MODELING AND EDITING REAL SCENES WITH IMAGE-BASED TECHNIQUES

by

Yizhou Yu

B.E.(Zhejiang University) 1992
M.S.(Zhejiang University) 1994

A dissertation submitted in partial satisfaction of the
requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Jitendra Malik, Chair
Professor David A. Forsyth
Professor Charles C. Benton

Fall 2000

The dissertation of Yizhou Yu is approved:

_____-

Chair Date

_____-

Date

_____-

Date

University of California, Berkeley

Fall 2000

Modeling and Editing Real Scenes with Image-Based Techniques

Abstract

Modeling and Editing Real Scenes with Image-based Techniques

by

Yizhou Yu

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Jitendra Malik

Image-based modeling and rendering techniques greatly advanced the level of photorealism in computer graphics. They were originally proposed to accelerate rendering with the ability to vary viewpoint only. This thesis focuses on capturing and modeling real scenes for novel visual interactions such as varying lighting condition and scene configuration in addition to viewpoint. This work can lead to applications such as virtual navigation of a real scene, interaction with the scene, novel scene composition, interior lighting design, and augmented reality.

This thesis has two important parts. The first part includes the techniques to extract an object-level representation of a real scene which can be rendered with modifications to the existing spatial configuration. The key components here involve automatic algorithms to segment the geometry from range images into distinct surfaces, and register texture from radiance images with the geometry. The top-down segmentation algorithm uses a pairwise similarity measurement to recursively partition a point set into a binary tree with individual surfaces as leaves. Our image registration technique can automatically find the camera

poses for arbitrary position and orientation relative to the geometry.

The second part includes the inverse global illumination technique which refers to recovering reflectance models of various materials present in a real mutual illumination environment. The method's input is a geometric model of the scene and a set of calibrated photographs taken with known light source positions. The result is a lighting-independent model of the scene's geometry and reflectance properties, which can be rendered under novel lighting conditions using traditional graphics methods. The underlying philosophy is using low-parameter BRDF models and solving optimization problems to recover the parameters. Synthetic images rendered using recovered BRDF models are comparable to real photographs.

Complementary to the above two parts, this thesis also presents a technique for recovering illumination conditions of outdoor scenes, algorithms for texture map synthesis from real photographs and texture map compression for efficient conventional texture-mapping, and a visibility algorithm for projective texture-mapping.

_____

Professor Jitendra Malik
Dissertation Committee Chair

To my wife Hong Wu and my parents

Contents

List of Figures

List of Tables

Acknowledgements

There are many people without whom this work would not have been possible. I would first like to thank my advisor, Jitendra Malik, for taking me as his student and allowing me to pursue my research interests, and for his guidance, encouragement and friendship during the course of this work. I would also like to thank Paul Debevec whose expertise in the Façade modeling system and dynamic range images made my work on photometric properties easier.

I would very much like to thank Andras Ferencz who worked with me on the laser range data project which would not have been finished without his creativity and diligence. And I would like to thank Tim Hawkins, George Borshukov, Tal Garfinkel, and Johnny Chang for their enjoyable collaboration on various projects I have worked on.

At Microsoft, I would like to thank Brian Guenter and Michael Cohen for giving me the opportunity to work on the face project from which I gained much experience and knowledge in human skin reflectance and reflectance recovery in general.

Also at Cyra Technologies, Inc., I am greatly indebted and thankful to Ben Kacyra, Mark Wheeler, Daniel Chudak and Jonathan Kung for their help and advice in using their time-of-flight laser scanner and the accompanying Cyrax software.

Many thanks to Gregory Ward Larson for advice in using RADIANCE rendering system and estimating reflectance, to David Culler and the Berkeley NOW (Network of Workstations) project for using the NOW to render some video sequences.

I would like to thank my committee members David Forsyth and Charles Benton for helping make this thesis happen. And I must thank professors Carlo Séquin and John Canny for their interest in this work and their valuable suggestions.

I would also like to thank the sponsors of this research: the Multidisciplinary Univeristy Research Initiative's 3D Visualization program and ONR BMDO, the California MICRO program, and the Microsoft Graduate Fellowship program.

Chapter 1

Introduction

Computer graphics started with the display of data on hardcopy plotters and cathode ray tube(CRT) screens soon after the introduction of computers themselves. The invention of raster display systems revolutionized the field by providing more efficient, systematic and convenient avenues to manipulate data for displaying. Frame buffers adopted by such systems can be seen as the earliest connection between graphics and images.

During the past decades, researchers in graphics developed many techniques to synthesize images and animation. Research directions in graphics include geometry modeling, reflectance modeling, light transport simulation and motion modeling. The procedure to create synthetic images and animation has multiple steps. First, we need to use some modeling software to build the geometry of a virtual scene whose components could be either polygonal or curved surfaces(NURBS). This is followed by assigning reflectance models to each surface. Geometry and reflectance provide a static lighting-independent model of a scene. Motion can be added on top of this to vary scene configurations by keyframing or physical simulation. The final stage is rendering which generates an image for each frame

based on the configuration for that particular frame. It is light that reveals the form and material of objects. Therefore, the most important part of rendering is lighting simulation where light interacts with surface geometry and reflectance to produce the final appearance of a scene.

One of the major goals of graphics is photorealism. However, most synthetic images for this purpose still do not look realistic enough. It takes significant manual efforts to fine-tune the geometry, reflectance and motion to make them look more photorealistic. Around early 1990's, a new research trend called image-based rendering(IBR) emerged as an alternative to traditional graphics. It was originally proposed to render real complex scenes from novel viewpoints only using a set of photographs, but without explicit geometry [5]. It was also exploited for accelerating rendering [32] because the performance of image based rendering is solely determined by the resolution of the output image and independent of the scene complexity. Since then, considerable amount of work has been done along this direction [30, 6, 36, 50, 32, 19, 12, 13, 45, 48, 39]. Some of the work is different from the rest by recovering geometry explicitly. It is based on previous work on stereopsis and structure-from-motion [15] in computer vision. Thus, image-based rendering was extended to image-based modeling and rendering (IBMR).

Since IBMR makes use of data from photographs of the real world, novel images generated afterwards certainly have more photorealism (Fig. 1.1). But what about the other aspects ? In traditional graphics, a user can edit the scene configuration, lighting, and motion until he/she obtains satisfying synthetic images. However, in IBMR, a real environment is usually captured as a whole. The spatial configuration of the digital copy of the scene becomes static and reflects the state of the real scene at the moment of data acqui-

sition. We cannot move the captured objects around because objects are not represented independently to one another. The lighting condition for the scene is also fixed in the photographs. It is not obvious how to change the lighting and synthetically render new images of the scene under novel illumination. In summary, photorealism has been gained at the expense of manipulability.

The research presented here tries to fix the above-mentioned problems with previous IBMR work, merge the advantages from both traditional graphics and IBMR so that we can not only capture and display reality on a computer screen, but also extend reality to create a novel scene with photorealistic quality. Recovering underlying geometric and photometric models of a real scene for novel image synthesis has been the philosophy of this research (Fig. 1.2). The major contributions of this research include algorithms that allow the spatial configurations(Fig. 1.3) and lighting conditions(Fig. 1.4) of a real scene to be varied digitally. The basic idea about changing scene configuration is to find object boundaries and represent objects independently. The underlying idea for changing lighting is to recover low-parameter reflectance models of the multiple surfaces in a scene simultaneously so that we can plug in novel illumination and re-render the scene. The presented techniques allow synthetic images to be comparable to real photographs. The applications of such techniques include virtual navigation of a real scene, interaction with the scene, novel scene composition, interior lighting design, and augmented reality. Obviously, they also have many applications in the movie and computer game industries.

Figure 1.1: The top image is a synthetic image rendered from traditional graphics. The geometry and reflectance are hand-crafted. Lighting simulation generates highlights on the walls and soft shadows on the floor. The bottom image is a synthetic image from image-based modeling and rendering. The geometry and texture are recovered from range and radiance images. The bottom image looks more realistic.

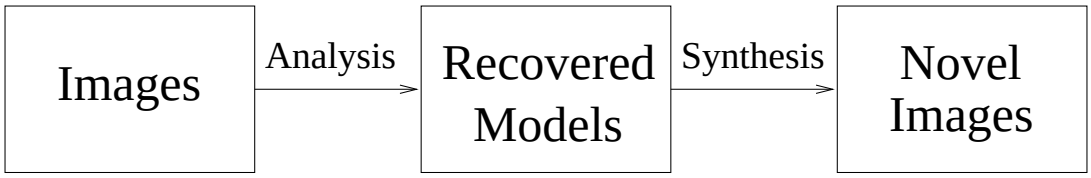


Figure 1.2: The underlying philosophy

Figure 1.3: The left image shows a real room in its original configuration. The right one shows a synthetic image of the same room in a novel configuration with a couch upside down sitting on the ceiling.

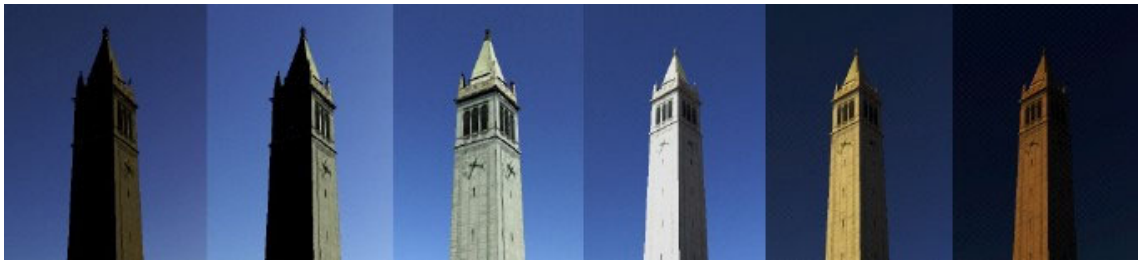


Figure 1.4: A synthetic sunrise to sunset sequence for a clock tower—-The Campanile.

Chapter 2

Modeling Objects from Range Images

2.1 Introduction

An object is made up of a collection of surfaces which in turn have geometric properties such as size and shape as well as photometric properties such as color and texture. Editing operations should be performed at object level, which requires us to give each object geometric and photometric representations that are independent of the rest of the scene. Since a scene is usually acquired as a whole, this kind of object-level information is not directly available from the captured geometry or from photographs.

We need to segment the scene into objects. In this chapter, we use a laser range finder to acquire a discrete representation of the geometry, a point cloud. Each point in the cloud has a 3D position, an estimated normal orientation of the underlying surface at that point, and a returned laser intensity value. These cues give adequate information to distinguish points that belong to different objects. Therefore, we do segmentation on the point cloud before a mesh is actually built. Our technique is an extension of a 2D image segmentation

algorithm using spectral graph theory. The result is a binary tree with individual surfaces as leaves. We first oversegment the scene into a set of coherent surfaces and then ask the user to interactively group surfaces into semantically meaningful objects.

Addressing this problem adds much more flexibility to geometric data capture, cleans the obstacles in extracting individual objects from range images, and allows humans to interactively manipulate them. It makes it easier to build an object library based on the real world, which can be composed to form novel scenes using the objects for rendering and animation.

2.1.1 Overview

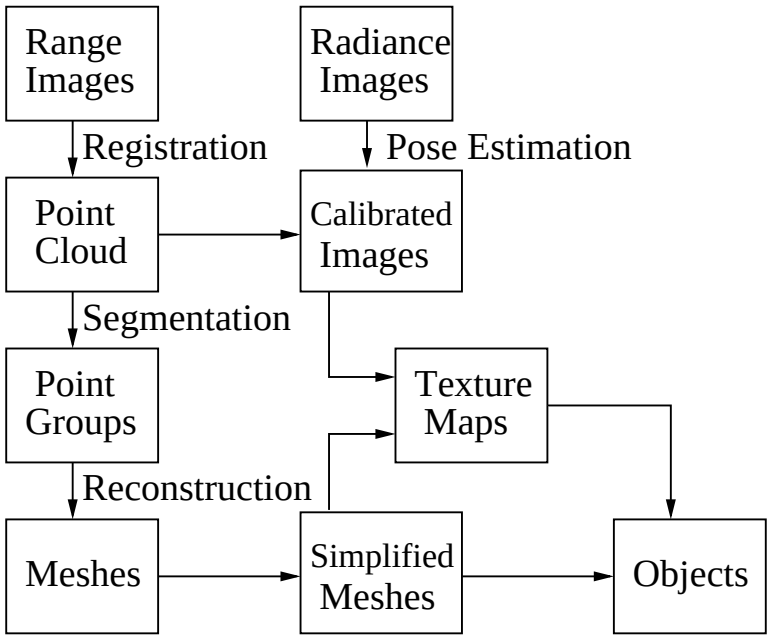


Figure 2.1: Multiple stages in our range and radiance data processing procedure

The input to our pipeline is a set of range and radiance images(Fig. 2.1). The range images are registered together first to give a unified point cloud. The segmentation algo-

rithm is then run on the point cloud, breaking it into groups. With some user interaction, we can assemble these groups into objects. We then build a mesh for each object and run mesh simplification to reduce its complexity. The part of the pipeline for radiance images includes camera pose estimation and texture map synthesis which will be introduced in the next chapter.

From this process, we can see that range data segmentation can be useful in multiple stages. Fitting smooth parametric surfaces to individual objects is much easier after segmentation, and memory capacity becomes less of a limitation if we only reconstruct a mesh for one object at a time. Mesh simplification is improved since we do not intend to simplify over segmentation boundary. Segmentation into surfaces with approximately uniform specularities would aid in recovering specular reflectance models of the surfaces.

2.2 Previous Work

The work I present here has been made possible by previous work in geometry acquisition, mesh reconstruction and simplification, 2D image registration and segmentation, and range image registration and segmentation.

Recent work in laser range scanning has made it possible to recover accurate geometry of real-world scenes. [2] introduced the iterative closest point(ICP) algorithm to register multiple range images. [41] addressed the same problem for large data sets. A number of robust techniques for merging multiple range images into complex models are now available [7, 53, 9, 54, 39]. [23, 14, 1] also introduced techniques to a more general problem which is mesh reconstruction from unorganized points. The reconstructed meshes from

the above techniques usually have huge amount of complexity, which makes rendering inefficient. There is a large amount of work on obtaining simplified meshes with minimal deviation in shape [24, 16, 8, 33].

Techniques have been developed recently for texture-mapping recovered geometry[12, 110, 37]. Some image-based rendering work has been introduced to implicitly make use of recovered depth information in addition to images[36, 45, 39]. [10] proposed a technique to integrate virtual objects with real environments. However, manipulating real objects in real scenes is a more difficult problem that requires extra data processing power.

Research in segmentation is most related. There has been much work on range image segmentation in the computer vision and graphics community [3, 22, 29, 38, 43, 52, 31, 26]. Most of these techniques consider every range image as a rectangular array of points with 3D positions. Due to the similarity between range images and 2D images, most of the image segmentation techniques, such as edge detection, region growing and pixel clustering, can be applied to range image segmentation[26]. The major cues used include the depth value and normal orientation at each pixel. Most of previous work in this area was for fitting surface primitives, such as planar and quadric patches, and generalized cylinders. Therefore, statistical tests are often incorporated into the segmentation algorithms to verify whether a particular surface primitive can fit well to a group of pixels, and determine whether region split or merge should be considered. This type of algorithm works well on range images of mechanical CAD models which consist of planar and simple curved surfaces.

However, most real objects have unknown free-form shapes, such as statues and curtains. To extract objects from a real scene, we are more interested in a segmentation al-

gorithm that is independent of any surface primitive. Region growing approaches based on local split and merge decisions are not very appropriate, either, because the decisions made are local, which is suboptimal in finding object boundaries. We need some top-down algorithm to make global decisions. On the other hand, we may need multiple range images from different angles to capture a complex model. A segmentation algorithm, that can work on multiple registered range images simultaneously, is more interesting. In this situation, the order of points defined by the rectangular array in a single image is lost. We need new techniques to solve this more complicated problem. In the 2D image segmentation literature, some recent spectral graph theory-based algorithms [47, 40, 35] appeared to perform better than other techniques. They make decisions on where to partition the data using global information. We will extend the normalized cut framework in [47] to 3D range image segmentation in the next section.

[34] proposes a technique for segmenting surface meshes by generalizing morphological watersheds. It is not directly relevant to the problem we are looking into because we consider segmentation as a very fundamental data processing stage which should happen at least in parallel to mesh reconstruction, not after. Effective segmentation of points into groups should be able to benefit mesh reconstruction.

2.3 Range Data Segmentation

The input data to this problem is a 3D point cloud created by merging the points from multiple registered range images. In addition to 3D position, each point has two associated attributes, a normal orientation estimated from neighboring pixels in the scan image, and a

returned laser intensity value which depends mostly on the surface reflectance corresponding to the wavelength of the laser beam. The output of this module is a partition of the point cloud, treated as a set, into subsets such that each subset defines a complete object. The subsets are mutually exclusive, and their union is the complete set.

We achieve this goal in two steps. For the first step, we developed an automatic algorithm to partition the points into surface regions, each of which has approximately uniform geometric and photometric properties represented by 3D locations, surface normals and returned laser intensities. In the second step, we interactively group surface regions into individual objects such that each object can be treated separately but points in the same object are treated in the same way. Note that surfaces in the same object may have very different surface properties. To give an example, suppose one of the objects in the scene is a cube resting on a table. The first step would return 5 surfaces; the user is responsible for indicating that all these surfaces belong to one object. Considerable semantic knowledge can be involved in judging whether an object is merely resting on another or is rigidly attached and is thus part of the same object; at this stage we think it prudent to leave this judgment to a user.

In the rest of the section, we introduce the algorithm for automatic segmentation of the point set into surface regions. This is done by generalizing the normalized cut algorithm [47] to range data. There are three key issues to consider here. Namely, a) what is the appropriate similarity measure for range data; b) precisely what is the criterion to partition the graph; and c) what is the technique to obtain approximate solutions for large datasets.

2.3.1 Normalized Cut Framework

We introduce some details of the *normalized cut* algorithm[47] here. A graph $G = (V, E)$ is defined on the input data. In our context, the nodes represent local clusters derived from the point cloud with the associated attributes. An edge in E, (s, t) with $s, t \in V$, has a weight $w(s, t)$ defined by the similarity between the location and attributes of the two nodes defining the edge. The idea is to partition the nodes into two subsets, A and B, such that the following disassociation measure, the normalized cut, is minimized.

$$Ncut(A, B) = \frac{cut(A, B)}{asso(A, V)} + \frac{cut(A, B)}{asso(B, V)} \tag{2.1}$$

where $cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$ is the total connection from nodes in A to nodes in B; $asso(A, V) = \sum_{s \in A, t \in V} w(s, t)$ is the total connection from nodes in A to all nodes in the graph; and $asso(B, V)$ is similarly defined. This measure works much better than $cut(A, B)$ because it favors relatively balanced subregions instead of cutting small sets of isolated nodes in the graph.

To compute the optimal partition based on the above measure is NP-hard. However Shi and Malik show that a good approximation can be obtained by relaxing the discrete version of the problem to a continuous one which can be solved using eigendecomposition techniques. Let y be the indicator vector of a partition. Each element of y takes two discrete values to indicate whether a particular node in the graph belongs to A or B. If y is relaxed to take on continuous real values, it can be shown that the optimal solution can be obtained by solving the generalized eigenvalue system,

$$(D - W)y = \lambda Dy \tag{2.2}$$

where D is a diagonal matrix with $D(i,i) = \sum_j w(i,j)$, W is the weight matrix with $W(i,j) = w(i,j)$. The eigenvector corresponding to the second smallest eigenvalue is the optimal indicator vector in real space. A suboptimal partition can be obtained by first allowing y to take on continuous real values, solving the above generalized eigenvalue system for y, and then searching a certain number of discrete values for the best threshold to partition the real-valued elements of y into two subgroups. The two resulting subregions from this partition can be recursively considered for further subdivision. To improve efficiency, the complete graph defined by the data is usually simplified to only have edges that connect two nearby nodes. This algorithm can be used to solve different segmentation problems by choosing different edge weight $W(u,v)$ [46, 35].

2.3.2 Setting Up the Graph

Since we have multiple high-resolution scans, solving the graph partition problem on the original point set is impractical. For example, we have a dataset for a large room with nineteen 800x800 scans. For comparison, note that in the application of normalized cut to image segmentation, Shi and Malik considered 200x200 images. Thus we group nearby [1] points into clusters such that each cluster is a node. All the points within the same cluster

[1] To accelerate nearest point lookup, we set up a two dimensional grid on a virtual plane with its normal set to the average normal orientation of all the input points. Each cell in the grid has a list of points that are projected into it. To look up points that nearby a certain point, we only need to check the points around the cell which that point is actually projected into.

have similar normal orientations and laser intensities. Clustering is actually carried out incrementally to minimize memory consumption. Only one of the original range scans remains in memory every time. Part of its points get integrated into existing clusters, while others create new clusters. The number of nodes after initial clustering is reduced to about 40,000 for the room dataset. We use the averaged spatial location, normal and returned laser intensity of each cluster as the attributes of its corresponding node in the graph. Here the returned laser intensity is a better cue than color from photographs because it is a better approximation of surface albedo which is lighting independent. Thus we do not need to worry about oversegmentation due to shadows and shading effects in photographs. Local edges are set up among clusters that are within a certain distance of one another. We also set up random long-range connections among clusters to help use global context [47]. The number of random edges incident to each node is fixed. In this way, the adjacency matrix of the graph is sparse, which makes it possible to solve the problem efficiently.

The weight $w(u, v)$ over an edge (u, v) is the product of a similarity term $S(u, v)$ and a proximity term $P(u, v)$ both of which are in the form of a Gaussian distribution. $w(u, v)$ is a local measure of how likely the points (or clusters) are to belong to the same surface. $w(u, v)$ is close to 1 for points which are likely to belong together, and close to 0 for points which are likely to belong to separate objects, as judged purely from local evidence.

$$S(u, v) = exp(-diff^2(u, v)/2\sigma_w^2) \tag{2.3}$$

where $diff(u, v)$ could be either the angular difference in normal orientation or the scalar difference in laser intensity.

The proximity term over an edge (u, v) is used to model the fact that nearby points are more likely to belong to the same surface. Both similarity and proximity are well known Gestalt grouping factors.

$$P(u, v) = exp(-dist^2(u, v)/2\sigma_c^2) \tag{2.4}$$

where $dist(u, v)$ is some distance measure. The parameters σ_w and σ_c should be set according to the noise level in the input data. Small values tend to produce oversegmented results.

Surfaces and objects at different depth should be separated in segmentation. To enhance the difference caused by depth discontinuity, we define an anisotropic distance metric $dist(u, v)$ as follows. First we define a tangential plane at each of the points u and v given the normals and positions. Let the vectors u_n and u_p be the projections of the vector $\vec{uv}$ onto the normal and tangential plane, respectively, at point u. Then $dist_u(u, v) = \sqrt{E * ||u_n||^2 + ||u_p||^2/E}$ where $E(\geq 1)$ is some adjustment parameter to magnify the difference along the normal direction and shrink the difference along the tangential direction. This parameter can make sure to separate surfaces at different depth. $dist_v(u, v)$ can be defined similarly. Finally,

$$dist(u, v) = MAX(dist_u(u, v), dist_v(u, v)). \tag{2.5}$$

2.3.3 Criterion for Graph Partition

Let us first look at the region shown in Fig. 2.2 where two subregions A and B are adjacent to each other. Surface attributes are uniform within the same subregion but different across

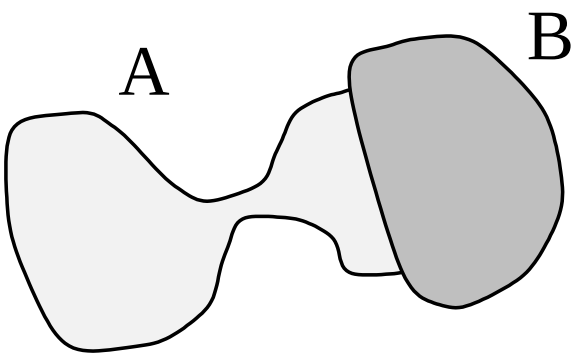


Figure 2.2: Two regions A and B are adjacent. A correct graph partition should happen at their border, not in the middle of A.

subregions. Although the correct partition should happen at the boundary between A and B, normalized cut with local connections tends to partition the region in the middle of A because subregion A is quite thin in the middle and the normalized cut value in Eq.(2.1) is very small there. In the context of image segmentation, this can be justified as a segmentation of the image into 'parts', but we felt that this was not too important a consideration for us. We therefore introduce a slight modification of the segmentation criterion by defining the *normalized weighted average cut.*

$$NWAcut(A,B) = MIN(\frac{WAcut(A,B)}{WAasso(A,A)}, \frac{WAcut(A,B)}{WAasso(B,B)}) \tag{2.6}$$

where $WAcut(A,B) = \frac{\sum_{u\in A, v\in B} S(u,v)P(u,v)}{\sum_{u\in A, v\in B} P(u,v)}$, $WAasso(A,A) = \frac{\sum_{u,v\in A} S(u,v)P(u,v)}{\sum_{u,v\in A} P(u,v)}$, and $WAasso(B,B)$ is similarly defined. It is quite easy to figure out that in Fig. 2.2, normalized weighted average cut is minimized at the boundary between A and B. This new measure does not favor balanced subregions as normalized cut, so it is not appropriate as the basic criterion to be used for segmentation; but by setting a threshold on the minimum weighted average cut we can reduce the chance of splitting in the middle of region A.

Algorithmically we proceed as follows. For the currently considered region, first solve the sparse eigensystem Eq.(2.2) using Lanczos algorithm [18] to come up with a candidate partition, then check the normalized weighted average cut along the candidate boundary. If both normalized weighted average cut and normalized cut are below the threshold T_{cut}, split the region and consider the two subregions in turn; otherwise, stop recursion on the considered region.

2.3.4 The Complete Algorithm

We have two postprocessing steps, boundary improvement and fine segmentation, which are aimed at getting high quality segmentations, in spite of the fact that we could not process the original dataset directly in Eq.(2.2) because of its large size.

After a region is split into two subregions, we have an initial boundary between them. In practice, this boundary may deviate a little from the real surface boundary. To improve its localization, we exploit the linear order on the nodes in the original region according to the magnitude of their corresponding elements in the second smallest eigenvector of Eq.(2.2). A local segmentation problem is solved at each point cluster close to the initial boundary. At each of these clusters, collect all clusters in its neighborhood and set up a complete graph among them since the size of graph is small. Then search for the node that best partition the linear order into two parts under normalized cut criterion. Every cluster near the initial boundary may be included into multiple local segmentations. Each local segmentation assigns the cluster to one side of the boundary or other. We use majority voting to decide which side of the boundary it should belong to. Once we have determined

the membership of all clusters, the final boundary also becomes clear.

To reduce the impact of the initial point clustering(which is suboptimal as it is local) at the beginning of the algorithm, we introduce an additional step at the end of the algorithm to refine segmentation results. Based on the previous segmentation, it reads in all points belonging to one group at a time, cluster them at a finer scale, and run segmentation on the new clusters once again.

The spirit of both these post-processing steps is a coarse-to-fine refinement, something that has been tried quite extensively in various computer vision settings.

The whole segmentation algorithm is summarized as follows.

- **Coarse Segmentation**
  - **Clustering** Group all points into clusters such that points in the same cluster are within a prescribed radius from its centroid, and have close normal orientation and laser intensity.
  - **Cluster Segmentation**
    - Recursive segmentation based on normal continuity and proximity.
    - Recursive segmentation based on continuity in returned laser intensity and proximity.
    - **Stopping Criterion** Both the normalized cut value and weighted average cut value are below a threshold.
    - **Boundary Improvement** Once the stopping criterion is satisfied, apply local optimal segmentation and voting at each boundary cluster to improve boundaries.

- **Fine Segmentation** Based on coarse segmentation results, every time only read all points that belong to one group; set smaller radius for clustering and smaller σ value in proximity term; and repeat the same steps in coarse segmentation on them.

2.4 Mesh Reconstruction and Simplification

Given the segmentation results on a point cloud, we can recognize the points that belong to an object and build a mesh or fit surfaces for that object. Although these are not the focus of this chapter, we introduce the techniques we use to perform these tasks here.

There are two different methods to build a mesh for an individual object. The first method tries to extract all the points that belong to the object. This is actually a set of unorganized points since the scan order inherent in a range image is lost. We can then build a mesh using the algorithm in [23] or [1]. In practice, we use the crust algorithm in [1].

The crust algorithm works well for objects of which we have dense samples. However, our range images are not dense enough for objects that have fine details. So we try to make use of the scan order in a range image and build nearest neighbor connections. In each of the original range images, we first mark the points that belong to the object, and then build a connection between two points if they are direct neighbors and both are marked. So we can extract a partial mesh from each range image that covers the object and put them together to represent the geometry of the object. It is possible to apply the algorithms in [53] and [9] to merge these partial meshes and come up with a single mesh.

Once we have the meshes for individual objects, we continue to simplify them to improve rendering performance. Most of the previous algorithms on mesh simplification can

be applied. In practice, we use the technique presented in [16];

While it is much less time-consuming to scan multiple objects simultaneously in an environment, there is little information for back-facing surfaces and surfaces that are heavily occluded. For the same reason, these surfaces are less visible and therefore less important. We interactively insert some simple polygons to model these back-facing and occluded surfaces.

2.5 Results

Our algorithms have been tested on a complete real scene—a large reading room—as well as on individual laser range scans. We took 19 800 by 800 scans of the reading room and also scanned a piano from three positions in a separate setting with Cyra Technologies' time-of-flight laser scanner. Most visible surfaces were covered at centimeter accuracy.

We arranged 50 Cyra's targets on the walls, the ceiling and floor for scan registration. The scans were registered together by using Cyra's software to locate targets in each scan and set up correspondences among different scans.

2.5.1 Geometry Segmentation and Reconstruction

Since there is noise and outliers in the scans, we filter the scans before sending them to our segmentation algorithm which runs on a Pentium II 450MHz PC. For the reading room, our segmentation code produced 393 groups in four hours which were further grouped into 95 objects and surfaces within two hours of user interaction. The result is shown in Fig. 2.3(c)-(d) with different colors for different groups. All the curtains and furniture,

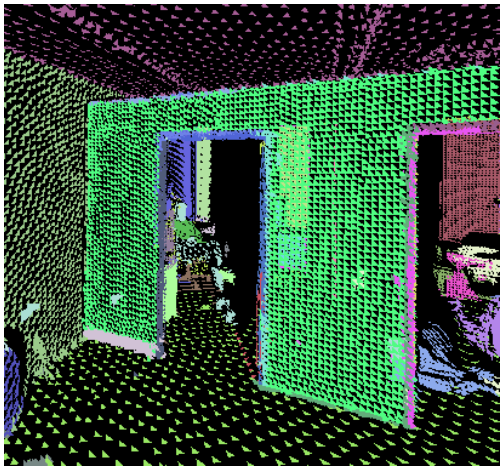
including lamps, tables, couches, dressers, and chairs, are correctly segmented out. Before user interaction, the number of groups per object ranges from 2 to 20 with an average of 4. The walls are oversegmented into multiple pieces and we did not grouped them together interactively because we do not have the need to manipulate a complete piece of wall. Fig. 2.3(a) and (b) also gives the segmentation results for a different but simpler room and a single facade. In Fig. 2.3(a), a person was sitting in the next room while it was scanned. We can see his head and torso are correctly segmented out from the rest of the environment. Fig. 2.3(b) shows that our anisotropic distance metric in Eq. 2.5 can effectively separate layers at different depth.

Most of the meshes, including the piano, were reconstructed using the crust algorithm[1]. Antique tables and chairs as well as curtains were reconstructed using nearest neighbor connections. Fig. 2.4 shows an image of the meshes which include lamps, tables, curtains, couches as well as the ceiling and walls. To demonstrate that we really have extracted individual objects, Fig. 2.5 shows the individual models of an antique table, a chair and the piano. The chandelier under the center of the ceiling and the floor were segmented out automatically and removed interactively, and a single plane is fit to the floor.

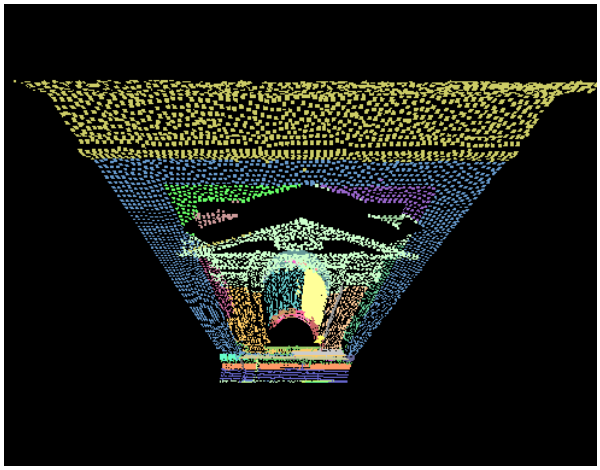
2.5.2 Scene Editing

Our ultimate goal is object manipulation. Fig. 2.6 shows two comparisons to demonstrate our ability to do scene editing. The images on the left are re-renderings of the original scene from a novel viewpoint by texture-mapping the reconstructed geometric models. The images on the right show synthetically composed scenes. In Fig. 2.6(b), a couch is moved

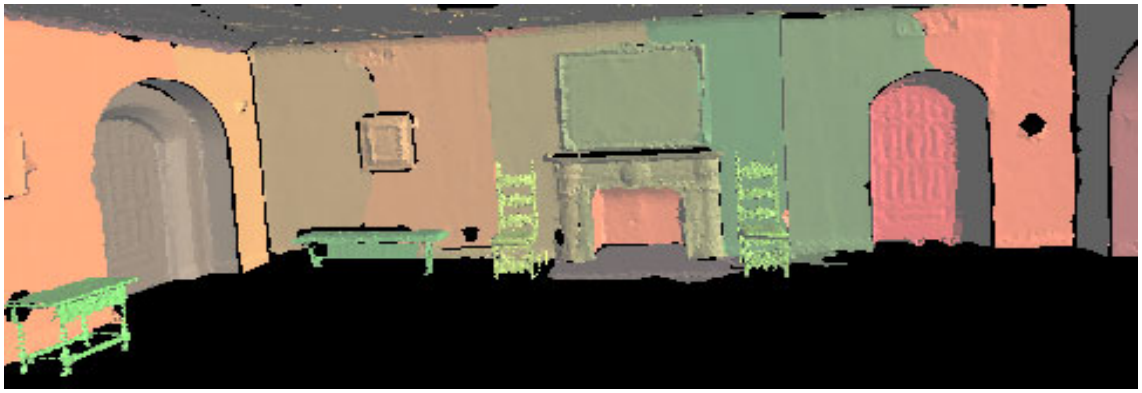
and replicated to places near the fireplace, a piano inserted and placed near where the couch was. In Fig. 2.6(d), we can see two lamps flying in the air.



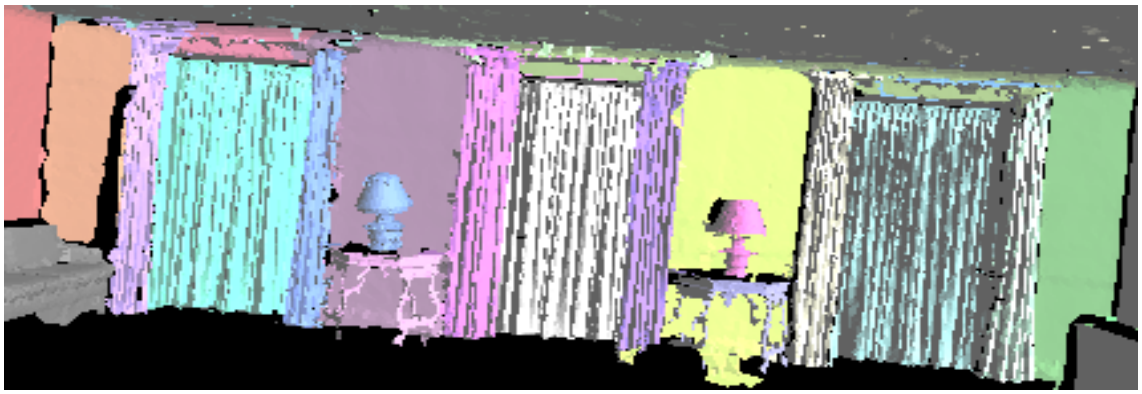
(a)



(b)



(c)



(d)

Figure 2.3: The segmentation results on three datasets. (a) a simple room with portals; (b) Albert Hall facade. (a) and (b) show the segmentation results directly from our automatic algorithm. Each dot represents one point cluster. Clusters in the same group are shown with the same color. (c) and (d) a large reading room with furniture and curtains, (c) and (d) show the segmentation results after the user interactively grouped surfaces into objects.

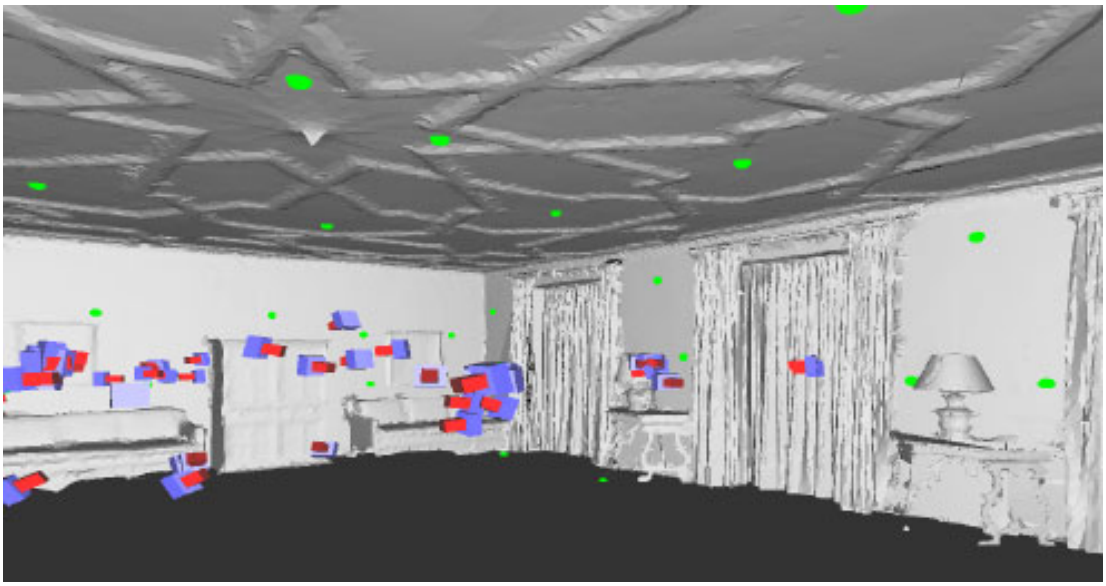
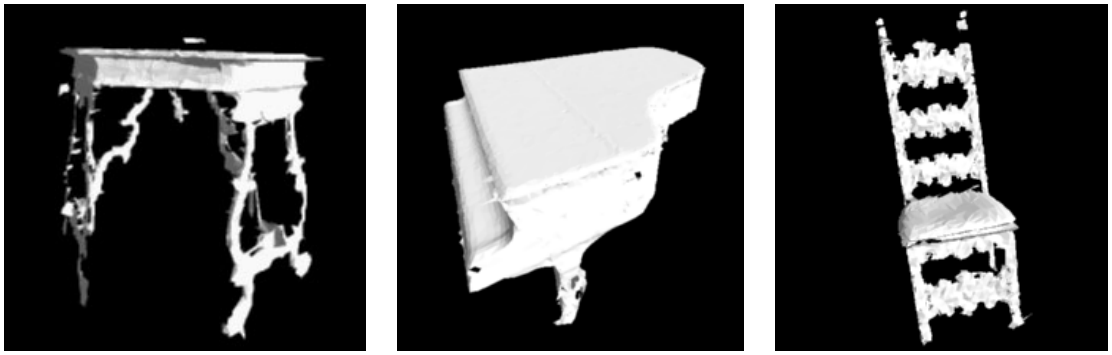
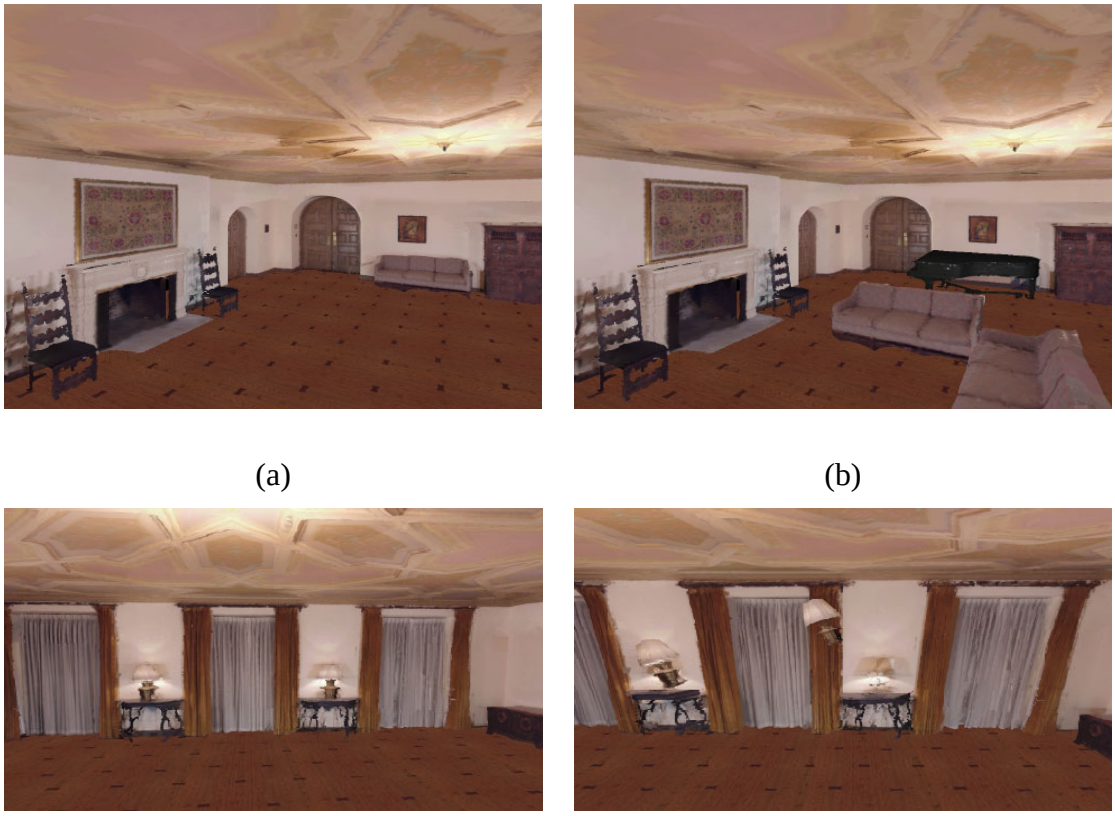


Figure 2.4: The reconstructed meshes with targets(green) and recovered camera poses(red and blue) for the reading room. There is a separate mesh for each group.



(a) (b) (c)

Figure 2.5: Geometric models of three objects. The meshes were built after their corresponding points were segmented out from the range images using our segmentation algorithm. (a) An antique table, (b) a piano, (c) an antique chair.



(a) (b)

(c) (d)

Figure 2.6: (a)&(c) Synthetic images with objects in their original positions rendered using texture-mapping. (b)&(d) Synthetic images with object insertion and relocation. A piano is added to the room, a couch and two lamps moved and replicated.

Chapter 3

Texture-Mapping Real Scenes from Photographs

3.1 Basics on Camera Pose Estimation

In this chapter, we are concerned about attaching texture information from photographs onto recovered geometry to improve photorealism. The geometry is represented as a triangular mesh which may be recovered either from photographs or from laser range images. There is a world coordinate system where the geometry resides. First of all, we need to align photographs with the geometry, i.e. we need to recover camera poses, including rotation and translation, in the world coordinate system. Here we assume that we know the internal parameters of the camera, such as focal length and radial distortion, with which we can convert the real camera into an ideal pinhole camera [15].

Both rotation and translation have 3 degrees of freedom. The translation is represented as a vector T. There are multiple ways to represent rotation, such as matrix, quaternion and

exponential coordinates [103]. The most compact representation is exponential coordinates which are used here. The exponential coordinates for a rotation also have 3 components, $w = [\ w_x\ w_y\ w_z\ ]^t$. The direction of this vector gives the rotation axis; the magnitude of this vector gives the angle $\theta = \|w\|$ by which we need to rotate around the axis. This vector has a corresponding matrix,

$$\hat{w} = \begin{pmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{pmatrix} \tag{3.1}$$

. With this matrix, exponential coordinates can be converted into a rotation matrix R by Rodrigues' formula

$$R = e^{\hat{w}} = I + \frac{\hat{w}}{\|w\|}\sin(\|w\|) + \frac{\hat{w}^2}{\|w\|^2}(1 - \cos(\|w\|)) \tag{3.2}$$

To obtain a unique solution to the 6 parameters of a camera pose, we need at least 4 pairs of corresponding points between the photograph and the geometry. With only 3 pairs of corresponding points, it is possible to have 4 feasible solutions. Corresponding points can be either feature points or calibration objects. Feature points can be either interactively selected or automatically detected. Correspondences between points in photographs and points in the geometry can also be either interactively selected or automatically obtained.

Combinatorial search may be used to find correspondences automatically. For example, if 4 feature points are detected in a photograph, we can try to match them with all possible combinations of 4 feature points in the 3D geometry. For every combination, recover a camera pose and obtain an error which reflects how well the image points match the 3D

points. If there were no coincidental combinations with the same error value, the one with the smallest error should give the correct correspondences. In the section 3.3, I will introduce a novel and efficient technique to automatically search for the correspondences.

Once the correspondences are obtained, camera pose parameters can be solved by using a least-squares procedure. We use the following objective function for least squares.

$$\sum_i (I_{x_i} - q_{x_i})^2 + \sum_i (I_{y_i} - q_{y_i})^2 \tag{3.3}$$

where (I_{x_i}, I_{y_i}) are the detected(or selected) image coordinates of a 3D point, and (q_{x_i}, q_{y_i}) are the projected image coordinates of the same 3D point using the current pose parameters of the camera, i.e.

$$q_i = P(R(p_i - T)) \tag{3.4}$$

where q_i is the 2D image plane projection of the 3D point p_i, P is the perspective transformation, R is the rotation matrix and T is the translation vector. We assume here that the correspondence between these two pairs of coordinates is known. Therefore, their difference should be minimized. To minimize the above summation, we could either apply gradient-based optimization techniques or nongradient-based techniques such as the down-hill Simplex method in [99]. In case of gradient-based optimization, gradient can be approximately estimated using numerical differentiation with respect to each parameter.

It is possible to decouple the rotation and translation parameters and solve them in two steps. Three rotation parameters should be solved first. Assume there are a pair of 3D points p_1 and p_2(Fig. 3.1). Their corresponding image points are detected to be at I_1 and

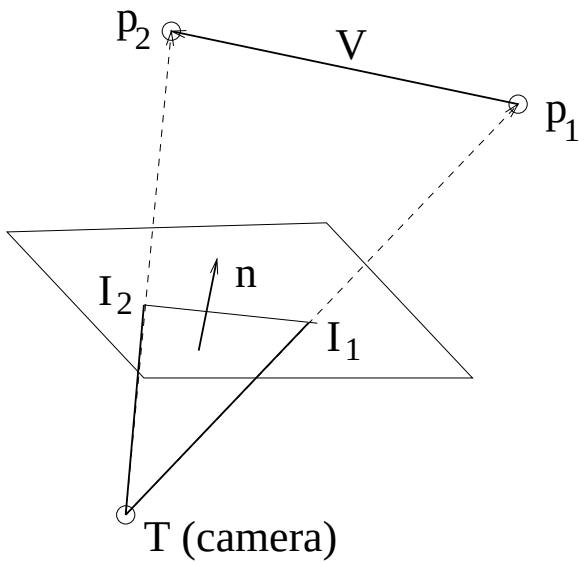


Figure 3.1: p_1 and p_2 are two 3D points. I_1 and I_2 are their projections on the image plane. The vector V is perpendicular to the vector n which is the normal to the plane passing through I_1, I_2 and the camera position.

I_2, respectively. There is a plane passing through I_1, I_2, and the origin of the local camera coordinate system which is the camera position itself. Assume the normal to this plane is n, and the vector between p_1 and p_2 is V. It can be easily verified that

$$V \cdot R\, n = 0 \tag{3.5}$$

where R is the rotation matrix converted from exponential coordinates. So one pair of 3D points gives one constraint like this. With multiple pairs, we can set up a least-squares problem to solve the 3 exponential coordinates.

If the translation vector is also unknown, the translation part can be solved using least-squares and constraints like the following once the rotation part is estimated.

$$(T - p_1) \cdot R\ n = 0 \tag{3.6}$$

where only vector T(camera position) is unknown.

3.2 Background and Related Work

The mathematical foundation for pose estimation from points, lines and curves has been extensively studied. Estimation techniques based only on point correspondences from four or more points can be found in [27] and [15, 20]. Haralick, et.al. [21] provides a review of many 3-point techniques with a careful analysis of their stability. Qiang, et.al. [42] develops an analytic least squares technique for pose estimation from points, lines and ellipse-circle pairs. These methods all assume a known correspondence between geometry and image features, which often requires extensive user involvement.

Several techniques have been developed for automatic detection of features in the image and the geometry and finding their correspondences. Most of these techniques are applicable to discrete geometric objects whose shape is known exactly. Huttenlocher and Ullman [28], find corners and run a combinatorial search to find matches. Wunsch and Hirzinger [55] propose another method based on the iterative closest point algorithm [2]. These methods, however, are very restrictive to the types of models they can handle, restricting themselves to simple CAD objects.

A compromise solution is to ask the user to suggest an initial pose by, for example, selecting a few point correspondences, and then using object silhouettes to refine the estimation. Neugebauer and Klein [37] uses this technique in addition to aligning the texture

maps on the surface. They require an exact model and numerous photographs of the object, conditions we are not guaranteed.

3.3 Image Registration

The material presented in this section was done jointly with Andras Ferencz.

To calculate the pose for arbitrary rotation and translation, we need to know correspondences between features in the image and the geometry. Automatically discovering suitable features (such as lines and corners) in general scenes and matching them is extremely difficult and currently no reliable methods exist. Another possibility is to ask the user to supply the features. However, this is very labor intensive and often not very accurate (users tend to label features with an accuracy of a few pixels at best, and their performance diminishes after the first dozen images). An alternative is to place unique calibration objects in the scene that are identifiable from both laser range data and images. With an ample number of such features in each image, the pose can be determined automatically and accurately without user intervention. The disadvantage of this method is that more planning must go into scene capture, ensuring that enough calibration objects are visible from each image, and that these artificial objects must then be removed from the scene. However, these limitations seem much less severe than the disadvantages of the other options.

Cyra Technologies' laser scanner, that we used for this project, can achieve best performance in registering multiple scans by using calibration targets taped to surfaces. We use these same targets to determine the camera pose. While these targets are specifically designed for this scanner, our techniques can be applied in general when calibration tar-

gets can be placed in the scene. These targets are flat square green patches with a white circular area in the middle. They are designed to be easily identifiable from both laser range and image data, while being small to cover as little of the surface as possible. These constraints, combined with a wide variety of lighting situations that inevitably changes the apparent cover of any object, prevent us from adding a unique identifier to each target. Thus for each image, finding the pose involves locating the targets in the image, finding the correspondences between these targets and known targets in the geometry, and finally calculating the six parameters of rotation and translation for the camera.

3.3.1 Finding Targets in Images

To find the targets in an image we first sweep target templates (white circles with green borders) of several scales over the image, using sum of squared distances (SSD) as a metric. As the circular targets actually project to elliptical patches in the image, template matching (with a liberal threshold) is only effective at locating candidate target locations (since ellipsoids centered at a point have three degrees of freedom, prohibitively many templates would be needed to accurately find targets using only template matching). To verify candidate regions, we attempt to fit an ellipse to the central white region, and evaluate the match, again using SSD. This is equivalent to matching against the best possible elliptical template. Since the ellipse is found using a region of the image, not just a few pixels, the amount of redundancy enables us to estimate the parameters of the ellipse to sub-pixel accuracy. Conveniently, this technique provides five parameters for each target, which can be used to greatly reduce the combinatorial search for target correspondences.

3.3.2 Finding Target Correspondences

Once targets in the image have been identified, we must find their correspondence to known target locations in the geometry. This can be posed as a combinatorial search problem: pick correspondences for enough targets to generate an overdetermined set of constraints, solve for the best pose and test the error. If the error is within a threshold, which, in turn, is based on the expected accuracy of the point locations, accept the conjecture. If we only use the location of the center of the ellipses (x_i, y_i), without any initial guess three correspondences is enough to find the six parameters of the pose to within four ambiguous locations, while four resolves the ambiguity and generates an overdetermined system. Unfortunately, this simple combinatorial search thus takes $O(n^4)$ time, where n is the number of targets in the geometry. Since a large scene may require a hundred or more targets (we used 66 for our room model), this search could be prohibitively expensive.

As noted above, we fit an ellipse to each target, yielding three additional parameters: major axis a_i, minor axis b_i, and rotation of the major axis to the vertical γ_i. Given a target in the geometry and an associated normal vector, we can compute the projection of the circle onto an arbitrary image plane giving a_g, b_g, γ_g. Let T_p and T_n be the position and normal of target T in the geometry, and C_p the camera location. For computational convenience we only consider image planes perpendicular to the vector $Q = C_p - T_p$ (i.e. where the target is at the center of the image). For targets not located at the center of the image, we reproject them onto such a plane and compute a_i, b_i and γ_i relative to this new plane (this needs to be done only once for each image target). Let C_{up} be the up vector for this new image plane, r be the physical radius of the inner circle of the targets, and f the

focal length of the camera. Given these, a_g, b_g, γ_g are computed by:

$$a_g = \frac{r}{\|Q\|} * f$$

$$b_g = a_g * (Q \cdot T_n)$$

$$\gamma_g = \cos^{-1}(\frac{T_n \times Q}{\|T_n \times Q\|} \cdot C_{up} * sign)$$

where

$$sign = (T_n \cdot C_{up})/|T_n \cdot C_{up}|.$$

Thus 2 target correspondences provide 10 parameters, which is enough to solve for a unique camera pose in the general case. We do this by anchoring the image target centers to their counterparts in the geometry, constraining the 6-dimensional system to a 2-dimensional manifold. We then minimize the function

$$\sum_{t=1}^{2} \left((a_{ti} - a_{tg})^2 + (b_{ti} - b_{tg})^2 + \left((\gamma_{ti} - \gamma_{tg}) * \frac{(a_{ti} - b_{ti}) + (a_{tg} - b_{tg})}{2} \right)^2 \right)$$

using a standard least squares optimizer. As this optimization is typically prone to a small number of local minima, we run it from multiple initial positions. A detailed treatment of pose estimation from circle/ellipse pairs can be found in [42].

Since this system with 10 equations and 6 unknowns is overdetermined, each solution returns an error that can be used to weed out most bad correspondences immediately. Otherwise, we use this initial guess for the pose to find a small set (typically less than 3) of candidate correspondences for each remaining image target. We try these one at a time, solving the optimization for a new pose given three targets, and using the remaining targets to confirm or reject the solution.

While this still has a worst-case running time of $O(n^4)$, for any practical arrangement of targets (without many targets clumped together) we expect the running time to be $\Omega(n^2)$. As reported in the results section, this is the observed behavior, which is much faster than previous algorithms.

3.3.3 Recovering Camera Pose

Once we find an acceptable set of correspondences, we fine tune the pose by solving a final least squares optimization, over all 6 parameters, using the previous estimate as an initial position. This optimization minimizes the function:

$$\sum_{t=1}^{m}(x_{ti} - x_{tg})^2 + (y_{ti} - y_{tg})^2$$

where (x_{tg}, y_{tg}) is the location of the projected center of target t in the image. (We do not attempt to fit a, b, γ in this function, for these vary much more slowly than the projection of the center points, so they become irrelevant when enough targets are available).

3.3.4 Results

To evaluate our camera pose estimation technique, we look at three aspects: a) the amount of user intervention, b) the accuracy of the resulting pose, and c) the computational cost of our algorithm. We ran our automatic algorithm on 62 images with four or more targets visible. In these images, the automatic target detector found 90% of the visible targets, while finding four false matches. These errors were easily correctable by prompting the user to localize the search. Poses were estimated correctly for 58 of the 62 images. The remaining 4 coincidentally lined up with an erroneous set of targets in the geometry. These

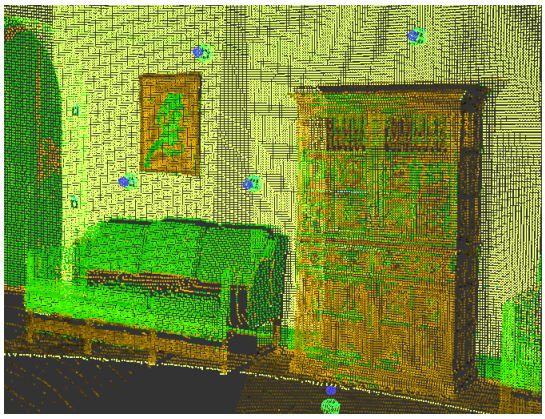
errors could also be easily corrected interactively by supplying one pair of correspondences. The amount of user intervention was approximately 15 minutes.

We found the accuracy of our estimated pose to be very high, typically within one pixel. An example of this is shown in Fig. 3.2 where a sample image is texture-mapped onto the geometry and the resulting surfaces are displayed from a different viewing direction (black areas in Fig. 3.2(c) indicate backfacing or occluded areas in the geometry) . Note the object boundaries in the image line up with geometric discontinuities in the scene.

As expected, our algorithm runs in $O(n^2)$ time for real-world inputs. For fifty targets in the geometry, the running time was 5.8 seconds, while for 100 targets, the algorithm took 21 seconds.

From those calibrated camera poses and simplified meshes , we synthetically composed 22 1024 by 512 texture maps that are used to render the original as well as the altered scene.

(a)



(b)

(c)

Figure 3.2: (a) A real photograph with targets located; (b) The scan cloud viewed from the recovered camera pose; (c) Low-resolution texture-mapping using a single photograph. Note the edge alignment between the image and the geometry.

3.4 Texture Map Synthesis and Compression

From photographs that have been aligned with the geometry, we wish to manufacture specialized texture maps, images broken up into triangular patches. The naive approach would allocate a fixed sized triangle in the texture map for each 3D triangle in the mesh and obtain texture coordinates for each vertex of the 3D triangle(Fig. 3.4). However this would be very costly, both in terms of memory and rendering speed.

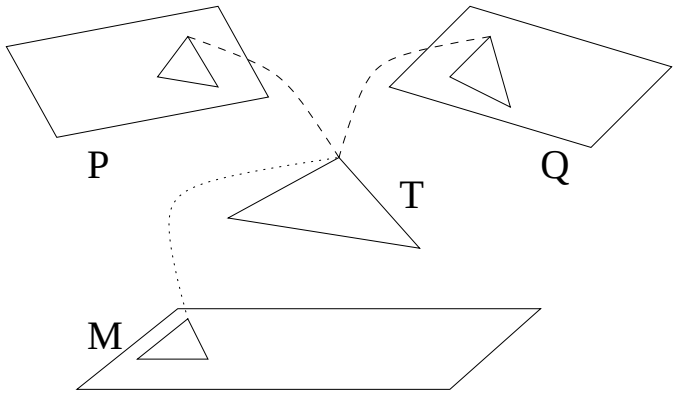


Figure 3.3: **Texture Map Synthesis** Assume 3D triangle T is covered by two photographs P and Q. We allocate a triangular texture patch in texture map image M corresponding to T, and populate it with a weighted average of the projected areas of T in photographs P and Q.

Our standard texture map creation scheme is as follows. Since each triangle in a mesh may be covered by multiple photographs, we actually synthesize one texture patch (a triangular shaped region) for each triangle to remove the redundancy. This texture patch is an appropriately weighted average of the projected areas of the triangle in all photographs(Fig. 3.3). The weight for each original area from photographs is set in such a way that the weight becomes smaller when the triangle is viewed from a grazing angle or its projected area is close to the boundaries of the photograph to obtain both good resolution and smooth tran-

Figure 3.4: An example of synthesized texture maps packed with triangular texture patches

sition among different photographs. Visibility is determined using Z-buffer for each pixel of each original area to make sure only correct colors get averaged. We use the scheme in [49] to place the synthetic triangular texture patches into texture maps, and therefore obtain texture coordinates. The basic idea to place texture patches can be considered as a generalization of memory allocation in operating systems to 2D situation where we initially have a set of unused rectangular texture maps. All the texture patches are first sorted in decreasing order in terms of their size. Every two subsequent texture patches are put together pairwise to form rectangular patches. Then they are inserted into the blank texture maps in that order. First-fit strategy is used to find an unallocated rectangular area in the maps for each pair. When texture-mapping using OpenGL APIs, we use the GL_NEAREST option instead of GL_LINEAR to prevent averaging and interpolation across texture patch boundaries.

At first, the size of each triangular texture patch is determined by the maximum number of pixels mapped onto it from any image. Thus, all else being equal, bigger triangles in the mesh will be allocated a bigger patch. But we run into problems with this approach,

because it generates too many and too big triangles. Graphics hardware has a limited amount of texture memory. If we cannot pack all texture information into that amount of space, we have to compose multiple texture maps and swap texture memory multiple times for each frame. Swapping texture memory is quite expensive (our SGI O_2 needs 0.04 second to load a 1024x512 image).

(a) (b)

Figure 3.5: (a) An original photograph; (b) the result of applying the derivative of the Gaussian to the image in (a). Areas with edges are allocated with more pixels during texture map synthesis.

One way that the above texture patch allocation method is wasteful is that it picks sizes for texture patches without considering the amount of variation in the photographs. However, we need fewer texture map image pixels (texels) to represent smooth areas than to encode highly varying regions. By using an information measure on the image, we can better decide the size of each texture patch. We use the response of an edge detection operator (the derivative of the Gaussian) as our information measure, and apply it to all original photographs (Fig. 3.5). For each texture patch, we use the maximum response at its corresponding pixels in the photographs to determine the number of texels it actually

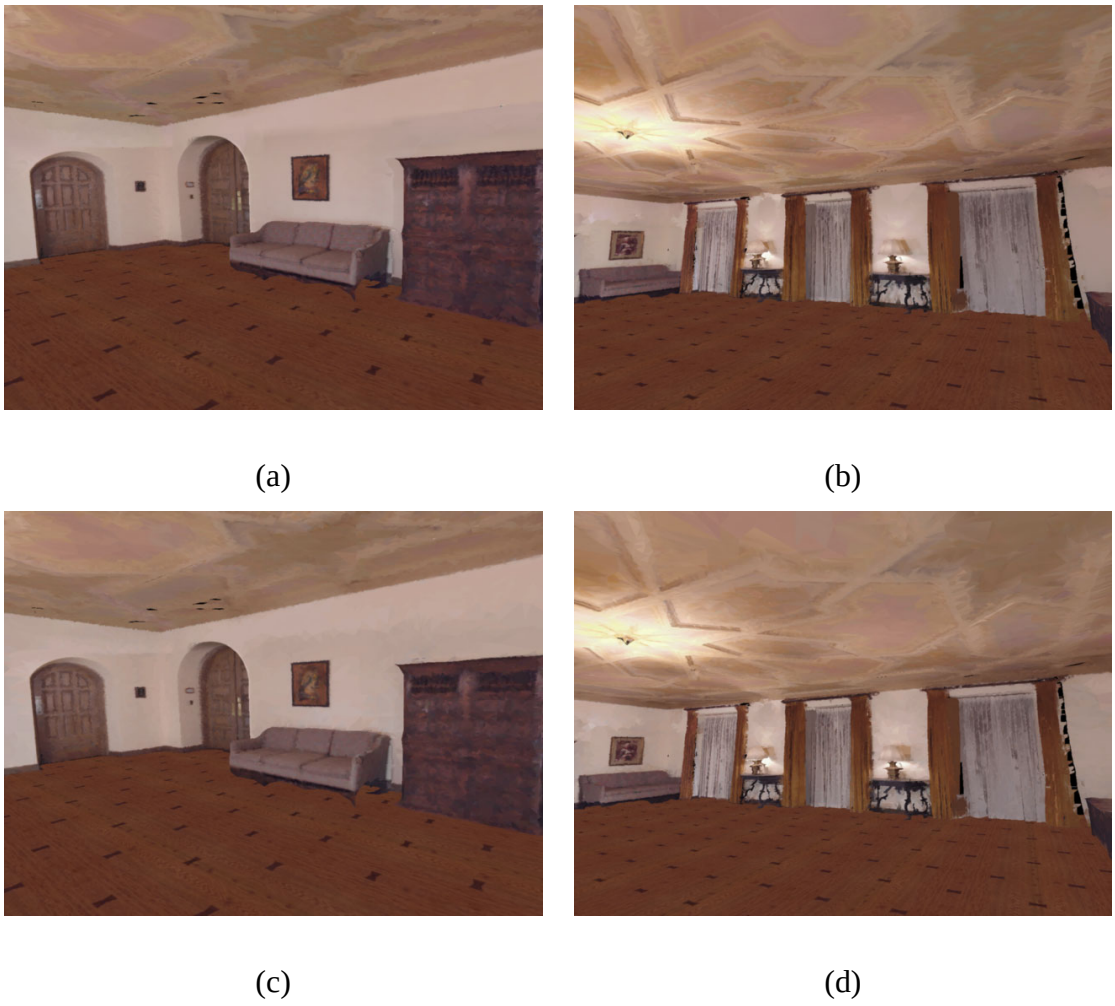
needs to keep the original color variations on the triangle. Thus smooth regions where the operator does not respond, are allocated few texels.

Additional savings can be achieved by *reusing* texture patches for multiple 3D triangles, when the texture over these triangles look similar. For example, if the walls in a room are all white, it is possible to represent the shading variations on the walls with a small number of texture patches even if the number of triangles for the walls is quite large. This requires that we cluster the texture patches from the previous section and set the same texture coordinates to all triangles in the same cluster.

We first quantize the edge length(number of texels along each edge) of every texture patch to be a power of 2. If a texture patch has N texels on each edge, it has $N(N+1)/2$ pixels in total, and can be considered as a vector of length $N(N+1)/2$. The K-mean algorithm (Lloyd algorithm) in vector quantization [104] can then be used to cluster all the texture patches with the same size. Because of Mach Band effect, slight color difference along the edge shared by two 3D triangles may be rather obvious. The K-mean algorithm adopts summed square difference as its objective function. We change it into a weighted summed square difference and use a larger weight for difference on edge texels to alleviate this effect. Given an error tolerance, we need to run a binary search to find the minimum number of clusters that can achieve that error. This process is quite time-consuming since each step of the binary search needs to run the K-mean algorithm whose complexity is $O(nmd)$ where n is the number of initial vectors, m is the number of clusters, and d is the dimensionality of each vector. This complexity becomes $O(n^2d)$ when m is a large fraction of n. We found out that a two-level scheme can optimize the performance by first grouping the n vectors into $\sqrt{n}$ clusters and then running the binary search on the vectors belonging

to each cluster which in turn splits into multiple clusters.

We have applied our texture map synthesis and texture patch resizing techniques to 45 high resolution images of a large room and obtained 20 1024 by 512 texture maps. Then we used our texture patch clustering algorithm on the 20 texture maps to obtain 5 new texture maps. The resulting compressed texture maps can still achieve good visual quality with a compression ratio of 4 (Fig. 3.6).



(a) (b)

(c) (d)

Figure 3.6: **A comparison** (a)-(b) Two synthetic images of a real room rendered with the original set of 20 texture maps; (c)-(d) two synthetic images rendered from the same viewpoints with the compressed set of 5 texture maps. The two pairs of images look similar.

3.5 Visibility Processing for Projective Texture-Mapping

In the previous sections, I have introduced techniques to conventional texture-mapping with texture coordinates. However, projective texture-mapping is an alternative way to map photographs onto recovered geometry. Projective texture mapping was first introduced in [56] and now has been implemented in OpenGL graphics package on SGI machines. In order to do projective texture mapping, the user needs to specify a virtual camera position and orientation, a virtual image plane with the textures. The texture is then cast onto a geometric model using the camera position as the center of projection.

For every image, we only want to map this image onto the polygons visible from the camera position where we took this image. We should not erroneously map it onto those occluded polygons. The current hardware implementation of projective texture mapping on SGI workstations cannot do this in hardware. It let the texture pierce through the geometry and get mapped onto all backfacing and occluded polygons on the path of the ray(Fig. 3.8(a)). So parts of the geometry that are occluded in the original image still receive legible texture coordinates and are incorrectly texture mapped instead of remaining in shadow. This indicates we need to obtain visibility information before texture-mapping.

We could solve this visibility problem in image-space using ray tracing or item buffer. But that means if we want to render a large number of frames for an animation or do a real-time demonstration, we need to compute visibility in image-space for each frame, which would be impractically slow. Hardware texture-mapping can be done in real-time if all the visibility information is known, which means we need a visibility preprocessing step in object-space to fully exploit the hardware performance. For any sequence of animation,

this object-space preprocessing needs to be done only once. It also allows the view point to be changed dynamically during a real-time demonstration.

What we need to do is to decide in which photographs a particular polygon from the geometric model is visible. If a polygon is partially visible in a photograph, we should clip it so that each resulting polygon is either totally visible or totally invisible. We only need to map the photograph to the visible ones. After this visibility processing, we can correctly and efficiently assign radiance values from the photographs to the visible polygons.

This algorithm is operated in both image space and object space. It is summarized as follows.

1. Give each original polygon an id number. If a polygon is subdivided later, all the smaller polygons generated share the same original id number.

2. If there are intersecting polygons, subdivide them along the intersecting line.

3. Clip the polygons against all image boundaries and user-specified planar polygons so that any resulting polygon is either totally inside or totally outside the desired texture regions.

4. Set each camera position as the viewpoint in turn, and render the original large polygons from the geometric model using their id numbers as their colors and Z-buffer hardware.

5. At each camera position, scan-convert each frontfacing polygon in software so we can know which pixels are covered by it. If at some covered pixel location, the retrieved polygon id from the color buffer is different from the id of the polygon cur-

rently under consideration, a potentially occluding polygon is found and it is tested in object-space whether it is really an occluding polygon.

6. Clip each polygon with its list of occluders in object-space.

7. Associate with each polygon a list of photographs to which it is totally visible.

Clipping in object-space does not take much time because we use the original large polygons in the hardware Z-buffering step, which results in a very small set of occluders for each polygon. So this algorithm has nearly the speed of image-space algorithms and the accuracy of object-space algorithms as long as the original polygons in the model are all larger than a pixel. Using identification(id) numbers to retrieve objects from Z-buffer is similar to the item buffer technique introduced in [59].

There are some variants of this algorithm. First, we can replace scan conversion with object-space uniform sampling. On each polygon, draw uniform samples and project these sample points onto the image plane to check if there are any occluding polygons. But scan-conversion is obviously faster. Second, under some circumstances, we need some data structure to maintain the connectivity of the polygons and T intersections are not allowed.

The objective of this algorithm is to minimize the number of polygons resulted from clipping to accelerate texture mapping at a later stage while safely detect all occluding polygons so that texture mapping are done correctly. To achieve this goal, the following two techniques are introduced, *conservative testing* and *shallow clipping*.

From the hardware Z-buffering results, we want to safely detect all occluding polygons. We donot want to miss any occluding polygons. But if a nonoccluding polygon is erroneously reported, that is fine because we can do object-space testing to verify if it is

really an occluding polygon and get rid of it if it is not.

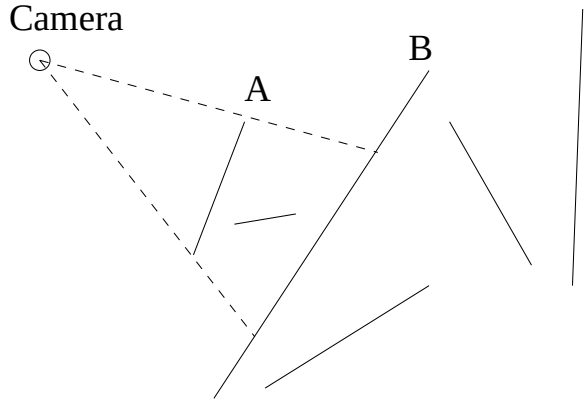


Figure 3.7: Our algorithm does *shallow clipping* in the sense that if polygon A occludes polygon B, we only use A to clip B, and any polygons behind B are unaffected.

Our algorithm does *shallow clipping* in the sense that if polygon A occludes polygon B, we only use A to clip B, and any polygons behind B are unaffected(Fig. 3.7). Only partially visible polygons are clipped. Those totally invisible ones are left intact. This is the major reason that our algorithm can minimize the number of resulting polygons.

By experiments, we found most polygons resulting from clipping are tiny polygons. To further reduce the number of polygons, we set a threshold on the size of polygons. If the object-space area of a polygon is below the threshold, it is not subdivided any more and is assigned a constant color based on the textures on its surrounding polygons. If a polygon is very small, it is not noticeable whether it has a texture on it or just a constant color. The rendered images can still maintain good quality.

To demonstrate our algorithm's efficiency in polygon clipping, we compare it with a pure object-space algorithm which is a modified version of Weiler-Atherton hidden surface removal algorithm [58, 105]. Since we have more than one camera positions in our situation, we apply Weiler-Atherton algorithm repeatedly at each camera position and the input

data for the processing at each camera position is the output data from the processing at the previous camera position.

The comparison was performed on two data sets. One is a very detailed model of a bell tower(Fig. 3.8(b)). The other is a coarse model of a university campus, including both the buildings and the terrain(Fig. 3.8(c)). These two models were recovered by the system in [12]. The number of polygons in the original models and the number of camera views used for visibility processing are shown in Table 3.1.

For fair comparison, in both algorithms we only process those frontfacing polygons falling into field of view at each camera position. The visibility processing results for the two models are shown in Fig. 3.9. The number of polygons generated after visibility processing are compared in Table 3.2-3.3 where our algorithm is named Hybrid. The comparison is done for three thresholds on polygon size. Any polygon smaller than the threshold is no longer subdivided. From these results, we can see our algorithm is significantly better in the number of polygons generated.

	Model I	Model II
# polygons	2409	660
# views	24	10

Table 3.1: Statistics for two geometric data sets: number of polygons and number of camera positions for visibility processing.

So far, we have successfully used this algorithm to process data in two applications. The first is for view-dependent projective texture-mapping [110](Fig. 3.10). This application has produced a fly-by animation for UC Berkeley campus [109]. Since our algorithm gener-

	Th = 0.0001	Th = 0.001	Th = 0.01
Hybrid	19018	7511	3584
WA	62556	12461	4779

	Th = 0.0001	Th = 0.001	Th = 0.01
Hybrid	112.14s	81.15s	61.47
WA	875.40s	234.97	175.53

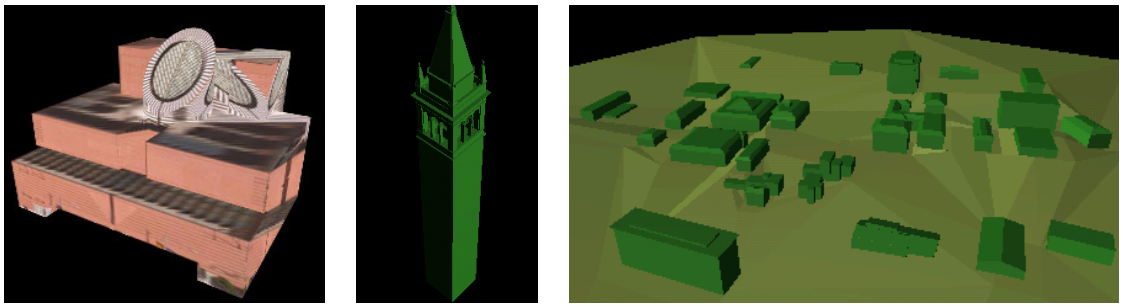
Table 3.2: Comparison between the Hybrid and Weiler-Atherton(WA) algorithms on Model I with three different thresholds on polygon size. The first half shows the number of polygons generated. The second half shows the running time in seconds on a SGI O_2 workstation.

ates much less polygons than previous algorithms, we are able to produce a corresponding real-time(60Hz) demonstration on SGI Onyx2 InfiniteReality Engine. The second application uses visibility processing to assign correct radiance values from photographs to their corresponding geometric surfaces and then recover the reflectance of the surfaces [111]. Thus we are able to re-render the model under novel lighting conditions such as a novel solar position for an outdoor scene.

	Th = 0.0001	Th = 0.001	Th = 0.01
Hybrid	5623	5541	5255
WA	10194	9878	8675

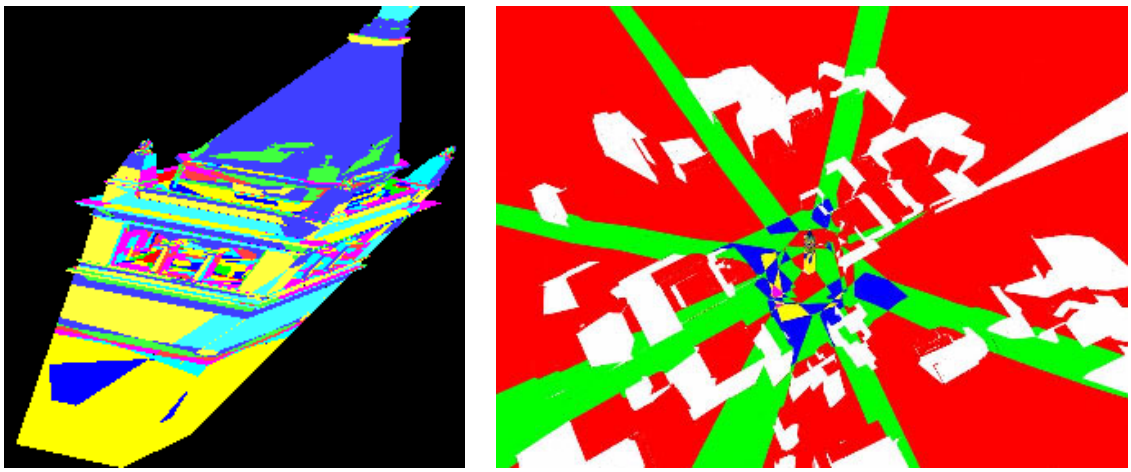
	Th = 0.0001	Th = 0.001	Th = 0.01
Hybrid	14.36s	14.63s	13.96s
WA	22.85s	21.60s	18.64s

Table 3.3: Comparison between the Hybrid and Weiler-Atherton(WA) algorithms on Model II with three different thresholds on polygon size. The first half shows the number of polygons generated. The second half shows the running time in seconds on a SGI O_2 workstation.



(a) (b) (c)

Figure 3.8: (a)Viewing the model from a viewpoint far from the original produces artifacts unless proper visibility pre-processing is performed, (b)A detailed bell tower model, (c)A model for a university campus, including both the buildings and the terrain.



(a) (b)

Figure 3.9: Visibility results for a bell tower model with 24 camera positions and for a university campus model with 10 camera positions. The color of each polygon encodes the number of camera positions from which it is visible: white= 0, red= 1, green= 2, blue= 3, yellow= 4, cyan= 5, magenta= 6 and unsaturated colors for larger numbers.

(a) (b)

Figure 3.10: Two re-rendered images of the university campus at two novel view points. The textures are actually from different photographs, but they seamlessly cover the geometry using visibility processing.

Chapter 4

Inverse Global Illumination

4.1 Introduction

Computer graphics is being increasingly used to visualize real objects and environments. Applications in entertainment, architecture, interior design, virtual reality, and digital museums often require that aspects of the real world be rendered realistically from novel viewpoints and/or under novel illumination. For example, one would want to see how a room in a house would look like with different lighting, or how a statue would look at various times of day in a different wing of a museum. Lastly, one might want to realistically render a film location in different lighting, and add in digital props and characters, with the expectation that the rendered results would be the same as what would have happened had it all been for real.

Whether it is changing the geometry or changing the lighting, generating a new rendering requires re-computing the interaction of light with the surfaces in the scene. Computing this interaction requires knowing the reflectance properties (diffuse color, shininess, etc.) of

each surface. Unfortunately, such reflectance property information is not directly available from the scene geometry or from photographs. Considerable work (e.g. [80, 71, 66, 44, 73]) has been done to estimate reflectance properties of real surfaces in laboratory settings from a dense set of measurements. However, reflectance properties of real scenes are usually spatially varying, and typically change with use and age, making *a priori* laboratory measurements impractical. It would clearly be preferable to estimate the reflectance properties of an entire scene at once, with the surfaces being illuminated *in situ* rather than as isolated samples, and from a relatively sparse set of photographs. This is difficult for two reasons.

The first is that we wish to use only a sparse set of photographs of the scene, rather than exhaustively photographing every point of every surface from a dense set of angles. With such a set of photographs, we can expect to observe each surface point from only a small number of angles. As a result, there will be too little data to determine fully general bi-directional reflectance distribution functions (BRDFs) for each surface. We address this problem in two ways. First, we limit ourselves to recovering low-parameter reflectance models of the surfaces in the scene. Second, we assume that the scene can be decomposed into areas with related reflectance properties. Specifically, we allow the diffuse reflectance, or *albedo*, of the object to vary arbitrarily over any surface; the estimated albedo is computed as an image called an *albedo map*[1]. In contrast, we require that the directional reflectance properties (such as specular reflectance and roughness) remain constant over each area. In this work, such areas are specified as part of the geometry recovery process.

The second problem we face is that in a real scene, surfaces will exhibit mutual illumi-

[1]The commonly used term *texture map* is sometimes used to refer to this same concept. However, texture maps are also sometimes used to store surface radiance information, which is not lighting-independent.

nation. Thus, the light that any particular surface receives will arrive not just from the light sources, but also from the rest of the environment through indirect illumination. As a result, the incident radiance of an observed surface is a complex function of the light sources, the geometry of the scene, and the as-yet-undetermined reflectance properties of all of the scene's surfaces. In this work, we use radiance data from photographs and image-based rendering to estimate the incident radiances of surfaces in the scene. This allows us to estimate the reflectance properties of the surfaces in the scene via an iterative optimization procedure, which allows us to re-estimate the incident radiances. We refer to this procedure as *inverse global illumination*.

Addressing these two problems makes it possible to robustly recover reflectance parameters from the limited radiance information present in a sparse set of photographs, and the accommodations made are appropriate for a wide variety of real scenes. Even when they are not met, the algorithm will compute the reflectance property parameters that best fit the observed image data, which in many cases can still yield a visually acceptable result.

The input to our algorithm is a geometric model of the scene, a set of radiance maps taken under known direct illumination, and a partitioning of the scene into areas of similar non-diffuse reflectance properties. The algorithm outputs a set of high-resolution albedo maps for the surfaces in the scene along with their specular reflectance properties, yielding a traditional material-based model. This output is readily used as input to traditional rendering algorithms to realistically render the scene under arbitrary lighting conditions. Moreover, modifications to the scene's lighting and geometry and the addition of synthetic objects is easily accomplished using conventional modeling methods.

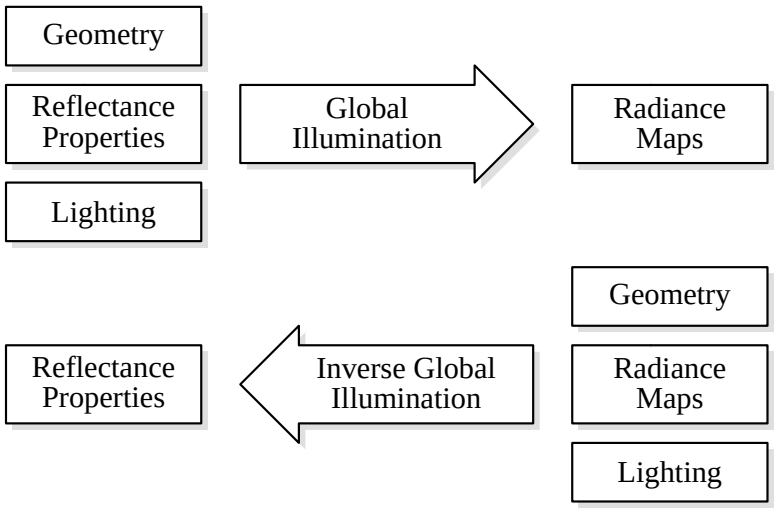


Figure 4.1: **Overview of the Method** This figure shows the relationship between global illumination and inverse global illumination. Global illumination uses geometry, lighting, and reflectance properties to compute radiance maps (i.e. rendered images), and inverse global illumination uses geometry, lighting, and radiance maps to determine reflectance properties.

4.2 Background and Related Work

The work presented here has been made possible by previous work in BRDF modeling, measurement and recovery, geometry acquisition, image-based rendering, and global illumination.

In graphics, there is a long history of modeling surface reflectance properties using a small number of parameters. Recent efforts in this direction include models introduced in [69, 80, 76, 89]. These models have been shown to yield reasonable approximations to the reflectance properties of many real materials, and they have been used to produce realistic renderings.

On the other hand, considerable recent work has presented methods for measuring and recovering the reflectance properties of materials using imaging devices. [80] and [71] pre-

sented techniques and apparatus for measuring reflectance properties, including anisotropic reflection. [66] measured directional reflectance properties of textured objects. [44] and [73] showed that diffuse and specular reflectance properties could be recovered from multiple photographs of an object under direct illumination. [67, 72] used a model of the scene and forward radiosity to estimate diffuse albedos to interactively modify the scene and its lighting. Although mutual illumination has been considered in the problem of shape from shading [74], it has not yet been fully considered for recovering non-diffuse reflectance properties in real environments. A survey of some of the methods is in Marschner [73].

Certain work has shown that changing the lighting in a scene does not necessarily require knowledge of the surface reflectance properties – taking linear combinations of a large set of basis images [75, 82] can yield images with novel lighting conditions.

Work in global illumination (e.g. [68, 70, 78, 108]) has produced algorithms and software to realistically simulate light transport in synthetic scenes. In this work we leverage the hierarchical subdivision technique [63, 64] to efficiently compute surface irradiance. The renderings shown here were produced using Gregory Ward Larson's RADIANCE system [81].

Photographs taken by a camera involve nonlinearities from the imaging process, and do not have the full dynamic range of real world radiance distributions. In this work we use the high dynamic range technique in [11] to solve these problems.

4.3 Inverse Radiosity

Most real surfaces exhibit specular as well as diffuse reflection. Recovering both diffuse and specular reflectance models simultaneously in a mutual illumination environment is complicated. In this section, we consider a simplified situation where all surfaces in an environment are pure diffuse (Lambertian). In this case, the global illumination problem simplifies considerably and can be treated in the radiosity framework [77]. We define *inverse radiosity* as recovering the diffuse albedo at each surface patch in the environment, provided that the geometry, the lighting conditions and the radiance distribution in the scene are known. In the next section we will discuss another simple case — recovering more general reflectance models with specularity considering only direct illumination — and we address the full problem in Section 4.5.

In the radiosity framework [77], the surfaces in the environment are broken into a finite number of patches. The partitioning is assumed to be fine enough that the radiosity and diffuse albedo of each patch can be treated as constant. For each such patch,

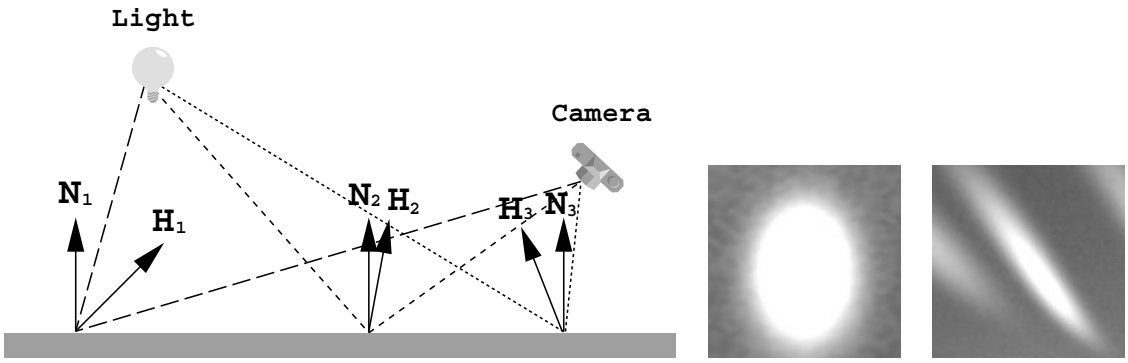
$$B_i = E_i + \rho_i \sum_j B_j F_{ij} \tag{4.1}$$

where B_i, E_i, and ρ_i are the radiosity, emission, and diffuse albedo, respectively, of patch i, and F_{ij} is the form-factor between patches i and j. The form-factor F_{ij} is the proportion of the total power leaving patch i that is received by patch j. It can be shown that this is a purely geometric quantity which can be computed from the known geometry of the environment [77].

We take photographs of the surfaces, including the light sources, to capture the radiance distribution. Since Lambertian surfaces have uniform directional radiance distributions,

one camera position is sufficient for each surface. Then B_i and E_i in Eqn. (4.1) become known. Form-factors F_{ij} can be derived from the known geometry. Once these are done, $\rho_i = (B_i - E_i)/(\sum_j B_j F_{ij})$. The solution to inverse radiosity is so simple because the photographs capture the final solution of the underlying light transport among surfaces.

4.4 Recovering Parameterized BRDFs from Direct Illumination



(a) (b) (c)

Figure 4.2: (a) The lighting and viewing directions at different points on a surface are different with respect to a fixed light source and a fixed viewpoint. This fact can be used to recover a low-parameter BRDF model for the surface from a single image. $\mathbf{N_i}$'s and H_i's are the normals and halfway vectors between lighting and viewing directions at different locations on the surface. We can infer that the surface point with normal N_2 is close to the center of the highlight, and the point with normal N_1 is relatively far away from the center. (b) An example of an isotropic specular highlight, (c) An example of an anisotropic specular highlight.

Before tackling the general case of reflectance recovery from photographs of mutually illuminated surfaces with diffuse *and* specular components, we study another special case.

Consider a single surface of uniform BRDF which is illuminated by a point light source in known position and photographed by a camera, also in a known geometric position with respect to the surface(Fig. 4.2). Every pixel in the radiance image provides a measurement of radiance L_i of the corresponding surface point P_i in the direction of the camera, and the known light source position lets us calculate the irradiance I_i incident on that point.

Our objective is to use these data (L_i, I_i) to estimate the BRDF of the surface. Since the BRDF is a function of four variables (azimuth and elevation of incident and viewing directions) it is obvious that the 2-dimensional set of measurements for a single camera/light source pairing is inadequate to do this in general. However for many materials it is possible to approximate the BRDF adequately by a parameterized BRDF model with a small number of parameters (e.g. Ward [80], Lafortune [89], He [69] etc). We use Ward's parameterization in which the BRDF is modeled as the sum of a diffuse term $\frac{\rho_d}{\pi}$ and a specular term $\rho_s K(\boldsymbol{\alpha}, \boldsymbol{\Theta})$. Here ρ_d and ρ_s are the diffuse and specular reflectance of the surface, respectively, and $K(\boldsymbol{\alpha}, \boldsymbol{\Theta})$ is a function of vector $\boldsymbol{\Theta}$, the azimuth and elevation of the incident and viewing directions, and parameterized by $\boldsymbol{\alpha}$, the surface roughness vector. For anisotropic surfaces $\boldsymbol{\alpha}$ has 3 components; for isotropic surfaces $\boldsymbol{\alpha}$ has only one component and reduces to a scalar. The precise functional form of $K(\boldsymbol{\alpha}, \boldsymbol{\Theta})$ in the two cases may be found in Appendix A.

This leads us to the following equation for each surface point P_i,

$$L_i = (\frac{\rho_d}{\pi} + \rho_s K(\boldsymbol{\alpha}, \boldsymbol{\Theta}_i))I_i \tag{4.2}$$

where L_i, I_i and $\boldsymbol{\Theta}_i$ are known, and the parameters ρ_d,ρ_s, $\boldsymbol{\alpha}$ are unknowns to be estimated. Depending on whether we are using an isotropic or anisotropic model for the specular

term we have a total of 3 or 5 unknown parameters, while there are as many constraining equations as the number of pixels in the radiance image of the surface patch. By solving a nonlinear optimization problem (see Appendix A for details), we can find the best estimate of ρ_d,ρ_s, $\boldsymbol{\alpha}$.

There are two important subtleties in the treatment of this optimization problem. One is that we need to solve a weighted least squares problem, otherwise the larger values from the highlight (with correspondingly larger noise in radiance measurements) cause a bias in parameter estimation. The second is the use of color information which needs to be done differently for dielectrics and metals. Both of these issues are discussed in Appendix A.

To obtain an obvious global minimum for this optimization problem and achieve robust parameter recovery, the radiance image should cover the area that has a specular highlight as well as some area with very low specular component. If the highlight is missing, we do not have enough information for recovering specular parameters, and can only consider the surface to be diffuse.

4.5 Recovering Parameterized BRDFs in a Mutual Illumination Environment

We are now ready to study the general case when the environment consists of a number of surfaces and light sources with the surface reflectances allowed to have both diffuse and specular components.

Consider a point P_i on a surface patch seen by camera C_v (Fig. 4.3). The radiance from

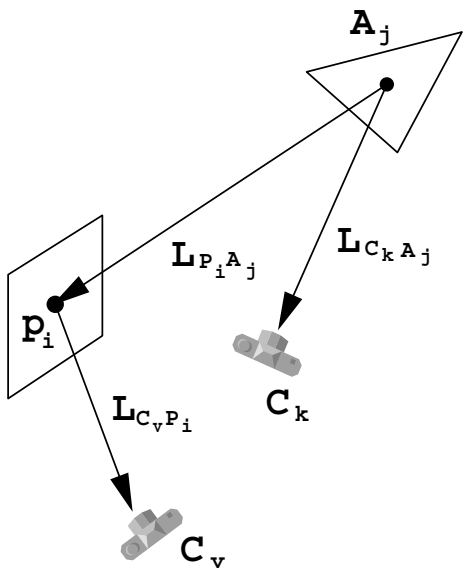


Figure 4.3: Patch A_j is in the radiance image captured by camera C_k. The specular component at A_j in the direction of sample point P_i is different from that in the direction of camera C_k. The difference is denoted by ΔS.

P_i in the direction of the camera is the reflection of the incident light contributed by all the light sources as well as all the surrounding surfaces. Eqn. (4.2) generalizes to

$$L_{C_v P_i} = E_{C_v P_i} + \rho_d \sum_j L_{P_i A_j} F_{P_i A_j}$$
$$+\rho_s \sum_j L_{P_i A_j} K_{C_v P_i A_j}, \tag{4.3}$$

where $L_{C_v P_i}$ is the radiance value in the direction of camera C_v at some sample point P_i on the surface, $E_{C_v P_i}$ is the emission in the direction of camera C_v, $L_{P_i A_j}$ is the radiance value along the direction from patch A_j to point P_i on the surface, $F_{P_i A_j}$ is the analytical point-to-patch form-factor [65] between sample point P_i and patch A_j, and $\rho_s K_{C_v P_i A_j}$ is the specular term evaluated at P_i for a viewpoint at camera C_v and a light source position at patch A_j. The arguments, $\boldsymbol{\alpha}$ and $\boldsymbol{\Theta}$, of K have been dropped to simplify notation.

As before, our objective is to estimate ρ_d, ρ_s, and specular roughness parameters $\boldsymbol{\alpha}$. Of

the other variables in Eqn. (4.3), $E_{C_v P_i} = 0$ for nonsources, and $L_{C_v P_i}$ can be measured directly from the radiance image at camera C_v. In general, the radiances $L_{P_i A_j}$ cannot be measured directly but have to be estimated iteratively. Suppose patch A_j in the environment appears in another radiance image taken by camera C_k(Fig. 4.3). Only if we assume A_j is Lambertian, does $L_{P_i A_j}$ in Eqn. (4.3) equal $L_{C_k A_j}$, the radiance from A_j to camera C_k. Otherwise, the diffuse components will be equal, but the specular components will differ.

$$L_{P_i A_j} = L_{C_k A_j} + \Delta S_{C_k P_i A_j} \tag{4.4}$$

Here $\Delta S_{C_k P_i A_j} = S_{P_i A_j} - S_{C_k A_j}$ is the difference between the specular components $S_{P_i A_j}$ and $S_{C_k A_j}$ of the radiances in the two directions. To compute the specular differences $\Delta S_{C_k P_i A_j}$, we need the BRDF of A_j, which is initially unknown. The estimation of ΔS (Section 4.5.1) therefore has to be part of an iterative framework. Assuming that the dominant component of reflectance is diffuse, we can initialize the iterative process with $\Delta S = 0$ (this sets $L_{P_i A_j} = L_{C_k A_j}$).

To recover BRDF parameters for all the surfaces, we need radiance images covering the whole scene. Each surface patch needs to be assigned a camera from which its radiance image is selected. At least one specular highlight on each surface needs to be visible in the set of images, or we will not be able to recover its specular reflectance and roughness parameters. Each sample point gives an equation similar to Eqn. (4.3). From these equations, we can set up a weighted least-squares problem for each surface as in Appendix 1. During optimization, we need to gather irradiance at each sample point from the surface patches in the environment. One efficient way of doing this is to subdivide each surface into a hierarchy of patches [63, 64] and link different sample points to patches at different levels in

the hierarchy. The solid angles subtended by the linked patches at the sample points should always be less than a prescribed threshold. There is a radiance value from the patch to the sample point and a ΔS associated with each hierarchical link.

For each sample point, we build hierarchical links to a large number of patches, and gather irradiance from these links. The amount of memory and computation involved in this process limits the number of samples for each highlight area. To make a reasonable tradeoff, we note that irradiance from indirect illumination caused by surrounding surfaces generally has little high-frequency spatial variation. Because of this, it makes sense to draw two sets of samples, one sparse set, and one dense set [2]. For the samples in the sparse set, we build hierarchical links and gather irradiance from the environment as usual. For the samples in the dense set, only their irradiance from light sources is computed explicitly, their irradiance from indirect illumination is computed by interpolation.

We are now ready to state the complete inverse global illumination algorithm. First detect all specular highlight blobs falling inside the radiance images using knowledge of the positions of the light sources, the camera poses, and the geometry (Fig. 4.4). Set the initial ΔS associated with each hierarchical link to zero. We can then recover an initial estimate of the BRDF parameters for each surface independently by solving a series of nonlinear optimization problems. The estimated specular parameters are used to update all ΔS's and $L_{P_i A_j}$'s associated with the hierarchical links. With the updated incident

[2] We choose the two sets of samples as follows. We first find the center of the highlight area in the image plane and rotate a straight line around this center to a number of different positions. The dense set of samples is the set of points on the surface corresponding to all the pixels on these lines. We choose the sparse set of samples on each line by separating two consecutive samples by some fixed distance in the object space.

```
For each camera position C
  For each polygon T
    For each light source O
      Obtain the intersection P between plane of T and line CO'
        (O' and O are symmetric about T);
      Check if P falls inside polygon T;
      Check if there is any occlusion between P and O;
      Check if there is any occlusion between C and any point
        in a local neighborhood of P;
      /* A highlight area is detected if P passed all the above tests.*/
End
```

Figure 4.4: The specular highlight detection algorithm.

radiances, we can go back and re-estimate the BRDF parameters again. This optimization and update process is iterated several times to obtain the final solution of the BRDFs for all surfaces. The overall algorithm is shown in Fig. 4.6.

4.5.1 Estimation of ΔS

Suppose there is a hierarchical link $l_{P_i A_j}$ between a sample point P_i and a patch A_j which is visible to a camera C_k (Fig. 4.5). The ΔS for $l_{P_i A_j}$ is defined to be the difference of the specular component in directions $\vec{A_j P_i}$ and $\vec{A_j C_k}$. To estimate this difference, we need to obtain the specular component along these two directions given the BRDF parameters of patch A_j. A one-bounce approximation of ΔS for link $l_{P_i A_j}$ can be obtained by using

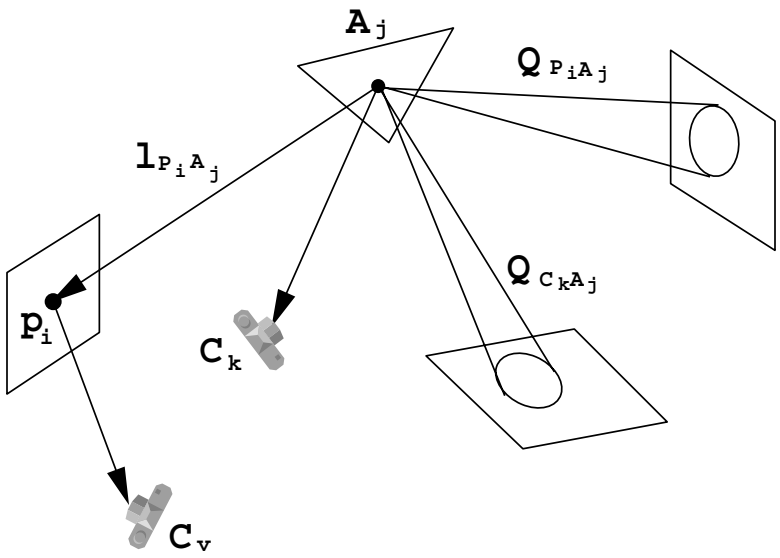


Figure 4.5: Random rays are traced around the two cones to obtain a one-bounce approximation of ΔS.

Monte Carlo ray-tracing [80]. Because of off-specular components, multiple rays should be traced and the direction of the rays is randomized around the mirror directions of $A_j\vec{P}_i$ and $A_j\vec{C}_k$, respectively. For each possible ray direction, the probability density of shooting a ray in that direction is proportional to $K(\boldsymbol{\alpha_j}, \boldsymbol{\Theta})$ where $\boldsymbol{\Theta}$ encodes the incident and outgoing directions. Intuitively, most of the rays fall inside the two cones $Q_{P_iA_j}$ and $Q_{C_kA_j}$ centered at the two mirror directions. The width of each cone depends on the specular roughness parameters $\boldsymbol{\alpha_j}$ of patch A_j. The radiance along each ray is obtained from the patch hit by the ray. Suppose $L_{Q_{P_iA_j}}$ and $L_{Q_{C_kA_j}}$ are the average radiance values of the rays around the two cones, respectively, and ρ_{sA_j} is the specular reflectance of patch A_j. Because the average value of Monte Carlo sampling approximates the total irradiance modulated by $K(\boldsymbol{\alpha_j}, \boldsymbol{\Theta})$, ΔS can simply be estimated as $\rho_{sA_j}(L_{Q_{P_iA_j}} - L_{Q_{C_kA_j}})$. This calculation could be extended to have multiple bounces by using path tracing [70]; we found that the one-bounce approximation was adequate for our purposes.

4.5.2 Practical Issues

We do not have a formal characterization of the conditions under which the inverse global illumination algorithm converges, or of error bounds on the recovered BRDF parameter values. In practice, we found it worked well (Section 4.7). Here we give some heuristic advice on how to acquire images to obtain good performance.

- *Use multiple light sources.* A specular highlight directly caused by one of the light sources should be captured on each surface. Having multiple light sources increases the probability that this can be achieved, and lets the whole scene receive more uniform illumination. This also increases the relative contribution of the diffuse component at any particular sample point P_i, and supports the $\Delta S = 0$ initialization, since highlights from different sources will usually occur at different locations on the surface.

- *Use concentrated light sources.* If the incoming radiance distribution is not very directional, the specular highlights will be quite extended and it will be difficult to distinguish the specular component from the diffuse one.

4.6 Recovering Diffuse Albedo Maps

In the previous sections, we modeled the reflectance properties as being uniform for each surface. In this section, we continue to do so for specular parameters because a small number of views of each surface does not provide enough information to reliably estimate specular parameters for each point individually. However, we relax this constraint on dif-

fuse albedo and model it as a spatially varying function, an *albedo map,* on each surface. The diffuse albedo for any point x on a surface is computed as:

$$\rho_d(x) = \pi D(x)/I(x) \tag{4.5}$$

where $\rho_d(x)$ is the diffuse albedo map, $D(x)$ is the diffuse radiance map, and $I(x)$ is the irradiance map.

Suppose there is an image covering the considered surface which gives a radiance map $L(x) = D(x) + S(x)$ where $S(x)$ is the specular radiance map seen from the image's camera position. Then the diffuse radiance map in Eqn. (4.5) can be obtained by subtracting the specular component from each pixel of the radiance map $L(x)$ using the specular reflectance parameters already recovered. We estimate the radiance due to specular reflection as the sum of specular reflection due to direct and indirect illumination. The specular reflection due to direct illumination is computed from the knowledge of the direct lighting and the estimated reflectance properties, and we estimate the indirect specular reflectance by tracing a perturbed reflected ray into the environment in a manner similar to that in Section 4.5.1.

The irradiance $I(x)$ can be computed at any point on the surface from the direct illumination and by using analytical point-to-patch form-factors [65] as in previous sections. For efficiency, we compute the irradiance due to the indirect illumination only at certain sample points on the surfaces, and interpolate these indirect irradiance estimates to generate estimates for all surface points x. Of course, care must be taken to sufficiently sample the irradiance in regions of rapidly changing visibility to the rest of the scene.

Something that complicates estimating diffuse albedos in this manner is that in highlight regions the specular component of the reflectance $S(x)$ will be much larger than the diffuse component $D(x)$. As a result, relatively small errors in the estimated $S(x)$ will cause large relative errors in $D(x)$ and thus $\rho_d(x)$. However, just as a person might shift her view to avoid glare while reading a movie poster, we make use of multiple views of the surface to solve this problem.

Suppose at a point x on a surface, we have multiple radiance values $\{L_k(x)\}_{k=1}^{p}$ from different images. The highest value in this set will exhibit the strongest specular component, so we simply remove this value from consideration. For the remaining values, we subtract the corresponding specular estimates $S_k(x)$ from the radiance values $L_k(x)$, to obtain a set of diffuse radiance estimates $D_k(x)$. We compute a final diffuse radiance component $D(x)$ as a weighted average of the $D_k(x)$, with weights inversely proportional to the magnitude of the estimated specular components $S_k(x)$ to minimize the relative error in $D(x)$. We also weight the $D_k(x)$ values proportionally to the cosine of the viewing angle of the camera in order to reduce the influence of images at grazing angles; such oblique images typically have poor texture resolution and exhibit particularly strong specular reflection. Since we are combining information taken from different images, we smooth transitions at image boundaries by gradually reducing the weight of an image towards the edges.

Once diffuse albedo maps are recovered, they could be used to separate the diffuse and specular components in the specular highlight areas. This would allow recovering more accurate specular parameters in the BRDF model. In practice, however, we have found good estimates to be obtained without further refinements.

```
Detect specular highlight blobs on the surfaces.
Choose a set of sample points inside and around each highlight
  area.
Build hierarchical links between sample points and patches in the
  environment and use ray tracing to detect occlusion.
Assign to each patch one radiance image and one average radiance
  value captured at the camera position.
Assign zero to ΔS at each hierarchical link.
For iter=1 to N
   For each hierarchical link,
      use its ΔS to update its associated radiance value;
   For each surface,
      optimize its BRDF parameters using the data
        from its sample points;
   For each hierarchical link,
      estimate its ΔS with the new BRDF parameters.
End
```

Figure 4.6: The *Inverse Global Illumination* algorithm.

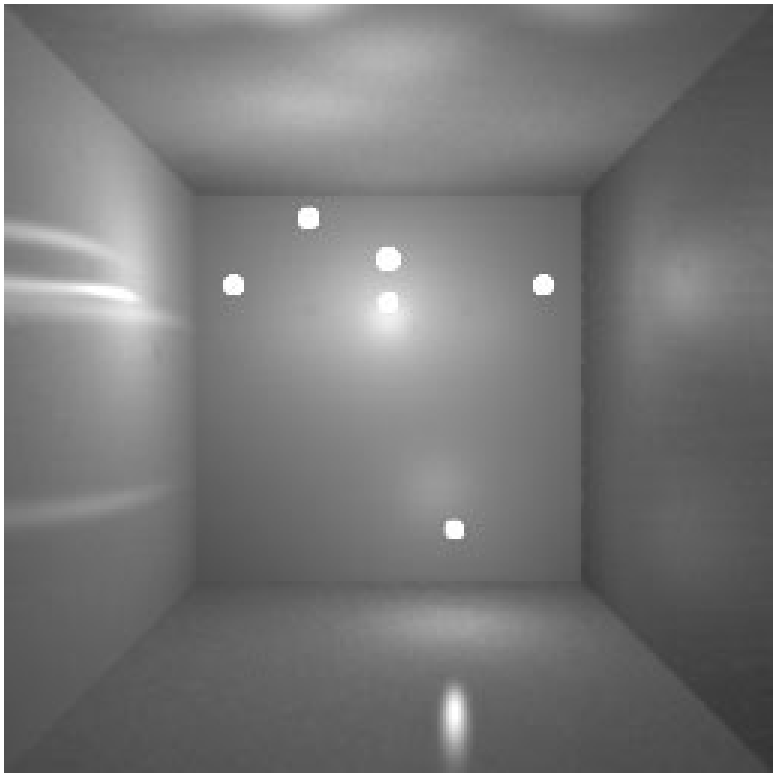
4.7 Results

4.7.1 Results for a Simulated Scene

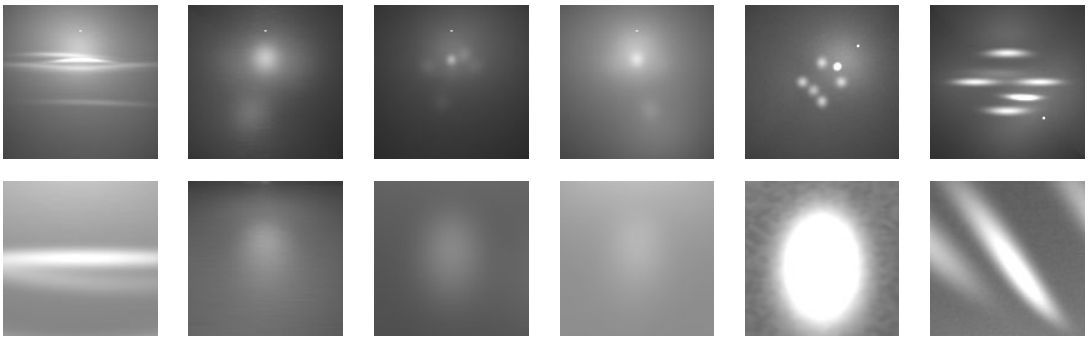
We first tested our algorithm on a simple simulated cubical room with mutual illumination. This allowed us to verify the accuracy of the algorithm and compare its results to ground truth. All the six surfaces of the room have monochromatic diffuse and specular components, but each one has a distinct set of parameters. Each of the surfaces has spatially uniform specularity. We assigned two surfaces to be anisotropically specular and added 10-20% zero mean white noise to the uniform diffuse albedo of two surfaces to simulate spatial variations. We used the RADIANCE rendering system [81] to produce synthetic photographs of this scene. Six of the synthetic photographs were taken from the center of the cube with each one covering one of the six surfaces. Another set of six zoomed-in photographs were taken to capture the highlight areas. The scene was illuminated by six point light sources so that specular highlights could be observed on each surface. These twelve images along with the light source intensity and positions were used to solve the BRDF parameters. The images of the specular highlights are shown in Fig. 4.7. Some of the highlights are visually very weak, but corresponding parameters can still be recovered numerically. The original and recovered BRDF parameters are given in Table 4.1. For the last two surfaces with noisy diffuse albedo, the recovered albedo values are compared to the true average values. The total running time for BRDF recovery is about half an hour on a SGI O_2 180MHz workstation.

The numerical errors shown in Table 4.1 are obtained by comparing the recovered parameters with the original ones. There are three sources of error: BRDF modeling error,

rendering error, and BRDF recovery error. BRDF modeling error comes from the inability of a given BRDF model to capture the behavior of a real material. By using the same model for recovery that RADIANCE uses for rendering, BRDF modeling error was eliminated for this test. However, because RADIANCE computes light transport only approximately, rendering error is present. We thus cannot determine the exact accuracy of our BRDF recovery. However, the test demonstrates that the algorithm works well in practice.



(a)



(b)

Figure 4.7: Synthetic grey-scale images of the interior of a unit cube in the presence of mutual illumination. (a) A wide-angle view of the cube with light sources(white dots) and specular(isotropic and anisotropic) highlights; (b) Actual images used for recovering the BRDF model of each surface. The top row shows the six images taken at the center of the cube with each one covering one of the six surfaces. The bottom row shows the six zoomed-in images taken to capture one specular highlight area on each surface. The first and last surfaces have anisotropic specular reflection. The last two surfaces have 20 and 10 percent zero mean white noise added to their diffuse albedo, respectively.

	ρ_d	ρ_s	$\alpha_x(\alpha)$	α_y	γ
True	0.3	0.08	0.6	0.03	0
Recovered	0.318296	0.081871	0.595764	0.030520	-0.004161
Error(%)	6.10	2.34	0.71	1.73	
True	0.1	0.1	0.3		
Recovered	0.107364	0.103015	0.300194		
Error(%)	7.36	3.02	0.06		
True	0.1	0.01	0.1		
Recovered	0.100875	0.010477	0.101363		
Error(%)	0.88	4.77	1.36		
True	0.3	0.02	0.15		
Recovered	0.301775	0.021799	0.152331		
Error(%)	0.59	8.90	1.55		
True	0.2	0.05	0.05		
Recovered	0.206312	0.050547	0.050291		
Error(%)	3.16	1.09	0.58		
True	0.2	0.1	0.05	0.3	45
Recovered	0.209345	0.103083	0.050867	0.305740	44.997876
Error(%)	4.67	3.08	1.73	1.91	

Table 4.1: Comparison between true and recovered BRDF parameters for the six surfaces of a unit cube. The first and last surfaces have anisotropic specular reflection. They have two more parameters: second roughness parameter α_y and the orientation γ of the principal axes in a local coordinate system. The errors shown are the combined errors from both rendering and recovering stages.

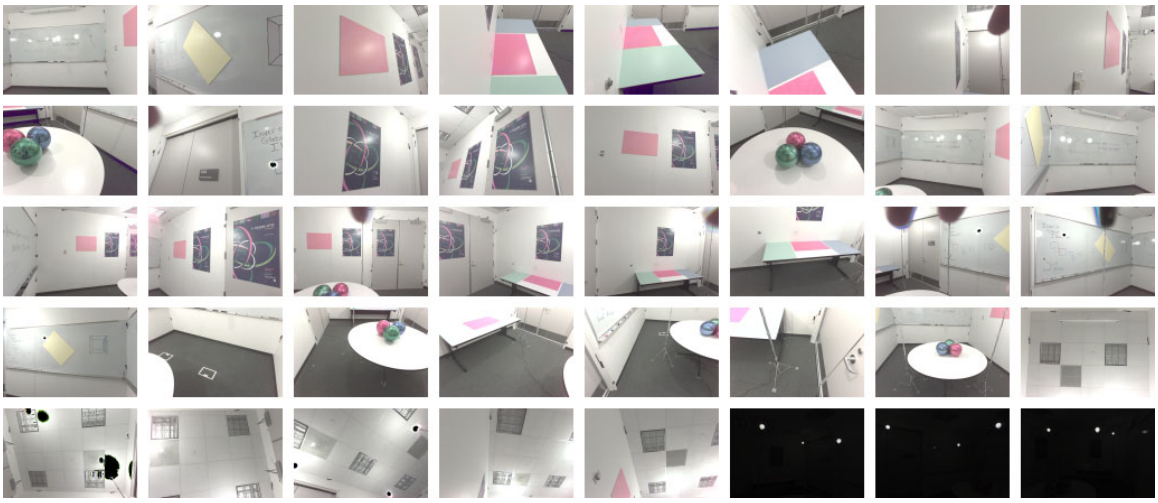


Figure 4.8: The complete set of forty radiance images of the room used to recover reflectance properties. Except for a few small areas, every surface in the room was seen in at least one radiance image. Each radiance image was constructed from between one and ten digital pictures depending on the dynamic range of the particular view. Black areas indicate regions which were saturated in all input images, and are not used by the recovery algorithm. The last three radiance images, reproduced ten stops darker than the rest, intentionally image the light bulbs. They were used to recover the positions and intensities of the sources.

4.7.2 Results for a Real Scene

In this section we demonstrate the results of running our algorithm on a real scene. The scene we chose is a small meeting room with some furniture and two whiteboards; we also decorated the room with colored cards, posters, and three colored metallic spheres Once the BRDFs of the materials were recovered, we were able to re-render the scene under novel lighting conditions and with added virtual objects.

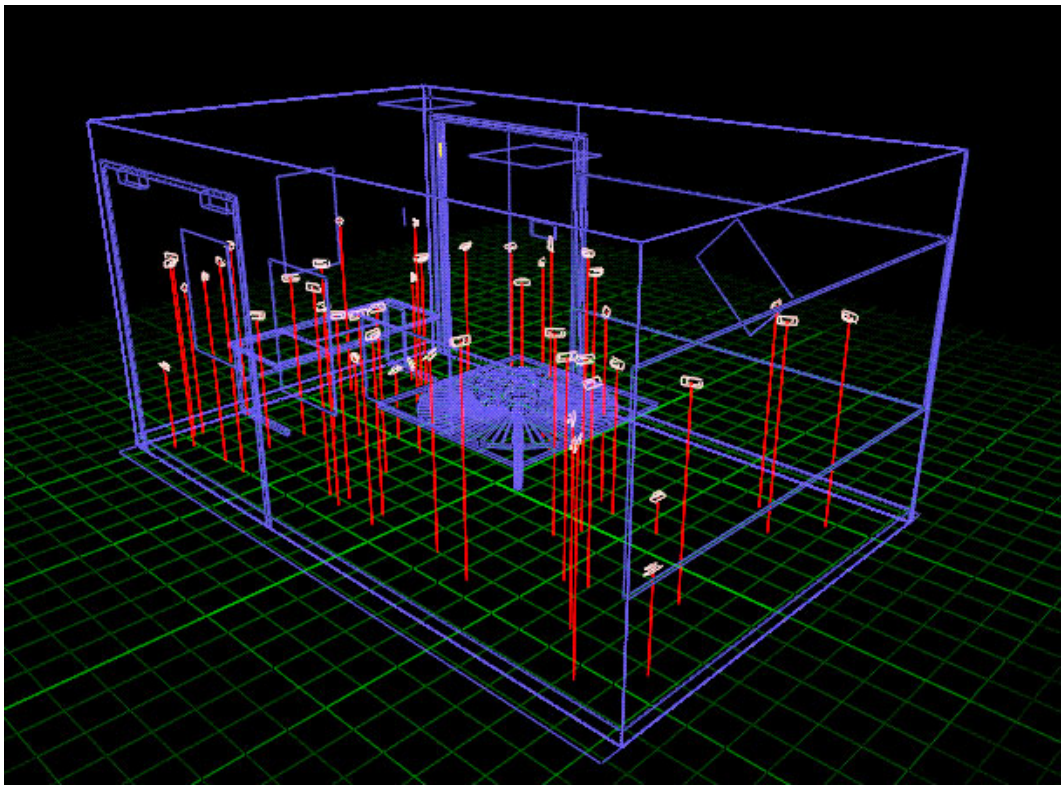


Figure 4.9: The model of the room, photogrammetrically recovered from the photographs in Fig 4.8. The recovered camera positions of the forty photographs are indicated.

Data Acquisition

We illuminated the scene with three heavily frosted 3-inch diameter tungsten light bulbs. Using high dynamic range photography, we verified that the lights produced even illumination in all directions. A DC power source was used to eliminate 60Hz intensity fluctuations from the alternating current power cycle.

We used a Kodak DCS520 color digital camera for image acquisition. The radiance response curve of the camera was recovered using the technique in [11]. We used a wide-angle lens with a 75 degree field of view so that we could photograph all the surfaces in the scene from a few angles with a relatively small number of shots. Forty high dynamic range radiance images, shown in Fig. 4.8, were acquired from approximately 150 exposures.

	ρ_d(red)	ρ_d(green)	ρ_d(blue)	ρ_s(red)	ρ_s(green)	ρ_s(blue)	α
whiteboard	0.5794	0.5948	0.6121	0.0619	0.0619	0.0619	0.0137
roundtable top	0.7536	0.7178	0.7255	0.0366	0.0366	0.0366	0.0976
door	0.6353	0.5933	0.5958	0.0326	0.0326	0.0326	0.1271
wall	0.8543	0.8565	0.8036	0.0243	0.0243	0.0243	0.1456
poster	0.1426	0.1430	0.1790	0.0261	0.0261	0.0261	0.0818
red card	0.7507	0.2404	0.3977	0.0228	0.0228	0.0228	0.0714
yellow card	0.8187	0.7708	0.5552	0.0312	0.0312	0.0312	0.1515
teal card	0.4573	0.5951	0.5369	0.0320	0.0320	0.0320	0.1214
lavender card	0.3393	0.3722	0.4437	0.0077	0.0077	0.0077	0.1144
red ball	0	0	0	0.5913	0.1862	0.3112	0
green ball	0	0	0	0.2283	0.3694	0.3092	0
blue ball	0	0	0	0.2570	0.3417	0.4505	0

Table 4.2: BRDF parameters recovered for the materials in the test room. All of them are isotropic, and most of them are plastic. The balls are metallic.

Twelve of the images were taken specifically to capture specular highlights on surfaces.

The radiance images were processed to correct for radial light falloff and radial image distortion. Each of these corrections was modeled by fitting a polynomial of the form $1 + ar^2 + br^4$ to calibration data captured with the same lens settings used for the scene images. To reduce glare and lens flare, we shaded the lens from directly viewing the light sources in several of the images. Regions in the images corresponding to the light stands (which we did not model) or where excessive remaining glare was apparent were masked out of the images, and ignored by the algorithm. The thin cylindrical light stands which

appear in the synthetic renderings have been added to the recovered model explicitly.

The radiance images were used to recover the scene geometry and the camera positions (Fig. 4.9) using the Façade [12] modeling system. Segmentation into areas of uniform specular reflectance was obtained by having each polygon of each block in the model (e.g. the front of each poster, the surface of each whiteboard, the top of each table) have its own uniform specular reflectance parameters.

The positions and intensities of the three light sources were recovered from the last three dynamic range radiance images. The positions were recovered using triangulation. During BRDF recovery, the area illumination from these spherical light sources was computed by stochastically casting several rays to each source.

BRDF Recovery

Given the necessary input data, our program recovered the surface BRDFs in two stages. In the first stage, it detected all the highlight regions and recovered parametrized BRDFs for the surfaces. In this stage, even if a surface had rich texture, only an average diffuse albedo was recovered. Surfaces for which no highlights were visible the algorithm considered diffuse. The second stage used the recovered specular reflection models to generate diffuse albedo maps for each surface by removing the specular components.

The running time for each of the two stages was about 3 hours on a Pentium II 300MHz PC. The results show our algorithm can recover accurate specular models and high-quality diffuse albedo maps. Fig. 4.10 shows how specular highlights on the white board were removed by combining the data from multiple images. Fig. 4.11 shows the albedo maps obtained for three identical posters placed at different places in the room. Although the

posters were originally seen in different illumination, the algorithm successfully recovers very similar albedo maps for them. Fig. 4.12 shows that the algorithm can remove "color bleeding" effects: colors reflected onto a white wall from the cards on the table do not appear in the wall's diffuse albedo map. Table 4.2 shows the recovered specular parameters and average diffuse albedo for a variety of the surfaces in the scene. We indicated to the program that all the materials are isotropic, and that the metallic spheres only have ideal specular components[3].

Re-rendering Results

We directly compared synthetic images rendered with our recovered BRDF models to real images. In Fig. 4.13, we show the comparison under the original lighting conditions in which we took the images for BRDF recovery. In Fig. 4.14, we show the comparison under a novel lighting condition obtained by removing two of the lights and moving the third to a new location, and adding a new object. There are a few differences between the real and synthetic images. Some lens flare appears in the real images of both figures, which we did not attempt to simulate in our renderings. We did not model the marker trays under the whiteboards, so their shadows do not appear in the synthetic images. In Fig. 4.14, a synthetic secondary highlight caused by specular reflection from the adjacent whiteboard

[3]For surfaces that have only ideal specular reflection, such as mirrors, there is no diffuse component and the roughness parameter is zero. We can still recover their specular reflectance ρ_s from a single image by noting that the specular reflectance can be computed as the simple ratio between two radiance values. One is the radiance value in the image corresponding to the intersection between the surface and a ray shot from the camera position; the other is the radiance value of the environment along the reflected ray. In practice, we shoot a collection of rays from the camera position to obtain the average reflectance.

appears darker than the one in the real image, which is likely due to RADIANCE's approximations for rendering secondary specularities. However, in both figures, real and synthetic images appear quite similar.

Fig 4.15 shows four panoramic views of the rendered scene. (a) shows the hierarchical mesh with the initial estimates of radiance obtained from the images. (b) shows the entire room rendered in the original illumination. (c) shows the entire scene rendered with novel lighting. The original lights were removed and three track lights were virtually installed on the ceiling to illuminate the posters. Also, a strange chandelier was placed above the spheres on the table. The new lights reflect specularly off of the posters and the table. Since the chandelier contains a point light source, it casts a hard shadow around the midsection of the room. The interior of the chandelier shade is turquoise colored which results in turquoise shadows under the spheres. A small amount of synthetic glare was added to this image. (d) shows the result of adding synthetic objects to various locations in the room, including two chairs, a crystal ball, two metal boxes, and a floating diamond. In addition, a very large orange sculpture, was placed at the back of the room. All of the objects exhibit proper shadows, reflections, and caustics. The sculpture is large enough to turn the ceiling noticeably orange due to diffuse interreflection.

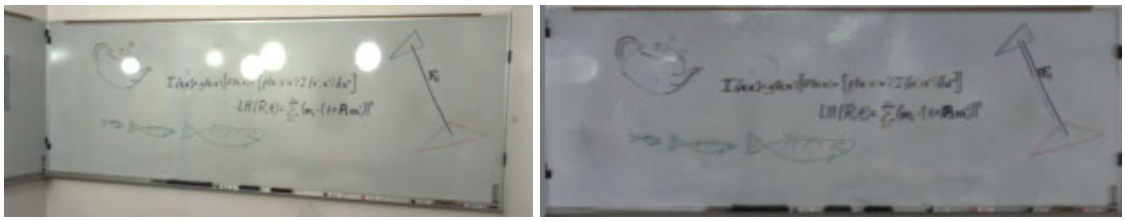


Figure 4.10: The left picture is a radiance image of a whiteboard, showing strong specular highlights. The right picture shows the diffuse albedo map of the whiteboard recovered from several images. Unlike the radiance image, the diffuse albedo map has a nearly uniform background, and is independent of the illumination.

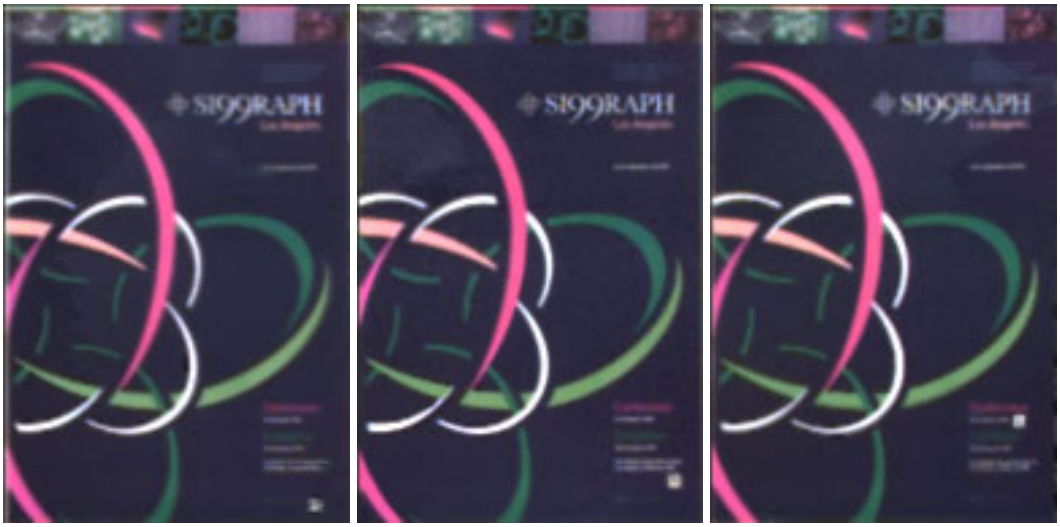


Figure 4.11: The diffuse albedo maps of three posters with the same texture. The posters were placed at different locations in the real scene with different illumination. Nonetheless, the recovered albedo maps are nearly the same. For identification purposes, a small yellow square was placed in a different location on the lower right of each poster.

Figure 4.12: The left image shows a part of a wall that becomes noticeably colored from light reflecting from the cards placed on the table below, an effect known as ”color bleeding”. The right image shows the recovered albedo map of the same part of the wall. It is nearly uniform, showing that the color bleeding was properly accounted for. The black line indicates where the table top aligned with the wall.

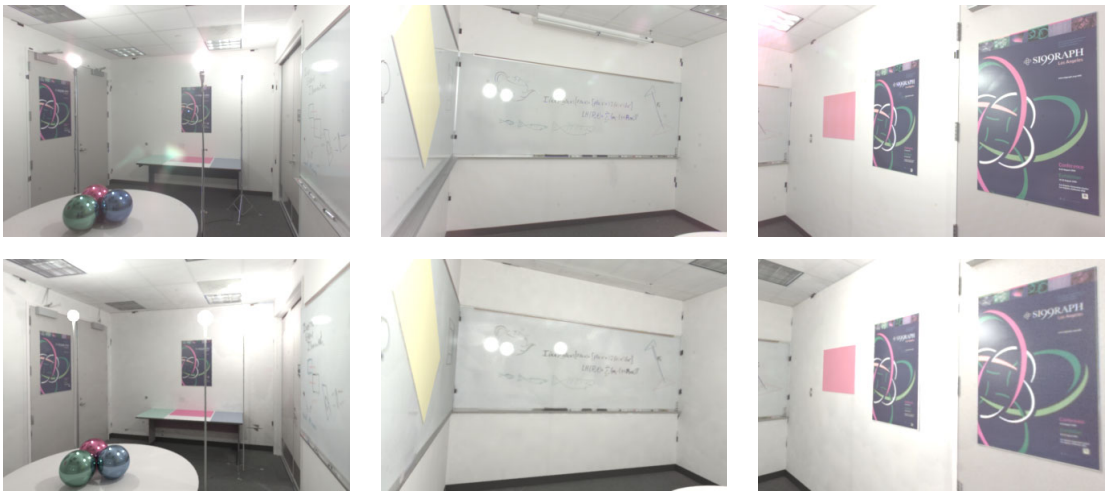


Figure 4.13: A comparison between real images (top) and synthetic renderings of our room with the recovered reflectance parameters (bottom). The simulated lighting is the same as in the original pictures, and the synthetic viewpoints have been matched to the recovered camera positions of the real images. The images show that good consistency was achieved.

Figure 4.14: A comparison between real and virtual, this time with novel lighting. Two of the lights were switched off and the third was moved to a new location. In addition, a real mirrored ball was placed on the red card. The scene was photographed from two locations and these real views are shown in the top row. To render the bottom row, we recovered the camera positions and light source position in the top views, estimated the material properties and position of the ball, and added a virtual ball to the model. The main noticeable difference is camera glare; however, some inaccuracies in the model (e.g. the whiteboard marker tray was not modeled) are also apparent. Otherwise, the illumination of the scene and appearance and shadows of the synthetic object are largely consistent.

(a) Initial hierarchical polygon mesh, with radiances assigned from images.

(b) Synthetic rendering of recovered properties under original illumination.

(c) Synthetic rendering of room under novel illumination.

(d) Synthetic rendering of room with seven virtual objects added.

Figure 4.15: Panoramic renderings of the room with various changes to lighting and geometry.

Chapter 5

Modeling and Recovering Illumination and Reflectance for Outdoor Architectural Scenes

5.1 Introduction

It is light that reveals the form and material of architecture. In keeping with its rhythms of light and dark, clear and cloudy, the architecture evokes distinct visual moods and impressions, something that many photographers and painters have sought to capture. Perhaps the most noteworthy of these attempts is the famous series of studies of the Cathedral at Rouen by Claude Monet–he painted the same facade at many different times of day and in different seasons of the year, seeking to capture the different 'impressions' of the scene.

The goal of this section is to develop this theme in the context of computer graphics. To produce renderings under new outdoor lighting conditions, we solve a series of optimiza-

tion problems to find the parameters of appropriate lighting and reflectance models that best explain the measured values in the various photographs of the scene. The lighting models include those for the radiance distribution from the sun and the sky, as well as a landscape radiance model to consider the effect of illumination from the secondary sources in the environment. Note that illumination from these secondary sources, such as the ground near the floor of a building can be very important and is often the dominant term in shadowed areas. To have sufficient data for parameter recovery, we take several photographs–of the sun, the sky, the architecture, and the environment surrounding the architecture. This enables us to recover radiance models for the sun, sky and environment for that time of day. The process is repeated for a few different times of the day; collectively all these data are used to estimate the reflectance properties of the architecture. It is assumed that a geometric model of the architecture had previously been created using a modeling system, so at this stage enough information is available to rerender the building under novel lighting conditions. The data-flow diagram of the system is given in Figure 5.1.

There are several technical challenges that must be overcome. We highlight a few of them here:

1. The photographs do not directly give us radiance measurements–there is a nonlinear mapping which relates the digital values from the photograph to the radiance in the direction of that image pixel. This can be estimated using the technique from [11], and subsequent processing performed using radiance images.

2. Measurements that we make from photographs are not sufficient to recover the full spectral BRDF. We need to define a new concept, the pseudo-BRDF associated with

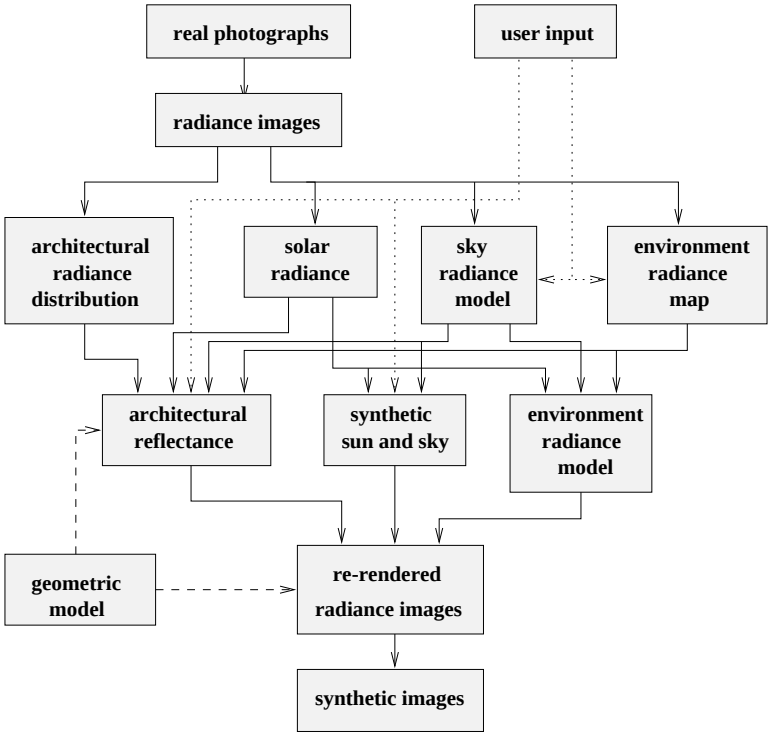


Figure 5.1: Data-flow diagram of the architectural re-rendering system.

a particular spectral distribution of the illuminant. This is done in Section 5.2. Our system is based on recovering pseudo-BRDFs for the architecture, and then subsequently using them for re-rendering. We recover two pseudo-BRDFs, one corresponding to the spectral distribution of the sun and one corresponding to the integrated light from the sky and landscape.

3. Producing renderings of the scene at novel times of day requires being able to predict lighting from the sun, sky and environment at such times. For the sun and sky, we rely on interpolated/extrapolated radiance models of the sun and sky (Section 5.5). Prediction of radiance from the environment at a novel time requires use of the computer vision technique of photometric stereo to recover a low resolution surface normal map of the environment, which can then be used in conjunction with the new

sun position to yield the new environment radiance map.

5.2 The Pseudo-BRDF Concept

The traditional way to formally define reflectance is using the concept of the bidirectional reflectance distribution function (BRDF) defined as follows:

$$\rho(\theta_i, \phi_i, \theta_r, \phi_r, \lambda) = \frac{dI(\theta_r, \phi_r, \lambda)}{I(\theta_i, \phi_i, \lambda) cos\theta_i d\omega_i} \tag{5.1}$$

where $I(\theta_i, \phi_i, \lambda)$ is the incident radiance and $dI(\theta_r, \phi_r, \lambda)$ is the reflected differential radiance.

Note the dependence on wavelength λ. There has been some previous work using a spectrophotometer to carefully measure spectral BRDFs [66]. However, we concluded that it is impractical to use such a technique to measure the BRDFs of complex, outdoor scenes. Our philosophy is to work with whatever information can be extracted from photographs, and we will use just an ordinary handheld digital video camcorder to acquire these photographs. Assume that the camera is geometrically calibrated, permitting us to identify ray directions from pixel locations.

In such a photograph, the value V obtained at a particular pixel in a particular channel (R, G, B) is the result of integration with the spectral response function $R(\lambda)$

$$V = \int R(\lambda) E(\lambda) d\lambda \ . \tag{5.2}$$

where $E(\lambda)$ is the incident radiance.

Suppose we take photographs of an area light source and of an object illuminated by this light source. Let us check the impact of this spectral integration over the traditional BRDF reflection model. What we can get from the photograph of the area light source is

$$I_{image}(\theta_i, \phi_i) = \int I(\theta_i, \phi_i, \lambda) R(\lambda) d\lambda \tag{5.3}$$

and what we can get from the photograph of the object is

$$I_{image}(\theta_r, \phi_r) = \int I(\theta_r, \phi_r, \lambda) R(\lambda) d\lambda$$

$$= \int \int I(\theta_i, \phi_i, \lambda) \rho(\theta_i, \phi_i, \theta_r, \phi_r, \lambda) R(\lambda) d\lambda \; cos\theta_i d\omega_i. \tag{5.4}$$

If we follow the definition of BRDF, but use $I_{image}(\theta_i, \phi_i)$ and $I_{image}(\theta_r, \phi_r)$ instead, we can define the following quantity which we will call the *pseudo-BRDF*

$$\rho_{pseudo}(\theta_i, \phi_i, \theta_r, \phi_r) = \frac{dI_{image}(\theta_r, \phi_r)}{I_{image}(\theta_i, \phi_i) cos\theta_i d\omega_i} \tag{5.5}$$

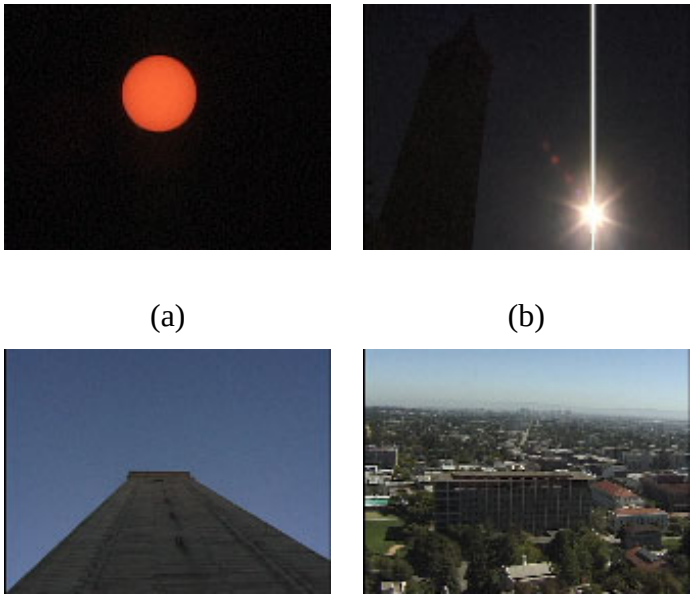
$$= \frac{\int I(\theta_i, \phi_i, \lambda) \rho(\theta_i, \phi_i, \theta_r, \phi_r, \lambda) R(\lambda) \; d\lambda}{\int I(\theta_i, \phi_i, \lambda) R(\lambda) \; d\lambda} \tag{5.6}$$

We note some properties of the pseudo-BRDF here:

- The pseudo-BRDF is equal to the real BRDF when the real BRDF does not vary with the wavelength. So they usually are not the same.
- In general, the pseudo-BRDF varies as the spectral distribution of the light source varies.
- If the spectral response function $R(\lambda) = \delta(\lambda - \lambda_0)$, then $\rho_{pseudo}(\theta_i, \phi_i, \theta_r, \phi_r) = \rho(\theta_i, \phi_i, \theta_r, \phi_r, \lambda_0)$.

Suppose we have a geometric model of some building. For the purpose of re-rendering under different lighting conditions, we need to recover the reflectance of the faces in the model. Since only pseudo-BRDFs can be recovered directly from photographs for each color channel and pseudo-BRDFs are sensitive to the spectral distribution of the light source, theoretically, we should divide the sky and the environment into small regions which have almost uniform spectral distributions spatially and recover distinct pseudo-BRDFs for each region. This is impractical because all these regions have their lighting effects on the considered architecture altogether and it is impossible to turn on only one of them and shut down the rest to recover individual pseudo-BRDFs. What we want to do is to recover as few pseudo-BRDFs as possible, but still get good approximations in rendering. It is possible to separate the sun from the sky since the sun moves across the sky during a day and a face of a building can be lit or unlit at different times. This has the same effect as turning the sun on or off for that face. It is also necessary to do this separation because the sun is the most important light source and its spectral distribution is so different from the blue sky. As to the rest of the sky and the environment, we find from experiments that recovering only one set of pseudo-BRDFs for them works very well. From now on, we will always recover two sets of pseudo-BRDFs, one of which corresponds to the spectral distribution of the sun, and the other to the integrated effect of the sky and environment. They will be used for re-rendering under novel lighting conditions under the assumption that the spectral distribution of daylight does not change much. Under extreme conditions, sunrise and sunset, we may expect these pseudo-BRDF's to cease being accurate.

5.3 Measuring and Modeling Illumination



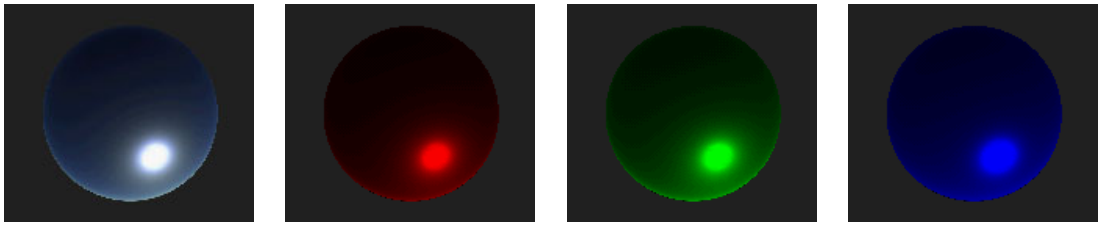
(a) (b)

(c) (d)

Figure 5.2: (a)Solar image obtained using a couple of neutral density filters, (b) Solar aureole obtained using fast shutter speed, (c) a photograph for the zenith, (d) a photograph for the landscape and the sky near the horizon.

We consider three sources of illumination. Light can be from the sun, the sky and the surrounding environment which serves as a secondary light source. Of course, in some fundamental sense, the sun is the only true light source. Both skylight and the light from the environment are ultimately derived from the sun. However, with an image-based approach, we need to measure and model these three sources separately. We shall not be constructing a physically correct global illumination model of the atmosphere and environment taking into account all the scattering and reflection effects!

To model these illumination sources, we take photographs of the sun, the sky and environment using a handheld CCD camera. To accurately measure the radiance, we need



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

Figure 5.3: (a) A sky radiance model obtained by data-fitting,(b)-(d) the R,G,B channels of the sky model in (a). All color channels are generated using the same sky luminance model, but each color channel has its own distinct parameters.

to convert the photographs into radiance images by inverting the nonlinear mapping between the incident radiance of the camera and its digital output. To recover this nonlinear mapping, we use the technique described in [11].

5.3.1 The Sun

We can measure the radiance of the sun with a camera and a couple of neutral density filters(Figure 5.2(a)) to make it unsaturated so that we can recover its dynamic radiance using the nonlinear mapping introduced before. The solid angle subtended by the sun can be obtained from the diameter of the sun and the distance between the sun and the earth. The solar position(altitude and azimuth) can be obtained from formula given in the appendix of [101], provided that the latitude and longitude of the site on the earth's surface, and the time and date are known. We model the sun as a parallel light source.

5.3.2 The Sky

We can take photographs of the sky in order to measure its radiance distribution. But there exist a couple of problems. First, it is hard to know the camera pose because there is no feature in the sky to calibrate camera orientation if the sky is clear; second, it is hard to get a picture of the whole sky even with a fish-eye lens because there might be some objects occluding part of the sky, such as trees, buildings, and mountains; third, the intensity of circumsolar region or solar aureole can be very high, and can easily get saturated at a normal shutter speed. To solve the first problem, we decided to include some buildings as landmarks in each photograph so that we can use them to recover the camera pose later. But this means we are going to have more occlusions. While we will take multiple photographs(Figure 5.2(c)(d)) and hope the invisible part of the sky in one photograph will become visible in some other photograph, there is no way to guarantee that every part of the sky will be seen. Our approach to solve this difficulty is to have a sky model which we can fit to the visible parts of the sky and extrapolate into the invisible parts. To solve the last problem, we use a set of different shutter speeds for the solar aureole with each speed capturing the radiance inside a circular band centered at the solar position(Figure 5.2(b)).

Several papers present physical models of sky radiance [85, 86, 94]. However, we do not know how closely they approximate the real sky. Furthermore, physical models often give the spectral distribution of any point in the sky. It is very hard to fit these models to RGB data taken from photographs.

On the other hand, there are also many empirical models for sky luminance or radiance distribution [90, 92, 93, 91]. All CIE standard sky formulae are fixed sky luminance

distributions. They can not be used for the purpose of data-fitting. The all-weather sky luminance model proposed in [90] is a generalization of the CIE standard clear sky formula. It is given by

$$Ls(\xi, \gamma) = Lvz \ f(\xi, \gamma)/f(0, Z) \tag{5.7}$$

where ξ is the zenith angle of the considered sky element and γ is the angle between this sky element and the position of the sun, Lvz is zenith luminance, Z is the zenith angle of the sun, and

$$f(\xi, \gamma) = [1 + a \exp(b/\cos\xi)][1 + c \exp(d\gamma) + e \cos^2\gamma] \tag{5.8}$$

where $a, b, c, d,$ and e are adjustable coefficients. These variable coefficients make this empirical model more flexible than others, which means we might have a better fit by using this model. Since both Lvz and $f(0, Z)$ in the above model are unknown constants, we replace them with one new variable coefficient which can be optimized during data fitting. Empirically, we also find it is better to have one more variable coefficient as the exponent of γ in the term with c and d. Thus, we obtain the following revised seven-parameter sky model

$$Ls(\xi, \gamma) = Lz[1 + a \exp(b/\cos\xi)][1 + c \exp(d\gamma^h) + e \cos^2\gamma] \tag{5.9}$$

where $a, b, c, d, e, h,$ and Lz are variable coefficients.

Up to now, we still only have a sky luminance model which does not have colors. We have not seen in the literature any approach converting sky luminance models to RGB color

distributions. The method proposed in [94] converts luminance data to color temperatures and then to spectral distributions. The relationship they use between luminance and color temperatures is not necessarily accurate for different weather conditions. Based on the fact that the sky radiance distribution at each color channel has a similar shape, we decided to use the same model but a distinct set of coefficients for each color channel by fitting the above revised model to the data from each channel. In practice, the error of data-fitting remains very small for each channel, which means our method is appropriate. Skies thus obtained have convincing colors.

Since there might be trees, buildings or mountains in photographs, we interactively pick some sky regions from each photograph and fit the revised sky model to the chosen sky radiance data by using Levenberg-Marquardt method [99] to minimize the weighted least-square

$$\sum_{i=1}^{N}[\frac{y_i - Ls(\xi_i, \gamma_i)}{\sigma_i}]^2 \tag{5.10}$$

where y_i's are the chosen sky radiance data from photographs and σ_i's are weights. We tried different weighting schemes, such as $\sigma_i = 1,\ y_i/\log(y_i),\ y_i/\text{sqrt}(y_i),$ or y_i, and found the best result was obtained when $\sigma_i = y_i/\log(y_i)$. With this weighting scheme, the fitting error at most places is within 5%. A recovered sky radiance model is given in Figure 5.3.

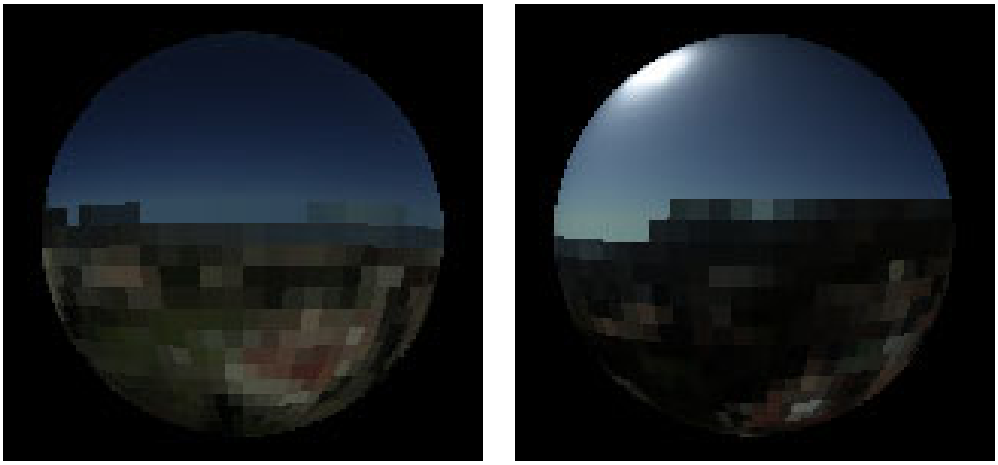


Figure 5.4: Two views of a spherical environment map. The upper hemisphere corresponds to the sky and the lower hemisphere has the radiance values from the surrounding landscape.

5.3.3 The Environment

By our definition, the environment of an outdoor object is its surrounding landscape. It can be a more significant light source than the dark side of a clear sky. There are mutual interreflections between an object and its environment. For reflectance recovery, we need to measure the radiance distribution of the whole environment which includes radiance from all visible objects and is the equilibrium state of mutual interreflections. It is assumed that we do not interfere with this equilibrium state when we take photographs of the environment.

For our purpose, we only need a coarse-grain environment radiance map to do irradiance calculation because irradiance results from an integrated effect of the incident radiance distribution. High-frequency components can therefore be ignored. We subdivide the environment sphere along latitudinal and longitudinal directions and get a set of rectangular spherical regions. Once we have those environment photographs(Figure 5.2(d)) and their

camera orientations, we project every pixel into one of the spherical regions. Finally, we average the color of the pixels projected into each region and give the result as the average radiance from that region. If the architecture has large size, we may need to capture more than one environment map at different locations because the surrounding light field is a four dimensional distribution. However, since the integrated irradiance over the surfaces changes smoothly and slowly, we do not need to capture more than a small number of coarse-grain environment maps.

Two images of a spherical environment map including radiance distribution from both the sky and the landscape are shown in Figure 5.4.

5.4 Recovering Reflectance

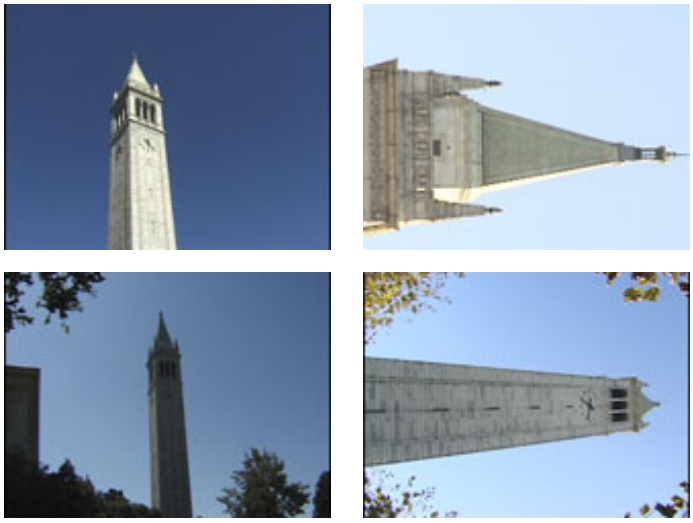
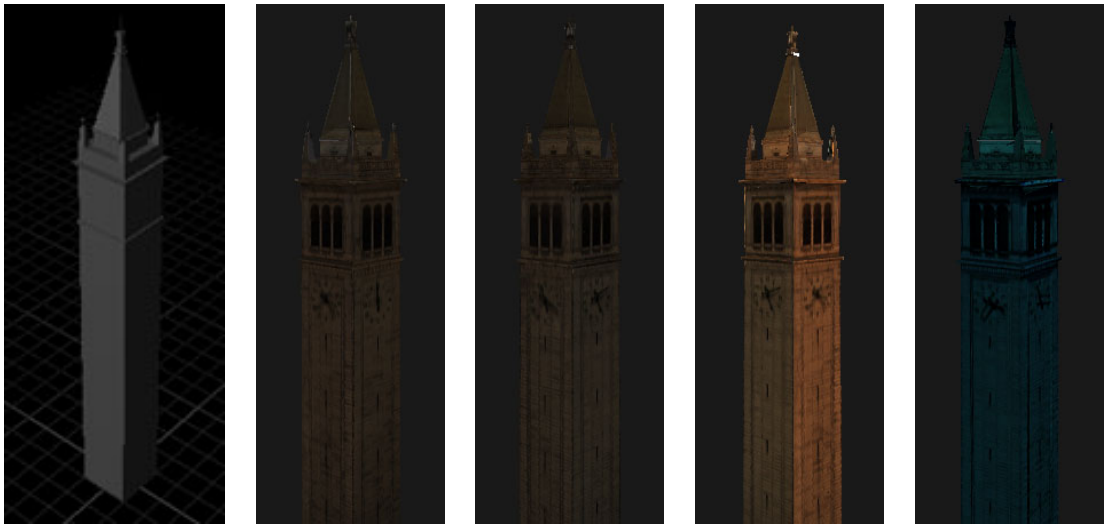


Figure 5.5: Some photographs of a bell tower for reflectance recovery

Recall from Section 5.2, that we decided to recover two sets of pseudo-BRDFs: one corresponding to the spectral distribution of the sun, and the other corresponding to the spectral distribution of the irradiance from both the sky and environment.



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

Figure 5.6: (a) A simple geometric model of a bell tower, (b)-(c) Diffuse pseudo-albedo recovered by using irradiance from both the sky and the landscape, (d) diffuse pseudo-albedo recovered incorrectly by only using irradiance from the sky, (e) diffuse pseudo-albedo corresponding to the spectral distribution of the sun.

Since the outside surfaces of most buildings are close to Lambertian, we spent most efforts on recovering pseudo-albedos. The recovery of diffuse pseudo-albedos at each surface point needs the incident irradiance and the outgoing diffuse radiance. The incident irradiance is obtained by gathering light from the sun, the sky, the environment, and possibly other polygonal faces occluding part of the previous three sources. We can get the irradiance from the sun by using the surface normal, the color and solid angle of the sun which we got from Section 5.3.1. The approach to gather light from the sky and environment is discussed in Appendix B. Gathering light from occluding faces can be done using the method in [83]. We use one-bounce reflection to approximate the interreflection among different faces.

Multiple photographs are taken for the considered building at different viewing direc-

tions and times(Figure 5.5). Since most architectural materials are only weakly specular except for windows, if our viewing direction is far away from the mirror angle of the current solar position, we can assume only diffuse radiance is captured in the photograph. Since each photograph can only cover some part of the architecture and there are occlusions among different faces, we need to decide which face is visible to which photographs. Visibility testing and polygon clipping was discussed in Section 3.5.

Since every surface of the building has its own surface texture, we need to incorporate these spatial variations into its pseudo-albedos. Each polygon in the geometric model is first triangulated and a dense grid is set up on each triangle in order to capture the variations. This step is similar to that introduced in [44]. Each grid point is projected onto the photographs to which it is visible and a radiance value is taken from each photograph. The diffuse pseudo-albedo at the grid point is obtained by dividing the average radiance by the irradiance.

We need at least two photographs for each face of the building to recover both sets of pseudo-BRDF's. And it should not be lit by the sun in one photograph and should be lit in the other. Thus we have two equations for each surface point, one from each photograph.

$$\pi I^{(1)} = \rho^{se} E_{se}^{(1)} \tag{5.11}$$

$$\pi I^{(2)} = \rho^{se} E_{se}^{(2)} + \rho^{sun} E_{sun} \tag{5.12}$$

where $I^{(1)}$ and $I^{(2)}$ are radiance values obtained from the two photographs, $E_{se}^{(1)}$ and $E_{se}^{(2)}$ are the irradiance from the sky and environment, E_{sun} is the irradiance from the sun, ρ^{se} is the pseudo-albedo corresponding to the spectral distribution of the sky and environment,

and ρ^{sun} is the pseudo-albedo corresponding to the spectral distribution of the sun.

From (5.11), we can solve ρ^{se}. By substituting it into (5.12), we can solve ρ^{sun} too. Of course, if we have more than two photographs, these estimations can be made more robust. Figure 5.6 displays the recovered diffuse pseudo-albedo. Figure 5.6(b)&(c) shows the diffuse pseudo-albedos of four different sides of a bell tower. These recovered pseudo-albedos are quite consistent with each other, providing an independent verification of our procedure since we recovered them from different photographs shot at different times.

We can choose solar positions to avoid large shadows cast on the architecture. When large shadows are inavoidable, we can interactively label the shadow boundaries to separate sunlit regions from shadowed ones. If there are several buildings located close to each other such that some sides of the buildings can not be lit by the sun or we can not simply take photographs for them, our method can not be applied. A solution to this difficulty might be to fill in reflectance values from adjacent faces.

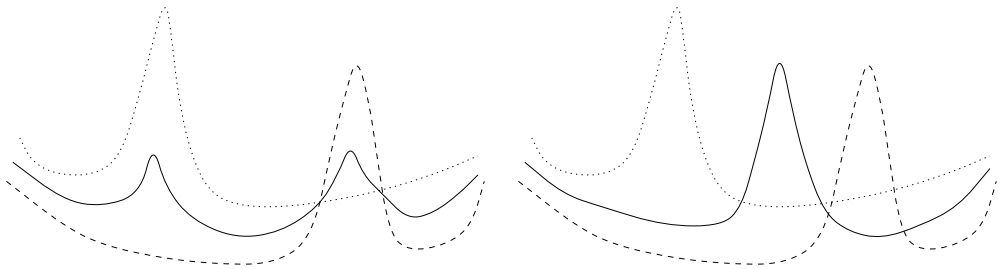
5.5 Modeling Illumination at Novel Times of Day

To generate renderings of the scene at a novel time of day, we need to predict what the illumination will be at that time. This requires us to construct sun, sky and environment illumination models appropriate to that time. We have available as a starting point, the illumination models for a few times of day where we took the initial photographs, recovered using the techniques introduced in Section 5.3.

5.5.1 The Sun And Sky

Given the local time of day, the solar position(altitude and azimuth) can be obtained directly from formula given in the appendix of [101], provided that the latitude and longitude of the site on the earth's surface and the day number in a year are all known.

Finding the appropriate sky model requires more work. First we consider sky interpolation during the main part of the day, ignoring sunrise and sunset. Note that the sky radiance distribution changes with the solar position, and naive pointwise radiance interpolation at each point in the sky would not work as shown in Figure 5.7.



(a) (b)

Figure 5.7: 1D Schematic of sky interpolation where peaks represent sky radiance at solar aureole. (a) A new sky(solid) obtained by pointwise interpolating two sky models(dot). It is not correct because it has two peaks. (b) A new sky(solid) obtained with our interpolating scheme.

Instead, let's examine the sky model in (5.9). It has three parts. The first part is the scaling factor Lz which controls the overall brightness of the sky. If not during sunrise or sunset, it should be almost a constant. The second part is the sky background. We denote it by $Bg(\xi)$. The third part is the solar aureole. We denote it by $Sa(\gamma)$. The shapes of $Bg(\xi)$ and $Sa(\gamma)$ remain unchanged during most times of a day. What changes is their relative position. $Sa(\gamma)$ rotates relative to $Bg(\xi)$ as the sun moves across the sky. Based

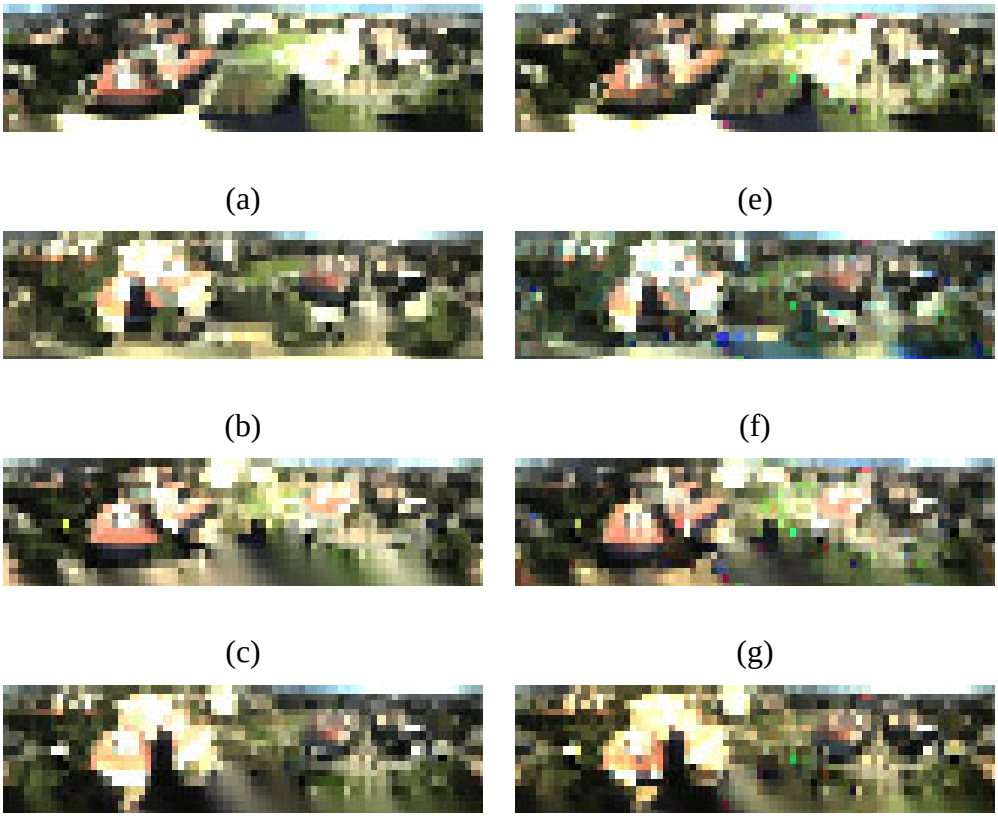


Figure 5.8: (a)-(d) Environment maps for four different times, obtained from multiple photographs, (e)-(h) corresponding environment maps generated with the recovered environment radiance models which were obtained by data-fitting. There is one recovered radiance model for each environment region.

on this observation, we derive a sky interpolation scheme. Suppose we have recovered k sky models. If we need a new sky model at a different time, a grid is first set up on the sky hemisphere. At each grid point with parameters (ξ_i, γ_i) corresponding to the new solar position, we can get three data sets $\{Lz^j, Bg^j(\xi_i), Sa^j(\gamma_i); j = 1, \cdots, k\}$ from the existing models. Set the sky radiance at the grid point to be the product of three weighted averages of the three data sets. The weight for each existing sky model is proportional to the reciprocal of the angular distance between the new solar position and the solar position of that sky

model. Finally, with the radiance values at the grid points, we can run an optimization to fit a new sky model.

During sunrise or sunset, there is less light from short wavelengths. So the sun and solar aureole appear more red. The whole sky is darker. But the color of the rest of the sky only changes a little. It is well known, e.g. [86], that the color of the sky and sun is caused by scattering in the atmosphere. If a light beam travels a distance d in a medium with scattering particles, its intensity will be decreased by a factor of $\exp(-\beta d)$ where β is a constant coefficient. With different β's for different wavelengths, the color of the beam will also change. The distance d that the sunlight travels through the atmosphere is the smallest when solar direction is perpendicular to the ground and it increases when the sun moves closer to the horizon. The optical depth of the atmosphere at the horizon is about 38 times that at the zenith. A formula to compute d for any solar position can be found in [86] which tries to get the color of the sun and sky from physics-based models. However, we want to fit the above scattering model to real measurements. We measured solar radiance during the day and sunset and fit a distinct β for each color channel. We use the same coefficients to get the color of solar aureole. For the sky background, we use an average β for all three color channels to decrease the brightness but keep the color unchanged.

5.5.2 Environment Radiance Model

Predicting radiance models for the sun and sky is not enough, because the building also receives light reflected from other surfaces in the environment. Predicting the environment radiance map at a novel time is a challenging problem, and it may appear that the only

solution would be to completely geometrically model the rest of the environment and then solve a global illumination problem to render the entire scene. However we have found an acceptable approximation for our purposes by a much simpler technique.

The idea is to recover not the detailed geometric structure of the environment, but rather a very crude, low frequency model adequate enough for our purpose – obtaining an approximation to the illumination resulting from it on the primary architectural piece of interest.

We use the technique of photometric stereo for shape-from-shading in computer vision [98] to recover the average reflectance, assumed lambertian, and surface normal for each region of the environment. One can solve for the albedo and normal orientation at each pixel location in an overdetermined system by taking multiple images of the same object with the same camera position but different positions of the single light source. In our context, the different positions of the light source are generated by the movement of the sun during the day. Interreflections within the environment are neglected. The Lambertian model appears reasonable because most surfaces in an outdoor scene are pretty diffuse. The big change is that we do not have a single light source, but must consider both the sky and the sun as light sources. Considering the sky as an ambient light source and the sun as a directional light source, we have the following formulation

$$I_{env} = \begin{cases} \rho^{sky} E_{sky} + \rho^{sun} E_{sun} (n_{env} \cdot l_{sun}) \\ \qquad\qquad\qquad\qquad , \text{ if } n_{env} \cdot l_{sun} \geq 0, \\ \rho^{sky} E_{sky} \qquad\qquad\qquad\qquad , \text{otherwise.} \end{cases} \tag{5.13}$$

where ρ^{sky} is the pseudo-albedo corresponding to the spectral distribution of the sky, E_{sky} is the magnitude of the total flux from the sky because we consider the sky as an ambient

source, ρ^{sun} is the pseudo-albedo corresponding to the spectral distribution of the sun, E_{sun} is the irradiance from the sun, n_{env} is the normal of the considered region and l_{sun} is the solar position. The reason why we allow ρ^{sky} and ρ^{sun} to be independent because they are related to pseudo-BRDF's corresponding to the spectral distributions of the sky and the sun and we expect them to be very different.

Since we have three color channels, both ρ^{sky} and ρ^{sun} have three components, and n_{env} has two degrees of freedom because it has unit length. There are eight unknowns for each environment region. If we photograph the environment for at least three solar positions and get the corresponding sky flux values, we would have at least nine equations at each environment region and the unknowns can be estimated by weighted least-square method. Note that the trajectory of the sun seen from the surface of the earth is not a planar curve, otherwise the three solar positions would not give us independent information. The estimated pseudo-albedos and normal can then be used to predict radiance under new lighting conditions.

How can we impose the constraint that n_{env} has unit length ? We could just add a penalty term in (5.13) to do this. However we find, among the six variables in ρ^{sun} and n_{env}, there are only five degrees of freedom. We can just set one of them to be a constant to impose the constraint more strictly. Any component of n_{env} can be either zero or nonzero. It is more appropriate to set one component of ρ^{sun} to be a positive constant, say 0.1. Now n_{env} does not necessarily have unit length. (5.13) should be rewritten in the following way

$$I_{env} = \begin{cases} \rho^{sky} E_{sky} + (\rho^{sun} \|n_{env}\|) E_{sun}(\frac{n_{env}}{\|n_{env}\|} \cdot l_{sun}) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{, if } n_{env} \cdot l_{sun} \geq 0, \\ \rho^{sky} E_{sky} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{, otherwise.} \end{cases} \tag{5.14}$$

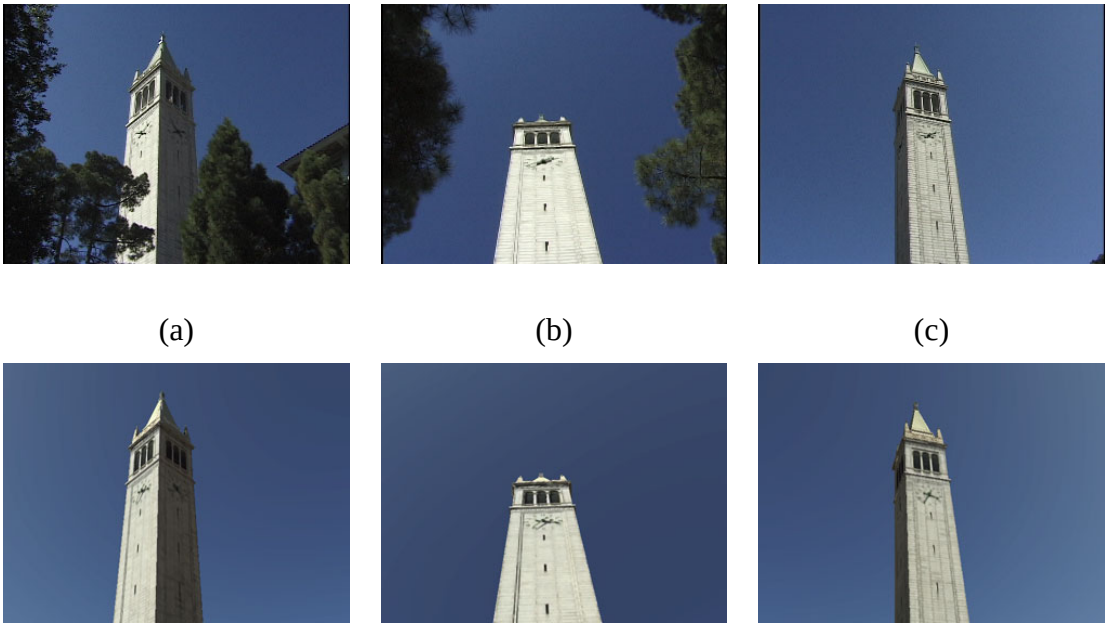
We use the Levenberg-Marquardt method to solve this nonlinear least-squares problem. However this technique requires the objective function to have a derivative everywhere while our formulation above does not have one when $n_{env} \cdot l_{sun} = 0$. One way to get around this is to reformulate (5.14) as follows

$$I_{env} = \begin{cases} \rho^{sky} E_{sky} + (\rho^{sun} \|n_{env}\|) E_{sun}(\frac{n_{env}}{\|n_{env}\|} \cdot l_{sun}) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{, if } n_{env} \cdot l_{sun} \geq 0, \\ \rho^{sky} E_{sky} + (\rho^{sun} \|n_{env}\|) E_{sun} \cdot \\ \qquad \{\frac{1}{\alpha}[\exp(\frac{\alpha}{\|n_{env}\|} n_{env} \cdot l_{sun}) - 1]\} \qquad \text{, otherwise.} \end{cases} \tag{5.15}$$

where α can be any large positive constant, say 1000.

We can check that (5.15) has derivative everywhere and its second term keeps very close to zero when $n_{env} \cdot l_{sun} < 0$, which is a good approximation to (5.14). Levenberg-Marquardt method can be easily used to minimize the least-square error criterion for (5.15). The start point of ρ^{sky} is set to the ratio between the average radiance and the average magnitude of incident flux from the sky, and the start point of ρ^{sun} is set to the ratio between the average radiance and the average irradiance from the sun. We may obtain meaningless values for the normal if some region is never lit by the sun. To alleviate this problem, during the optimization, if the data fitting error at some region is larger than a threshold and the obtained normal is pointing away from the building, we remove the solar term in the above model and only try to get an estimation for ρ^{sky}. Adding a smoothing term between

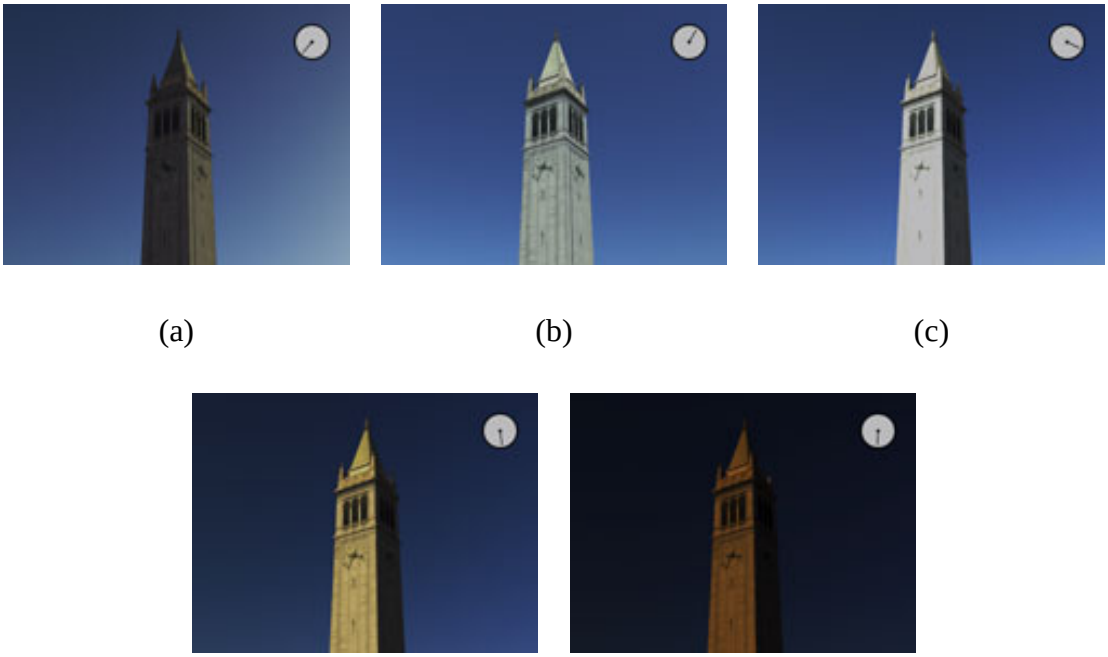
adjacent regions may also help. Some recovered environment radiance maps with the above modeling method are given in Figure 5.8(e)-(h). For every region of the environment, we have eight unknowns in the model and twelve equations obtained from four different times of day. Since the system is overdetermined, the good fit in Figure 5.8 provides justification for our simplifying assumptions that the environment is Lambertian and that interreflections within the environment can be neglected.



(a) (b) (c)

(d) (e) (f)

Figure 5.9: (a)-(c) Three real photographs of a bell tower taken with shutter duration 1/1500 a second, (d)-(f) three corresponding synthetic images for the same time and shutter speed. They look similar although the real photographs in (a)-(c) are not used for training and generating the synthetic images.



(a) (b) (c)

(d) (e)

Figure 5.10: Five synthetic images of a bell tower under a clear sky with shutter duration 1/1500 a second. They represent the appearances of the bell tower at different times(solar positions) on a sunny day close to the end of August at a location with latitude 37.8 and longitude -122.3. (a) 7am, (b) 1pm, (c) 4pm, (d) 6pm, (e) 6:30pm.

5.6 Results

We chose the Berkeley bell tower(Campanile) as our target architecture and took a total of about 100 photographs for the tower, the sky and the landscape at four different times. These photographs are used as source in data-fitting. They can be considered as training data. From the various measurements and recovered models, we found the relative importance of each illumination component and reflectance component in our example. On shaded sides of the tower, the irradiance from both the sky and landscape has the same order of magnitude, but the irradiance from the landscape is larger. On sunlit sides, the sun dominates the illumination if its incident angle is not too large. The percentage varies with

(a) (b)

(c) (d)

Figure 5.11: Four synthetic images of a bell tower with shutter duration 1/1500 a second under an overcast sky with different percentages of blocked sunlight(PBS). (a) PBS=0.0, (b) PBS=0.5, (c) PBS=0.9, (d) PBS=0.95.

different color channels. If the incident angle is less than 60 degrees, the light from the sun may exceed 90% in the red channel, and 60% in the blue channel.

We also took photographs at a fifth time. Those photographs are used for comparison with re-rendered images. They can be considered as testing data.

Relative positions and orientations of the cameras are currently calibrated by using the FACADE system in [12]. Alternatively, we could use any standard mosaicing technique for the environment photographs. Exterior orientation is calibrated with a compass map or the solar position.

Photometric calibration of the camera is done using the technique in [11]. Once we have recovered the nonlinear mapping between incident radiance and camera output, we can use it to further recover the radiance at each pixel by taking photographs at different shutter speeds and combining them. All subsequent processing in the system uses radiance values. At the end, re-rendered radiance images are converted back to normal images using the nonlinear response curve of the sensor.

5.6.1 Comparison With Ground Truth

Our approach makes a number of simplifying assumptions and approximations. It is therefore necessary to check the accuracy of our re-rendering by rendering the bell tower at the fifth time and comparing the synthetic images with real photographs shot at the same time. Three pairs of images from three different viewpoints are shown in Figure 5.9. The sky in the synthetic images are obtained by clear sky interpolation introduced in Section 5.5.1.

5.6.2 Sunrise To Sunset Simulation

A sequence of images are shown in Figure 5.10. It includes images at sunrise and sunset simulated with the technique in Section 5.5.1. Images rendered for sunrise and sunset can only be considered as approximations to real photographs because the solar spectrum changes at these periods, but we still use previously recovered pseudo-BRDF's. However, these approximations look realistic.

5.6.3 Intermediate And Overcast Sky Simulation

By intermediate and overcast skies, we mean there is a uniform layer of clouds covering the sky which blocks some or all of the sunlight. To simulate this kind of sky, we can either get a overcast sky model by data fitting or use CIE standard overcast sky luminance model along with a user-specified color for the clouds which is usually close to gray. A coefficient specifying the percentage of the sunlight blocked by the clouds should also be given. Then the color at a point in the sky is simply a linear interpolation between the color of a clear sky and the color of the overcast sky. Actually some sky luminance models reviewed in [93] really use this kind of interpolation between two extreme sky models.

A sequence of images are shown in Figure 5.11. It gives re-rendering results with various sky interpolation coefficients.

5.6.4 High Resolution Re-Rendering

Since we used a fixed size grid on each triangular patch to capture the spatial variation of surface reflectance, as the viewpoint moves sufficiently close to the surface of the object, each grid cell will correspond to multiple image pixels. The resulting rendering then takes on a somewhat blurred appearance, as variation in surface texture at a resolution finer than the grid size is lost. In this section we show results from a simple technique by which the resolution can be boosted to the pixel resolution. The basic idea is to use a high resolution zoom photograph of the architecture available as a texture map in the "right" way. Since the lighting conditions can be different, we need pixel wise reflectance values. Let $I(x, y)$ be the radiance measured from the high-resolution photograph at pixel (x, y) and $\rho(x, y)$, and

$E(x, y)$ be the corresponding high-resolution pseudo-albedo and irradiance at the surface point corresponding to pixel (x, y). $\tilde{I}(x, y)$, $\tilde{\rho}(x, y)$ and $\tilde{E}(x, y)$ are the corresponding low-resolution versions. Both $\rho(x, y)$ and $E(x, y)$ are unknown, but we can exploit the fact that the spatial variation in lighting $E(x, y)$ has only low frequency components, and therefore is quite well approximated by $\tilde{E}(x, y)$. We can obtain $\tilde{\rho}(x, y)$ from previously recovered pseudo-albedo at surface grid points, and $\tilde{I}(x, y)$ by smoothing $I(x, y)$; then $\tilde{E}(x, y) = \frac{\tilde{I}(x,y)}{\tilde{\rho}(x,y)}$ and the high resolution pseudo-albedo $\rho(x, y)$ can be estimated by

$$\rho(x, y) = \frac{I(x, y)}{E(x, y)} \approx \frac{I(x, y)}{\tilde{E}(x, y)} \tag{5.16}$$

The recovered high-resolution pseudo-albedo $\rho(x, y)$ can be used for re-rendering under novel lighting conditions. In Figure 5.12, we give a resulting image from this kind of re-rendering. A low-resolution image from previously recovered reflectance is also given for comparison.

(a)

(b)

(c)

Figure 5.12: (a) A re-rendered zoom-in image with shutter duration 1/500 a second with the sun behind the bell tower using the previously recovered surface pseudo-BRDF's, (b) a reference photograph at the same viewpoint, but with a different solar position, (c) a synthetic image with the same illumination and shutter speed as in (a), but with higher resolution, rendered using the view-dependent re-rendering technique. It uses both the reference photograph and the previously recovered low-resolution surface pseudo-BRDF's.

Appendix A

BRDF Model and Parameter Recovery

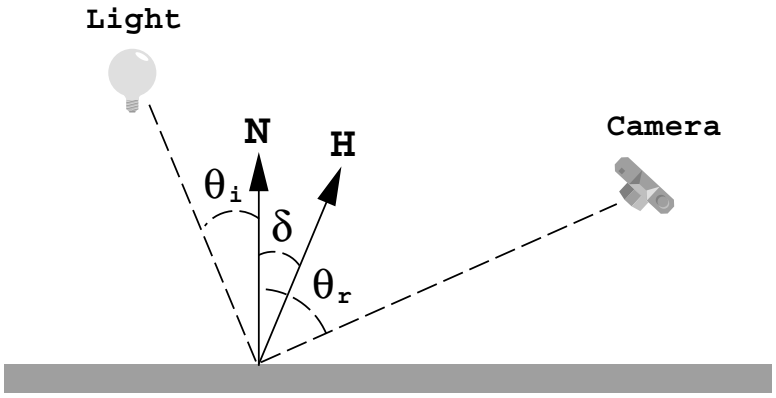


Figure A.1: Definitions of the angles used in Ward's BRDF model. θ_i is the incident angle, θ_r is the viewing angle, and δ is the angle between the surface normal N and the halfway vector H between the lighting and viewing directions.

In this section we present more details on the BRDF model, introduced in Section 4.4, and how its parameters are recovered. We use Ward's [80] model for the specular term in the BRDF, which could be modeled as either isotropic or anisotropic. In the isotropic case,

$$K(\alpha, \Theta) = \frac{1}{\sqrt{\cos\theta_i \cos\theta_r}} \frac{\exp[-\tan^2\delta/\alpha^2]}{4\pi\alpha^2} \tag{A.1}$$

where α is a scalar surface roughness parameter, θ_i is the incident angle, θ_r is the viewing

angle, and δ is the angle between the surface normal and the halfway vector H between the lighting and viewing directions (Fig. A.1). θ_i, θ_r are two components (along with ϕ_i, ϕ_r) of the vector $\mathbf{\Theta}$ which represents the incidence and viewing directions.

In the anisotropic case, we need two distinct roughness parameters α_x, α_y for two principal axes on the surface and an azimuth angle γ to define the orientation of these principal axes on the surface relative to a canonical coordinate system. Then, the parameter vector $\boldsymbol{\alpha}$ actually has three components $(\alpha_x, \alpha_y, \gamma)$ and we have:

$$K(\boldsymbol{\alpha}, \mathbf{\Theta}) = \frac{1}{\sqrt{\cos\theta_i \cos\theta_r}} \frac{\exp[-\tan^2\delta(\cos^2\phi/{\alpha_x}^2 + \sin^2\phi/{\alpha_y}^2)]}{4\pi\alpha_x\alpha_y} \tag{A.2}$$

where δ is the same as in the isotropic case, and ϕ is the azimuth angle of the halfway vector H projected into the local 2D coordinate system on the surface patch defined by the two principal axes. To compute ϕ, γ, which relates this coordinate system to the canonical coordinate system, is necessary.

Now to parameter recovery. We wish to find ρ_d, ρ_s and $\boldsymbol{\alpha}$ that minimize the squared error between the measured and predicted radiance,

$$e(\rho_d, \rho_s, \boldsymbol{\alpha}) = \sum_{i=1}^{m} (L_i - \frac{\rho_d}{\pi} I_i - \rho_s K(\boldsymbol{\alpha}, \mathbf{\Theta}_i) I_i)^2 \tag{A.3}$$

where L_i is the measured radiance and I_i is the irradiance (computable from the known light source position) at sample point p_i on the surface, and m is the number of sample points.

Note that given a guess of $\boldsymbol{\alpha}$, $K(\boldsymbol{\alpha}, \mathbf{\Theta}_i)$ becomes a known quantity, and minimizing the error e reduces to a standard linear least-squares problem for estimating ρ_d and ρ_s. Plugging in these values in the right hand side of Eqn. (A.3) lets us compute e as a function of $\boldsymbol{\alpha}$. The optimization problem thus simplifies to a search for the optimum value of $\boldsymbol{\alpha}$ to

minimize $e(\boldsymbol{\alpha})$. This is either a one-dimensional or three-dimensional search depending on whether an isotropic or anisotropic model of the specular term is being used. We use golden section search [99] for the isotropic case, and the downhill simplex method [99] in the anisotropic case. It is convenient that neither method requires evaluating the derivative $e'(\boldsymbol{\alpha})$, and both methods are fairly robust.

To deal with colored materials, we estimate both diffuse and specular reflectance in each of the red, green, blue color channels. The specular roughness parameters $\boldsymbol{\alpha}$ are the same for all color channels. The nonlinear optimization is still over 1 or 3 parameters, since given $\boldsymbol{\alpha}$, ρ_d and ρ_s estimation for each channel remains a linear least squares problem.

To make the parameter estimation additionally robust, we make two simple extensions to the basic strategy derived above. The first is to solve a weighted least squares problem instead of the vanilla version in Eqn. (A.3). Radiance measurements from the highlight area have much larger magnitude than those from the non-highlight area. Correspondingly the error in those measurements is higher both because of noise in imaging as well as error in the BRDF model. Giving all the terms in (A.3) equal weight causes biased fitting and gives poor estimation of the diffuse reflectance. From a statistical point of view, the correct thing to do is to weight each term by the reciprocal of the variance of expected error in that measurement. Not having a good model for the error term, we chose a heuristic strategy in which the weight w_i for the i-th term in the summation in Eqn. (A.3) is set to $\frac{1}{K(\boldsymbol{\alpha_c},\boldsymbol{\Theta}_i)}$ where $\boldsymbol{\alpha_c}$ is some *ad hoc* or iteratively improved roughness vector. Since the roughness of most isotropic materials is less than 0.2, we used an initial value between 0.1 and 0.2 for scalar α_c.

The second refinement to improve parameter recovery is to use specular color infor-

mation. For instance, specular highlights on dielectric and plastic materials have the same color as the light source, while the color of specular highlights on metals is the same as their diffuse components, which is the color of the light modulated by the diffuse albedo. For plastic objects, there would be one distinct variable ρ_d for each color channel, but the same variable ρ_s for all color channels. For metallic objects, there would be one variable ρ_d for each channel and a common ratio between the specular and diffuse reflectance in all channels. Thus, we can reduce the degree of freedom from $2N$ to N+1 where N is the number of color channels. For plastic, we can still obtain both analytic and numerical linear least-squares solutions for the N+1 variables provided the other parameters are fixed. The program performs a heuristic test to determine whether a material should be estimated with the metal or plastic specular reflectance model. Our program first solves for the specular reflectance of each color channel separately and then checks to see if they are larger than the estimated diffuse components. If they are larger, then the material is considered metallic. Otherwise, the plastic model is used. Then the smaller number of parameters corresponding to these material types are solved.

Our approach can be easily extended to accommodate data from multiple light-camera configurations. The cost function for multiple configurations is the summation of the cost function for each individual configuration as shown in Eqn. (A.3). All the above techniques can be applied on this new cost function as well.

Appendix B

Irradiance Calculation

We designed an efficient algorithm for gathering light from the sky based on adaptive subdivision. Since irradiance is an integration of the incident radiance, it varies slowly over the surface of the architecture. Thus we assume the irradiance over a triangular patch is a constant. For each triangle, we only gather the light at its centroid, and the centroid can always be handled as the effective center of the sky dome hemisphere because of the dome's very large radius. Each triangle defines a plane and only the part of the sky which is on the correct side of this plane, can be seen by the triangle. Further, there might be other faces in front of the triangle occluding part of the sky. So clipping the sky is necessary. The algorithm is summarized as follows

- Give each original polygon in the architecture model an id number; for each triangle, set its centroid as the viewpoint, Z-buffer the polygons with their id numbers as their color, scan the color buffer to retrieve the polygons in front of the current triangle.

- Discretize the sky hemisphere into a small set of large rectangular spherical polygons.

For each triangle, use its tangent plane and those occluding polygons to clip these spherical polygons. As a result, we get back a list of visible spherical polygons. Subdivide these spherical polygons until the sky radiance over each of them is almost uniform. The sky vector flux is the summation of the flux vectors of these subdivided sky patches. Finally, the irradiance from the sky is the inner product between the sky vector flux and the local surface normal.

The vector flux of a sky patch gives the direction and magnitude of the flux of that sky patch [95]. This algorithm is efficient because we only do visibility clipping on the initial small set of spherical polygons. This does not affect the accuracy because we do adaptive subdivision after the clipping.

The vector flux of a spherical triangle with uniform unit radiance can be obtained using a formula from [95]. We can assume the sky hemisphere has unit radius and its center is O because the irradiance from the sky is determined by its solid angle which is fixed no matter how large the radius is. Let A, B, C be three vertices on the sphere, L_{AB} be the length of the arc on the great circle passing through A and B, Π_{AB} be normalized $-(\vec{OA} \times \vec{OB})$. Then the vector flux of the spherical triangle ABC is

$$F(\triangle ABC) = \frac{L_{AB}}{2}\Pi_{AB} + \frac{L_{BC}}{2}\Pi_{BC} + \frac{L_{CA}}{2}\Pi_{CA}. \tag{B.1}$$

This formula can be easily generalized to compute the vector flux of any kind of spherical polygons.

Clipping a spherical polygon with a planar polygon can be done by connecting its vertices with straight line segments and treating it as a planar polygon. The only thing we

need to remedy after clipping is pushing back onto the sphere every new vertex generated by clipping.

We calculate the irradiance from the environment in the same way except that we do not subdivide each environment region adaptively. We only have a constant radiance value over each region and adaptive subdivision will not help improve the accuracy here.

Bibliography

[1] AMENTA, N., BERN, M., AND KAMVYSSELIS, M. A New Voronoi-Based Surface Reconstruction Algorithm. In *Proc. of SIGGRAPH 98*, pp.415-421.

[2] BESL, P.J., AND MCKAY, N.D. A method for registration of 3-d shapes. In *IEEE Trans. Patt. Anal. Machine Intell.*(1992), 18(5), pp. 239–256.

[3] BESL, P.J., AND JAIN, R.C. Segmentation Through Variable-Order Surface Fitting. In *IEEE Trans. Patt. Anal. Machine Intell.*(1988), 10(2), pp. 167–192.

[4] BOUGUET, J.-Y., AND PERONA, P. 3d photography on your desk. *ICCV* (1998).

[5] CHEN, E., AND WILLIAMS, L. View interpolation for image synthesis. In *SIGGRAPH '93* (1993), pp.279–288.

[6] CHEN, E. QuickTime VR - an image-based approach to virtual environment navigation. In *SIGGRAPH '95* (1995).

[7] Y.CHEN, AND MEDIONI, G. Object modeling from multiple range images. *Image and Vision Computing 10*, 3 (April 1992), pp.145–155.

[8] COHEN, J., OLANO, M., AND MANOCHA, D. Appearance-Preserving Simplification. In *SIGGRAPH'98*, (1998), pp. 115–122.

[9] CURLESS, B., AND LEVOY, M. A volumetric method for building complex models from range images. In *SIGGRAPH '96* (1996), pp. 303–312.

[10] DEBEVEC, P. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH 98* (July 1998).

[11] DEBEVEC, P. E., AND MALIK, J. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH 97* (August 1997), pp. 369–378.

[12] DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH '96* (August 1996), pp. 11–20.

[13] SEITZ, S. M., AND DYER, C. R. Photorealistic scene reconstruction by voxel coloring. In *CVPR* (1997), pp.1067-1073.

[14] EDELSBRUNNER, H., AND MÜCKE, D.P. Three-dimensional Alpha Shapes. In *ACM Transactions on Graphics*, 13, pp.43-72, 1994.

[15] FAUGERAS, O. Three-Dimensional Computer Vision. The MIT Press, Cambridge, Massachusetts, 1993.

[16] GARLAND, M, AND HECKBERT, P.S. Surface Simplification Using Quadric Error Metrics. In *SIGGRAPH '97* (1997), pp. 209–216.

[17] GEMAN, S., AND GEMAN, D. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. In *IEEE Patt. Anal. Machine Intell.* (1984), vol.6, no.6, pp. 30–50.

[18] GOLUB, AND VAN LOAN Matrix computations. John Hopkins Press, 1989.

[19] GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. The Lumigraph. In *SIGGRAPH '96* (1996), pp. 43–54.

[20] HARALICK, R.M., JOO, H., LEE, C.-N., ZHUANG, X., AND KIM, M.B. Pose estimation from corresponding point data. In *IEEE Trans. Systems, Man Cybernetics*(1989), 19(6):1426.

[21] HARALICK, R.M., LEE, C., OTTENBERG, K., AND NOLLE, M. Review and analysis of solutions of the three point perspective pose estimation. In *International Journal of Computer Vision*(1994), 13(3), pp.331-356.

[22] HOFFMAN, R.L., AND JAIN, A.K. Segmentation and Classification of Range Images. In *IEEE Patt. Anal. Machine Intell.* (1987), 9(5), pp. 608–620.

[23] HOPPE, H., DEROSE, T., DUCHAMP, T., MCDONALD, J., AND STUETZLE, W. Surface reconstruction from unorganized points. In *Proc. SIGGRAPH'92*, pp.71-78.

[24] HOPPE, H., DEROSE, T., DUCHAMP, T., AND MCDONALD, J., AND STUETZLE, W. Mesh Optimization. In *Proc. SIGGRAPH'93*, pp.19-26.

[25] HOPPE, H., DEROSE, T., DUCHAMP, T., JIN, H., MCDONALD, J., AND STUETZLE, W. Piecewise smooth surface reconstruction. In *Proc. SIGGRAPH'94*, pp.19-26.

[26] HOOVER, A., JEAN-BAPTISTE, G., JIANG, X.Y., FLYNN, P.J., BUNKE, H., GOLDGOF, D.B., BOWYER, K., EGGERT, D.W., FITZGIBBON, A., AND FISHER, R.B. An Experimental Comparison of Range Image Segmentation Algorithms. In *IEEE Patt. Anal. Machine Intell.* (1996), pp. 673–689.

[27] HUNG, Y., YEH, P.S., AND HARWOOD, D. Passive ranging to known planar point sets. In *IEEE Int. Conf. on Robotics and Automation*(1985), pp.80-85.

[28] HUTTENLOCHER, D.P. AND ULLMAN, S. Recognizing solid objects by alignment with an image. In *International Journal of Computer Vision*(1990), 5(2), pp.195-212.

[29] LAVALLE, S.M., AND HUTCHINSON, S.A. A Bayesian segmentation methodoogy for parametric image models. In *IEEE Patt. Anal. Machine Intell.* (1995), 17(2), pp. 211–217.

[30] LAVEAU, S., AND FAUGERAS, O. 3-D scene representation as a collection of images. In *Proceedings of 12th International Conference on Pattern Recognition* (1994), vol. 1, pp. 689–691.

[31] LEONARDIS, A., GUPTA, A. AND BAJCSY, R. Segmentation of Range Images as the Search for Geometric Parametric Models. In *Int'l J. Computer Vision* (1995), vol.14, no.3, pp. 253-277.

[32] LEVOY, M., AND HANRAHAN, P. Light field rendering. In *SIGGRAPH '96* (1996), pp. 31–42.

[33] LINDSTROM, P. AND TURK, G. Evaluation of Memoryless Simplification. In *IEEE Transactions on Visualization and Computer Graphics*, 5(2), 1999, pp. 98-115.

[34] MANGAN, A.P., AND WHITAKER, R.T. Partitioning 3D Surface Meshes Using Watershed Segmentation. In *IEEE Trans. on Visualization and Computer Graphics*, 5(4), 1999, pp.308-321.

[35] MALIK, J., BELONGIE, S., SHI, J., AND LEUNG, T. Textons, Contours and Regions: Cue Combination in Image Segmentation. In *International Conference on Computer Vision*, 1999.

[36] MCMILLAN, L., AND BISHOP, G. Plenoptic Modeling: An image-based rendering system. In *SIGGRAPH '95* (1995).

[37] NEUGEBAUER, P.J. AND KLEIN, K. Texturing 3D Models of Real World Objects from Multiple Unregistered Photographic Views. In *Eurographics'99* (1999), pp 245–256.

[38] NEWMAN, T.S., FLYNN, P.J., AND JAIN, A.K. Model-based classification of quadric surfaces. In *CVGIP:Image Understanding* (1993), 58(2), pp 235-249.

[39] NYLAND, L. ET AL The Impact of Dense Range Data on Computer Graphics. In *CVPR MVIEW'99*.

[40] PERONA, P., AND FREEMAN, W.T. A factorization approach to grouping. In *Proc. ECCV'98*, pp. 655–670.

[41] PULLI, K. Multiview Registration for Large Data Sets. In *International Conference on 3D Digital Imaging and Modeling* (1999), pp. 160–168.

[42] JI, Q., COSTA, M., HARALICK, R., AND SHAPIRO, L. An integrated linear technique for pose estimation from different features. In *International Journal of Pattern Recognition and Artificial Intelligence*(1999), June.

[43] SABATA, B., ARMAN, F., AND AGGARWAL, J.K. Segmentation of 3D range images using pyramidal data structures. In *CVGIP: Image Understanding* (1993), 57(3), pp. 373–387.

[44] SATO, Y., WHEELER, M. D., AND IKEUCHI, K. Object shape and reflectance modeling from observation. In *SIGGRAPH '97* (1997), pp. 379–387.

[45] SHADE, J., GORTLER, S., HE, L.-W., AND SZELISKI, R. Layered Depth Images. In *SIGGRAPH '98* (1998), pp. 231–242.

[46] SHI, J., AND MALIK, J. Motion Segmentation and Tracking Using Normalized Cuts. In *International Conference on Computer Vision* (1998).

[47] SHI, J., AND MALIK, J. Normalized Cuts and Image Segmentation. In *IEEE Conf. Computer Vision and Pattern Recognition* (1997).

[48] SHUM, H.-Y., AND HE, L.-W. Rendering with Concentric Mosaics. In *SIGGRAPH'99* (1999), pp. 299–306.

[49] SOUCY, M., GODIN, G., AND RIOUX, M. A texture-mapping approach for the compression of colored 3D triangulations. In *Visual Computer* (1996), vol.12, pp. 503–514.

[50] SZELISKI, R. Image mosaicing for tele-reality applications. In *IEEE Computer Graphics and Applications* (1996).

[51] SZELISKI, R., AND SHUM, H.-Y. Creating full view panoramic image mosaics and environment maps. In *SIGGRAPH 97* (1997), pp. 251–258.

[52] TAYLOR, R.W., SAVINI, M., AND REEVES, A.P. Fast segmentation of range imagery into planar regions. In *CVGIP* (1989), 45(1), pp. 42–60.

[53] TURK, G., AND LEVOY, M. Zippered polygon meshes from range images. In *SIGGRAPH '94* (1994), pp. 311–318.

[54] WHEELER, M.D., SATO, Y., AND IKEUCHI, K. Consensus surfaces for modeling 3D objects from multiple range images. In *DARPA Image Understanding Workshop* (1997).

[55] WUNSCH, P., AND HIRZINGER, G. Registration of CAD-Models to Images by Iterative Inverse Perspective Matching. In *International Conference on Pattern Recognition* (ICPR), 1996, pp.77-83.

[56] M. Segal, C. Korobkin, R. van Sidenfelt, J. Foran, and P. Haeberli, Fast Shadows and Lighting Effects Using Texture Mapping. *Computer Graphics*, 26(2), 249-252 (1992).

[57] P.S. Heckbert, Fundamentals of texture mapping and image warping. *Master's thesis, Computer Science Division, UC Berkeley* (1989).

[58] K. Weiler, and P. Atherton, Hidden Surface Removal Using Polygon Area Sorting. In *Proc. of SIGGRAPH 77*, 214-222 (1977).

[59] H. Weghorst, G. Hooper, and D.P. Greenberg, Improved Computational Methods for Ray Tracing. In *ACM TOG*, 3(1), 52-69 (1984).

[60] N. Greene, M. Kass, and G. Miller, Hierarchical Z-Buffer Visibility. In *Proc. of SIGGRAPH 93*, 231-238 (1993).

[61] S. Teller, and P. Hanrahan, Global Visibility Algorithms for Illumination Computations. In *Proc. of SIGGRAPH 93*, 239-246 (1993).

[62] G. Drettakis, and E. Fiume, A Fast Shadow Algorithm for Area Light Sources Using Backprojection. In *Proc. of SIGGRAPH 94*, 223-230 (1994).

[63] HANRAHAN, P., SALZMAN, P., AND AUPPERLE, L. A rapid hierarchical radiosity algorithm. In *SIGGRAPH 91* (1991), pp. 197–206.

[64] AUPPERLE, L., AND HANRAHAN, P. A hierarchical illumination algorithm for surfaces with glossy reflection. In *SIGGRAPH 93* (August 1993), pp. 155–164.

[65] BAUM, D. R., RUSHMEIER, H. E., AND WINGET, J. M. Improving radiosity solutions through the use of analytically determined form factors. In *SIGGRAPH 89* (1989), pp. 325–334.

[66] DANA, K. J., GINNEKEN, B., NAYAR, S. K., AND KOENDERINK, J. J. Reflectance and texture of real-world surfaces. In *Proc. IEEE Conf. on Comp. Vision and Patt. Recog.* (1997), pp. 151–157.

[67] DRETTAKIS, G., ROBERT, L., AND BOUGNOUX, S. Interactive common illumination for computer augmented reality. In *8th Eurographics workshop on Rendering, St. Etienne, France* (May 1997), J. Dorsey and P. Slusallek, Eds., pp. 45–57.

[68] GORAL, C. M., TORRANCE, K. E., GREENBERG, D. P., AND BATTAILE, B. Modeling the interaction of light between diffuse surfaces. In *SIGGRAPH '84* (1984), pp. 213–222.

[69] HE, X. D., TORRANCE, K. E., SILLION, F., AND GREENBERG, D. P. A comprehensive physical model for light reflection. In *SIGGRAPH 91*, (August 1991).

[70] KAJIYA, J. The rendering equation. In *SIGGRAPH '86* (1986), pp. 143–150.

[71] KARNER, K. F., MAYER, H., AND GERVAUTZ, M. An image based measurement system for anisotropic reflection. In *EUROGRAPHICS Annual Conference Proceedings* (1996).

[72] LOSCOS, C., FRASSON, M.-C., DRETTAKIS, G., WALTER, B., GRANIER, X., AND POULIN, P. Interactive Virtual Relighting and Remodeling of Real Scenes.

Technical Report, iMAGIS-GRAVIR/IMAG-INRIA, (May 1999), http://www-imagis.imag.fr/Membres/Celine.Loscos/relight.html.

[73] MARSHNER, S. *Inverse Rendering for Computer Graphics*. PhD thesis, Cornell University, August 1998.

[74] NAYAR, S. K., IKEUCHI, K., AND KANADE, T. Shape from interreflections. *International Journal of Computer Vision 6*, 3 (1991), 173–195.

[75] NIMEROFF, J., SIMONCELLI, E., AND DORSEY, J. Efficient re-rendering of naturally illuminated environments. In *5th Eurographics Workshop on Rendering* (1994).

[76] Oren, M., and Nayar,S.K., "Generalization of Lambert's Reflectance Model", *Computer Graphics Proceedings, Annual Conference Series*, pp.239-246 (1994).

[77] SILLION, F. X., AND PUECH, C. *Radiosity and Global Illumination*. Morgan Kaufmann Publishers, San Francisco, 1994.

[78] VEACH, E., AND GUIBAS, L. J. Metropolis light transport. In *SIGGRAPH 97* (August 1997), pp. 65–76.

[79] JENSEN, H.W., CHRISTENSEN, P.H. Efficient Simulation of Light Transport in Scenes With Participating Media Using Photon Maps. In *SIGGRAPH 98* (July 1998), pp. 311-320.

[80] WARD, G. J. Measuring and modeling anisotropic reflection. In *SIGGRAPH '92* (July 1992), pp. 265–272.

[81] WARD, G. J. The RADIANCE lighting simulation and rendering system. In *SIGGRAPH '94* (July 1994), pp. 459–472.

[82] WONG T.-T., HENG P.-A., OR S.-H. AND NG W.-Y. Image-based Rendering with Controllable Illumination. In *8th Eurographics Workshop on Rendering*, (June 1997), pp.13–22.

[83] Nishita,T., and Nakamae,E., “Continuous Tone Representation of Three-dimensional Objects Illuminated by Sky Light”, *Computer Graphics*, **20(4)**, pp.125-132 (1986).

[84] Nakamae,E., Harada,K., Ishizaki,T., and Nishita,T., “A Montage Method: the Overlaying of the Computer Generated Images onto a Background Photograph”, *Compter Graphics*, **20(4)**, pp.207-214 (1986).

[85] Tadamura,K., Nakamae,E., Kaneda,K., Baba,M., Yamashita,H., and Nishita,T., “Modeling of Skylight and Rendering of Outdoor Scenes”, *EUROGRAPHICS’93, in Computer Graphics Forum*, **12(3)**, pp.189-200 (1993).

[86] Klassen,R.V., “Modeling the Effect of the Atmosphere on Light” , *ACM Transactions on Graphics*, **6(3)**, pp.215-237 (1987).

[87] Nimeroff,J.S., Simoncelli,E., and Dorsey,J., “Efficient Re-rendering of Naturally Illuminated Environments”, *Eurographics Workshop on Rendering*, pp.373-388 (1994).

[88] Sato, Y., and Ikeuchi, K., “Reflectance Analysis for 3D Computer Graphics Model Generation”, *Graphical Models and Image Processing*, **58(5)**, pp.437-451 (1996).

[89] Lafortune, E.P.F., Foo, S., Torrance,K.E., and Greenberg, D.P., “Non-Linear Approximation of Reflectance Functions”, *Computer Graphics Proceedings, Annual Conference Series*, pp.117-126 (1997).

[90] Perez,R., Seals,R., and Michalsky,J., “All-weather Model for Sky Luminance Distribution–Preliminary Configuration and Validation”, *Solar Energy*, **50(3)**, pp.235-245 (1993).

[91] Ineichen,P., Molineaux,B., and Perez,R., “Sky Luminance Data Validation: Comparison of Seven Models with Four Data Banks”, *Solar Energy*, **52(4)**, pp.337-346 (1994).

[92] Brunger,A.P., and Hooper,F.C., “Anisotropic Sky Radiance Model Based on Narrow Field of View Measurements of Shortwave Radiance”, *Solar Energy*, **51(1)**, pp.53-64 (1993).

[93] Littlefair,P.J., “A Comparison of Sky Luminance Models with Measured Data from Garston, United Kingdom”, *Solar Energy*, **53(4)**, pp.315-322 (1994).

[94] Takagi,A., Takaoka,H., Oshima,T., and Ogata,Y., “Accurate Rendering Technique Based on Colorimetric Conception”, *Computer Graphics*, **24(4)**, pp.263-272 (1990).

[95] Koenderink,J.J., and van Doorn, A.J., “Illuminance texture due to surface mesostructure”, *J. Opt. Soc. Am.A*, **13(3)**, pp.452-463 (1996).

[96] Lee,H., Breneman,E.J., and Schulte,C.P., “Modeling Light Reflection for Computer Color Vision”, *IEEE Trans. PAMI*, **12(4)**, pp.402-409 (1990).

[97] Hall,R.A., and Greenberg,D.P., “A Testbed for Realistic Image Synthesis”, *IEEE CG&A*, **3(8)**, pp.10-20 (1983).

[98] Woodham, R., ”Photometric method for determining surface orientation from multiple images. In *Shape from Shading*, B. Horn and M. Brooks, Eds. MIT Press, 1989, pp.513-532.

[99] Press,W.H., Flannery,B.P., Teukolsky,S.A. and Vetterling,W.T., *Numerical Recipes in C*, Cambridge Univ. Press, New York, 1988.

[100] Hampel,F.R., Rousseeuw,P.J., Ronchetti,E.M., and Stahel,W.A., *Robust Statistics*, John Wiley & Sons, New York, 1986.

[101] Rees,W.G., *Physical Principles of Remote Sensing*, Cambridge Univ. Press, 1990.

[102] Horn,B.K.P., *Robot Vision*, MIT Press, Cambridge,Mass. 1986.

[103] MURRAY, R.M., LI, Z., AND SASTRY, S.S. A Mathematical Introduction to Robotic Manipulation CRC Press.

[104] GERSHO, A., AND GRAY, R.M. Vector quantization and signal compression Boston : Kluwer Academic Publishers, 1992.

[105] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics: principles and practice*, 2nd Edition, Addison-Wesley (1996).

[106] Yu, Y., Peng, Q., " Multiresolution B-spline Radiosity ", Proc. of *EUROGRAPHICS'95*, Maastricht, The Netherlands, September 1995. In Journal of *Computer Graphics Forum*, 14(3), pp.285-298, 1995.

[107] Yu, Y., Ibarra, O.H., and Yang, T., " Parallel Progressive Radiosity with Adaptive Meshing ", Journal of *Parallel and Distributed Computing*, 42(1), pp.30-41, April 1997. A short version appears in the Proceedings of IRREGULAR'96 (3rd international workshop), Santa Barbara, CA, August 1996, pp.159-170.

[108] Yu, Y., and Wu, H., " A Rendering Equation for Specular Transfers and Its Integration into Global Illumination", Proc. of *EUROGRAPHICS'97*, Budapest, Hugary, September 1997. In Journal of *Computer Graphics Forum*, 16(3), pp.283-292 (1997).

[109] Debevec, P., Borshukov, G., Yu, Y. et. al. " FACADE: Modeling and Rendering Architecture from Photographs and The Campanile Model ", SIGGRAPH'97 Electronic Theatre and Visual Proceedings, pp.254.

[110] DEBEVEC, P., YU, Y., AND BORSHUKOV, G. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. In *9th Eurographics Workshop on Rendering*, (1998), pp. 105-116.

[111] YU, Y., AND MALIK, J. Recovering photometric properties of architectural scenes from photographs. In *SIGGRAPH'98 Conference Proceedings*, Orlando, FL, July 1998, pp. 207–217.

[112] Yu, Y., " Efficient Visibility Processing for Projective Texture-mapping ", Journal of *Computers and Graphics*, Vol.23, No.2, 1999, pp.245-253.

[113] YU, Y., DEBEVEC, P., MALIK, J., AND HAWKINS, T. Inverse Global Illumination: Recovering Reflectance Models of Real Scenes from Photographs. In *SIGGRAPH'99 Conference Proceedings*, Los Angeles, CA, August 1999, pp. 215–224.

[114] Yu, Y., " Surface Reconstruction from Unorganized Points Using Self-Organizing Neural Networks ", Proc. of IEEE Visualization'99 Late Breaking Hot Topics, San Francisco, CA, October 1999, pp.61-64.

[115] YU, Y., FERENCZ, A., AND MALIK, J. Compressing Texture Maps for Large Real Environments. In *SIGGRAPH'2000 Conference Abstracts and Applications*, New Orleans, Louisiana USA, July 2000.

[116] YU, Y., DANA, K., MARSCHNER, S., PREMOZE, S., RUSHMEIER, H., AND SATO, Y. Image-Based Surface Details. *SIGGRAPH'2000 Course Notes*, New Orleans, Louisiana, July 24, 2000.