Automatic Lighting Design using a Perceptual Quality Metric

A thesis submitted in fulfillment
of the requirements for the degree of
Master of Science

by

Ram Shacked

supervised by
Dr. Dani Lischinski

School of Computer Science and Engineering
The Hebrew University of Jerusalem
Jerusalem, Israel

February, 2001

Contents

List of Figures

Abstract

Lighting has a crucial impact on the appearance of 3D objects and on the ability of an image to communicate information about a 3D scene to a human observer. This work presents a new automatic lighting design approach for comprehensible rendering of 3D objects. Given a geometric model of a 3D object or scene, the material properties of the surfaces in the model, and the desired viewing parameters, our approach automatically determines the values of various lighting parameters by optimizing a perception-based image quality objective function. This objective function is designed to quantify the extent to which an image of a 3D scene succeeds in communicating scene information, such as the 3D shapes of the objects, fine geometric details, and the spatial relationships between the objects. Our results demonstrate that the proposed approach is an effective lighting design tool, suitable for users without expertise or knowledge in visual perception or in lighting design.

1 Introduction

Lighting design for image synthesis involves specifying values for lighting parameters, such as position, color, and intensity, for each of the light sources in a 3D scene model. Once the scene geometry, the material properties, and the viewing parameters have been specified, the appearance of the scene in a rendered image depends exclusively on the lighting. Poorly designed lighting may result in incomprehensible images, containing under- and over-illuminated regions, exhibiting poor contrast, and failing to effectively communicate the three-dimensional structure of the scene to a human observer.

In order to find an image with a desired appearance, one has to search through the space of possible lighting specifications. The traditional approach towards lighting design for image synthesis typically uses a *direct design* paradigm, where the user iteratively specifies all of the required lighting parameters, renders the scene, evaluates the results, makes modifications in the design, and so forth. This is essentially a trial-and-error approach, with the obvious drawback that the user must actively participate in each iteration. Thus, the design process is time-consuming and tedious. Furthermore, since the user may need to manipulate several lighting parameters and predict how their values will affect the the resulting image, the process requires expertise in lighting design as well as an understanding of visual perception issues.

An alternative approach is based on an *inverse design* paradigm. The user is presented with some interface that enables him to specify a set of objectives and/or constraints that the lighting design should satisfy, and the parameters are then solved for in an automatic fashion [6, 14, 23, 30]. These methods, reviewed in Section 2, are certainly less tedious, but still require users to know and to be able to articulate *a priori* what is the appearance that they desire to achieve. Thus, this approach might still be difficult to use for a non-expert user, whose goal is merely to render a comprehensible image of the scene at hand. There seems to be a need in helping a user to define his lighting design goals. That is, supply some directives for selecting the desired image out of the range of possible images that can be rendered for the scene.

This work presents a novel fully automatic approach to lighting design, geared towards generation of comprehensible, communicative images of 3D objects. More specifically, given a geometric model of a 3D object or scene, the material properties of the surfaces in the model, and the desired viewing parameters, our approach automatically determines the values of various lighting parameters. This is done by optimizing a *perception-based image quality objective function* designed to quantify the extent to which an image of a 3D scene succeeds in communicating scene information, such as the 3D shape of each object, fine geometric details, and the spatial relationships between the objects in the scene.

In Section 6 we utilize our image quality function as an objective function for lighting design: we present a system that searches the space of lighting designs spanned by several free lighting parameters for a locally optimal lighting design (corresponding to a local minimum in the objective function). In conjunction with some heuristics for automatic setting of the initial lighting specifications, our system provides a *fully automatic* tool for lighting design. Thus, our approach provides both the lighting design goals and the methodology to achieve them.

Our method is most suitable for ordinary users, with no expertise in lighting design or visual perception, who simply wish to synthesize a comprehensible image of their scene model. It is also suitable for incorporation into various modeling tools for CAD and animation. The experiments reported in Section 7 demonstrate that our approach is able to quickly and automatically generate lighting designs that are significantly superior to commonly used default lighting configurations.

The remainder of this paper is organized as follows: the next section contains an overview of releated works; Section 3 contains relevant background on human visual perception; In sections 4 and 5 we construct the quality function. In section 6 we describe and implement our lighting design system. Results of lighting solutions generated by our system are given in section 7. Finally, section 8 contains conclusions and future work issues.

2 Related Work

Most previous automatic lighting design systems use the inverse design paradigm. From the user's point of view such systems are primarily characterized by the set of design goals the user is free to specify, and the design space that they search. Most systems search a very limited portion of the lighting parameters space, and still require considerable knowledge and expertise from the user.

Schoeneman et al. [30] control colors and intensities of light sources by "painting" desired colors onto the scene's surfaces. Kawai et al. [14] control light emissions and directions, as well as surface reflectances by requesting the user to specify various constraints and objectives for the illumination. Both of these techniques work for mostly diffuse scenes, and do not change the positions of lights. The design goals are achieved using optimization. Poulin and Fournier [22] and Poulin et al. [23] let the user specify shadows and highlights as design goals, from which they infer the light source position and surface roughness.

Costa et al. [6] present a methodology in which fictitious luminaires can be defined and placed in the scene to describe desired radiance distribution. Free design variables are then chosen (e.g. light location and direction), and optimization is used to determine their values. This is a powerful approach, capable of handling a wide range of design variables and constraints; however, specifying the design goals and the constraints in this system appears to be a difficult task even for expert lighting designers. Furthermore, the objective function for the optimization process must be also specified by the user by programming it using a supplied scripting language.

An entirely different approach (non inverse design) for exploring the space of lighting designs is presented in the Design Galleries framework of Marks et al [18]. Given a set of lighting parameters, they try to optimally disperse the space of solution images in terms of perceptual quality, and allow the user to browse these possible results and linearly combine them to try and compose a desired solution. However, the metric that they use to measure the perceptual quality distance between two given images is a simple pixel intensity distance.

Several other relevant works belong to the area of non-photorealistic rendering. These works are concerned with generating visually comprehensible renderings of 3D objects. Having the privilege of using non-photorealistic enhancement techniques, these methods usually draw the edges of the objects in black, and enhance the appearance of surfaces by techniques such as adding cool-to-warm tone gradations[12] or drawing contour lines and curved hatching[29]. Our work has similar goals, but we limit ourselves to photorealistic computer graphics techniques, and enhance visual comprehensibility by manipulating only the lighting parameters.

3 Visual Perception

In order to design a perceptual quality metric for automatic lighting design, we must first define what visual information we would like our images to communicate, and then find practical computational ways to *quantify* the effectiveness with which this information is communicated in specific images.

Our approach is based on a fundamental assumption that the perceptual quality of computer-generated images is determined by several distinguishable aspects of visual information:

Shape Since computer-generated photorealistic images are normally concerned with displaying 3D scenes, the most fundamental requirement from such an image is that it should clearly convey the 3D shape of the visible surfaces and objects, as well as their spatial organization and relationships.

Details A photorealistic image of a 3D scene should capture, as much as possible, the fine geometric details present in the model, and display them conspicuously.

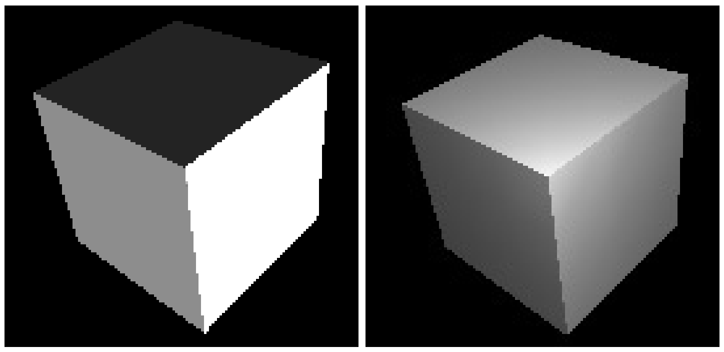
Surface properties An image should communicate surface properties, such as color, reflectance (and in particular reflectance borders), and roughness.

Realism Beyond communicating shape information a photorealistic image should convey a realistic impression. For our purposes, we are concerned with the *presence* of visual information that gives a sense of realism and not with the degree of accuracy of the image in terms of similarity to a real world scene [28]. Figure 3.1 demonstrates this issue.

The next step is to address the question of how these types of information are represented in the image (and in particular how they might be affected by changes in lighting), and consequently the problem of finding a way to detect and measure them.

3.1 The human visual system

The main task of the human visual system (HVS) is to derive a representation of shape [2, 19]. The HVS can do much more than this, but informing the perceiver about brightness, color, texture an so



(a) (b)

Figure 3.1: Realistic appearance.
In both images the object is easily recognized as a cube. However, as a result of the different shading patterns and contrasts, (b) appears more realistic than (a).

on, is secondary to deriving a representation of shape. The input for the process of recovering the 3D structure of a scene is the spatial and spatiotemporal patterns of light arriving from the world (or from an image of a scene, in our case) and falling on the retinas . The HVS analyzes these patterns of light to retrieve information about surfaces and objects in the enviroment being viewed. The questions of interest for us are *what* features in the retinal image convey the information leading to three dimensional perception, *how* can they be detected, and *how-much* each feature contributes to visual perception.

The work and research done in the fields of visual perception and vision can provide us with only limited answers to the “what” question, even fewer answers to the “how” question, and hardly any useful answers to the “how-much” questions. In other words, the current results in these fields are insufficient for our purpose of quantifying the perceptual quality of a given image. Nevertheless, some of the visual perception theories and psychophysical research did provide us with useful information about features that affect image perception.

The following subsections contain a brief examination of psychophysical experiments and research, visual perception theories and computer vision, discussing their limitations regarding to our purposes, and focusing on the aspects useful to our work.

3.2 Psychophysic and Vision research

Psychophysical experiments try to isolate different components from the input of the visual system, and draw connections between these components and the performance of subjects with given perceptual tasks [3, 7, 15, 16, 21, 25, 31]. Such experiments are the basis to the development of broader vision theories, which attempt to provide inclusive explanations of how, in psychological and computational terms, the information is actually extracted from the input, and how the process is physiologically carried out by the neurons. Unfortunately, due to the high complexity of the visual process, the lack of profound physiological understanding, and the nature of techniques used for these experiments, the research and theories in the field of visual perception are often inadequate when trying to generalize them for use in computer graphics, and especially for our purposes of formulating mathematical connections between a given image and a quantitative perceptual quality. The reasons are:

1. experiments are conducted under limited laboratory conditions and are made on very simple objects, taking into consideration only a single type of stimulus at a time. The input for the visual system, however, is usually more complex, containing features which have mutual affection on the way we interpret them.

2. Results of different researches sometimes lead to contradictory conclusions, and to date there are quite few approaches trying to account for different aspects in the visual process.

3. The connections between causes (stimuli) and effects (perceptual performance) are often formulated in a more qualitative than quantitative manner. Furthermore, most qualitative results are formulated in threshold rather than supra-threshold manner, which make them less useful for us.

It is somewhat ironical that it was the development of computer vision and artificial intelligence research that formed the basis for the establishment of more coherent visual perception theories, with a more computational and algorithmic approach. In computer vision research, some mathematical models were developed trying to solve the problem of recovering the geometry of surfaces from images (e.g. [31, 16]). On the face of it such models could be a source of quantitative connections between image information and perceptual ability. However, they are typically developed under very restrictive assumptions, and more important, since these methods derive from solving problems for computer performance it is questionable how similar can they be to the processes carried out by human.

Coming from the field of artificial intelligence, Marr [19] used a computational approach to devise a comprehensive theory of vision, which meant to be applicable to human perception, and hence he relied at least in part on human psychophysical evidence to support his statements. Furthermore, an important part of his work was to provide possible explanations about how the information-processing algorithms suggested by the theory are implemented by the neurons. Although some aspects of Marr's theory have been shown to be faulty, and there are still doubts whether it is in principle possible for machines to simulate human processes such as perceiving, such approach is considerably more reliable, when working in the context of human perception, than pure computer driven approaches.

Although suffering from problems mentioned above, some aspects in visual perception theories and psychophysical researches can supply some basic explanations regarding image features that lead to three-dimensional perception, and in particular to the recovery of the *depth* and local surface orientation at each point in the viewing field, which provide the description of surfaces properties in the final stages of early visual processing.

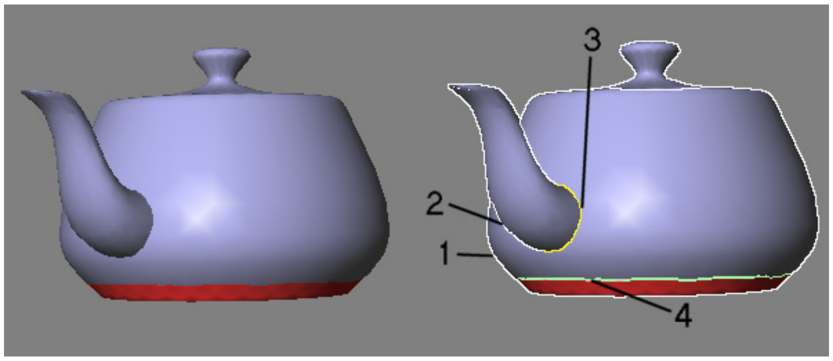


Figure 3.2: Edge types.
The edge-lines of the teapot on the left are drawn on the right: object-background occluding edge (1), self occlusion (2), boundary edge (3) and reflectance edge (4).

3.3 Edges

According to Marr [19] and others, the HVS is organized as a multistage information processing system. The function of the early visual processing is to achieve a description of surfaces in the scene. The process starts with a luminance analysis of the retinal image, looking for intensity changes that may potentially represent what seem to be the most fundamental features of surfaces in the image: edges. The edge information achieved in the early stage of visual perception is a fundamental input for further stages to come, and in particular for the recovery of three-dimensionality.

Two significant types of edges are used to characterize object shape [3, 15, 20, 25, 29]:

occluding (profile) edges including edges caused by self occlusion of object's surfaces, and those caused by occlusions between different objects or between object and background.

boundary (internal) edges edges where surfaces belonging to the same objects meet, forming a discontinuity in surface orientation. These edges can be characterized by a first order discontinuity of surface [29].

We refer to these two types of edges collectively by the term *feature edges*. Another type of edges inherent in the scene are *surface-reflectance edges* [1] (which are often represented by textures in synthetic image rendering). Figure 3.2 demonstrates the different type of edges mentioned.

Edge detection Clearly, there is a relationship between the places in an image where light intensity changes, and the places in the scene where feature edges exist, but this relation is complex. In most natural images, the changes in light intensity associated with the edges of objects are embedded in a mass of changes caused by other scene attributes such as reflectance edges, shadows, and other strong shading variations due to highlights or surface curvature. One result of the early stage of visual processing is a complex representation of all these intensity changes that have the potential of being feature edges. Marr termed it “the raw primal sketch”. Using this noisy representation, the system has to properly derive surface edges in further visual stages. Several computational models were presented for achieving the intensity changes representation described above[2]. Most of them start with filtering the image I with multi scale Gaussian low-pass filters (G), followed by a Laplacian operator (∇^2). A phase of edge detection is then applied to the multiscale $\nabla^2 * G * I$ representations, by methods such as combining zero-crossing information from the different representations [20], or by measurements made on the recombination of the representations [32]. Yet, as mentioned above, the edge map derived by this process is by no means guaranteed to contain only surfaces edges, and further visual and cognitive processing is to be performed for properly attributing these luminance edges to scene features. This processing involves aggregation of similar patches together to form larger units. A higher level knowledge or assumptions on the viewed world may constrain the interpretations of the edge map (the extent to which such knowledge is involved at early visual stages is controversial). A key point here is that if an edge of a surface in the scene is for some reason not represented by a correspondent luminance change that can be detected as an edge in the image, then the image might be misinterpreted. And this is exactly where computer generated images have an advantage from our point of view: given the scene model and viewing parameters, it is possible to predict where surface edges should occur in the rendered imaged, and practically derive a target edge map which the image must conform to, in the sense that an edge in the target map is expected to be seen as an edge in the rendered image. In particular, this enables us (a) to use a simple (and hence computationally cheap) edges operator, that only needs to determine whether a predicted location in the image have the feature on an edge, (b) measure a quality component of the image in terms of the prominence of surfaces edges, and (c) aim to illumination conditions such that this quality is satisfied.

Edges and shape perception In essence the importance of edges in shape perception is quite intuitive: in order to identify an object, the visual system must locate the borders between the various surfaces that make up the object and those that distinguish it from the surrounding objects. Also, the outer contours of an object provide information on both object shape and surface relief (outer contours follow convexities on the object surface that are normal to the observer’s line of gaze). Koenderink [15] and Marr [19] show how the occluding contour of surfaces can lead under certain conditions to shape recovery without any further visual information. Ramachandran [26] demonstrates a direct interaction between edges and the derivation of shape from shading (this will be further discussed in section 3.4). Non-photorealistic algorithms that integrate image enhancement techniques into the rendering process [12, 29] perform an explicit drawing of surface edges to make them more pronounced. This technique is also commonly used by technical illustrators [12]. It is interesting to note that Ramachandran[26] found out that sometimes an illusory contour created by occlusions leads to a better 3-D perception than a black painted contour.

3.4 Derivation of shape information

There is evidence that the information in the visual scene is broken down into several separate pathways of representation and processing, including luminance, color, textures, motion and binocular disparity [3] (when the information arrives from a 2D image, only the first three exist). These representations interact with each-other during various stages of the visual process. Thus, edge information obtained in early visual stage is integrated with other luminance information, as well as information from the rest of the perceptual pathways, and form the input for the visual task of perceiving three-dimensionality. The representation of three-dimensionality is achieved then by deriving several *maps* of information: local surface orientation and rough depth maps [19], as well as that luminance map is split in this stage into two complementary maps — reflectance and shading — by decomposing luminance at each location into its components assigned either to the reflectance properties of the surface or to the light falling on the surface [1]. Clearly, these maps must conform to each other so they can provide a consistent 3-D interpretation. Apparently the maps are derived simultaneously and mutually contribute to each other.

The information for the construction of the different maps of representation, and in particular the depth map, is mainly provided by what is known as the *depth cues* which are discussed in following subsections.

3.4.1 Depth Cues

Depth cues are those stimulus characteristics that are used by the visual process to perceive depth. These cues can be divided into two groups: physiological and pictorial cues [2, 5]. Physiological cues include degree of convergence of the eye, accommodation of the lens, and stereopsis. These cues are irrelevant when processing information arriving from a single 2-D image of the world. Pictorial depth cues (also called monocular cues) include motion cues, which are also irrelevant when dealing with a static image, and the following list of cues that may exist in rendered images:

Geometrical Linear perspective and relative size and height of the objects on the retinal image. When viewed in textures, these cues are referred to as “texture gradients”.

Feature edges Profile edges due to silhouette and self occlusion, and internal edges due to surface boundaries.

Shading Luminance gradients created by the reflection of light on surfaces.

Shadows Self and cast shadows.

Occlusion Objects being occluded by closer ones.

Atmospheric perspective Attenuated appearance of distant objects due to light scattering and absorption by the atmosphere (in photorealistic rendering this is simulated mainly by “fog” effects).

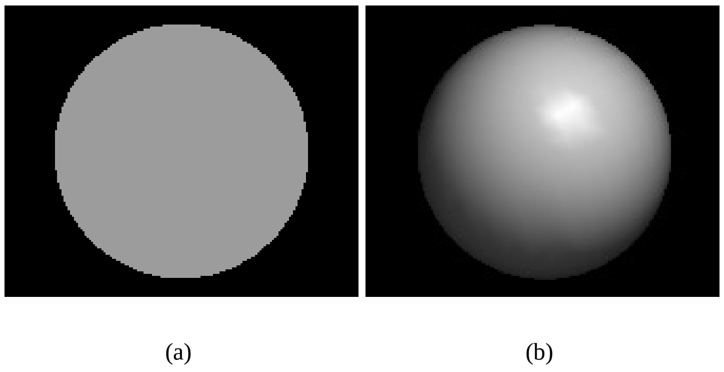


Figure 3.3: Shading as a depth cue.
Left object (a) looks like a flat disk due to lack of any shading variations. Adding appropriate shading (b) gives a vivid impression of depth and object looks like a 3D sphere.

The depth and orientation maps are most directly inferred from shading gradients and from surface and textural contours. The division of luminance map into separate reflectance and shading maps is also affected by existence of reflectance edges, which can be either achromatic or chromatic.

In terms of image quality, the very existence of such cues in the image can be used as indication for quality. Among these cues, the ones that are most directly affected by illumination configuration, apart from edges that were already discussed, are shading and shadows. Some psychophysical research examined the effect of these two cues on derivation of shape.

3.4.2 Shape From Shading

Types of shading Restricting the discussion to shading caused by the reflection of direct illumination on surfaces, two aspects of shading should be distinguished: (a) shading caused by light falling on surface boundary locations, which in general will give rise to the visibility of edges; and (b) shading on other areas of a surface, which may give rise to patterns of “smooth” shading gradients in relation to the shape of the surface. The significance of the former in recovering shape information is embedded in the edges it pronounce, whereas the latter provides further cues for surface depth and orientation, which will be discussed here.

Shading patterns as depth cues Shading variations alone can convey an impression of depth, as demonstrated by Figure 3.3. However, shading by itself is relatively a weak cue for perception of local depth and surface orientation. Several studies (e.g. [16, 31]) revealed that subjects showed substantial inaccuracy in tasks of estimating local surface curvature or orientation based on shading information. Shading is much more effective when integrated with other cues, if available. Ramachandran[26] demonstrates that the visual system recovers information about the shape of objects by combining outline contours and shading cues. These outlines are naturally represented by the feature edges of surfaces. This combination of feature edges and shading gradients to convey shape information is a fundamental basis for establishing the quality metric in section 4. Todd[31] found that adding textures to the surface improves the performance of the observers. Another factor that is found to improve performance is existence of a *specular highlight* on the surface [7, 31]. Specular highlights are also mentioned as important for shape communication in the work of Gooch[12] which was inspired by principles of technical illustration.

How exactly the patterns of shading are analyzed by the HVS, and how can the contribution of existing shading information in the image be estimated for a purpose such as image quality metric, are questions with no clear answers. Although mathematical models, that have been developed in the field of computer vision can, under certain conditions, recover geometry of surfaces from shading analysis, there is substantial evidence that they don't simulate actual human observers: First, as mentioned earlier, these models are generally applicable under restrictive assumptions such as Lambertian surfaces only, or advance knowledge of surfaces reflectance, etc. Second, results from psychophysical experiments do not conform to the computational models. For example, human observers showed poor performance in estimating local surface orientation or illumination direction [16, 21], and yet they may have excellent understanding of an object's shape. And indeed, a third argument is that the mathematical models try to recover shape by local analysis of the shading, whereas there are evidence that shape derivation, in general, and shape from shading in particular, is a global process, involving either the entire visual field or a large portion of it, with integration of high level knowledge and other types of visual information [15, 21, 26].

Cavanagh et al. [3] suggest that the contribution of shading to surface understanding may not be mediated by the computation of surface normals at all, but perhaps proceed on the basis of *surface contours* by interpreting the shading on a surface as tangent field, with each local tangent oriented at the same direction as the luminance gradient function, and thus act as a set of surface contours sufficient for the reconstruction of the surface relief.

Thus, without a real computational understanding of how the visual system is deriving depth and shape information from shading gradients, our best bet is to take advantage of the knowledge that the visual system *is* actually making use of shading gradients, in a manner that is perhaps more global then local, and to measure the very existence of those gradients in the image. When rendering a scene, illumination conditions can be chosen so as to encourage this existence of shading gradients in the image.

Illuminant direction Apart from the explicit recovery of shape, a few experiments were concerned also with certain effects of illuminant direction on the interpretation of the scene, which is a related interesting subject from our point of view. A series of experiments examined the influence of illuminant direction on the performance in local surface curvature tasks [7, 21, 31]. These experiments studied only a very limited set of simple objects and settings, and there is no thorough investigation of the issue. Furthermore, results from various experiments are not consistent. It seems that the most systematic study was conducted by [7], although inadequate (e.g. the only objects they used were spheres). Their conclusion was that most accurate performance of subjects in estimating local surface curvature was achieved with illuminant direction at about 45 degrees above simulated eye position.

A more general observation, though compelling, made by several researches is that the visual system preferably assumes that the light illuminating the scene is coming from *above*. Ramachandran [26] demonstrates three aspects of this principle: (a) ambiguities in surface relief (i.e. is it concave or convex) are often resolved such that shading conforms to light coming from above, (b) impression of depth is more powerful when the illumination is from above than from the side, and (c) lighting from above greatly enhances one's ability to group and segregate objects (operations performed in final stages of visual perception). Notice that this observation is usually made in conjunction to assuming that the scene is lit by a single light source, what seems to be another preferable assumption of the visual system [1, 26]. This is firstly attributed to an inherent assumptions of the HVS on the world, but also to the fact that when trying to adapt a global interpretation to the scene, understanding of light sources presence is helpful (although not necessary): several light sources are more difficult to recover, and if not properly recovered can yield illumination effects that may perceived by the visual system as contradictory.

Shading and realism One more aspect of shading is relevant to our work: even when shading cue is not indispensable for purposes of shape derivation, it is still a fundamental image feature for conveying an impression of realism. For a cube to be recognized, prominence of edges is enough. However, a cube with a close-to-uniform shading on each of its faces will convey less impression of realism than a cube with more noticeable shading gradients (see Figure 3.1).

3.4.3 Shadows

Shadows in the scene introduce a certain conflict: on the one hand shadows can provide important clues to help understanding the geometry of the scene and the interrelationship between objects. On the other hand a shadow might hide shape and material information in its dark regions. Consequently, the contribution of shadows to a displayed scene should be considered carefully. Cavanagh et al. [3] found in their study of shape perception from shadows that the only requirement that is necessary for the perception of depth due to shadows was that shadow regions be darker (in terms of luminance) than the surrounding nonshadow regions and that there be consistent contrast polarity along the shadow border. Therefore, a shadowed surface area doesn't have to be completely in the dark in order to provide cues associated with shadows. That is, in cases where important information lies in a shadowed area, if a dim light reaches that area, it may reveal some of the hidden information without offsetting the useful effects of the shadow.

Two types of shadows are distinguished [3]:

Attached shadow Surface areas which the angle between their normal and direction of light falling on the object is equal or greater than 90 degrees are not directly illuminated by the light and consequently are in shadow with respect to that light. Such shadow is said to be attached to the feature casting the shadow.

Cast shadow A shadow created where object is blocking light from reaching some parts of the scene. This includes both the case where shadow from one object falls on another, and the case where the shadow of one part of the object falls on another part (self shadowing).

An attached shadow provides information about the surface orientation. In particular, the border of such shadow follow convexities on the surface that are everywhere normal to the direction of the illuminant. Such a constraint is probably well known by the human visual system. Therefore, it is generally desirable to illuminate a scene such that object are partially shadowed, possibly with some dim illumination on shadowed areas. This approach is adopted both by photographers, who prefer side illumination upon strong front illumination, with some secondary illuminant for the shadowed areas, and by technical illustrators who use single light source from above the scene, with techniques like cool-to-warm hue shift to account for surface curvature in shadowed areas [12].

Cast shadows can provide information about the position and shape of the object casting them, on the one hand, and a subset of object contours on the object on which the shadow falls, on the other hand. A useful constraint related to cast shadows is that enclosed shadows regions signal concavities [3]. Automatically evaluating the contribution of cast shadows to scene interpretation, and directing illumination for optimizing shadow cues while considering the tradeoff with the need to minimize lost of information in darkened areas, are difficult tasks, and were not handled in the frame of the current work.

3.5 Other perceptual issues

Spatial frequency domain

Although the visual process as described above is concerned primarily with the spatial luminance distribution, some aspects of the human visual system behavior are directly related to the spatial frequency content of the retinal image, and as such are attributed to a spatial processing of the frequency patterns by multiple bandpass mechanism, believed to be performed by specialized brain cells in the visual cortex [9, 27, 4]. Some approaches suggest that image is internally represented in the brain by channels of spatial frequency and orientation. These models can explain visual characteristics such as [4] *contrast sensitivity* - the contrast sensitivity of the visual system is a function of the spatial frequency and orientation of the stimulus pattern. Sensitivity to high frequency content is generally poor; *spatial masking* - detectibility of a particular pattern is reduced by the presence of a second pattern of similar frequency content; *contrast adaptation* - sensitivity to selected spatial frequency is temporarily lost after observing high contrast patterns of the same frequencies; as well as other characteristics. Spatial frequency representation is also suitable for analyzing geometrical depth cues, and especially texture gradients [9]. In terms of image quality, spatial frequencies can supply estimation of fine details visible in the image, indicated by high frequencies.

In our metric, however, we currently don't use spatial frequencies analysis from several reasons specified in Section 4 below.

Non linearity of luminance perception

The subjective brightness (i.e. brightness as perceived by the human visual system) is a non-linear function of luminance [10, 11, 13, 4]. The particular non-linear relationship is not well established, but the function is usually found to behave logarithmically. Consequently, when working with a linear luminance scale, the eye responses to logarithmic contrast. That is, the perceived contrast of two luminance intensities I_1 and I_2 is proportional to $\log(I_2) - \log(I_1)$. Hence the eye is sensitive to the ratio of intensities, I_2/I_1. Therefore, if accurate measurements of perceived contrast (or luminance gradients) is to be performed with a linear luminance scale, this non-linearity should be taken into account. This can be done by either mapping luminance to contrast scale by techniques such as taking the $\log_{10}$ or the *cube root* of the luminance intensities [13], or by using suitable contrast formula such as Michelson contrast: $C = (I_{max-}I_{min})/(I_{max} + I_{min})$, used for computing contrast in luminance gratings [10].
For computer images another relevant factor that should be considered is the non-linearity of the CRT: displayed luminance is exponential function of the input voltage (which is in the general case proportional to pixel value). The exponent is usually denoted by γ and its value is typically 2.8 ± 0.3 [10]. In *gamma-corrected* monitors the input signal is adjusted before sent to the monitor, so the resulting pixels intensity values linear are linearly scale. Interestingly, the monitor's gamma correction domain is close to the visual system's cube root domain, and therefore if a monitor is not gamma corrected, the response of the eye to pixels intensities will be close to linear, without any mapping.
In this work measurements of luminance gradients and contrasts are applied to image pixels without regarding any of the non-linearities mentioned above. It is stated here that these measurements have higher perceptual accuracy when perceived brightness is linear with the pixels intensities scale.

Brightness adaptation

The human visual system can adapt to light intensities in a wide dynamic range, in which intensities change by factor of more than 10^6 for photopic vision alone. However, it cannot operate over such a range simultaneously. Rather, this large variation is accomplished by changes in the overall sensitivity, a phenomenon called *brightness adaptation* [2, 10, 11, 4]. At any situation the visual system is adapted to a certain light intensity, which is called the *brightness-adaptation level*, and is most sensitive to intensities around that level, and totally insensitive to intensities at some distance below it, which are

all perceived as black. Sensing much higher intensities will result in shifting the adaptation level to a higher point.
One implication of this phenomenon in image perception is that images with large portions displayed in intensities much above or much below the average (which implies the adaptation level) are generally not desirable.

Contrast sensitivity

The sensitivity to luminance contrast is a function of the brightness adaptation level (as well as of spatial frequencies, as mentioned in 3.5 above) [10, 11, 27, 4]. One variable that was measured in psychophysic experiments is the *just-noticeable-difference* (JND) contrast. That is, what is the minimal luminance change required for an observer to perceive a difference. For a uniform background level at luminance B with a small spot in the middle at luminance $B + \Delta B$, the JND is constant with respect to the ratio $\Delta B/B$, with value of about 0.02 for a wide range of intensities (this is known as Weber's law). A more realistic setting is when background is at some level B_0, and the contrast is introduced by two adjacent spots of intensities B, $B + \Delta B$. The results show that the farther B is from the background, the higher is the threshold for perceiving the luminance difference. Once again this demonstrates that the visual system is most sensitive to luminance changes around the adaptation level.

Lightness constancy

Human observers can properly perceive the albedo of an object under a wide range of illumination intensities (a piece of white paper appears to have approximately the same albedo whether it is viewed in dim light or bright light) [5, 4].

Color domain

Thus far the discussion about image perception was focused on luminance information and ignoring chromaticity, and for good justification. Although in some situations the spectral distribution of light in the image can be used by the visual system for spatial tasks, it is usually by far less effective than luminance image. In particular it has been shown that color image is not good for depth perception [3, 26, 8]. On the other hand a chromatic map can directly provide information about material composition and textures.

3.6 Conclusions

In this section we have discussed relevant issues in perception theories and psychophysical research and the difficulties in applying them to our work. Here we bring a summary of information and features in the image, as well as some principles of the HVS, which we have nevertheless found useful to our needs:

- **Edges:** Occluding edges, boundary edges and reflectance edges should be displayed prominently by the image.
- **Shading gradients:** The presence of shading gradients on surfaces provides important shape information, as well as impression of realism.
- **Shadows:** Attached shadows and cast shadows helps to recover 3D shape. In this work we take into account only attached shadows.
- **Highlights:** Specular highlights can also help in perceiving the shape of the object, as well as providing information about the nature of the material.
- **Luminance and Color:** Luminance information is much more important for shape perception than color information.
- **Brightness adaptation and contrast sensitivity:** At any situation the HVS is adapted to a certain light intensity level. When viewing an image that intensity is determined by the average image intensity. The HVS to luminance variations is highest around the adaptation level, and decreases as the intensity levels get farther.
- **Lightness constancy:** There is no unique illumination intensity for the albedo (reflectance) of an object to be perceived correctly by the HVS.
- **Light source:**
  - **Light direction:** The HVS tends to assume that the light illuminating the scene is coming from above.
  - **Spectral distribution:** There isn't any benefit from illuminating the scene with colored light. On the other hand, using achromatic light guarantees that material colors are correctly shown.
  - **Number of light sources:** The HVS preferably assumes that the scene is illuminated by a single light source. Furthermore, multiple light sources can cause effects which may be confusing and contradictory to the HVS. Therefore, adding more light sources to a scene should be generally done only when necessary for resolving some illumination deficiencies.

4 Perceptual Image Quality Function: Principles

In this section we construct a perceptual image quality function for lighting design. More specifically, guided by our knowledge about the human visual perception of 3D shape and spatial relationships, we define a mapping f_Q that receives as input a 3D scene model M and an image I that was rendered from this model, and maps its input to a single non-negative scalar value. This value attempts to quantify the extent to which the image I contains features and exhibits properties that make it easy for a human observer to comprehend the 3D shape and structure of the scene M. The smaller the value of $f_Q(I)$, the higher the estimated perceptual quality of the image. Naturally, f_Q is designed in such a way that it is strongly dependent upon the lighting in the scene. Changes in the lighting design cause changes in I, which are in turn reflected in the value of $f_Q(I)$. Thus, the problem of finding an optimal lighting design for the scene is cast as an optimization problem — finding a local minimum of f_Q.

It should be noted that the images I are luminance images, rendered from the model M without applying any surface textures that might be present in the model. The reason is that we operate on pixel intensities when quantifying the perceptual quality of an image. Surface textures perturb these intensities, making it difficult to isolate the effects of changes in lighting on the perceptual image quality. Therefore, if If the scene does contain surface textures, the lighting design process is performed ignoring them, but they can be integrated back into the scene once the lighting has been determined. Further discussion about other types visual information we have chosen *not* to handle directly is given in subsection 4.3.

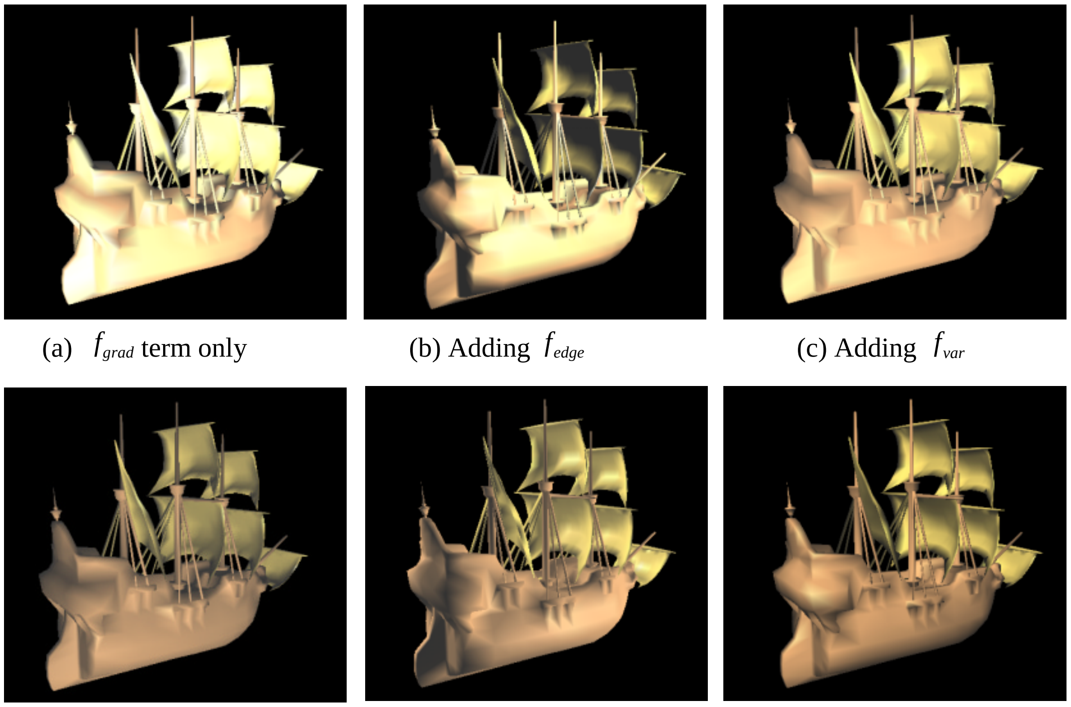
The function f_Q is defined as a linear combination of six *target terms*, each responsible for measuring a different feature or property in the image:

$$f_Q = f_{grad} + f_{edge} + f_{var} + f_{mean} + f_{hist} + f_{dir} \tag{4.1}$$

More specifically, these six terms have been designed to respond to:

1. *Local luminance patterns:* The term f_{grad} measures the magnitudes of the shading gradients present in the image. The term f_{edge} detects edges in the image and measures their prominence.
2. *Pixel luminance statistics:* f_{var} measures the distance of the luminance variance from a target value. f_{mean} measures the distance of the mean luminance from a target value. f_{hist} measures the distance of the luminance histogram shape from an ideal equalized histogram.
3. *Illumination direction:* f_{dir} measures the elevations of the light sources with respect to the viewing direction.

In subsection 4.1 we describe each of the six target terms in more detail, and illustrate their impact on the lighting of a scene. The order in which the functions are presented is similar to the order in which the terms were added into the quality function in the course of our research: each new term was



(a) f_{grad} term only (b) Adding f_{edge} (c) Adding f_{var}

(d) Adding f_{mean} (e) Adding f_{hist} (d) Adding f_{dir}

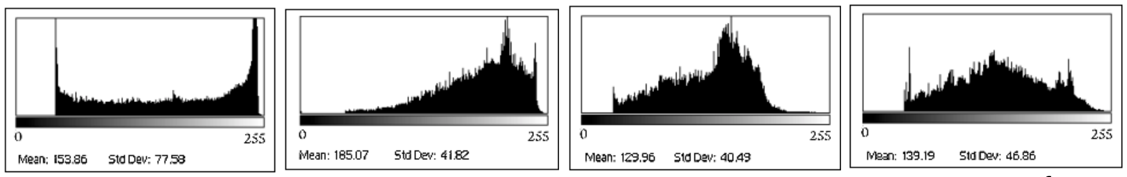
Figure 4.1: Quality function: cumulative effect of target terms on the image. All scenes were rendered using two light sources.

added in order to overcome deficiencies unresolved by the previous terms. For clarity of presentation, the terms are described at the level of principles. A precise definition for each term will be given in Section 5.

4.1 Target terms

4.1.1 The shading gradients term f_{grad}

This term measures the average shading gradient magnitude in the image. Only shading gradients in geometrically smooth regions of the scene are taken into account here, since gradients at the edges are accounted for by a different term f_{edge}. The value computed by this term is the difference between the measured average gradient and a target value representing the maximum average gradient that could potentially be measured in an image of that particular scene with the given viewing parameters.



(a) $f_{grad} + f_{edge}$ (c) Adding f_{var} (d) Adding f_{mean} (e) Adding f_{hist}

Figure 4.2: Quality function: cumulative effect of target terms on the histogram. Images a-d are the histograms of the images in Fig. 4.1 b-e, respectively. The black background was excluded.

Example: Figure 4.1a was rendered using lighting parameters obtained by optimizing only the f_{grad} term. It is obvious that this term alone is unable to produce satisfactory results: the resulting image is too bright, and much of the detail is washed-out by the strong illumination.

4.1.2 The detected edges term f_{edge}

Given a 3D scene model it is easy to establish which edges could be visible in an image of the scene, rendered from a given viewpoint [29, 17]. The extent to which these potential edges are in fact perceived by a human observer, depends mostly on the lighting. The f_{edge} term measures this extent by applying an edge detection operator at pixels located on potential edges, and summing the responses.

Example: Figure 4.1b was rendered using lighting parameters determined by optimizing the function f_{grad}. As expected, several edges that were not visible in 4.1b become clearly visible now. A secondary result of adding f_{edge} that it reduces the overall excessive intensity caused by using only a gradient component. The result is still not satisfactory, however, since the image tends to contain extremely dark (under-illuminated) regions alongside with extremely bright (over-illuminated) ones. Important details are often lost in both types of these extreme regions. It is difficult to perceive fine detail in such regions, because of the global brightness adaptation of the visual system, which causes poor contrast sensitivity in dark and bright areas.

4.1.3 The variance term f_{var}

In order to overcome the problem of large extreme dark and bright regions, we introduce a term that inhibits extreme variations in intensity, by measuring the distance between the variance in the pixel luminances and a target variance value. This target variance must be chosen carefully, so as to reduce the extreme variations in intensity, but still allow a sufficient dynamic range in which shading gradations can take place.

It should be noted that our use of a variance reducing term is consistent with the low dynamic range principle used in (non-photorealistic) technical illustration[12].

Example: Figure 4.1c was rendered using lighting parameters determined by optimizing the function $f_Q = f_{grad} + f_{edge} + f_{var}$, using a target standard deviation value of 42 (on a 0 to 255 scale). The histogram of the image, shown in Figure 4.2b, has a standard deviation of 41.8 that is very close to the target value (instead of 77.6 in Figure 4.2a). Figure 4.1c shows a major improvement with respect to the high intensity variance found in 4.1b; However, since there was no constraint on the mean image intensity it still appears too bright.

4.1.4 The mean term f_{mean}

The overall brightness of an image is an important factor in its appearance. From perceptual quality point of view, it is undesirable for the image to appear too dark or too bright, since this tends to weaken the effect of the shading and may hide various features and detail in the scene. Even if no loss of detail occurs, there is still some subjective notion of an "appropriate" brightness for the image. For instance, the ship in Figure 4.1c might appear too bright to most observers. In order to control the overall brightness of the image, we add another target term f_{mean} in order to pull the mean luminance in the image towards a desired target value.

Example: Figure 4.1d was rendered using lighting parameters determined by optimizing the function $f_Q = f_{grad} + f_{edge} + f_{var} + f_{mean}$, using a target mean value of 124. The optimization achieved a mean value of 130 (instead of 185 in Figure 4.1c), along with a standard deviation of 40.5. The histogram is shown in Figure 4.2c. The resulting image 4.1d is correspondingly darker than 4.1c.

4.1.5 The histogram equalization term f_{hist}

Lighting design optimized using the terms introduced so far has the tendency to inhibit shadows and highlights, often producing large areas with very uniform shading, such as the side of the ship's body in Figure 4.1d. This is manifested by the histogram in Figure 4.2c, which shows that most of the pixels have luminances in a rather narrow range around 160–170. In order to address these deficiencies we introduce another term f_{hist} designed to make the image histogram closer in shape to a more equalized histogram. This term tends to increase the variance, conflicting with the f_{var} term, but in practice it turns out that it is the proper balance between these two terms that generates the best results.

Example: Figure 4.1e was rendered using lighting parameters determined by optimizing the function $f_Q = f_{grad} + f_{edge} + f_{var} + f_{mean} + f_{hist}$. Comparing this image with Figure 4.1d we see improved shading on the side of the ship's body, as well as more highlights (on the sails) and shadows. Note the corresponding change in the shape of the histogram in Figure 4.2d.

4.1.6 Light direction term f_{dir}

Psychophysical tests indicate that it is sometimes easier for the human visual system to correctly interpret a 3D shape when it is illuminated from above. However, it is not clear how general and fundamental this phenomenon is. Furthermore, it is not clear what is the necessary elevation above the horizon, and in many scenes a horizontal illumination direction appears to yield the best results. Therefore, we decided to add to the quality function a term that constrains the light source to illuminate the scene from above, but we treat this term as *optional*, as opposed to the first five terms, which are considered *fundamental*. This term simply measures the difference between the elevation of the light sources and some target elevation angle.

Example: Figure 4.1f was rendered using two light sources: the parameters of the first light were determined by optimizing the complete quality function $f_Q = f_{grad} + f_{edge} + f_{var} + f_{mean} + f_{hist} + f_{dir}$. A secondary light source was fixed at the viewpoint, and could not be modified by the optimization process. Note that both images 4.1e and 4.1f, are quite satisfactory in terms of the lighting, despite significant differences in appearance due to the additional constraint on the illumination direction.

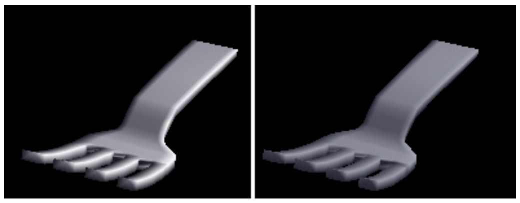
4.2 Integration of the target terms

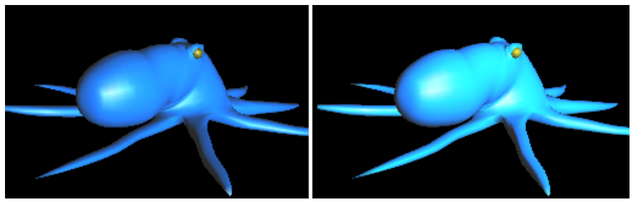
Above we have introduced the six target terms comprising our overall perceptual image quality function. Each of these terms measures the "quality" of a certain feature in the input image by computing a distance from an ideal target value. The target terms were designed so as to encourage the following desired features: The image should conspicuously show the edges of the scene (feature and reflectance edges), and introduce shading gradients on scene surfaces, such that these gradients are sufficiently strong to be noticeable. The global appearance of the image should be such that most regions are displayed by intensities within a limited range, centered at some mid-intensity level, and yet some shadowed and highlighted regions are still allowed to exist in relatively small regions, or in regions that do not contain important details and shape information.

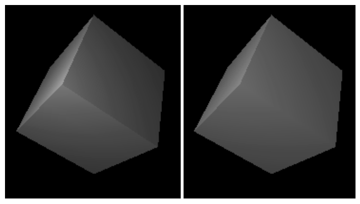
Although the resulting quality function may appear to be overconstrained, our experiments have shown that each and every one of the five fundamental target components is necessary in order to ensure satisfactory results, as demonstrated by the five examples in Figure 4.3.

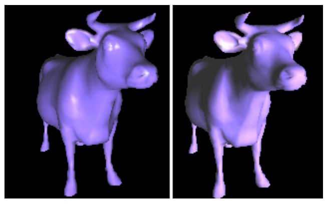
4.3 Spatial frequency and color domains

Other channels of information that could be used apart from luminance are chromaticity and spatial frequencies. We choose *not* to perform analysis in these domains for the following considerations:


(a) Gradient component


(b) Mean-intensity term


(c) Edge term


(d) Variance term

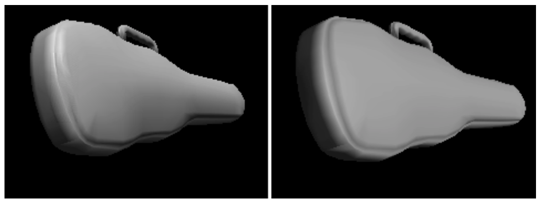

(e) Equalized histogram term

Figure 4.3: Contribution of target terms.
Each of the five examples (a)–(e) in this figure demonstrated the effect of omitting a single target term from f_Q. The left image in each example was rendered with lighting parameters obtained using the complete quality function, while the right image shows the effect of excluding a single component. (a) Excluding the f_{grad} term causes the right image to appear substantially duller than the left. (b) Excluding f_{mean} results in a partial loss of highlights and an undesirable shift in material color. (c) Excluding the f_{edge} term causes one of the cube edges to almost disappear. (d) Exclusion of f_{var} results in undesirable under- and over-illuminated regions on the cow. (e) Exclusion of f_{hist} results in a flatter appearance of the curved side of the violin case.

Spatial frequencies spatial frequencies analysis can be mainly used for measurements related to three classes of information (see also section 3.5): (a) sensitivity of the human visual system to image content, e.g. contrast sensitivity; Although taking such characteristics into account can increase the accuracy of perception-based measurements, it still seems to have relatively minor effect to our metric, and in any case involving such factors is a complex task beyond the scope of this work; (b) geometrical depth cues, especially texture gradients; Texture maps information is not included in the input to the metric, and the remaining cues are usually provided by edges which are features available in the luminance domain. (c) fine image details; Again, details due to texture mapping are not directly handled and geometrical details are generally indicated by edges and shading.

Chromaticity Color does not play an important role in recovering shape information (see Section 3.5). On the other hand it *does* provide information about the material of a surface. However, since we choose to use achromatic light with controlled intensity to illuminate our scenes (Sections 3.5 and 6), and our idea of well illuminated scene is such that deep shadows do not normally occur, the color of the materials is guaranteed to be properly displayed. Furthermore, since there is no appropriate metric to measure distance between two colors, border lines between different material are directly handled only if they present reflectance borders as well. All these requirements supplies sufficient conditions for inferring the reflectance map (Section 3.4).

5 Perceptual Image Quality Function: Practice

While the previous section described the principles around which our perceptual quality function was designed, in this section we give the precise definitions of the target terms, while addressing some of the issues that must be resolved in order to use this function in practice, in the context of automatic lighting design. These issues are:

Normalization each of the target terms of f_Q must be normalized such that the values they generate all lie in the same range, e.g., $[0, 1]$. This is a critical requirement, because if the values produced by one term are significantly smaller than those produced by the rest, the effect of that term on the behavior of f_Q is minor, and in practice it is as if this component was totally excluded, leading to deficiencies such as those shown by fig. 4.3. On the other hand, if a term's values are much larger than the rest, the quality function will be dominated by this term, and the desired balance will not be achieved (e.g. Fig. 4.1a).

Weights rather than simply taking the sum of the normalized target terms as the quality function f_Q, we assign each target term f_i a weight w_i , and define

$$\begin{aligned} f_Q &= \sum_i w_i f_i \ , \text{ where :} \\ i &\in \{grad, edge, var, mean, hist, dir\} \\ w_i &\in [0,1] \end{aligned} \tag{5.1}$$

The weights w_i can be used to manipulate the dominance of the different target terms, and thus to control the sensitivity of the quality function to different features in the image.

Target values: most of the target terms in the previous section were defined as a difference between some measured quantity in the image, and a target value for that quantity. These target values determine the desired appearance of the different features in the image. The target values typically depend on the particular scene and viewing parameters, which means that the quality function should be uniquely calibrated for each scene and viewing before performing the measurement. Although some manual calibration is possible, an automatic procedure is obviously preferred. Finding such procedures that would work fine with a wide range of scenes and viewing parameters is by no means easy. Furthermore, knowledge about the nature of the rendering algorithm may help achieving a more accurate calibration. Further in this section we will discuss how to determine the target values for each target term.

Auxiliary data structures: in order to be able to efficiently evaluate the target terms we will need an auxiliary data structure, which will be precomputed before the lighting design optimization process begins. We refer to this data structure as the *precomputed image map* (PIM). The PIM is essentially a classification of the pixels in the image into three categories: background pixels, edge pixels, and surface pixels. Figure 5.1 demonstrates a PIM for the teapot from Fig. 3.2.

5.1 Precise definition of the target terms

5.1.1 The detected edges term f_{edge}

Recall that the goal of this term is to make sure that the lighting parameters are chosen so as to accentuate, as much as possible, all of the feature and reflectance edges in the scene that can potentially be visible from the specified viewing position. In other words, if an image pixel has been marked as an edge pixel in the PIM, we would like the lighting to create an easily perceived edge at that pixel. To that end, the target for f_{edge} is given by the *edge-pixels* in the PIM, which represent an edge map of the image.

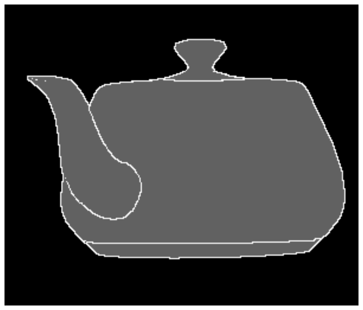


Figure 5.1: precomputed image map.
Background pixels are drawn in black, edge pixels in white, and surface pixels in gray.

In order to extract the edge map from the scene information, the visibility of edges should first be determined, and then the visible edges should be mapped onto the 2D image. Deriving feature edges from the scene can be done in several techniques, such as the modified Appel algorithm introduced by Markosian[17], or the Z-buffer based technique suggested by Saito et al.[29]. The reflectance edges can be derived by using an item-buffer of the scene polygons, and marking those pixels on the borders of adjacent polygons having a reflectance difference of a certain magnitude.

The f_{edge} term is defined as

$$f_{edge} = \frac{1}{N}\left(N - \sum_{i,j \in E} O_{edge}(p_{ij})\right) \tag{5.2}$$

where I is a luminance image, E is the set of edge pixels in the PIM, N is the total number of edge pixels in the PIM, p_{ij} is the (i,j)-th pixel of I and O_{edge} is an edge detection operator, defined such that a pixel is detected as an edge if the following two conditions are satisfied [11, 13]:

1. The magnitude of the luminance gradient $\bigtriangledown(p_{ij})$ is above some threshold.
2. A zero-crossing of the Laplacian $\bigtriangledown^2(p_{ij})$ occurs at the pixel.

The second condition can be formulated by defining a boolean operator z

$$z(p_{ij}) = \begin{cases} 0 \quad \text{if} \quad sign(\bigtriangledown^2(p_{ij})) + sign(\bigtriangledown^2(p_{kl})) \neq 0 \\ \qquad \text{for all } (k,l) \in \{(i-1,j),(i,j-1),(i,j+1),(i+1,j)\} \\ 1 \quad \text{otherwise} \end{cases}$$

and the edge detection operator O_{edge} is given by:

$$O_{edge}(p_{ij}) = \begin{cases} 1 & t_{max} < |\nabla p_{ij}| \\ & \text{and } z(p_{ij}) = 1 \\ \frac{|\nabla p_{ij}| - t_{min}}{t_{max} - t_{min}} & t_{min} < |\nabla p_i| < t_{max} \\ & \text{and } z(p_{ij}) = 1 \\ 0 & |\nabla p_{ij}| < t_{min} \\ & \text{or } z(p_{ij}) = 0 \end{cases} \tag{5.3}$$

where t_{min} and t_{max} are edge detection thresholds, $|\nabla p_{ij}|$ is the magnitude of the gradient at location p_{ij} in the luminance image I , and z is the Laplacian zero-crossing operator given above.

f_{edge} produces values in the range $[0, 1]$. It achieves the optimal value of 0 only if at all edge pixel locations the luminance gradient exceeds a threshold of t_{max}. The worst case value of 1 is obtained if none of these pixels exhibit a Laplacian zero-crossing with gradient exceeding at least t_{min}.

5.1.2 The shading gradient term f_{grad}

The goal of the f_{grad} term is to encourage the presence of shading gradients at surface pixels, providing important perceptual cues regarding the shape and the orientation of object surfaces in the scene. The term measures the average shading gradient over all *surface pixels* $g(I)$, and computes the difference between this average and a target value g_t, which represents the largest average shading gradient that can possibly be measured for this scene. Clearly, this target value strongly depends on the particular scene and viewing parameters: scenes containing shiny curved objects will naturally exhibit much stronger shading gradients than a scene consisting of matte polyhedra.

Accurate calculation of the target value requires solving an extremum problem involving the full computation process performed by the specific rendering algorithm, with the lighting parameters as free variables, and does not appear to be analytically solvable for general scenes. Accurate numeric solution would be similar in terms of computational expense to the entire lighting design optimization. Fortunately, our experience has shown that evaluating the target value with a certain degree of inaccuracy is tolerable in the sense that the behavior of the quality function is not very sensitive to such errors. Too rough evaluations, however, may damage the function behavior. In our implementation we perform an educated guess of the target value, which was found sufficient for satisfactory results. We base this guess on a single rendering of the scene: the idea is to predict what illumination conditions would generate the strongest shading gradients, set the lighting parameters accordingly, render the scene and compute the average shading gradient in the resulting image. Detailed explanation of this method is given in subsection 6.3.6.

The average shading gradient $g(I)$ is defined as:

$$g(I) = \sqrt{\frac{1}{N} \sum_{(i,j) \in S} |\nabla p_{ij}|^2} \tag{5.4}$$

where S is the set of surface pixels in the PIM and N is the total number of these pixels, and the shading gradient target term is defined as

$$f_{grad}(I) = \begin{cases} \frac{g_t - g(I)}{(1-\alpha)g_t} & \text{if } g(I) \geq \alpha g_T \\ 1 & \text{otherwise} \end{cases} \tag{5.5}$$

The parameter α is used to achieve proper scaling for the values generated by this term. The value $\alpha = 0.3$ was found to work well in practice.

5.1.3 The variance term f_{var}

The f_{var} term measures the difference between the standard deviation $\sigma(I)$ of the surface pixel luminances and a target standard deviation value, denoted by σ_t. The result is normalized by the target value:

$$f_{var}(I) = \min(\frac{|\sigma(I) - \sigma_t|}{\sigma_t}, 1) \tag{5.6}$$

Our experiments have shown that a target standard deviation between 40 and 45 generally produces satisfactory results. This value is generally valid when all displayed surfaces have uniform reflectance, and under the assumption that large cast shadows are not suppose to occur. If the scene contains a wide range of reflectances, the target value is corrected by a factor proportional to the variance in the reflectances of the visible surfaces. Further implementation details are given in subsection 6.3.6.

5.1.4 The mean term f_{mean}

The f_{mean} term measures the difference between the mean intensity of the surface pixels $m(I)$ and a target mean intensity, denoted by m_t. The result is scaled to the interval $[0, 1]$:

$$f_{mean}(I) = \frac{|m(I) - m_t|}{\max(m_t, 255 - m_t)} \tag{5.7}$$

Our experiments have shown that a reasonable target value for m_t is obtained by setting it to around 150. This value was found suitable for scenes consists of surfaces with "standard" reflectance. This standard reflectance value was selected arbitrarily (as 1.0 in our system), and for scenes with different reflectance the target value is corrected accordingly in order to create an image faithful to the reflectance nature of the scene: to avoid over-illumination for low-reflectance scenes and under-illumination for high-reflectance scenes, if the scene reflectance is lower than the standard, then the target value is lowered, vice versa. However, to refrain from the inverse illumination deficiencies (i.e. under-illumination for low-reflectance scenes etc.) this correction should not be linear. For example, if the image of a scene with reflectance 1.0 is expected to have mean intensity of 150, then the image of a scene with reflectance of 0.2 should not be rendered with mean value of 30 ($\frac{0.2}{1.0} * 150$), which is too dark an appearance, but rather with some intermediate intensity, e.g. 90. This intensity should be chosen such that on the one hand the illumination on the scene is strong enough to make it sufficiently visible, and on the other hand the low-reflectance nature of the scene is preserved. Further implementation details are given in subsection 6.3.6.

5.1.5 The histogram equalization term f_{hist}

The f_{hist} term measures the distance between the histogram $h(I)$ of the luminance image and a target histogram h_t. In an ideal equalized histogram each luminance values occurs an equal number of times $n_t = N/255$, where N is the number of surface pixels. Denoting by n_i the height of the i-th column in the actual histogram $h(I)$, the distance between $h(I)$ and h_t is defined as

$$d = \sqrt{\frac{1}{255}\sum_{i=0}^{255}(n_i - n_t)^2} \tag{5.8}$$

While searching for appropriate scaling for this term, we found that neither the ideal histogram, nor the worst case histogram (where all pixels have the same luminance) occur in practice. Therefore, the term is scaled using two scaling parameters, α_1 and α_2, such that $0 < \alpha_1 < \alpha_2 < 1$. The parameter α_1 (α_2) defines a more practical worst (best) case histogram, with all pixels equally distributed in α_1 (α_2) percent of the columns. Denoting the distance between these two histograms and the ideal histogram by d_{α_1} and d_{α_2} , the properly scaled version of f_{hist} is defined as

$$f_{hist}(I) = \begin{cases} 1 & \text{if} \quad d > d_{\alpha_1} \\ \frac{d-d_{\alpha_2}}{d_{\alpha_1}-d_{\alpha_2}} & \text{if} \quad d_{\alpha_2} < d < d_{\alpha_1} \\ 0 & \text{if} \quad d < d_{\alpha_2} \end{cases} \tag{5.9}$$

In our experiments we set $\alpha_1 = 0.1$ and $\alpha_2 = 0.8$.

5.1.6 The light direction term f_{dir}

This optional target term constrains the light direction to come from above. The input to this term are the elevations of the light sources, each given by the polar angle θ, as shown in Figure 5.2. The term simply measures the differences between each θ and a target polar angle θ_t. As suggested by Curran[7], a target angle $\theta_t = 45°$ was used in our experiments. The properly scaled f_{dir} term is defined as

$$f_{dir} = \begin{cases} \frac{d}{\alpha} & \text{if} \quad d < \alpha \\ 1 & \text{otherwise} \end{cases} \tag{5.10}$$

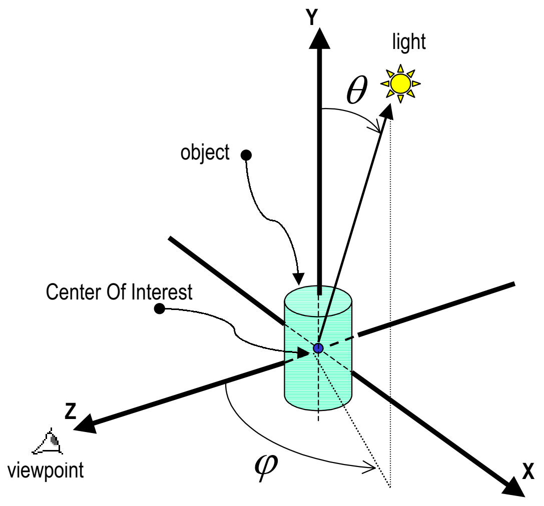


Figure 5.2: Spherical coordinate system for light source location.
Light location is defined in object-centered spherical coordinate system by the radius and two angles related to the vector connecting the center-of-interest (COI) with the light source location: the *polar angle* θ - angle from the positive Y axis $(0..\Pi)$, and the *azimuth* φ - angle from the positive Z axis to the projection on the X-Z plane $(0..2\Pi)$.

where d is the RMS difference between the elevations of the light sources and the target elevation θ_t:

$$d = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\theta_i - \theta_t)^2} \tag{5.11}$$

and α is a scaling constant such that for differences above α° the term always return 1.

5.2 Quality function evaluation procedure

Given the scene model, the viewing parameters and the rendering tool, the quality function is first calibrated by setting the target values and determining the weight of each component. Then lighting parameters can be set, and the scene is rendered by the rendering tool. The resulting image and lighting parameters are supplied to the quality function for quality evaluation. We next describe the evaluation procedure performed by the quality function.

The following data types are used:

***pim* - precomputed image map** Recall from subsection 5 that the PIM is a map classifying each image pixel as either *background*, *edge*, or *surface* pixel.

***img* - an image object** contains rendered image information such as dimensions and array of pixels with luminance values.

lights contains information about the illumination used to render the scene, and in particular an array of light source objects with the geometry of each light source used.

Target functions each target function (target term) is implemented as an object (*f*_grad, *f_edge*, etc.) that contains the data required for its computations (target values, scaling factors, etc.), and some functions. The functionality of these objects is designed such that they are first provided with relevant information during a loop that passes through the image (or light sources), and after all the information is accumulated, an evaluation of the final result can be performed.
For convenience, each target function object contains also its *weight* for the linear combination that constructs the quality function.

The function *Eval* receives the image and lights information as input, and performs the quality function evaluation using the data types described above. The function performs a single pass through the image, invoking for each pixel the relevant target functions operations (e.g. edge detector, etc.). The functions then collects the relevant lights geometry information, and finally calls for evaluation of each target function and returns the linear combination of the results:

```
float Eval(img, lights)
{
  // Loop over image pixels:
  N = img.width * img.height; //image is 1D array
  for (i=0..N)
  {
    switch(pim[i])
    {
    case EDGE:
      // Apply edge detection on pixel i and store result:
      f_edge.AddEdgePixel(img, i);
      break;
    case SURFACE:
      // Calculate shading gradient in
      // pixel i and store result:
      f_grad.AddShadePixel(img, i);
      break;
    case BACKGROUND:
      // Skip pixel:
      continue loop;
    }; //switch
```

```
      // Add pixel i to histogram.
      f_hist.AddPixel(img.pixel[i]);

    } // 'for' loop
    // Loop over light sources:
    L = lights.number_of_lights;
    for(i=0..L)
      // Not all light sources are
      // necessarily to be constrained by $f_{dir}$:
      if (constrain_light_dir(i))
        // Accumulate polar angle of light i:
        f_dir.AddAngle(lights.light_source[i].teta);
    f_grad.Eval();
    f_edge.Eval();
    f_hist.Eval();
    f_mean.Eval(f_hist); //use histogram information
    f_var.Eval(f_hist, f_mean); // use hist and mean info
    f_dir.Eval();
    return  f_grad.weight * f_grad.result +
            f_edge.weight * f_edge.result +
            f_hist.weight * f_hist.result +
            f_mean.weight * f_mean.result +
            f_var.weight  * f_var.result  +
            f_dir.weight  * f_dir.result;
  }
```

Notice that both objects 'f_edge' and 'f_grad' may employ operations involving local luminance gradients calculations. To avoid repeated calculation of the gradients at some image location, a gradient cache buffer is used to store gradients information after it was first calculated.

6 Lighting Design

The purpose of formulating the quality function described in the previous sections was to supply a comparative tool that can make quality evaluations of images rendered under different illumination conditions. In this section we utilize the quality function to construct a lighting design system: given a model of a 3D scene (including geometry and material properties), a set of viewing parameters for a desired image, and a rendering tool, the system uses the quality function as an objective function, and by an optimization process it can manipulate any set of lighting parameters to achieve illumination that best satisfies the function (i.e. minimizes the function).

6.1 Methodology

The lighting design system is first drawn as a framework comprises four major components: (i) scene model (including geometry, materials, viewing and lighting parameters), (ii) a rendering tool, (iii) the objective function (quality function) and (iv) the process control unit, that actually runs the lighting design algorithm by interacting with the other components. The process in a nutshell: the input to the process is the scene geometry, materials and viewing parameters; After configuring and calibrating the quality function appropriately for the given scene model, the set of lighting parameters to use should be specified. Each lighting parameter can be declared as either as fixed or free. The free parameters are those whose values we wish to determine automatically and in an optimal fashion using our system (searching variable). Next, the initial lighting is constructed such that the fixed parameters are given values which will remain constant in the course of optimization, and the free parameters are assigned "first guess" values; Finally, an optimization stage is performed in order to find settings for the free parameters that minimize the quality function. The core of the optimization is a loop of (i) setting lights parameters, (ii) rendering, and (iii) evaluating the quality function. This is repeated until function minimum is reached.

6.2 Process framework

Figure 6 describes the lighting design process and the interaction between the different components. The system comprises four main components which communicate throughout the process:

(1) Scene model This is basically a storage unit, which holds all the data about scene geometry, material properties, viewing parameters, and lighting parameters.

(2) Rendering tool A renderer that can receive the scene definition and render it. The result should be an accessible image pixmap in the computer's memory. The thick arrow connecting the Scene-model and the Rendering components in fig. 6 represents the constant communication channel between them: whenever a rendering should be performed, the scene data is used as input for the renderer.

(3) Quality function A quality function as described in section 4, that servers as the objective function of the process. The input for function evaluation is the image generated by the renderer, and light sources locations.

(4) Process control The main module that performs the lighting design process and integrates the operation of all components. This module should implement procedures for initializing and calibrating the quality function according to the scene information and the nature of the rendering tool, procedures for specifying the lighting parameters that participate the process, and optimization procedures. These procedures can be implemented either as automatic procedures, where the computer makes all the required decisions and sets parameters according to appropriate heuristics and calculations, or as semi-automatic, where the system interacts with the user and directed by his decisions or settings. In section 6.3 we give an implementation which is in practice a fully automatic system.

Following is an explanation about the stages and steps performed by the lighting design process, as described by figure 6:

Process input (step 1 in Fig. 6) The process is designed to searching the lighting space for a given scene geometry, materials and viewing parameters which are supplied as input to the process and kept fixed throughout the search. Texture mapping information is not received as process input, because — recall from the quality function formulation — texture mapping should not be rendered when the image is to be supplied as input to the quality function. The input may also include some predefined illumination settings, which should participate the rendering as they are given, and not changed by the process.
Another input parameter is the required image resolution. This is important, because the effect of illumination may vary with the resolution, due to the change in the actual size of scene features in the image. On low resolution images, for example, certain features or even entire objects may may not appear in the image (aliasing).

Lighting parameters (steps 2-4) Specifying the set of lighting parameters that will be used to render the scene. This may include any kind of illumination parameters supported by the scene model and the rendering tool, such as the number of light sources, their types, locations and intensities, etc. Each lighting parameter can be declared as fixed or free parameter. The values of fixed parameters are set only once, and not changed during the optimization stage. The free parameters will be used as optimization variables and will be manipulated by the optimization process in order to find their settings that best satisfies the quality function.
The values of all specified parameters should be chosen by trying to predict values that will best match the requirements of the quality function, and therefore these values should generally not be chosen arbitrarily, but rather computed by taking into account the scene data, the nature of the rendering algorithm, and the configuration of the quality function. The initial proposal to the optimization variables may significantly affect both the number of iterations required to reach solution, and the solution itself, which may be taken as the nearest local minimum.

Quality function configuration and setup step 5 - Weights: choose the weights for the linear combination of the quality function's component. In particular the weights of the light direction function f_{dir} which is considered as an optional component, and as a such its weight can be set to 0.
step 6 - Initial target values: set the initial target values for f_{var} , f_{mean}, and f_{dir}. These initial values express the desired image appearance. For example, higher target value for f_{mean} will lead to brighter

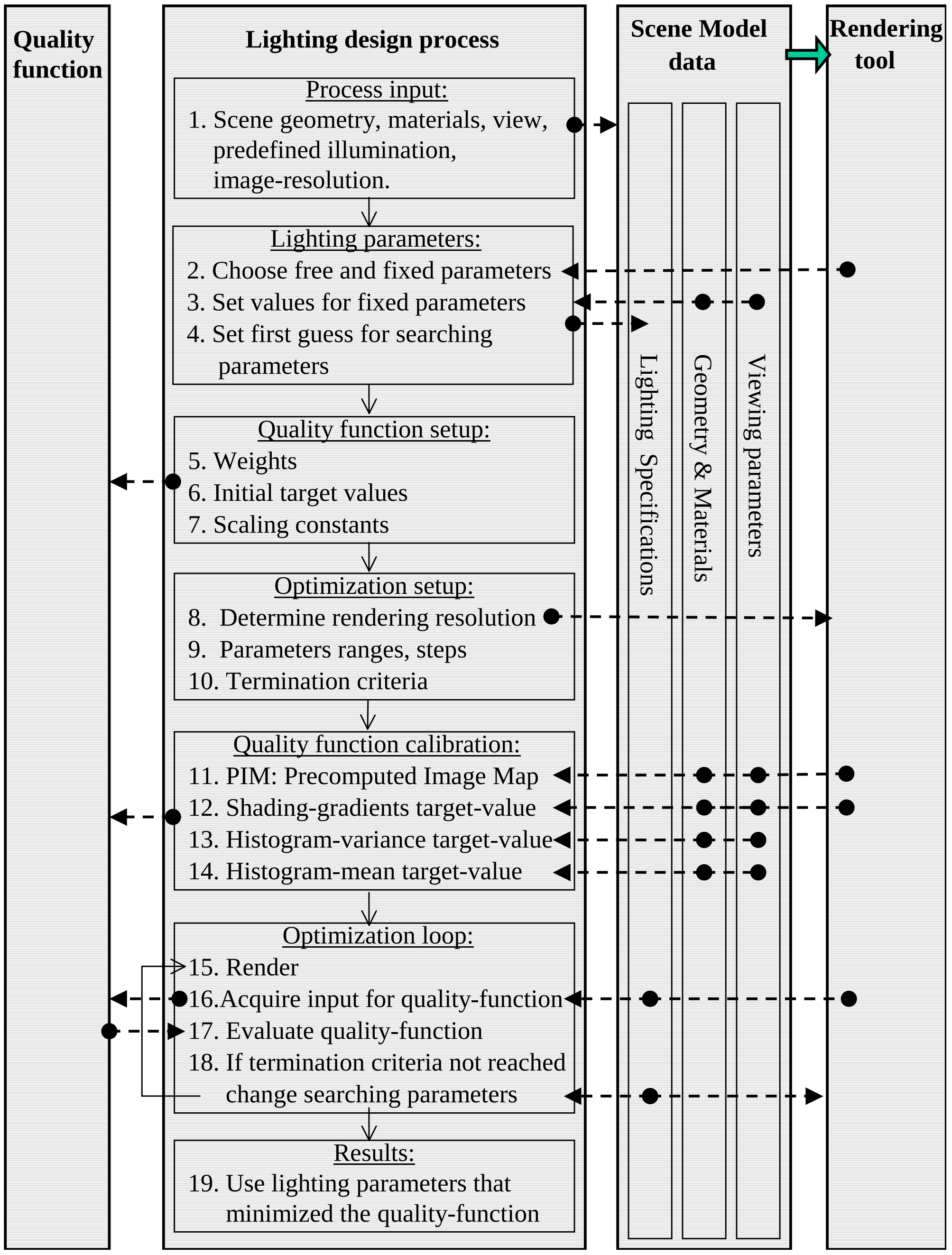

Quality function
Lighting design process
Scene Model data
Rendering tool
Process input:
1. Scene geometry, materials, view, predefined illumination, image-resolution.
Lighting parameters:
2. Choose free and fixed parameters
3. Set values for fixed parameters
4. Set first guess for searching parameters
Quality function setup:
5. Weights
6. Initial target values
7. Scaling constants
Optimization setup:
8. Determine rendering resolution
9. Parameters ranges, steps
10. Termination criteria
Quality function calibration:
11. PIM: Precomputed Image Map
12. Shading-gradients target-value
13. Histogram-variance target-value
14. Histogram-mean target-value
Optimization loop:
15. Render
16.Acquire input for quality-function
17. Evaluate quality-function
18. If termination criteria not reached change searching parameters
Results:
19. Use lighting parameters that minimized the quality-function
Lighting Specifications
Geometry & Materials
Viewing parameters

Figure 6.1: System framework.
The solid arrows describe the algorithm flow. The dashed arrows describe data transfer between the components, with the the round mark-points indicate the source of data and arrow-heads indicate the destination. The thick arrow connecting the Rendering and Scene components indicates the constant communication channel where each rendering is performed using the scene.

image. Another parameter that can be set here is the threshold values used by the edge operator of f_{edge}. These values may also affect image appearance in terms of the desired acuity of edges. Note that in this stage the values are set regardless of the particular scene being handled.
step 7 - Scaling constants: set the scaling constants of the target functions (see Section 5).

Rendering resolution and Optimization setup step 8 - Rendering resolution: the optimization stage involves repeated operations whose computational cost is directly related to the resolution of the image: rendering, converting image from RGB to luminance, Evaluating f_Q, and if the rendering is hardware-assisted, reading the resulting image from the frame buffer to the main memory (steps 15 to 17). On the other hand, it is often the case that performing the lighting design process with image resolution *lower* than the original resolution provides the same lighting solution, with significant reduction in computation time. In practice, there is a *threshold resolution* where scene features are still present in the image similarly to the original resolution, whereas lower resolutions introduce aliasing of scene features which affects the lighting solution found by the process, and makes it inappropriate for the requested resolution. Therefore, it is desirable to determine an image rendering resolution which is as close as possible to the threshold resolution mentioned above.
steps 9, 10 - Parameters settings and Termination criteria: Further setup of optimization factors. Termination criteria may include precision epsilon and other factors that determines tolerance of the searching process, etc.

Quality function calibration In this stage the scene data and the rendering tool are used to perform preliminary calibrations and computations that adjust the quality function to the scene and renderer in use.
step 11 - Precomputed image map: acquire the PIM that are used by the quality function in its evaluation operation, as described in subsection 5. Computing this map involves visible surface determination and extracting edges from scene geometry and materials.
step 12 - f_{grad} target: compute the target value for f_{grad}. As explained in Sec. 5, this value should be an approximation to the maximum average-shading gradients that can be measured under the given scene and rendering tool. For this purpose this step may involve a rendering operation.
steps 13, 14 - f_{var} and f_{mean} targets: as described in Sec. 5, the initial target values for f_{var} and f_{mean} (as were set by the previous stage) should be calibrated according to the scene materials and viewing parameters. That is, the inherent reflection and variance of scene materials should be should be considered, and the initial target values should be shifted accordingly. For accurate calibration, visible surfaces determination may be used.

Optimization loop Using the free lighting parameters as optimization variables and the quality function as an objective function, an optimization process is performed to find the optimal settings for the lighting parameters, with respect to the function. The process is an iterative search in the multidimensional space of the free variables, looking for a minimum of the quality function. At each iteration,

the following steps are repeated:
step 15 - Render: rendering the scene model, using the current lighting specifications.
step 16 - Acquire input: read the pixmap of the rendered image (if needed convert it to luminance representation), get lights geometry from the scene model, and transfer to the quality function.
step 17 - Evaluate quality function.
step 18 - Change the optimization variables in order to advance the search towards minimum. The implementation of this step depends on the particular optimization technique chosen. When the optimization variables are restricted to light sources positions, directions, and spectral distribution, the quality function behaves in a continuous manner and a gradient-based technique can be used, such that at each iteration a search direction is set based upon evaluation of the local gradient, and the solution then moves in that direction. The situation is more complex if the optimization should also manipulate parameters such as the number of light sources and their types. Here, a several steps technique that alternately manipulates sets of parameters should be considered.

In our experiments we have found that the quality function may have several local minima, and the solution typically converges to the minimum nearest to the initial guess. Unfortunately, putting much effort in searching the global minimum may dramatically increase the cost of the process, and therefore expending more effort in establishing the initial proposal and accepting the nearest minimum as the final lighting solution can be a reasonable compromise.

6.3 Implementation

In this section we describe an implementation of the system described above. It should be emphasized that any full implementation cannot avoid from using heuristics and procedures that are specific to the rendering algorithm and the lighting model. In addition, the implementation of some procedures, including some procedures used by the quality function, is not unique, and several techniques may be considered and implemented to perform the task required. In particular, we used a very simple and naive optimization technique, which may need to be further enhanced to improve system performance.

6.3.1 Scene and illumination model specifications

Rendering: we use a straightforward OpenGL-based rendering tool: Z-buffering for hidden surface removal, and OpenGL's default shading model [33].
Scene geometry and view: scenes are given by a polygonal mesh, with per-vertex normals. Each polygon can be assigned different material properties. The system assigns each polygon a unique ID number. Viewing parameters are specified using OpenGL's camera model: camera is located at the *viewpoint* and looking at the scene's *center-of-interest* (COI).
Lighting specifications: light is defined by the three components *Diffuse*, *Specular*, and *Ambient*, with each has a spectral distribution defined by R, G, and B channels.

Two types of light sources are considered: positional light source, described by its location in the 3-D space and the spectral distribution of the light components, and directional light source, described by a direction in the 3-D space and spectral distribution (a directional light source is equivalent to a positional light source at “infinity”). Further in this text the term “position” will be used both for location of positional light and direction of directional light.

We use spherical coordinates to describe light sources positions, with a coordinates system where the center-of-interest (COI) is the origin, and the location the viewpoint direction is given by polar angle $\theta = 90^o$ and azimuth angle $\varphi = 0^o$ (see figure 5.2). Consequently, the location of a positional light is defined by the triple $(r,\ \theta\ ,\varphi)$ and direction of directional light if defined by the pair $(\theta\ ,\varphi)$.
Materials: in correspondent to the description of light, materials are described by the three components Diffuse, Specular, and Ambient, and a material roughness factor to describe the behavior of the specular component (Phong exponent). No texture maps were used.

The scene object supports the following functions, which receive a polygon’s ID number as argument:

- Returns the normal of that polygon:

  ```
  Normal getPolygonNormal(int polygon_id);
  ```

- Calculates some approximation to *polygon intensity* by transforming RGB representation to Luminance (using the standard transformation $L = 0.3R + 0.59G + 0.11B$), and perform some averaging on material coefficients:

  ```
  float getPolygonIntensity(int polygon_id);
  ```

6.3.2 Reducing the number of free parameters

An elementary operation in the process is declaring a “free light source”, which basically refers to adding to the illumination a new set of light source specification parameters, and use them as optimization variables. However, generally each free lighting parameter significantly contributes to the computation time of the optimization stage, since approximating the quality function’s partial derivative with respect to that variable (as part of gradient calculations) involves an extra rendering and quality function evaluation. Therefore, we have investigated the tradeoff between the desire to search the whole lighting parameters space and the need to keep the process as cheap as possible. After examining various test cases we have reached the conclusion that some light source parameters can be regularly defined as *fixed parameters* (i.e. not participate the optimization process), and provided they are initialized by appropriate heuristics, they do not significantly divert the final illumination solution. These parameters are as follows:
Light source type and position: all free light sources are defined as *positional* lights, with a *fixed radius*. Positional lights were found to generally yield illumination that better matches the requirements of the quality function, compare to directional lights. The radius value should be calculated

from the dimensions of the scene and the COI: denote the COI as coi and V the set of scene vertices. Let $d = max(distance(coi, v_i))$ where $v_i \in V$, then the radius is $r = 2d$. This value was found suitable for the vast majority of scenes examined.
Light source spectral distribution: all light components of a light source are assigned white light values only. That is, the R,G,B channels of each component are constrained to have the same value: $R = G = B$. This guarantees that the scene surfaces are rendered according their original material definitions, without any bias caused by colored light source. In fact, since the quality function operates in the luminance domain of the image, rendering under colored illumination is practically meaningless for its behavior.
Light source ambient component: changing the ambient component independently of the other lights components simply causes to a shift in the histogram of the image, which directly change the f_{mean} component of the quality function, but causes no "real" change in illumination patterns. In practice this can result in redundant iterations around the target mean intensity, which can be reduced when the ambient intensity becomes a function of the diffuse intensity, and instead of using both as optimization free variables we fix the ambient intensity at 15 percent of the diffuse intensity: $ambient = 0.15 * diffuse$.

Consequently, each free light source in our system involves 4 free parameters: the two direction parameters $(\theta\,, \varphi)$, and the parameters of diffuse and specular intensities.

6.3.3 Preliminary geometric computations

In order to calculate initial lighting parameters values (steps 3-4 in fig. 6), derive the PIM (step 11), and calibrate target values (steps 12-14), the following geometrical data-structures are constructed by using the rendering tool and the scene model:
(1) Z-Buffer: the buffer generated by the Z-buffer hidden surfaces removal algorithm.

(2) Item-Buffer: a buffer created by rendering the scene with no lighting calculations, but rather giving each pixel an unique ID value identifying the polygon rendered at that pixel. These values are supported by the functions defined earlier for the scene model.

The Item-Buffer supports the following functions:

- A function that calculates an approximation of the average intensity of the visible surfaces. Denote the Item-Buffer as *iBuff* and the Scene model as *scene*:

```
float getAverageIntensity(){
  float acc = 0;
  int counter = 0;
  for (i=0 .. iBuff.size)
```

```
      if (iBuff.value[i] != BACKGROUND){
        acc += scene.getPolygonIntensity(iBuff.value[i]);
        counter++;
      }
    return acc / counter;
  }
```

- A function that calculates an approximation of the standard deviation of the visible surfaces:

```
  float getSTDDev(){
    float acc = 0;
    int counter = 0;
    float mean = getAverageIntensity();
    for (i=0 .. iBuff.size)
      if (iBuff.value[i] != BACKGROUND){
        acc +=
          (mean - scene.getPolygonIntensity(iBuff.value[i]))^2;
        counter++;
      }
    return (acc / counter)^(1/2);
  }
```

(3) Diffuse-Directions Map (DDM): a unit sphere, that holds 3-D mark-points on its surface, corresponding to the normals of scene's surfaces. It is constructed by using the Item-Buffer: for each pixel in the item-buffer, add a 3-D point on the sphere's surface, such that this point corresponds to the direction of the normal to the surface rendered at that pixel. The normals directions are expressed in terms of the two angles (θ, φ), according to fig. 5.2.
Let us define $SphereDir$ as structure containing the two angles (θ, φ):

```
  struct SphereDir{
     float teta, phi;
  };
```

Denote the Item-Buffer as $iBuff$, let $scene$ be the scene model object, then the construction of this data-structure is given by the loop:

```
  for (i=0 .. iBuff.size)
    if (iBuff.value[i] != BACKGROUND){
      Normal n = scene.getPolygonNormal(iBuff.value[i]);
      addPoint(n.getTeta(), n.getPhi());
    }
```

Each such point on the direction sphere is simply referred to as *Direction*. Each direction in the DDM can be treated as the potential position of a light source for maximum diffuse illumination at the corresponding pixel.

The DDM support the following operation:

- Cluster the directions on the sphere to *n_clusters* clusters, and return the centroid of cluster number *clust_num* :

```
SphereDir getCluster(int n_clusters, int clust_num);
```

(4) Reflection-Directions Map (RDM): another directions sphere, only now for each pixel in the item-buffer the directions added to the sphere is the ideal-reflection direction, calculated by considering the *viewpoint*, the *normal* and the *Z value*, at that pixel. The construction of this data-structure is given by the loop:

```
for (i=0 .. iBuff.size)
  if (iBuff.value[i] != BACKGROUND){
    Normal n = scene.getPolygonNormal(iBuff.value[i]);
    float z  = zBuff.value[i];
    SphereDir dir_point = CalcReflection(viewpoint, n, z);
    addPoint(dir_point);
  }
```

Each direction in the RDM can be treated as the potential position of a (directional) light source for maximum specular illumination at the corresponding pixel.

The RDM supports the operation:

```
SphereDir getSolidAngleDir(int rate);
```

- given some solid angle patch (expressed by area on the unit sphere surface), return a direction such that the patch located on the surface of the direction sphere, centered at that direction, is rated as the *rate*'th patch that contains highest number of directions (where *rate* is the function's argument). That is, to find the direction with rate 1, all possible locations for solid angle patch on the surface of the sphere are considered, and the one containing maximum number of directions is selected. To find the angle rated next, the process is repeated but with no overlap allowed with previously located patches. The size of the solid angle used by this function is calculated from material information of the visible surfaces, such that if a directional light source is located at any one of the directions contained in a solid angle patch of that size, the pixels corresponding to all directions in the patch are still "significantly" affected by the specular component of that light.

6.3.4 Initializing free variables

As explained in sec. 6.3.2, for each free light the parameters that are considered as free variables are the direction of the light source in terms of the angles (θ, φ), and the intensity of its diffuse and specular components. The initial values for these parameters are calculated by using the scene information and the geometric data structures constructed in sec. 6.3.3.

Light direction

Given that the system has N free lights, the initial values for the n-th light, $0 < n < N$ are determined as follows:
The light direction is calculated by first using the Diffuse-Directions-Map (DDM) to get the centroid of cluster n , and then using the Reflection-Directions-Map (RDM) to find a reflection direction in the "neighborhood" of that centroid. The idea is to direct illumination on the largest region in the scene not yet being illuminated by previous lights, and to fine-tune this illumination direction as to capture specular effects that can occur as result of illuminating from this approximate direction. The following procedure demonstrates this idea:

```
SphereDir CalcInitialDir(int light_num)
{
  SphereDir diffuse_dir = DDM.getCluster(light_num);
  SphereDir reflect_dir;
  const MAX_DIRS = 8; //empirical value
  float min_angle = 180; //degrees
  for (i=0 .. MAX_DIRS){
    float curr_angle =
      angle(RDM.getSolidAngleDir(i),diffuse_dir);
    if (curr_angle < min_angle){
      reflect_dir = RDM.getSolidAngleDir(i);
      min_angle = curr_angle;
  }
  return 0.5*diffuse_dir + 0.5*reflect_dir;
}
```

In practice the weights used to calculate the result direction should be more carefully considered, and the final direction should not be diverted by more than about 45 degrees from the initial diffuse direction.

Light intensity

The initial light intensities (diffuse and specular components) are calculated for all free lights considering the target value of f_{mean} the intensities of visible surfaces materials, and the number of lights sources. Denote the Item-Buffer as *iBuff*:

```
float CalcInitialIntensity(int n_lights, float tar-
get_intensity)
{
  float average_material = iBuff.getAverageIntensity();
  float single_light_intensity =
        target_intensity / (0.65 * average_material);
  return 1.25 * single_light_intensity / n_lights;
}
```

6.3.5 Constant initializations

Target terms weights: We used the following empirically determined weights:
$w_{grad} = 0.6$

$w_{edge} = 0.6$

$w_{var} = 0.5$

$w_{man} = 0.4$

$w_{hist} = 0.55$

$w_{dir} = 0.5$ (or 0, when f_{dir} is disabled).

Initial target value: For pixels intensity in the scale [0..255], the following values were used:

- $t\_f_{var} = 42$;
- $t\_f_{mean} = 156$;
- $t\_f_{dir} = 45^o$

The final value of the first two target values may be corrected according to scene materials properties (see next subsection).

Rendering resolution Constructing a heuristic that finds a good approximation to the *threshold resolution* (mentioned in Section 6.2) for each given scene model and viewing parameters is a topic for future work.

Having examined all of our test models, we were able to choose some common resolution, which is the smallest resolution that still yielded a satisfactory solution for all test cases. This resolution was found to be of around 62,000 pixels (e.g., 250 x 250). Therefore, in all our experiments we reduced the desired image to 62,000 while preserving its aspect ratio, and performed the optimization at that resolution.

6.3.6 Quality function calibration

PIM The precomputed image map is derived by using the preliminary geometric data-structure computed in sec. 6.3.3 as follows: first the Background pixels are derived directly from the Item-Buffer; the Edge map is derived by using both the Z-buffer ([29]) and Item-Buffer to account for both feature and reflectance edges, and the remaining pixels are marked as surface pixels.

f_{grad} **target value** To approximate the target value of f_{grad} , a single rendering is performed under "extreme" illumination conditions, according to the following procedure:

1. Set a positional light with the following specifications:
   direction = average normal of all visible surfaces (achieved by using the *Diffuse-Directions Map* from sec. 6.3.3);
   radius = 1.5 * the radius calculated in section 6.3.2;
   diffuse_intensity = MAX_INTENSITY - *average_surface_diffuse*, as can be derived from the ItemBuffer (6.3.3);
   specular_intensity = ambient_intensity = 0;

2. Set a directional light source with the following specifications:
   direction = the direction that has most effect on the specular component of the visible surfaces. This is yielded by: $RDM.getSolidAngleDir(0)$, where RDM is the *Reflection-Directions Map* from sec. 6.3.3.
   specular_intensity = *MAX_INTENSITY* - *average_surface_specular*, as can be derived from the ItemBuffer (6.3.3);
   diffuse_intensity = ambient_intensity = 0;

3. Render and read image.

4. Evaluate the f_{grad} component of the quality function.

5. Use $1.25f_{grad}$ as the target value.

f_{var} and f_{mean} target values: Correct the target values according to the materials properties of the visible surfaces. Denote *iBuff* as the Item-Buffer, then:

- $t\_f_{var} = t\_f_{var} + 0.15 * iBuff.getSTDDev();$
- $t\_f_{mean} = t\_f_{mean} * (0.55 + 0.45 * iBuff.getAverageIntensity())$;

6.3.7 Optimization

In our current implementation we use a simple steepest descent optimization scheme [24]. To approximate the gradient of the quality function, partial derivatives for the free variables must be computed. We approximate partial derivatives numerically, by taking differential steps at the direction of each free variable, rendering, and evaluating the quality function. Once an optimization direction is chosen, the solution advances in that direction until reaching a minimum, and then a new direction is calculated. When optimizing over more than one free light source, the optimization process alternates its steps between the different light sources.

6.3.8 Using the f_{dir} component

As described in section 4, the quality function's term f_{dir}, which is used to constrain illumination to come from above, is considered as an optional component. In a lot of cases, lighting results achieved without using the f_{dir} component (weight = 0) are satisfactory. When activating this component, however, results often suffer from lack of illumination in the "low" regions of the scene. Therefore, whenever the f_{dir} component is on, a minor fixed light source is added to supply some extra weak illumination, mainly intended to help illuminating those low regions. More specifically, the following two options are adopted by the system:

1. f_{dir} disabled (weight=0) - here all participant lights are free lights which are optimized with no constraints imposed on their directionalities.
2. f_{dir} enabled (weight=0.5) - in this case a fixed light source is added, and positioned at the viewpoint ($\theta = 90^o$, $\varphi = 0^o$), with some weak intensity used for its diffuse component only. Other light sources are free lights and optimized under the constraint imposed by f_{dir}.

The system is fully configured by determining the exact number of free lights for each of the two cases above.

6.3.9 Full automation

The previous subsections describe the constants, heuristics and procedures used to setup the system. One substantial parameter that still needs to be handled is the number of free lights involved in the optimization process. One approach could be to leave this parameter as an optimization variable, and let the optimization process choose the number of lights yielding the best result. By using the current system design this could be performed by the following procedure:

```
1. set number of free lights $n = 1$;
2. optimize;
   keep lighting parameters results as $L_1$;
   keep quality function result as $F_1$.
3. repeat{
     $n = n + 1$;
     optimize;
     keep results as $L_n$ and $F_n$;
   } until $F_n \geq F_{n-1}$;
4. accept $L_{n-1}$ as final lighting results;
```

This procedure, however, is highly expensive due to the repeated optimizations that should be performed, with each newly added free light significantly increase optimization time. Therefore, another approach is to determine the number of free lights before the optimization stage, and keep it as a fixed parameter. A fully automatic system should try to establish some heuristic to make this decision.

Hence, in order to fully configure the system such that all parameters are set and optimization can be performed two decisions are left to be made:

1. For free light: which of the two options for using f_{dir} (as described in subsection 6.3.8) should be used.
2. How many free lights should be optimized.

In practice, for all the test cases examined, we found that the problem can be reduced, and considering only three *full system configurations* is enough to achieve adequate results:

Configuration1 f_{dir} disabled, and a single free light is optimized according to the specifications given earlier in this section.

Configuration2 f_{dir} enabled, one free light is optimized and a secondary light is fixed (as described in the previous subsection).

Configuration3 f_{dir} is disabled, and two free lights are optimized.

Hence, the only decision that should be made by the system is which of this configurations should be used. Cases where f_{dir} is crucial to the illumination are rare, and its effect is more to direct the solution to a different image appearance, which may be considered preferable by some perceptual considerations (see sec. 3.5), even if not solving any crucial visual issue. Using two free lights is often not required, and may even lead to less satisfactory results than the two other configuration. As a rule, two primary lights should be used only when illumination by a single primary light is found to be inadequate (recall that scenes illuminated by a single light are more easily and naturally interpreted by human viewers). By this stage of our work, we did not establish a heuristic that automatically determines which configuration should be used. Rather, the procedure used is to find solutions by using the first two configurations, and only if both seem to be inadequate, run the system with the third configuration.

6.4 Multiresolution

Unsurprisingly, the optimization loop is the main responsible for computation cost of the process. The performance of each loop iteration is directly affected by two main factors: the resolution of the image and the performance of the rendering tool. As discussed in the "Optimization setup" stage above, the rendering resolution during the optimization search can be reduced towards some threshold resolution which still preserves the illumination effects occurs in the original resolution.

If working with scenes which requires high threshold resolution, or if the rendering algorithm is highly expensive for the threshold resolution, a *multiresolution* optimization framework can be found beneficial. With this technique several intermediate low resolutions are determined, and the optimization is first performed at the lowest resolution, then moves to the next level with the previous solution taken as the initial proposal.

Our experience shows that performing a multiresolution optimization process significantly reduces the number of optimization steps performed in the highest resolution, at the expense of an increase in the total number of steps which are mostly performed in low resolutions, plus a certain overhead caused by a partial need to re-calibrate the system when moving from one resolution to another.

7 Results

We used the system described in the previous section to automatically determine the lighting design in a large number of test scenes, with a variety of different materials and different viewing parameters. In the vast majority of cases the lighting designs produced by the system were found satisfactory in terms of the visual quality goals we set to ourselves in Section 3. We also found these designs to favorably compare with the best designs we were able to generate using manual manipulations of the lighting parameters (the direct design paradigm mentioned in Section 1).

To illustrate the effectiveness of our lighting design system, we compare our lighting solutions with naive lighting settings which do not take into account any particular knowledge about neither the scene model being illuminated nor the viewing parameters. We chose to use two approaches commonly used for setting default illumination:

1. Locate a single directional light source at the viewpoint ($\theta = 90^o$, $\varphi = 0^o$), producing monochromatic light at some average intensity. This setting is denoted as **Default1**.
2. Locate four directional light sources, located below and above the viewpoint, as follows: (i) above-left: $\theta = 45^o, \varphi = -45^o$, (ii) above-right: $\theta = 45^o, \varphi = 45^o$, (iii) below-left: $\theta = 135^o$, $\varphi = -45^o$ and (iv) below-right: $\theta = 135^o$, $\varphi = 45^o$. This setting is denoted as **Default4**.

In all the examples brought here the default illuminations suggested above produced unsatisfactory results, which were significantly improved by the automatic system. The original image resolution was chosen according to size of scene features.

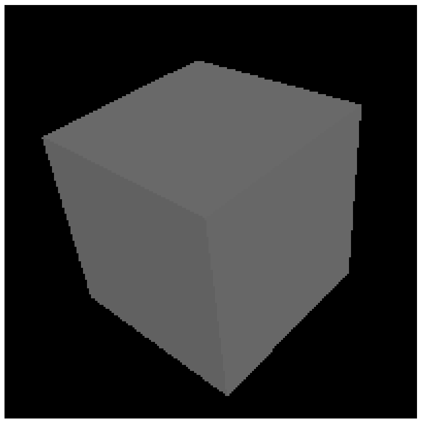
All examples contains images rendered under the two default illuminations - **Default1** and **Default4** - and images rendered under illuminations chosen by the optimization process using **Configuration1** and **Configuration2**, as defined in 6.3.9. In cases where using the 2 free lights (**Configuration3**) produced an improved result over these two, that result is given too.

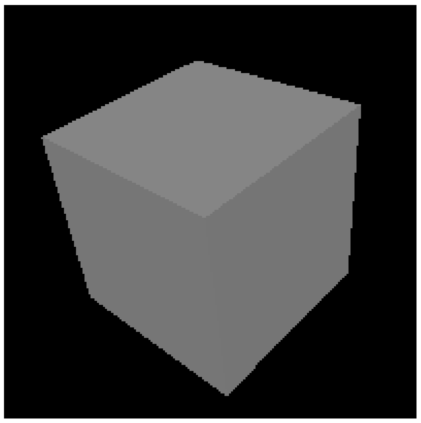
Table 1 specifies the performance of the process for all example cases. Each scene model in the table has a corresponding figure with the set of images as described above.

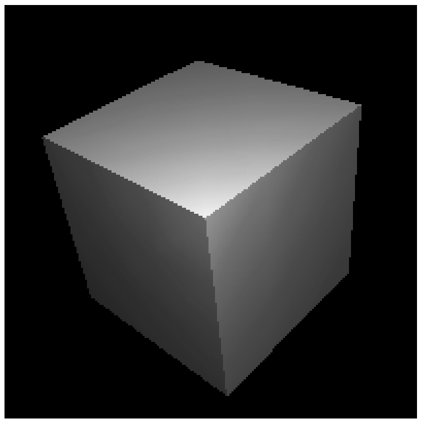
Table 2 describes the a typical distribution of the operation times of the major process procedures. The numbers are strongly depend on whether the rendering was performed with or without hardware support: on the one hand, when hardware support was used, rendering time is significantly improved, but on the other hand reading the frame-buffer from the graphic adapter (rather than from processor memory) is significantly slowed down (this was found to be true for all adapters we tried).

Model name	# of polygons	Resolution	Figure	# of optimization iterations	# of renderings performed	Optimization time (sec)
Cube	54	200x200	7.1c	9	38	2.0
			7.1d	13	47	2.4
Fork	812	167x117	7.2c	12	41	2.2
			7.2d	7	36	2.1
Cow	5,805	186 x 208	7.3c	11	39	3.5
			7.3d	4	20	2.3
Baby	12,712	193 x 318	7.4c	5	21	3.2
			7.4d	5	25	3.6
Cherries	824	238 x 293	7.5c	5	24	2.8
			7.5d	5	22	2.5
Dagger	4,981	448 x 173	7.6c	7	29	3.9
			7.6d	7	27	3.7
Sword	1,520	339 x 220	7.7c	7	30	4.1
			7.7d	4	40	4.2
			7.7e	6	48	4.9
Foot	4204	267x199	7.8c	6	28	3.9
			7.8d	4	24	2.4
Air-boat	6686	695 x 379	7.9c	7	31	4.1
			7.9d	5	27	3.6
Galleon	4,699	362 x 389	7.10c	5	25	3.3
			7.10d	8	35	4.8
			7.10e	9	57	6.5
Paddle boat	11,435	492 x 388	7.11c	11	39	5.0
			7.11d	5	22	4.0
			7.11e	5	48	8.3

Table 1: Results of the lighting design process for various scene models. For each scene the process was performed using the system configurations described in sec. 6.3.9. Images of the resulting rendering of each model are brought in a corresponding figures (as specified in the table). Times were measured on a 866 MHz Pentium III PC with an Nvidia GeForce2 graphics accelerator.


(a) Default - 1 light source


(b) Default - 4 light sources


(c) Optimized - Configuration1

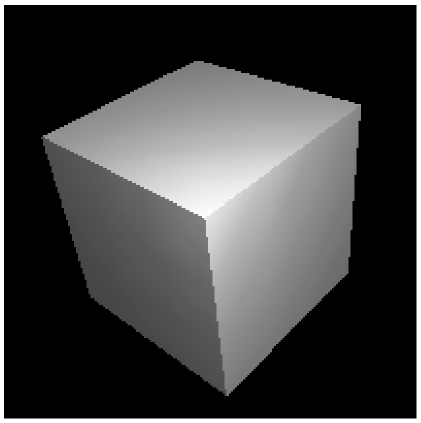
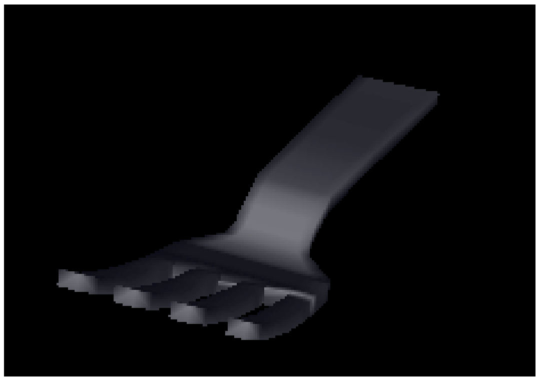
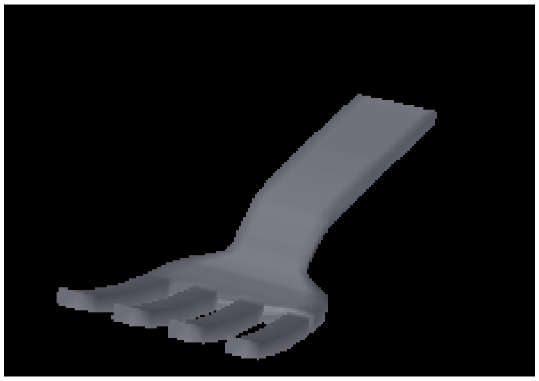

(d) Optimized - Configuration2

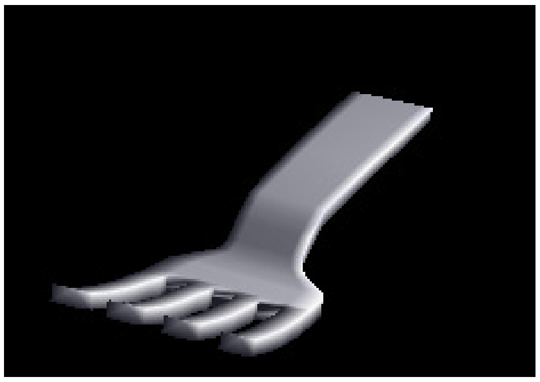
Figure 7.1: Cube.
Default illuminations: both (a) and (b) introduce totally uniform shading on cube faces and fails to display all edges.
Optimized illuminations: (c) all edges are prominent; shading gradients on cube faces convey enhanced 3-D impression.



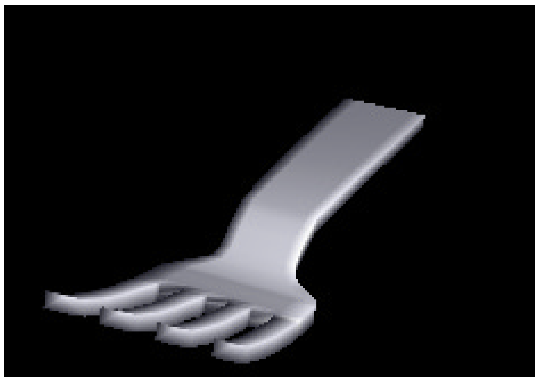
(a) Default - Configuration1



(b) Default - 4 light sources

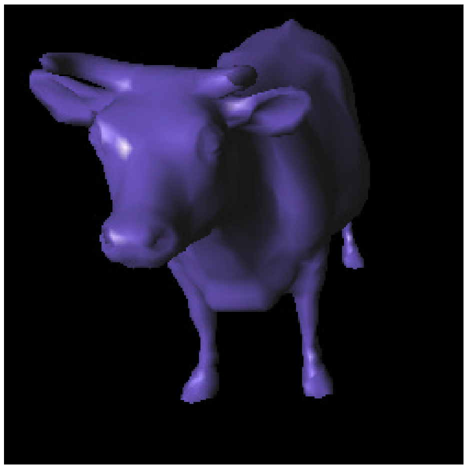


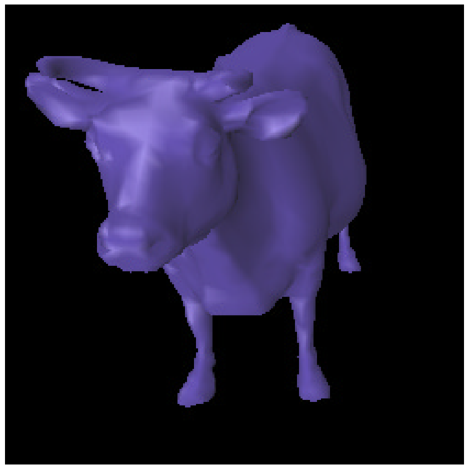
(c) Optimized - 1 light source

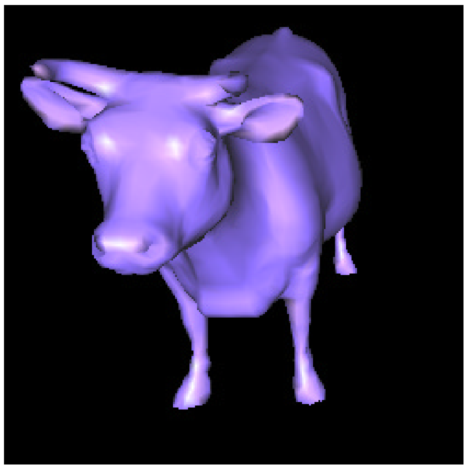


(d) Optimized - Configuration2

Figure 7.2: Fork.
Default illuminations: (a) and (b) fails to capture material and shape information in several locations mainly due to lack of illumination (a) and too flat patterns (b).
Optimized illuminations: (c) introduce major improvement by both highlights and shadows; (d) same improvements with illumination coming from above.


(a) Default - Configuration1


(b) Default - 4 light sources


(c) Optimized - Configuration1

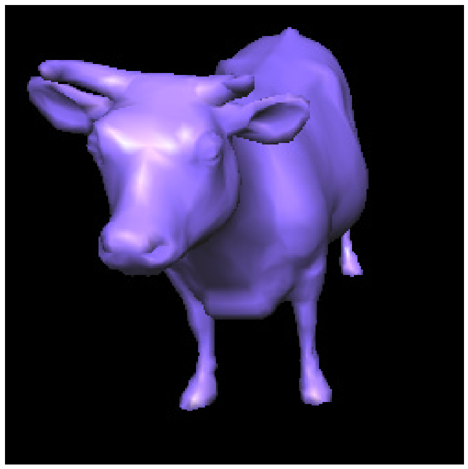
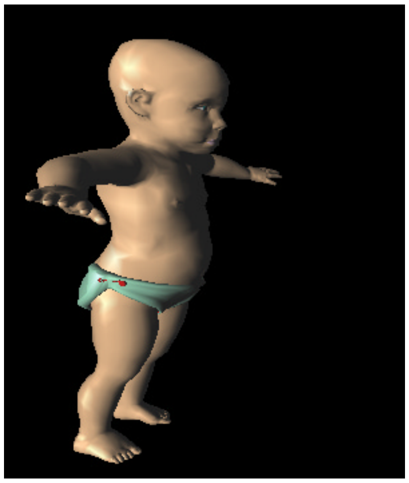
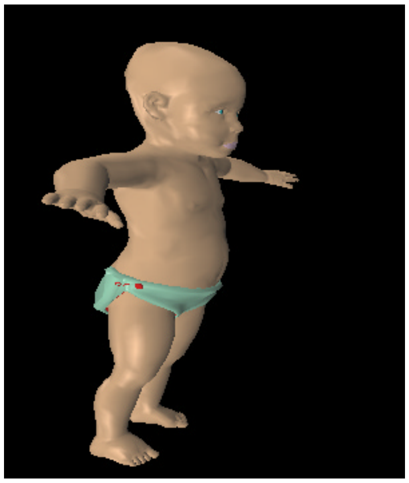
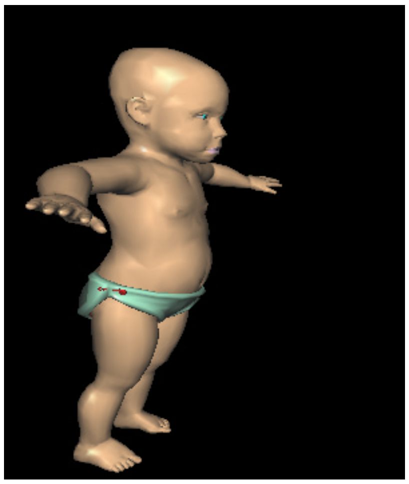

(d) Optimized - Configuration2

Figure 7.3: Cow.
Default illuminations: (a) shadowed regions hide shape information, and image is generally too dark, and (b) flat illumination lacking of shading patterns - both shape and edges are lost.
Optimized illuminations: (c) introduce major improvement in general brightness; All shape features are visible. (d) similar qualities with illumination coming from above;


(a) Default - Configuration1


(b) Default - 4 light sources


(c) Optimized - 1 light source

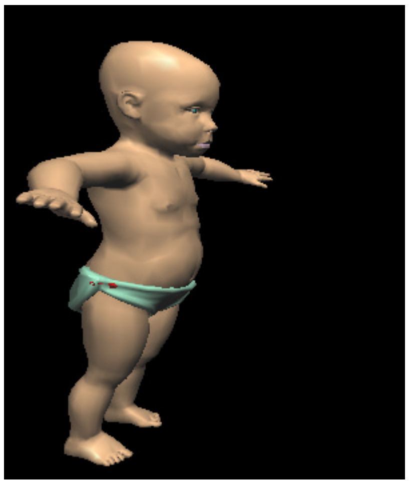
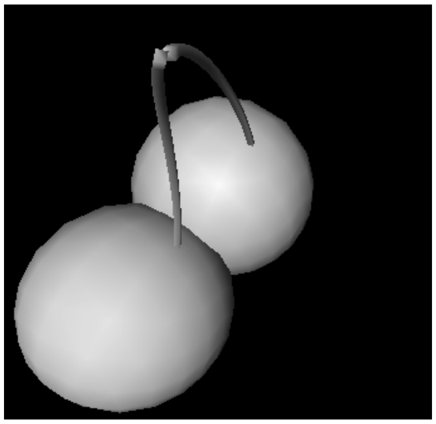
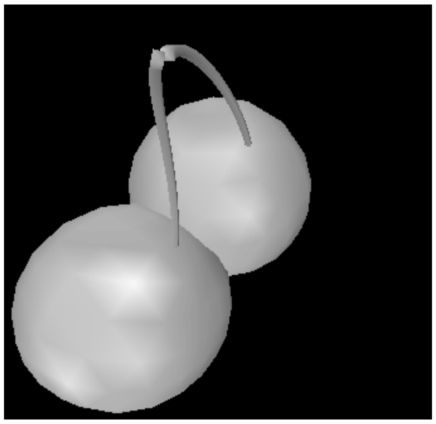
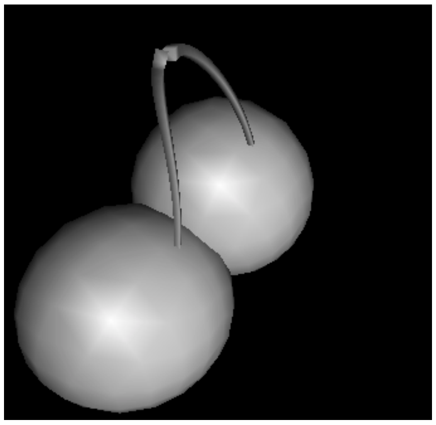

(d) Optimized - Configuration2

Figure 7.4: Baby.
Default illuminations: (a) body shape and details are lost in the shadowed far side and in the near arm and hand; (b) too much illumination yields in blurred appearance.
Optimized illuminations: (c) and (d) all body parts are properly illuminated and deficiencies from default cases are removed.


(a) Default - Configuration1


(b) Default - 4 light sources


(c) Optimized - 1 light source

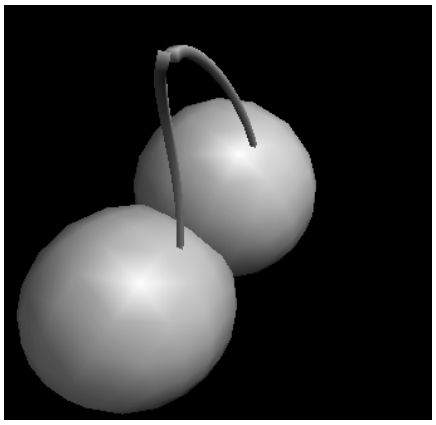
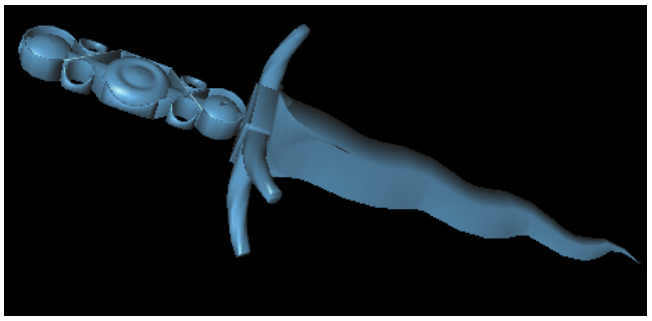

(d) Optimized - Configuration2

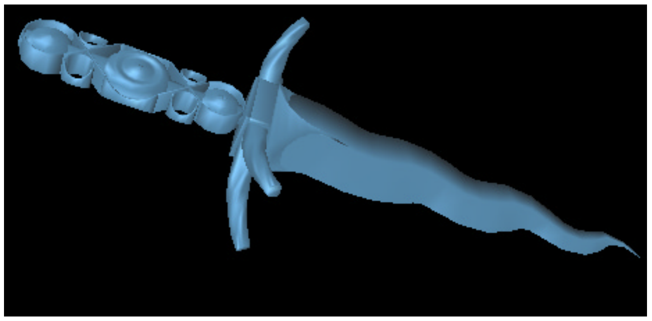
Figure 7.5: Cherries.

Default illuminations: (a) front cherry lacks highlight, while highlight on back cherry is very dim; Both cherries somewhat lack shading gradation and 3-D appearance is of limited prominence; Some edges around the contact point of front cherry and stem are invisible; Stems appear dark and hardly any shape information is conveyed. (b) “contradictory” illumination, where 3-D impression is lost due to the excess in illumination directions; Some edges between the cherries and between front cherry and stem are not prominent enough or even invisible; Shape information of stems is lost due to too uniform shading.

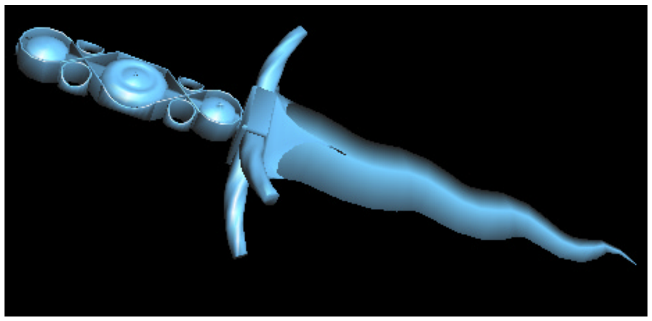
Optimized illuminations: (c) highlights and shading gradation on both cherries and stems convey shape information and creates vivid 3-D impression; All edges are prominent; (d) shading on stems suffer from the fact that primary illumination arrives from above ($\theta = 50^{\circ}$) and is somewhat deficient, but still improved relatively to default cases; cherries have similar qualities to (c).



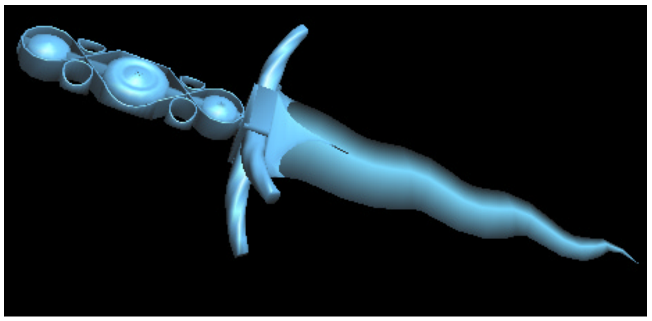
(a) Default - Configuration1



(b) Default - 4 light sources

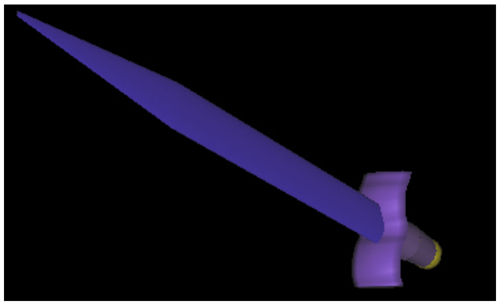


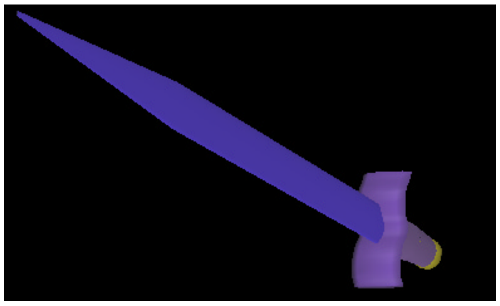
(c) Optimized - 1 light source

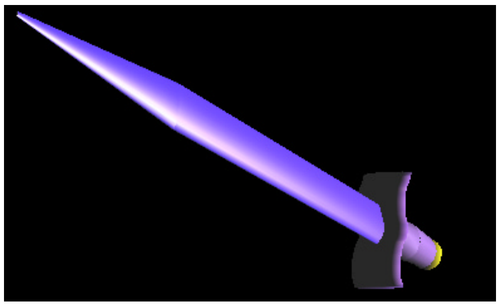


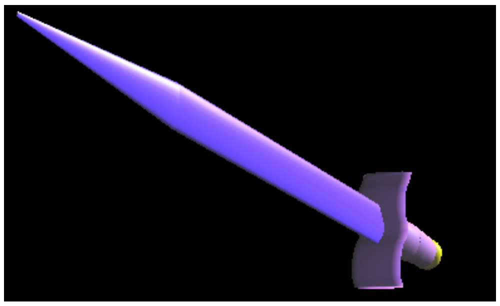
(d) Optimized - Configuration2

Figure 7.6: Dagger.
Default illuminations: (a) and (b) shape can be perceived although illumination on blade is too flat and can be enhanced; some details are missing in the base of the blade and shield;
Optimized illuminations: (c) and (d) perception of blade shape and material is enhanced, details missing in previous images are more prominent.


(a) Default - Configuration1


(b) Default - 4 light sources


(c) Optimized - 1 light source


(d) Optimized - Configuration2

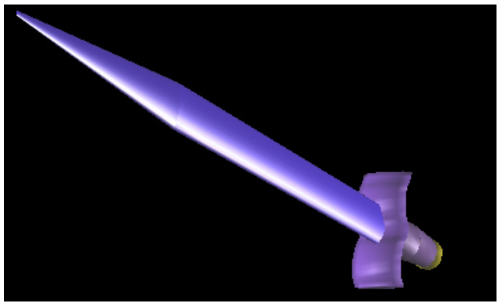
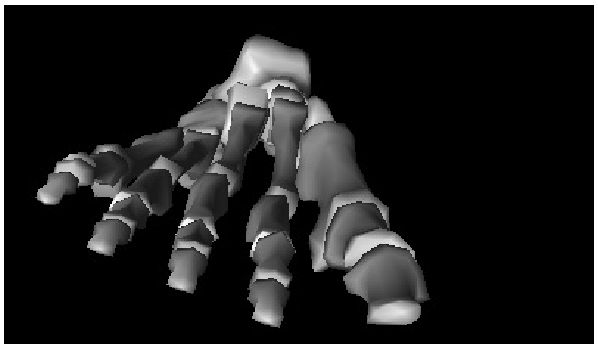
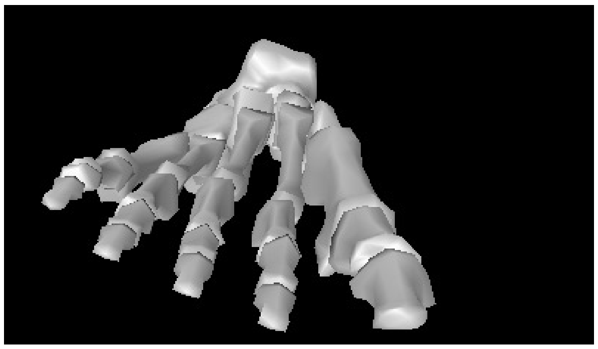

(e) Optimized - Configuration3

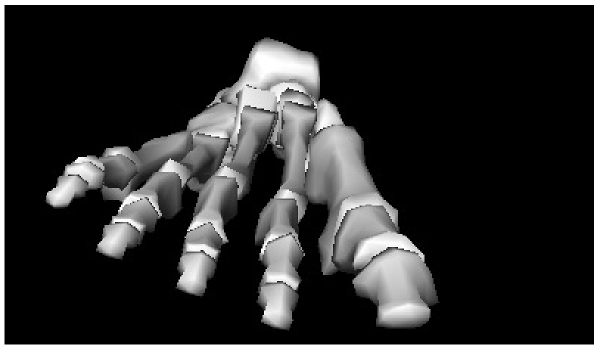
Figure 7.7: Sword.

Default illuminations: (a) and (b) introduce almost uniform shading on both the blade and hilt, and fails to communicate shape and material information, as well as some fine details on the hilt. Edges between hilt and shield are insufficiently prominent.

Optimized illuminations: (c) introduce major improvement in blade and hilt illumination, but at the cost of neglecting the shield, where shape information is lost in the dark shade; (d) suffers from the same deficiency, although more light reaches the shield; Adding a second optimized light (e) enables proper illumination on the shield too.


(a) Default - Configuration1


(b) Default - 4 light sources


(c) Optimized - 1 light source

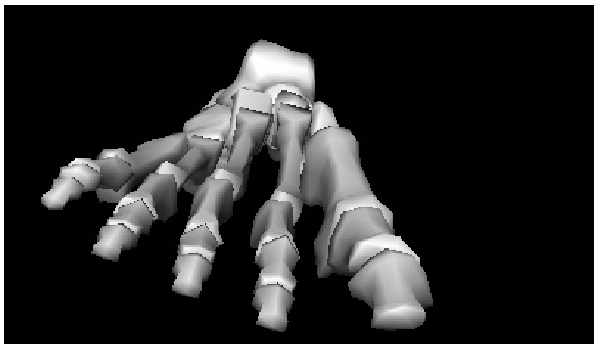
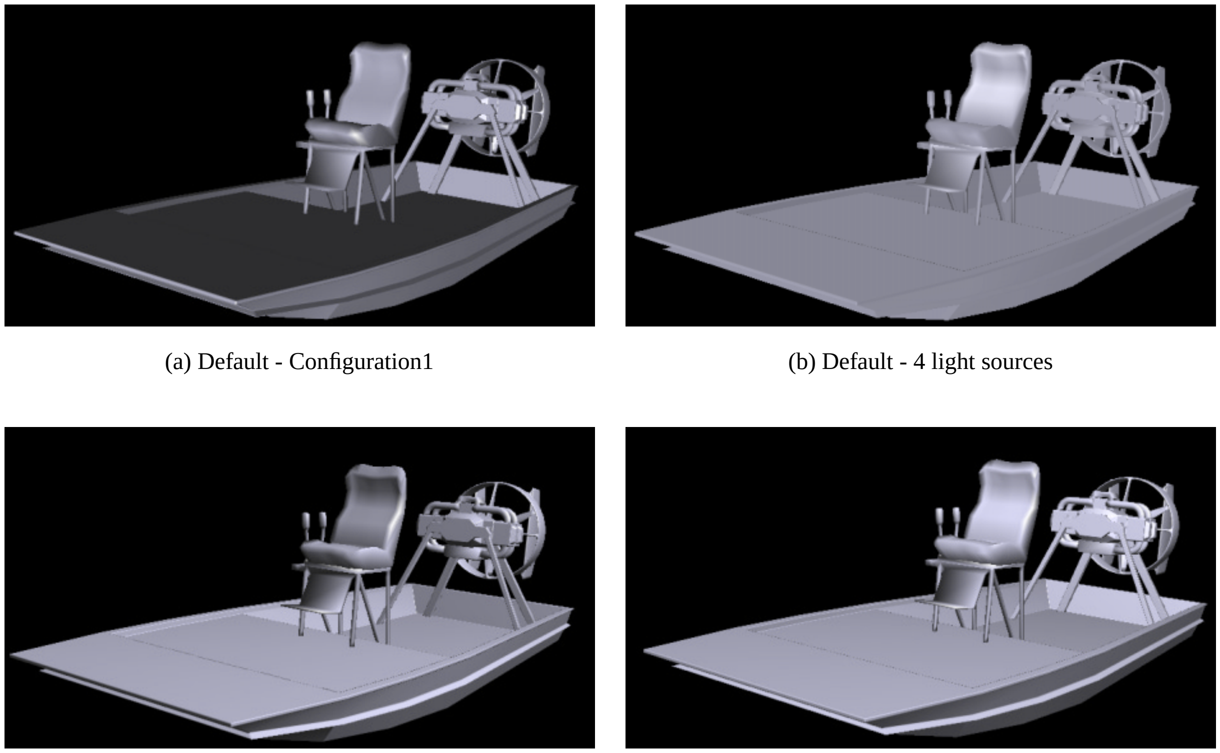

(d) Optimized - Configuration2

Figure 7.8: Foot.
Default illuminations: (a) dark regions on most finger bones hide fine shape and geometry information; (b) here shape information is deficient due to weak or missing shading gradients on some of the finger bones.
Optimized illuminations: (c) and (d) successfully communicate the geometry of the finger bones.

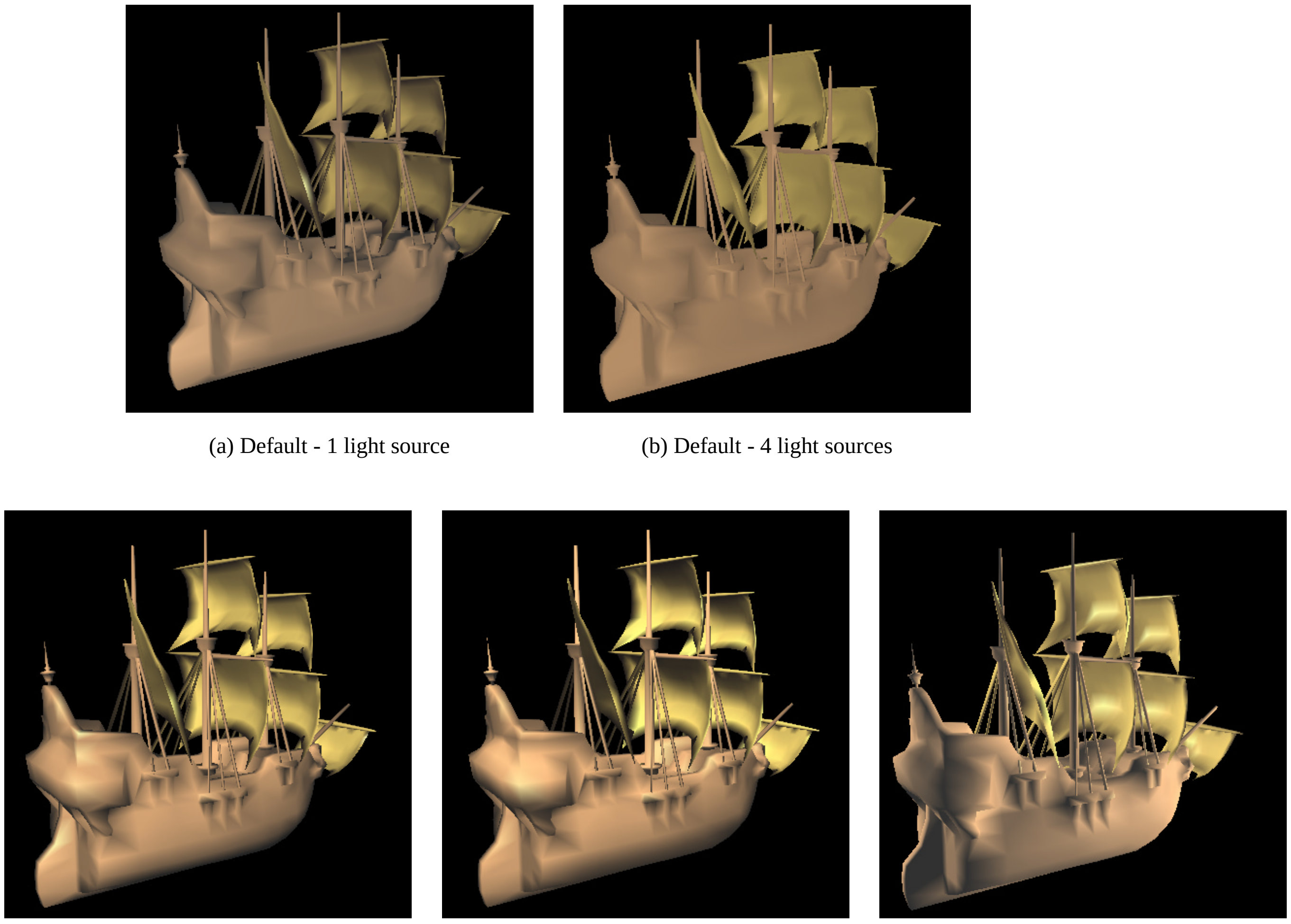

(a) Default - Configuration1

(b) Default - 4 light sources

(c) Optimized - 1 light source

(d) Optimized - Configuration2

Figure 7.9: Air-boat.
Default illuminations: (a) fails to display the the different 3 levels of the boat's floor: all three are displayed in constant dark intensity and edges are invisible; this deficiency occurs also with the top right face of the boat's frame; back of the chair lacks any shading gradients yielding in poor communication of its curved shape; (b) introduce too uniform shading resulting in reduced 3-D impression; floor levels are hardly noticeable; some edges on boat's side are lost; fails to clearly display the engine construction on the back.
Optimized illuminations: (c) All edges are visible; floor levels are clearly separated due to different shading intensities and more prominent edges; shading-gradients on the back of the chair helps to communicate its curved shape. All parts of engine construction are conspicuous; (d) Similar qualities to (c), however engine construction is somewhat less conspicuous.



(a) Default - 1 light source

(b) Default - 4 light sources

(c) Optimized - Configuration1

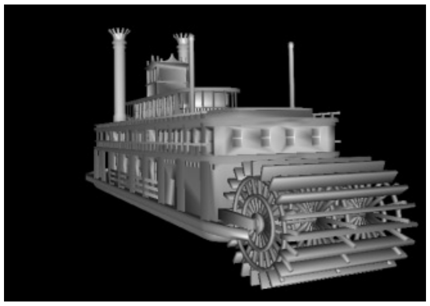
(d) Optimized - Configuration2

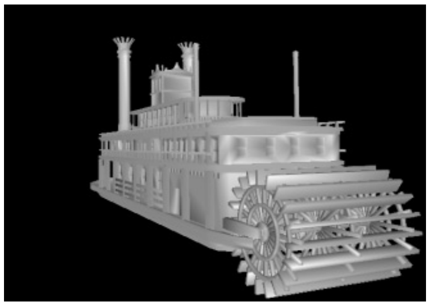
(e) Optimized - Configuration3

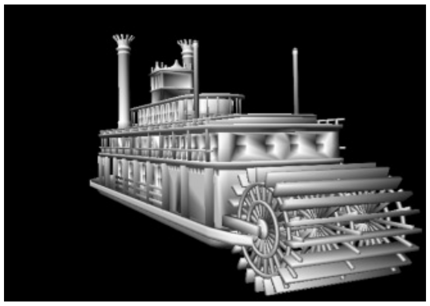
Figure 7.10: Galleon.

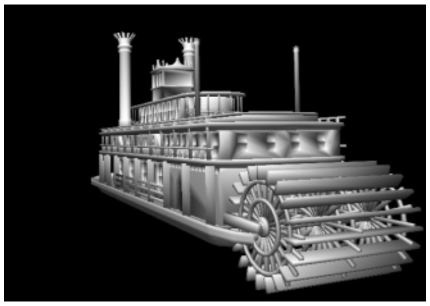
Default illuminations: (a) illumination is too flat on side of ship's body and on back sales. some details appear blurred and edges at front and back sales are not prominent enough; and (b) almost uniform shading on most ship's regions.

Optimized illuminations: (c) and (d) introduce major improvement in both ship's body and sales; Adding a second optimized light (e) enables highlights on sales.


(a) Default - Configuration1


(b) Default - 4 light sources


(c) Optimized - 1 light source


(d) Optimized - Configuration2

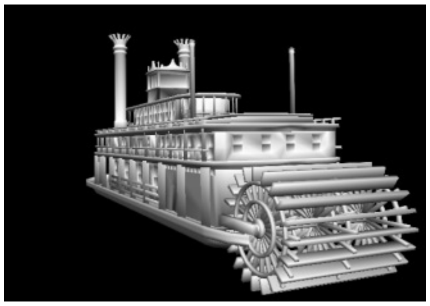

(e) Optimized - Configuration3

Figure 7.11: Paddle-boat.
Default illuminations: (a) left side appears blurred and details are lost. Details are also lost in the three surrounding panels and top part of the chimneys; (b) extra illumination causes all image to appear blurred.
Optimized illuminations: (c) and (d) introduce major improvement in left side, panels and chimneys; some minor deficiencies are found on the wings of the wheel, where not all edges are sufficiently prominent. Although this is mostly due to insufficient geometry specifications of the model (yielding problems with the normals), adding a second optimized light (e) is still able to overcome it.

Stage	Procedure	% time of process - No hardware support	% time of process - With hardware support
Preliminary operations	(Total)	15	15
Optimization	Lighting parameters calculations	1	1
	Rendering	39	20
	Read frame buffer	22	33
	Quality function	23	22
	(Total)	85	85

Table 2: Time distribution in the lighting-design process with and without using hardware-support for rendering. Process was performed for the cow model with resolution of 200x200, using single free-light (configuration1). For both cases optimization process took 8 iterations and 27 renderings, and total time was approximately the same (4 seconds).

8 Summary and Future work

We have presented a fully automatic lighting design system for traditional (photorealistic) rendering of 3D models. The lighting designs are obtained by optimizing a perception-based image quality function, yielding comprehensible images of 3D object, which effectively communicate information about shapes, materials, and spatial relationships.

There are many promising directions for future research, some of which are outlined in the remainder of this Section.

System automation:

The automation of the system can be further enhanced by developing methods that based on analysis of the scene, viewing-parameters and perhaps the rendering algorithm, can automatically determine the preferred number of free light-sources, and recommend whether the optional f_{dir} term should be enabled (currently, a high-level decision should be made to select from a choice of several system configurations (Section 6.3.9)). Another issue that is not adequately handled is the selection of the *threshold resolution* for performing efficient optimization (Section 6.3.3).

Extending lighting model:

Extending the system to work with global illumination models requires proper handling of new effects such as cast shadows, inter-reflections and refraction, both on the level preliminary scene analysis and system setup, and the level of image analysis and quality evaluations. An interesting starting point can be to try and apply the lighting solution found by the current system to such global illumination algorithm. We believe that for a wide range of cases these solution will remain satisfactory.

If using a "heavy" rendering algorithm, multi-resolution optimization technique will be probably found beneficial to reduce the number of renderings required in the higher resolution levels. Another approach can be using a *reduced-rendering* : under the assumption that full lighting calculation at each iteration may not be required to achieve an adequate evaluation for the optimization process, a reduced version of the rendering algorithm may be used to save redundant lighting calculation time (e.g. reduced recursion depth in Raytracing, or reduced number of patches in Radiosity). Furthermore: it seems that for a wide variety of scene definitions, the effects of direct illumination may be adequate. Full lighting calculation can then be applied either at a fine-tuning stage of the optimization, or only to render the final image.

In addition to using other lighting models, using other lighting parameters can be considered, and in particular other light types such as spotlights may be also considered.

Types of scene models:

The performance and behavior of the system should be further examined for scenes highly reach with small detail, and scenes that are completely occupied with objects and surfaces (e.g. indoor scenes).

Optimization:

More sophisticated optimization methods should be considered and implemented. Using techniques that reduces the number of iterations performed, as well as the total number of function-evaluations required (and consequently number of renderings too) may significantly enhance system performance.

View-independent solutions:

Extending the system to find view-independent lighting solutions can be very useful. However, this is a difficult task, due to the fact that the core of the system is an analysis performed on the 2-D image created for the specific viewing parameters supplied.

Quality function:

Clearly, the formulation and implementation given for the quality function is not unique and definitive. Although several methods of measuring and evaluating the visual and geometrical information were examined, applying different techniques may be proven beneficial.

One example is the use of more accurate edges evaluation technique, which takes into account a scale of expected prominence for various edges.

Another example is to elaborate the shading-gradients term such that it expresses a particular requirement for enhanced shading gradients in regions of high-curvature of surfaces.

Using further visual information such as spatial frequencies or colors (Section 3) for perceptual quality evaluation may be considered.
Various features of the HVS, which affects the response of human to the visual information (Section 3), may be integrated into the quality metric. For instance, the non-linear response and threshold sensitivity of the HVS to luminance contrast [13] [4] can be used to enhance the accuracy of measuring the effect of shading gradients.

References

[1] E.H. Adelson and A.P. Pentland. The perception of shading and reflectance. In D. Knill and W. Richards, editors, *Perception as Bayesian Inference*, chapter 11. Cambridge University Press, 1996.

[2] Vicki Bruce and Patrick R. Green. *Visual Perception: Physiology, Psychology and Ecology*, chapter 5–9. Lawrence Erlbaum Associates Ltd., 2nd edition, 1991.

[3] Patrick Cavanagh and Yvan G. Leclerc. Shape from shadows. *Journal of Experimental Psychology: Human Perception and Performance*, 15(1):3–27, 1989.

[4] Alan Chalmers, Scott Daly, Ann McNamara, Karol Myszkowski, and Tom Troscianko. Image quality metrics. Course notes, Siggraph 2000, July 2000.

[5] Stanley Coren and Lawrence M. Ward. *Sensation and Perception*. Harcourt Brace Jovanovich, third edition, 1989.

[6] Antonio Cardoso Costa, A. Augusto de Sousa, and F. Nunus Ferreira. Lighting design: A goal based approach using optimization. In *Rendering Techniques '99*, pages 317–328. Springer-Verlag, 1999.

[7] William Curran and Alan Johnston. The effect of illuminant position on perceived curvature. *Vision Research*, 36(10):1399–1410, 1996.

[8] Karen K. De Valois and Frank Kooi. The role of color in spatial vision. In L. Harris and Michael Jenkin, editors, *Proc. 1991 conference on spatial vision in humans and robots.*, pages 149–159. Cambridge University Press, 1993.

[9] R. L. De Valois and K. K. De Valois. *Spatial Vision*. Oxford University Press, Oxford, UK, 1988.

[10] Andrew S. Glassner. *Principles of Digital Image Synthesis*, chapter 1 - The human visual perception. Morgan Kaufmann, 1995.

[11] Rafael C. Gonzalez and Paul Wintz. *Digital image processing*. Addison-Wesley, 1987.

[12] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A non-photorealistic lighting model for automatic technical illustration. In *Computer Graphics Proceedings*, Annual Conference Series, pages 447–452, August 1998.

[13] Anil K. Jain. *Fundamentals of Digital Image Processing,* chapter 3, 9. Prentice-Hall, Englewood Cliffs, NJ, 1989.

[14] John K. Kawai, James S. Painter, and Michael F. Cohen. Radioptimization — goal-based rendering. In *Computer Graphics Proceedings, Annual Conference Series,* pages 147–154, 1993.

[15] Jan J. Koenderink. What does the occluding contour tell us about solid shape? *Perception*, 13:321–330, 1984.

[16] Pascal Mamassian and Daniel Kersten. Illumination, shading and the perception of local orientation. *Vision Research*, 36(15):2351–2367, 1996.

[17] Lee Markosian, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein, and John F. Hughes. Real-time nonphotorealistic rendering. In Turner Whitted, editor, *SIGGRAPH '97 Conference Proceedings*, pages 415–420. Addison Wesley, August 1997.

[18] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design Galleries: A general approach to setting parameters for computer graphics and animation. In *SIGGRAPH '97 Conference Proceedings*, pages 389–400. Addison Wesley, August 1997.

[19] D. Marr. *Vision. A Computational Investigation into the Human Representation of Visual Information*. Freeman, New York, 1982.

[20] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings Royal Society of London Bulletin*, 204:301–328, 1979.

[21] Ennio Mingolla and James T. Todd. Perception of solid shape from shading. *Biological Cybernetics,* 53:137–151, 1986.

[22] Pierre Poulin and Alain Fournier. Lights from highlights and shadows. In David Zeltzer, editor, *Proceedings of the 1992 Symposium on Interactive 3D Graphics*, pages 31–38, March 1992.

[23] Pierre Poulin, Karim Ratib, and Marco Jacques. Sketching shadows and highlights to position lights. In *Proceedings of Computer Graphics International 97*, pages 56–63, 1997.

[24] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, second edition, 1992.

[25] Vilayanur S. Ramachandran. Perceiving shape from shading. *Scientific American*, pages 58–65, August 1988.

[26] Vilayanur S. Ramachandran. Perception of shape from shading. *Nature*, 331(14):163–166, 1988.

[27] Mahesh Ramasubramanian, Sumanta N. Pattanaik, and Donald P. Greenberg. A perceptually based physical error metric for realistic image synthesis. *Computer Graphics*, 33(Annual Conference Series):73–82, 1999.

[28] Holly Rushmeier, George J. Ward, Christine Piatko, Phil Sanders, and Bert Rust. Comparing real and synthetic images: Some ideas about metrics. In Patrick M. Hanrahan and Werner Purgathofer, editors, *Rendering Techniques '95*, Eurographics, pages 82–91. Springer-Verlag Wien New York, 1995.

[29] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-D shapes. *Computer Graphics*, 24(4):197–206, 1990.

[30] Chris Schoeneman, Julie Dorsey, Brian Smits, James Arvo, and Donald Greenberg. Painting with light. In *Computer Graphics Proceedings, Annual Conference Series, 1993*, pages 143–146, 1993.

[31] James T. Todd and Ennio Mingolla. Perception of surface curvature and direction of illumination from patterns of shading. *Journal of Experimental Psychology: Human Perception and Performance*, 9(4):583–595, 1983.

[32] R. J. Watt and M. J. Morgan. A theory of the primitive spatial code in human vision. *Vision Research*, 25:1661–1674, 1985.

[33] Mason Woo, Jackie Neider, and Tom Davis. *OpenGL Programming Guide*. Addison-Wesley Developers Press, second edition, 1997.