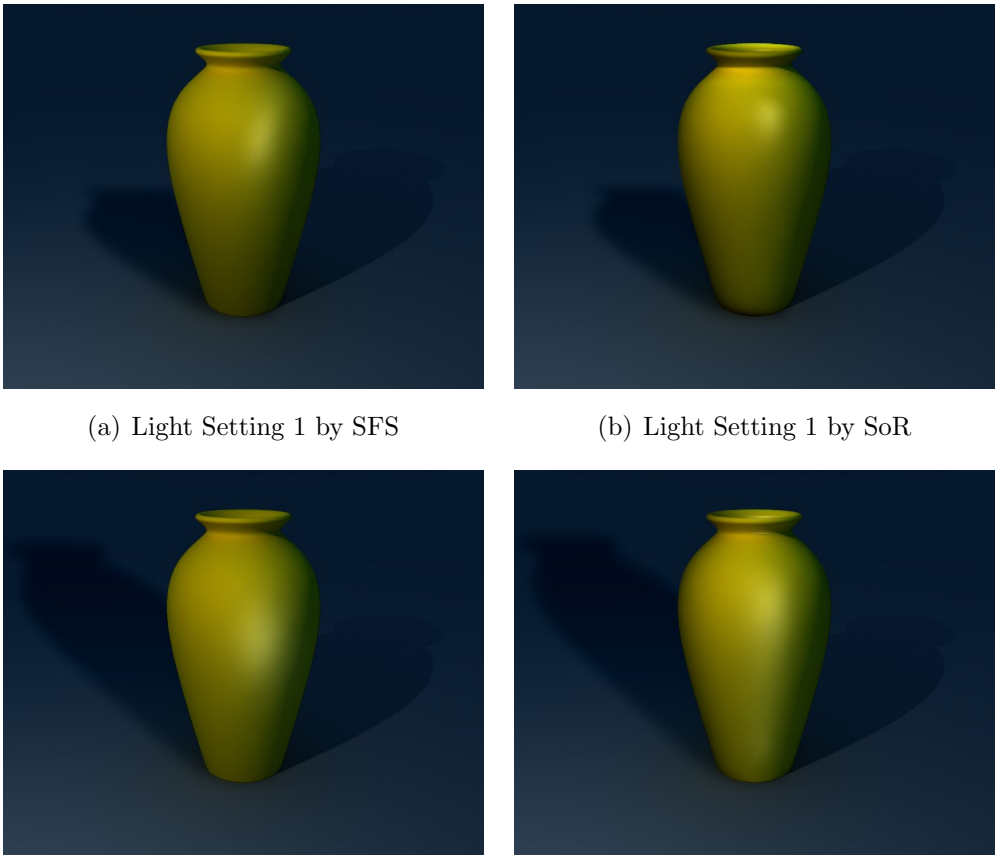
(a) Relit Image: Ground Truth

(b) Relit Image using SFS

(c) Relit Image using SoR

Figure 5.3: Change Light – Jar

The results of adding a new light to the image are quite good in the first light setting (Figure 5.4). The problems in the previous set of images are less visible because they are hidden by the original image. In the second setting, the highlight spot in the SFS case still appears wrong and affect the overal appearance of the image.



(a) Light Setting 1 by SFS

(b) Light Setting 1 by SoR

(c) Light Setting 2 by SFS

(d) Light Setting 2 by SoR

Figure 5.4: Add Light – Jar. Light setting 1 has a new light at $\phi = 30°$ and $\theta = 60°$. Light setting 2 has a new light at $\phi = 30°$ and $\theta = 20°$.

5.2 Two Apples

The second image (Figure 5.5) has two apples placed on the ground plane. Both SFS and SoR are used to generate relit images.

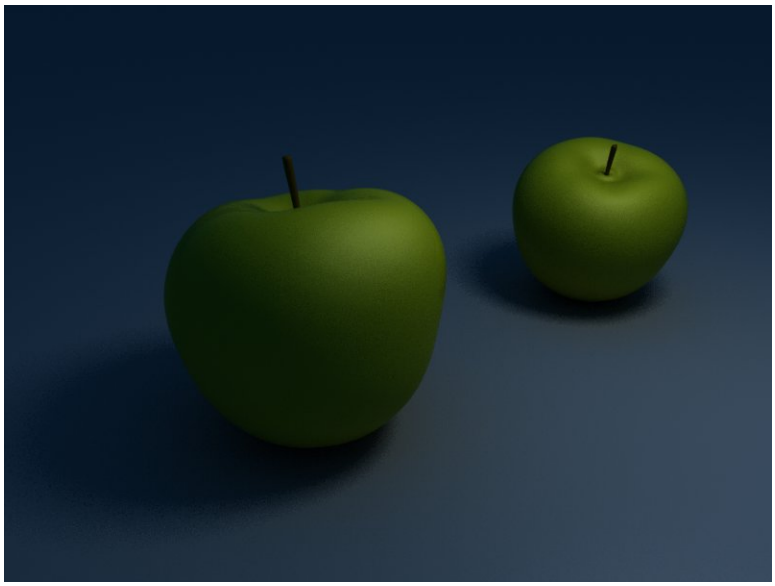
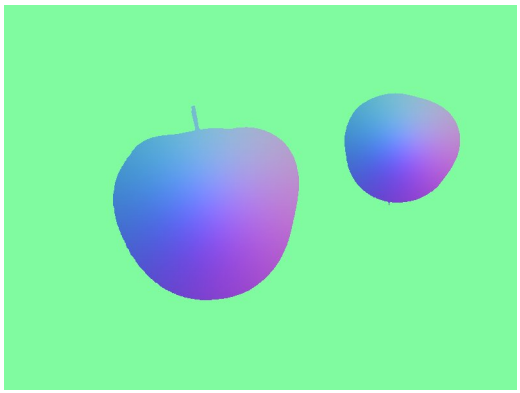


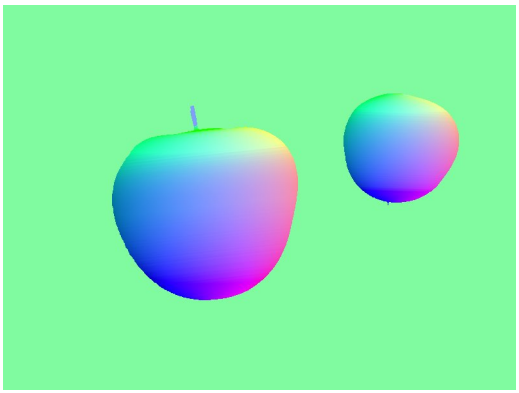
Figure 5.5: Original Image – Apples

The normal maps of each method are shown in Figure 5.6. As in Figure 5.2, the normal map generated using SFS is very smooth, but with incorrect values on the boundary. The normal map of SoR shows that the estimated symmetry axis of the small apple is not accurate, causing the normals to deviate from their correct values.

The results of changing the existing light are shown in Figure 5.7, and they are quite similar. Images generated using SFS method have poorly defined self shadows because the normals have small xy components along the boundary. In the SoR case the inaccuracy of the normal map of the small apple has little effect on the quality of the resulting images.

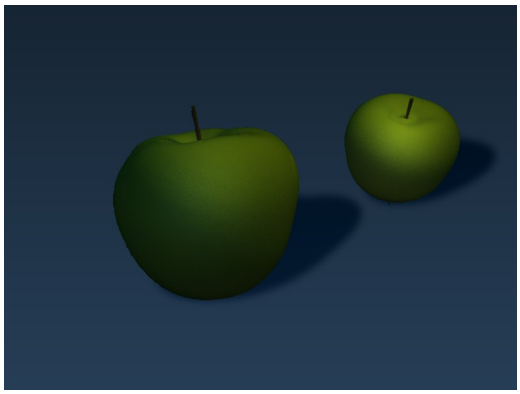


(a) Normal Map by SFS

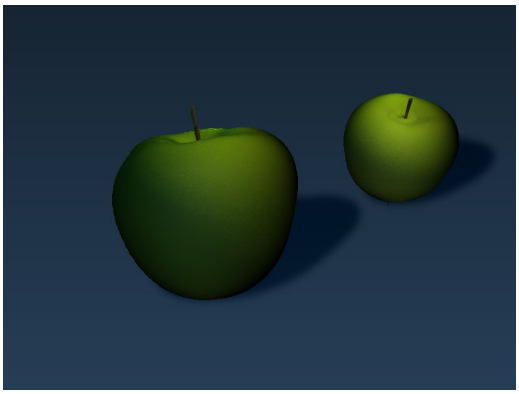


(b) Normal Map by SoR

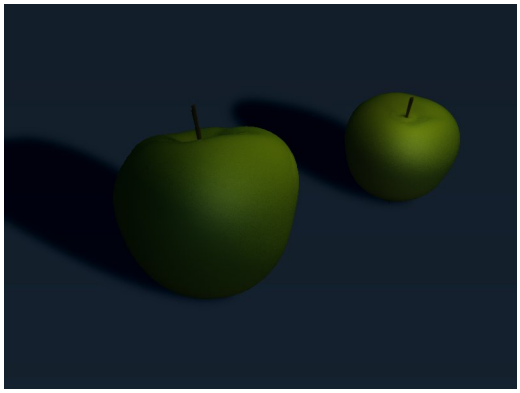
Figure 5.6: Normal Map Comparison – Apples



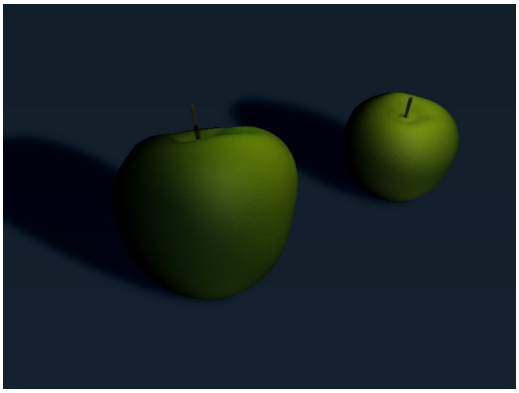
(a) Light Setting 1 by SFS



(b) Light Setting 1 by SoR



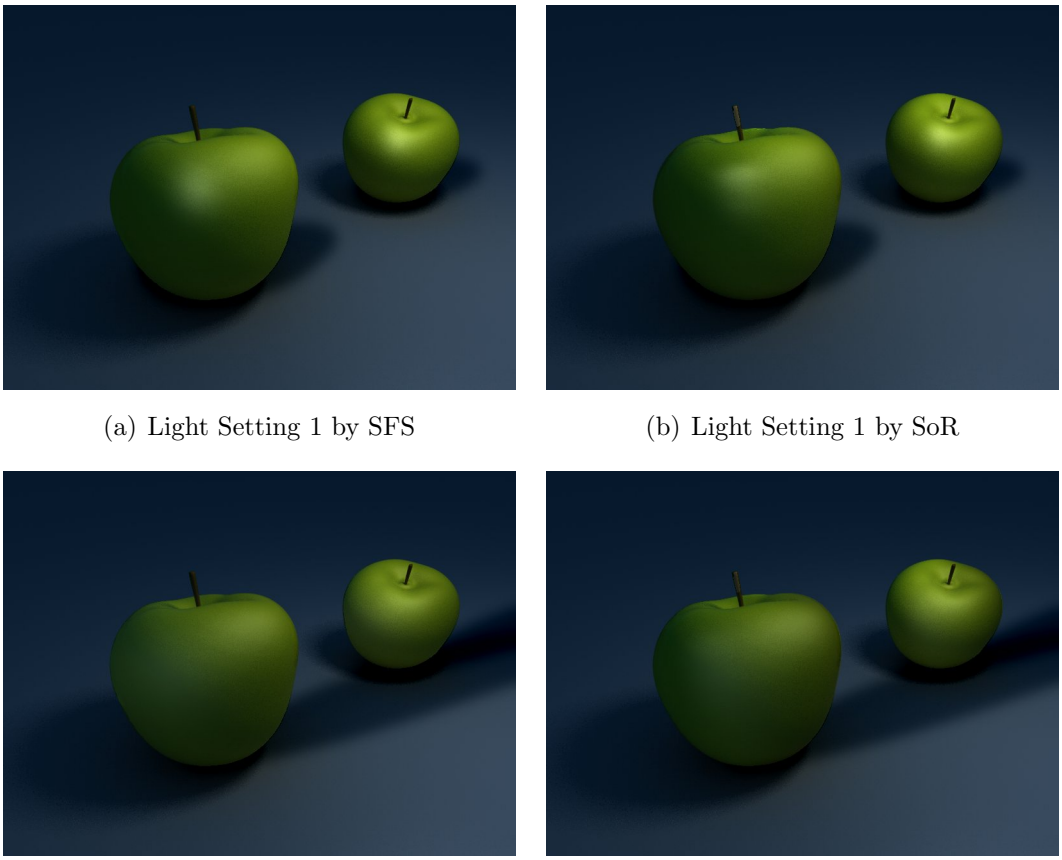
(c) Light Setting 2 by SFS



(d) Light Setting 2 by SoR

Figure 5.7: Change Light – Apples

In the set of images shown in Figure 5.8, the specular hardness has been increased to create a more shiny look. The results of two methods are similar.



(a) Light Setting 1 by SFS

(b) Light Setting 1 by SoR

(c) Light Setting 2 by SFS

(d) Light Setting 2 by SoR

Figure 5.8: Add Light – Apples

5.3 Three Vases

The third test image (Figure 5.9) contains three vases of different shapes. Again we run our relighting algorithm using both SFS and SoR.

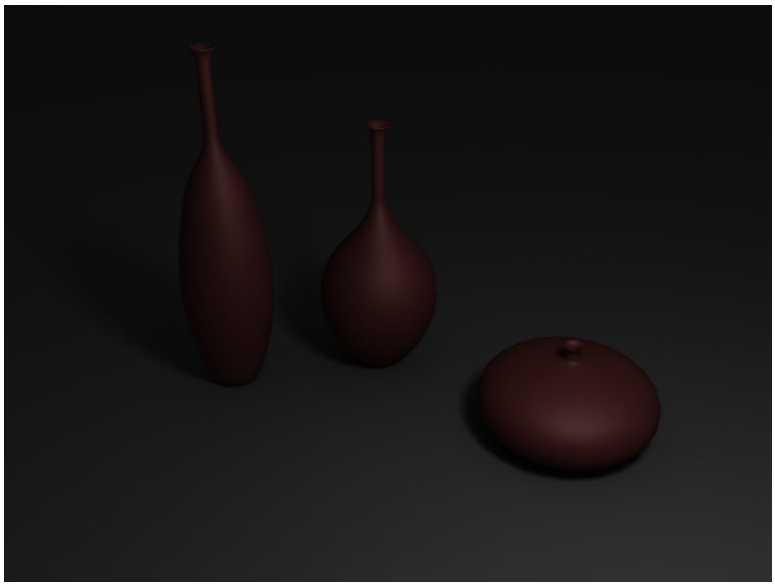
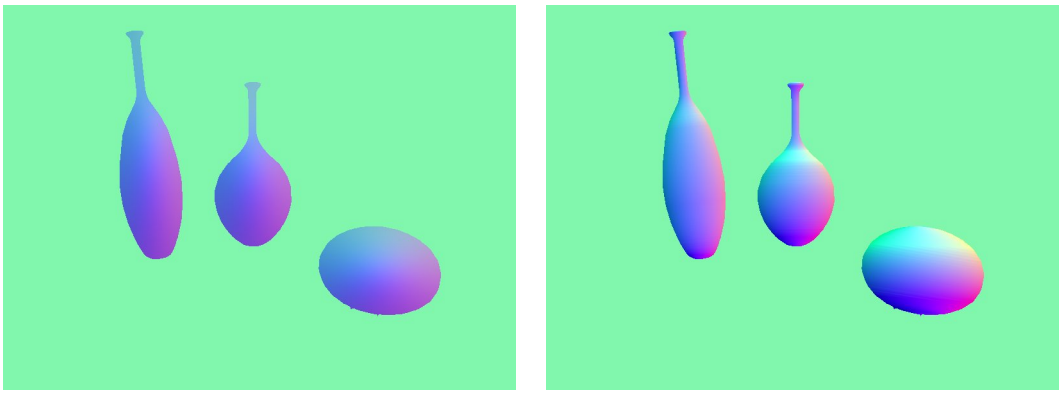


Figure 5.9: Original Image – Vases

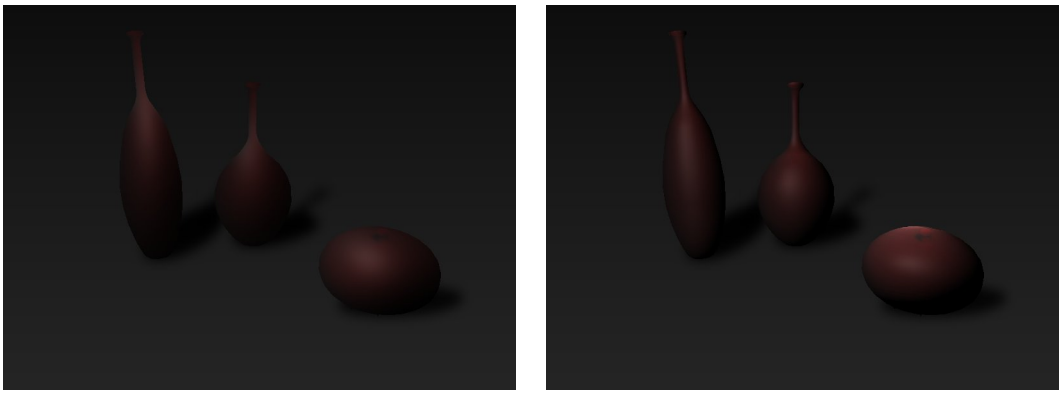
In this case, the disadvantage of SFS again reveals itself. It approximates the normals of the vase bodies, but fails to get the details in the necks. The SoR method has no such drawback. The normal map is a good description of the vase surfaces.

Because of the faulty normal map, the relighting results of SFS method in both light settings do not look good. The vases appear flat, especially in the first light setting. The results of the SoR method, on the contrary, are visually pleasing (Figure 5.11).



(a) Normal Map by SFS (b) Normal Map by SoR

Figure 5.10: Normal Map Comparison – Vases

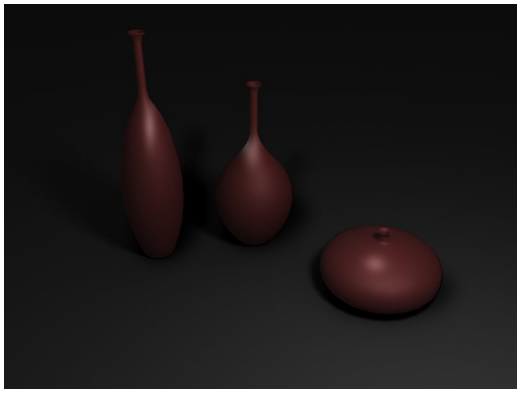


(a) Light Setting 1 by SFS (b) Light Setting 1 by SoR

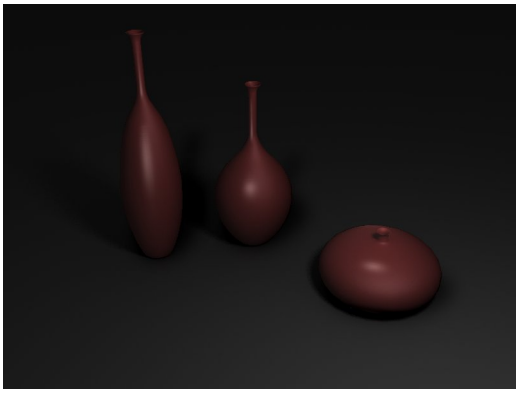
(c) Light Setting 2 by SFS (d) Light Setting 2 by SoR

Figure 5.11: Change Light – Vases

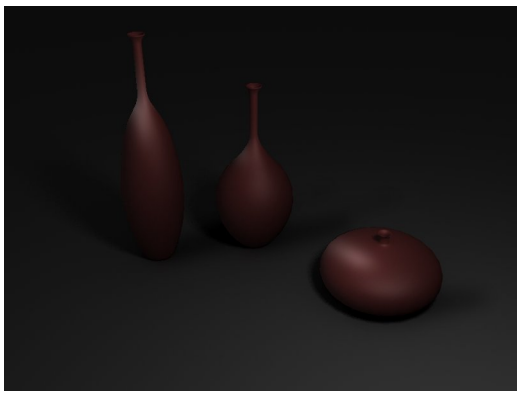
The results in Figure 5.12 when a new light is added are similar to the relighting results in Figure 5.12. The SFS method fails to show the curvature of the vases, and the specular highlights look wrong. Images generated by SoR method, on the other hand, look pretty good. Increasing the specular hardness gives the final images a nice touch.



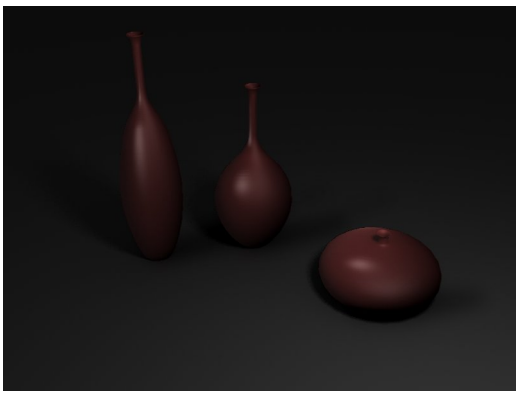
(a) Light Setting 1 by SFS



(b) Light Setting 1 by SoR



(c) Light Setting 2 by SFS



(d) Light Setting 2 by SoR

Figure 5.12: Add Light – Vases

5.4 One Orange

The fourth image is an actual photograph of an orange on a textured grey ground. It is shown in Figure 5.13.

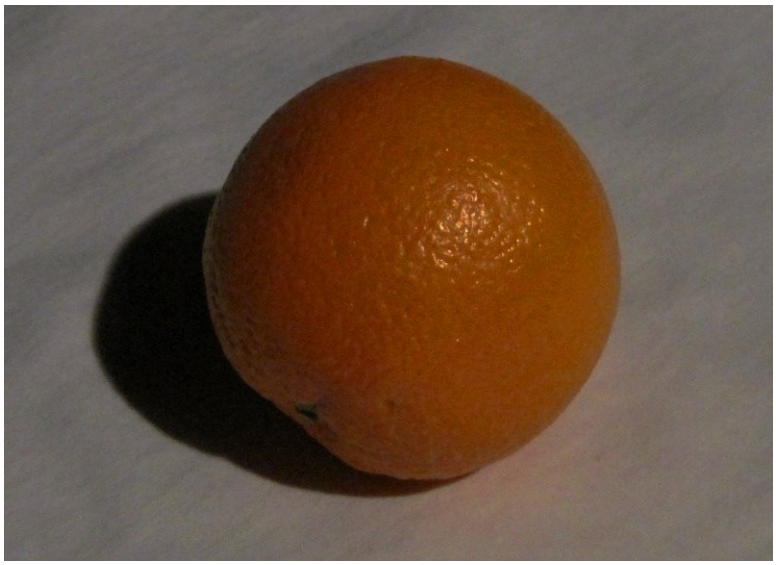
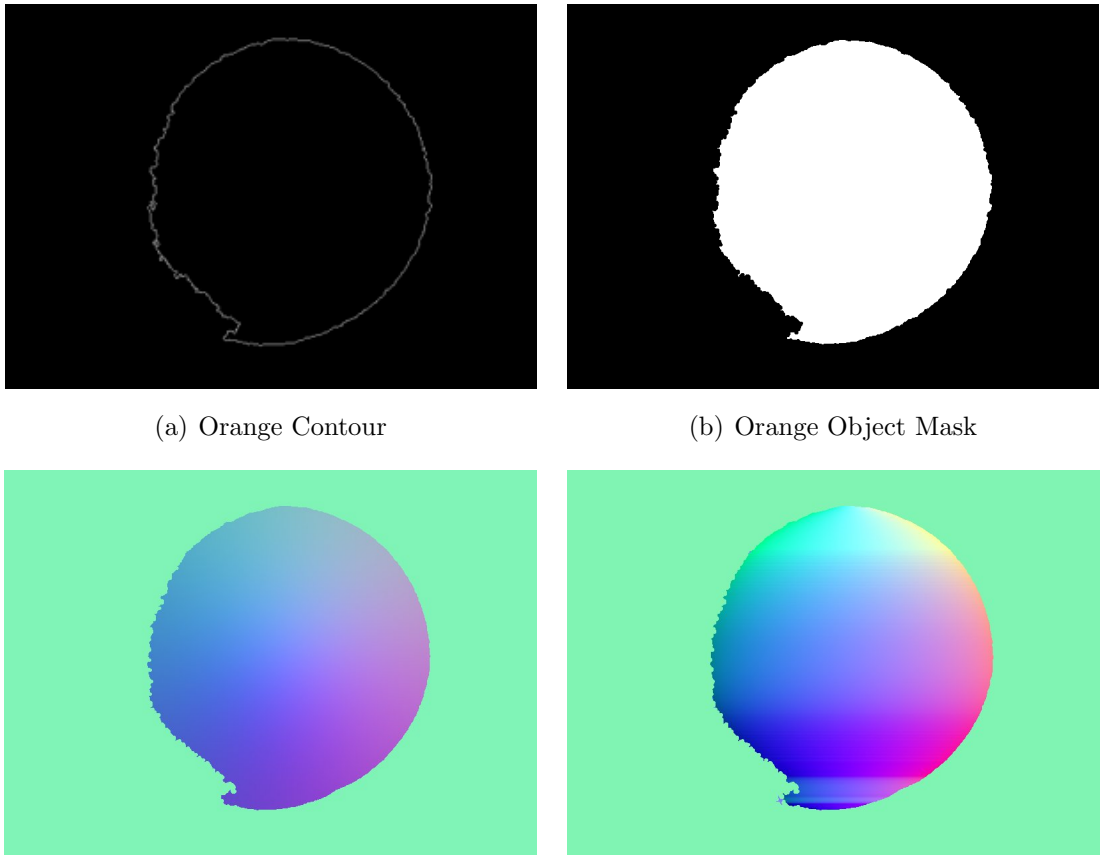


Figure 5.13: Original Image – Orange

In this case the contour detection step fails to get the correct contour of the orange. Because of colour bleeding onto the grey background, the final contour is noisy (Figure 5.14(a)), leading to normal maps with low quality (Figure 5.14(c)(d)).



(a) Orange Contour

(b) Orange Object Mask

(c) Normal Map by SFS

(d) Normal Map by SoR

Figure 5.14: Normal Map Comparison – Orange

The result of changing the original light, as shown in Figure 5.15, thus is unsatisfactory because the detected contour does not demonstrate the correct shape of an orange. The ground plane also loses its original texture.

Adding a new light is less sensible to such errors (Figure 5.16). Many errors are cancelled out by the original image. The method is able to produce shadow and specular as the light source changes, both of which are additional depth cues that improve the visual perception of the orange. In particular, the specular highlights, though exaggerated and too smooth for an orange, demonstrate the round shape of the orange more effectively.

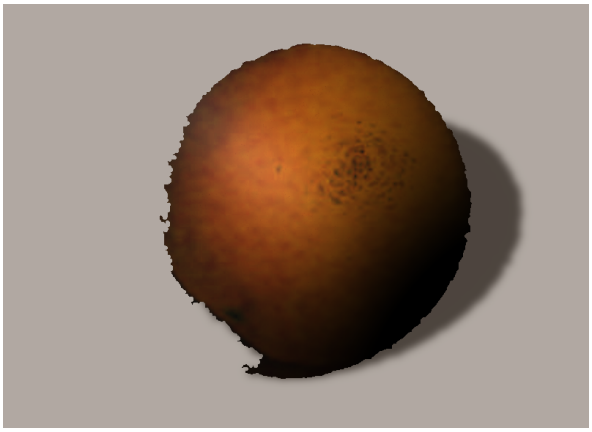
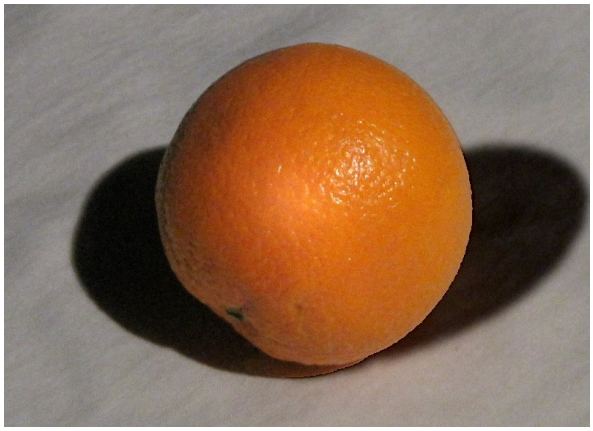
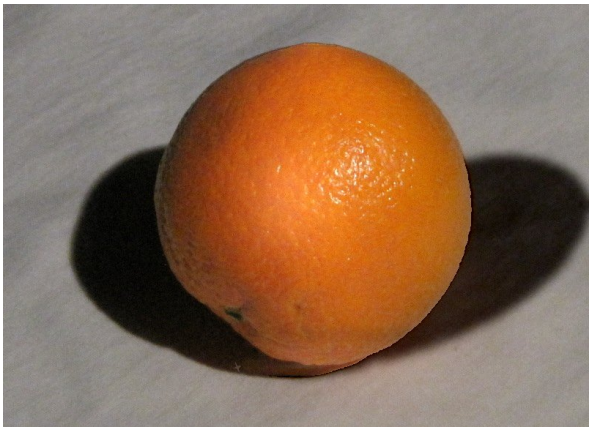


Figure 5.15: Change Light – Orange



(a) New Light Setting by SFS



(b) New Light Setting by SoR

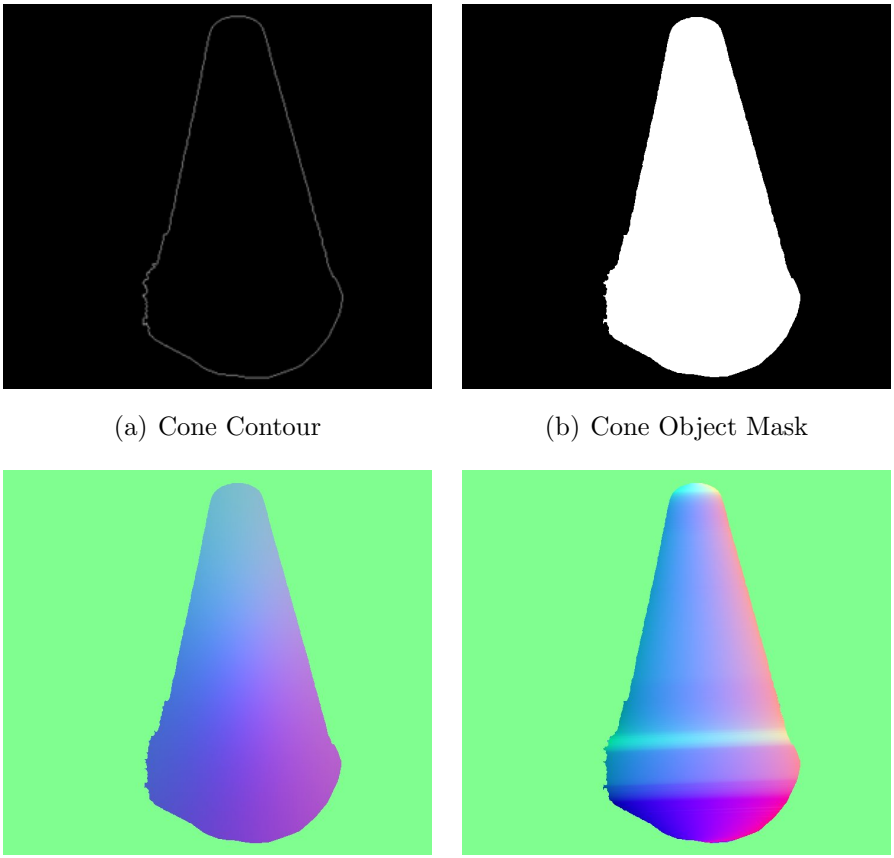
Figure 5.16: Add Light – Orange

5.5 One Construction Cone

The fifth test image is an actual photograph of a construction cone, posed on the same ground as the orange, shown in Figure 5.17.

Figure 5.17: Original Image – Construction Cone

The normal maps by SFS and SoR methods are shown in Figure 5.18. The contour detection cannot extract a precise representation of the cone silhouette; the left half of the contour is noisy (Figure 5.18(a)). The normal map generated using SFS is not affected by the noisy contour; the normals are still smooth. The normal map generated using SoR method, on the other hand, is not satisfactory, containing sudden changes of values in several places because samples are taken from noisy locations for Bézier curve fitting.



(a) Cone Contour (b) Cone Object Mask

(c) Normal Map by SFS (d) Normal Map by SoR

Figure 5.18: Normal Map Comparison – Construction Cone

Similar to the orange case, changing the original light does not yield good images (Figure 5.19). The ground texture is missing. The noisy parts of the contour are also very noticeable. The results of adding a light are shown in Figure 5.20. Again the inaccuracy in the object contour is hidden by the original image. The highlights in the SoR case is more convincing, even though the bottom of the cone has a spurious bump.

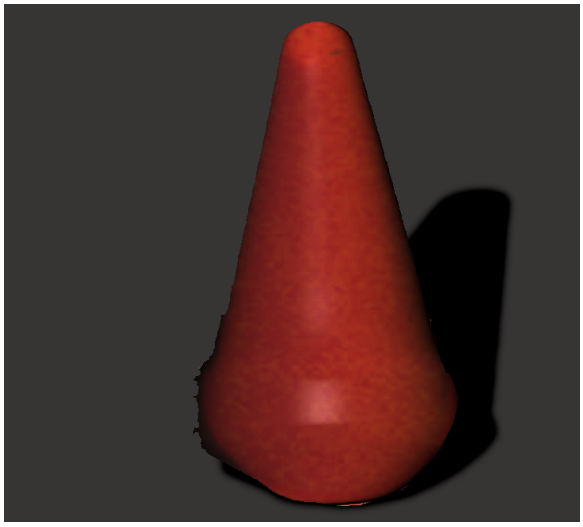


Figure 5.19: Change Light – Cone

(a) New Light Setting by SFS

(b) New Light Setting by SoR

Figure 5.20: Add Light – Construction Cone

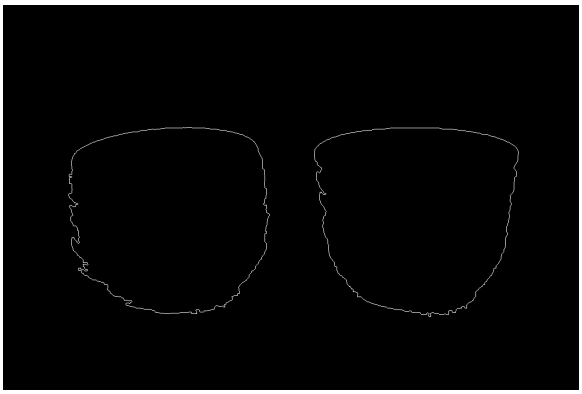
5.6 Two Cups

The last image we use is a real photograph of two ceramic cups. Unlike the orange and the construction cone, which were photographed by the author specifically as test cases, this photograph was found online from public domain. It is therefore typical of the random photographs to be relit, and at the same time difficult to relit well. Note that this photograph actually breaks two of our assumptions. The cups are not uniformly coloured and have small variations on their surface geometry. The original photograph also has at least two light sources. The algorithm, therefore, is expected to fail to a certain extent.

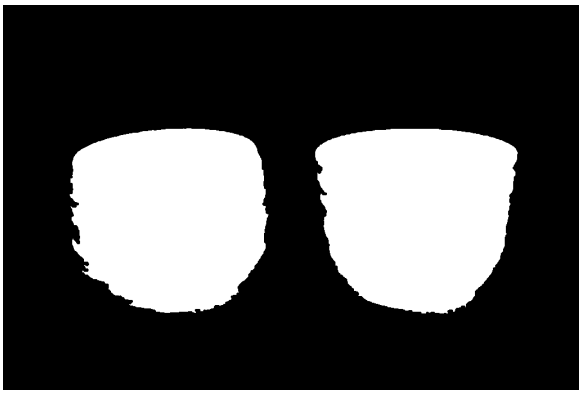
Figure 5.21: Original Image – Cups [24]

Both SFS and SoR were used to generate the normal maps (Figure 5.22). We encounter the problem of imprecise contour extraction again. While the contours respond to the real features of the cups, they also contain extra noises that are not a part of the shape.

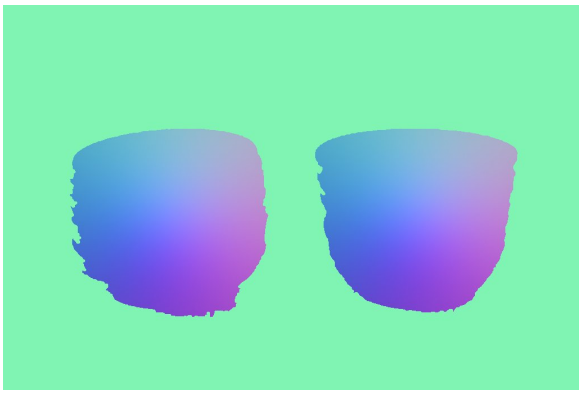
The algorithm is still capable of calculating light direction and plane orientation. Figure 5.23 shows the results of adding an additional light. The disadvantage of SFS method shows up once more. The highlight shapes do not fit to the actual surface shapes of the cups. The normal map by SoR method, on the other hand, is quite close to the actual shapes of the cups, so the relit image contains correct specular highlights which enhances the 3D appearance of the cups.



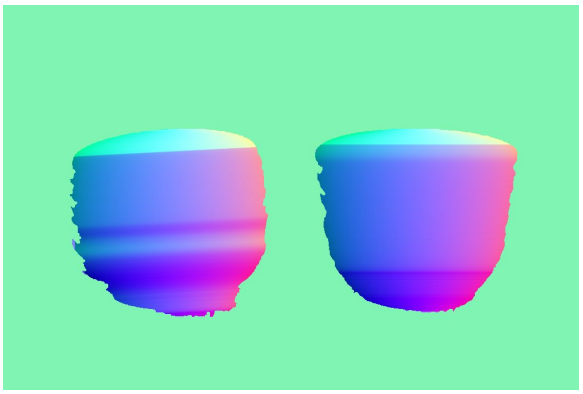
(a) Cup Contours



(b) Cup Object Masks



(c) Normal Map by SFS



(d) Normal Map by SoR

Figure 5.22: Normal Map Comparison – Cups

(a) Light Setting 1 by SFS

(b) Light Setting 1 by SoR

Figure 5.23: Add Light – Cups

5.7 Summary

For all synthetic images, the new method generates visually pleasing images for both relighting tasks, changing the original light and adding a new light. The contours are extracted cleanly, and the shadows are computed correctly. Using the SFS method to compute normal maps always gives us smooth results, but often with normals far from their true values. The SoR method can produce in rough surfaces. Normals can vary on short scales because noise inherent in contour extraction affects the Bezier approximation. However, normal maps generated using SoR describe the actual object surfaces more accurately.

The method produces less good results for the real photographs. The most serious problem lies not in the algorithm but in contour extraction. The contour detection method does not work well with real photographs and produces jagged outlines that are poor representations of the object silhouettes. The shape recovery step hence fails to build accurate surfaces around the contour given the jagged inputs. However, in areas far away from the contour, accurate normals are constructed and highlights are generated in a visually convincing manner. Because several parts of the algorithm use orthogonal projection, the resulting normals are more exaggerated compared to their true values. Nonetheless the larger normal values add additional depth cues and make the relit images appear more three-dimensional than the original ones.

From the tests and our analysis, it is possible to answer the question: what is the most difficult aspect of the relighting problem? The most difficult aspect, regardless of whether contours extracted are precise (in the synthetic cases) or only poorly defined (in the real photograph cases), is to reconstruct object surfaces correctly. A badly constructed $2\frac{1}{2}$D normal map can give false impression of the surface shapes and degrade the visual appearance of the relit images. Hence the key to creating an visually pleasing relit image is to build an accurate representation of the object surface.

Chapter 6

Discussions

6.1 Conclusions

To change illumination in a single input image is not an easy endeavour. There are too many unknown factors such as scene geometry, materials, and illumination condition. The method developed in this thesis makes use of the 2D contours, shadows, and highlights presented in the input to derive a $2\frac{1}{2}$D representation of the image regardless of many unknowns. It is able to add and change lighting for a subset of images and produce visually convincing outputs.

In the course of developing this method, we can see a significant use of 2D information, especially the object contours and shadow map. The object contours are used in almost every step in the entire algorithm and are useful for both light recovery and shape recovery. Shadow is an additional cue that is utilized for verifying the correctness of light recovery results and identifying background shape. It is also an essential light position indicator. These two elements are fundamental for the algorithm, and prove the point that we can learn a lot about the scene from 2D information.

This thesis also explores the effectiveness of surfaces of revolution in determining object surface shapes. While symmetry and surfaces of revolution are frequently studied in computer vision, it is not commonly used in computer graphics for image-based rendering. Reconstructing 3D shapes from a single 2D image is impossible without any prior knowledge of the scene. Cues such as symmetry or revolution can greatly help us build the

geometry with or without other cues.

6.2 Limitations and Future Work

We also see many limitations of our algorithm and there are various ways to improve it. Possible improvements include but not are limited to:

- Better Contour Detection
  The contour detection method used in this thesis cannot retrieve good results from real images. Since the algorithm is heavily based on a precise description of the object silhouette, a better contour detection algorithm is essential.

- Better curve sampling for Bézier approximation
  To generate piece-wise cubic Bézier curve, we sample the original contour to get a collection of points for approximation. However it is difficult to determine how many samples to take for each contour, and what gradient threshold should be used. A better curve sampling routine can improve the results of Bézier approximation, leading to more accurate surface normals.

- Better SFS algorithm
  Although we modified Daniel and Durou's SFS algorithm to incorporate light direction, the method is still unable to generate good normal maps for surfaces like vases or mugs. It would be great if a more accurate SFS method can be adopted.

- Normal refinement
  A normal refinement method similar to the one used by Okabe et al. [45] can be adopted to make the object appearance (such as an orange) more convincing.

- Diffuse-only image
  We use the MSF image in our algorithm instead of a diffuse-only image, the reason being that previous highlight removal methods cannot produce very smooth images. However, such choice also result in loss of hue and affect the final relit images. An algorithm producing good diffuse-only images would eliminate this problem.

- Perspective in SFS and SoR
  We do not consider perspective projection in either of our shape recovery methods. Previous studies such as [49], [64], and [68] work on perspective SFS and surfaces of revolution under perspective projection. Adopting such methods can improve the quality of reconstructed shapes.

In addition to the above improvements, there are several extensions that can be considered to expand the use of this algorithm.

- Extend to multiple inputs
  With digital photography it is far more convenient to take multiple shots of a scene. If the photographs have spatial correspondence, other reconstruction methods that take multiple inputs can be used to recover geometry, producing better surface shapes. If there is no spatial correspondence among the inputs, we can always fall back on this new method for shape recovery.

- Handle occluded objects
  Our method assumes that objects in the image do not occlude each other. This assumption can be loosened if the algorithm is extended to use LDI or other means to separate the layers of objects. We can even add more flexibility by modifying illumination effects for the a few selected objects. This will create some useful applications in areas such as image compositing.

- Combine with perspective reconstruction methods
  We mainly focus on curved objects in this thesis. In real life there are many man-made objects that have straight edges, such as books and boxes. Combined with previous method such as [22], our method can be extended to handle objects with both orthogonal planes and curved surfaces. The additional perspective information from box-like objects can help refine a lot of the approximations in the algorithm.

Appendix A

User Interface

A simple user interface is designed to make the relighting process easier. Figure A.1 shows a screenshot of the user interface.

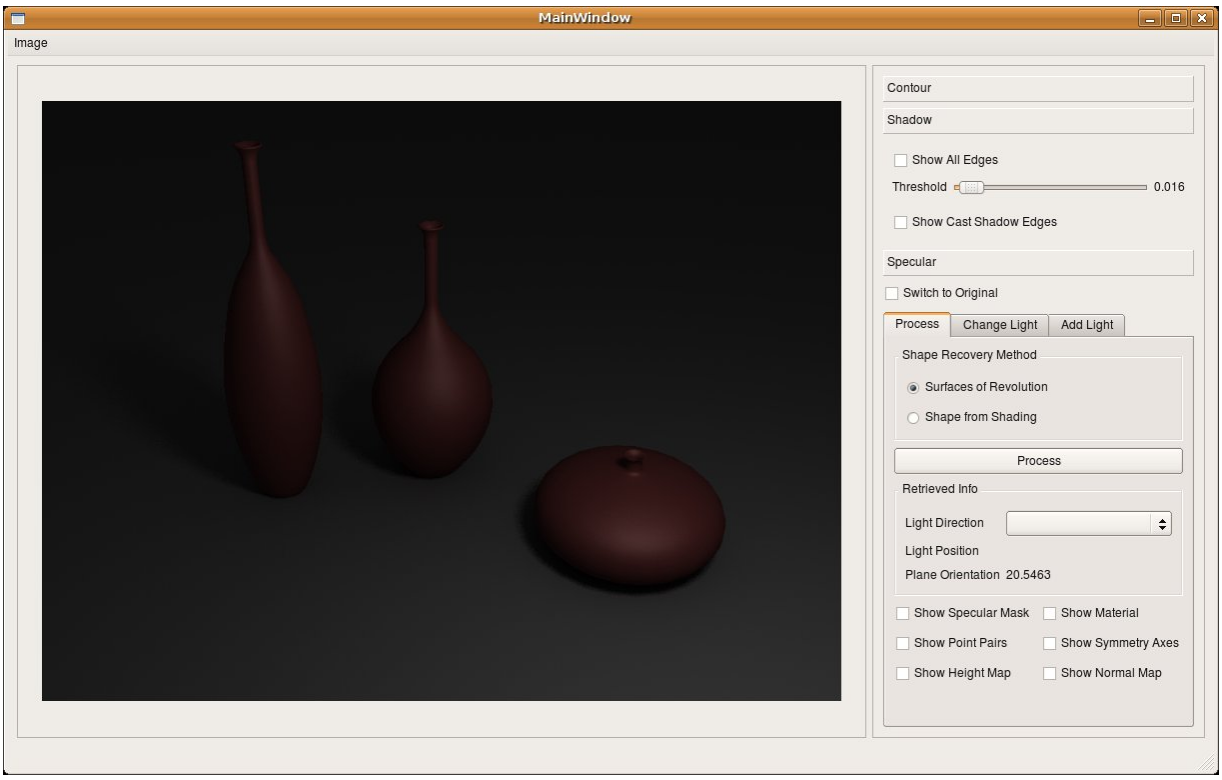
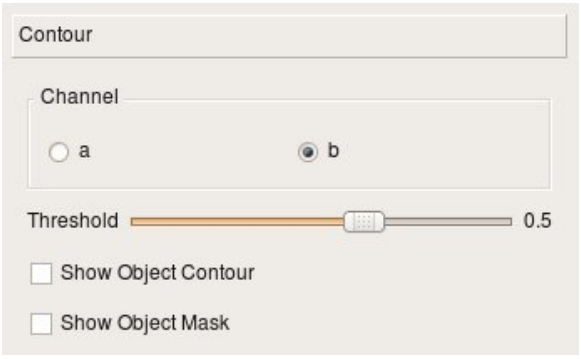
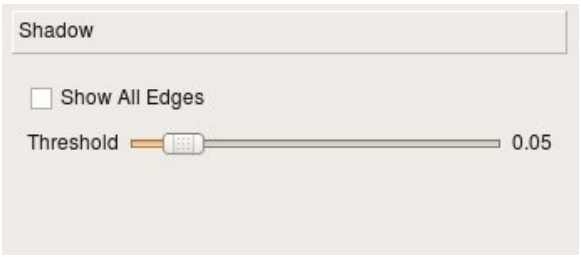


Figure A.1: Screenshot of the User Interface

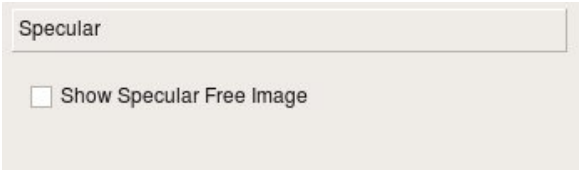
Several controls are used to adjust the parameters for the preprocessing stage (Figure A.2. The contour controls give users the freedom to choose the channel for contour detection and adjust object contours using a parameter. Users can also tune the cast shadow map by adjusting the threshold. The checkboxes can be toggled to display the original image or the results from the pre-processing stages.



(a) Contour Controls



(b) Shadow Controls



(c) Specular Controls

Figure A.2: Preprocessing Stage Controls

Before the processing stage starts, users need to choose the surface recovery method, being either surfaces of revolution or shape from shading (Figure A.3). The computation starts after users click the process button, and the retrieved information is displayed, such as light direction, light position, and plane orientation. Users can also toggle all the checkboxes to view different masks and maps.

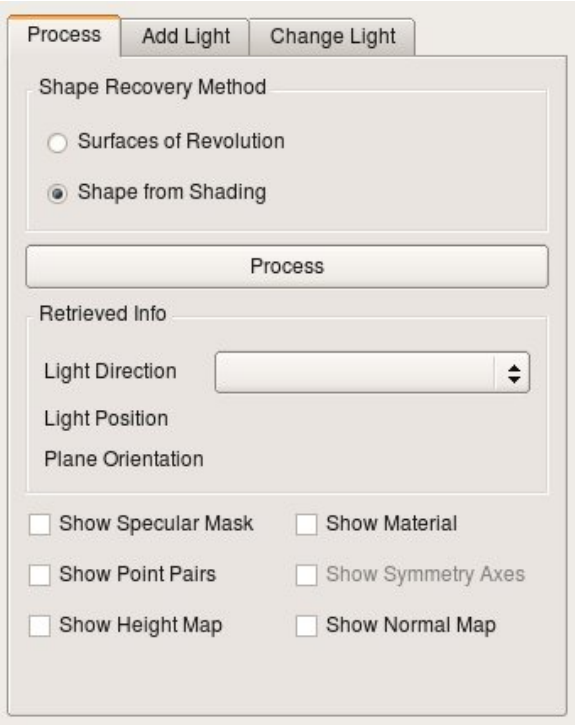
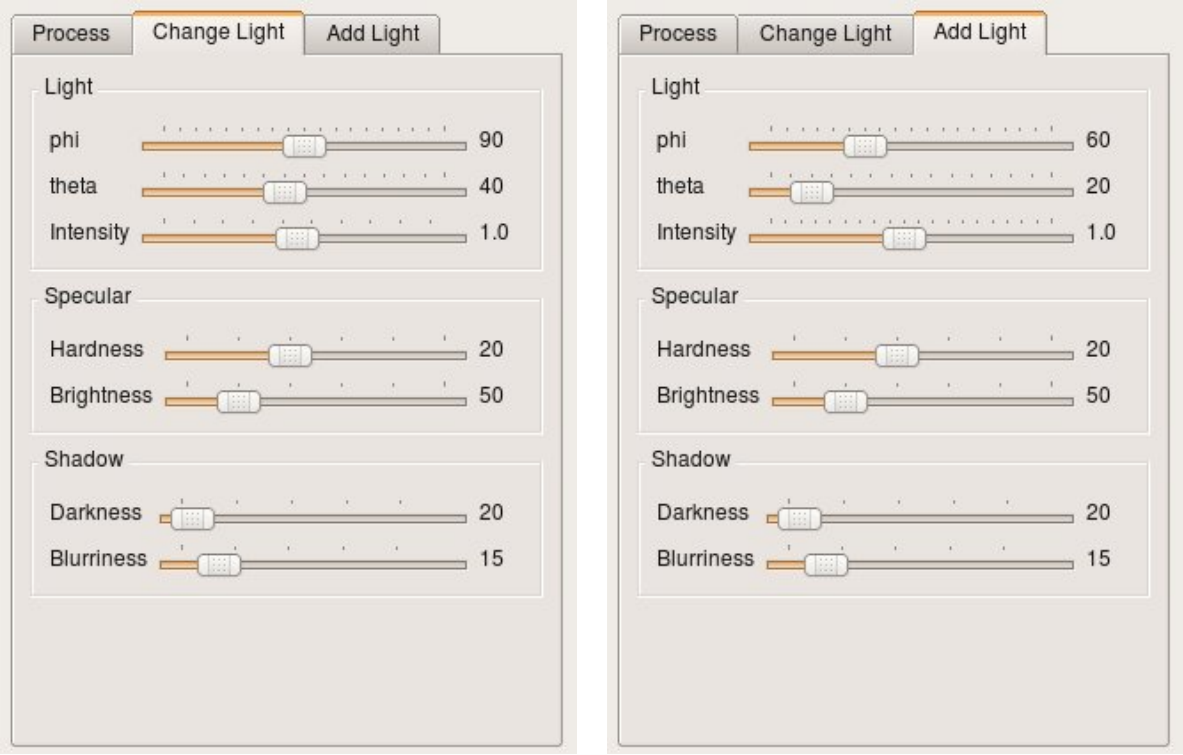


Figure A.3: Processing Stage Controls

Users have a wider range of controls are given for light modifications. The interfaces for changing light and adding light are similar, as shown in Figure A.4.

Light direction is controlled by two angles ϕ and θ. We limit ϕ to be 0° to 180° and θ to be 5° to 85° because human beings usually perceive light sources to be above the ground [2]. User can also adjust the light intensity.

The specular is controlled by two parameters: hardness and brightness. These two parameters are taken by the algorithm to calculate the correct highlight. Shadows can also be adjusted by their darkness and blurriness by users.



(a) Change Light Controls

(b) Add Light Controls

Figure A.4: Light Modification Controls

Bibliography

[1] http://en.wikipedia.org/wiki/Lens_(optics). xvii, 35

[2] Wendy J. Adams. A common light-prior for visual search, shape, and reflectance judgments. In *Journal of Vision*, 2007. 82

[3] Michael Baxandall. *Shadows and Enlightenment*. Yale University Press, 1995. 22

[4] Samuel Boivin, Samuel Boivin, Andr Gagalowicz, Andr Gagalowicz, and Projet Mirages. Image-based rendering from a single image. In *In Proceedings of IS&T CGIV*, 2001. 8

[5] Max Chen. Interactive specification and acquisition of depth from single images. Master's thesis, Massachusetts Institute of Technology, 2001. 14

[6] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 279–288, New York, NY, USA, 1993. ACM. 6

[7] A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. *Int. J. Comput. Vision*, 40(2):123–148, 2000. 8, 15

[8] Pascal Daniel and Jean-Denis Durou. Creation of real images which are valid for the assumptions made in shape from shading. In *ICIAP '99: Proceedings of the 10th International Conference on Image Analysis and Processing*, page 418, Washington, DC, USA, 1999. IEEE Computer Society. 16, 41, 42, 43

[9] Paul Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography.

In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 189–198, New York, NY, USA, 1998. ACM. 7

[10] Erick Delage, Honglak Lee, and Andrew Y. Ng. Automatic single-image 3d reconstructions of indoor manhattan world scenes. In *In ISRR*, 2005.

[11] Jean-Denis Durou, Maurizio Falcone, and Manuela Sagona. Numerical methods for shape-from-shading: A new survey with benchmarks. *Computer Vision and Image Understanding*, 109(1):22 – 43, 2008. 16, 24, 41

[12] Ralph M. Evans. *Eye, Film, and Camera in Color Photography.* John Wiley & Sons, Inc., 1959. 8

[13] David A. Forsyth and Jean Ponce. *Computer Vision: A Modern Approach.* Prentice Hall, 2003. 8

[14] Elodie Fourquet. Learning about shadows from artists. *Computational Aesthetics (in press)*, 2010. 51

[15] Robert T. Frankot and Rama Chellappa. A method for enforcing integrability in shape from shading algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(4):439–451, 1988. 16

[16] Theo Gevers and Arnold W. M. Smeulders. Color based object recognition. In *ICIAP '97: Proceedings of the 9th International Conference on Image Analysis and Processing-Volume I*, pages 319–326, London, UK, 1997. Springer-Verlag. 21

[17] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA, 1996. ACM. 6

[18] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2003. 8

[19] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Automatic photo pop-up. *ACM Trans. Graph.*, 24(3):577–584, 2005. 8

[20] Berthold K. P. Horn and Michael J. Brooks, editors. *Shape from Shading*. The MIT Press, 1989. 8, 9, 16

[21] Youichi Horry, Ken-Ichi Anjyo, and Kiyoshi Arai. Tour into the picture: using a spidery mesh interface to make animation from a single image. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 225–232, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co. 8, 15

[22] Jingyuan Huang and Bill Cowan. Simple 3d reconstruction of single indoor image with perspective cues. In *CRV '09: Proceedings of the 2009 Canadian Conference on Computer and Robot Vision*, pages 140–147, Washington, DC, USA, 2009. IEEE Computer Society. 8, 15, 36, 77

[23] Caixia Jiang and Matthew O. Ward. Shadow segmentation and classification in a constrained environment. *CVGIP: Image Underst.*, 59(2):213–225, 1994. 23

[24] Denise Joyal. http://www.flickr.com/photos/niso30/3941889234/in/set-421558/. xix, 71

[25] Erum A. Khan and Erik Reinhard. A survey of color spaces for shadow identification. In *APGV '04: Proceedings of the 1st Symposium on Applied perception in graphics and visualization*, pages 160–160, New York, NY, USA, 2004. ACM. 21

[26] David C. Knill, Pascal Mamassian, and Daniel Kersten. Geometry of shadows. In *Optical Society of America*, volume 14, 1997. 22

[27] Sumanta Pattanaik Kvin Boulanger, Kadi Bouatouch. Atip: A tool for 3d navigation inside a single image with automatic camera calibration. In *EG UK Theory and Practice of Computer Graphics*, 2006. 15

[28] K. M. Lee and C. C. J. Kuo. Shape from shading with a linear triangular element surface model. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(8):815–822, 1993. 16

[29] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, New York, NY, USA, 1996. ACM. 6

[30] Wai Ho Li, Alan M. Zhang, and Lindsay Kleeman. Real time detection and segmentation of reflectionally symmetric objects in digial images. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006. 45

[31] Yuanzhen Li, Stephen Lin, Sing Bing Kang, Hanqing Lu, and Heung-Yeung Shum. Single-image reflectance estimation for relighting by iterative soft grouping. In *PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications*, page 483, Washington, DC, USA, 2002. IEEE Computer Society. 8

[32] Yuanzhen Li, Stephen Lin, Hanqing Lu, and Heung-Yeung Shum. Multiple-cue illumination estimation in textured scenes. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 1366, Washington, DC, USA, 2003. IEEE Computer Society. 13, 21, 22

[33] Claus B. Madsen and Rune Laursen. Image relighting: Getting the sun to set in an image taken at noon. In *13th Danish Conference on Pattern Recognition and Image Analysis*, pages 13–20, 2004. 8

[34] Dhruv Mahajan, Ravi Ramamoorthi, and Brian Curless. A theory of spherical harmonic identities for brdf lighting transfer and image consistency. In *In European Conference on Computer Vision*, pages 41–55, 2006. 8

[35] Pascal Mamassian, David C. Knill, and Daniel Kersten. The perception of cast shadows. In *Trends in Cognitive Sciences*, volume 2, pages 288–295, 1998. 22

[36] William R. Mark, Leonard McMillan, and Gary Bishop. Post-rendering 3d warping. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 7–ff., New York, NY, USA, 1997. ACM. 7

[37] David Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information*. W. H. Freeman, 1983. 21

[38] Stephen R. Marschner and Donald P. Greenberg. Inverse lighting for photography. In *Fifth Color Imaging Conference*, pages 262–265, 1997. 8

[39] Dave McKane. http://www.flickr.com/photos/davemckane/1813362201/in/photostream/. xvii, 2

[40] Leonard McMillan, Jr. *An image-based approach to three-dimensional computer graphics*. PhD thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1997. 7

[41] Ross R. Middlemiss, John L. Marks, and James R. Smart. *Analytic Geometry*. McGraw-Hill, 1968. 45

[42] Shree K. Nayar, Xi-Sheng Fang, and Terrance Boult. Separation of reflection components using color and polarization. *Int. J. Comput. Vision*, 21(3):163–186, 1997. 24

[43] J. Farley Normana, Young lim Leeb, Flip Phillipsc, Hideko F. Normana, L. RaShae Jenningsa, and T. Ryan McBridea. The perception of 3-d shape from shadows cast onto curved surfaces. In *Journal of Vision*, volume 9, 2009. 22

[44] Byong Mok Oh, Max Chen, Julie Dorsey, and Frédo Durand. Image-based modeling and photo editing. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 433–442, New York, NY, USA, 2001. ACM. 14

[45] M. Okabe, G. Zeng, Y. Matsushita, T. Igarashi, L. Quan, and H.-Y. Shum. Single-view relighting with normal map painting. In *Proceedings of Pacific Graphics*, pages 27 – 34, 2006. 15, 76

[46] Alex P. Pentland. Finding the illuminant direction. In *J. Optical Soc. Am.*, pages 448–455, 1982. 8, 12, 36

[47] Alex P. Pentland. *Linear shape from shading*, pages 29–38. Jones and Bartlett Publishers, Inc., , USA, 1992. 17

[48] Pierre Poulin and Alain Fournier. Lights from highlights and shadows. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 31–38, New York, NY, USA, 1992. ACM. 13

[49] E. Prados and O. Faugeras. "perspective shape from shading" and viscosity solutions. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 826, Washington, DC, USA, 2003. IEEE Computer Society. 77

[50] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 117–128, New York, NY, USA, 2001. ACM. 8

[51] E. Salvador, A. Cavallaro, and T. Ebrahimi. Shadow identification and classification using invariant color models. In *ICASSP '01: Proceedings of the Acoustics, Speech, and Signal Processing, 2001. on IEEE International Conference*, pages 1545–1548, Washington, DC, USA, 2001. IEEE Computer Society. 22

[52] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Illumination from shadows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(3):290–300, 2003. 13, 21

[53] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. 3-d depth reconstruction from a single still image. ijcv. *International Journal of Computer Vision (IJCV)*, 2007. 8

[54] Philip J. Schneider. *An algorithm for automatically fitting digitized curves*, pages 612–626. Academic Press Professional, Inc., San Diego, CA, USA, 1990. 46

[55] Chris Schoeneman, Julie Dorsey, Brian Smits, James Arvo, and Donald Greenberg. Painting with light. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 143–146, New York, NY, USA, 1993. ACM. 8

[56] Jonathan Shade, Steven Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242, New York, NY, USA, 1998. ACM. 7

[57] Hui-Liang Shen, Hong-Gang Zhang, Si-Jie Shao, and John H. Xin. Chromaticity-based separation of reflection components in a single image. *Pattern Recogn.*, 41(8):2461–2469, 2008. 25

[58] Heung-Yeung Shum, Shing-Chow Chan, and Sing Bing Kang. *Image-Based Rendering*. Springer, 2007. 6

[59] Ping sing Tsai and Mubarak Shah. Shape from shading using linear approximation. *Image and Vision Computing*, 12:487–498, 1994. 17, 41

[60] J. B. Sykes, editor. *The Concise Oxford English Dictionary*. Oxford University Press, USA, 1982. 1

[61] Robby T. Tan and Katsushi Ikeuchi. Separating reflection components of textured surfaces using a single image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(2):178–193, 2005. 24, 25

[62] Robby T. Tan, Ko Nishino, and Katsushi Ikeuchi. Illumination chromaticity estimation using inverse-intensity chromaticity space. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2003*. IEEE Computer Society, 2003. 25

[63] Robby T. Tan, Ko Nishino, and Katsushi Ikeuchi. Separating reflection components based on chromaticity and noise analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(10):1373–1379, 2004. 25

[64] Ariel Tankus, Nir Sochen, and Yehezkel Yeshurun. Shape-from-shading under perspective projection. *Int. J. Comput. Vision*, 63(1):21–43, 2005. 77

[65] Yang Wang and Dimitris Samaras. Estimation of multiple directional illuminants from a single image. *Image Vision Comput.*, 26(9):1179–1195, 2008. 8

[66] E. K. Warrington and A. M. Taylor. Two categorical stages of object recognition. In *Perception*, volume 7, 1978. 21

[67] T. Werner, R. D. Hersch, and V. Hlavac. Rendering real-world objects using view interpolation. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 957, Washington, DC, USA, 1995. IEEE Computer Society. 6

[68] Kwan-Yee K. Wong, Paulo R. S. Mendona, and Robert Cipolla. Reconstruction of surfaces of revolution from single uncalibrated views. In *Proc. British Machine Vision Conference 2002*, pages 93–102, 2002. 77

[69] Yibing Yang and Alan Yuillie. Sources from shading. In *Computer Vision and Pattern Recognition*, pages 534–539, Maui, HI, USA, 1991. 12, 27

[70] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: recovering reflectance models of real scenes from photographs. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive*

techniques, pages 215–224, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co. 8

[71] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape from shading: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(8):690–706, 1999. 16, 24, 41

[72] Yufei Zhang and Yee-Hong Yang. Multiple illuminant direction detection with application to image synthesis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(8):915–920, 2001. 8

[73] Qinfen Zheng and Rama Chellappa. Estimation of illuminant direction, albedo, and shape from shading. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(7):680–702, 1991. 8, 12