Realistically Editing Indoor Scenes

Edward Zhang

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

Doctor of Philosophy

University of Washington

2021

Reading Committee:

Brian Curless, Chair

Michael F. Cohen

Steven M. Seitz

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Realistically Editing Indoor Scenes

Edward Zhang

Chair of the Supervisory Committee:
Professor Brian Curless
Computer Science and Engineering

Mixed reality is an exciting application of computer graphics that seamlessly combines the real and the virtual. Many of the most compelling mixed reality applications involve modifying the contents of the surrounding real-world scene, whether it is adding a virtual game character running around your room, replacing your existing couch with a new one in a furniture retail app, or changing the style of your clothing in a fashion app. For most applications, especially those for which the primary goal is aesthetic evaluation, these edits need to be done in a visually realistic way: something that looks like a bad Photoshop job is nowhere near as useful as something that looks like you had physically placed or moved something in your room.

In this thesis, I describe methods for reconstructing models of an indoor scene's lighting and materials in a way that enables realistic, physically consistent edits to the scene. These edits include not only inserting virtual objects into a room, but also removing existing objects, as well as relighting and retexturing.

I first outline a method to scan scenes to acquire a 3D mesh textured with a high dynamic range representation of the scene's appearance. I then show how to use this captured data to infer parametric models of the room's base geometry (*e.g* walls, floor, doors, windows). I present an inverse rendering framework to use the high dynamic range data to simultaneously infer light intensity distributions and diffuse reflectances across an entire indoor scene that enables realistic visualizations of emptying, refurnishing, and relighting a room. I subsequently derive a method

to solve for not just the intensities but also the number and locations of unobserved local light emitters in a scene. Finally, I describe a neural rendering method to reconcile the differences between a scene's true appearance and its reconstructed scene model.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Mixed reality involves the seamless combination of the virtual world and the real world. The degree of overlap between real and virtual worlds varies, ranging from the fully synthetic worlds of consumer virtual reality experiences, to the current generation of mobile augmented reality applications which simply overlay text and imagery onto a camera feed.

However, the most compelling applications of mixed reality involve a meaningful interplay of the user, the virtual world, and the real world. For example, in a furniture retail application, a user may wish to preview how a new couch might look in their living room, which may further involve virtually replacing their existing couch and shuffling around the remaining furniture to fit. In more general terms, many mixed reality experiences involve editing the user's surroundings in a visually realistic way.

The visual realism of the mixed reality experience is vital for many applications. For example, for interior design applications, the primary goal is the aesthetic evaluation of potential designs, which necessitates that visualizations be accurate. The edits to the room need to look completely real, as if the user had physically rearranged the contents of their room.

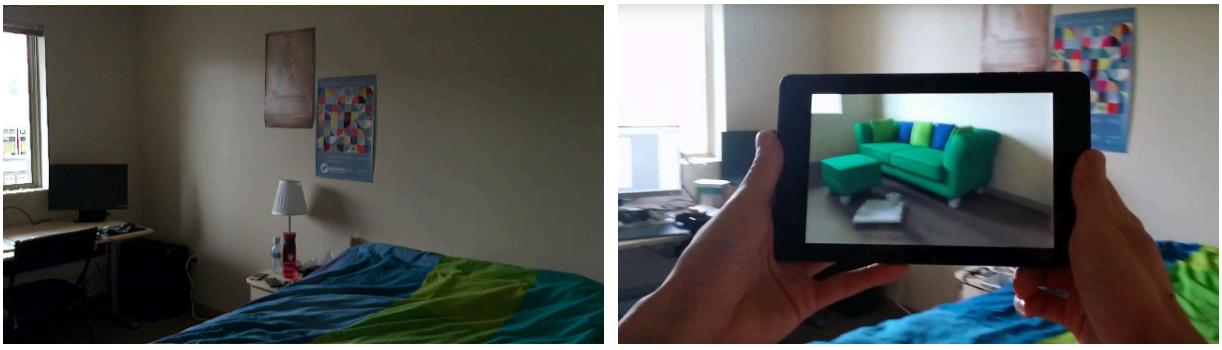
Current augmented reality applications can already directly insert 3D models into scenes (Figure 1.1). However, the results often have inconsistencies that prevent them from looking fully realistic. Frequently, an inserted object does not look like it fits in the room because it is not illuminated consistently with the rest of the room. For example, a stock model of a chair will be rendered under fairly uniform lighting, while the room may be lit by a window casting noticeable shadows. Even if the lighting in the room is visually similar to the rendered lighting, there still may be noticeable inconsistencies, since a real physical object also affects the appearance of the rest of the scene: it may cast shadows on neighboring objects, or get reflected in a glossy wooden

floor. Thus, in addition to scene geometry, understanding the characteristics of **illumination** and **surface reflectance** in the scene is crucial for realistic scene editing.

Figure 1.1: A screenshot from the IKEA Places AR application, inserting a virtual couch into the scene. While the results are plausible, the appearance of the couch is noticeably inconsistent with the rest of the scene – the shadow underneath the couch does not match the shadows of the objects on the left side of the image, and the lighting on the couch itself suggests strong overhead lighting, which is not present in the scene itself.

To enable these scene edits, I take the approach of reconstructing an *editable*, *renderable* model of the geometry, lighting, and materials constituting an indoor scene. Having this model be editable means that the reconstructed properties are semantically meaningful – a user should manipulate a chair rather than a triangle in a mesh, or turn off a particular light in a room rather than edit a set of light probes. Without a semantically meaningful scene model, performing user-specified edits is tedious and nonintuitive. Meanwhile, representing the geometry, materials, and lighting in a renderable way allows the use of photorealistic rendering to produce realistic-looking, self-consistent edited images, some examples of which are shown in Figure 1.2.

In this thesis, I describe methods to infer editable, renderable models of the geometry, lighting, and materials in indoor scenes, and show how these scene models can be used to enable various realistic scene edits. These edits extend beyond simply inserting virtual objects into the scene, but also include realistically removing real objects from the scene, as well as relighting and retexturing.



(a) Input (b) Refurnished

(c) Input (d) Refurnished

Figure 1.2: Refurnishing results from this thesis

- In Chapter 3, I present a method to capture the high-quality data needed to construct an accurate scene model, focusing on acquiring a consistent, high-dynamic-range estimate of the scene's appearance.
  - **Input:** An autoexposed RGBD scan of the entire scene, with known camera poses.
  - **Output:** A triangle mesh with per-vertex radiance estimates in a linear space. This type of reconstruction can be used in all subsequent chapters.
  - **Assumptions:** The scene is largely diffuse.
- In Chapter 4, I show how to construct a parametric model of a room's geometry from a scan,

including not only the room's floorplan, but also its architectural elements such as doors, windows, and baseboards.

  - **Input:** For floorplan estimation, a triangle mesh. Architectural feature estimation additionally uses RGB images with camera poses.
  - **Output:** A parametric geometric model of the room, as well as per-vertex semantic labels of surfaces corresponding to the room's architectural elements.
  - **Assumptions:** The ceiling and floor can each be represented as a single flat plane. The floorplan obeys the Manhattan world assumption, as do the doors, windows, and baseboards.

- In Chapter 5, I present an inverse rendering method that computes light intensities (but not their locations) and material properties. I show how combining these results with those in the previous chapters enables realistic refurnishing and relighting applications.

  - **Input:** Triangle mesh with per-vertex radiance estimates; light source positions (can be estimated with heuristics); material segmentation (can be rough and incomplete, *e.g* wall/floor/ceiling labels).
  - **Output:** Diffuse reflectances for each material segment and light intensity distributions for each light.
  - **Assumptions:** Materials are diffuse. The scan is fairly complete, covering most of the room's surfaces.

- In Chapter 6, I investigate the problem of inferring the cardinality and positions of lights in a scene. Specifically, I introduce the *light localization* problem: given only the appearance and geometry of the scene, can the number and location of the point lights illuminating it be determined? I then derive a novel voting scheme and refinement algorithm to solve the light localization problem for isotropic point lights. In comparison to Chapter 5, I assume known materials but unknown positions and numbers of lights.

  - **Input:** Triangle mesh with per-vertex radiance estimates; per-vertex materials.
  - **Output:** Light positions and intensities.
  - **Assumptions:** Unknown point lights are isotropic.

- Finally, in Chapter 7, I go beyond the inferred scene models and show how to compensate for the differences between model and reality in edited results. Specifically, I present a neural rendering system that removes objects and their shadows from images using very rough scene proxies.

  - **Input:** RGB image; renderable model of scene geometry and lighting (such as those produced from the inverse rendering methods in previous chapters); a mask specifying which object in the original RGB image to remove.
  - **Output:** RGB image with the object and its shadows removed.
  - **Assumptions:** Shadow receivers are diffuse and have a regular texture.

Chapter 2

BACKGROUND

What sorts of scene manipulations might be needed for mixed reality applications? Based on the furniture retail and interior design examples, the manipulations can be categorized into the following four operations:

- Virtual objects may be inserted into the scene (Augmenting). For example, a user wants to see what a new couch will look like in their room.

- The visual properties of existing objects may be modified; this includes changing reflectance properties of objects, as well as changing the lighting conditions in the scene (Retexturing and Relighting). For example, a user might want to see what their newly refurnished living room looks like during the daytime versus under artificial lighting; they also might want to know how the room would look if they replaced the wallpaper or changed the fabric on the couch.

- Existing real objects may be removed from the scene (Diminishing). For example, before adding a new couch, the user may want to make room for it by removing our existing couch.

- Existing real objects may be geometrically manipulated, which includes moving them around or rotating them (Manipulating). For example, the user may wish to shift their existing end tables to lie at the ends of the new couch.

While the Manipulation task seems like an extremely difficult problem, it can be simplified by decomposing it into a three-step process, including two other fundamental mixed reality operations: first, a standard Diminishing operation removing the object under manipulation, followed by a

3D inference task solving for the appearance and shape of the object under manipulation (which is beyond the scope of this thesis), and finally a standard Augmenting operation inserting the inferred model into the scene. Based on this breakdown of Manipulation operations, effective implementations of Augmenting and Diminishing operations become doubly important.

The insertion of virtual objects into scenes has a long history, dating back to Debevec's seminal work in 1998 [27]. The general approach taken is to use the scene's illumination conditions as well as the geometric and reflectance properties of surfaces local to the area in which the virtual object is to be placed to render the virtual object, as well as its effects on the local scene. While Debevec assumed known illumination and geometry, recent works have added the ability to infer reflectances [11], geometry [63], and illumination [62], all from a single image. A large area of focus in the augmented reality community is achieving these results on a video stream in realtime [111, 112].

Retexturing and relighting are also well-studied tasks. Retexturing is often accomplished with the aid of intrinsic image decomposition [9, 45], where an input image is separated into two components representing diffuse reflectance (frequently assumed to be piecewise constant) and residual lighting variation (frequently assumed to be slowly varying) [54, 5]. Retexturing edits are easily performed on the diffuse reflectance image, which can then be recomposed with the lighting image to obtain a convincing result. Such an approach is taken by recent video retexturing works such as [12, 90]. The retexturing problem is also frequently framed more generally as an image-to-image translation problem, although because of the more general approach the results are often less realistic than results which explicitly consider lighting and shading [58, 166].

Some relighting methods infer the parameters of a parametric BRDF model (as well as the initial lighting conditions) for surfaces of interest; this reflectance model can then be rendered under arbitrary lighting conditions [77, 139]. Other methods take several aligned images of the scene under different lighting conditions and infer a set of basis images, which can then be recombined to match novel lighting conditions [28, 82, 98]. Recently, relighting approaches based on deep neural networks also achieve impressive results [110, 148].

While the operations of Augmenting, Retexturing, and Relighting are well understood, the

study of removing real objects from scenes has been extremely limited. Systems which perform the Diminishing task usually operate solely in image space using inpainting methods, whether parametric [10] or nonparametric [4]. Some inpainting methods take the 3D structure of the scene into account (e.g. PixMix [51]), but still do not model lighting or materials, and as a result only provide convincing results for removing isolated objects (where shadows and specularities lie only on easily inpainted surfaces). More recent work attempts to automatically perform the diminishing task using neural networks [118]; however, in general end-to-end networks for image manipulation do not produce highly realistic, physically consistent results.

As with diminished reality, the manipulation of real objects in scenes has been restricted to methods that operate primarily in the image domain. These works primarily attempt to infer geometric structure to perform geometric manipulations, and do not deal with accurate modeling of lighting or surface reflectance [18, 21]. Kholgade [67] presents the sole system enabling visually realistic manipulations, inferring diffuse reflectances and distant illumination properties from a single image, with some user annotation.

Given the lack of existing work in the areas of Diminishing and Manipulating applications, and the fact that Diminishing is an important component of Manipulating, it is clear that further investigation of realistic object removal is of great importance to mixed reality applications. Much of the work in this thesis is designed to be applicable to object removal applications as well as other scene manipulations.

Chapter 3

CAPTURING THE APPEARANCE OF INDOOR SPACES

3.1 Introduction

(a) Scanned Scene (b) Without Radiometric Calibration (c) With Radiometric Calibration

Figure 3.1: My capture system allows for the capture of geometry and appearance of a room, such as an office (left). The crucial component of this system is the radiometric calibration of captured images to obtain a high dynamic range mesh. Directly using autoexposed images to texture a mesh results in visible boundary artifacts and inconsistent appearance (center). Performing radiometric calibration results in a smooth appearance (right).

In this chapter I describe how to use a commodity RGBD sensor and leverage auto-exposure to recover 3D meshes with high dynamic range (HDR) textures, such as the one shown in Figure 3.1[1]. High dynamic range imaging is particularly important, as windows and directly lit surfaces can be far brighter than many other surfaces, and inferring materials and lighting in the scene will require having all measurements in a single linear radiance space. Auto-exposure provides a natural mechanism for adaptive sampling across exposures for HDR recovery. A novel global exposure

[1]This chapter describes work originally published in the ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)[155]

optimization framework borrowing ideas from bundle adjustment is the key contribution enabling consistent HDR scans from autoexposed images.

3.2 Background and Related Work

3.2.1 Simultaneous Localization and Mapping

My capture system is heavily reliant on Simultaneous Localization and Mapping (SLAM) methods to obtain camera poses and scene geometry. Real-time volumetric fusion, introduced in [96], has been extended for use in larger scale scenes [97, 137], and combined with global consistency methods [138, 26] to obtain high quality geometry.

These SLAM systems perform very well for many kinds of scenes. However, for indoor scenes, large textureless planar regions are common, and the field of view of most common RGBD sensors is not wide enough to capture these regions while still retaining tracking. Using specialized hardware, such as multiple cameras, wide field-of-view cameras, and/or IMUs, helps address these problems. [71, 94] describe the localization system used in Google's Project Tango[2] device, which uses visual-inertial odometry with a wide field-of-view camera for robust localization. I use the Project Tango device in this thesis for room-scale scanning.

3.2.2 High Dynamic Range Capture

Operating in linear color space is important for many computer graphics applications. Images captured from camera sensors must be radiometrically calibrated to account for changes in exposure and for camera response.

Recovering relative exposure values for a set of images has been well studied for 2D problems such as panoramic photo stitching [42] and undoing video autoexposure [68, 46]. Grossberg and Nayar [44] describe several ambiguities that arise when computing camera responses and unknown exposures simultaneously and how to resolve them. These methods rely on point feature tracking and affine image warping, which are unsuitable for 3D captures with significant parallax. I extend

[2] https://www.google.com/atap/project-tango/

these methods for 3D scenes by leveraging the scene geometry and camera poses reconstructed during scanning, resulting in high dynamic range mesh textures.

Several recent works recover high-dynamic range textures for meshes by extending realtime volumetric fusion frameworks [89, 79], and estimating per frame exposures. These results suffer from global consistency problems when returning to previously scanned regions. This work on radiometric calibration is complementary to these works, because my system optimizes for a globally consistent texture, analogous to bundle adjustment processes to handle loop closures.

3.3 Overview

The goal of this capture system is to produce a mesh reconstruction of the scene accurate appearance represented by high dynamic range colors. As input, the system takes a set of low dynamic range images, the associated camera poses for these images, and a triangle mesh of the scene. A variety of devices and algorithms can provide this data. I use a Project Tango tablet as a capture device. Camera poses are computed by the Project Tango visual-inertial tracking system, while meshes are generated by aligning the raw depth data using the camera poses and then running Poisson Surface Reconstruction [66] on the resulting point cloud.

3.4 Formulation

Given the mesh and images, the first step is to radiometrically align these images into a common linear radiance space. The original images are taken with automatic camera settings, which vary exposure as well as white balance.

For radiometric calibration of exposure, I use similar methods to those used in panoramic photo stitching, such as the one presented by Goldman [42]. These methods first put pixels of the original images into correspondence; while these pixels may have differing low dynamic range values, they should represent the same radiance incident on the camera. In the panorama formulation, this assumption holds because the camera centers are assumed to be coincident and therefore corresponding pixels fall along the same ray from the camera center. Then the camera exposures and the unknown scene radiances are simultaneously recovered by minimizing the reprojection error in

pixel space. This method can also recover global camera parameters such as vignetting functions (as in [42]) and response functions (as in [43]).

In my 3D formulation, correspondences are obtained by projecting images onto the scene geometry, associating each vertex with the set of pixels projected onto it. I assume that these pixels all represent the same radiance incident on the camera by assuming a diffuse world. While camera response and vignetting parameters can be solved for as in [43, 42], I focus on the exposure and radiance recovery and assume pre-linearized images with no vignetting. The result is a nonlinear optimization problem:

$$\min_{t_j, b_i} \sum_{i,j} (t_j b_i - X_{ij})^2 \tag{3.1}$$

where t_j are the per-image exposures, b_i are the vertex radiances, and X_{ij} is the observed pixel value of vertex i in image j.

With linear response, there is a scale ambiguity: given an optimal (t, b), $(ct, b/c)$ for some constant c yields the same value for the cost function. Thus I fix the exposure in the first frame of the input $t_0 = 1$ to remove this ambiguity.

Camera white-balance is modeled as per-frame, per-channel scale factors W_j^R, W_j^G, W_j^B. Combined with exposure, the per-channel scales are then $(t_j^R t_j^G t_j^B) = t_j(W_j^R W_j^G W_j^B)$. I solve Equation 3.1 for each color channel independently, resulting in the final per-frame, per-channel scale factors $(t_j^R t_j^G t_j^B)$ directly.

3.5 Radiometric Calibration

I first perform an inverse gamma correction ($\gamma = 2.2$) on the input images. I verified using a Colorchecker Chart that this was sufficient to bring the images to be close to linear in the range of lighting conditions I captured. Vignetting was not found to significantly affect Project Tango images.

After linearizing the images, I then project them onto the scene geometry: for every mesh vertex V_i, I trace a ray from each camera center C_j. If ray $\overrightarrow{C_j V_i}$ does not intersect the scene, and

falls within the field of view of the camera, I locate the pixel that the ray intersects and add it to the list of pixels associated with V_i.

Errors in camera poses and scene geometry result in incorrectly projected pixels at depth discontinuities and texture edges. I ignore projections of pixels that fall near strong gradients in the input images, which usually coincide with depth discontinuities and texture edges.

For efficiency, I optimize over a random subset of the vertices ($N < 200,000$). For each associated pixel of each vertex, I add a data term to the cost function. The minimization is performed using Ceres Solver [2]. I found that results were not highly dependent on initialization, so I arbitrarily initialized radiances and exposures to 1.

The cost function closely resembles the one used in Structure from Motion, but, instead of camera poses and 3D point locations, I solve for camera exposures and scene point radiances. The similar sparsity patterns allows the solver to take advantage of advances in sparse bundle adjustment, as described in [87], for more efficient minimization using Schur-based methods.

3.6 Radiance Reprojection

In this step, I associate radiance samples with mesh vertices. For every image, I trace a ray from the camera center to each mesh vertex. If this ray does not intersect any geometry and falls within the bounds of the camera frustum, I add a radiance sample with value $\frac{X_{ij}}{t_j}$ to that vertex.

A radiance sample from camera j assigned to vertex i receives a weight w_{ij}. This weight is proportional to the projected differential area of the mesh vertex onto the pixel, analogous to the differential form factor between the camera pixel and the mesh vertex [20]. Let g_{ij} be this projected area, v_{ij} be the vector from the center of camera j to vertex i, $\hat{v}_{ij} = \frac{v_{ij}}{||v_{ij}||}$, o_j be the direction of the optical axis of camera j, and n_i be the normal at vertex i; then

$$g_{ij} = \frac{(-\hat{v}_{ij} \cdot n_i)(\hat{v}_{ij} \cdot o_j)}{||v_{ij}||^2}. \tag{3.2}$$

Intuitively, the weight factor g_{ij} represents how much the pixel value is affected by vertex i's appearance: if a large mesh surface area projects onto the same pixel, then each vertex in that area

has a correspondingly smaller impact on the value of that pixel. This weight thus favors radiance samples that are taken from head-on directions that are closer to the surface.

The total weight is also proportional to the confidence value c of the radiance sample. This confidence value is based on the original 8-bit input images; saturated and underexposed pixels are less likely to be reliable. I use a hat function for the confidence value, as suggested in [29], but slightly modified to have nonzero confidence for a saturated pixel so that vertices only having saturated projected pixels will not be black:

$$c(X) = \begin{cases} \frac{256-X}{127}, & X > 127 \\ \frac{X}{127}, & X \leq 127 \end{cases}. \tag{3.3}$$

If X_{ij} is the original LDR pixel value, then the total weight is

$$w_{ij} = c(X_{ij})g_{ij}. \tag{3.4}$$

For this capture system, I assume diffuse surfaces and assign each vertex a single radiance value, which for robustness is computed as the per-channel weighted median of the vertex's radiance samples. Handling specular surfaces by analyzing the full set of radiance samples is an area for future work.

An example of the results of the capture process is shown in Figure 3.2.

3.7 Evaluation

I demonstrate the importance of this globally optimal exposure estimation framework by comparing to two other methods: a frame-to-frame method, and a frame-to-model method.

The frame-to-frame baseline method is based on video radiometric calibration pipelines such as [68, 46], which typically only compute exposure ratios between consecutive frames. In my implementation of the frame-to-frame method, dense correspondences between consecutive frames of the input are obtained via optical flow, and the relative exposure is estimated as the median ratio of corresponding pixel values. This method does not have any notion of global consistency across all frames, since it only evaluates exposures between consecutive frames.

(a) Input Frame 2557 (b) Re-exposed Frame 2557 (c) Input Frame 2585 (d) Re-exposed Frame 2585

(e) Uncalibrated Mesh Texture (f) Calibrated Mesh Texture (g) Calibrated Mesh, Low Exposure (h) Calibrated Mesh, High Exposure

Figure 3.2: Comparison of office mesh, before and after radiometric calibration. (a) and (c) are two of the original input frames showing a similar area of the scene; note that (c) is significantly darker in appearance than (a) due to autoexposing for the lights. (b) and (d) are derived from (a) and (c), respectively, but rescaled to have the same exposure level. (e) shows a naive texturing of the scene geometry without any exposure correction; there are many noticeable artifacts on the wall. (f) shows the texturing after radiometric calibration (linearly tonemapped and clipped). The artifacts on the wall have been completely smoothed. (g) and (h) show another view of the mesh with calibrated texture at differing exposure values; notice how the lights are all brought within range.

The frame-to-model method is based on the KinectFusion-like methods proposed in [89, 79]. In my implementation of the frame-to-model method, I maintain an online estimate of radiance for each vertex in the 3D model. For each frame, I first estimate the exposure by projecting pixels onto the 3D model and taking the mean ratio of pixel value to current vertex radiance. I then integrate the frame into the model by updating the average vertex radiances. As in basic KinectFusion, some global consistency is maintained but the lack of explicit loop closure handling means that drift still occurs.

(a) No exposure correction (b) Frame-to-frame (c) Frame-to-model (d) Ours

Figure 3.3: Qualitative comparison of exposure estimation methods on Bedroom 1 dataset.

For these comparisons, I avoid the difficulties of simultaneous estimation of camera response in existing frame-to-frame methods, and pose estimation in existing frame-to-model methods. Thus, I use our camera poses and prelinearized images as input to my reimplementations of the exposure estimation components of these methods. This allows a qualitative comparison of these two methods to mine by reprojecting the estimated radiances onto the scene geometry (see Section 3.6). The results are shown in Figure 3.3.

The frame-to-frame method smooths some of the exposure variation (e.g. along the bottom of the wall to the right of the bed), but shows extensive banding and discontinuities across loop closures. The frame-to-model method is a significant improvement, but still shows some artifacts due to its lack of explicit loop closure handling. My method gives results that, while not perfect, are smooth almost everywhere.

Several more results of captured real scenes are shown in Figure 3.4.

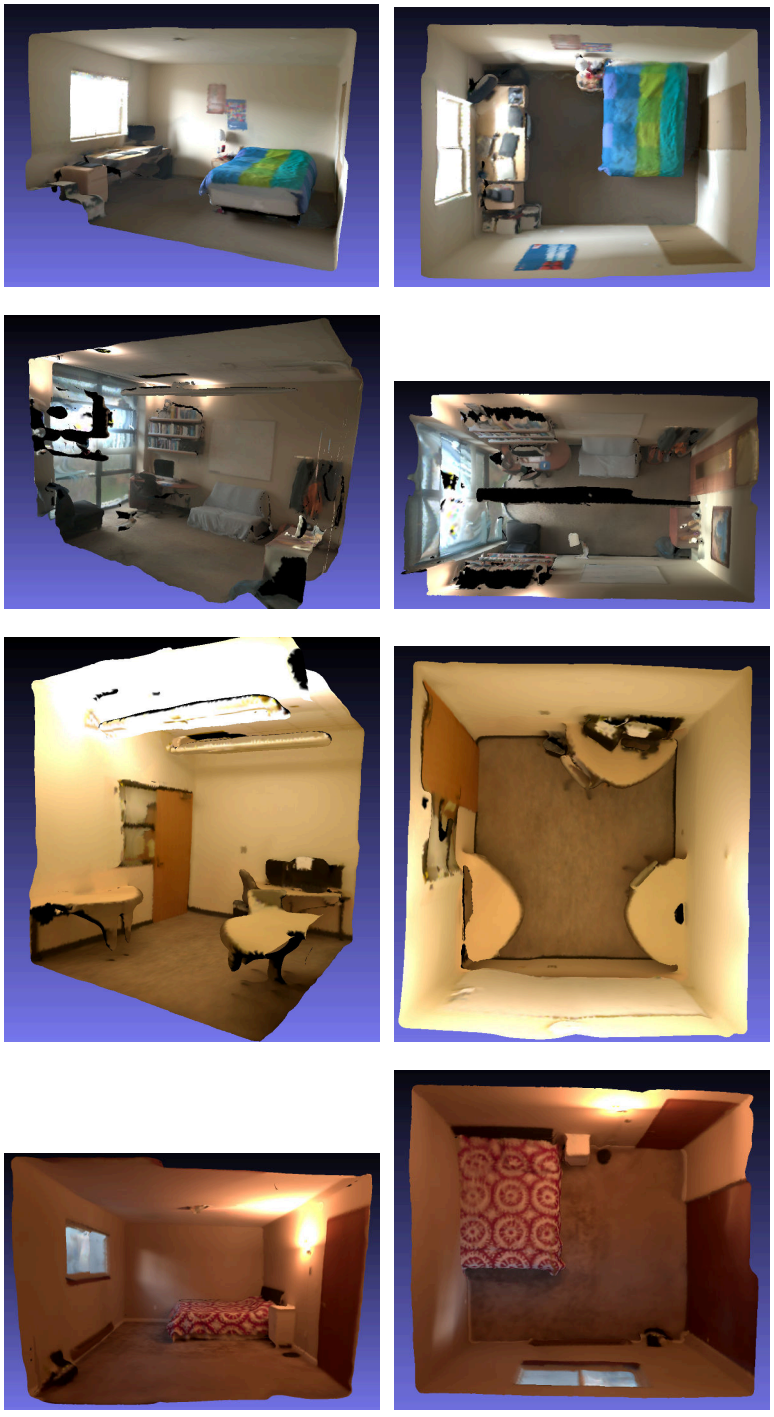


Figure 3.4: Results of the capture system on real scenes. The left column shows a wide angle view of the scene, while the right column shows an overhead view of the scene.

Chapter 4

INFERRING FLOORPLANS AND ARCHITECTURAL FEATURES FROM INDOOR SCANS

4.1 Introduction and Related Work

In this chapter, I describe a collection of methods to construct an editable parametric geometric model of a room from a triangle mesh[1]. The inferred scene geometry contains not only the outer surfaces of the room (*i.e* walls, floor, and ceiling), but also important architectural elements such as doors, windows, and baseboards.

Constructing such an "empty room" model has several benefits. It forms an segmentation of the input mesh into moveable room contents and stationary room structures. It corrects some scan inaccuracies by smoothing out noise on flat surfaces while sharpening corners. Finally, when performing object removal operations on the contents of the scene, the scene model provides a way to fill in the holes behind the removed objects.

A major simplifying assumption is that the floorplan and architectural elements all obey the Manhattan-World assumption - that is, all surfaces are perpendicular to one of a common set of three orthogonal axes. Many prior works on floorplan estimation also use this Manhattan-world assumption. Sankar produces Manhattan-world floorplans from a single smartphone panorama using a small amount of user interaction [114]. Colburn creates more photorealistic scenes by using high dynamic range view-dependent textures for rough user-specified planar proxies, and allows large-scale scene edits, such as removing entire walls [21]. I leverage many of the insights from a number of works from Furukawa which aim to model complete indoor floorplans from sparse point clouds obtained from photographs [15, 57] or from range data [146].

---

[1]This chapter describes work originally published in the ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)[155]

4.2 Floor Plan Estimation

In this section I outline how a rough floorplan is computed from a 3D triangle mesh scan of an indoor scene. I estimate 2D wall layout as well as the height of the room. I assume that walls, floor, and ceiling obey the Manhattan-world assumption, and furthermore that the floor and ceiling are each a single plane. While these are restrictive assumptions, they suffice for many indoor scenes. I refer to the input mesh resolution, approximated by the mean edge length, as r.

4.2.1 Manhattan-World Coordinate System

I first determine a coordinate system that aligns the y axis with the world up vector (with the floor plane at $y = 0$) and the x, z axes with the Manhattan-world coordinate system for the walls. I do this by histogramming mesh vertex normals and then extracting quasi-perpendicular axis vectors, as in [38]. Note that this step assumes that the y axis of the input mesh is less than 45° from the true world up vector (in other words, the y axis is closer to the up vector than any other axis). This is a reasonable assumption in most natural scanning setups; furthermore scanning devices with accelerometers (such as the Project Tango tablet) can directly determine the up direction.

4.2.2 Floor and Ceiling

I determine the floor plane by histogramming the y coordinate of all vertices with approximately vertical normals, where the width of a histogram bin is set to r. The first local maximum is classified as the floor plane. The ceiling plane is determined in the same manner.

4.2.3 Walls

My floor plan determination process is similar to Cabral and Furukawa's [15] graph-based formulation. I first project all vertices onto the floor plane and discretize into a grid. The width of a grid cell is r. A wall likelihood score s is computed for each grid cell G_p at coordinates $p = (x_p, z_p)$ and each of the four axis-aligned normal directions $n \in \{(1, 0), (-1, 0), (0, 1), (0, -1)\}$, by examining

the vertices V (with coordinates (x_V, y_V, z_V) and normal n_V) that project into the cell:

$$s(G_p, n) = \sum_{V \in G_p} \log(1 + y_V)(n_V \cdot n). \tag{4.1}$$

This score counts vertices with normals aligned with the axis in question, weighted by the degree to which the normal aligns with the axis as well as by the height of the vertex. Since walls are vertical, a uniformly sampled mesh of the scene should project many vertices onto the same cell. Note that the dot product term penalizes any points with normals facing away from the axis. The height-weighting is inspired by [146], which makes the observation that, while walls may be occluded at lower heights, they tend to be unoccluded near the ceiling.

I also observe that walls tend to be local bounding planes of the scene (if not globally bounding), which is not directly leveraged in prior floor plan work. If cell G_p were a wall, then one would expect not only that $||G_p||$ is large, but also that the cells behind the wall (*i.e* adjacent cells in the negative normal direction) are empty, and thus any points projecting into those cells are due to noise. Assume a Gaussian noise distribution on vertex coordinates, with $\sigma = \frac{r}{2}$. Due to discretization, it is possible that the directly adjacent cell G_{p-n} also has a large count (the worst case being a wall at $p - \frac{n}{2}$). However, in this worst case no more than $||G_p|| \frac{1-\text{erf}(\sqrt{2})}{2} \approx 0.023||G_p||$ vertices should fall in the next cell G_{p-2n} (two cells away from the original G_p). Thus, I set $s(G_p, n) = 0$ if $||G_{p-2n}|| > 0.023||G_p||$. An example of the final weighted histogram is shown in Figure 4.1.

I trace scanlines of the grid, looking for contiguous sets of cells with $s(G_p, n) > N$ to extract candidate wall segments, where N is the expected minimum weight of a wall cell. If the height of the room is h, then one would expect a perfect wall to have weight $\sum_{i=0}^{h/r} \log(1 + ir)$; assuming that for any vertical section of wall, at least half of it is unoccluded, then I set $N = \frac{\sum_{i=0}^{h/r} \log(1+ir)}{2}$. In practice, if the wallfinding fails I simply decrease N and repeat. The candidate wall segments are pruned using non-maximum suppression.

To construct a floorplan, I seek a cyclic, ordered list of these wall segment candidates. This is performed via a graph-based approach, where the endpoints of wall segment candidates are the nodes in a graph. A single endpoint of a segment can only be connected to one of the endpoints of a perpendicular segment, since the normals of the two segments must be consistently oriented.

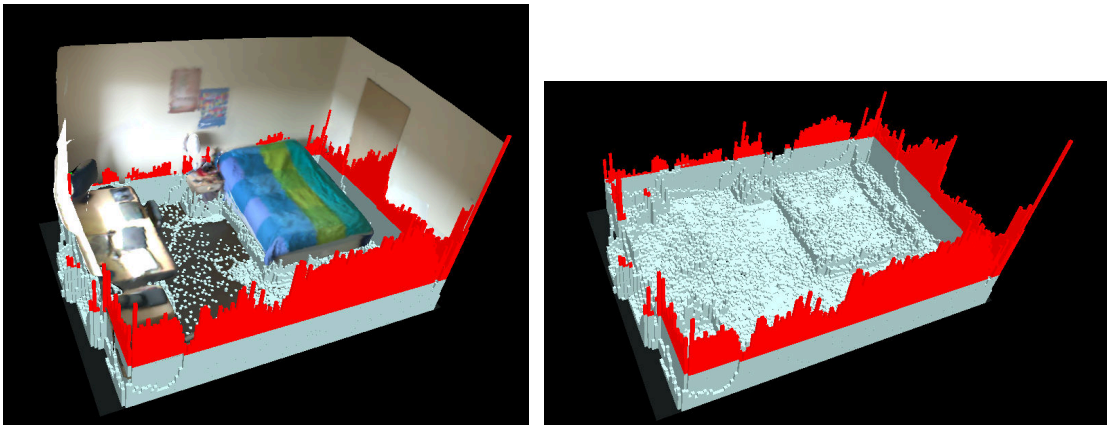


Figure 4.1: Weighted histogram for wall detection (right), overlaid on scene (left). The portion of histogram bins above the wall detection threshold N are shown in red.

The weight of each edge is simply the Manhattan distance between the two endpoints it connects. There is an edge between the two endpoints of a wall segment candidate, which has weight $\alpha = 1$ to penalize model complexity (favoring models with fewer edges), as in [15]. I assume the longest candidate line segment will be part of the floor plan, and find the minimum cost path between its endpoints using Djikstra's algorithm.

4.2.4 Vertex Labeling

After determining walls, floor, and ceiling, I assign each vertex of the mesh a label. A vertex is classified as a wall, floor, or ceiling if its vertex normal approximately matches the plane normal and it is no farther than r from the plane; the vertices that do not fall in these categories are labeled as "other". I morphologically dilate and erode the labels along the mesh topology to close small holes in the labeled surfaces.

4.3 Architectural Features

I identified doors, baseboards, and windows as common elements of scenes that are easy to identify automatically, simple to model, and distinctive features that are important for the perception of the

indoor scene. As with the floorplan estimation pipeline, I make heavy use of Manhattan world assumptions to extract the location of these components.

4.3.1 Manhattan-World Analysis

Most architectural features, not only baseboards and doors but also windows, light switches, and outlets, are located on walls (possibly recessed or protruding), and are axis-aligned and rectangular. Due to inaccuracies in camera poses and geometry, finding rectangles in reprojected mesh space is difficult. Instead, I locate axis-aligned edges in individual input images, project the edges into 3D mesh space, and then infer rectangular features in that 3D space.

First, I transform the camera poses into the axis-aligned coordinate system described in Section 4.2, and then compute the coordinates of the three vanishing points for each image. I then run the oriented edge filter used in [38] to obtain oriented edge images for each camera viewpoint. I use the wall, floor, and ceiling labels to identify pixels that project onto walls (rather than onto occluders) and only consider edge responses for these pixels. This process is illustrated in Figure 4.2.

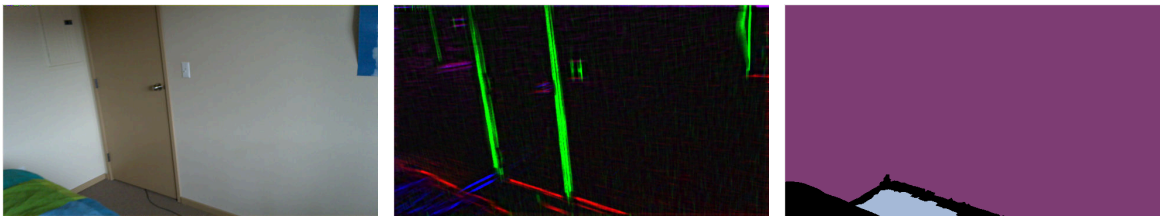


Figure 4.2: From the camera pose of the input image (left), an oriented edge filter is computed to detect edges aligned with the scene's Manhattan-world coordinate system (center). Rendering the labels of the scanned mesh from the same camera viewpoint (right) isolates edge responses that lie only on walls (purple), rather than occluders (black) or floor (blue).

4.3.2 Baseboards

Baseboards are primarily parameterized by their height, although they may also have a depth (i.e. they may protrude from the wall). I locate a baseboard by assuming that there should be a horizontal edge on all observed walls at the same height.

I first discretize each wall in the room into a 2D grid of cells, and then project the masked oriented edge images onto these grids. Each cell will record how many pixels project onto it $N(x, y)$, as well as the total horizontal edge response of pixels projecting onto it $F(x, y)$. I then define a function $B(y)$ to measure the likelihood of the baseboard height being at y

$$B(y) = \frac{1}{\sum_x \text{nonzero}(N(x, y))} \sum_x \frac{F(x, y)}{N(x, y)}$$

where $\text{nonzero}(x) = 1$ if $x > 0$ and 0 otherwise. Note that when $N(x, y) = 0$, that grid cell is occluded for all views. I take the first local maximum for B as the baseboard height.

The motivation for this function is as follows: $\frac{F(x,y)}{N(x,y)}$ is the average horizontal edge weight in a cell. The $\frac{1}{\sum_x \text{nonzero}(N(x,y))}$ term accounts for occlusions: $\sum_x \text{nonzero}(N(x, y))$ is a count of how many wall cells at height y were observed. If only a few cells at a particular height were visible, but they all had strong edge responses, it is still likely to be a baseboard candidate and that height should not be unfairly penalized.

Once a baseboard height is computed, vertices corresponding to the baseboard can be labeled. Then, the approximate depth of the baseboard can be computed as the median horizontal distance between the wall plane and the vertex position.

4.3.3 Doors

I assume doors to be axis-aligned rectangles on walls that extend from the floor to a height above 1.5 meters. They are parameterized by: the index of the wall they are on i, the height h, the position of the center of the door p, and the width w. Doors may also be recessed (such as many closet doors) or protruding (specifically, door trim may protrude from the wall).

Similarly to the baseboard case, I project the masked oriented edge image into the discretized

walls. I then trace scanlines of the grid of responses to extract candidate line segments and prune the candidates using non-maximum suppression. Note that when tracing line segments, I allow line segments to extend into unobserved cells to account for occlusion. A visualization of these candidate line segments is shown in Figure 4.3.

Figure 4.3: An example of the detected edge responses over all input images is shown overlaid on the scanned mesh on the left. The same candidates overlaid on the computed floorplan is shown on the right.

For each wall, I iterate through all pairs of vertical line segments on that wall. If both segments start near the ground and end near the same height greater than 1.5 meters, and there exists a horizontal line segment at approximately that height that covers at least the extent of the door, then that defines a candidate door. If several candidate doors overlap, I take the widest candidate to be the true door.

Vertices within the bounds of the rectangle and lying on the wall are labeled as door pixels. The depth (positive for protruding doors, and negative for recessed doors) is computed the same way as the baseboard depth.

4.3.4 Windows

Windows are treated similarly to doors. I trace scanlines in the discretized wall images formed from projecting masked oriented edge images. However, assembling window candidates is much less restricted than for doors, as windows may have many different aspect ratios and positions on the

walls. In this thesis, I assume for simplicity that windows to be reconstructed are uncovered (having no blinds or curtains). Thus, glass windows are not reconstructed by most geometric capture systems; therefore rectangular holes in the mesh (or, in the case of the Poisson reconstructions described in Chapter 3, heavily recessed parts of walls) are used to locate candidate bounding rectangles.

For daytime scans from the high dynamic range capture system described in Chapter 3, another identifying feature of windows is their brightness. Outdoor lighting is many orders of magnitude brighter than indoor scene surfaces. Thus when images containing images of a window are projected onto the Poisson mesh, the window can be identified by segmenting the high dynamic range reconstruction based on overall brightness.

4.4 Results

Results of this floorplan and architectural feature estimation system are shown in Figure 4.4, along with the input scans from which they are derived.

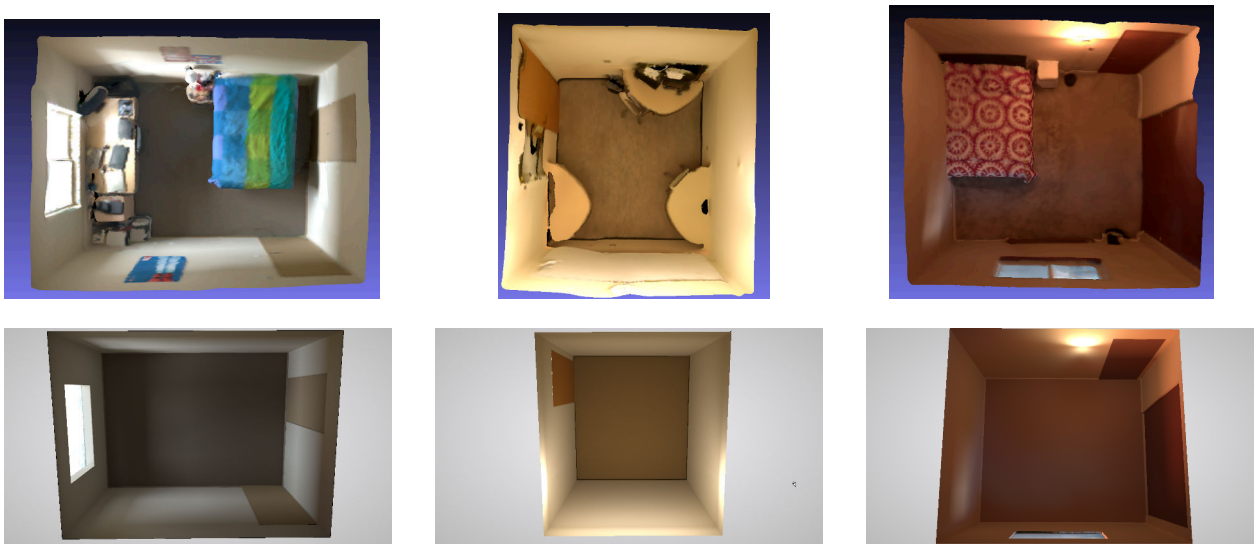
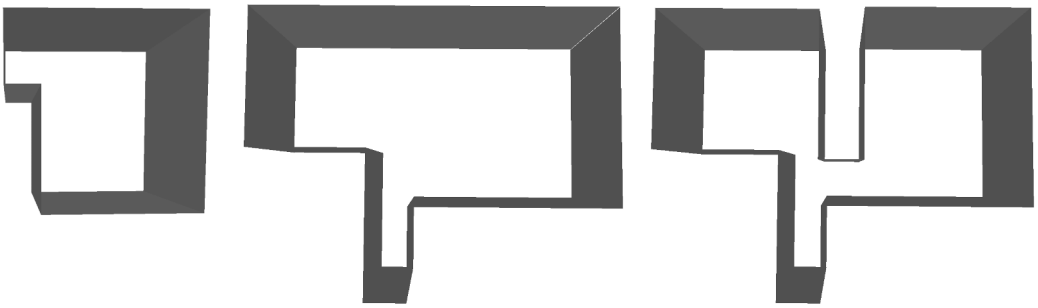


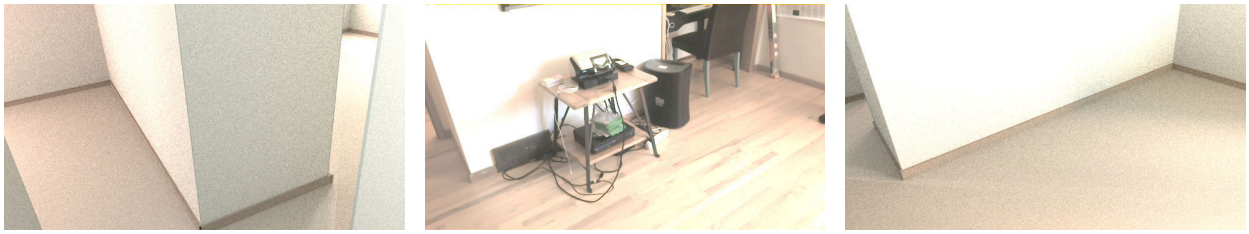
Figure 4.4: Inferred floorplans and architectural features for several scenes. The top row shows the input scans, while the bottom row shows the inferred floorplan and architectural features.

The floorplans in Figure 4.4 are fairly simple box rooms; however, the floorplan estimation algorithm is able to handle more complex layouts. On more complex scenes, such as extending the "Playroom" dataset into the adjacent hallway and bedroom (see Figure 4.5), artifacts in reconstructing thin walls and doorways become apparent. However, with suitable choices of parameters, the system computes a reasonable floor plan. The inverse lighting system described in the following chapter is independent of the reconstructed floorplan, so rendering the edited scene with the computed floor plan still produces plausible results.

(a) Input data



(b) Floorplan with default parameters (c) Floorplan with increased N (d) Floorplan with tweaked geometry



(e) Both rooms, no doorways (f) Original input image (g) Rerendered empty room image

Figure 4.5: Results of a multiroom dataset. (a) shows a top view of the input scan. (b) shows the reconstructed floorplan with default parameters, which only includes one room. (c) adjusts parameters such that doorways do not get detected as walls, but the thin wall on the side of the circled region of (a) still does not get reconstructed. With some manual adjustment of the circled region, the system arrives at a reasonable floorplan (d). Although this floorplan does not include the doorways, it still gives reasonable results when rerendered in (e) and (g); some incorrect shadow boundaries can be seen near the doorway in (g) due to the missing wall around the doorframe.

Chapter 5

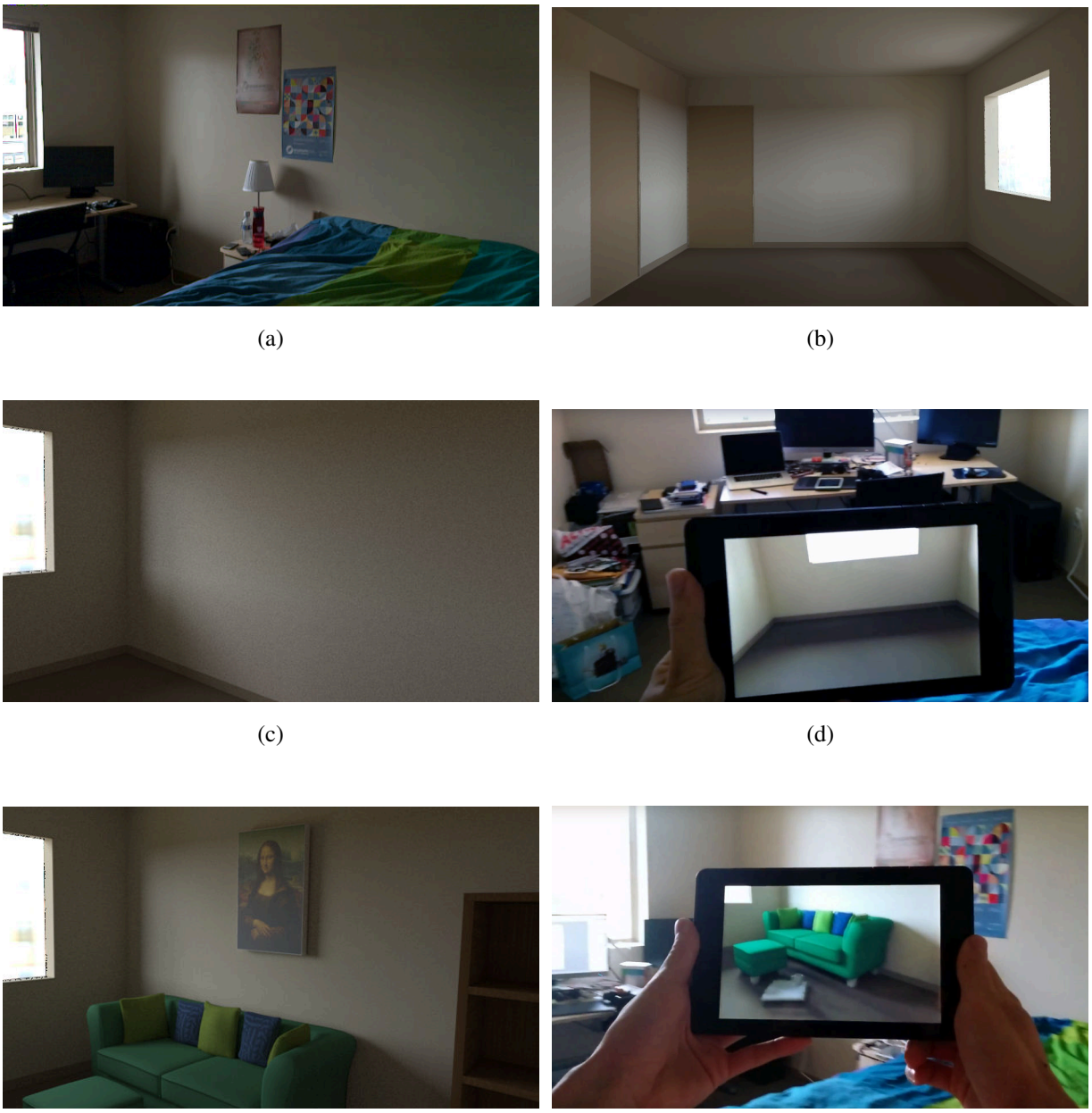
INFERRING LIGHTING AND MATERIALS FROM INDOOR SCANS

5.1 Introduction

Recovering the lighting and materials in a scene is vital to producing accurate renderings. However, unlike geometry, which can be directly captured using depth sensors, lighting and materials are much more difficult to infer. A sensor typically observes only the final appearance of a scene surface (specifically, the radiance coming from the surface). It is difficult to determine if, for example, a surface appears red because the object itself is red (material properties) or if it is illuminated with a red light (lighting properties). Fortunately, real-world lights and surfaces are not arbitrarily complex. Using parametric models of lights, combined with the assumption that many points in the scene share the same material properties, makes this inverse rendering problem much more tractable.

In this chapter I present a scene-scale inverse rendering framework for simultaneously inferring diffuse reflectances and lighting parameters[1]. In particular, in contrast to the limited lighting models used in previous work, the diverse lighting models I describe are crucial for accurately representing the lighting in the scene, ranging from expressive models of point and line emitters, to area lights and distant illumination. The inverse rendering system combined with these lighting models enables a variety of convincing reconstructions and edits visually consistent with the originally captured scenes, some examples of which are shown in Figure 5.1.

[1]This chapter describes work originally published in the ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia 2016)[155]



(a) (b)

(c) (d)

(e) (f)

Figure 5.1: From an RGBD scan of an indoor scene (one RGB frame shown top left), I produce a scene model of the empty room (top right), including lighting and materials. Users can visualize this empty room from the original camera scan locations (middle left) or in situ (middle right). The inferred scene model allows for realistic scene edits, such as refurnishing the room (bottom row).

5.2 Background and Related Work

5.2.1 Lighting Models

The incident light on an object is commonly approximated with the distant-scene assumption: the intensity of any incident ray on the object is dependent only on the direction of the ray, and not on the location on the surface. Thus illumination can be represented as a simple environment map, also known as Image-Based Lighting (introduced in [27]). This assumption has been used in most works that aim to insert synthetic objects into real scenes. The distant-scene assumption is insufficient for scene-scale mixed reality effects for two reasons: it does not capture spatial variation of lighting across the scene, and it does not allow the object to affect the appearance of the rest of the scene or participate in global illumination effects. Works such as [129] capture spatially-varying environment maps, and in general light-field capture methods such as [140] start to address the spatially-varying lighting issue. Cossairt et al. [24] solve the object-scene interaction issue by computing light-field transfer functions, but have specialized capture requirements and limited scale. Several works use uniformly emitting point lights in addition to environment map or directional lighting [125, 122, 136], but again do not consider global illumination.

5.2.2 Inverse Rendering

Physically accurate inverse rendering has a long history. However, very few works attempt to simultaneously recover lighting and reflectance parameters [103]. As mentioned above, most approaches work on the scale of a single object of interest, and do not consider scene-scale effects such as occlusion or global illumination, or only do so in the local region of the object (such as [27]). Shape-from-Shading (SfS) approaches, such as [142, 167], also involve inverse rendering problems; these methods use similar assumptions (distant illumination with no occlusion or inter-reflection) but simultaneously recover scene geometry as well as diffuse albedo and temporally-varying lighting. Ramamoorthi and Hanrahan [108] theoretically analyze the recovery of distant lighting and reflectance using a signal processing approach, and show when this simultaneous recovery is well-conditioned. Beyond distant illumination, several works recover point and direc-

tional light source positions simultaneously with reflectance parameters from multiple images of an object of interest [91, 147].

A few inverse rendering works do operate on the scene-scale. Yu et al. [152] derive specular parameters and spatially-varying diffuse reflectances in a global-illumination-aware fashion across an entire scene, but assume known lighting. Kawai et al. [65], using the radiosity framework, perform a nonlinear optimization over diffuse reflectance and light emitter intensity to minimize a perceptually-based cost function; for this purpose they treat scene appearance as a variable which can be eliminated, rather than optimizing to match observed appearance across the scene. In his thesis, Colburn [22] computes light intensities and both specular and diffuse material properties in indoor scenes represented with simplified planar proxies and view-dependent high dynamic range textures; however, the inaccurate geometry severely limits the accuracy of the solution. In addition, his system requires substantial user interaction. Lombardi and Nishino [84, 83] solve for a general BRDF and distant illumination using statistical and information-theoretical priors on a single object with known geometry. They later extend this work [83] to work on a set of RGBD images of a scene with spatially varying BRDFs; this is solved using an expensive gradient-based optimization where gradients are computed via path tracing, allowing for interreflections and occlusions. However, the assumption of distant illumination limits the extent of the scenes that this method can handle. Forsyth [34] models incident illumination on an object per point on the surface as coming from a single global direction but with spatially varying intensity; while nonphysical, this captures some of the effects of interreflection. Forsyth also incorporates a known shadow map to deal with occlusion.

Other recent works aim to compute scene models from single images [11, 63, 64, 5]. While Boivin and Gagalowicz [11] require known lighting, Karsch et al. [64] recover full scene models (including light emitters, diffuse albedo, and scene geometry), automatically locating in-scene diffuse area lights, and selecting out-of-scene illumination from beyond the field of view of the image from a database of environment maps. Karsch et al. use recent advances in intrinsic image decomposition to provide reflectances, and single image depth estimation to provide geometry. Barron and Malik [5] also take an intrinsic image decomposition approach, recovering spatially-varying il-

lumination, diffuse albedo, and geometry from an RGBD image by using soft segmentations of the input image; each illumination segment has an independent environment map, providing spatially-varying lighting that approximates occlusion and interreflection effects. These works based on intrinsic image decomposition rely primarily on natural image and smoothness priors (e.g. [6]) to compute albedo and thus do not give physically accurate reflectances; furthermore, they only create a scene model for a particular camera viewpoint and not an entire 3D scene.

5.3 Overview

Given scene geometry and linearized scene appearance as computed in Chapter 3, the next task is to compute the lighting and materials in the scene. I first describe a general inverse rendering formulation based on analyzing the incident direct and indirect light on vertices in the mesh using methods similar to radiosity-based formulations of inverse lighting [152, 11]. In order to use this formulation, I then describe several lighting models for the light emitters in the scene as well as the process of grouping scene surfaces into clusters with the same material properties. In this chapter I assume that scene surfaces are all diffuse, so that the material properties for a surface are simply a diffuse albedo. These material and light models allow the recovery of physically consistent values for the light emitter intensities and diffuse reflectances.

5.4 Inverse Rendering Formulation

Consider a single, non-emitting vertex V. The rendering equation over surfaces in the scene is [61]:

$$L(V \to x') = \int_S f_r(x \to V \to x') G(V, x) L(x \to V) dA \tag{5.1}$$

where $L(a \to b)$ is the radiance along the ray from a point a to a point b, $f_r(x \to V \to x')$ is the bidirectional reflectance distribution function (BRDF) at point V giving what proportion of light coming from the ray from x to V is reflected towards x', and $G(V, x)$ is the geometric visibility term between V and x including occlusion and projected area.

For a diffuse world, each vertex has a reflectance ρ_V where $f_r(x \to V \to x') = \frac{\rho_V}{\pi}$, and $L(V \to x') = L_V$ is independent of the viewing direction:

$$L_V = \int_S \frac{\rho_V}{\pi} G(V, x) L(x \to V) dA. \tag{5.2}$$

I then decompose the integral into indirect and direct lighting components, where $\mathcal{L}$ is the set of light emitting surfaces and each $l \in \mathcal{L}$ is one of these emitting surfaces:

$$\begin{aligned} L_V = &\frac{\rho_V}{\pi} \int_{S \notin S_{\mathcal{L}}} G(V, x) L(x \to V) dA \\ &+ \frac{\rho_V}{\pi} \sum_{l \in \mathcal{L}} \int_{S \in S_l} G(V, x) L(x \to V) dA \end{aligned}. \tag{5.3}$$

For brevity I refer to the incident indirect lighting term as

$$L_{V,\text{indirect}} = \int_{S \notin S_{\mathcal{L}}} G(V, x) L(x \to V) dA. \tag{5.4}$$

For the incident direct lighting, I parameterize each light emitter by a set of intensities $I_l = (I_{l0}, \ldots I_{ln})^T$ such that the incident direct light on a vertex V due to a light l is a linear combination of these intensities. In other words, each light emitter is composed of a set of basis lights. Thus,

$$\int_{S \in S_l} G(v, x) L(x \to v) dA = F_l(V) \cdot I_l \tag{5.5}$$

where $F_l(V) = (F_{l0}(V), \ldots F_{ln}(V))^T$, and $F_{li}(V)$ is the transfer coefficient of light l's ith basis light's radiance incident on vertex V (i.e. if basis light i had unit intensity, $F_{li}(V)$ would be the incident radiance on vertex V due to that basis light). I also include direct lighting due to infinitesimal light sources such as point and line emitters; although they do not have areas and thus are not included in S, the incident lighting can still be represented as $F_l(V) \cdot I_l$. Substituting Equations 5.4 and 5.5 into Equation 5.3,

$$L_V = \frac{\rho_V}{\pi} \left(L_{V,\text{indirect}} + \sum_l F_l(V) \cdot I_l \right). \tag{5.6}$$

(a) Wall Vertex (b) Environment Map (c) Hemicube-Weighted (d) Light Sources and Missing Data

Figure 5.2: This figure illustrates the use of the hemicube in computing incident lighting and other relevant form factors. Consider a single vertex (a); rendering the mesh into the front face of the cube and four half-side faces is visualized in (b). The pixel values are weighted based on the projected area of the hemicube pixel onto the surface (c). I also use the hemicube to compute other important information in (d): light form factors in the green channel, showing light index, and directions of unknown incident light (see Section 5.4.2) in the red channel, for which black regions are missing data.

I calculate the coefficients of basis lights $F_l(V)$ using the scene geometry, given the locations of the light emitters (described in more detail in Section 5.5). I compute the incident indirect light directly from the final scene appearance: using the hemicube algorithm [20], I rasterize the mesh onto the faces of a 500 by 500 pixel cube centered at V with the per-vertex median radiance values computed in Section 3.6. I compute the the total indirect irradiance at V as the sum of the pixel values weighted by the hemicube form factors[2]. Note that Monte Carlo sampling methods could also be used to compute indirect illumination; because of the diffuse assumption, the rasterization approach provides similar accuracy but is significantly more performant. This process is illustrated in Figure 5.2.

The outgoing radiance at vertex V is approximately $\bar{L}_V$, the weighted median of projected radiances described in Section 3.6. Thus, the unknowns are the ρ_V, I_l. To solve for these, I construct

[2]Note that using the hemicube form factors given in [20] actually results in an extra $\frac{1}{\pi}$ factor in the irradiance. For clarity, I use the standard hemicube weights multiplied by π.

a constrained nonlinear optimization

$$\arg\min_{\rho_V, I_l} \sum_V d\left(\bar{L}_V - \frac{\rho_V}{\pi}\left(L_{V,\text{indirect}} + \sum_l F_l(V)\cdot I_l\right)\right) \tag{5.7}$$

subject to $0 \leq \rho_V < 1$ (where $d(\cdot)$ is a loss function). I perform this optimization simultaneously over several sets of vertices that share diffuse reflectances ρ for robustness; determining these sets of vertices is based on the architectural features described in Chapter 4 and is expanded upon in Section 5.6. Once the lighting parameters I_l are computed, the reflectances for the remaining unclustered vertices can then be solved for directly using linear least squares. Additional regularization terms for different light types are described in Section 5.5.

I use Ceres Solver [2] to optimize Equation 5.7, using the Huber norm as a robust loss function. This norm helps handle outlier objects such as posters, rugs, and doors that are geometrically indistinguishable from the classified surfaces, but have differing reflectance parameters.

5.4.1 Lighting Ambiguity

Inverse rendering problems that do not consider indirect illumination have a scale ambiguity between diffuse reflectance and light intensity; if an optimal solution were (I, ρ) then $(kI, \rho/k)$ would also be an equivalent solution because the final appearance is $L_V = \frac{\rho}{\pi} F_l(V)\cdot I_l = \frac{\rho}{\pi} L_{\text{V,direct}}$.

Once indirect lighting and occlusion is taken into account, this ambiguity disappears: $L_V = \frac{\rho}{\pi}\left(L_{V,\text{indirect}} + L_{V,\text{direct}}\right)$, and $L_{V,\text{indirect}}$ is a known computed quantity.

However, if $L_{V,\text{direct}}$ is much greater than $L_{V,\text{indirect}}$, the ambiguity comes back into play. This is not uncommon in real world scenes; for example, many lights throw most of their light upwards, where it bounces off the ceiling and then illuminates the room indirectly. For points on the ceiling near the light, the amount of incident indirect light is negligible compared to the amount of incident direct light. This results in an unstable solution.

To deal with this problem, I reweight the optimization to favor data terms for vertices that have nonnegligible amounts of incident indirect light. Some of these vertices will likely be in the same material cluster as the vertices that receive too much direct light, but prevent the diffuse reflectance

estimate from varying too much.

The reweighting term I use is based on the ratio between the vertex exitant radiance and the incident indirect light on the vertex. If this ratio is $R = \frac{\bar{L}_V}{L_{\text{V,indirect}}}$, then the reweighting term is for some constant c ($c = 10$ in this work)

$$w = \begin{cases} 1, & R \leq 1 \\ \exp(c(1-R)), & R > 1 \end{cases}. \tag{5.8}$$

5.4.2 Missing Data

In the formulation above, I assume that the hemicube provides an accurate measurement of the direct and indirect lighting at a point. However, in real world captures, many vertices or regions of the mesh will be missing data – they will have no radiance estimates. For example, it is impossible to scan the top of a tall shelf or a light fixture.

To deal with this issue, I first determine what proportion of the hemicube consists of unobserved incident radiance values. I then assume that, on average, the incident indirect light coming from unobserved rays will be the same as the average indirect incident light from the rest of the scene. Thus, if p_0 is the proportion of the hemisphere consisting of unobserved radiances, and p_i is the proportion of the hemisphere consisting of indirect illumination, I rescale the indirect illumination as

$$L'_{\text{indirect}} = L_{\text{indirect}} \frac{p_0 + p_i}{p_i}. \tag{5.9}$$

5.5 Light Source Models

In this thesis, I group lights into one of the following categories: distant environment lights, diffuse area lights, generalized point lights, and generalized line lights.

As described in Section 5.4, each of these lights is parameterized as a linear combination of basis lights, rather than parametric models (such as the Phong-like spotlight). This simplifies the

structure of the problem (since now a single light can be treated as a set of independent lights) and keeps the cost function well-behaved (linear for fixed reflectances). In this chapter I assume that the number, location, and types of lights are specified by a user, although I propose several heuristics for identifying potential lights to simplify this process.

In this section, I describe the direct lighting computation at a vertex V for each light type. This includes describing how each light is decomposed into basis lights, computing the coefficients $F_{li}(V)$ for each basis light, and specifying the geometric properties of each light type (these properties are not explicitly optimized for, but are specified by the user). For conciseness, for any vector v, the normalized vector is represented as $\hat{v} = \frac{v}{||v||}$.

5.5.1 Diffuse Area Lights

Diffuse area lights are assumed to emit a constant radiance in every direction in the hemisphere and are thus simply parameterized by a single intensity (i.e. only one basis light).

To identify diffuse area lights, I threshold the vertex radiances on the mesh using a user-specified value, and find connected components of vertices with L_V above this threshold. The vertices in each component of more than 50 vertices are labelled as belonging to the same area light. Some minor user interaction allows the system to prune these candidate lights. Diffuse area lights are useful to model ceiling panel lights that are often found in office scenes.

To compute $F_l(V)$ for area light l, I simply render the hemicube using the per-vertex light labels as vertex colors, and then take the sum of the hemicube form factors that correspond to light l.

5.5.2 Distant Environment Lights

Distant environment lights come from an infinitely large sphere surrounding the scene. Rays which do not intersect the scene contribute direct illumination from the distant environment to the vertex, with a dependence only on the ray direction (and not the position of the vertex). These lights are used to represent outdoor illumination coming from a window, and approximates window illumination well assuming that no objects out the window are too near. In practice, windows are

hole-filled during the Poisson Surface Reconstruction of the scene geometry. Because of this, windows are identified concurrently with area lights during the thresholding process described above for the purposes of this chapter. The user interface for light source specification suggests that an area light is in fact a window if it lies near a wall and approximately forms a rectangular shape.

As a spherical function, distant illumination can be represented in many ways. I implemented two bases: a five-band spherical harmonic basis (with 25 light components), and a cubemapped environment map (piecewise constant basis). For the cubemap basis, I use 4×4 cubemaps, for a total of 96 component lights (i.e. 96 parameters I). While most distant illumination works use spherical harmonic bases, I found that the piecewise constant basis was more suitable to represent the lighting in most captured rooms. This is primarily because the spherical harmonic lights are ill-behaved at viewpoints that are not observed in the optimization (e.g. sharp upward angles out the window); the piecewise basis makes enforcing nonnegativity and smoothness at glancing angles much simpler.

To compute the set of $F_{li}(V)$ for distant illumination, I again use the hemicube. For hemicube pixels that correspond to distant illumination rays, I compute the orientation of the ray through the center of the pixel, determine which cubemap cell j that direction corresponds to, and then increment $F_{lj}(V)$ by the hemicube pixel form factor.

For piecewise constant basis distant illumination, I constrain light intensities to be nonnegative. I also add a weak smoothness term between adjacent cells of the cubemap. This ensures that cubemap cells that are not incident on any vertices during optimization will still have reasonable values. This is important because these directions may be sampled when rerendering the scene. If $A = \{(i,j)|i,j \text{ are adjacent cubemap cells}\}$, then I add a set of regularization terms to the cost function ($\lambda_{\text{smooth}} = 1 \times 10^{-5}$):

$$\lambda_{\text{smooth}} \sum_{(i,j)\in A} (I_{li} - I_{lj})^2.$$

An example of the solved distant lighting is shown in Figure 5.3.

(a) Input Frame (b) Synthetic Empty Room Image (c) Environment Map (d) Wall, floor, and ceiling reflectance

Figure 5.3: Recovered distant illumination parameters. (a) is an input image from the capture sequence. (b) shows a rerendered version with my distant lighting model. (c) shows a (θ, ϕ) parameterization of the recovered cubemap; note that the central region corresponds to the directions out the window. While this approximation does not appear representative of the actual appearance out the window and only imprecisely matches shadow boundaries, it captures most of the illumination effects that affect the indoor scene. (d) shows the recovered wall, floor, and ceiling reflectances.In the actual scene, the wall and ceiling have identical properties; the solved reflectances are very similar even though the system did not constrain them to be equal.

5.5.3 Generalized Point Lights

Many light manufacturers describe the intensity distribution of their lights using an IES (Illumination Engineering Society) lighting profile, as specified in [123]. Light intensity emitted from a point source $I(\phi, \theta)$ is a spherical function, and the IES format specifies the values of this function at a discrete set of exitant angles (much like our piecewise constant representation of distant lighting). The IES format also builds in strong symmetry assumptions, most commonly assuming radially symmetric light distributions. These profiles are used to specify many types of lights found in residential indoor scenes, such as lamps with shades, standing lights, recessed ceiling lights, and general spotlights, and are commonly used in architectural visualization. Thus I follow the example of the IES and model point lights as radially symmetric emitters.

Such point lights have two geometric parameters: a 3D location P, and an axis of symmetry $\hat{v}$. I default to a vertical axis of symmetry $\hat{v} = (0, 1, 0)$, since this is how most radially symmetric

lights in IES formats are used. The results in this chapter require the user to specify the positions of point lights; however in Chapter 6 I describe methods to automatically determine point light locations.

A radially symmetric distribution means the spherical function $I(\phi, \theta)$ (centered around $\hat{v}$) is independent of ϕ, the azimuthal angle. To represent this 1D function in a linear basis, either the Fourier basis or a discretized basis can be used (analogous to the two representations of environment lighting). In the results shown in this chapter, I use a 32-bin discretization of angle θ, the inclination angle.

As these point lights are infinitesimal, their contribution to the direct lighting cannot be determined using the hemicube. Therefore I separately compute the incident direct lighting on every vertex due to these point lights. The incident lighting basis coefficient due to a point light using the piecewise constant parameterization is

$$F_{li}(V) = G(V, P)\frac{(\hat{a} \cdot n_V)}{||a||^2} \tag{5.10}$$

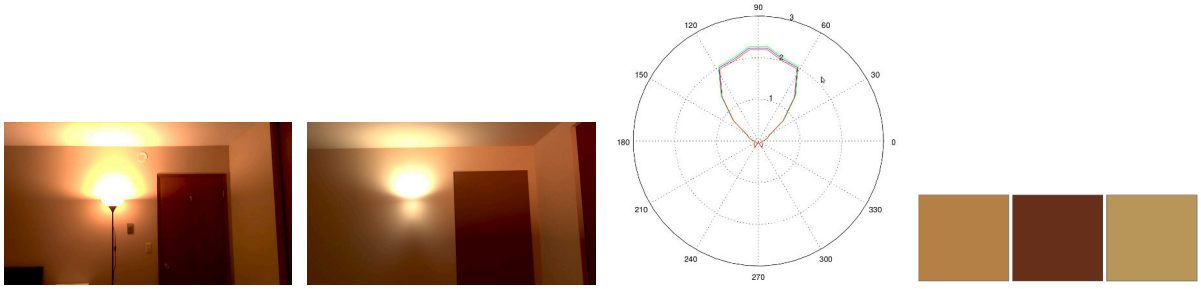
where $a = P - V$, n_V is the surface normal at V, and $G(V, P)$ is a visibility term. The visibility term is computed by checking if a ray between V and P intersects the scene geometry. i corresponds to the angle subdivision cell for direction $\theta = \cos^{-1}(\hat{a} \cdot \hat{v})$ (and $F_{lj}(V) = 0$ if $j \neq i$).

Constraints and regularization for point lights are analogous to distant lighting (nonnegativity and smoothness). However, I found that adding the additional constraint of monochromaticity created better results. To achieve this, I give each point light a three-component color, and let the intensity of this color vary around the light with the (now single-channel) basis light intensities,

$$\left(I_l^R(\theta), I_l^G(\theta), I_l^B(\theta)\right) = (r_l, g_l, b_l)\, I_l(\theta),$$

giving the light three additional parameters r_l, g_l, b_l. I then add a term strongly ($\lambda_{\text{monochrome}} = 1 \times 10^8$) enforcing the color to have unit norm:

$$\lambda_{\text{monochrome}}(r_l^2 + g_l^2 + b_l^2 - 1).$$



(a) Input Frame (b) Synthetic Empty Room Image (c) Point Light Distribution (d) Wall, floor, and ceiling reflectance

Figure 5.4: Recovered point light parameters. (a) is an input image from the capture sequence. (b) shows a rerendered version with my point light model. (c) shows a goniometric diagram of the recovered point light distribution using a piecewise constant basis. In rendered images, the distribution is interpolated using a Catmull-Rom spline. (d) shows the recovered wall, floor, and ceiling reflectances. In the actual scene, the wall and ceiling have identical properties; the solved reflectances are very similar even though the system did not assume they were equal. Note that this scene is also illuminated by a window.

The monochromaticity constraint couples the color channels, so the optimization actually minimizes data terms for all three channels simultaneously when monochromatic lights are included.

An example of the solved point lighting is shown in Figure 5.4.

5.5.4 Generalized Line Lights

Fluorescent tube lights are commonly encountered in non-residential indoor scenes. While uniformly emitting line lights have been analyzed in the literature, they are unsuitable for modelling these tube lights. These lights often have baffles that change the directional variation of light intensity.

Figure 5.5 illustrates how I parameterize line lights. Geometrically, generalized linear lights have the following properties: an endpoint P, a direction vector d (so that the line extends from P to $P + d$), and a perpendicular vector $\hat{d}_{\perp}$. The light distribution of the line light varies around the

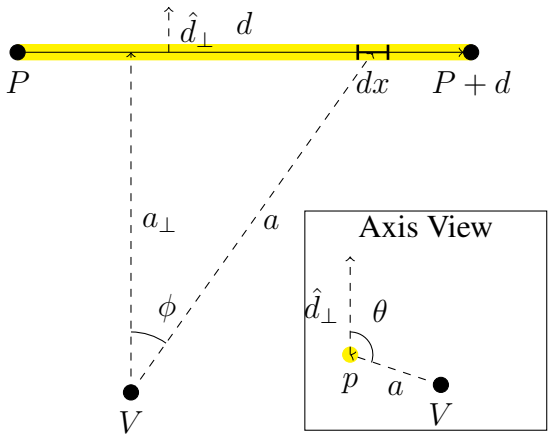


Figure 5.5: Side view of line light, perpendicular to axis d of light. Inset shows view along the axis d of the light

light with θ, but does not vary in the direction parallel to d.

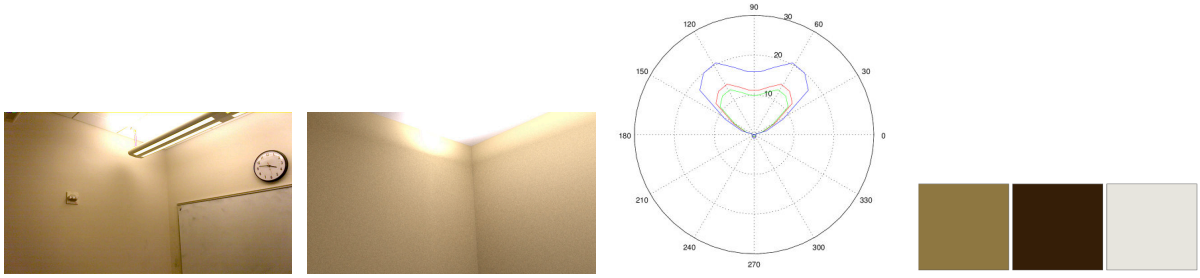
Analogously to environment maps and point lights, line lights can use the Fourier basis or a constant basis based on discretized θ as component lights. In this thesis, as for distant and point illumination, I use the piecewise constant local basis, uniformly dividing the circle of directions into 32 subdivisions.

The equation for the incident lighting basis coefficient at a vertex V due to a line light is given by integrating over the length of the light; each differential line segment is treated similarly to a point light, except with an extra term accounting for the projected length of the segment onto V:

$$F_{li}(V) = \int_0^{||d||} G(V, P + x\hat{d}) \frac{n_V \cdot \hat{a}}{||a||^2} (\hat{a} \cdot \hat{a}_\perp) dx \tag{5.11}$$

where $a = (P + x\hat{d}) - V$, the vector from V to the differential line segment dx, $a_\perp = a - (a \cdot \hat{d})\hat{d}$ is the component of a perpendicular to the light, and $G(V, P + x\hat{d})$ is the visibility term as described for point lights. The $(\hat{a} \cdot \hat{a}_\perp)dx = \cos(\phi)dx$ term represents the projected length of the segment. Figure 5.5 visually shows these quantities relative to the light and V.

While ϕ varies along the length of the light, θ is constant for a particular vertex V relative to any point along the light. Since I use a piecewise constant basis, each vertex will thus only have



(a) Input Frame (b) Synthetic Empty Room Image (c) Line Light Distribution (d) Wall, floor, and ceiling reflectance

Figure 5.6: Recovered line light parameters. (a) is an input image from the capture sequence. (b) shows a rerendered version with our line light model. (c) shows a goniometric diagram of the recovered line light distribution using a piecewise constant basis. In rendered images, the distribution is interpolated using a Catmull-Rom spline. (d) shows the recovered wall, floor, and ceiling reflectances.

one of the basis light directions incident on it, so only one of the F_{li} will be nonzero. I compute which i this is by finding $\theta = \cos^{-1}(\hat{a}_\perp \cdot \hat{d}_\perp)$. Note that this expression for θ conveniently enforces symmetry around $\hat{d}_\perp$, which is common for real world line lights (where $\hat{d}_\perp$ is the world up vector). However, for full generality, θ would range from 0 to 2π.

The integral in Equation 5.11 could be analytically solved in the absence of occlusion; however the visibility term makes this considerably more difficult. I discretize the line light into N segments for the purposes of computing occlusion, so this becomes

$$F_{li}(V) = \sum_{k=1}^{N} G(V, P + (k - 0.5)\hat{d}) \frac{n_V \cdot \hat{a}}{||a||^2} (\hat{a} \cdot \hat{a}_\perp) \frac{||d||}{N}. \tag{5.12}$$

Generally RGBD scans of fluorescent tube lights will pick up the surface of the light (or at least its housing and reflectors). Thus, my system can identify line lights by looking for mesh components that are located near the ceiling and determining if they are long and thin. I do this by examining the eigenvectors and eigenvalues of the matrix of coordinates of the vertices for each mesh component near the ceiling. If the second and third eigenvalues are small relative to

the first, then the component is likely to be a line light, for which the largest eigenvector provides $\frac{d}{||d||}$. I determine the extent of the line light by transforming the vertex coordinates into the basis formed by the eigenvectors and finding the maximum and minimum coefficients for the principal eigenvector. The perpendicular vector $\hat{d}_\perp$ is set to be the world up vector (more precisely, the component of the up vector perpendicular to d). Other non-axis-aligned line lights must be added manually.

Regularization for line lights is analogous to the regularization on point lights: nonnegativity and smoothness between adjacent cells in the constant basis, and monochromaticity across the light.

An example of solved line lighting is shown in Figure 5.6.

5.6 Material Clustering

The tractability of Equation 5.7 depends on many vertices sharing only a few different reflectance values ρ_V, as otherwise each term in the optimization (one per vertex) would have an independent unknown (the reflectance of that vertex). Grouping vertices together into material clusters is therefore an important step in the inverse rendering process.

However, note that the optimization does not need to include every vertex in the mesh – it only needs to contain enough information to solve for the lighting parameters I_l. Once the lighting parameters are known, the reflectances of the remaining unclustered vertices can be solved for directly using linear least squares on Equation 5.7.

The generalized lighting models described in Section 5.5 consist of sources that vary their intensity based on direction. Therefore, in order to accurately solve for their lighting parameters, the vertices for the terms in the optimization would ideally cover many different parts of the scene. For indoor scenes, the walls form an ideal material cluster for this purpose: frequently, the walls in a scene have a consistent diffuse color and enclose almost the entire scene. This is similarly true for the ceiling and frequently the floor.

The per-vertex semantic labels obtained from the process in Chapter 4 provide a clustering that is suitable for Equation 5.7. In practice, I used only the wall, floor, and ceiling clusters to

compute lighting and used the resulting lighting to directly solve for the reflectances of doors and baseboards.

5.6.1 Textured Surfaces

For rooms with patterned wallpaper, stucco ceilings, and carpets or patterned floors, the assumption that the vertices share a single reflectance appears to be violated. This framework can in fact handle such surfaces if the scale of such the textural variation is smaller than the resolution of the mesh.

In these cases, my system can compute the average reflectance over the entire texture. During the projection operation of the appearance reconstruction process (from Chapter 3), instead of storing a single pixel value per vertex per input image, I use a sampling kernel to compute an average over several pixels. The size of this sampling kernel is scaled based on the distance from the camera to the vertex as well as the mesh resolution. This means that the vertex color $\bar{L}_V$ now represents the average texture color (rather than a point sample). However, if the scale of the texture results in variation much larger than the mesh resolution (e.g. a checkerboard with large squares) the optimization may attribute that variation to lighting, for example by changing the intensity of a single basis light.

5.7 Results and Discussion

Putting together the capture system described in Chapter 3, the floorplan estimation system described in Chapter 4, and the inverse rendering system described in this chapter, an indoor scene can be virtually refurnished.

To demonstrate the abilities of this system, I scanned four scenes with varying lighting conditions, including three different types of emitters: distant illumination, generalized point lights, and generalized line lights. Results from these scenes, including the relit empty room as well as virtually refurnished rooms, are shown in Figures 5.1 and 5.7. All renderings are performed with PBRT [104] for physically accurate global illumination effects. For scenes with distant illumination through windows, the scene is rendered using the recovered environment map as a light source.

Figure 5.7: Results from the described system in four scenes. The first row is an unadjusted frame from the original input. The second row shows a synthetic rendering of the empty room from the same camera viewpoint and the same exposure. The third row inserts some synthetic objects into the scene. The fourth and fifth rows show wider views of the entire relit mesh with baked global illumination at a suitable global exposure.

For the final image, an environment map derived from the original input images is composited into the scene to render what is seen through the window.

The unoptimized runtimes of various components in the entire system running on a 3.4GHz Intel i7 processor are as follows: ray-traced image reprojection, which is used in several places in this pipeline, takes approximately 5 minutes for 1000-1200 images at 640×360 resolution and

a mesh of approximately 200,000 vertices. The optimization process for radiometric calibration takes approximately 15 minutes for these images. Floor plan recovery takes about 5 seconds. The inverse lighting optimization process ranges from 15 to 30 minutes depending on the light types present in the scene.

As mentioned in Section 5.5, the inverse rendering system requires some user assistance to specify light source positions. For point lights in Bedroom 2 and Play Room, light coordinates were manually specified. The axis of symmetry was left as the default world up vector. In the Bedroom 1 scene, regions of high specularity on the desk that reflected the window light were labeled as light emitters based on the threshold. These were manually pruned. Similarly, in Bedroom 2, bright regions near the point light were pruned from being light sources. The Study Room's line light was automatically detected and modeled. The geometry of the corner windows in the Play Room was not captured by the sensor and subsequent Poisson Surface Reconstruction. Thus the geometries of these windows were manually adjusted. Automating these steps is an area for future work.

In addition to adding virtual objects to a scene, the recovered, decluttered scene model also enables other types of edits. This includes changing the illumination conditions in the scene and changing the material properties of surfaces in the scene (Figure 5.8).

Altogether, the system described over the previous three chapters enables interactive mixed reality applications, where a user can actually walk through the scene and visualize scene edits in situ (Figure 5.1). The Tango tablet provides its position and orientation in the scene. Simply rendering the view of the empty room or room filled with new furniture as it would look like from that vantage point provides a compelling mixed reality experience. The scene is rendered as a static mesh with baked global illumination computed using PBRT, while the Project Tango software ensures that the camera pose is correct relative to the original scene.

5.7.1 Limitations

The above results show that this system handles a number of scenes and leads to convincing rerendered results. However, this work has several limitations that suggest future directions for research.

Figure 5.8: This figure shows how the reconstructed scene model enables edits to the lighting and materials of the scene. These edits are shown in increments; the first column shows the input frame, while the second shows the refurnished room. The third column shows the walls repainted a different color. The fourth column changes the lighting conditions, such as changing the time of day to sunset (top), or moving the point light illuminating the room (bottom).

Firstly, the framework as described assumes diffuse surfaces. It is fairly straightforward, albeit computationally intensive, to extend this framework into non-diffuse surfaces by rendering incident light hemicubes using view-dependent texture mapping, and optimizing over each observation of each vertex rather than over the per-vertex weighted median. However, preliminary experiments suggest that significantly more complete data captures, with views of surfaces from many angles, are necessary for this optimization to accurately recover specular properties.

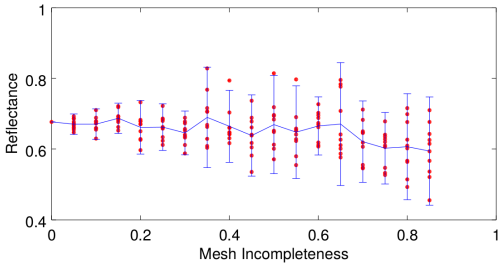


Figure 5.9: Reflectance estimates in the Bedroom 1 dataset as a function of mesh incompleteness, with 95% confidence intervals.

Having an accurate estimate of the indirect light in a scene is important to the inverse rendering framework. Incomplete scans reduce the accuracy of the estimate. To determine the importance of having complete scans, I conducted an experiment where I artificially removed sections of one of the datasets and examined the effects on the resulting reflectance estimates.

I deleted random connected regions of the mesh, each comprising 5% of the total triangle count, by selecting a random triangle and deleting adjacent triangles in a breadth-first fashion. For each increment of mesh incompleteness between 0% and 90%, I generated 10 meshes randomly ablated in this fashion. I then ran the inverse rendering pipeline on each mesh, and plotted the resulting reflectance estimates in Figure 5.9. The variance in reflectance estimates rises dramatically after about 20% of the mesh is deleted. There also appears to be a slight bias toward lower reflectances as mesh incompleteness increases; this bias is likely to be scene-dependent as a result of the averaging scheme to fill in missing data (see Section 5.4.2).

Thus far, as described this system explicitly recovers reflectances of only wall, floor, and ceiling vertices simultaneously with lighting. One obvious next step would be to compute reflectances of all surfaces in the scene using the solved lighting, which would allow users to leave some of the original room contents present in rerenderings. While per-vertex reflectances on the original mesh (Equation 5.7) can be directly recovered using the solved lighting parameters (shown in Figure 5.10), the resulting reflectances are often inconsistent or nonphysical ($\rho_V > 1$). This is primarily due to the relatively low resolution and accuracy of the input mesh and camera poses, resulting in incorrect occlusion boundaries, incorrect normals, and subsequent incorrect calculations of incident lighting, as shown in Figure 5.10. Modest mesh quality and inaccurate poses also create artifacts in the HDR diffuse scene appearance. Furthermore, the low resolution of the input data results in unconvincing rerenderings. Combining existing SLAM algorithms with better hardware to handle typical indoor scenes more robustly will result in higher quality meshes, which would potentially mitigate this problem. Alternatively, rather than directly rendering the scene model and scanned geometry, which will always have some inaccuracies, neural rerendering methods can bypass the need for high quality meshes and accurate scene models. I describe one such method in Chapter 7.

Figure 5.10: I compute a "reflectance mesh" by taking the radiances from the input mesh (inset) and the solved lighting, and compute a per-vertex reflectance as radiance divided by incident light.
One would expect that the walls will have a uniform reflectance, but some artifacts are visible. The red circles show regions where the inaccurate wall geometry results in inaccurate normals. The blue circles show regions where the inaccurate scene geometry results in incorrectly projected scene radiance. The yellow circles show regions where the inaccurate scene geometry results in incorrect occlusion boundaries. The green circles show artifacts from the low resolution of the lighting. The large white spot on the desk (circled in purple) is due to the specularity of the desk; the radiance given by the input mesh is an overestimate of the diffuse appearance of the desk, and thus the computed reflectance is also too high. Although not an issue in this case, this specularity can also cause an overestimate in the the incident indirect illumination on other points in the scene.

While the recovered lighting matches the original appearance of the scene, it may not represent the true emitters in the scene accurately. However, [63, 109] present evidence that a range of lighting conditions are visually similar to humans, suggesting that rough lighting estimates are enough for many applications. Furthermore, Ramamoorthi and Hanrahan [108] determine that it is impossible to recover true high-frequency lighting from diffuse object appearance.

There are also some noticeable mismatches in color between rerendered results and the original

input. Much of this can be attributed to the aforementioned problems with mesh quality. Explicitly solving for a camera response function might improve the results as well. Furthermore, there may be artifacts introduced by the per-channel decomposition of the problem, which does not account for spectral effects.

Automatically determining the locations and classification of light sources is perhaps the most interesting problem. It is difficult to directly optimize for light source positions because occlusion boundaries cause discontinuities in the solution space. It is especially challenging to compute the properties of unobserved light sources, or to identify e.g. that the reflection of an unseen light source in a mirror is not itself an emitter. While machine learning methods for light source identification and classification can in many cases provide compelling results, it is difficult for them to provide light sources that are physically consistent with the scene appearance. Instead, in the following chapter, I lay out a physically-based method for estimating the position and number of light sources in a scene.

Chapter 6

LIGHT LOCALIZATION

6.1 Introduction

Estimating the lighting in a scene has a long history in computer vision and graphics, and is a key subproblem in many areas such as reconstruction from shading cues (e.g. [53, 74, 167]), where inferred shape depends heavily on the incident lighting, or augmented reality (e.g. [27]), where virtual objects must be lit consistently with the real world to appear convincing. The majority of prior work assumes *distant illumination*, where lighting incident on a surface does not vary spatially and only depends on the normal of the illuminated surface. Distant illumination, however, is insufficient to represent the lighting in many environments, such as indoor scenes, which are commonly lit by a small number of local light emitters.

Compared with distant illumination, inferring discrete, local lighting models introduces two new unknowns: *cardinality* (the number of lights) and *position* (where the lights are), in addition to *intensity*. I showed in Chapter 5 that solving for intensity was a straightforward optimization problem. However, optimizing for positions is highly discontinuous and inefficient because of occlusion, and so I assumed that light cardinality and positions were known, or estimated using heuristics. This raises the question: Can light cardinality and position be determined, even in the ideal case? More concretely, given all the information about the scene except the lighting (i.e. given geometry, materials, and final lit appearance), the **light localization problem** is to completely recover the number of lights, their positions, and their intensities.

In this chapter, I introduce a method for solving the light localization problem[1]. The key contributions are: (1) a formulation of the problem for an unknown number of local, discrete

[1]This chapter describes work originally published at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018 [156]

emitters; (2) a novel scene transform that proposes multiple candidate light positions based on the reflected light in the scene; and, (3) an iterative algorithm that uses the light proposals to recover the full set of scene illuminants, including positions and intensities. I will show that this method works on both synthetic and real-world scenes; though the solution can be inherently ambiguous (e.g., two very close lights may be recovered as a single light), this approach is generally able to arrive at a low-error solution.

6.2 Related Work

The distant illumination model introduced by Debevec [27] is the most widely used lighting model in inverse problems. In this section, I examine prior light estimation works that go beyond distant illumination, with methods to estimate light cardinality and position.

Several works attempt to decompose distant illumination into the combination of a discrete number of directional light sources. Earlier works identify occlusion boundaries, referred to as critical points, on a diffuse sphere [149, 161]. Wang and Samaras [134] examine the regions within the boundaries, avoiding the issue of detecting critical points, and subsequently [135] extend this framework for arbitrary reference objects. Lopez-Moreno et al [85] iteratively add directional sources based on reconstruction error. My algorithm performs a similar iterative light discovery process. However, it does not rely on the presence of critical points and thus is more robust to missing data.

Many works relax the distant illumination assumption by extending the parameterization of each directional light to include a distance [48, 127]. This direction-distance formulation, however, remains an object-centric approach, and lacks spatial variation.

A few works directly optimize the location of a single point source in a scene, without the direction-distance formulation, via a least-squares approach [13, 56, 101] These methods avoid considering occlusions, as hard shadows result in a discontinuous objective function.

Instead of optimizing for the lighting across all observed surfaces, another approach for locating light emitters is to look at strong, discrete lighting effects, such as shadow boundaries, specularities, and symmetries. These methods are often very accurate, but rely on the presence of appropriate

conditions in the scene and will not work in general cases. The most common approach is to use specular surfaces (usually light probes), identifying maxima as light sources and triangulating corresponding specularities [1, 60, 72, 77, 80, 106]. Triangulation of specularities can also be used to recover the shapes of area light sources [117, 164]. All of these works rely on both the presence of strongly specular surfaces (manually inserted into the scene or already present in the scene) and on the visibility and distinctness of light sources in the specular surface. Triangulation can also be used when light sources are directly imaged from multiple views [31, 35]. These methods assume that the light sources will be directly visible from the camera viewpoints.

Cast shadows are another strong lighting cue, and have been widely used in distant illumination environments [69, 80, 100, 115], but have also been used to identify near point sources [126] via triangulation. For triangulation-based methods, there must not only exist strong shadow edges in the scene, but they must also be identified and associated with object boundaries.

Other works [102, 131] are able to recover the position of a single point source with varying intensity distribution by analyzing symmetries of the light cast on planes. Although these assume that planar surfaces exist near the light source, this assumption is reasonable in most indoor scenes. Like these methods, my candidate light selection process also analyzes the distribution of direct illumination to directly extract light positions without optimization; however, illuminated surfaces in my setup can have arbitrary shape.

6.3 Candidate Light Proposal

6.3.1 Formalizing the Light Localization Problem

Consider a scene with known geometry G, lit by a set of unobserved isotropic point lights $L = \{(p_i, I_i)\}$, with positions p_i and intensities I_i. Assume that, at each point in the scene, the BRDF $f(x, \omega_i, \omega_o)$ specifying the proportion of the incident light from direction ω_i scattered in direction ω_o is known. A set of observations $Q = \{(x, \omega)\}$ of points in the scene are made; each of these observations $q \in Q$ measures the outgoing radiance in a direction ω from a point in the scene x due to reflected direct illumination from L. Let these radiance measurements be $B(q)$; $B(q)$ is thus a

surface light field [141]. The **light localization** problem is as follows: Given G, $f(x, \omega_i, \omega_o)$, and $B(q)$, completely recover L.

Note that, while observed radiances will normally include reflected direct and indirect illumination, $B(q)$ containing only directly reflected light can be computed by using the observations themselves to estimate and remove the indirect illumination. That indirect illumination can be directly removed may be surprising. In fact, the observations themselves are rays that comprise the indirect illumination to the rest of the scene. The observed radiances can thus be used to to estimate the indirect incident illumination on the rest of the scene. Practically, if the scene is largely diffuse and the origins of the observed rays cover most of the scene, then a good estimate of the indirect illumination incident on each point in Q can be computed. Then, given the BRDF at each point in the scene, the portion of reflected light due to this indirect illumination can be determined and subtracted from the observation to leave the desired $B(q)$. This process is similar to the inverse illumination process described in Chapter 5, except with known BRDFs.

6.3.2 Deriving the Intensity-Distance Function

At a point x on the scene surface, the outgoing radiance in direction ω due to a single, unoccluded, isotropic point light at point p with intensity I is

$$\begin{aligned} B(x, \omega) &= I \frac{V(x,p) f(x, \hat{l}, \omega)(n(x) \cdot \hat{l})_+}{||l||^2} \\ &= I \frac{V(x,p) f(x, \frac{p-x}{||p-x||}, \omega)(n(x) \cdot (p-x))_+}{||p-x||^3} \end{aligned} \tag{6.1}$$

by expanding $l = p - x$, $\hat{l} = \frac{l}{||l||}$. Here, the visibility term $V(x, y)$ is 0 if x and y are mutually occluded and 1 otherwise, $n(x)$ is the normal at x, and $(m)_+ = \max(m, 0)$.

Let the shading function $S(q, p)$ be

$$S(q,p) \equiv \frac{V(x,p) f(x, \frac{p-x}{||p-x||}, \omega)(n(x) \cdot (p-x))_+}{||p-x||^3} \tag{6.2}$$

such that $B(q) = I\, S(q, p)$ for the single light case.

For any possible light position p in a single-light scene, what intensity would a point light at that position have, given the information about observation q? This intensity can be computed as

$$I(q,p) \equiv \frac{B(q)}{S(q,p)}. \tag{6.3}$$

Each observation q induces such a function $I(q,p)$. An example of $I(q,p)$ for the diffuse case is shown in Figure 6.1a, and for a specular case in Figure 6.1b. Note that if the true light parameters were (p_1, I_1), then by construction, for every observation q, $I(q,p)$ will be equal to the true light intensity at the true light location p_1: $\forall q, I(q,p_1) = I_1$.

In the case of a multi-light scene,

$$B(q) = \sum_{i=1}^{|L|} I_i S(q,p_i) \tag{6.4}$$

Expanding and rearranging for I_1,

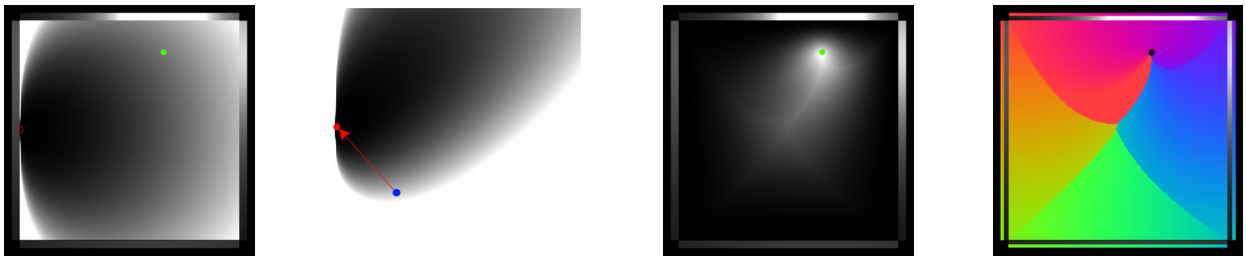
$$\begin{aligned} B(q) &= I_1 S(q,p_1) + \sum_{i=2}^{|L|} I_i S(q,p_i) \\ I_1 &= \frac{B(q)}{S(q,p_1)} - \sum_{i=2}^{|L|} I_i \frac{S(q,p_i)}{S(q,p_1)} \\ &= I(q,p_1) - \sum_{i=2}^{|L|} I_i \frac{S(q,p_i)}{S(q,p_1)} \end{aligned} \tag{6.5}$$

Both $S(q,p)$ and I_i are nonnegative, so $I_1 \leq I(q,p_1)$. Thus, the $I(q,p)$ derived from the single-light case is an upper bound on the brightness of a light at point p, based on the observation q. Put another way, no light at point p could be brighter than $I(q,p)$, since otherwise the observed brightness at point q would have been greater than $B(q)$.

Every observation q induces such an upper bound, so taking the most restrictive bound gives $I_1 \leq \min_q I(q,p_1)$. Now define $\mathcal{D}(p)$ to be

$$\mathcal{D}(p) \equiv \min_q I(q,p) \tag{6.6}$$

implying a light at point p can be no brighter than $\mathcal{D}(p)$ given all the observations of the scene.



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

Figure 6.1: Derivation of the Intensity-Distance Field in a one-light scene. A diffuse box-shaped room in 2D is lit by a single light at the green circle. The lit appearance (i.e. $B(q)$) on each wall is shown as a 1D intensity plot. As input, the intensity plots are known but not the location of the light. 6.1a shows $I((x_0, \omega), p)$ for p varying across the interior of the scene, where x_0 is the surface in the red circle (note that $I((x_0, \omega), p)$ is independent of ω for diffuse surfaces). For comparison, 6.1b shows $I((x_0, \omega), p)$ for a microfacet BRDF if x_0 were a shiny surface, with ω pointing along the red arrow. 6.1c shows $\mathcal{D}(p)$. 6.1d shows $\mathcal{L}(p)$, where the color at each point in the scene maps to the single point on the scene boundary bounding $\mathcal{D}(p)$ (mapping shown adjacent to intensity plots).

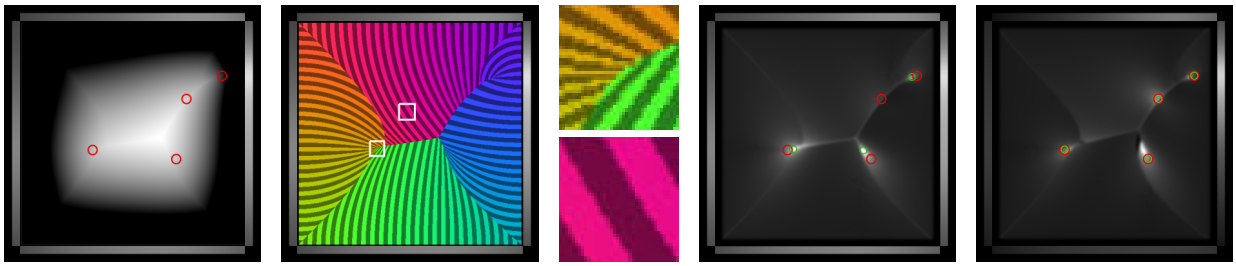
Since $I(q, p)$ is informally a distance function between points on the scene surface and points inside the scene, taking the minimum distance to the scene boundary results in a distance field, which I have termed the **Intensity-Distance Field** (IDF), given by $\mathcal{D}(p)$. An example of the Intensity-Distance Field for a single-light case in a diffuse scene is shown in Figure 6.1c.

6.3.3 Limiter Field

A related function for analyzing the IDF is the **Limiter Field**, which denotes which surface point(s) induced the value of $\mathcal{D}(p)$:

$$\mathcal{L}(p) \equiv \pi_x(\text{argmin}_q\, I(q, p)). \tag{6.7}$$

Here, $\pi_x(q)$ extracts the position of the observed point from q. Thus, $\mathcal{L}(p)$ represents which surface most strictly limits the brightness of a potential point light at p. To visualize $\mathcal{L}(p)$ in 2D, I have



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

Figure 6.2: A set of four isotropic point lights (red) lights a diffuse 2D scene. Given only the lighting on the surfaces of the scene shown along the edges of the square, the two functions $\mathcal{D}(p)$ (a) and $\mathcal{L}(p)$ (b) can be computed. A closeup of $\mathcal{L}(p)$ near a light source (c, top) shows about 25 unique colors in a neighborhood, i.e. unique values of $\mathcal{L}(p)$, while away from light sources (c, bottom) only 5 different values exist in the neighborhood. $\mathcal{V}(p)$ plots these counts (d), the peaks of which form the initial estimates for the number of lights and their positions (three green circles). Iterating the refinement algorithm recomputes the $\mathcal{V}(p)$ (e) and reveals the fourth light. The updated estimates then initialize a nonlinear optimization, which successfully recovers the original lighting parameters.

assigned each surface point a different color according to a linear gradient, and then mapped each p in the scene to the color corresponding to $\mathcal{L}(p)$ (Figure 6.1d). Examining $\mathcal{D}(p)$ and $\mathcal{L}(p)$ for different scene geometries and light configurations suggests that the medial axis of the distance field is often correlated with light positions. The medial axis is defined as the set of points for which $\mathcal{L}(p)$ has multiple values (visually, discontinuities in $\mathcal{L}(p)$).

6.3.4 Voting Function

Finally, to identify candidate light positions, $\mathcal{L}(p)$ can be treated as a discrete-valued function (since a finite number of surface radiances are measured at a finite number of points). This allows the construction of a voting measure for the light positions. The intuition here is that some surfaces will primarily be lit by a single light, and will thus vote consistently on the position of that particular light. Due to quadratic distance falloff, it is likely that there exists some such set of surfaces for at

least one light in the scene (unless the scene lighting is inherently ambiguous, as described in the introduction).

Figure 6.2b shows a refined visualization highlighting the discreteness of $\mathcal{L}(p)$, essentially a Voronoi diagram based on $\mathcal{D}(p)$. The number of surfaces voting for a particular location is given by the number of unique values that $\mathcal{L}(p)$ takes in a small neighborhood (Figure 6.2c). The surfaces corresponding to these values all have similar $I(q,p)$ intensities in the neighborhood, and so are effectively voting for the same intensity of a light at p. The count of locally unique $\mathcal{L}(p)$ values gives the Voting Function (shown in Figure 6.2d): for some neighborhood size δ,

$$\mathcal{V}(p) \equiv |\{\mathcal{L}(p'), ||p - p'|| < \delta\}| \tag{6.8}$$

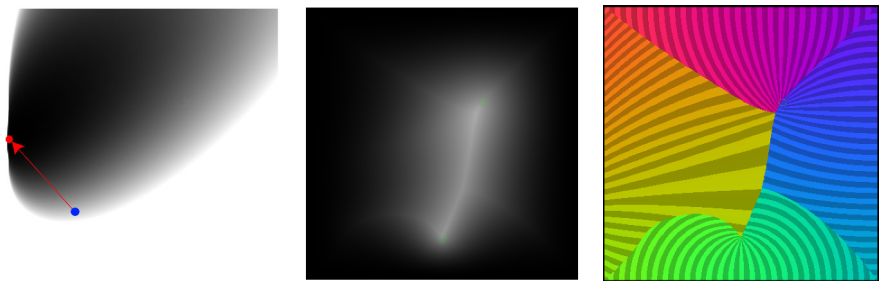
6.3.5 Specular Materials

Figure 6.3a shows an example of $I(q,p)$ for a Cook-Torrance microfacet BRDF[23]. In comparison to the diffuse $I(q,p)$ in Figure 6.3d, the darker area of the lobe is oriented such that it will more strongly limit the potential intensity of lights in the mirror direction from observation q.

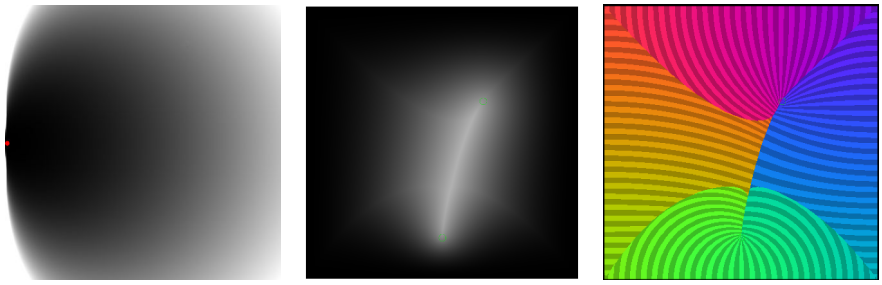
To more intuitively reason about how specularity interacts with the IDF, it is helpful to assume a minimum intensity for point lights illuminating a scene. This is reasonable because the number of point emitters in a scene is usually fairly small; for a fixed amount of total power in a scene, fewer lights imply that each individual light should have higher power. Therefore, lights are unlikely to lie in areas with low IDF values. This implies that the observations limiting these areas are pruning away these regions from consideration.

Traditional specularity triangulation methods interpret a single bright specular highlight as a strong indicator of a light source in the mirror direction. However, with the above formulation, a single bright observation is not assumed to be due to specularity – it could just indicate a nearby point light. Instead, a *lack* of specular highlight means that it is *unlikely* for a light to fall along the mirror direction.

This implies that additional observations of the same point greatly help our method. For example, if the point in the specular highlight is observed from a different angle such that it did not lie



(a) (b) (c)



(d) (e) (f)

Figure 6.3: Two scenes with the same light locations, but different materials, are shown above. The top row corresponds to a scene where the left wall is specular and the remaining walls are diffuse; the bottom row corresponds to the scene where all walls are diffuse. 6.3a shows $I(q, p)$ for a microfacet BRDF, where observation q is the red point viewed from the direction of the arrow. For comparison, a diffuse $I(q, p)$ is shown in 6.3d. 6.3b shows the full IDF for the specular scene, where all the points on the left wall are viewed from the red point. Due to the specular observations, the center of the scene is darker than the same region in a diffuse scene with the same light positions in 6.3e. The Limiter field behaves similarly at the true light positions in both cases (6.3c and 6.3f).

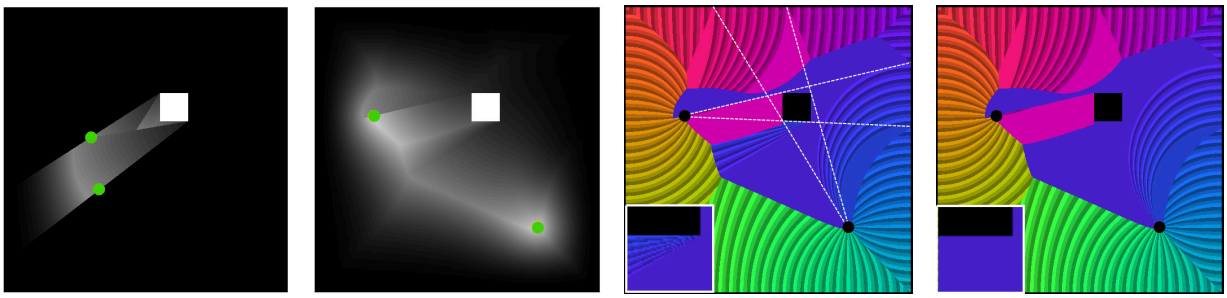
within the specular highlight, that would provide evidence against the presence of a nearby point light (because the observation would limit the brightness of nearby points), but leave open the possibility of a light in the original mirror direction. Viewing non-specular highlights thus allows the method to prune out more possible light positions.

An example of a full IDF for a scene with one specular wall (the left wall) is shown in Fig-

ure 6.3b, where each point on the specular wall is viewed just once. Note that $\mathcal{L}(p)$, and by extension, $\mathcal{V}(p)$, qualitatively functions the same way as in the diffuse case; however, in specular scenes, a denser set of observations is preferred in order to make the maxima clearer.

6.3.6 Occlusion

Occlusions do not significantly impact the performance of $\mathcal{V}(p)$ as a candidate position proposal scheme (Figure 6.4), since the voting function does not explicitly identify occlusion boundaries. $\mathcal{D}(p)$ is often discontinuous along shadow boundaries, and sometimes can directly result in candidate light locations at the intersections of the discontinuities (Figure 6.4a). This is equivalent to tracing rays from shadow edges to the associated occluders and finding the intersection.



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

Figure 6.4: In an extreme case, occluders entirely block some surfaces from being lit (6.4a). More commonly, occlusion boundaries result in discontinuities in the IDF (6.4b). The shadow edges (white dashed lines) are visible in the Limiter field (6.4c) as the limiters of larger uniformly-colored regions. The inset shows a region in $\mathcal{L}(p)$ where shadow edges can result in false peaks in the $V(p)$. These can be removed (6.4d) by determining whether or not occlusion was responsible for the local variation in the limiter field based on the two conditions outlined in Section 6.3.6.

For a point near an occluder, a small change in position can result in a large change in the surface regions visible to that point. If the limiter happens to lie in these disoccluded regions, then false peaks in $\mathcal{V}(p)$ will appear (Figure 6.4c-6.4d). These false peaks are caused when two

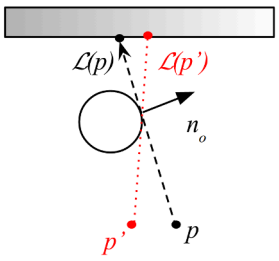


Figure 6.5: Consider point p and $\mathcal{L}(p)$ on the scene surface, with the line from p to $\mathcal{L}(p)$ tangent to some occluder. Note that $B(\mathcal{L}(p))$ decreases to the left, in the opposite direction from the normal to the occluder at the tangent. If we examine another point p' to the left of p, without the occluder we would also expect $\mathcal{L}(p')$ to be left of $\mathcal{L}(p)$ (since p' is the same distance from $\mathcal{L}(p')$ as p is from $\mathcal{L}(p)$, and $B(\mathcal{L}(p')) < B(\mathcal{L}(p))$). However, due to the occluder, instead $\mathcal{L}(p')$ is the first unoccluded point to the right of $\mathcal{L}(p)$. As p' moves to the left, $\mathcal{L}(p')$ continues being forced to the right due to the occluder, and if the occluder is far from $\mathcal{L}(p)$ the amount by which $\mathcal{L}(p')$ moves can become very large. This gets detected as a peak in $\mathcal{V}(p)$.

conditions are met (as illustrated in Figure 6.5):

- the line from $\mathcal{L}(p)$ to p is tangent to an occluding surface, and

- $B(\mathcal{L}(p))$ increases in the direction of the normal of the occluding surface.

These conditions can be detected when computing $\mathcal{L}(p)$; however, in practice, discarding candidate positions that are too close to scene surfaces is enough to filter out these false peaks.

The main practical issue with working with occlusions is that each $I(q, p)$ computation requires an occlusion test against the entire scene. The 2D test sets in Section 6.6 had few enough primitives to allow such full scene intersection tests to be feasible. In the 3D case, computing a depth environment map for each p reduces occlusion testing between x and p to a depth lookup. In the real-world 3D scenes in Section 6.6, the surfaces used to compute $\mathcal{V}(p)$ were effectively unoccluded from the entire scene, so to save computation time occlusion was ignored for those scenes.

6.4 Implementation of Scene Transforms

6.4.1 Computing the IDF

In implementation, the Intensity-Distance Transform (both $\mathcal{D}(p)$ and $\mathcal{L}(p)$) is constructed directly from Equations 6.6 and 6.7, computing $I(q, p)$ for each possible light position p, discretized over a regular grid, and each observation q. This brute force algorithm is trivially parallelizable.

For a 2D scene, the distance fields can be computed at interactive frame rates (45ms per frame) at 256×256 resolution with 400 surface intensity measurements, run on a GTX760M graphics card. Incorporating occlusion computations takes an additional 8ms per primitive.

6.4.2 Medial Axis

The discontinuities in $\mathcal{L}(p)$ can be used to identify the medial axis in a discrete fashion. $\mathcal{L}(p)$ is continuous off of the medial axis; that is, a small change in p will result in a small change in $\mathcal{L}(p)$. More specifically, if $d(x, y)$ is the distance along the scene surface between points x, y, then $d(\mathcal{L}(p), \mathcal{L}(p + \epsilon)) < T$ for a small perturbation ϵ with an appropriate threshold T. Discontinuities resulting in large $d(\mathcal{L}(p), \mathcal{L}(p + \epsilon))$ only occur on the medial axis. Thus, in 2D, by examining the neighbors of a pixel in clockwise order, a pixel is a medial axis pixel if the distance between limiters of any consecutive neighboring pixels is greater than T. In my experiments I set T to 5% of the scene's radius.

Although the medial axis is not used directly for light localization, the method described above for computing the medial axis directly leads to a way of discretely computing the voting function.

6.4.3 Voting Function

The basic definition of $\mathcal{V}(p)$ at a point is the number of different values $\mathcal{L}(p)$ takes in a small neighborhood around that point. With a discrete sampling of $\mathcal{L}(p)$, simply counting unique values in a pixel's neighbors gives insufficient granularity; for example, examining the immediate neighbors of a pixel can give at most 9 unique $\mathcal{L}(p)$ values. Instead, note that when $d(\mathcal{L}(p), \mathcal{L}(p + \epsilon)) < T$

```
T ←distance threshold
r ←circle radius
function IsMedialAxis(p, B, G)
    N(p) ←the pixels r away from p in clockwise order
    for p_i ∈ N(p) do
        if d(L(p_i, B, G), L(p_{i+1}, B, G)) > T then
            return true
        end if
    end for
    return false
end function
```

(where T is the medial axis discontinuity threshold, as described above), the continuity of $\mathcal{L}(p)$ implies that, for any x in between $\mathcal{L}(p)$ and $\mathcal{L}(p+\epsilon)$, there exists some point p' between p and $p+\epsilon$ such that $\mathcal{L}(p') = x$. Thus, if $d(\mathcal{L}(p), \mathcal{L}(p+\epsilon))$ is large (but less than T) then $\mathcal{V}(p)$ at p is large. However, if $d(\mathcal{L}(p), \mathcal{L}(p+\epsilon)) > T$, then there is a discontinuity somewhere between p and $p+\epsilon$; not every x between $\mathcal{L}(p)$ and $\mathcal{L}(p+\epsilon)$ will necessarily contribute a vote.

Thus, the vote total at a pixel p is computed based on the pixels on the perimeter of a circle with radius r centered on p. The total vote $\mathcal{V}(p)$ is the mean distance between the limiters of consecutive pixels on this circle, as long as the distance is less than the medial axis discontinuity threshold. Larger values of r are required for noisy data, while $r = 1$ pixel usually suffices for the noiseless case.

Sometimes $\mathcal{L}(p)$ may be constant in a region, which is the reason that the mean distance between limiters is used. These constant regions most commonly occur in the case of occlusion (see Figure 6.4), but also occur in the presence of noise (see Figure 6.7). The neighboring pixels that are inside these constant regions cannot contribute to the total distance. Using the mean distance ensures that the magnitude of the $\mathcal{V}(p)$ peaks on these boundaries is comparable to the magnitude of $\mathcal{V}(p)$ peaks elsewhere. To properly ignore constant regions, the distance between the neighbors

of consecutive pixels along the circle do not contribute to the mean when the neighbors are equal, i.e. when $d(\mathcal{L}(p), \mathcal{L}(p+\epsilon)) = 0$.

$T \leftarrow$distance threshold

$r \leftarrow$circle radius

function VOTINGFUNCTION(p, B, G)

  $N(p) \leftarrow$the pixels r away from p in clockwise order

  numcontinuous $\leftarrow 0$

  $\bar{d} \leftarrow 0$

  for $p_i \in N(p)$ **do**

    if $0 < d(\mathcal{L}(p_i, B, G), \mathcal{L}(p_{i+1}, B, G)) < T$ **then**

      numcontinuous $\leftarrow$ *numcontinuous* $+ 1$

      $\bar{d} \leftarrow \bar{d} + d(\mathcal{L}(p_i, B, G), \mathcal{L}(p_{i+1}, B, G))$

    end if

  end for

  return $\bar{d}/$*numcontinuous*

end function

Candidate light positions are proposed as local maxima in the voting function with a value greater than some minimum H. In practice, a very high value of $H = |Q|/10$ is initially used and then is reduced until at least one peak is found. Multiple peaks are combined if the distance between them is less than some user-defined threshold (10% of the scene radius in this thesis).

6.4.4 Moving to Three Dimensions

In 3D scenes, the computation of $\mathcal{D}(p)$ and $\mathcal{L}(p)$ is identical to the computation in 2D. However, this process is no longer real-time. Several orders of magnitude more surface measurements must be made to reduce discretization artifacts in the distance field. Computing the distance field at 512^3 resolution with $|Q| = 60\,000$ takes about 1 hour on a Titan X GPU.

```
r ←circle radius
function GETLIGHTCANDIDATES(B, G)
    candidates ← {}
    for p ∈ scene do
        N(p) ←the pixels within distance r from p
        isLocalMax ←true
        for p_i ∈ N(p) do
            if VOTINGFUNCTION(p, B, G) < VOTINGFUNCTION(p_i, B, G) then
                isLocalMax ←false
            end if
        end for
        if isLocalMax then
            candidates ← candidates ∪ p
        end if
    end for
    return candidates
end function
```

Computing $\mathcal{V}(p)$ is slightly more complex in 3D. In 2D, $d(x, y)$ was computed by taking the distance along the perimeter of the scene. In 3D, the analogous approach to compute $\mathcal{V}(p)$ is to triangulate the points on the surface of a sphere centered at p, and then for each triangle with vertices A, B, C let $d(A, B, C)$ be the geodesic area (i.e. area along the surface of the scene) spanned by the triangle $\mathcal{L}(A), \mathcal{L}(B), \mathcal{L}(C)$. In the fairly simple test cases in Section 6.6, the geodesic area is approximated by the solid angle spanned by the triangle relative to the center of the scene. While this resulted in some artifacts (e.g. underestimating votes near the corners of boxy scenes) it still gave good localizations.

6.5 Refinement Algorithm

The candidate light positions proposed by the voting function are often incomplete or inaccurate, for example in Figure 6.2d. A light might be "hidden" by other lights, such that no surface in the scene is primarily lit only by that light, and thus no peak is produced in $\mathcal{V}(p)$ (e.g. the second light from the right). Even identified lights have inaccurate positions because of the influence of other light sources (e.g. the lower-right light). Therefore the proposed lights must be refined.

The refinement algorithm is shown in Algorithm 1. The algorithm stores a set of hypothesis lights with positions X and intensities I, and keeps track of how much of the scene's original appearance remains unexplained by these hypothesis lights. Iteratively increasing the intensity of some of the hypothesized lights and then refining the positions of all hypothesized lights explains more of the scene's illumination. Then, recomputing the voting function based on the remaining unexplained illumination may reveal new lights not yet in the hypothesis set. Any time a new light is found, a full nonlinear optimization is performed to check for convergence.

6.5.1 Light Discovery

Light discovery begins by computing the voting function of the current unexplained illumination $B'(q)$,

$$B' = B - \textsc{Render}(G, X, I)$$

where RENDER computes the illumination due to direct lighting for point lights at positions X with intensities I for all surfaces in the scene.

A set of candidate light positions is extracted from the voting function; these candidates are then associated with the current set of hypothesized lights based on a distance threshold (See Algorithm 2). Note that not all hypothesized lights will appear in the set of current candidates; for example, if a correctly placed hypothesis light's intensity reaches its true intensity, then it will completely cancel out the effects of the true light. On the other hand, if there is a candidate light that is not yet in the hypothesis list, then it is added to the hypothesis list with 0 intensity and an optimization step is triggered.

Algorithm 1 Light refinement algorithm

$\delta \leftarrow$light increment

$\epsilon \leftarrow$error threshold

function LOCALIZELIGHTS(B, G)

$X \leftarrow [], I \leftarrow []$

while not *success* **do**

$B' = B -$ RENDER(G, X, I) ▷ Find light candidates

if $\exists\, b \in B' \mid b < 0$ **then**

return $\{\}$

end if

$X' \leftarrow$ GETLIGHTCANDIDATES(B', G)

$M', N' \leftarrow$ ASSOCIATE(X, X') ▷ Add new lights

for $x \in N'$ **do**

Append x to X

Append δ to I

end for

if $N' \neq \{\}$ **then** ▷ Optimize if new lights are detected

$\hat{I} \leftarrow$ OPTIMIZEINTENSITIES(B, X, I)

$\hat{X}, \hat{I} \leftarrow$ OPTIMIZE($B, X, \hat{I}$)

if $||B -$ RENDER$(G, X, I)||_2 < \epsilon$ **then**

return $\hat{X}, \hat{I}$

end if

end if

for $m \in M'$ **do** ▷ Increment light intensities

$I_m \leftarrow I_m + \delta$

end for

$X \leftarrow$ REFINEPOSITIONS(G, B, X, I) ▷ Refine positions

end while

end function

Only the lights that are in the current set of candidates will have their intensities incremented in this iteration. The algorithm terminates in failure if incrementing intensities results in $B'(q) < 0$ for any q.

Algorithm 2 Associating hypothesis lights with candidate lights

```
d ←light equality distance threshold
function ASSOCIATE(X, Y)
    Matches← {}
    New← []
    for y ∈ Y do
        Matched← false
        for x ∈ X do
            if ||x − y||_2 < d then
                Matches ← Matches ∪ x
                Matched← true
            end if
        end for
        if not Matched then
            Append y to New
        end if
    end for
    return Matches, New
end function
```

6.5.2 Position Refinement

Suppose that all but one of the hypothesis lights were in the correct locations with the correct intensities; then subtracting the hypothesized illumination due to the correct lights from the measured

illumination should leave the scene illumination as if it were lit only by the single incorrect light. But for a single-light scene, the scene transforms will directly give the true light position.

Based on this intuition, the position refinement step proceeds as follows: For each hypothesis light, recompute $\mathcal{V}(p)$ based on the scene illumination with the contributions of the other hypothesis lights removed; in other words, compute the votes using

$$B^x = B - \textsc{Render}(G, X \setminus x, I \setminus I_x)$$

Then, extracting the peaks of $\mathcal{V}(p)$ and finding the one that corresponds to the current hypothesis light will give its refined position. Pseudocode for this method is shown in Algorithm 3.

Algorithm 3 Light position refinement algorithm

```
d ←light equality distance threshold
function RefinePositions(G, B, X, I)
    X' ← []
    for x ∈ X do
        B^x ← B − Render(G, X \ x, I \ I_x)
        X^x ← GetLightCandidates(B^x, G)
        d̂ ← d, x' ← x
        for y ∈ X^x do
            if ||x − y||_2 < d̂ then
                x' ← y
            end if
        end for
        Append x' to X'
    end for
    return X'
end function
```

6.5.3 Optimization

The optimization procedure consists of two steps. First, holding the positions of the hypothesized lights constant, their best-fit intensities are determined. This step is linear and can be directly computed. We form the matrix of $S(q, p)$ values, where positions q take on the locations of the observations B and positions p take on the current (fixed) hypothesis light locations. Then, from Equation 6.4, we have $B = SI$ with I unknown, which can be solved in a least squares manner.

This first intensity-only optimization step is necessary because the hypothesized intensities for recently discovered lights are likely to be far below their correct intensities, and are therefore poor initializations for the final optimization. Using the computed intensities provides a much better initialization. The subsequent nonlinear optimization for both position and intensity comes from expanding the $S(q, p)$ term in Equation 6.4 and is performed using the Levenberg-Marquardt method implemented in Ceres Solver [2]. If the resulting error is low enough, then the algorithm is successful. This optimization can be accelerated by terminating early if the light positions stray too far from their initial estimates. A failed optimization usually implies that the algorithm has not yet found all the lights in the scene. Note that this optimization does not take occlusion into account, since no closed form expression for $V(q, p)$ in Equation 6.2 exists. In scenes with occlusion, I assume that the final light positions in a successful optimization will be close to the initializations; therefore rather than recomputing the visibility $V(q, p)$ based on an iteratively updated position p, the optimization uses a fixed $V(q, p)$ computed from the initial position of the light.

6.6 Results and Discussion

6.6.1 2D Synthetic Data

To validate this light localization method, I generated a synthetic 2D dataset with 400 diffuse square room scenes with varying lighting conditions. I generated 100 random configurations each for two-light, three-light, four-light and five-light scenes, each light having random intensities between 1 and 5. While generating these configurations, I ensured that no light was closer than 5% of the scene's side length to another light or to a wall.

The scene geometry for this quantitative evaluation was a simple diffuse box scene. This functions as a worst-case setup for light localization, as none of the stronger cues such as specular highlights or occlusions are present. Although scenes comprised of large planar surfaces admit other approaches, this method does not rely on the presence of planar surfaces.

In addition to the noise-free case, I also examined how small amounts of noise affect the performance of the method. I approximated the effects of shot noise in CCD sensors by replacing the true direct intensity value $B(q)$ with a random variable with a Gaussian distribution with mean $B(q)$ and standard deviation $k\sqrt{B(q)}$; in these results $k = 0.05$.

For the noiseless case, the method computes $\mathcal{V}(p)$ with a radius of 1, and the threshold for convergence is set to a MSE of 0.0025. For the noisy case, the method computes $\mathcal{V}(p)$ with a radius of 15, and the threshold for convergence is set to 105% of the MSE of the ground truth lighting. For both cases, the per-iteration hypothesis light intensity increment was 0.025.

The runtimes and the success rates of the light localization algorithm are shown in Table 6.1, including results with and without noise. A single computation of $\mathcal{V}(p)$ frequently provides an initialization giving a successful optimization without further iterations. Runtimes were measured on an Intel i7-7700HQ CPU with a GTX1060 GPU.

For the noiseless case, this algorithm can successfully recover the original lighting conditions in all of the two, three, and four light cases. However, it fails in three of the five-light test cases. All of these failures, as well as three other cases in which the ground truth lighting was not recovered, involve clusters of several nearby lights that resulted in some ambiguity. Some of these failure cases are shown in Figures 6.6-6.7.

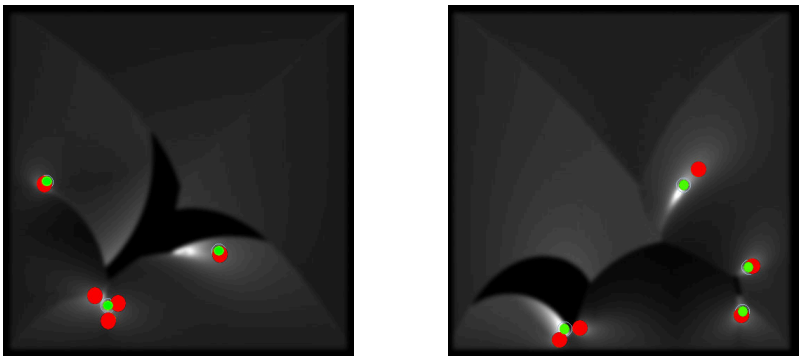
In the noisy case, this algorithm is slightly less successful, although it still matches the original lighting conditions the majority of the time. The winner-take-all nature of the distance field means that noise can cause large constant regions in $\mathcal{L}(p)$ (where the same surface point is the limiting point for a large region), especially closer to the center of the scene. These constant regions hide any useful information from $\mathcal{V}(p)$. This effect, illustrated in Figure 6.7a-6.7b, is the cause of all the two- and three- light failure cases, and most of the four- and five-light failure cases. Several of the four- and five-light failure cases are due to spurious peaks, caused by noise, being confused

	Success Rates				Runtimes (s)	
$\|L\|$	Random	$\mathcal{V}(p)$	Solved	Full	Median	Max
Noiseless						
2	0.77	0.96	1.00	1.00	2	64
3	0.52	0.73	1.00	1.00	3	105
4	0.29	0.35	1.00	1.00	29	250
5	0.26	0.15	0.94	0.97	85	442
Noisy						
2	0.80	0.83	0.78	0.99	2	52
3	0.53	0.66	0.73	0.98	13	492
4	0.32	0.41	0.64	0.95	34	301
5	0.23	0.21	0.38	0.84	80	378

Table 6.1: Synthetic 2D Dataset results and runtimes. The Random column shows the proportion of randomly initialized nonlinear optimizations which achieved a low-error solution. The $\mathcal{V}(p)$ column gives optimization success rates after a single round of voting, while the Full column corresponds to running the full algorithm. The Solved column gives the proportion of scenes resulting in successful recovery of the ground truth lighting, while the $\mathcal{V}(p)$ and Full columns include low-error solutions that did not recover the ground truth lighting.

with actual light peaks, while the remaining failures come from scenes similar to Figure 6.6b. Determining a more robust way of collecting votes, or in general designing a more robust candidate proposal method, is an important area for further investigation. In practice, real-world captures often involve multiple samples of the brightness of each surface; combining these samples can significantly reduce the amount of noise.

As a baseline, I ran a Levenberg-Marquardt nonlinear optimizer (using Ceres Solver) with random initialization 100 times on each of the scenes, and computed the proportion of the runs which successfully converge. In this baseline, I assumed that the number of lights is known a priori. Despite this extra knowledge, the baseline frequently fails to find a solution reproducing the original illumination conditions.



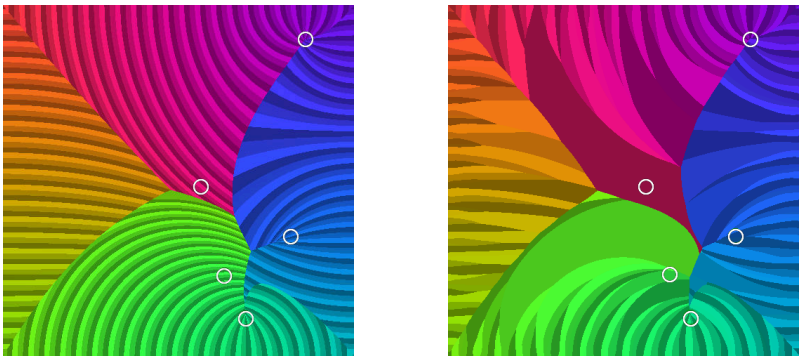
(a) (b)

Figure 6.6: Failure case due to light clustering: 6.6a shows $\mathcal{V}(p)$ where the method converged on a low error solution (green) different from the ground truth (red), replacing the lower-left cluster of lights with a single light. In 6.6b, the cluster of two lights near the bottom of the scene is closer to the wall and a single light cannot fit a low-error solution. $\mathcal{V}(p)$ never manages to distinguish two separate peaks, and the algorithm terminates without finding a solution.

It is worth noting that for noiseless 2D box scenes, an algorithm that simply picks the lowest-error solution out of these 100 runs will perform about as successfully as the described light localization algorithm. The difficult scenes for the repeated baseline algorithm tend to involve cases where some lights are very close to the edges of the scene; the failure cases for this method are completely disjoint from the failure cases of my algorithm.

6.6.2 3D Synthetic Data

Beyond 2D experiments, I also ran this algorithm on several synthetic 3D scenes (Figure 6.8). I informally observed that in 3D, it is more likely for the initial voting function proposals to provide the full solution. This is likely due to the fact that, in a higher-dimensional space, more points in the interior of a region are close to that region's boundaries (a well-known implication of the curse of dimensionality). Thus, each light is more likely to be very close to a scene boundary, and therefore is more likely to be the dominant source of incident illumination for that boundary.



(a) (b)

Figure 6.7: Failure case under noise: Noiseless $\mathcal{L}(p)$, in 6.7a, compared with $\mathcal{L}(p)$ of the same scene with noise, in 6.7b. The region near the leftmost light (near the center of the scene) is completely taken over by just two Voronoi cells. There is no way of computing a meaningful vote in these areas.

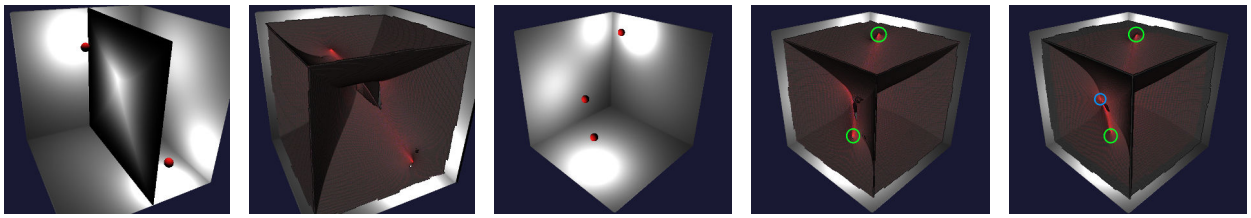
Because of the brute-force IDF approach, each computation of a 3D $\mathcal{V}(p)$ takes about one hour, while each iteration of the algorithm requires numerous $\mathcal{V}(p)$ computations. I expect that investigating alternate methods of computing $\mathcal{D}(p)$, or approaches that avoid computing the full IDF (e.g. focusing on the medial axis), would lead to better performance in 3D scenes.

For reference, I also ran an optimization baseline analogous to the one performed in 2D, initializing a cubical room with two to five lights, each in 5 random configurations. The results are shown in Table 6.2. The baseline achieved similarly low success rates to the 2D optimization, but each optimization attempt took much longer as well. In comparison, despite taking much longer to run, my algorithm was successful in all cases.

6.6.3 Real-World Data

I scanned two real world scenes to demonstrate that my method works even when some of the assumptions are violated. In both scenes, I set up two bare light bulbs (approximating isotropic point sources) and measured their positions by hand.

I scanned the scene using the Lenovo Phab2 Pro Tango phone, reconstructing the scene geom-



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

Figure 6.8: Results on synthetic 3D scenes. In 6.8a-6.8b, the maxima of $\mathcal{V}(p)$ directly give the light locations in a two-light scene. A visualization of $\mathcal{D}(p)$ on a planar slice is shown in 6.8a. The medial axis is shown in 6.8b, colored with $\mathcal{V}(p)$. In a three-light case (6.8c), the initial $\mathcal{V}(p)$ maxima give the locations of two of the lights (6.8d, circled in green); after a few iterations the third is revealed (6.8e, circled in blue).

etry using the Tango software. After using the HDR exposure correction pipeline from Chapter 3, I projected the RGB images onto the geometry, shown in Figure 6.9a-6.9b. In more detail, the process for each scene was as follows:

1. First, I scan the scene using the Lenovo Phab2 Pro Tango phone, reconstructing the scene geometry using the Tango software.

2. I then perform exposure correction on the RGB images, project the images onto the geometry, and take the robust minimum (average of the values between the first quartile and the median) intensity as the $B(q)$ (with a diffuse assumption).

3. I manually segment out some surfaces in the scan that are believed to have the same diffuse reflectance (generally the walls).

4. Using only $B(q)$ at these surfaces, I run the iterative light localization algorithm, computing $\mathcal{V}(p)$ and extracting candidate positions.

5. During candidate position extraction, I filter out unlikely light positions, namely those that lie very close to scene surfaces. I also filter out light positions p where $\mathcal{D}(p)$ is smaller than

# lights	Success Rate	Median time (s)	Max time (s)
2	0.49	95	165
3	0.27	283	398
4	0.07	499	665
5	0.05	1007	1083

Table 6.2: Synthetic 3D Dataset results and runtimes using the optimization baseline. Five scenes per light count were generated, and each scene was initialized randomly and optimized 50 times. The Success Rate column gives the overall proportion of runs that recovered the original illumination. The timing columns give the median and maximum time for a single optimization run.

some minimum light intensity threshold, following the reasoning in Section 6.3.5.

6. During the optimization portion of the algorithm, I solve for light intensities, positions, as well as an unknown constant ambient light intensity to deal with indirect illumination.

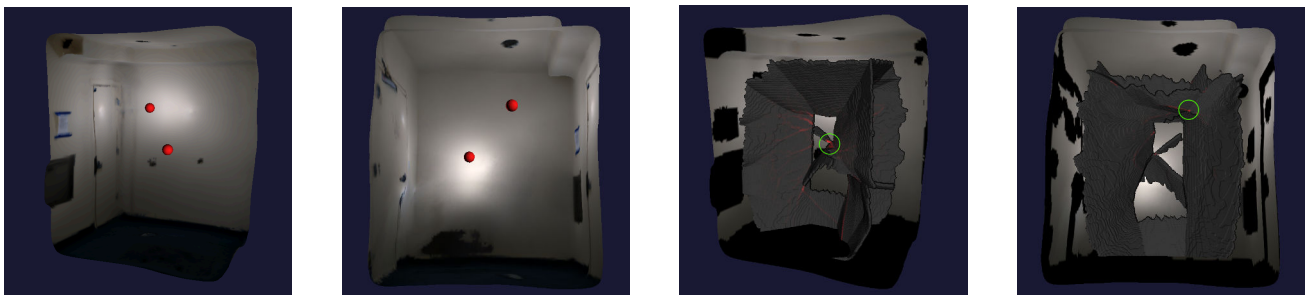
7. To evaluate the positional accuracy of the system, I compare the optimized positions with the hand-measured ground truth positions.

8. To evaluate the accuracy of the estimated intensities, I compared the relative intensities of the two lights by fixing one of the intensities to 1 (since the captured images could not provide absolute intensities). I obtained ground truth relative light intensities by taking an image of a diffuse scene lit only with one light, then replacing that light with the second light in the exact same position. Due to linearity of light, the average ratio between the two images provides the relative light intensity.

To avoid having to recover materials for the entire scene, I take advantage of two properties of this method. First, since the algorithm does not rely on having a complete or closed scene, the scene transforms and optimization only need to be run over a subset of scene points. Furthermore, if this subset consists of diffuse surfaces with identical albedo (e.g. wall points), then $f(x, \omega_i, \omega_o) = \rho/\pi$

for constant ρ, and π/ρ is a global scaling of all $I(q, p)$. This means that the limiter field $\mathcal{L}(p)$ is independent of the actual albedo ρ.

I manually segment out wall vertices to form Q (although this can easily be replaced by an automatic material segmentation pipeline). Because of the diffuse assumption, the projected RGB colors of wall vertices can be directly used as $B(q)$.

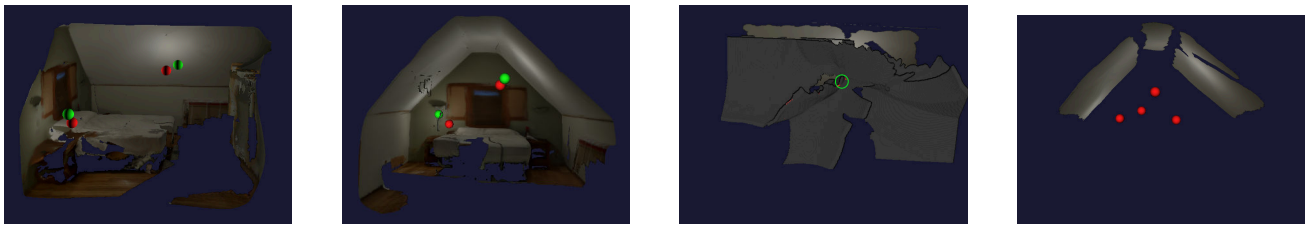


(a) (b) (c) (d)

Figure 6.9: Results of running my method on a scanned 3D dataset. 6.9a-6.9b show the scanned scene colored with the averaged observations $B(q)$. The recovered light locations are rendered as red spheres; compared with the ground truth, the position error is 4% of the scene radius. The bottom two images only show the $B(q)$ used in the computation. In 6.9c, the initial medial axis and $\mathcal{V}(p)$ are shown, with a clear maximum (circled in green) at the location of one of the light sources. After a few iterations (6.9d), the second light position is also revealed.

Using camera images directly for B introduces the issues of noise, indirect illumination, and unmodelled material properties (including specularity). With diffuse Q, the viewing direction ω can be ignored and all the observations of a single wall point can be combined by taking the robust minimum (average of the values between the first quartile and the median) of the projected colors This also removes the erroneous projections of the light stands, bulbs, and glare onto the walls.

For simplicity, in this thesis I ignore the effects of indirect illumination on the candidate proposal scheme, and assume that incident indirect illumination is negligible relative to direct lighting at the surfaces given by $\mathcal{L}(p')$ for p' near the maxima in $\mathcal{V}(p)$. I model indirect illumination in



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

Figure 6.10: Results of running my method on a scanned 3D dataset. 6.10a-6.10b show the scanned scene colored with the averaged observations $B(q)$. The original light locations are rendered as green spheres, while the recovered light locations are rendered as red spheres. The bottom two images only show the $B(q)$ used in the computation. In 6.10c, the initial medial axis and $\mathcal{V}(p)$ are shown, with a clear maximum (circled in green) at the location of one of the light sources. After a few iterations (6.10d), several more maxima appear (rendered as red spheres); two among these are close enough to the true light positions for the optimization to succeed.

the optimization as a constant ambient term. If the reflectances had been directly measured, then indirect illumination could just be removed directly, as described in Section 6.3.

The first scene, shown in Figure 6.9, is a small, relatively bare room. Examining $\mathcal{V}(p)$ shows a clear maximum at one of the true light positions (Figure 6.9c), and after a few iterations of the refinement algorithm, the other light also appears (Figure 6.9d), at which point the optimization succeeds. No other maxima are detected. The error in the estimated positions, relative to hand-measured ground truth positions, were 0.05m and 0.08m (in a $2\text{m} \times 2.6\text{m} \times 1.6\text{m}$ room), for a relative error of about 5%. As the captured images did not give absolute intensities, I compared relative intensities by fixing one of the light intensities to 1. The estimated intensity for the second light was 1.43, while the measured relative intensity (based on raw images captured with a DSLR) was 1.39.

The second scene (Figure 6.10) is a harder case: the room was larger, had more complex geometry (a vaulted ceiling), included clutter, and was incomplete and not closed. One light gives a clear maximum in $\mathcal{V}(p)$ (Figure 6.10c); after several iterations, several new maxima appear.

Most of them are heuristically filtered out, leaving four candidates, including the original maximum (Figure 6.10d). After optimization, the maxima that did not correspond to true lights had very small intensities compared to those that did represent true lights. Specifically, the false lights had less than 5% of the intensity of the true lights. To get a final solution, I optimized again but removed the lights with low intensities. The error in the estimated positions, relative to hand-measured ground truth, were 0.26m and 0.31m in a $3.5\text{m} \times 3.4\text{m} \times 2.6\text{m}$ room, for a relative error of about 10%. The estimated intensity for the second light was 1.42, while the measured relative intensity was 1.39.

6.6.4 Limitations and Future Work

Known Materials The main practical issue with using this method in real-world scenes is the assumption of known materials. It is difficult to separately use a BRDF estimation pipeline and then apply this method because BRDF estimation in scenes is highly dependent on lighting models. While it is straightforward to incorporate material estimation into the optimization part of this pipeline, I would like to investigate ways to adapt the candidate position proposal method and scene transforms to handle unknown materials.

Directly Imaged Lights This method is designed for cases in which light emitters are not directly imaged, for example due to physical constraints during the scanning process. However, if emitting surfaces are imaged, they must be removed before computing the IDF. Ideally, information about known light emitters could be incorporated into the system, rather than discarded.

Beyond Isotropic Point Lights In Chapter 5, I noted that real-world emitters are much more complex than isotropic point sources; instead, they frequently have radially-varying intensity distributions and different geometries (e.g. lines or areas). While the described refinement algorithm can apply to any lighting model, examining how directional and geometric variations affect the candidate light proposal method and scene transforms is an important area for further investigation.

Chapter 7

MAKING EDITED MODELS MORE REALISTIC

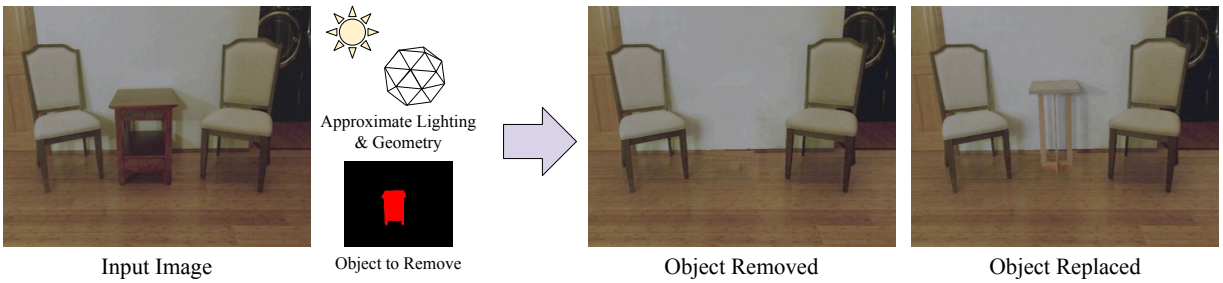


Figure 7.1: I present a method to remove an object and its shadows from an image, to enable applications like home refurnishing. This method takes as input an image, approximate scene lighting and geometry, and an object mask, and generates a new version of the image that depicts the scene as if the object had not been present. This not only includes inpainting the occluded pixels, but removing any shadows cast by the object.

7.1 Introduction

In Chapter 1, I described how existing scene manipulation pipelines work by constructing an approximate proxy model of the scene, and then perform edits on that scene model. The final image of the manipulated scene is obtained by rendering that model. Indeed, the refurnishing and relighting results shown in Chapter 5 were such direct renderings of reconstructed scene models.

Because these results come from photorealistic rendering processes, the final images are self-consistent. Shadows and interreflections appear correctly based on simulations of light propagation. However, are they truly realistic? Reconstructed models are rarely expressive enough to accurately represent the appearance of the scene. Limited sensor resolution, inaccuracies in the capture process, and incomplete data all contribute to noisy models. Simplifying priors on the reconstructions, such as the Manhattan world assumptions or diffuse assumptions used in previous

chapters, are necessary to obtain clean results, but result in models that are less representative of the original scene's appearance.

In this chapter I present a process by which the inaccuracies in a scene model can be compensated for when performing scene edits, specifically, object removal[1]. Unlike previous works in object removal, this method removes not just the object, but also the shadows it casts. My system combines recent advances in neural rendering with classical scene-proxy-based approaches to accurately perform object removal, with consistent lighting, while preserving the detail present in the original scene.

7.2 Related Work

7.2.1 Object Removal

Research on object removal has traditionally focused on the inpainting problem, ranging from classical techniques such as PatchMatch [4] to recent learning-based techniques such as DeepFill [153] and HiFill [151]. These methods do not consider lighting interactions between the removed object and the rest of the scene; thus when removing the object by inpainting, the user-specified mask must be extended to include the object's shadow. Recent work by Wang *et al* [133] employs deep networks to associate shadows with their casters; however, their instance segmentation approach produces hard boundaries and does not work for soft shadows.

7.2.2 Shadow Removal

Removing shadows from images is another problem that has a long history. Note that the goal of these works is to remove all shadows from an image, while the goal in this chapter is to remove the shadow of a single object; however, in this thesis I assume the presence of a rough scene proxy.

Classical intrinsic image decomposition methods are designed with various priors, typically specializing in low-frequency lighting and thus handling soft shadows well [6, 5, 9, 45]. Another

[1] This chapter describes work originally published at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2021 [157]

set of methods specialize in hard shadows and classify gradients as shading or texture [8, 128]; however, these methods break down when shadow receivers have complex texture. Finlayson *et al* [32] place assumptions on light source chromaticity, allowing for removal of both soft and hard shadows at the expense of generality.

Recent methods use deep networks to perform shadow detection and removal, starting with work by Qu *et al* [107]. Advances such as adversarial losses [132, 30], a two-stage detection-then-removal scheme [55], or lighting inference [75] have resulted in great improvements on shadow removal on the common ISTD [132] and SRD [107] datasets. However, these datasets only contain hard shadows produced from outdoor lighting. My system is trained to handle much more diversity in lighting conditions.

7.2.3 Scene Editing

Editing scenes in a visually realistic manner has long been an area of interest in the graphics community. Most of this work has focused on virtual object insertion. Classical methods construct an approximate model of the scene to help perform these edits, ranging from Debevec's early work [27], which assumes lighting and geometry were directly captured, to more recent work by Karsch et al. [62], which infers geometry, albedo, and lighting from a single image. Beyond simply inserting objects, Kholgade *et al* [67] are able to move an object around, although they assume that a high-quality 3D model of the object is available.

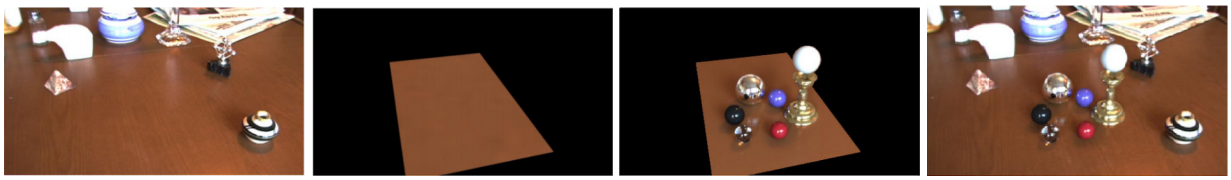
The issue of the limited range of scene models is inherent to all of the methods that rely on such models for scene editing. Most works (e.g. Kholgade [67]) use Debevec's differential rendering method [27] to account for differences between the model and the real scene. Recent approaches for neural rerendering use image-to-image translation to map from the domain of the approximate model to a realistic result [88, 92]. Philip *et al* [105] introduced a method for relighting outdoor photographs that also leverages proxy scene models. Their system is designed to handle global changes in illumination, like changing the position of the sun; furthermore they rely heavily on shadow masks that cannot handle complex environment lighting. Instead, I focus on local changes informed by the differences in the appearance of the proxy scene, based on an intrinsic decompo-

sition that can handle multiple soft shadows.

Modern deep-learning approaches have begun to approach scene editing problems in an end-to-end fashion. Bau *et al* [7] manipulate the activations within a GAN to allow for semantic manipulations such as object removal and insertion, frequently with nonlocal lighting effects; however, the results exhibit many artifacts and manipulations are limited to those that lie in the network's latent space.

7.3 Background: Appearance Compensation and Differential Rendering

When considering mismatches between model and reality for scene manipulation, four different images naturally arise: the original scene appearance I, the rendering of the proxy model of the original scene P, the proxy model with the scene edit applied P', and the desired final scene appearance under manipulation I'. These images are shown in Figure 7.2 for an insertion operation. The challenge for many systems is how to effectively incorporate knowledge of I, P, P' in order to produce I'. Note that this formulation is specifically for analyzing the lighting effects on the existing scene; usually some masking or inpainting is done so that the pixels occupied by inserted or removed objects are separately handled.



(a) I (b) P (c) P' (d) I'

Figure 7.2: Compensating for inaccurate models. The original input image I contains a textured surface as the local scene, while the rendered model P uses a uniformly colored surface and does not represent the textural variation of the wood grain. The desired scene edit, in this case object insertion, is applied to the model and rendered to produce P'. The desired ground truth appearance, as if the scene edits were physically performed, is I' (synthetic result shown above). These images are from Debevec [27]

(a) O (b) M (c) $(1 - M)D$ (d) $A + (1 - M)D + O$

Figure 7.3: Differential rendering based on the example from Figure 7.2 The inserted objects in P', without the local scene, are shown as image O. The pixels corresponding to rays coming directly from O are masked out to produce image M. The difference image $D = P' - P$ is multiplied by the mask to get the change in appearance of the local scene due to the edits. This image is added to the input image I, and then finally composited with the inserted objects O, to obtain the result I'. Images from Debevec [27]

In this formulation, methods which only compute results based on inferred models, such as the results in Chapter 5, ignore I and take $I' = P'$. Modern methods for image manipulation based on deep neural networks, such as [143, 150], essentially try to learn a mapping directly between I and I' such that $I' = G(I)$, where G represents a deep network. While results are sometimes impressive, these methods are limited to scenes resembling the training set, and are often of very low resolution. More importantly, due to the 2D (or 2.5D at best) nature of the networks, they do not have an explicit understanding of light transport, and therefore have difficulty generalizing to different scenes and produce results with limited realism.

One major limitation of this four-image formulation is that secondary lighting effects due to unmodeled features of the scene cannot be represented. For example, if a textured surface is modeled with a single diffuse color, then the reflection of the textured surface will not appear in shiny virtual objects. Therefore, it is still important to make our underlying models as expressive as possible.

The most common method for bridging the gap between model-based rendering and image-based synthesis is known as *differential rendering*, introduced by Debevec [27]. Designed specifically for object insertion operations, Debevec assumes that $I' - I = P' - P$ and uses this formula-

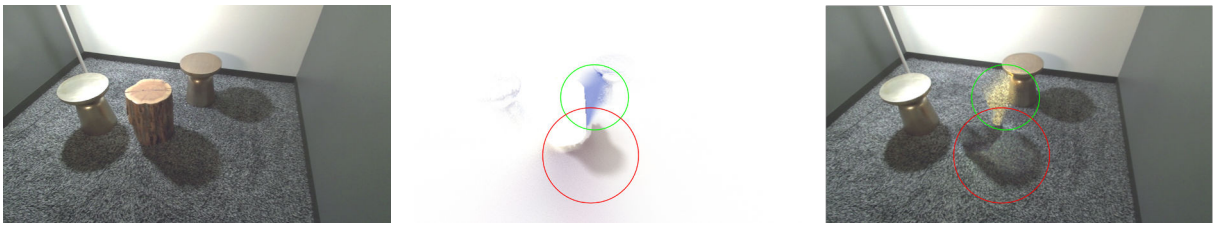
tion to insert reflections and shadows of virtual objects onto real surfaces. $D = P' - P$, computed from the model, is termed the difference image (see Figure 7.3), and represents the change in scene appearance due to the introduction of the virtual objects. This approach was primarily designed to compensate for unmodeled variations in the diffuse reflectance of the local scene, which are very apparent to human observers, and as such has been widely applied in subsequent work (including [63, 64, 67]). Note that this additive model for the difference image is actually physically incorrect for shadows: changes in illumination are *multiplicative* with diffuse albedo variation (while specularities are usually additive with diffuse appearance).

Recent research has leveraged advances in real-time rendering techniques, such as precomputed radiance transfer [119] and voxel cone tracing [25], in order to directly compute difference images in real time [36, 37].

Differential rendering produces visually compelling results to compensate for unmodeled textural variation; however, handling other inaccuracies in the estimated model, such as incorrect specular parameters, incorrect lighting, or incorrect geometry, has not been well studied. While lighting and specularity mismatches tend to be of low frequency, the artifacts resulting from incorrect geometry estimation tend to be of high frequency (as are textural details) and are much more apparent to human observers. For example, incorrectly estimating the shape of an object might result in an estimated hard shadow edge falling very far away from the true location of the shadow boundary. Because of this fact, systems using differential rendering tend to model only simple local scenes, usually consisting of a single ground plane.

While differential rendering has been widely used for insertion operations, it is not directly applicable to object removal. Applying the pixelwise difference directly results in very obvious artifacts since the shadows in the proxy model are only a rough estimate of the real shadows. For example, applying differential rendering to the scene from Figure 7.10 results in many artifacts, as can be seen in Figure 7.4. The inaccurate geometry in the proxy causes the extent of the removed stool's shadow to be underestimated. The scene lighting, consisting of one lamp visible in the image and one area light behind the camera, is poorly represented by the distant lighting model captured by the HDRI environment map in Figure 7.10. In the environment map, both light sources

appear to be of similar intensity and therefore cast shadows of similar appearance. In reality, the area light is much further away and only casts a faint shadow. Thus differential rendering attempts to "remove" a shadow that is not truly present, resulting in a brightening of the image. Furthermore, the captured lighting had to be radiometrically calibrated to match the input image. In contrast, the neural rendering method I will present can handle not only textural inaccuracies in the proxy model, but also inaccuracies in lighting and geometry.



(a) Input I (b) Ratio Image P/P' (c) Differential Rendering Result I'_{diff}

(d) Our Result I' (e) Ground Truth $\hat{I}'$

Figure 7.4: Differential rendering on a real scene. The incomplete geometry causes the proxy model to underestimate the extent of the object's shadow (red), while the captured distant illumination drastically overestimates the intensity of the secondary light source (green).

7.4 Method

The differential rendering shadow removal pipeline consists of a series of convolutional neural networks, followed by an inpainting stage to produce the final results. In this thesis, each component is a single U-Net [113]. A visual overview can be seen in Figure 7.5. The inputs to this system are the original image, a rendering of the approximate scene model before object removal (referred to

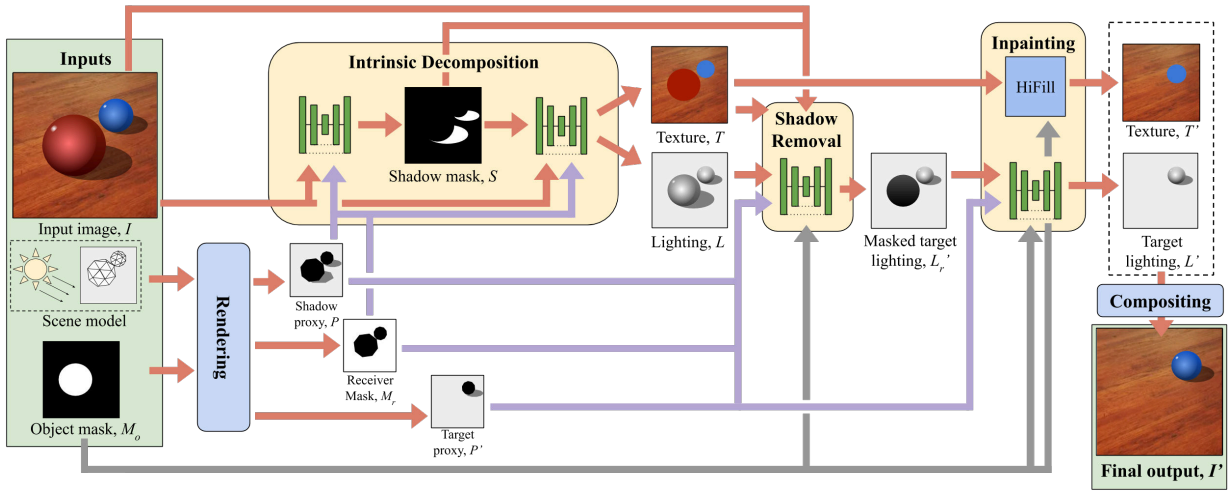


Figure 7.5: Overview of the differential rendering shadow removal approach. The inputs are an image to edit, a mask of the object to remove, and renderings of an approximate scene model with and without the object. The system first performs an intrinsic decomposition of the input image into texture and lighting. Then, the object's shadows are removed from the lighting image. The object mask region is inpainted separately in both texture and lighting before recompositing to get the final result.

as the *shadow proxy*), a rendering of the scene model after object removal (*target proxy*), and binary masks denoting the object to remove and the receiving surface from which to remove shadows. An overview of the pipeline follows:

- The **intrinsic decomposition subsystem** separates the input image into texture and lighting, guided by the shadow proxy. Following existing works [55] on shadow removal, this subsystem uses a two-stage scheme with an initial shadow segmentation network.

- The **shadow removal** network removes the shadow of the removed object from the decomposed lighting, aided by the shadow proxy and target proxy images.

- The **inpainting subsystem** separately inpaints the lighting and texture behind the removed object. A learned lighting inpainting network uses the target proxy to inform where the

remaining shadows in the scene should continue behind the removed object. An off-the-shelf inpainting method is used for texture inpainting. Inpainting the decomposed texture, rather than the final composite, prevents the inpainting method from hallucinating its own shadows.

- Lastly, the lighting and texture images are recomposed back together to produce the final result.

Let I, I' be the input and output images, respectively. P, P' are the shadow proxy and target proxy. The intrinsic decomposition is denoted by $I = LT, I' = L'T'$ with L, L' being the lighting images and T, T' being the reflectance (texture) images. M_o is a binary mask which is 1 for pixels lying on the object to be removed and 0 elsewhere. M_r is a binary receiver mask which is 1 for pixels lying on the local scene receiving the shadows, and 0 for pixels elsewhere. This restricts the network to operate on a surface with a single texture, as otherwise the intrinsic decomposition frequently mislabels shadows as changes in reflectance if the surface reflectance can vary arbitrarily.

RGB images are processed in the log domain, turning the intrinsic decomposition $I = LT$ into a sum $\log(I) = \log(L) + \log(T)$ that is more naturally represented by CNNs. Using synthetic training data enables full supervision of each subnetwork's intermediate outputs. Examples of the predicted intermediates are shown in Figure 7.6.

Shadow Segmentation: This subnetwork produces a 1-channel soft segmentation in the range $[0, 1]$ with 1 denoting full shadow.

$$S = f_{\text{SS}}(\log(I), \log(P), M_r) \tag{7.1}$$

Intrinsic Decomposition: This subnetwork decomposes the input image into two 3-channel outputs: lighting L and texture T.

$$L, T = \exp(f_{\text{ID}}(\log(I), \log(P), S, M_r)) \tag{7.2}$$

Shadow Removal: This subnetwork removes the shadow of the object from the predicted lighting image, producing one 3-channel output, the masked target lighting.

$$L'_r = \exp(f_{\text{SR}}(\log(I), \log(T), \log(L), \log(P), \log(P'), S, M_r, M_o)) \tag{7.3}$$

Lighting Inpainting: This subnetwork fills in the predicted lighting L'_r behind the removed object, continuing shadows cast by other objects through the mask if necessary. The target lighting L' is then the composite of the inpainted lighting and the masked target lighting.

$$L'_o = \exp(f_{\text{LI}}(\log(L'_r), \log(P'), M_o)) \tag{7.4}$$

$$L' = (1 - M_o)L'_r + M_o L'_o \tag{7.5}$$

Texture Inpainting: The texture image is inpainted with an inpainting operator $g(T, M_o)$, synthesizing the pixels of T in the hole region specified by the mask image M_o. In this thesis, HiFill [151] was used for $g(T, M_o)$, trained on the Places2 dataset [162].

$$T' = g(T, M_o) \tag{7.6}$$

Final Composite: The previous stages predicted the appearance of the receiver within the target receiver mask $M'_r = M_r + M_o$. The remaining pixels from the original image, consisting of unaffected surfaces beyond the local scene and other objects within the local scene, are then composited into the result.

$$I' = M'_r T' L' + (1 - M'_r)I \tag{7.7}$$

7.4.1 Training

Each subnetwork is independently trained with the Adam optimizer for 60 epochs on 60000 training scenes, with ground truth intermediates substituted for the outputs of earlier subnetworks with a learning rate of 10^{-4} decaying by 0.5 every 10 epochs. The whole system is then trained end-to-end for 60 epochs. The networks were implemented in Tensorflow and trained on four Tesla V100 GPUs with a batch size of 16.

The loss functions used in training are described below. For all networks except the lighting inpainting network, the inputs to the losses are masked with M_r to only apply to pixels lying on the receiver. For notational brevity, assume that the norm flattens across input channels and image dimensions. A ground truth supervision image is denoted with a hat, so that the ground truth intrinsic decomposition is $\hat{L}, \hat{T}$, ground truth output image is $\hat{I}'$, and so on.

Shadow Segmentation: It is difficult to define a ground truth for what constitutes a shadow in a scene lit by an HDRI map, since any object will occlude some part of the distant illumination. Supervising this stage with synthetic training data involves examining the ratio of pixel values between the ground truth lighting with all the objects compared to the ground truth lighting of only the receiving surface. A shadow is defined as any pixel where this ratio is less than the median ratio on any of the three color channels using a soft threshold:

$$\hat{S} = \max\left(\sigma\left(\frac{\text{median}(\hat{L}/L_r) - \hat{L}/L_r}{\alpha}\right)\right) \tag{7.8}$$

The shadow segmentation subnetwork is supervised by a class-balanced binary cross entropy term as well as a loss on the gradients of the shadow segmentation:

$$E_{\text{SS}} = \lambda_S E_S + \lambda_{\nabla S} E_{\nabla S} \tag{7.9}$$

$$E_S = \frac{-\hat{S}\log(S)}{||\hat{S}||_1} - \frac{(1-\hat{S})\log(1-S)}{||1-\hat{S}||_1} \tag{7.10}$$

$$E_{\nabla S} = ||\nabla S - \nabla \hat{S}||_2 \tag{7.11}$$

Intrinsic Decomposition: The intrinsic decomposition loss function is the most involved of the losses, including a data term, two terms for the decomposition, and a data prior.

$$E_{\text{ID}} = \lambda_{LT} E_{LT} + \lambda_{\text{excl}} E_{\text{excl}} + \lambda_I E_I + \lambda_{\nabla L} E_{\nabla L} \tag{7.12}$$

For the data term, a multiscale loss on the predicted lighting and texture images was vital to ensure the model would work well on high-contrast textures.

$$E_{LT} = P(L, \hat{L}) + P(T, \hat{T}) \tag{7.13}$$

where $P(X, \hat{X})$ is an L2 loss on a Gaussian pyramid decomposition of the images $X, \hat{X}$.

The exclusion losses of Zhang *et al* [160] ensure the predicted lighting and texture images form a clean decomposition. The exclusion losses construct 0-to-1-valued edge maps at multiple scales, and penalize edges lying at the same location in the two decomposed images.

$$E_{\text{excl}} = \sum_{i=0}^{i=3} 4^i ||\Psi(T \downarrow i, L \downarrow i)|| \tag{7.14}$$

where $X \downarrow n$ denotes image X downsampled bilinearly by a factor of 2^n, and Ψ is as defined by Zhang et al.

An L1 loss is also applied to the two decomposed images recomposing into the input image.

$$E_I = ||I - LT||_1 \tag{7.15}$$

Finally, a sparse gradient prior is imposed on L to discourage textural details from leaking into the lighting.

$$E_{\nabla L} = ||\nabla L||_1 \tag{7.16}$$

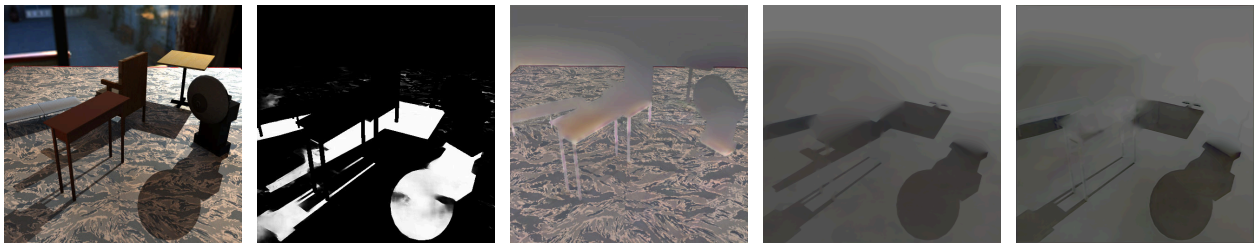
Shadow Removal and Lighting Inpainting: As with the intrinsic decomposition, a multiscale loss is used on the predicted lighting after shadow removal and inpainting.

$$E_{L'} = \lambda_{L'} P(L', \hat{L}') \tag{7.17}$$

Note that because L' is a composite of the results of the shadow removal and lighting inpainting networks, the shadow removal network is only penalized for pixels lying on the receiver in the original input image while the lighting inpainting network is only penalized for pixels within the mask of the removed object.

The final output is also supervised with a recomposition loss.

$$E_{I'} = \lambda_{I'} ||\hat{I}' - L'T'||_1 \tag{7.18}$$



(a) Input (b) Predicted Shadow Mask (c) Predicted Texture (d) Predicted Lighting (e) Predicted Target Lighting

(f) Input (g) Predicted Shadow Mask (h) Predicted Texture (i) Predicted Lighting (j) Predicted Target Lighting

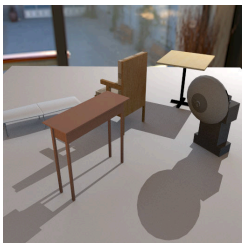
Figure 7.6: Predicted intermediates for synthetic and real scenes.

7.5 Training Data

Capturing a large set of paired images with and without a particular object would require a prohibitive amount of labor. Therefore, the system is trained on a synthetic dataset, which can also generate ground truth for the intermediate stages such as the intrinsic decomposition.

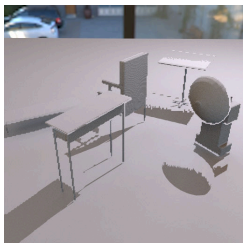
For training data, 60000 input scenes were generated with randomly selected geometry, textures, lighting, and camera parameters. These scenes were rendered using PBRT [104] to produce images of resolution 512×512. The dataset exhibits a wide variety of shadow casters (e.g large objects, thin structures, and objects with unusual silhouettes) and lighting conditions (hard or soft shadows, very dark or very subtle shadows, multiple shadows). A full example of a synthetic scene from the test set, including all generated intermediates and the scene proxy, is shown in Figure 7.7.

(a) Input I



(b) GT Lighting $\hat{L}$

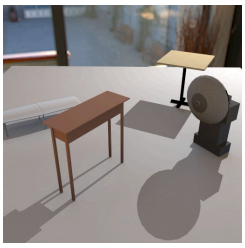
(c) GT Shadow Mask $\hat{S}$



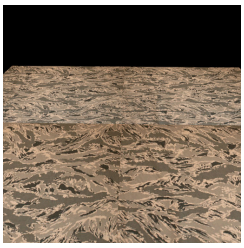
(d) Shadow Proxy P

(e) Object Mask M_o

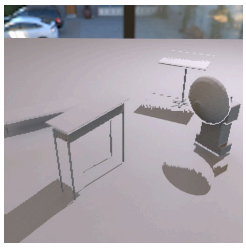
(f) GT Output $\hat{I}'$



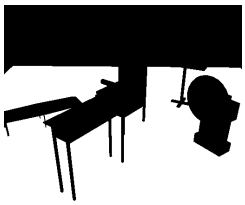
(g) GT Target Lighting $\hat{L}'$



(h) GT Texture $\hat{T}$

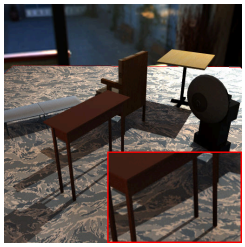


(i) Target Proxy P'



(j) Receiver Mask M_r

Figure 7.7: Synthetic scene components for a test scene, including network inputs and ground truth intermediates.



(a) Input I

(b) Ratio Image P/P'

(c) Differential Rendering Result I'_{diff}

(d) Our Result I'

(e) Ground Truth $\hat{I}'$

Figure 7.8: Differential rendering on a synthetic scene, with area of interest shown in the red inset. The incomplete geometry causes the proxy model to underestimate the extent of the object's shadow.

7.5.1 Scene Generation

Geometry: The generated scenes consist of a ground plane supporting six to seven objects randomly selected from the ~50000 3D models in the ShapeNet [16] dataset, which include a variety of object classes ranging from furniture and tableware to cars and airplanes. These objects are arranged in a ring around a central object, and are scaled such that the bounding boxes are nonintersecting. Each object is translated such that it lies entirely on top of the plane, and has a random rotation around its up axis. The ground plane is large enough to support all the shadow casters, plus an additional margin for shadows to potentially fall upon.

Materials: The supporting plane is given a matte material, and is assigned a random texture (*e.g* carpet, wood, stone, tile). Existing texture datasets are too small (*e.g* the Brodatz textures [130]) or have textures which are nonuniform (*e.g* the Describable Textures Dataset [19]). A manually curated texture dataset of about 8000 images from Google Image Search results was used instead. The ShapeNet objects come with prespecified materials.

Lighting: Each scene is illuminated by one of the ~400 HDRI maps at HDRI Haven[2], randomly rotated around the up axis. To supplement the lighting, a point light with random intensity (setting the maximum to the peak intensity of the HDRI map) is randomly placed between a minimum and maximum distance from the center of the plane, in the upper hemisphere.

Camera: A range of suitable camera positions was generated, lying on an upper hemisphere of fixed radius in terms of spherical coordinates facing the center of the scene. The azimuthal angle can vary freely, but a minimum and maximum elevation angle are enforced (as people rarely observe scenes from directly overhead or from very low angles). After selecting an initial camera pose, the camera's position is then perturbed while keeping the same orientation.

7.5.2 Image Generation

PBRT is used to render three RGB images of each scene: $\hat{T}$, the ground plane alone with diffuse texture; $\hat{L}$, the complete scene with the plane material replaced by a diffuse white material; and

[2]https://hdrihaven.com/

$\hat{L}'$, the scene with the central object removed with the same alteration to the plane material. These images comprise the ground truth intrinsic decomposition's texture and lighting, and the lighting post-object-removal. Note that these images do not form a true intrinsic decomposition of the entire scene, only of the receiving plane. To aid in our intrinsic decomposition, L_r is rendered as the ground plane alone with no texture, capturing the unshadowed appearance of the receiving surface.

Next, depth maps D, D' of the unedited and target scenes are rendered, as well as a depth map D_r of solely the ground plane receiving shadows, all using the same camera pose as the RGB images. From these a pixel mask of the object to be removed $M_o = \mathbb{I}(D' < D)$ is computed to be 1 where the object is and 0 everywhere else. A receiver mask $M_r = \mathbb{I}(D_r \neq \infty, D_r = D)$ is also generated, which is 1 for pixels lying on the ground plane in the unedited scene and 0 everywhere else.

To allow for further augmentation, the input and output images $I, \hat{I}'$ are not directly raytraced and are instead computed at train time from the decomposition: $I = (M_r\hat{T} + (1 - M_r))\hat{L}$ and $\hat{I}' = (M_r'\hat{T} + (1 - M_r'))\hat{L}'$. This enables modification of the hue, saturation, and brightness of texture and lighting at train time. Note that this process forgoes indirect bounce lighting in the synthetic data to enable this augmentation, as indirect illumination depends on the surface reflectance, *i.e* texture.

To mimic real capture, noise is added to the depth map of the unedited scene before a triangle mesh is constructed from the depth map to form the approximate geometry, replacing the ground plane vertices with a best-fit plane (which continues behind the removed object). To form the target proxy geometry, the depth pixels occupied by the removed object are deleted. This scene is lit with a perturbed version of the input lighting: the point light's position, color, and intensity are jittered, a random nonlinear scale is applied to the HDRI map, and the HDRI map is randomly rotated by a small amount. All materials are set to a diffuse white; note that the surface reflectance (texture) of the plane cannot be modeled as it would imply already knowing the intrinsic decomposition. Rendering these elements produces P, P', respectively the images of the unedited and target scene proxy.

7.5.3 Normalization

The intrinsic decomposition has a scale ambiguity, which is resolved by normalizing the ground truth lighting $\hat{L}, \hat{L}'$ at train time, expecting that the network will produce normalized lighting images. Specifically, a per-channel scale is applied to both $\hat{L}, \hat{L}'$ such that the maximum pixel value on the receiver across both images is $(1, 1, 1)$.

Similarly, a scale factor is computed to normalize the images of the scene proxy P, P' (which are just approximations of $\hat{L}, \hat{L}'$). This occurs at both test and train time.

For both train and test the images in the input domain (i.e. I and $\hat{I}'$) are scaled to have a channelwise mean pixel value of 0.5 on the ground plane.

7.5.4 Augmentation

Further augmentations of the input data include:

- Randomly flipping inputs left-to-right.
- Randomly replacing textures with a solid color.
- Randomly desaturating textures.
- Randomly shifting hues of textures.
- Randomly tinting the lighting image (i.e. apply a random scale and bias before renormalization). Because of the lighting normalization, this really means randomly tinting the shadows.
- Randomly dropping out the scene proxy renderings, replacing them with black images.

7.6 Results

I evaluate this differential rendering method both qualitatively and quantitatively on 5000 synthetic test scenes, generated the same way as the training data, and 14 real scenes captured manually, which included ground truth object removal results.

	Synthetic			Real		
	RMSE	Shadow RMSE	Inpaint RMSE	RMSE	Shadow RMSE	Inpaint RMSE
No-op	0.0455	0.2785	0.3755	0.0405	0.1413	0.2702
PatchMatch	0.0460	0.2790	0.2565	0.0401	0.1378	0.1288
HiFill	0.0479	0.2700	0.2555	0.0408	0.1305	0.1160
PatchMatch + Shadows	0.0402	0.2143	0.2282	**0.0351**	0.0969	0.1025
HiFill + Shadows	0.0461	0.2193	0.2346	0.0365	0.0855	**0.0993**
Pix2Pix (all)	0.3583	0.3243	0.4477	0.2502	0.2649	0.3129
Pix2Pix (receiver)	0.0820	0.2118	0.2323	0.1091	0.1526	0.1266
Pix2Pix + Proxy (receiver)	0.0802	0.1766	0.2244	0.0872	0.1187	0.1153
+Intrinsic Decomposition	0.0362	0.0882	0.2238	0.0618	0.0843	0.1055
+Shadow Segmentation	**0.0246**	**0.0713**	0.2213	**0.0340**	0.0631	0.1040
+Lighting Inpainting (Ours)	**0.0248**	**0.0712**	**0.2143**	**0.0340**	**0.0616**	**0.0983**

Table 7.1: Comparison of error rates for various shadow removal methods

For quantitative results, I report three separate RMSE metrics. The basic RMSE measured across all pixels in the scene is a poor representation of the quality of shadow removal results. A perceptually negligible color cast produced by a deep network across the entire image has an outsized effect on the RMSE. To better represent the performance of various systems, I also report the Shadow RMSE, which is computed across pixels within the ground truth binary shadow mask $\hat{S}$. Finally, I separately report the RMSE within the removed object's pixels.

7.6.1 Test Data

For the real scenes, the proxy geometry and the images were captured using the RGBD Kinect v2 device mounted on a tripod. I captured three frames for each scene: I the complete scene, $\hat{I}'$ the target image with one or more objects removed, and a bare scene with all objects removed. The approximate lighting was captured using a Ricoh Theta S 360 camera with 5 exposures for high dynamic range placed approximately in the center of the scene, roughly pointed at the Kinect.

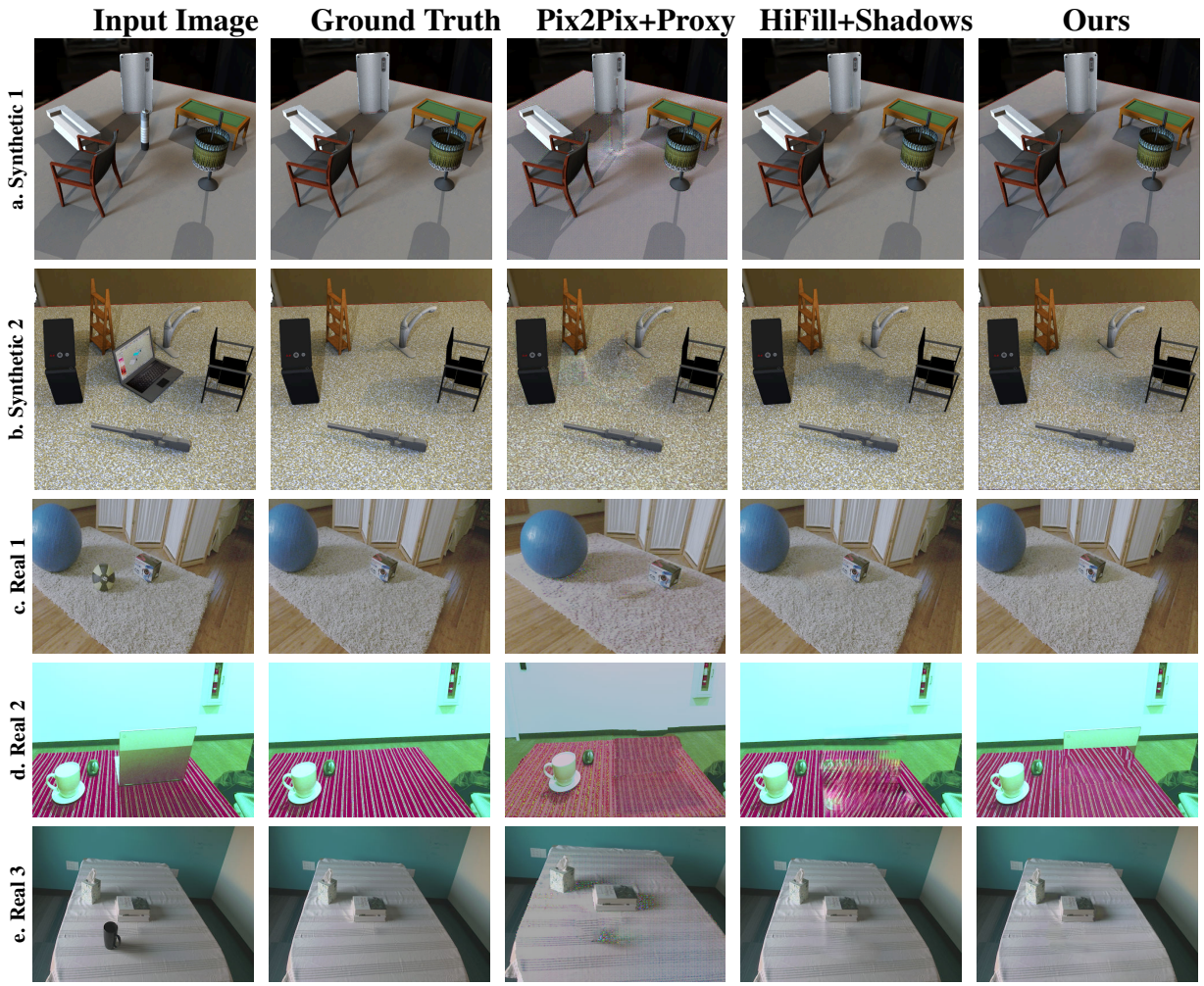
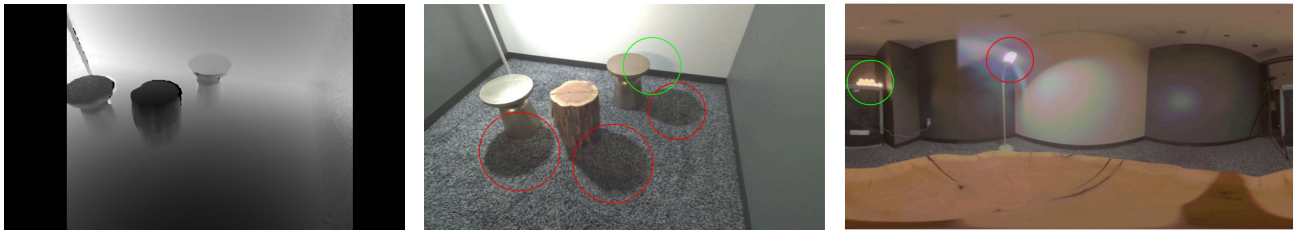


Figure 7.9: I compare my system to two object removal baselines on both synthetic and real test images. The first baseline is an image-to-image translation network based on Pix2Pix [58] which is supplied with our renderings of the proxy scene. The second baseline is HiFill [151], a state-of-the-art inpainting method, that inpaints both the removed object and an explicitly specified shadow mask.

The depth images were median-filtered to remove noise. The best-fit plane for the input depth image was computed using RANSAC [33], and the receiver mask M_r was computed to indicate pixels which approximately lie on the plane. For this work I manually specified the object masks M_o; in real applications a more sophisticated automatic segmentation method could be used.



(a) Depth Map D (b) Input Image I (c) HDRI Environment Map

(d) Geometry (top view) (e) Shadow Proxy P (f) Target Proxy P'

Figure 7.10: Example of a rough proxy model for a real scene. The raw inputs are shown in the top row, where I and D come from the RGBD sensor and the HDRI environment map comes from a 360 camera. The HDRI environment map (shown tonemapped) contains two light sources: a nearby lamp just visible in the input image that causes most of the shadows (red), and a set of LED lights further away that just barely cast a visible shadow (green). A top view of the proxy meshes of the three stools obtained from the depth sensor (the ground plane and wall geometry is omitted for clarity). The scanned geometry is not only incomplete, but also distorted, especially around the metal stools. The shadows in the proxy images P, P' are too rough to use directly (as shown in Figure 7.4, but they are good enough for the shadow removal system to remove the necessary shadows.

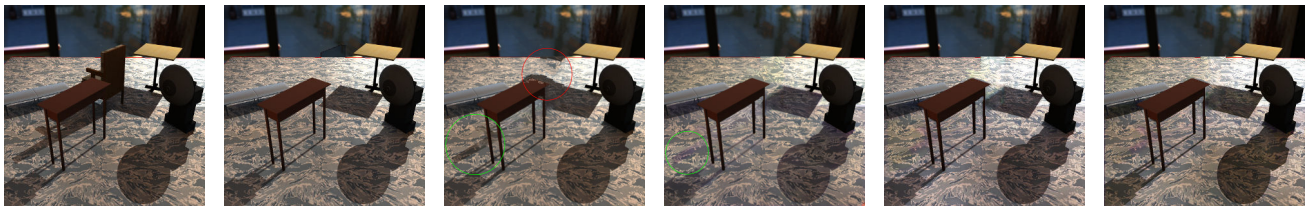
To compute the proxy geometry, a triangle mesh was formed from the depth map similar to the process used for synthetic data; however, for cleaner shadows, vertices comprising the ground plane were replaced with a fitted plane. This geometry was then rendered with the captured HDR environment lighting to produce P, P'. An example of a real proxy model is shown in Figure 7.10. As the ground truth intrinsic decomposition was unknown, the difference between I and $\hat{I}'$ was used instead of $\hat{L}$ and $\hat{L}'$ to produce the shadow mask $\hat{S}$ used in the Shadow RMSE metric.

7.6.2 End-to-End Comparisons

Most existing works on object removal do not focus on removing shadows cast by the object. I compared this method to two general approaches as baselines: pure inpainting and generic image-to-image translation. I also computed the numerical error of the "no-op" procedure, which does not transform the input at all. Quantitative results are shown in Table 7.1 and qualitative results in Figures 7.9,7.15,7.16, exhibiting varying lighting conditions (multiple overlapping shadows, soft and hard shadows, high and low contrast shadows) and background textures in both synthetic and real test scenes.

The inpainting baselines were realized using two methods: the classical nonparametric PatchMatch [4], and a recent learning-based approach HiFill [151]. For both baselines, I show quantitative results both for a naive hole-fill, where the object's shadows are not handled at all, as well as for an inpainting mask which includes the object's shadows. In Figure 7.9 I show the results of HiFill inpainting when provided the *ground truth* shadow region. With this extra information, simple inpainting approaches worked well for simple textures (7.9a,7.9c), but often hallucinated shadows within the inpainted region (7.9b). They also failed to take advantage of texture detail inside the shadow region, and the resulting artifacts were compounded when the region to inpaint was large (7.9d). Note that, unlike my method, none of the inpainting baselines use the proxy renderings.

For the image-to-image translation baseline, I used the well-known Pix2Pix method [58]. I compared against three variants of this baseline: one trained to predict the entire output image I' from the input I and object mask M_o; one trained to predict only the appearance of the receiving surface (using HiFill to inpaint the object region); and one trained to predict only the appearance of the receiver but supplied with the proxy scene renderings P, P' in addition to the input image and object mask. I show the results of this last version in Figure 7.9. The method fails to accurately identify the extents of shadows (7.9c,7.9e) and their intensities (7.9a) and generalizes poorly to complex textures (7.9b,7.9d).



(a) Input (b) Ground Truth (c) Pix2pix (e.g. U-net) (d) +Intrinsic Decomposition (e) +Shadow Segmentation (f) +Lighting Inpainting

Figure 7.11: A synthetic example (a) shows the importance of each component of the shadow removal system. (b) shows the ground truth removal. A single U-net (c) hallucinates shadows in the inpainted region (red) and misidentifies shadows (green) with high-contrast textures. Adding in a texture decomposition subnetwork significantly improves the inpainting (d); however shadow edges are still faintly visible (green). The shadow segmentation subnetwork eliminates these shadow ghosts (e). Some final artifacts around the silhouette of the removed object are fixed with the lighting inpainting network (f).

7.6.3 Architecture Validation

I show the importance of each step of this pipeline, starting with a single network to perform the generalized differential rendering task and adding in each component one by one. The results are shown in Figure 7.12 and the bottom rows of Table 7.1. The baseline system starts with a single Pix2pix U-Net generator, that given all the inputs, directly predicts the output image excluding the pixels under the object mask, which are inpainted using HiFill (Figure 7.11c). The most obvious artifacts are within the inpainted region, where the inpainting method frequently fills in shadow pixels; this method also frequently misidentifies shadows, especially in high-contrast textures. A separate intrinsic decomposition network is then introduced, so that the shadow removal network works only on the resulting lighting image (Figure 7.11d). While results are greatly improved, sometimes this system does not completely remove hard or high-contrast shadows. Introducing a shadow segmentation network (Figure 7.11e) makes decompositions of hard shadows much cleaner. Finally, a dedicated lighting inpainting network (Figure 7.11f) is introduced, as the shadow removal network alone has trouble continuing shadows behind the object and sometimes

	Synthetic			Real		
	RMSE	Shadow RMSE	Inpaint RMSE	RMSE	Shadow RMSE	Inpaint RMSE
Ours	**0.0248**	**0.0712**	**0.2143**	**0.0340**	**0.0616**	**0.0983**
No Sparse Gradient Prior	0.0290	0.0783	0.2161	0.0586	0.0673	0.0991
No Exclusion Losses	0.0292	0.0792	0.2182	0.0375	0.0683	0.0990
No Multiscale Loss	0.0277	0.0926	0.2185	0.0355	0.0672	0.0997

Table 7.2: Comparison of shadow removal error rates under various loss function ablations

leaves visible artifacts in the hole region.

7.6.4 Ablation Study on Loss Terms

I show the importance of several loss terms in Table 7.2 and Figure 7.12. The multiscale loss was the most important loss term – replacing it with an L1 loss caused many shadows to be left behind, resulting in a large drop in performance on the Shadow RMSE metric. The sparse gradient prior is important to maintain texture fidelity, as without it texture details are sometimes assigned to lighting, and subsequently get removed as shadows. The exclusion losses perform a similar role but also prevent shadows from remaining in the texture image.

7.6.5 Comparison of Intrinsic Decomposition Subsystem

Figure 7.13 shows two representative results comparing the intrinsic decomposition subsystem of this work to recent systems for shadow removal (Zou *et al* [168]) and intrinsic decomposition (Li and Snavely [81] with author's weights).

Unlike the described differential rendering system, these methods do not use proxy renderings, which provide significant benefit. Moreover, shadow removal systems are generally trained on SRD[107] and ISTD[132], which are greatly restricted in lighting conditions. Furthermore, images from these datasets do not include the objects that cast the shadows. Shadow removal approaches based on these datasets do not generalize at all to the full scenes with more diverse lighting shown

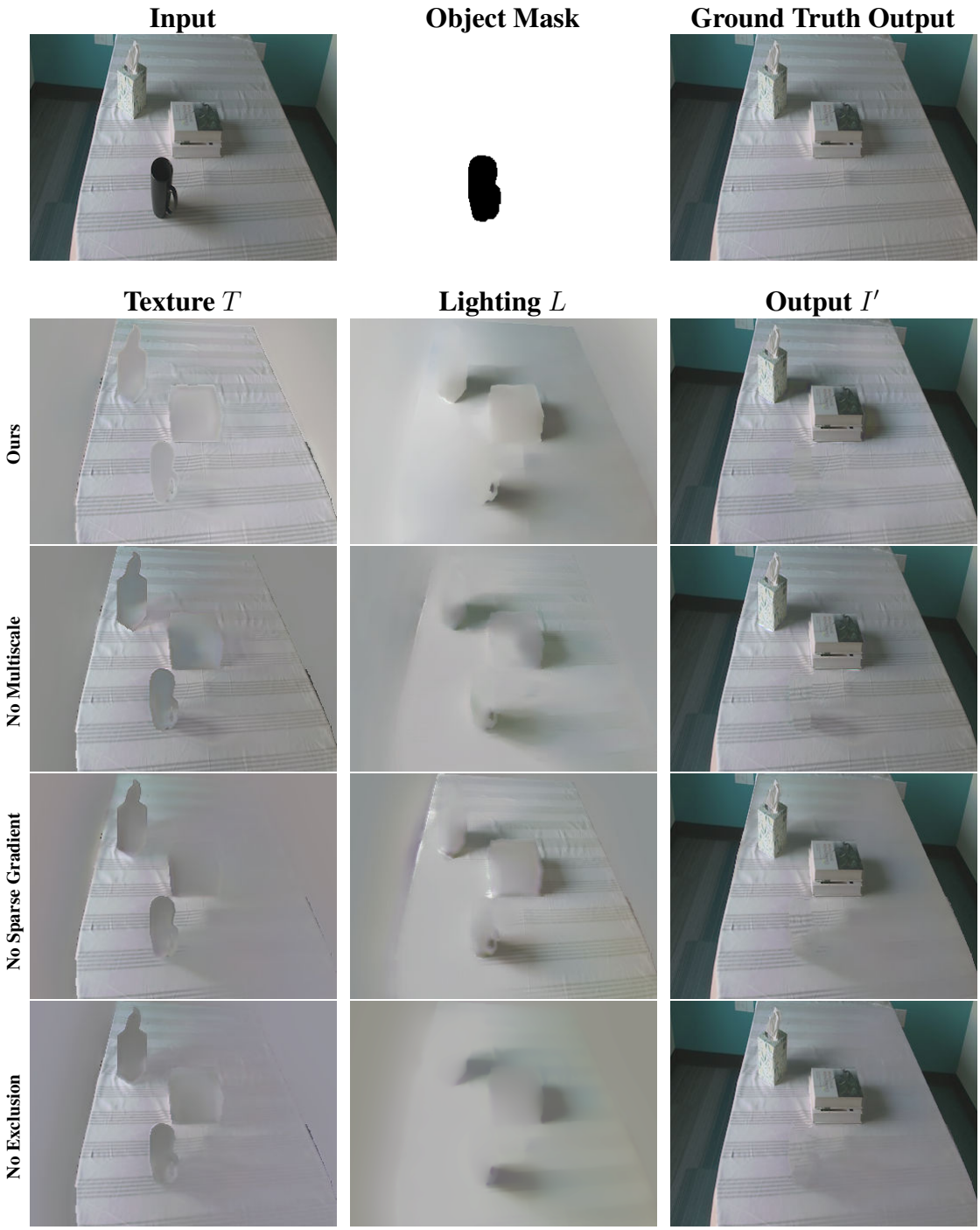


Figure 7.12: Effects of various loss terms on intrinsic decompositions. The multiscale loss results in better identification of the extents of shadows. The sparse gradient prior helps keep texture out of the lighting image (so that the texture is not removed by the shadow removal network). The exclusion losses perform a similar role but also prevent shadows from remaining in the texture image.

Figure 7.13: Decomposition of synthetic (top) and real (bottom) scenes. From left to right: input, shadow removal using Zou *et al* [168] trained on ISTD, shadow removal using Zou *et al* trained on our dataset, intrinsic image reflectance using Li and Snaveley [81], our reflectance. Note that the three compared methods only operate on the input image, while the method described in this chapter additionally receives a rendering of a rough proxy model.

in this chapter. Existing intrinsic decomposition works have slightly more diversity in data, and are trained either on images of entire rooms (e.g. IIW[9]) or on single objects (e.g. MIT[45]). However they typically focus on removing lighting variation due to surface orientation and do not generalize to cast shadows. The complexity of lighting conditions, shadow appearance, and textural variation in the datasets used for this thesis show the necessity of more diverse training data, as well as careful choices of losses to effectively deal with the challenges of this more diverse data.

7.6.6 Further Results on Real Data

This shadow removal enables much more realistic mixed reality applications, ranging from consumer applications in real estate and furniture retail, to socially beneficial uses for understanding physical resource allocation in environments such as hospitals and schools. As an example, Figures 7.1 and 7.14 show results for a refurnishing scenario. To generate these results, the pipeline is run twice – once for the wall, and once for the floor. The two results are then composited together,

Figure 7.14: Virtual refurnishing applications. Left: input images. Right: refurnished results.

and a virtual object is inserted using standard differential rendering.

Figure 7.15: Additional results of shadow removal on real scenes. Removed object(s) are indicated by the red arrows.

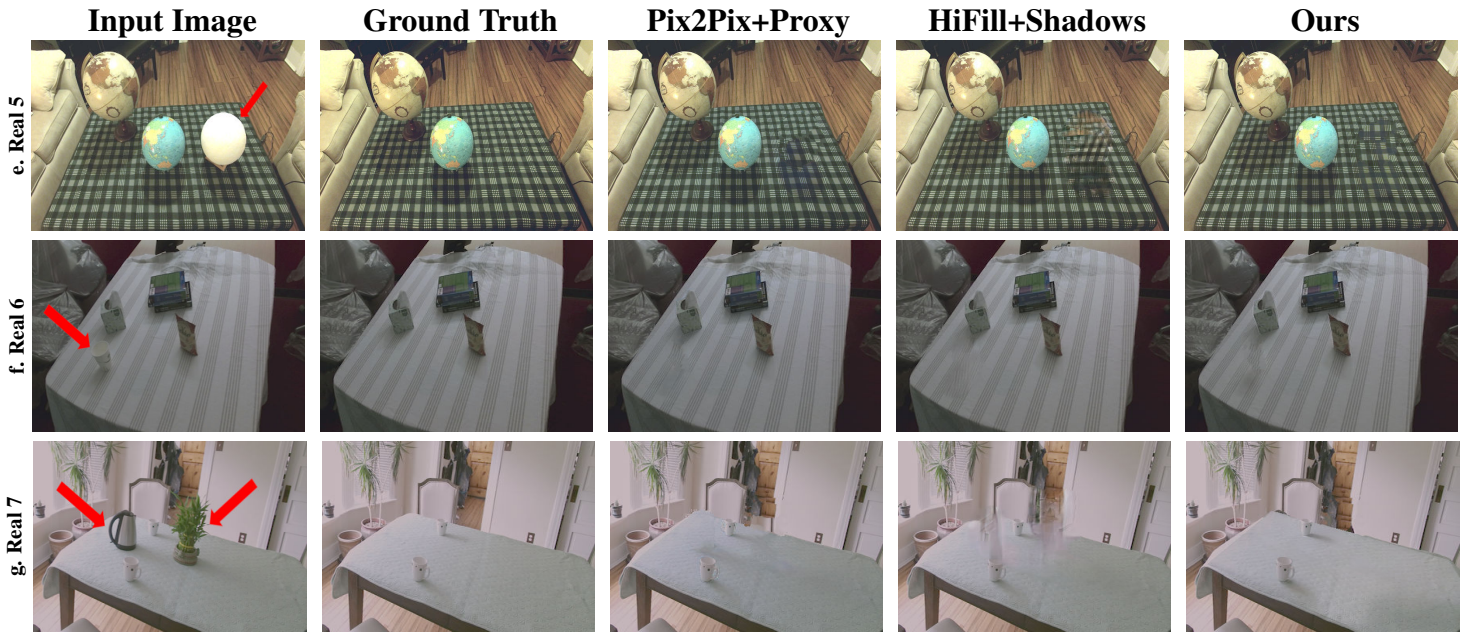


Figure 7.16: Additional results of shadow removal on real scenes. Removed object(s) are indicated by the red arrows.

7.6.7 Limitations and Future Work

Perceptually, this method is consistently better at removing shadows from receiving surfaces than either baseline (especially on the high-contrast texture in Figure 7.16e), and its decomposition-based inpainting scheme results in fewer artifacts in the inpainted regions. However, it still has some limitations. Some overall color shifts (most evident in Figures 7.15b,7.16g) cause large quantitative effects on the results, but perceptually are less evident than leftover shadows.

In the second row of Figure 7.14, the glossy hardwood floor shows a specularity of the couch that is to be removed; by adding a specular component to the proxy model's floor plane, the pipeline is able to remove the specularity as well as the shadow. Of course, this specularity is fairly simple, where the couch "occludes" the reflection of the much brighter white wall. Since glossy surfaces are present in many indoor scenes, handling more complex specularities is an important area for further investigation. In this vein, handling higher order light transport effects, such as color bleeding, is also important for realistic results.

Several other artifacts are visible in Figure 7.14. A faint outline on the left side of the removed couch can be seen; this is due to the light from the window casting nonuniform illumination onto the wall, which was not represented in the training data. The baseboard is also partially removed, even in regions where it was visible in the input image near the piano leg. Built into the training data is the assumption that the texture is consistent over the entire receiving surfacce. Without this restriction, the intrinsic decomposition network tends to incorrectly assign shading variation to the texture image. For similar reasons, the system also does not consider shadows cast on the other objects in the scene and focuses on a single receiver. Extending the pipeline to handle multiple receivers, or using different representations of textures, would be important additions to this work.

Chapter 8

CONCLUSION

This thesis presents several methods to capture and recover the geometry, lighting, and materials for indoor scenes. I show how these methods enable realistic edits to the scene such as relighting, object removal, and object insertion. Unlike previous works, I focus on scene-scale reasoning about lighting and light propagation, considering local light emitters and indirect lighting. Furthermore, several of my methods are designed to apply to realistic object removal, which has not been addressed by prior work. This thesis describes the following contributions:

- A method for computing globally consistent high dynamic range textures for a scanned scene mesh from autoexposed low dynamic range images. This optimization framework, described in Chapter 3, provides the radiometrically correct data necessary for accurate lighting inference.

- A set of methods based on Manhattan-world assumptions for recovering the raw geometry of an indoor room, including not only floors, walls, and ceilings, but also architectural features such as doors, windows, and baseboards. These methods are described in Chapter 4.

- A set of parametric models for light emitters, described in Section 5.5. These models are diverse enough to realistically represent the illumination in a wide range of real scenes, but are simple enough to optimize for.

- A framework for scene-scale inverse rendering, described in Chapter 5, that solves for diffuse reflectances simultaneously with lighting intensity parameters for the aforementioned lighting models. Unlike previous works, this framework takes into account scene-scale lighting

effects such as interreflections and occlusions, and ultimately enables realistic visualizations of emptying, refurnishing, and relighting the scene.

- A definition of the **light localization problem** in Chapter 6, which asks, given only a scene's appearance, materials, and geometry, can the number and positions of the light sources illuminating the scene be determined?

- A derivation from first principles of a novel set of scene transforms that can be manipulated to produce a voting scheme for likely light emitter positions. Together with an iterative refinement algorithm, these transforms provide one way to attack the light localization problem. These transforms and algorithms are described in Chapter 6.

- A neural rendering method to compensate for the inaccuracies in a recovered model when performing scene edits, specifically for object removal operations. Motivated by work in differential rendering, this method, described in Chapter 7, ensures that the shadows an object casts are removed with that object.

8.1 Future Work

The work described in this thesis enabled exciting, realistic scene editing results that had not been achieved by previous systems. These results show especially that a) working with expressive models of local light sources as well as b) compensating for inaccuracies in scene models are important areas for further research to enable more generalizable approaches in scene editing. In this section, I describe potential future work based on these crucial observations.

8.1.1 Inverse Rendering and Local Light Sources

My work in parametric emitter models and light localization has highlighted the importance of reasoning about local light sources, but there is much work remaining in inferring the properties of more general light emitters.

In this thesis, I outlined two methods for lighting inference. One assumed known lighting positions and simultaneously solved for light intensity distributions and diffuse reflectance parameters, while the other assumed known materials and solved for isotropic point light positions. An ideal inverse rendering system would be able to simultaneously solve for both lighting intensity distributions and positional parameters, as well as infer more general material properties beyond diffuse surfaces.

This high dimensional optimization problem is severely underconstrained, and so will likely require sophisticated priors and new techniques, perhaps based on the Intensity Distance Transform I have developed, to work in real scenes. Modern machine learning methods are a promising way to encode data-driven priors that can represent likely configurations of light positions and intensity distributions, as well as material properties. Furthermore, unlike the physically based methods I described in this thesis, learned models will be able to consider scene surfaces in context – rather than isolated points with radiances, learned models can reason about entire objects at a time.

8.1.2 Realtime Scene-Scale Inverse Rendering

The methods for inverse rendering described in this thesis are offline methods, requiring on the order of minutes to hours to process a real world scene. However, users of augmented reality applications will ideally be able to edit scenes interactively, without lengthy capture and processing steps.

Performance optimization was not a focus of the works in this thesis, and several processes could be easily parallelized or simplified to run on mobile hardware. For example, the exposure optimization in Chapter 3 and the inverse rendering processes in Chapters 3 are both simple bilinear optimization problems of a scale similar to those used in the global pose optimization in BundleFusion [26], which occur in an online fashion. However, much work remains in analyzing the light localization pipeline in Chapter 6 before it can reasonably be brought into realtime.

In addition to the inverse rendering processes, the editing and rendering processes also need to occur interactively. In this thesis, the edits were painstakingly specified manually; designing interfaces to allow users to easily specify scene edits will be a vital part of actual mixed reality

applications. Finally, visualizing these edits in a way that takes full advantage of the reconstructed lighting and material parameters will also require further work into realtime photorealistic rendering. Recent work using feed-forward deep neural networks for rendering, including the work described in Chapter 7, are a promising way to lessen or entirely bypass the need for full simulations of light propagation.

8.1.3 Generalized Deep Differential Rendering

In Chapter 7 I extended the concept of differential rendering, compensating for the inaccuracies in a reconstructed scene model, to object removal. This concept has not been explicitly explored, but has wide applications for any type of scene manipulation. Any method that reconstructs a scene in order to perform edits on it can benefit from a more general method of compensating for inaccuracies in the model, whether they come from errors in reconstruction, insufficiently general models, or low-quality input data.

I envision a single differential rendering network that can handle any types of scene edits, including object insertion, object removal, and relighting, based on renderings of a reconstructed proxy model. Designing such a network is much more practical than attempting to construct a single deep model that represents, edits, and renders the scene. This is because, using renderings, the network can operate solely in 2D, leaving the reasoning about photorealism and light propagation to the rendering process.

Instead of rendering a model of a scene and then correcting it using neural rerendering, one nascent approach that has seen some promise is to construct an encoder-decoder model with a latent space that is itself editable, whether explicitly [99] or implicitly [7], using the trained decoder to render edited results. While current works in this vein lack the quality necessary for photorealistic applications, they show great potential in enabling more general manipulations.

8.1.4 Enabling Other Types of Scene Edits

In this thesis I showed examples of object removal, object insertion, relighting, and retexturing. However, manipulating existing objects, as is done in [18, 67], is much more challenging. One might perform object manipulation operations by first removing the object in question, then reinserting it in its new configuration. Not only does this require realistic object removal and object insertion, it also requires the accurate reconstruction of the manipulated object, including both the geometry and appearance of hidden surfaces.

Fortunately, novel view synthesis and object reconstruction are both very popular areas of research in the vision community. While the reconstructions of objects are currently of rather modest quality, neural differential rendering methods such as the one I describe in Chapter 7 can help compensate for this and make the results more realistic. Of course, the needs of an network for object manipulation are very different than one for object removal; further investigation into neural rerendering for different editing applications is therefore also important for satisfying results.

BIBLIOGRAPHY

[1] Jens Ackermann, Simon Fuhrmann, and Michael Goesele. Geometric point light source calibration. In *Symposium on Vision, Modelling, and Visualization*, 2013. 54

[2] Sameer Agarwal, Keir Mierle, and Others. Ceres Solver. 13, 35, 71

[3] Nina Amenta and Marshall Bern. Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry*, 22(4), 1999.

[4] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *SIGGRAPH*, 2009. 8, 82, 101

[5] Jonathan T Barron and Jitendra Malik. Intrinsic scene properties from a single rgb-d image. In *CVPR*, 2013. 7, 31, 82

[6] Jonathan T. Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. In *TPAMI*, 2015. 32, 82

[7] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. In *SIGGRAPH*, 2019. 84, 113

[8] Matt Bell and ET Freeman. Learning local evidence for shading and reflectance. In *ICCV*, 2001. 83

[9] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. In *SIGGRAPH*, 2014. 7, 82, 105

[10] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *SIGGRAPH*, 2000. 8

[11] Samuel Boivin and Andre Gagalowicz. Inverse rendering from a single image. *Conference on Colour in Graphics, Imaging, and Vision*, 2002(1), 2002. 7, 31, 32

[12] Nicolas Bonneel, Kalyan Sunkavalli, James Tompkin, Deqing Sun, Sylvain Paris, and Hanspeter Pfister. Interactive intrinsic video editing. In *SIGGRAPH Asia*, 2014. 7

[13] Bas Boom, Sergio Orts-Escolano, Xi Ning, Steven McDonagh, Peter Sandilands, and Robert B Fisher. Point light source estimation based on scenes recorded by a rgb-d camera. In *BMVC*, 2013. 53

[14] G. Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin Institute*, 310(1), 1980.

[15] Ricardo Cabral and Yasutaka Furukawa. Piecewise Planar and Compact Floorplan Reconstruction from Images. In *CVPR*, 2014. 18, 19, 21

[16] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], 2015. 95

[17] Qifeng Chen and Vladlen Koltun. A simple model for intrinsic image decomposition with depth cues. In *ICCV*, 2013.

[18] Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu, and Daniel Cohen-Or. 3-sweep: Extracting editable objects from a single photo. *ACM Trans. Graph.*, 32(6), 2013. 8, 114

[19] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *CVPR*, 2014. 95

[20] Michael F. Cohen, John Wallace, and Pat Hanrahan. *Radiosity and realistic image synthesis*. Academic Press Professional, 1993. 13, 34

[21] Alex Colburn, Aseem Agarwala, Aaron Hertzmann, Brian Curless, and Michael F. Cohen. Image-based remodeling. *TVCG*, 19(1), 2013. 8, 18

[22] R. Alex Colburn. *Image-Based Remodeling: A Framework for Creating, Visualizing, and Editing Image-Based Models*. PhD thesis, University of Washington, 2014. 31

[23] Robert L Cook and Kenneth E Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1982. 59

[24] Oliver Cossairt, Shree Nayar, and Ravi Ramamoorthi. Light field transfer: global illumination between real and synthetic objects. *ACM Trans. Graph.*, 27(3), 2008. 30

[25] Cyril Crassin, Fabrice Neyret, Miguel Sainz, Simon Green, and Elmar Eisemann. Interactive indirect illumination using voxel cone tracing. In *Computer Graphics Forum*, volume 30, 2011. 86

[26] Angela Dai, Matthias Nießner, Michael Zollöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans. Graph.*, 2017. 10, 112

[27] Paul Debevec. Rendering synthetic objects into real scenes. In *SIGGRAPH*, 1998. 7, 30, 52, 53, 83, 84, 85

[28] Paul Debevec, Andreas Wenger, Chris Tchou, Andrew Gardner, Jamie Waese, and Tim Hawkins. A lighting reproduction approach to live-action compositing. In *SIGGRAPH*, 2002. 7

[29] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH*, 1997. 14

[30] Bin Ding, Chengjiang Long, Ling Zhang, and Chunxia Xiao. Argan: Attentive recurrent generative adversarial network for shadow detection and removal. In *CVPR*, 2019. 83

[31] Farshad Einabadi and Oliver Grau. Discrete light source estimation from light probes for photorealistic rendering. In *BMVC*, 2015. 54

[32] Graham D Finlayson, Steven D Hordley, Cheng Lu, and Mark S Drew. On the removal of shadows from images. *TPAMI*, 28(1), 2005. 83

[33] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 1981. 99

[34] D. A. Forsyth. Variable-source shading analysis. *IJCV*, 91(3), 2011. 31

[35] Jan-Michael Frahm, Kevin Koeser, Daniel Grest, and Reinhard Koch. Markerless augmented reality with light source estimation for direct illumination. In *CVMP*, 2005. 54

[36] Tobias Alexander Franke. Delta light propagation volumes for mixed reality. In *ISMAR*, 2013. 86

[37] Tobias Alexander Franke. Delta voxel cone tracing. In *ISMAR*, 2014. 86

[38] Y. Furukawa, B. Curless, S.M. Seitz, and R. Szeliski. Manhattan-world stereo. In *CVPR*, 2009. 19, 22

[39] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *CVPR*, 2016.

[40] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, 2017.

[41] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth prediction. In *ICCV*, 2019.

[42] Dan B Goldman. Vignette and exposure calibration and compensation. *TPAMI*, 32(12), 2010. 10, 11, 12

[43] M.D. Grossberg and S.K. Nayar. What is the space of camera response functions? In *CVPR*, 2003. 12

[44] Michael D. Grossberg and Shree K. Nayar. What Can Be Known about the Radiometric Response from Images? In *ECCV*, 2002. 10

[45] Roger Grosse, Micah K Johnson, Edward H Adelson, and William T Freeman. Ground truth dataset and baseline evaluations for intrinsic image algorithms. In *ICCV*, 2009. 7, 82, 105

[46] M. Grundmann, C. McClanahan, and I. Essa. Post-processing approach for radiometric self-calibration of video. In *ICCP*, 2013. 10, 14

[47] Ruiqi Guo, Qieyun Dai, and Derek Hoiem. Paired regions for shadow detection and removal. *TPAMI*, 35(12), 2012.

[48] Kenji Hara, Ko Nishino, and Katsushi Ikeuchi. Light source position and reflectance estimation from a single view without the distant illumination assumption. *TPAMI*, 27(4), 2005. 53

[49] Kenji Hara, Ko Nishino, and Katsushi Ikeuchi. Multiple light sources and reflectance property estimation based on a mixture of spherical distributions. In *ICCV*, 2005.

[50] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[51] Jan Herling and Wolfgang Broll. Pixmix: A real-time approach to high-quality diminished reality. In *ISMAR*, 2012. 8

[52] Berthold K.P. Horn. Shape from shading: a method for obtaining the shape of a smooth opaque object from one view. 1970.

[53] Berthold K.P. Horn. Obtaining shape from shading information. *The Psychology of Computer Vision*, pages 115–155, 1975. 52

[54] Berthold K.P. Horn. *Robot vision*. MIT Press, 1986. 7

[55] Xiaowei Hu, Chi-Wing Fu, Lei Zhu, Jing Qin, and Pheng-Ann Heng. Direction-aware spatial context features for shadow detection and removal. *TPAMI*, 42(11), 2019. 83, 88

[56] Xiang Huang, Marc Walton, Greg Bearman, and Oliver Cossairt. Near light correction for image relighting and 3d shape recovery. In *Digital Heritage*, volume 1. IEEE, 2015. 53

[57] S. Ikehata, H. Yang, and Y. Furukawa. Structured indoor modelling. In *ICCV*, 2015. 18

[58] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 7, 99, 101

[59] Katrien Jacobs and Celine Loscos. Classification of Illumination Methods for Mixed Reality. *Computer Graphics Forum*, 25(1), 2006.

[60] Salma Jiddi, Philippe Robert, and Eric Marchand. Reflectance and illumination estimation for realistic augmentations of real scenes. In *ISMAR*, 2016. 54

[61] James T. Kajiya. The rendering equation. In *SIGGRAPH*, 1986. 32

[62] K Karsch, K. Sunkavalli, S. Hadap, N. Carr, H. Jin, R. Fonte, M. Sittig, and D. Forsyth. Automatic scene inference for 3d object compositing. In *SIGGRAPH*, 2014. 7, 83

[63] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. In *SIGGRAPH Asia*, 2011. 7, 31, 50, 86

[64] Kevin Karsch, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, Hailin Jin, Rafael Fonte, Michael Sittig, and David Forsyth. Automatic Scene Inference for 3D Object Compositing. *ACM Trans. Graph.*, 33(3), 2014. 31, 86

[65] John K. Kawai, James S. Painter, and Michael F. Cohen. Radioptimization. In *SIGGRAPH*, 1993. 31

[66] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3), 2013. 11

[67] Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. 3d object manipulation in a single photograph using stock 3d models. In *SIGGRAPH*, 2014. 8, 83, 86, 114

[68] Seon Joo Kim and Marc Pollefeys. Robust radiometric calibration and vignetting correction. *TPAMI*, 30(4), 2008. 10, 14

[69] Taeone Kim and Ki-Sang Hong. A practical approach for estimating illumination distribution from shadows using a single image. *International Journal of Imaging Systems and Technology*, 15(2), 2005. 54

[70] Matthew Klingensmith, Ivan Dryanovski, Siddhartha Srinivasa, and Jizhong Xiao. Chisel: Real Time Large Scale 3D Reconstruction Onboard a Mobile Device. In *Robotics Science and Systems*, 2015.

[71] Dimitrios G Kottas, Joel A Hesch, Sean L Bowman, and Stergios I Roumeliotis. On the consistency of vision-aided inertial navigation. In *Experimental Robotics*, pages 303–317. Springer, 2013. 10

[72] Pascal Lagger and Pascal Fua. Using specularities to recover multiple light sources in the presence of texture. In *ICPR*, 2006. 54

[73] Michael S Langer and Steven W Zucker. What is a light source? In *CVPR*, 1997.

[74] Fabian Langguth, Kalyan Sunkavalli, Sunil Hadap, and Michael Goesele. Shading-aware multi-view stereo. In *ECCV*, 2016. 52

[75] Hieu Le and Dimitris Samaras. Shadow removal via shadow image decomposition. In *ICCV*, 2019. 83

[76] Chloe LeGendre, Wan-Chun Ma, Graham Fyffe, John Flynn, Laurent Charbonnel, Jay Busch, and Paul Debevec. Deeplight: Learning illumination for unconstrained mobile mixed reality. In *CVPR*, 2019.

[77] Hendrik Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatial appearance and geometric detail. *ACM Trans. Graph.*, 22(2), 2003. 7, 54

[78] Pan Li, Bin Wang, Feng Sun, Xiaohu Guo, Caiming Zhang, and Wenping Wang. Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Trans. Graph.*, 35(1), 2015.

[79] Shuda Li, Ankur Handa, Yang Zhang, and Andrew Calway. HDRFusion: HDR SLAM using a low-cost auto-exposure RGB-D sensor. arXiv:1604.00895, 2016. 11, 15

[80] Yuanzhen Li, Stephen Lin, Hanqing Lu, and Heung-Yeung Shum. Multiple-cue illumination estimation in textured scenes. In *ICCV*, 2003. 54

[81] Zhengi Li and Noah Snavely. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. In *ECCV*, 2018. 103, 105

[82] Zhouchen Lin, Tien-Tsin Wong, and Heung-Yeung Shum. Relighting with the reflected irradiance field: Representation, sampling and reconstruction. *IJCV*, 49(2-3), 2002. 7

[83] Stephen Lombardi and Ko Nishino. Radiometric Scene Decomposition: Scene Reflectance, Illumination, and Geometry from RGB-D Images. arXiv:1604.01354, 2016. 31

[84] Stephen Lombardi and Ko Nishino. Reflectance and Illumination Recovery in the Wild. *TPAMI*, 38(1), 2016. 31

[85] Jorge Lopez-Moreno, Elena Garces, Sunil Hadap, Erik Reinhard, and Diego Gutierrez. Multiple light source estimation in a single image. In *Computer Graphics Forum*, volume 32, 2013. 53

[86] C. Loscos, G. Drettakis, and L. Robert. Interactive virtual relighting of real scenes. *TVCG*, 6(4), 2000.

[87] Manolis I. A. Lourakis and Antonis A. Argyros. SBA. *ACM Transactions on Mathematical Software*, 36(1), 2009. 13

[88] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan B Goldman, Cem Keskin, Steve Seitz, Shahram Izadi, and Sean Fanello. Lookingood: Enhancing performance capture with real-time neural re-rendering. In *SIGGRAPH Asia*, 2018. 83

[89] Maxime Meilland, Christian Barat, and Andrew Comport. 3D High Dynamic Range dense visual SLAM and its application to real-time object re-lighting. In *ISMAR*, 2013. 11, 15

[90] Abhimitra Meka, Michael Zollhöfer, Christian Richardt, and Christian Theobalt. Live intrinsic video. In *SIGGRAPH*, 2016. 7

[91] Bruno Mercier, Daniel Meneveaux, and Alain Fournier. A framework for automatically recovering object shape, reflectance and light sources from calibrated images. *IJCV*, 73(1), 2007. 31

[92] Moustafa Meshry, Dan B. Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *CVPR*, 2019. 83

[93] T. Mitsunaga and S.K. Nayar. Radiometric self calibration. In *CVPR*, 1999.

[94] Anastasios I. Mourikis and Stergios I. Roumeliotis. A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation. In *ICRA*, 2007. 10

[95] Natalia Neverova, Damien Muselet, and Alain Trémeau. Lighting estimation in indoor environments from low-quality images. In *ECCV*, 2012.

[96] Richard A. Newcombe, Andrew J. Davison, Shahram Izadi, Pushmeet Kohli, Otmar Hilliges, Jamie Shotton, David Molyneaux, Steve Hodges, David Kim, and Andrew Fitzgibbon. KinectFusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 10

[97] Matthias Niessner, Michael Zollhöfer, Shahram Izadi, and Marc Stamminger. Real-time 3D Reconstruction at Scale Using Voxel Hashing. *ACM Trans. Graph.*, 32(6), 2013. 10

[98] Jeffry S Nimeroff, Eero Simoncelli, and Julie Dorsey. Efficient re-rendering of naturally illuminated environments. In *Photorealistic Rendering Techniques*, pages 373–388. Springer, 1995. 7

[99] Kyle Olszewski, Sergey Tulyakov, Oliver Woodford, Hao Li, and Linjie Luo. Transformable bottleneck networks. In *ICCV*, 2019. 113

[100] Alexandros Panagopoulos, Chaohui Wang, Dimitris Samaras, and Nikos Paragios. Illumination estimation and cast shadow detection through a higher-order graphical model. In *CVPR*, 2011. 54

[101] Thoma Papadhimitri and Paolo Favaro. Uncalibrated near-light photometric stereo. In *BMVC*, 2014. 53

[102] Jaesik Park, Sudipta N Sinha, Yasuyuki Matsushita, Yu-Wing Tai, and In So Kweon. Calibrating a non-isotropic near point light source using a plane. In *CVPR*, 2014. 54

[103] Gustavo Patow and Xavier Pueyo. A Survey of Inverse Rendering Problems. *Computer Graphics Forum*, 22(4), 2003. 30

[104] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering: From Theory to Implementation*. Morgan Kaufmann Publishers Inc., San Francisco, USA, 3rd edition, 2016. 45, 93

[105] Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei A. Efros, and George Drettakis. Multi-view relighting using a geometry-aware network. In *SIGGRAPH*, 2019. 83

[106] Mark W. Powell, Sudeep Sarkar, and Dmitry Goldgof. A simple strategy for calibrating the geometry of light sources. *TPAMI*, 23(9), 2001. 54

[107] Liangqiong Qu, Jiandong Tian, Shengfeng He, Yandong Tang, and Rynson WH Lau. Deshadownet: A multi-context embedding deep network for shadow removal. In *CVPR*, 2017. 83, 103

[108] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In *SIGGRAPH*, 2001. 30, 50

[109] Ganesh Ramanarayanan, James Ferwerda, Bruce Walter, and Kavita Bala. Visual equivalence. *ACM Trans. Graph.*, 26(3), 2007. 50

[110] Peiran Ren, Yue Dong, Stephen Lin, Xin Tong, and Baining Guo. Image based relighting using neural networks. In *SIGGRAPH*, 2015. 7

[111] Thomas Richter-Trummer, Denis Kalkofen, Jinwoo Park, and Dieter Schmalstieg. Instant mixed reality lighting from casual scanning. In *ISMAR*, 2016. 7

[112] Kai Rohmer, Johannes Jendersie, and Thorsten Grosch. Natural environment illumination: Coherent interactive augmented reality for mobile and non-mobile devices. *TVCG*, 23(11), 2017. 7

[113] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, 2015. 87

[114] Aditya Sankar and Steven Seitz. Capturing indoor scenes with smartphones. In *UIST*, 2012. 18

[115] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Illumination from shadows. *TPAMI*, 25(3), 2003. 54

[116] Yoichi Sato, Mark D. Wheeler, and Katsushi Ikeuchi. Object shape and reflectance modeling from observation. In *SIGGRAPH*, 1997.

[117] Dirk Schnieders, Kwan-Yee K Wong, and Zhenwen Dai. Polygonal light source estimation. In *ACCV*, 2009. 54

[118] Rakshith R Shetty, Mario Fritz, and Bernt Schiele. Adversarial scene editing: Automatic object removal from weak supervision. In *NeurIPS*, 2018. 8

[119] Peter-Pike Sloan, Jan Kautz, and John Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH*, 2002. 86

[120] Shuran Song and Thomas Funkhouser. Neural illumination: Lighting prediction for indoor environments. In *CVPR*, 2019.

[121] Pratul P. Srinivasan, Ben Mildenhall, Matthew Tancik, Jonathan T. Barron, Richard Tucker, and Noah Snavely. Lighthouse: Predicting lighting volumes for spatially-coherent illumination. In *CVPR*, 2020.

[122] Jürgen Stauder. Point light source estimation from two images and its limits. *IJCV*, 36(3), 2000. 30

[123] Subcommittee on Photometry of the IESNA Computer Committee. Iesna standard file format for the electronic transfer of photometric data and related information. Technical Report LM-63-02, 2002. 39

[124] Richard Szeliski. Computer Vision: Algorithms and Applications. 2010.

[125] T. Takai, K. Niinuma, A. Maki, and T. Matsuyama. Difference sphere: an approach to near light source estimation. In *CVPR*, June 2004. 30

[126] Takeshi Takai, Atsuto Maki, and Takashi Matsuyama. Self shadows and cast shadows in estimating illumination distribution. In *CVMP*, 2007. 54

[127] Takeshi Takai, Atsuto Maki, Koichiro Niinuma, and Takashi Matsuyama. Difference sphere: an approach to near light source estimation. *Computer Vision and Image Understanding*, 113(9), 2009. 53

[128] Marshall F Tappen, William T Freeman, and Edward H Adelson. Recovering intrinsic images from a single image. *TPAMI*, 2005. 83

[129] Jonas Unger, Joel Kronander, Per Larsson, Stefan Gustavson, Joakim Löw, and Anders Ynnerman. Spatially varying image based lighting using HDR-video. *Computers & Graphics*, 37(7), 2013. 30

[130] Kimmo Valkealahti and Erkki Oja. Reduced multidimensional co-occurrence histograms in texture classification. *TPAMI*, 20(1), 1998. 95

[131] Marco Visentini-Scarzanella and Hiroshi Kawasaki. Simultaneous camera, light position and radiant intensity distribution calibration. In *Pacific-Rim Symposium on Image and Video Technology*, 2015. 54

[132] Jifeng Wang, Xiang Li, and Jian Yang. Stacked conditional generative adversarial networks for jointly learning shadow detection and shadow removal. In *CVPR*, 2018. 83, 103

[133] Tianyu Wang, Xiaowei Hu, Qiong Wang, Pheng-Ann Heng, and Chi-Wing Fu. Instance shadow detection. In *CVPR*, 2020. 82

[134] Yang Wang and Dimitris Samaras. Estimation of multiple illuminants from a single image of arbitrary known geometry. In *ECCV*, 2002. 53

[135] Yang Wang and Dimitris Samaras. Estimation of multiple directional light sources for synthesis of augmented reality images. *Graphical Models*, 65(4), 2003. 53

[136] Martin Weber and Roberto Cipolla. A practical method for estimation of point light-sources. *BMVC*, 2001. 30

[137] Thomas Whelan, Michael Kaess, Maurice Fallon, Hordur Johannsson, John Leonard, and John McDonald. Kintinuous: Spatially Extended KinectFusion. Technical report, MIT CSAIL, 2012. 10

[138] Thomas Whelan, Stefan Leutenegger, Renato Salas Moreno, Ben Glocker, and Andrew Davison. ElasticFusion: Dense SLAM Without A Pose Graph. *Robotics: Science and Systems*, 11, 2015. 10

[139] Tien-Tsin Wong, Pheng-Ann Heng, Siu-Hang Or, and Wai-Yin Ng. Image-based rendering with controllable illumination. In *Rendering Techniques*. Springer, 1997. 7

[140] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3D photography. In *SIGGRAPH*, 2000. 30

[141] Daniel N Wood, Daniel I Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *SIGGRAPH*, 2000. 55

[142] Chenglei Wu, Michael Zollhöfer, Matthias Nießner, Marc Stamminger, Shahram Izadi, and Christian Theobalt. Real-time shading-based refinement for consumer depth cameras. *ACM Trans. Graph.*, 33(6), 2014. 30

[143] Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. Neural scene de-rendering. In *CVPR*, 2017. 85

[144] Tai-Pang Wu, Chi-Keung Tang, Michael S. Brown, and Heung-Yeung Shum. Natural shadow matting. *ACM Trans. Graph.*, 26(2), 2007.

[145] Hao Xia and Paul G Tucker. Distance solutions for medial axis transform. In *Proceedings of the 18th International Meshing Roundtable*. 2009.

[146] Jianxiong Xiao and Yasutaka Furukawa. Reconstructing the World's Museums. *IJCV*, 110(3), 2014. 18, 20

[147] S. Xu and A. M. Wallace. Recovering surface reflectance and multiple light locations and intensities from image data. *Pattern Recogn. Lett.*, 29(11), 2008. 31

[148] Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. Deep image-based relighting from optimal sparse samples. In *SIGGRAPH*, 2018. 7

[149] Yibing Yang and Alan Yuille. Sources from shading. In *CVPR*, 1991. 53

[150] Shunyu Yao, Ming Harry Hsu, Junyan Zhu, Jiajun Wu, Antonio Torralba, William T. Freeman, and Joshua B. Tenenbaum. 3D-Aware Scene Manipulation via Inverse Graphics. arXiv:1808.09351, 2018. 85

[151] Zili Yi, Qiang Tang, Shekoofeh Azizi, Daesik Jang, and Zhan Xu. Contextual residual aggregation for ultra high-resolution image inpainting. In *CVPR*, 2020. 82, 90, 99, 101

[152] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination. In *SIGGRAPH*, 1999. 31, 32

[153] Yu Zeng, Zhe Lin, Jimei Yang, Jianming Zhang, Eli Shechtman, and Huchuan Lu. High-resolution image inpainting with iterative confidence feedback and guided upsampling. In *ECCV*, 2020. 82

[154] Yu Zeng, Zhe Lin, Jimei Yang, Jianming Zhang, Eli Shechtman, and Huchuan Lu. High-resolution image inpainting with iterative confidence feedback and guided upsampling. In *ECCV*, 2020.

[155] Edward Zhang, Michael F Cohen, and Brian Curless. Emptying, refurnishing, and relighting indoor spaces. In *SIGGRAPH Asia*, 2016. 9, 18, 28

[156] Edward Zhang, Michael F Cohen, and Brian Curless. Discovering point lights with intensity distance fields. In *CVPR*, 2018. 52

[157] Edward Zhang, Ricardo Martin-Brualla, Janne Kontkanen, and Brian Curless. No shadow left behind: Removing objects and their shadows using approximate lighting and geometry. In *CVPR*, 2021. 82

[158] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[159] Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape-from-shading: a survey. *TPAMI*, 21(8), 1999.

[160] Xuaner Zhang, Ren Ng, and Qifeng Chen. Single image reflection separation with perceptual losses. In *CVPR*, 2018. 92

[161] Yufei Zhang and Y-H Yang. Multiple illuminant direction detection with application to image synthesis. *TPAMI*, 23(8), 2001. 53

[162] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *TPAMI*, 2017. 90

[163] Wei Zhou and Chandra Kambhamettu. Estimation of illuminant direction and intensity of multiple light sources. In *ECCV*, 2002.

[164] Wei Zhou and Chandra Kambhamettu. Estimation of the size and location of multiple area light sources. In *ICPR*, 2004. 54

[165] Wei Zhou and Chandra Kambhamettu. A unified framework for scene illuminant estimation. *Image and Vision Computing*, 26(3), 2008.

[166] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 7

[167] Michael Zollhöfer, Angela Dai, Matthias Innmann, Chenglei Wu, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Shading-based refinement on volumetric signed distance functions. *ACM Trans. Graph.*, 34(4), 2015. 30, 52

[168] Zhengxia Zou, Sen Lei, Tianyang Shi, Zhenwei Shi, and Jieping Ye. Deep adversarial decomposition: A unified framework for separating superimposed images. In *CVPR*, 2020. 103, 105