Computational Discovery of Hidden Cues in Photographs

by

Tristan Swedish

B.S., Northeastern University (2014)
S.M., Massachusetts Institute of Technology (2017)

Submitted to the Program for Media Arts and Sciences,
School of Architecture and Planning,
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2022

Author .....
Program for Media Arts and Sciences,
School of Architecture and Planning,
June 28, 2022

Certified by .....
Ramesh Raskar
Associate Professor
Thesis Supervisor

Accepted by .....
Tod Machover
Academic Head
Program in Media Arts and Sciences

Computational Discovery of Hidden Cues in Photographs

by

Tristan Swedish

Submitted to the Program for Media Arts and Sciences,
School of Architecture and Planning,
on June 28, 2022, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Images of everyday scenes often contain hidden information that can be extracted to localize objects outside the view of the camera and to see around corners. For example, we show that it is possible to look at shadows cast by an object on a table, such as a teapot, and reconstruct an image of the surrounding room. We describe how to identify and make use of these *hidden cues* such as shadows, reflections, and other subtle changes in an image caused by the interaction of light with objects in a scene that are not in the direct-line-of-sight. We use the term *computational discovery* to describe techniques that can be used to uncover these cues and reveal hidden information.

Despite incredible advances in computer vision in recent years, cameras are limited to a single viewpoint of a scene, requiring invasive multi-camera setups or active imaging modalities to solve many perception tasks today. Prior work has identified hidden cues that are present in photographs of certain environments, but these methods often require human insight to identify cues, and extensive calibration to make use of them. In order to address the limitations found in prior work, we propose an end-to-end machine learning framework to identify hidden cues. More generally, we show that object localization is approximately equal to localizing a point light source, and describe how this insight can be used to identify situations when object localization is possible. Furthermore, we show that physically-based "inverse rendering" can be used to estimate how light travels within a scene, turning objects, like coffee cups or picture frames, into "object cameras". Physical models are quite fragile to small errors in estimated scene parameters. As such, we suggest reconstruction methods that make use of the uncertainty in scene parameters to improve robustness.

The thesis suggests a number of other interesting ways hidden cues may be used in combination with imaging systems. This work could inspire future cameras that incorporate the environment itself as part of the imaging system, blurring the line between observer and subject.

Thesis Supervisor: Ramesh Raskar
Title: Associate Professor

Computational Discovery of Hidden Cues in Photographs

by

Tristan Swedish

This dissertation has been reviewed and approved by the following committee members

Thesis Advisor . . . . . .

Ramesh Raskar

Associate Professor

MIT Media Arts and Sciences

Thesis Reader . . . . . .

Ashok Veeraraghavan

Professor

Rice University

Thesis Reader . . . . . .

Roarke Horstmeyer

Assistant Professor

Duke University

Acknowledgments

This thesis is the product of many collaborations with incredible people. There are so many people to thank: Prof. Ramesh Raskar taught me the value of fearless curiosity, presentation, and work ethic, and whose discussions and comments guided me through my graduate journey. I also want to thank my committee members, Prof. Roarke Horstmeyer and Prof. Ashok Veeraraghavan, who have been supportive and flexible amidst the unique challenges posed by Covid-19 and asking just the right questions to help shape this thesis.

It's impossible to properly credit those at the Media Lab: admins Maggie Church, Maggie Cohen, Ashley Clark, Linda Peterson, and Sarra Shubart—thank you for your time and encouragement. During my PhD, I'm honored to have worked closely with Guy Satat, Connor Henley, Tomohiro Maeda, Lagnojita Sinha, Devesh Jain, and Subhash Sadhu, among many others during my time at the Media Lab (I adore all of my co-authors). I also want to thank our UROPs and visiting researchers, where Summer 2019 was a catalyzing time for this work: Amey Chaware, Matthew Baugh, and Ankit Ranjan. I'm very grateful for the creativity and early encouragement from Matt Tancik, who showed many early results as an MEng, and whose project inspired much of the early ideas that led to this thesis.

Praneeth Vepakomma and Abhishek Singh provided much feedback and essential support as group members, and I'd like to thank the entire Camera Culture Group extended family and those who I may not have worked with closely during my PhD, but provided consistent feedback, support, and leading by example during my time at the Media Lab—Achuta Kadambi, Ayush Bhandari, Otkrist Gupta, and Barmak Heshmat.

Completing a PhD Thesis is probably not easy for anyone, and I certainly was no exception. My friends and family listened to my stories of triumph and failure, provided much needed emotional support, and asked "when are you graduating again?" a perfect amount of times. Chuck, you opened my eyes to my potential, and elucidated the wonders of phosphoromancy. Peter Henry, you and the inspiring team at Skydio

Figure 0-1: A picture is worth 1000 words. From Left: my advisor, Prof. Ramesh Raskar, my committee members, Prof. Roarke Horstmeyer and Prof. Ashok Veeraraghavan, and a montage (random ordering) of a few of the people who made this thesis possible. <3

introduced me to computer vision for autonomy, and exposed me to a rigorous and technical world outside of academia. Ubicept—Sebastian Bauer and the whole team—have been extremely supportive and I'm looking forward to the future together. Hardwick and TGL: love you people. My family—Mom, Jim, Dad, Maria, Grandma Susannah and Grandpa Dale, Nana and Nono, Grandma Vicki and Grandpa Jerry, the Breadens, the Swedish, the Bolognesi, the Graves, Carlos and Charles. So many people in my life radiate a kindness, love, curiosity, and intellectualism that has illuminated my way and taught me that Truth is that which remains consistent in spite of everything else.

Caroline Moy—a real-life "Shadow Baker"—this thesis is dedicated to you, it would not have happened without you. Thank you!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Consider a small object sitting on a coffee table in your living room. The object is illuminated by light sources from all directions—this includes direct sources such as the sun or overhead lights, but also indirect sources, like the foliage outside that scatters sunlight through your window. The appearance of the object and the shadows cast on the surface that it rests upon results from the complex interaction between the incident illumination and the geometry and material properties of the object and the surface of the table. By finding certain cues within this scene, such as specular paths or shadow edges, we might be able to extract more information from a given photograph, such as moving objects outside the line of sight, or viewpoints from the perspective of objects in the scene.

The idea that photographs can contain subtle clues is familiar in the forensics of photos posted to social media or reconnaissance operations using satellite imagery. In these cases, human investigators might use their knowledge and experience to make guesses about what caused certain visible phenomena, or use computer vision to perform photogrammetry. For example, the 1969 moon landing photos contain phenomena that can be fully explained by actual astronauts on the moon, debunking conspiracy theorists[1].

In this thesis, we explore a class of image analysis that makes use of knowledge of light transport, either through a large dataset or using inverse rendering, to localize

[1] https://blogs.nvidia.com/blog/2019/07/19/real-time-ray-tracing-apollo-11-turing/

objects and reconstruct views of a scene outside the line-of-sight of a camera that captured a particular image. One potential impact of such techniques is the ability to examine historical footage or photographs and revealing new views encoded in this image data that has previously eluded discovery.

1.1 Scope

This thesis considers the problem of discovering hidden cues to resolve information outside the direct line of sight of a camera that captures visible light intensity. There are number of ways information outside the line of sight could be obtained. For example, active illumination time-of-flight systems have realized rapid progress in the last few years [74]. Other modalities may include optical coherence based approaches [62, 76], acoustic [68], thermal [75], terahertz [32], and radio frequency [129, 1].

The prevalence of intensity based images sensors futher motivates the limited scope of this thesis. Rather than introduce additional hardware complexity, it is an interesting question to examine how much information is already present in an easy to acquire photograph. Furthermore, techniques that make use of commonly available image sensors may be relevant for future application of forensic anthropology or other forms of data mining that may be possible given existing photographic archives. As such, this thesis restricts the kind of photographic input to that which may be easily obtained using photograph or video data from commodity image devices.

1.2 Goals

The primary goal of this thesis is to examine the idea of using hidden cues within an image of a scene to extract additional information beyond the line of sight of the camera.

A secondary goal of this thesis is to demonstrate the idea of "computational discovery" to find hidden cues automatically by solving an inverse problem.

Once a scene's forward light transport model is known, we can define an inverse problem where we estimate the incident illumination on the scene from a camera image. We hope to show practical methods for solving such inverse problems, but also examine how these solutions can be used for computational discovery. For both data-driven and traditional inverse methods, we can in some situations obtain an inverse operator that can be analyzed to reveal useful hidden cues.

This thesis has technical implications for augmented reality and robotic perception, but it also challenges expectations of privacy given photographic data. By understanding what kind of information can be inferred from photographs, we may inspire novel techniques to analyze historical imagery or strategies to minimize undesirable violation of privacy via video surveillance. We might also be inspired to design imaging systems that make use of particular cues or are well-adapted to a given environment.

1.3 Research Questions

What are the kinds of hidden cues can be found in photographs? What scene properties must be known ahead of time to take advantage of these cues?

While this thesis is in part about discovering hidden cues in photographs, the physics of light transport constrains the different kinds of hidden cues that may be present. We can describe a few broad classes of cues such as specular reflections, diffuse scattering intensity drop-offs, and shadow edges. In order to take advantage of these cues, some scene properties may need to be known ahead of time. Identifying what knowledge about a scene must be obtained is important for the practical use of these cues.

Can we solve vision tasks such as object localization outside the line-of-sight using a purely data-driven or end-to-end approach? How can this task be used to discover hidden cues? What are the limitations of such an approach?

Prior work has shown that some level of object localization is possible using shadow cues, but require prior knowledge about the cue itself. We will investigate if it is possible to use a learning based approach, such as training a deep neural network, to predict the location of a hidden object from a sufficient dataset. In order to solve this task, the network would need to uncover these cues purely from data, without any direct prior knowledge of physics available.

If such a network can be trained, then it would be useful to describe strategies for uncovering the cues it uses to solve the problem. However, a lack of physics knowledge may be disadvantageous, and should be understood.

What are opportunities for combining machine learning and physics? How can a machine learning based approach be augmented with physics-based constraints?

We understand the physics of light transport quite well, but this knowledge is underutilized in a purely data-driven approach. It would be useful to understand how to combined this knowledge with that uncovered by a model trained using data. For the object localization task in particular, we want to ensure that the resulting solution from a model trained using machine learning is consistent with physical laws.

Are there opportunities to develop imaging strategies that utilize objects in the scene as part of image formation? What properties of objects enable more accurate scene reconstruction?

In this thesis, we are particularly interested in the cue of cast shadows. Shadows are cast when an object absorbs or reflects a subset of light paths traveling through the scene. Since objects are typically localized to a particular region of space, they typically only block a subset of light paths traveling in the same direction over the entire scene. Effectively, this property can make certain objects behave as an 'inverse pinhole" that can be used to estimate the incident illumination on an object from each direction. If we recover the incident direction of all light paths on the sphere surrounding the object, we can obtain a "360 photo" of the scene from the perspective

of the object. We should understand the practical limitations such as the impact of captured image signal-to-noise ratio, object shape, and model mismatch on the scene reconstruction.

How might imaging systems be designed in the future to support computational discovery of hidden cues in the environment?

Lastly, once we understand what cues can exist, how to find them, and how to make use of them, it would be interesting to imagine how to better design cameras that make use of these cues. In particular, what future directions are available to imaging system designers based on the knowledge and strategies discussed in this thesis?

1.4 Background

1.4.1 What is a photograph?

This work is focused on hidden cues in photographs, but what do we mean by *photograph*? We refer to photographs as images captured by a conventional camera. While we will not dwell on the finer points of image capture, we describe the basic components of a conventional camera, and their impact on the work described in this thesis. Detailed coverage of photography and its relevance in the context of this thesis can be found in computer vision textbooks, such as [39, 113].

Light At a high level, a conventional camera measures the visible light radiance of a scene from a certain perspective. Radiance is the amount of light, or flux, coming from a particular location in a scene over a given set of angles. Light is composed of particles of light, called photons, that are subject to the laws of quantum mechanics. The details of these laws are not important for the purposes of photography, but impacts image formation in a few important ways. First, light typically travels along straight lines called rays. However, light paths can bend due to refraction, or rapidly change direction when photons scatter off surfaces. Second, photons have different amounts of energy, or wavelength, that we perceive as color in the visible spectrum.

Third, the speed of light is limited. Lastly, flux is an integer count of photons. As such, cameras seek to measure the direction, amount, and energy of photons passing through a region in space. A photograph is a representation of these measurements.

The visible scene A camera takes pictures of a *scene*. Natural scenes typically contain things like objects and light sources. Light sources emit photons that illuminate objects, and the surface of the objects scatter these photons in different directions. Later in this chapter we will describe this in more detail, but the important part is that illuminated objects scatter light, and this scattered light forms the radiance of the scene.

Pinhole Camera A simple model for a camera is called a *pinhole camera*, which allows light to pass only through a tiny hole before being projected onto a flat surface. Pinhole cameras separate light rays passing though a point in space such that each ray direction is mapped to a point along a 2D surface. Thus, the resulting image is a perspective projection of the radiance of the scene. Cameras have a field of view, which represents the set of angles of light they can measure that pass through the pinhole or aperture.

Lens and Aperture In practice, cameras must have a larger aperture to allow enough light to be collected by the sensor for a given period of time. Light can be focused by a lens such that the incoming angle of light can be mapped to 2D sensor position. Since a lens can only perform this mapping exactly for a single scene depth, it must be focused to produce sharp images for a given scene depth.

The aperture of the lens controls the *depth of field*, or the set of similar depths that produce sharp images. Small apertures produce sharp images over a large set of depths, but let in less light. Smaller apertures also reduce the resolution of the imaging system. Since light is best described by quantum particles, or *photons*, it is subject to the Hesienberg uncertainty principle. This means that for small apertures, we can more precisely limit the position of a photon, but the ability to preserve the direction of travel, or momentum, is degraded. We call our ability to measure the

angle of incoming light rays the *optical resolution*.

Sensor When a light ray reaches the back of a camera, it collides with the sensor. Conventional camera sensors are composed of an array of pixels that convert photons into electrons, which are stored in a capacitor. As such, the voltage across this capacitor measures the total count of photons that hit the sensor for a given period of time. The camera exposure time is the amount of time photons are added to the capacitor before the voltage is read by an analog to digital converter and then allowed to drain charge back to zero. Sensors often have an optical band-pass filter over each pixel called a Bayer pattern, such that each pixel only observes photons with a range of wavelengths, or color.

Image Signal Processing Finally, a photograph is obtained when the data is read from the sensor and a series of post-processing steps is applied to produce full-color images.

1.4.2 What is the hidden scene?

In this thesis, we are interested in recovering hidden information about a *hidden scene* from photographs of a visible scene. As such, we are interested in the geometry and appearance of the hidden scene. In a general sense, we would like to recover similar information about the hidden scene that we obtain from images of the visible scene. This might include then position and orientation of hidden objects, or an image of the hidden scene from the perspective of objects in the visible scene.

A general way to describe the hidden scene is function that returns the radiance for position, $u \in (x, y, z)$, and angle, $\omega \in (\theta, \phi)$ within the volume outside the visible scene. We call the tuple, (u, ω), a *ray* to emphasize that a single point in ρ describes the origin and direction of a light path.

$$\rho(u, \omega) \tag{1.1}$$

This function returns the spectral radiance and absorption at every point. The

domain of ρ is over 5 dimensions: 3 space and 2 angle. This representation of the scene is quite general, and encodes both the spatial location of objects, but also their appearance. In a way, this representation is a light field in 3D space, which has been referred to as the *radiance field* [78] in the literature. In the next section, we describe how this scene representation used across the visible and hidden scene can be used to define a forward model for light transport.

1.4.3 Forward light transport

The forward light transport of a scene describes how light travels from light sources through the scene before eventually reaching the camera. A physically accurate forward model is the starting place for understanding how hidden cues in the visible scene are created by changes in the hidden scene.

The Rendering Equation The rendering equation by [48] is a seminal contribution to physically accurate rendering. It describes how the radiance at each surface point in the scene is a function of the incident illumination. The incident illumination can be determined recursively by integrating all light that originates from some set of light sources, much of which might arrive at a scene point after many scattering events with other scene points. Due to the nature of this recursion, the domain of the integrand is practically infinite, so the solution to this equation is often approximated using monte carlo techniques known as *path tracing*.

Using the rendering equation, we can calculate the amount of light from a ray origin that reaches the observed image Y.

$$L_o(u, \omega_o) = \rho'(u, \omega_o) + \int_\Omega f(u, \omega_o, \omega) L_i(u, \omega) d\omega \tag{1.2}$$

Here, the equation above is intended to sketch the idea and obscures some detail in favor of clarity. The function $f(\cdot)$ maps an incident light ray, (u, ω), to a ray (u, ω_o). This function is called the bi-directional reflectance distribution function (BRDF) when describing scattering from surfaces, and we assume here that f contains the

necessary cosine terms. More generally, f is called the *phase function*, which describes how incident light is scattered or absorbed at position u in the scene. L_i describes the radiance of incident light rays at point u, which is determined by a similar integral, leading to the recursive nature of the equation.

The equation integrates over the surrounding sphere of angles Ω. The light must originate from somewhere, here we designate ρ' a function that describes source light rays over the full space. This function may be the hidden scene, if we consider the hidden scene a light source.

For conceptual convenience, we define a function $a(u, \omega, v, \phi) \in [0, 1]$ that maps the total transmission from source light ray (v, ϕ) to output light ray (u, ω). We note that this can be calculated by the the volumetric rendering equation, omitted for brevity here taking a similar form of the rendering equation above, by setting $\rho'(u, \omega) = \delta(v, \phi)$. A discrete version of this over a vector of (v, ϕ) values would effectively set $\rho'(v, \phi) = 1$ and 0 everywhere else. In physical systems, a is conserved, so it must lie in the range $[0, 1]$.

Image formation model We can write the observed image as a function of the hidden scene volume, V. We assume the hidden scene is the only source of illumination and a properly accounts for occlusion and differential terms. In this way, we can integrate over the hidden scene, ρ, to generate an observed image Y.[2]

$$Y(c, \omega_o) = \int_V \int_\Phi a(c, \omega_o, v, \phi)\rho(v, \phi)d\phi dv \tag{1.3}$$

Here, c is the camera position and ω_o is the direction corresponding to each camera pixel.

We can also solve for the light field generated by ρ on the hemisphere surrounding the visible scene: $x(u, \omega) = \int_V \int_\Phi a(u, \omega, v, \phi)d\phi dv$. From the perspective of the visible scene, the generated image is identical.

[2]This equation is only intended to provide a sketch of the idea. In practice we are interested in the discrete version: we render an image for the columns of a matrix, A, each corresponding to a single ray tuple (v, ϕ). The double integral is used to emphasize we are integrating over a light-field.

$$Y(c, \omega_o) = \int_U \int_\Omega a(c, \omega_o, u, \omega) x(u, \omega) d\omega du \tag{1.4}$$

We write this equation to show the general form we use when discretizing the problem: $y = Ax$, where y is the observed image pixels, x is the unknown illumination, and A is the ray transport matrix. If we assume the hidden scene is far away, then we can remove the integration over u and instead only consider rays originating on the hemisphere, ω. In this way, x is an illumination source known as the *environment map*.

Source of hidden cues

Hidden cues may arise from a few features in the visible scene. If the visible scene is illuminated by x, then intuitively cues occur when changes to unique values in x correspond to unique changes in y. In other words, two incident rays (u_1, ω_1) and (u_2, ω_2) each create changes in y that are orthogonal.

Diffuse Reflections Lambertion surfaces, or diffuse surfaces, remove a significant amout of information about the direction of illumination. For this reason, they are not good sources of cues.

Specular Reflections Reflections from shiny surfaces like polished metal are specular. Using knowledge of the surface normal, reflections are great cues for recovering the hidden scene. This is commonly exploited in practice: chrome balls are often used to estimate the environment map in a real scene. The problem with specular reflections is that suitable objects must be available in the environment. The surfaces of the object must be specular, but also have a good variety of surface normals.

Shadows Interestingly, shadows cast by objects in the visible scene create fantastic cues. Shadows create diverse changes in the observed images with respect to incident light direction and only require a scene with sufficient self-occlusion. Shadows are

incredibly common in natural scenes. As such, shadows a are good cue, and for this reason are extensively studied in this thesis.

1.4.4 The inverse light transport problem

In this work, we would like to discover and make use of hidden cues that allow us to solve the inverse light transport problem. We would like to recover the hidden scene radiance, x, given observations y. This admits the familiar linear system.

$$y = Ax \tag{1.5}$$

Where A is the light transport matrix. We may only have partial information about A, and the nullspace, or kernel, of A may make recovery of x ill-posed without regularization. These two problems form the fundamental motivation for this thesis. In practical terms: we must account for the *calibration* of A and utilize any prior knowledge about x. These high-level relationship between these concepts are shown in 1-1.

Model Priors We might not fully know the entries of A when we observe a scene. Perhaps the exact texture of geometric details are unknown. However, many spaces follow predictable shapes, potentially making it possible to "fill in" A with the most likely values. For example, if we observe a room with a table and chairs, we do not need to know the full geometry to provide a good guess for the geometry and shape of the table and chair legs.

Data Consistency When capturing an image y, we want to ensure that any guess we provide for A and x are consistent with $Ax = y$. This may actually be fairly challenging as measurement noise and model mismatch can make this exact equality impossible to achieve. As such, we want to at least ensure data consistency in a least squares sense.

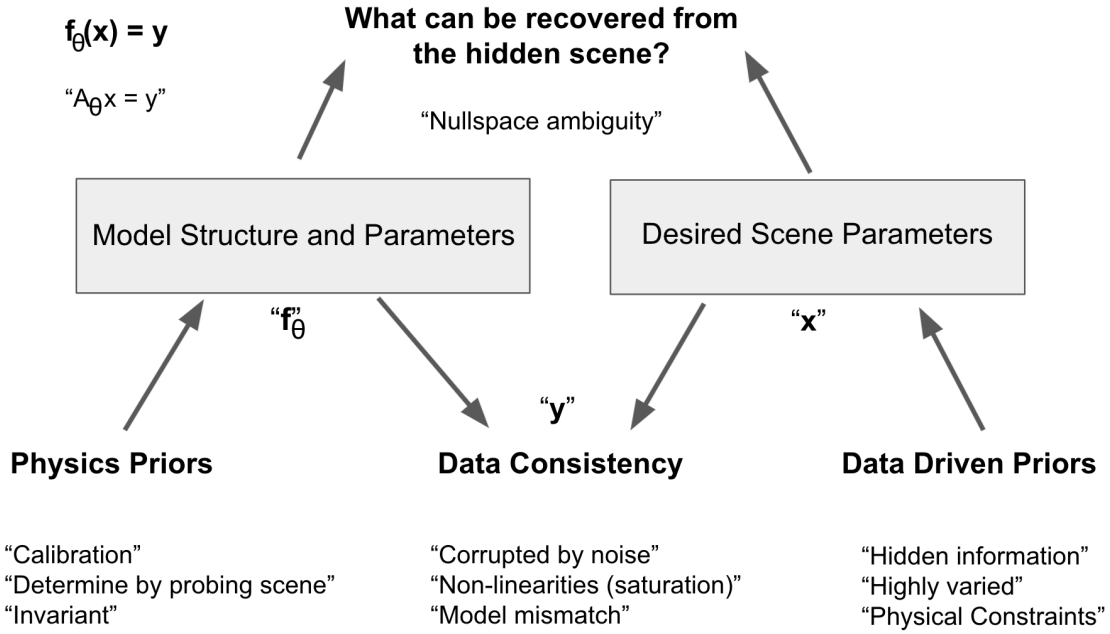


Figure 1-1: When solving an inverse problem, it is useful to analyze the components of the forward model. There are numerous considerations when defining a forward model that maps the hidden scene to an image that we can observe. Often, we must calibrate a physics-based model to match the current environment which may require some probing of the visible scene. When solving the inverse problem we want to ensure our observations match the output of our forward model, but model-mismatch and noise may complicate this process. Furthermore, the forward model may have some "nullspace" making some variation in the hidden scene undetectable when we only have access to observations. This makes the inverse problem ill-posed, and data-driven priors can be useful for selecting from a potentially infinite set of possible solutions.

Hidden Scene Priors While we may not know much about the hidden scene, there may be common scene priors given the context of the visible scene. For example, we know that the hidden scene radiance should at least be non-negative. Data-driven methods can provide reasonable guesses when observing the visible scene context. A car is more likely than an elephant when reconstructing an object near a city intersection.

Recoverability The recoverability of the hidden scene from measurements can be formally understood by analyzing the nullspace of the forward model operator A. There may be changes to x, for example $x' = x + a$ such that $Ax' = Ax$. In this way, a is not recoverable. There may also be particular directions that are not in the nullspace of A exactly, but are still pointing in the direction of an eigenvector with a small eigenvalue. This problem is made more complex when there is uncertainty about the forward model, so the entries of A may not be known exactly.

1.5 Overview of contributions

This thesis explores the idea of using hidden cues in photographs to extract information from a hidden scene. Chapters 3-6 contain the majority of the ideas contributed to this thesis and are summarized below.

1.5.1 Chapter 3

Chapter 3 explores the use of data-driven methods to localize hidden objects. We show how localizing hidden objects is related to the problem of localizing point sources outside the line of sight of the camera. We also examine localization when no calibration of the visible scene is available. Instead, we train a neural network to infer the visible scene parameters to solve the localization task end-to-end. We also propose a number of ideas for future work.

	Input	Output	Algorithm	Results	Applications
Chapter 3 **Data Driven Object Localization: outside line of sight of camera**	Camera Image from scene type (e.g. “wall corner”)	2D occupancy probability grid	CNN Classifier trained using Domain Randomization Dataset	Real-time localization of a moving object around a corner-like scene without scene calibration	ADAS vehicle safety (detect moving pedestrian around the corner), NLOS museum problem
Chapter 4 **Localizing using Shadow Edges**	3D Shadow and Corner Edges or Camera Image and Known Scene	3D Location of (multiple) point radiance sources	Plane Intersection or Gradient Descent with Differentiable Renderer	Determine when localization is possible using shadows	Point light / Scene / Camera Calibration, Hybrid data-driven physics-based object localization
Chapter 5 **Turning Objects Into Cameras**	Camera Image and partially known scene parameters (geometry and reflectance)	Illumination Environment Map “360 photo from object’s perspective”	Regularized Linear Inverse with Non-negativity constraints	Reconstruct view from perspective of object in a scene	Museum Problem, AR: Object Insertion, Historic Image Forensics
Chapter 6 **Discovering Hidden Cues**	“Inverse Operator” - Trained Neural Network - Regularized “Inverse” Matrix	Relevant regions of image / General features useful for solving inverse task	Gradient Saliency, Cook’s Distance, Covariance	Reveal hidden cues found by solving the inverse problem	Active Imaging, Designing task oriented cameras

Figure 1-2: An overview of the different algorithms and techniques proposed in this thesis by chapter number. There are different inputs and outputs each technique is designed to accommodate. We also list the demonstrated results and possible applications.

1.5.2 Chapter 4

In Chapter 4, we provide more analysis of shadows as a cue for localizing hidden objects. We also build on this analysis to propose "rules of thumb" for visible scenes that contain sufficient information for localization or reconstruction. In particular, we find that like the perspective ambiguity for monocular pinhole cameras, cast shadows from a single edge are insufficient to localize a point source in 3D. We also provide some ideas for analyzing shadow edges in a scene to efficiently discretize the hidden scene volume.

1.5.3 Chapter 5

Chapter 5 utilizes more knowledge about the forward model to reconstruct an image of the hidden scene from the perspective of an object in the visible scene. We use a rendering engine to generate the ray transport matrix, which we can then invert using standard methods. We also describe how to handle unknown albedo, and perform numerous experiments to understand the limits of reconstruction. We propose a

number of extensions for future work, such as handling uncertainty in the forward model and making use of differentiable rendering.

1.5.4 Chapter 6

In Chapter 6, we perform further analysis to develop ways to automatically uncover hidden cues from images. We also provide additional analysis to the image reconstruction problem. Given these insights, we propose a framework for automatically designing imaging systems that may make use of hidden cues.

Chapter 2

Related Work

2.1 Capturing and exploiting light transport in computational imaging

Computational Imaging, and the closely related field of Computational Photography, was partly conceived from an increasing understanding of light transport in realistic scenes in the field of Computer Graphics[48]. Some researchers realized that more information about a real scene could be captured by specialized cameras that were created to estimate properties of a scene relevant to light transport[124, 64, 30, 83]. The desire for capturing this information motivated the evolution of camera systems that estimated more aspects of the underlying light transport in the scene[119, 94, 63, 103, 86].

Today, computational imaging is an integral part of the most widely distributed consumer cameras in smartphones, robotic perception for autonomous vehicles, and improvements to health monitoring and medical imaging. Increasingly, ideas from computational imaging have been used to push the boundaries of fundamental science as computational methods must be employed to measure parts of the world that are smaller, faster, or further away than ever before[120, 70, 5].

2.1.1 Inverse problems in computational imaging

In imaging, we are often interested in the *Inverse* problem of light transport: what inputs to some physical model generated the observed image? We often must choose a forward model to describe our measurement process, and then use various techniques to invert the forward model to explain these measurements. There are two primary approaches used to solve inverse problems today.

The first approach is model-based and seeks to cast the problem as a classic optimization problem, where a scalar objective function is minimized using an iterative update scheme, such as gradient descent or Newton's method. Objective functions with regularization, such as total variation [21], can be solved using iterative schemes such as [18].

The second approach builds on the rapid progress of machine learning to use a dataset of input-output pairs to learn an approximate inverse operator. Recently, the use of deep neural networks has been applied to inverse problems in computational imaging [49, 51, 96, 130, 112, 52]. Traditional iterative methods have also started to incorporate more deep neural networks in their design, where a fixed number of iterations can be represented as a multi-layer neural network, known as "unrolled optimization"[77, 31, 38].

In addition to deep neural networks, Computational Imaging has a long history of "data-driven" ideas to serve as priors for inverse problems, ranging from learned basis functions using singular value decomposition to the addition of ideas in sparsity via sparse dictionaries[2, 79].

2.1.2 Inverse Rendering

Recently, several data-driven approaches have been proposed to learn a mapping between reference photographs and scene parameters [89] [8, 25] directly. These learned mappings typically incorporate a differentiable rendering module to serve as conditioning during training [8, 25, 50, 115].

Volumetric scene representations have been proposed for spatially varying lighting

estimation [109]. Similarly, implicit neural representations have shown impressive results for inverse rendering tasks [78**?**], but estimates the unmixed product of material reflectance and lighting.

Recently, practical implementations of differentiable path tracing for the rendering of mesh-parameterized geometries have been developed [65], as well as a reparameterization that makes use of modern autodifferentiation techniques [85, 71].

Demonstrations of de-rendering have typically been used for re-lighting or 3D reconstruction tasks and are less focused on "image-like" illumination estimates.

2.1.3 Shadow Volumes and Edge-shadow Boundaries

First invented by Crow [29] and later expanded [13], shadow volumes are a general algorithm for computing shadows in computer graphics. Shadow volumes have been of more recent interest for rendering volumetric shadows, and specialized coordinate systems have been proposed for this task [123]. Typically, research interest in shadow volumes has focused on efficient computation for real-time graphics [23]. Instead, we are interested in the inverse problem that shadow volumes were proposed to solve and our parameterization of occluded light paths is inspired by this early work.

2.1.4 Differentiable Rendering

Automatic differentiation of computer programs has made it possible in some cases to invert complex physical simulations using gradient descent [92]. In computer graphics, practical implementations of differentiable path tracing for the rendering of mesh-parameterized geometries have been developed [65], as well as a reparameterization that makes use of modern autodifferentiation techniques [85, 71]. Even though these physics-based forward models are differentiable, gradient descent is still susceptible to finding solutions that are at a local, rather than global, minimum. Hybrid methods combining neural networks and differentiable rendering have been proposed, and shows the promise of combining the advantages of data-driven inverse rendering with a model-based approach [25].

2.1.5 Learning important features and hidden cues for inverse problems

One of the key concepts in this thesis is the ability to exploit hidden cues within images. While both data-driven and traditional approaches for solving inverse problems may make use of these hidden cues, additional analysis may be necessary to characterize these cues for human understanding. In machine learning and computer vision, this is often under the umbrella of "explainable AI" [126, 67], with popular methods such as [102] that highlight regions of an image that are important for solving an inference task. Within inverse problems, this is closely related to understanding uncertainty in estimates due to a model's sensitivity to inputs. A statistical tool known as Cook's distance [28] can be used in analyzing inverse problems to understand what input pixels are most influential for a given estimate.

2.2 Non-line-of-sight (NLoS) Imaging

Non-line-of-sight (NLoS) imaging for the problem of "seeing around the corner" has received significant attention in recent years. These methods often utilize time-of-flight measurements with the primary goal to recover hidden geometry [120, 87, 70, 69, 3]. Rendering and optical transport models have also been developed specifically for time-of-flight NLoS reconstruction [43].

Bouman et al. [17, 101] demonstrated 1D tracking of objects around a corner with angular filters that are used to track the angular position of the hidden object, which has also been used for a robotics navigation task [82]. Other approaches have utilized active sources to illuminate hidden objects [53, 114, 26] and utilized reflected light for localization and reconstruction. Visual deprojection [10] proposed to learn to estimate 2D scenes from 1D projections with examples of NLoS applications. Despite these impressive results, data driven techniques have been limited to specific scene setups and have not been shown to work on novel and diverse scenes.

Exploiting occlusions for NLoS has been an active area of research, for time-of-flight

based approaches [116], blind deconvolution in intensity based NLoS [81, 99, 125], and the recovery of light fields [11]. Critically, these works have shown reconstructions from calibrated planar surfaces.

Recently, clever use of deep image priors has been shown to be effective for the highly ill-posed problem of blindly recovering a ray transfer matrix for a NLoS task [4]. Our work differs from previous work, by exploiting edges occlusions of any orientation relative to floor surface shadows. Furthermore, we demonstrate how our method can be used without extensive calibration and for more practical camera placements. We note that our formulation for a single edge is similar to [17], but our construction supports any edge and surface orientation, is not restricted to vertical edges, and does not require a homography.

2.2.1 NLoS using L-Corner Geometries

NLOS imaging has been explored in a variety of computer vision applications such as image dehazing [14], imaging through fog [97], and underwater imaging [105]. One unique instance of NLOS imaging is the seeing around corner problem which was initially demonstrated by Velten et al. [120]. Since then, a wide range of advances that cover different aspects of the problem have been suggested.

In the hardware side, ultrafast streak cameras were initially used [120], followed by single photon avalanche diodes (SPAD) cameras [35], SPAD pixels [19], and AMCW time-of-flight cameras [40, 47]. With the advances in sensing hardware the community developed many improvements in the recovery algorithms. Initial works solved the problem with back propagation and dictionaries [120**?**]. More recently, O'Toole et al. [87] suggested a reparameterization of the problem and casted it as a deconvolution for improved reconstruction quality.

Several recent works introduced seeing around corners with traditional cameras. These can broadly be categorized as methods that rely on:

- **Active illumination:** A method by Klein et al. [53] used a laser pointer and a regular camera to track objects around a corner with an optimization technique

on top of a graphical renderer to track an object with six degrees of freedom. Tancik et al. [114] used a traditional camera with a spot light to track objects and reconstruct scenes around corners with a data-driven approach.

- **Passive illumination (rely on ambient light):** Bouman et al. [17] demonstrated 1D tracking of objects around a corner with angular filters that are used to track the angular position of the hidden object.

Our suggested approach is based on hardware similar to Bouman et al. (consumer camera without calibrated active illumination). However, we use a data-driven approach and demonstrate 2D tracking around corners (compared to 1D tracking by Bouman). Tancik et al. used a data-driven approach but their solution was limited to a specific known corner.

The use of data-driven techniques in computer vision has many advantages, most notably is their ability to avoid model mismatch. Data-driven techniques have recently been applied to many problems in the computational imaging such as imaging through scattering media [96], phase imaging [107], compressive imaging [57], and microscopy [41]. In the context of seeing around corners a data-driven approach based on active time of flight system (SPAD camera) [20] was able to localize and identify people around a corner. More recently Tancik et al. [114] used a data-driven approach with a traditional camera to localize, identify, and reconstruct occluded scenes around corners. Both [20, 114] were trained for a particular corner and didn't show robustness to unseen corners. Our approach is robust to variations in the scene and is demonstrated on multiple corner geometries and instances that were not explicitly in the training set.

2.2.2 Occlusion Assisted Imaging

Early occlusion-based non-line-of-sight (NLoS) approaches make use of specific scene features such as accidental cameras or pinspecks [27, 117]. More recently, some different capture methodologies have been proposed for time-of-flight based approaches [93], blind deconvolution in intensity-based NLoS [81, 99, 125, 116], and the recovery of

light fields [11]. Critically, these works have shown reconstructions from calibrated planar surfaces. The use of the deep image prior has been proposed to approximate the ray transfer matrix [4], but assumed planar hidden scenes and did not investigate realistic high dynamic range hidden scenes.

Bouman and Seidel [17, 101] each demonstrated how the occlusion of light at a flat edge enables the reconstruction of 1D projected views of a scene hidden around a corner. Other approaches have utilized active sources to illuminate hidden objects [53, 114, 26]. Visual deprojection [10] is a learning-based approach to estimate 2D scenes from 1D projections, and like other learning-based approaches, is limited to specific scene setups and has not been shown to work on novel and diverse scenes. Our method supports the use of objects with arbitrary shape, and does not require planar or one-dimensional masks or planar, calibrated relay surfaces.

There is significant similarity between incident illumination estimation and NLOS imaging. In essence, if we can estimate the incident illumination for each visible surface, then each visible surface serves as a "camera" that provides a diversity of views of the hidden scene. If the surface BRDF and normals are known for an image patch, we can estimate incident illumination by inverting a linear system. For patches with high-specularity and diverse surface normals, incident illumination can be accurately estimated. However, diffuse surfaces make this inversion poorly conditioned. There has been a growing body of work demonstrating the power of occluding surface edges to make this inversion better conditioned for NLOS problems, and these methods typically require estimation of the light transport matrix: mapping some set of illumination positions in the hidden scene to the observed image.

2.3 Incident Illumination Estimation

There has been a wide variety of research applied to the problem of incident illumination detection. In general, these methods require specific capture setups [30], or seek approximations useful for downstream tasks [58, 124]. Sato et al [98] were one of the first to define the problem of approximating illumination from shadows of objects on

diffuse surfaces. Later work has emphasized the use of sharp shadow boundaries from a few bright sources for mixed reality applications [45, 66, 128], rather than extended sources.

Recently, Jiddi et al [46] demonstrated illumination estimation using both specular paths and shadow information. Specular paths are a useful cue for estimating incident illumination [37, 88] potentially "in the wild," but suitable surfaces must be present in the scene.

Data driven techniques have been used to estimate incident illumination for a database of objects [121]. Sophisticated capture methodologies have been used to generate large datasets used to train neural networks that are capable of estimating incident illumination in natural scenes [36, 60] for augmented reality applications or learning illumination for portrait relighting [61]. Spatially varying illumination estimation has been shown using RGBD images [12] and regular images [36], but these models do not predict exact light source location. There has been recent interest in precise localization of point light sources from images using material shading models [127], however our work utilizes occlusions and shadows to accomplish this task.

2.4 Computational Discovery of Optical Designs

2.4.1 Natural Evolution of Animal Eyes

As the eye developed through natural evolution, [84] were among the first to demonstrate how incremental changes in organism morphology could lead to the evolution of the complex optical system found in animal eyes while only optimizing for visual acuity at each step. A simple iterative optimization scheme like gradient descent could explain how the eye developed—starting with a collection of photo-sensitive cells and ending with the familiar spherical shape containing a refractive element to form an image. Later, [33] and [100] describe the necessary evolutionary biology and the evidence supporting incremental develop of eye designs. More recently [118] has

shown how eye evolution can be simulated using iterative optimization schemes. These works demonstrate how eyes have evolved in nature, but also serve to support the idea that iterative optimization can be well suited for the design of optical systems. Less well studied is the corresponding development of the visual processing in the visual cortex of animals, but it is reasonable to assume these brain areas developed in a similarly incremental manner as described by [56].

2.4.2 Traditional Optical Design

Traditional optical design has largely consisted of human designers, such as [104], who select suitable lens configurations and the use of ray tracing to validate designs. In practice, lens designers may reference a library of optical designs or rely on their intuition and experience. Many optical designs optimize for objectives that take into account visual acuity across the visible spectrum, but also robustness to errors in manufacturing or perturbations such as vibration. As such, many optical designs are part art and part science, and automated design frameworks have not entirely replaced human designers.

With the advent of computer aided design tools, designers typically sketch a rough design and refine some selected parameters, such as lens radius of curvature or exact component position, using tools such as ZeMax.[1] More recently, as described by [42], more exotic optical designs have been made possible using free-form optics and design and simulation tools. In any case, the primary objective of these design tools is to optimize for low-level criteria such as a target point-spread function (PSF) over a set of wavelengths and mechanical tolerances.

2.4.3 End-to-end optimization of camera designs

The use of gradient descent to train deep networks has also made possible the use of backpropagation to update parameters of physical optical design by including the design parameters as part of the optimization itself. One of the first examples was

[1] https://www.zemax.com/

demonstrated by [22], who learned better color filter masks for image demosaicing. Such "end-to-end" systems may use gradient descent to automatically learn the computational part of the problem via the weights of a neural network, and the choice of components of the imaging system, such as aperture codes, to make the problem better suited to a particular environment.

There has been a growing interest in designing optical systems and deep learning models together using end-to-end frameworks, such as [24, 111, 122, 108]. These approaches typically define a parametric optical design, where parameters corresponding to optical components are differentiable with respect to the sensor values. A more detailed review is provided by [90].

2.4.4 Rendered Synthetic Data for Deep Learning

Optical design tools often make use of ray tracing, but often do no simulate the full environment. The use of engineering design tools, such as [7], have recently pushed for integrated simulation of the environment and optical design in order to validate designs in realistic environments. Recently, [95] has launched a simulation framework to generate training data for machine learning models such as deep neural networks. These simulation tools can be adapted to emulate specific camera designs or sensing modalities, and are thus well suited for the end-to-end optical design proposed in this paper.

2.4.5 Joint Camera and Algorithm Design

There has been a growing interest in designing optical systems and deep learning models together using end-to-end frameworks, such as [**?** 111, 122, 108]. These approaches typically define a parametric optical design, where parameters corresponding to optical components are differentiable with respect to the sensor values. A more detailed review is provided by [90].

The key problem with this approach is that the parametric model is rigidly defined, such that it would be impossible to discover novel designs outside the range of paramet-

ric values. This limitation suggests that a high-level "parametric optical architecture generator" is needed to propose truly novel designs. Such high-level architecture generators has been explored by the automatic machine learning community, such as the architecture grammar proposed by [9].

Chapter 3

Learning cues to locate hidden objects

The challenge of seeing around corners has been widely studied in recent years. Various solutions tackle different aspects of the problem such as the sensing hardware, recovery algorithms, or expanding the range of tasks that can be accomplished. One particularly interesting aspect of the problem is in the case of using consumer cameras for sensing around corners. This approach has many advantages such as reduced costs and relaxed need for additional hardware on current platforms (e.g. vehicles). Due to the reduced measurement dimensionality, such techniques are used primarily for tracking [53, 17]. Here we propose a data-driven approach for localizing objects in 2D around corners with consumer cameras. The suggested approach is robust to variations in the scene and does not require extensive calibration or assumed known geometry. Furthermore, the suggested approach naturally benefits from increased scene complexity that result in improved localization accuracy without additional modeling.

Data-driven approaches and convolutional neural networks (CNNs) have been widely used in many computer vision and computational photography problems, but have not been extensively explored in the context of sensing around corners. The main challenge in traditional sensing around corner algorithms is the need to solve an inverse problem based on a physical forward model. Such algorithms are very sensitive to calibration and to model mismatch. Data-driven techniques on the other hand directly

learn an inverse mapping from the measurement to the occluded scene. Thus, these methods do not need to solve a challenging inverse problem. Instead, the challenge in these approaches is found in data gathering and ensuring model generalizability. Our suggested approach tackles these challenges as follows:

Data gathering: instead of extensive experimental data gathering we use synthetic rendering to render the inputs (i.e. measurements) along with a known ground truth target location, alleviating the need for extensive experimental data gathering and labeling.

Model generalizability: to make sure our model is robust to novel scenes we introduce variations in the rendering parameters such as scene geometry and material properties. This helps the trained model to localize even for scenes that were not directly in the dataset.

Beyond robustness, a data-driven solution naturally benefits from additional cues in the scene that are hard to model such as complex geometry and intensity variations on continuous surfaces. Thus it has a potential to scale in localization quality with increasing scene complexity. For example, we show that by introducing a table into the visible scene our localization accuracy improves by 10%. This improvement is achieved with the same network training procedure and does not require a more complicated model. Furthermore, a data-driven approach has many other advantages in the context of sensing around corners as it naturally enables other computer vision tasks such as object counting, object classification, tracking, and full image reconstruction.

The main contributions of our work are:

- Robust data driven technique for 2D tracking of objects around multiple, unseen corner instances of different geometries (including L shape, T shape, and L shape with a table). The model is trained on synthetic data. The technique is based on consumer cameras.

- Demonstrated improved localization accuracy around corners by leveraging increased scene complexity with a data-driven algorithm.

- An efficient rendering procedure for around the corner scenes that is $\sim 20\times$

faster compared to vanilla rendering procedure.

3.1 Revealing hidden cues with differential imaging

The idea of "differential rendering" was made widely known by [30], who described a method for inserting rendered objects in photographs that preserved realistic shadows on surfaces. Previously, synthetic objects could be inserted using traditional matting techniques, but these methods do not look realistic since objects typically change how their surrounding surfaces are illuminated. The idea of differential rendering is to render two images: one that roughly models the real surface in a photograph, and another with synthetic objects added to the scene. When the rendered image with the objects is subtracted from the rendered image without the objects, the resulting differential rendering can be subtracted from the original photograph to model shadows and reflections.

Differential Imaging: We can build on this idea to capture two photographs: one with and one without a *hidden* object. Background subtraction is a common method for revealing small changes in photographic measurements. Notably, [17] used background subtraction to amplify the effect of hidden objects scattering light into a scene with an occluding edge. The shadow cast by this edge can be used to localize objects around a corner.

We show in this section how background subtraction reveals the "differential image" of a scene. Practically, new objects inserted into the hidden scene act like hidden illumination sources. This reduces the problem of tracking hidden objects to localizing hidden illumination sources. However, this picture is not entirely complete, as these hidden illumination sources are "differential" in the sense that they can contain both positive and negative values. A more accurate model would be to reproduce the differential light emitted from the surface of the object. Figure 3-2 explains this idea a bit further.

Figure 3-1: Two images of a cork were taken with and without a person standing just out of view of the camera. The person did not cast a visible shadow on the scene, and there is no obvious difference between the two images. However, when the image without the person is subtracted from the image with the person, a difference image shows a clear "shadow," as if the person were an illumination source. While subtle, these cues are detectable and can be explained using the principle of differential imaging.

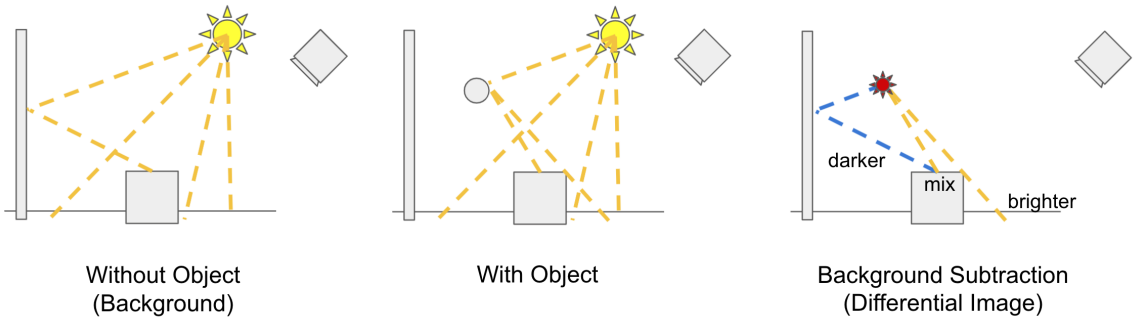


Figure 3-2: When an object is added to a scene, it interacts with paths of light that would pass through the object volume if the object was not present. When a background image is subtracted from an image with the object, all rays cancel out except the rays that either scatter from the object or are absorbed. Light rays that scatter make the object appear like an illumination source, and rays that are absorbed or redirected make the object appear like a "negative light source" for the corresponding direction. In effect, if the object is small it can be modeled by a anisotropic light source that can emit positive and negative valued light rays.

3.1.1 Temporal differential imaging

While the majority of our discussion so far has described the idea of differential imaging with and without the object present, we can also detect small motion of the hidden object. This can be accomplished by capturing images at time t and $t+1$ some time later, such that the differential image is $I_{t+1} - I_t$.

In general, light paths that intersect the *union* of the two volumes encapsulating the hidden object at two points in time are changed by object motion. We observed that the light paths that change the most are those that pass through the volume the object has entered and recently left.

For example, imagine a cube traveling in the direction of one of its face normals: a small "sliver" of volume will make up the changing volume displacement by the cube to the front and back of the object as it moves. In this way, providing temporal difference images should be sufficient for localizing objects or reconstructing the differential illumination created by their motion.

If the object strictly scatters light back into the scene, such that it appears as a purely non-negative differential illumination source when performing background subtraction, then the temporal differential image will appear like the scene is illuminated by a positive object and a negative light source. Localizing these two sources can indicate the direction of motion of the object.

Rendering Differential Images

In order to render differential images, a path tracer could be used to sample paths that intersect the object volume using importance sampling. Half of these paths would be selected to scatter off the object, and the other half would continue as if the object was not present, but their contribution to the rendering equation would be marked as negative. For small objects, this would be similar to rendering a scene illuminated by a point source that emits these rays (some directions positive, other directions the rays would be negative). As shown in Figure 3-3, if the contribution of negative paths is small, the hidden object can often by roughly modeled as an isotropic point source.

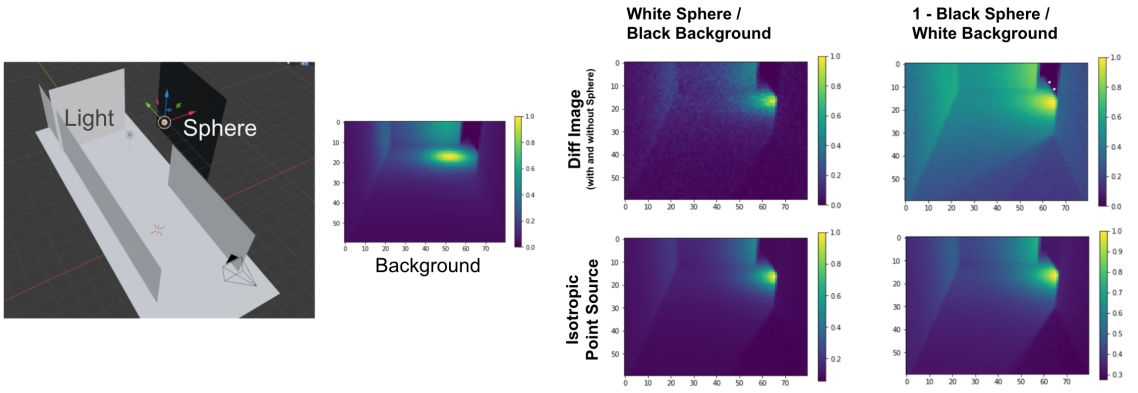


Figure 3-3: (Left): A scene was created with an object around a corner illuminated by a point light source. We show a black background surface, but also rendered images with a black sphere and white background surface. (Middle): A background image was rendered without the object. (Right-Left Column): When the object is the only source of reflected light, it is well modeled by a point source, generating a single shadow edge aligned with object. (Right-Right Column): When the object blocks light paths that normally would eventually contribute to the visible image, it generates a "virtual point source" by projecting negative valued light onto the background surface. We subtract the difference image from 1 to maintain positive values. Both the object occluding the background and the virtual point source contribute to two shadow edges.

With this basic premise, we can now pose the problem of locating hidden objects outside the line of sight of the camera as the problem of localizing light sources from photographs. There a few useful cues that are useful for solving this problem, such as shadows and specular reflections. In this chapter, we propose to develop an imaging system that learns to make use of these cues for the purpose of localizing hidden objects from differential images.

3.2 An imaging system that learns to use differential cues

The imaging system consists of a consumer camera that observes the visible part of the scene. In some cases we add additional light to ensure the hidden object is well illuminated. We then use images from this camera to generate differential images that are used to localize the occluded object (see Fig. 3-5).

The recovery algorithm is sketched in Fig. 3-4 and composed of an offline and

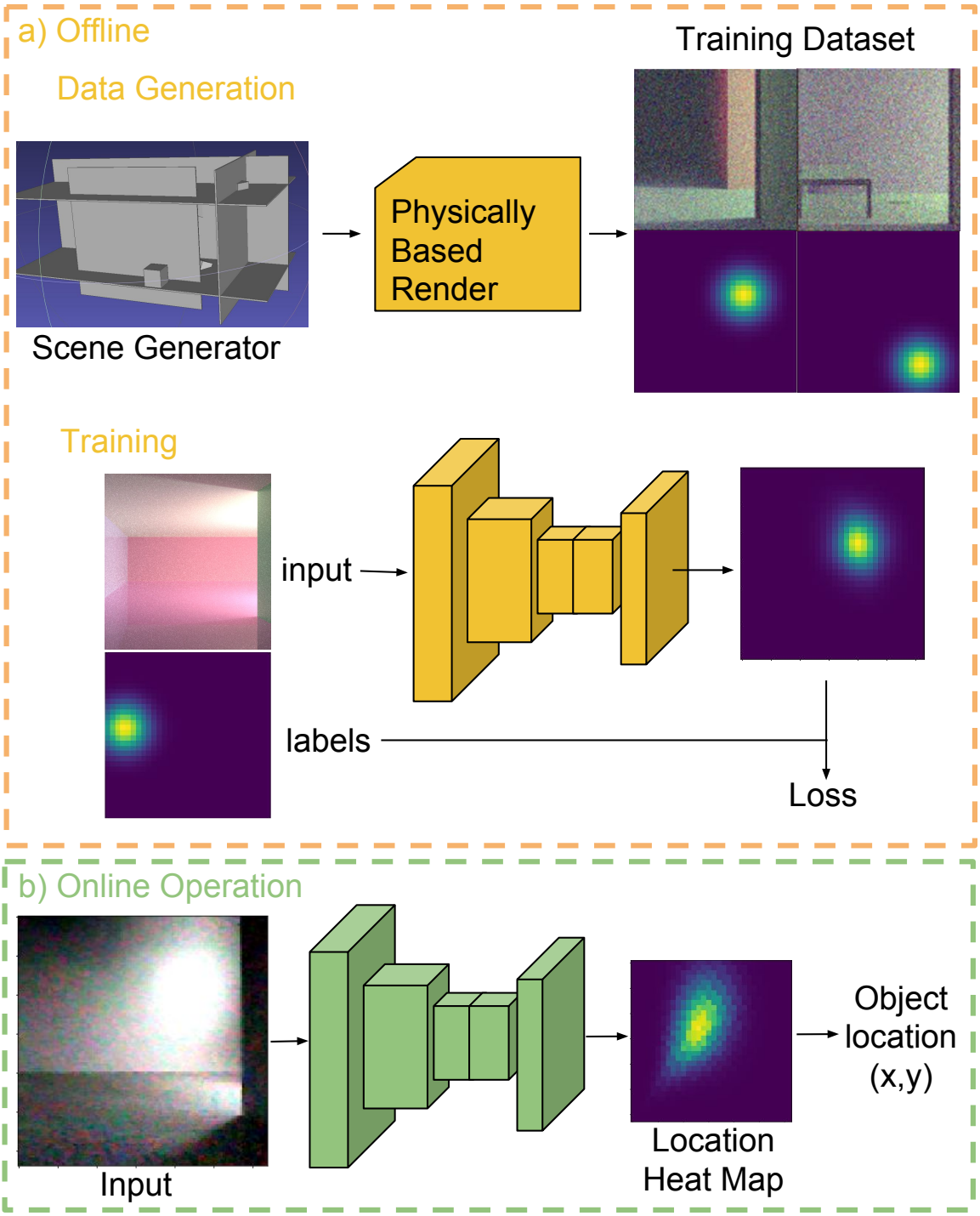


Figure 3-4: Overview of the Imaging Pipeline. a) In the online phase a scene generator produces a scene with random properties that are rendered to generate a dataset. The dataset is used to train a CNN to predict the hidden object location in a 32×32 grid. b) In the online phase, a background subtracted measurement is fed to the network for location prediction. This prediction is refined with a Gaussian fit.

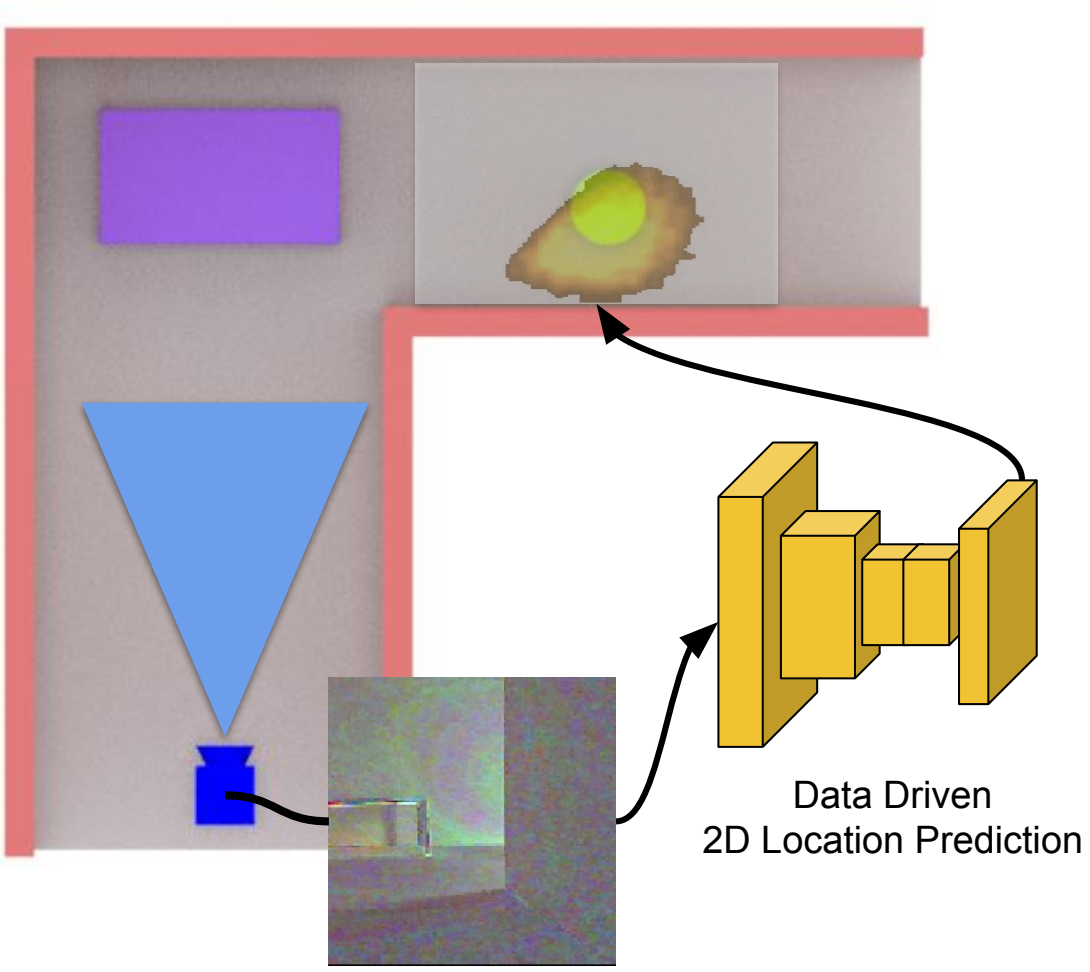


Figure 3-5: Leveraging complex geometry for seeing around corners. A data-driven technique is used to predict the 2D location of objects around corners. The network is trained only on synthetic data and robustly works on a wide range of geometries. Data-driven techniques naturally learn to leverage additional cues in complex geometries which are hard to traditionally model. This results in the ability to locate objects in 2D with improved accuracy compared to a simpler geometry with the same computational pipeline.

online steps. In the offline phase a scene generator produce random scene geometries that are rendered by a physics based path tracer. This synthetic data is used to train a CNN, which is trained using the known object locations used to render images with the rendering engine. In the online prediction phase, the captured differential image fed into the network, which produces probability of the object being located at each location in a 2D grid. The predicted location is refined by performing a Gaussian fit to the predicted probabilities on the grid, which produces a location estimate with sub-grid resolution. This section provides the details of the imaging pipeline.

3.2.1 Dataset Generation

We render a synthetic dataset to provide sufficient training data for the CNN. The dataset was created using Blender Cycles [15]. The scenes are produced by a scene generator written in python that generates a new scene file for each sample in the dataset. The scene generator is parameterized by various scene properties including geometry, materials and the textures composing the different objects in the scene, location of the occluded object, and noise characteristics of the measurement. To increase the robustness of the trained model we introduce perturbations to all of the generator parameters (similarly to [96]). These perturbations are sampled from uniform random distributions over physically plausible values. To introduce even further variability in the dataset we introduce clutter to the scene. The clutter appears in the form of cubes with random orientations and positions in the visible part of the scene. In this work we consider three types of scenes (see Fig. 3-6): 1) L shape corner, 2) T intersection corridor, and 3) L shape corner with additional geometry (a table in the visible scene).

One of the challenges in data generation for the seeing around corner problem is the need for physically realistic rendering that accounts for subtle changes in the hidden scene. Traditionally rendering such scenes accurately requires complete ray tracing which is computationally intensive. To overcome that challenge we propose an alternative faster rendering technique.

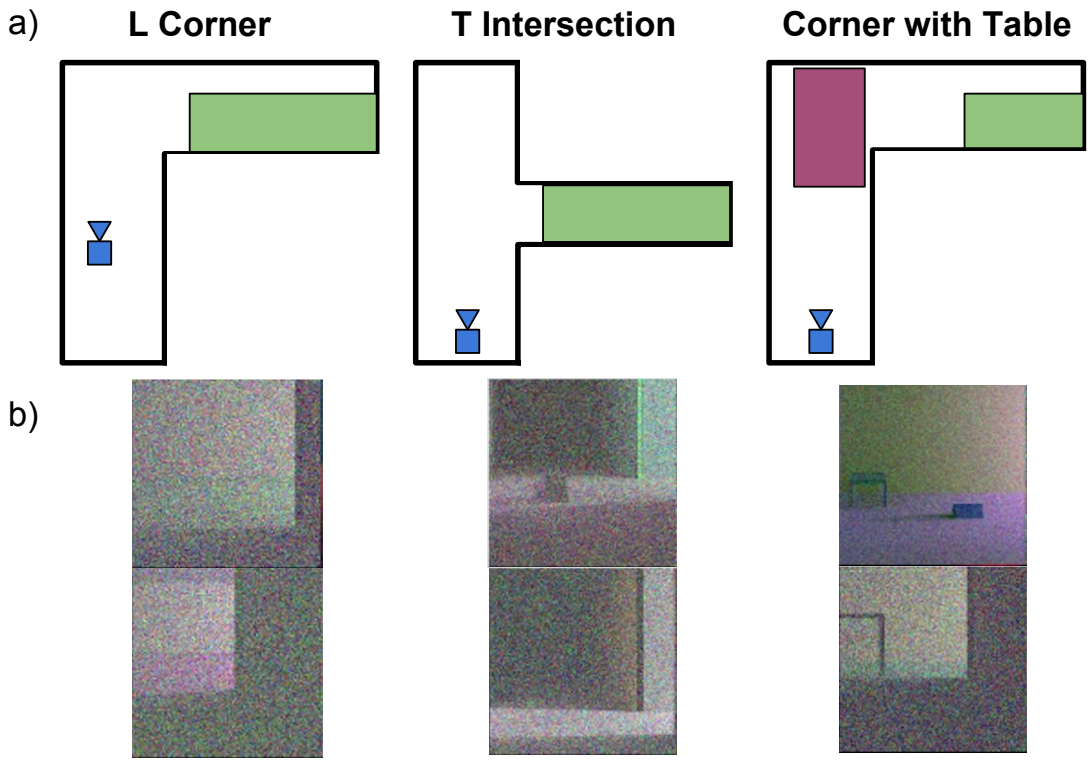


Figure 3-6: Data generation for different geometries. a) Demonstrating the three different geometries considered here: L corner, T intersection, and an L corner with a table. b) Examples of the rendered data for the different geometries. The green region in a) shows the area that the hidden object may be located. The purple region is a possible position and orientation of a generated table.

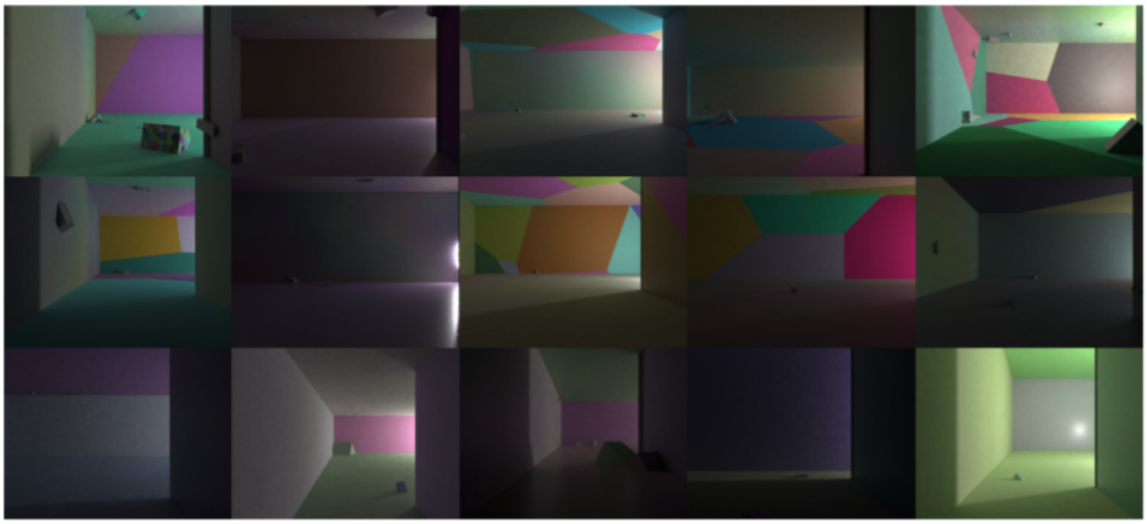


Figure 3-7: A large degree of variation was achieved in order to encourage domain randomization when producing the synthetic dataset. Variation in surface textures, material properties, corner geometry, and random clutter were added to encourage the network to not overfit to the training data.

Efficient Rendering of Non-Line-of-Sight Scenes

Similarly to previous techniques [53, 17, 114] that leverage a camera for imaging around corner we assume a background subtracted input. In terms of reconstruction, the main benefit of background subtracted input is an effective invariance to ambient light. Another major advantage of background subtracted inputs is that we can efficiently render them which greatly simplifies the rendering procedure.

Consider the scene in Fig. 3-5 with a bright ambient light. To calculate the background subtracted measurement we take the radiance from the scene without the object (i.e. the background) and subtract it from the scene radiance with the object. The result would appear as if the object is the only emitter in it. Thus we can efficiently render directly the background subtracted measurement by modeling the object as an emitter without additional illumination. As shown in Figure 3-2, rendering two images and subtracting them would eliminate the contribution from the majority of sampled paths in a path tracer and would be extremely inefficient.

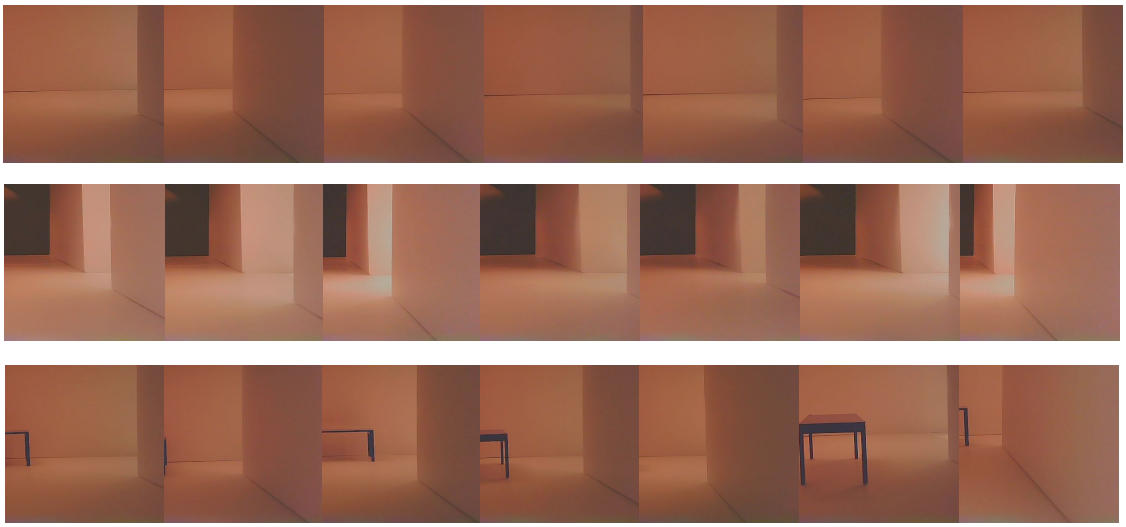


Figure 3-8: The 21 considered geometries. The measurements presented before background subtraction demonstrate the variability in the considered scenes. The top row corresponds to L corner geometry, the middle row to T intersection, and the bottom row to L corner with table. The ordering from left to right corresponds to the indexing in Figure 3-13. Each photo is a single instance of the 150 measurements per geometry.

An area for future work would be to develop path tracers that can efficiently perform differential rendering.

This rendering procedure takes approximately 20 secs compared to 300 secs for full rendering of the illuminated scene and subtracting the background from it (on an Nvidia GTX 1080 GPU). Fig. 3-6 shows several rendering examples.

3.2.2 Localization Prediction Network

The rendered data is used to train a CNN. The input to the network is the background subtracted 128×128 image of the visible scene. The network is relatively shallow and composed of three convolutional layers followed by two fully connected layers (see Fig. 3-9). The output is a 32×32 heat map of potential object locations. We selected a relatively small network architecture to reduce the risk of over-fitting and reduce the amount of training data required.

The ground truth location of the object is known from the rendering engine. It is essential to define an origin of the coordinate system on which the localization is

performed. Since the position of the camera with respect to scene is unknown, we set the origin to the closest corner in the scene (in the T intersection we choose the closer corner to the camera). To enable realistic scenarios, this position is not provided to the network, and the network effectively have to find this position and produce outputs with respect to that position.

It has been previously observed [17] that shadows cast from the corner's vertical edge encode the angular position of the occluded object. Thus we choose a polar coordinate system to represent the 32×32 prediction grid. Labels in this coordinate system should help the network to learn similar angular filters as well as other global features that provide the distance of the object.

In order to provide the network more meaningful gradients when it miss-classifies, the ground truth location is provided as a Normal distribution centered on the actual location. The distribution variance is a hyper-parameter that encodes the spatial dependency of labels (we use $\sigma = 3$). The Gaussian is applied in polar coordinates, so it naturally covers more area in euclidean coordinates when the object label is far from the corner.

Rather than predict the object location 2D coordinates directly, the network is trained to predict the probability of the object being located at a location on a predefined grid of 2D locations. We model this as a classification problem, which is known to be more stable and easier to train. The training loss is the mean of the sigmoid cross entropy over the location heat map:

$$\mathrm{CNN}_{\mathrm{loss}} = \sum_{i=1}^{K} \frac{1}{K} \max(w_i, 0) - w_i y_i + \log\left(1 + e^{-|w_i|}\right) \tag{3.1}$$

where w_i is the logit value, y_i is the ground truth label for the ith element, and $K = 32 \times 32$ is the number of elements in the grid. This loss represents multi-label classification task (labels are not mutually exclusive), where the labels are independent.

The output heat map is converted to a predicted location by fitting a 2D Gaussian to the heat map. Where the Guassian's mean μ_0 is the predicted location, and the variance Σ_0 is the uncertainty. While this approach could be extended to predict

multiple object locations, in this work we consider only a single hidden object.

3.2.3 Implementation Details

Rendered Scenes

We consider three scene types:

- **L Corner:** The geometry of this corner is parameterized by the 2D location of the two corners and the height of the walls.
- **T Intersection:** This is a hallway that splits to the right. It is parameterized by 2D coordinate of the close corner, the widths of the two hallways, the depth of the hallway, and the height of walls.
- **L Corner with Table:** In addition to the L corner parameters we add 7 table parameters: 3D size, leg thickness, table top thickness, and the table 2D position.

On top of these geometric parameters we also parametrize the optical properties of the different materials with surface roughness and three channel albedo.

All of the parameters are randomly sampled to produce a scene that is rendered by Blender.

3.2.4 Image Capture

The CNN model was trained using entirely synthetic data, but in practice we need some method of capturing images. We rely on light being reflected by the hidden object when it is introduced to the scene. Since we can not rely on ambient light serendipitously illuminating the hidden object, we provide a strong light source that is projected into the visible scene. The intent is to ensure the hidden object is illuminated by this light, providing a strong differential image signal. Ambient light may drift over time, so we introduce a 850nm bandpass filter to match our 850nm NIR illumination source. This light is not visible to the human eye, making our method suitable even with a lack of visible ambient light. Figure 3-10 shows the image capture setup for our table top evaluation environment.

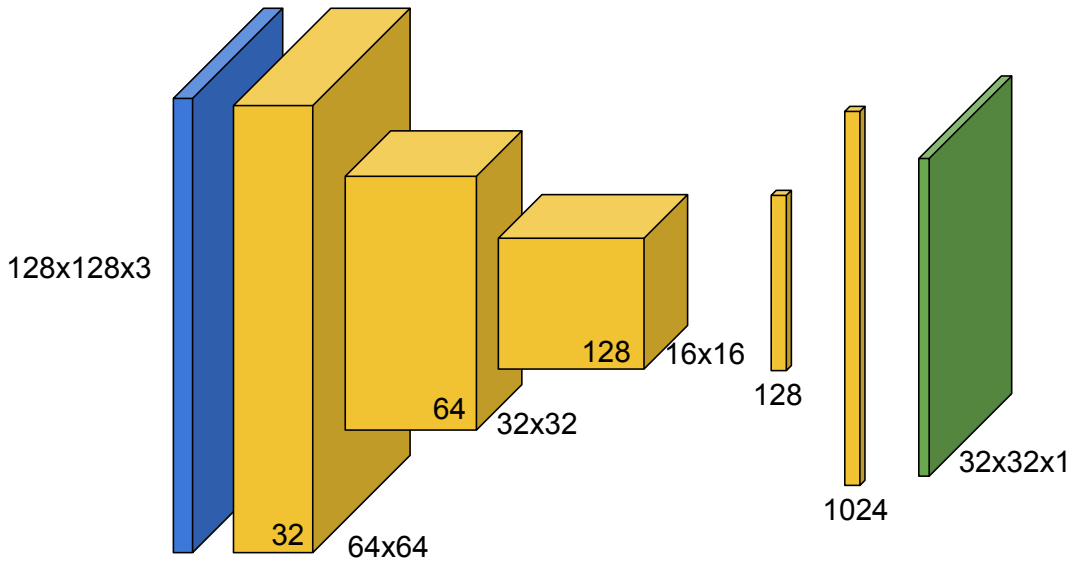


Figure 3-9: Overview of label generation and model architecture. a) The ground truth location is by selecting the origin point (corner), and calculating the object location in polar coordinates. This location is mapped to a 32×32 grid. Finally a Gaussian centered on the location is applied with a variance just large enough to cover adjacent pixels. b) The neural network detailed architecture. Three convolutional layers are followed by two fully connected layers.

3.2.5 Evaluation Environment

We developed a table-top evaluation environment to model variations in scene geometry, object size, and albedo. Figure 3-12 shows an overview of this setup. In some experiments, we use a fixed set of known locations to place the hidden object, while in others we use a color-label tracking system to label ground truth.

Since our CNN has been pre-trained on synthetic data, it can provide predictions in real-time with the full setup. Figure 3-11 shows an overhead view along with a screen capture of the full processing pipeline.

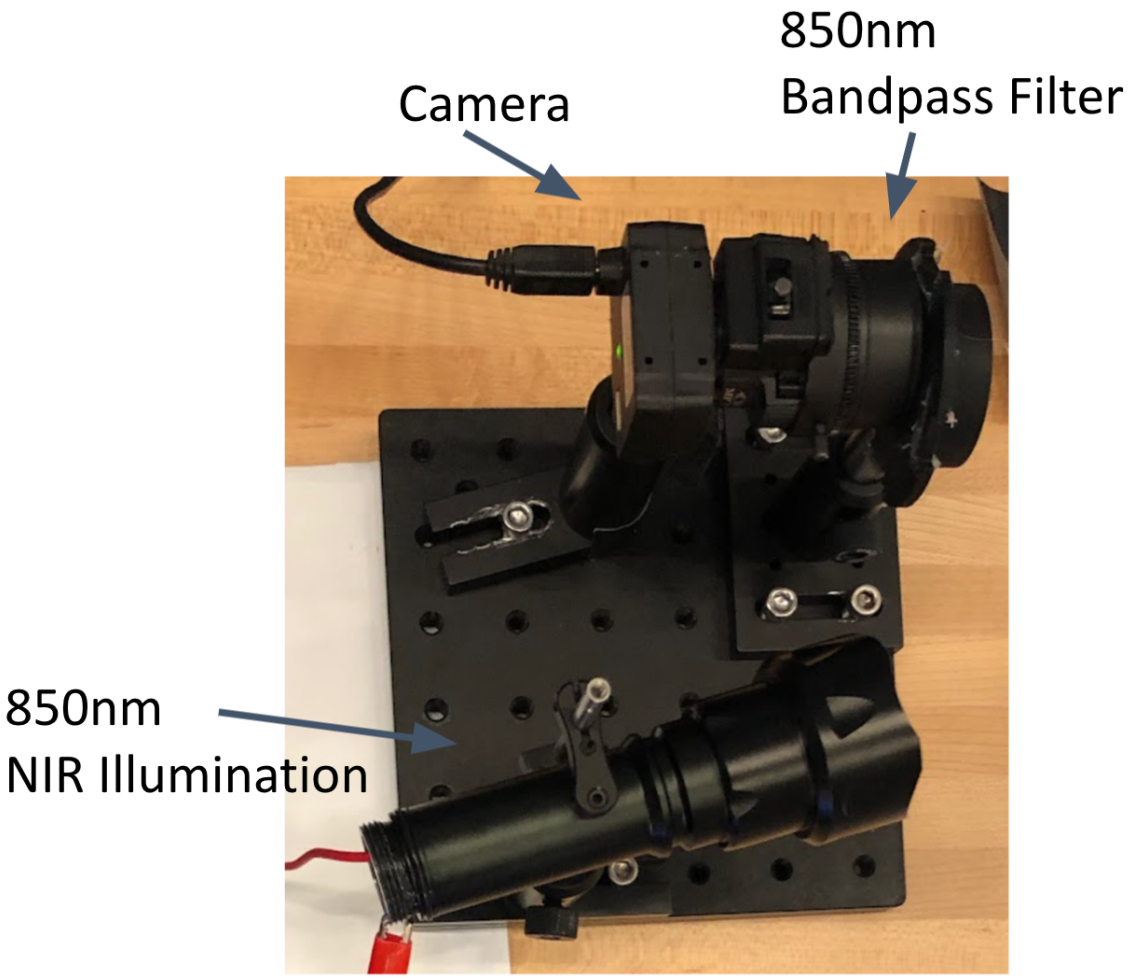


Figure 3-10: The image system used to valide the trained network on real data includes a camera and an illumination source. The illumination source is a flashlight that projects 850nm NIR light into the visible scene. The camera is fitted with a matching 850nm bandpass filter.

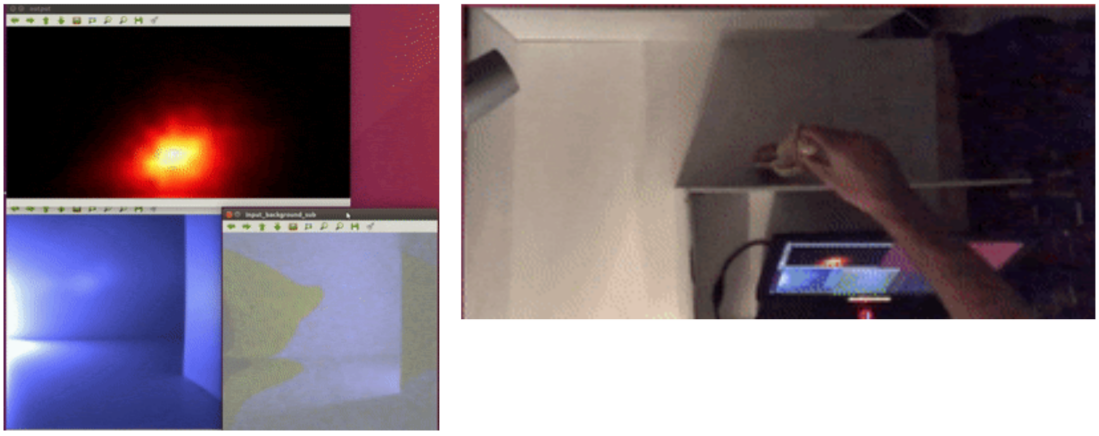
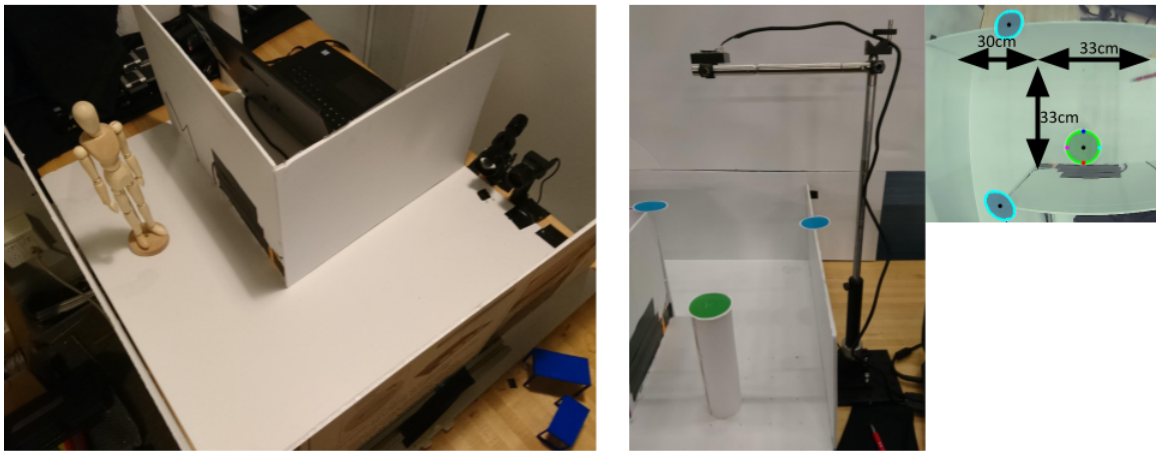


Figure 3-11: The validation environment consisted of a tabletop scene constructed from poster board to form an L-shape. All the walls can be moved to simulate different room geometries and camera positions. A color-label based tracking system was used to provide ground truth labels of the scene geometry and object position.



Hardware Setup **Ground Truth Tracking**

Figure 3-12: On the left, we see a screen capture of the live processing of the CNN network. The input image can be seen in the bottom left, with the background subtracted image shown to the immediate right. The predicted object location distribution produced by the neural network is shown above these two images. On the right, we show a top down view of the scene while manipulating a hidden mannequin object.

3.3 Experimental Results

To experimentally demonstrate the proposed technique we first train three CNN models (one for each of the considered geometries). We built a re-configurable scaled experimental setup that can be easily perturbed. For each scene type (L shape, T intersection, and L shape with table) we created 7 instances by moving the walls of the scene. Thus, each network is evaluated on 7 different corner instances, and our approach is evaluated on 21 different geometries. In each geometry we move a target object (a 6 cm wide cylinder) in the occluded scene and track its ground truth position with an additional camera positioned above the scene. On average we capture 150 images per geometry. Fig. 3-8 shows example measurements for these 21 geometries.

3.3.1 Trained CNN Models

Each network was trained with 15K rendered training examples. The dataset contains a significant amount of variation among samples, some examples from this training set can be found in Fig. 3-6. We set aside 10% of the generated data as a validation set. We use instance normalization for each input (zero mean, unit variance) and add noise to the normalized rendered data during training, sampled from a Gaussian distribution with a variance of 0.05. After training each model for 30K iterations with a batch size of 64 and learning rate of 0.01, the final sigmoid cross entropy loss was 6612 for L corner, 5280 for T intersection, and 6352 for the L corner with table geometry.

We found that it was important to add noise to the inputs of each training batch. Without adding noise to the training images, the network quickly learned to pick up on artifacts in the rendered images and memorize the training set.

3.3.2 Error Metrics

We measured the error between the ground truth labels and our network predictions using three primary metrics: angular error (θ), range error (ρ), and the Euclidean distance (d) between the predicted and ground truth location. As can be seen in

		L	T	L+Table
Synthetic	d (cm)	3.33 ± 2.16	2.07 ± 1.48	2.95 ± 1.82
	θ	-0.05 ± 0.08	-0.01 ± 0.06	-0.04 ± 0.08
	ρ (cm)	0.40 ± 3.13	0.49 ± 2.26	0.95 ± 2.76
Experiments	d (cm)	10.35 ± 5.14	7.88 ± 2.84	8.32 ± 4.67
	θ	-0.04 ± 0.31	-0.43 ± 0.28	-0.12 ± 0.30
	ρ (cm)	1.96 ± 9.64	-3.57 ± 5.14	1.59 ± 7.88

Table 3.1: Localization prediction error for different geometries

Table 3.1, the models produce good localization performance even when provided with a large range of possible scene geometries for each scene type.

The error metrics we used highlight the difference between 1D angular localization (θ) cues (e.g. corner edges) and 2D localization that benefits from the fusion of other cues in the scene such as intensity changes on continuous surfaces or complex geometry with occlusions. Since the occluded region starts at an unknown distance from the camera, we define our occluded object coordinates from an origin point on the ground plane at the start of the occlusion boundary (i.e. the corner edge). We orient our right-handed coordinate system such that the y-axis points towards and perpendicular to the ground plane and the x-axis points towards the occluded scene around the corner. From this, we define the position error as follows:

Angular and Range Error (θ, ρ): The angular and range error is the difference from ground truth in polar coordinates, where we define θ as the angle in degrees from the x-axis to the z-axis.

Euclidean Distance (d): The Euclidean distance from the prediction to the position of the occluded target in the x-z plane. This metric should be compared to the 6 cm width of the target.

3.3.3 Varying Geometry and Object Albedo

In Table 3.2, we show the localization error as average euclidean distance from ground truth over a range of L-corner scene for different object sizes and albedo values. The objects were constructed of diffuse card-stock paper formed into a cylinder shape with diameters 4.5cm, 6cm, and 8cm. The ground truth albedos were not measured directly,

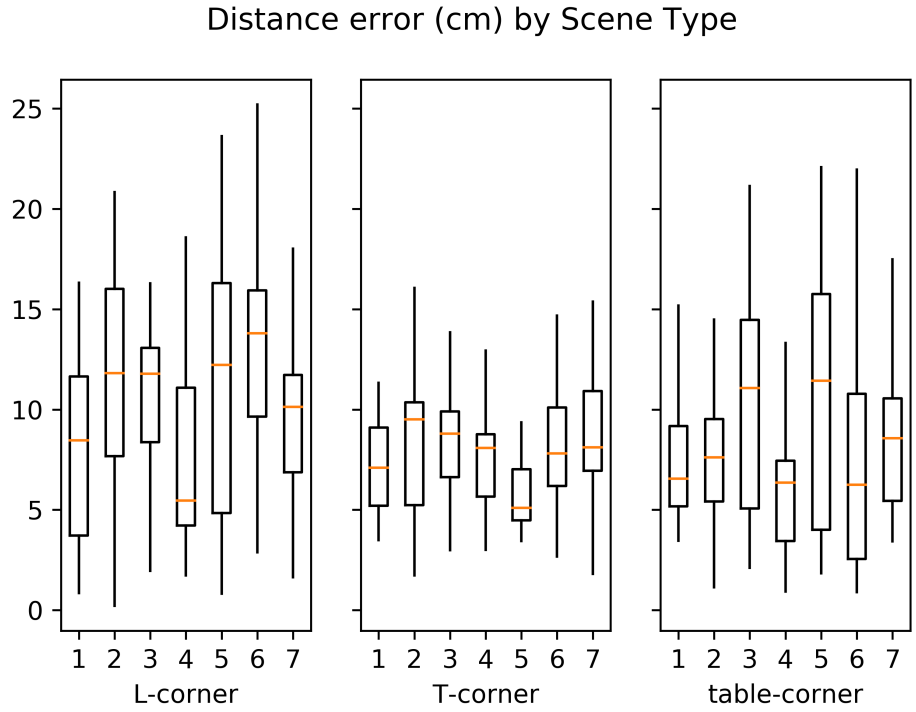


Figure 3-13: Box plots showing the distribution of error for the 21 experimental geometries (7 geometry instances within the 3 corner types). The lines show the min and max values and the boxes correspond to the region from the lower to the upper quartile of the errors. The orange line shows the median error. We note that the complicated geometries (T intersection and L corner with table) are superior to the simple L corner across all geometry instances.

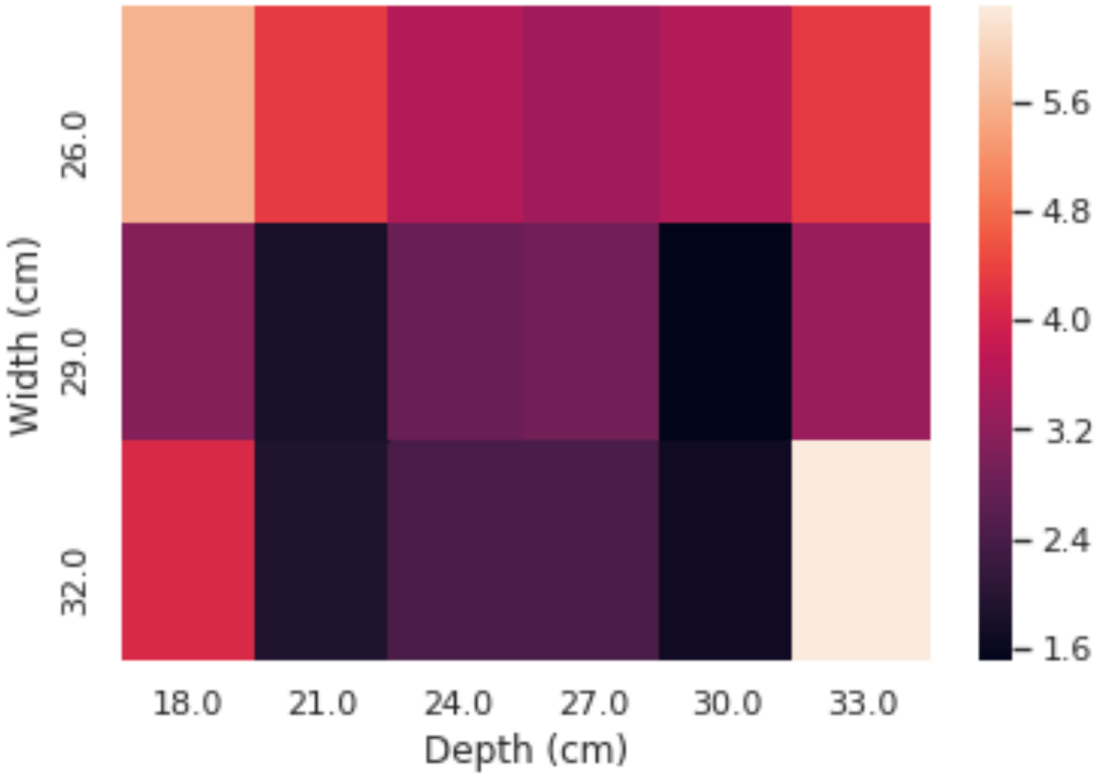


Figure 3-14: This figure shows the average 2D distance error in cm of localization by the network when the corner position was varied. The corner position was moved to cover a wide range of positions in camera screen space. The average error was lowest when the corner position was roughly centered in in the lower right quadrant of image space, which closest resembled the average position of the corner in the synthetic training data.

but where approximately 0.95 for "white", 0.7 for "anthracite", 0.2 for "dreadnought" and 0.05 for "black".

It is interesting to note that the average localization error was less than the diameter of the objects in most cases, and localization performance improved for larger objects. Large objects reflect more light into the scene which may have reduced the signal to noise of the background subtracted images, improving localization performance. Notably, the network was robust to deviations from the point-source approximation of the hidden objects, and sharp shadow boundaries did not appear necessary for good localization.

size \albedo	"white"	"anthracite"	"dreadnought"	"black"
4.5 cm	4.9 cm	6.7 cm	5.6 cm	Failed
6 cm	4.1 cm	4.7 cm	3.3 cm	Failed
8 cm	3.7 cm	4.6 cm	3.4 cm	Failed

Table 3.2: Error by object size and albedo. Three cylinder diameters and four albedos were chosen to measure the average error of object localization when the hidden object varied in size and reflectance.

3.3.4 Varying corner scene geometry for a single trained network

The object location prediction network was trained with widely varied synthetic data. The idea was to ensure the trained network was robust to model error introduced by different corner geometries, surface albedos, and even clutter objects (see Figure 3-7).

The core assumption when creating training data was to sample from a fixed "scene class" such as an L-corner scene. Within this class, the exact position of walls could change drastically. That said, we assume the camera is roughly pointed at the visible scene "corner" so that the corner wall meeting the ground was centered roughly in the lower quadrant in image space. In our experiments, we varied the position of the L-corner and evaluated the average error of the L-corner scene. Changing the position of the L-corner changed both the length and width of the hallways, but also the position of the corner-floor intersection in the camera image space.

3.3.5 Discussion

The final optimized localization errors and their standard deviation are presented in Table 3.1. We show the errors for each type of scene (L, T, and Table+L) for synthetic validation and experimental data. As expected, a trained CNN performs well on the synthetic validation set. However, the model does demonstrate generalization to the experimental data without any additional fine tuning.

To demonstrate the robustness to scene perturbations we plot the localization error for the 21 different geometries (3 types and 7 instances) in Fig. 3-13. We note that the error is comparable across the different scene geometries within a scene class. The

scene class plays a very important roll in the final error. As predicted, more complex scenes such as the T or L corner with table, achieve lower errors overall.

Angular vs. Absolute Position Error

Overall, we observed that angular predictions were much more accurate than 2D localization. Figure 3-15 shows this result for a representative test scene over 12 ground truth locations. This is not surprising, one of the strongest cues for localization is the shadow cast by the occluding edge, and the shadow angle relative to the wall contains all the necessary information to find the angle to the hidden object.

Camera scale ambiguity

It is more surprising that 2D localization is possible at all. The network must be using a combination of shadow edge angle, floor illumination, and diffuse back wall reflectance to triangulate the object position. While the 2D location was reasonably accurate in a relative sense, it often seemed globally shifted or scaled in non-intuitive ways. Despite a rather extensive ablation of possible sources, there was not an clear reason to explain the various bias introduced in the 2D localization.

Perhaps the best explanation is that scale and depth of perspective cameras is inherently ambiguous. It is amazing that the network was able to estimate 2D location at all in this context. As such, one of the most reliable outputs of the network is the *relative* position of the hidden object in the sense of moving closer or further away from the corner. While this was difficult to measure experimentally, we observed that the network had an impressive ability to predict rather small movements of the hidden object, even if there was a large bias in the error relative to ground truth.

3.3.6 Practical real-time implementation

In order to show the practicality of this approach, we built a full-scale real world scene with configurable wall positions and floor materials. This full-scale scene demonstrated that the localization network can perform well in corner environment scenes with no

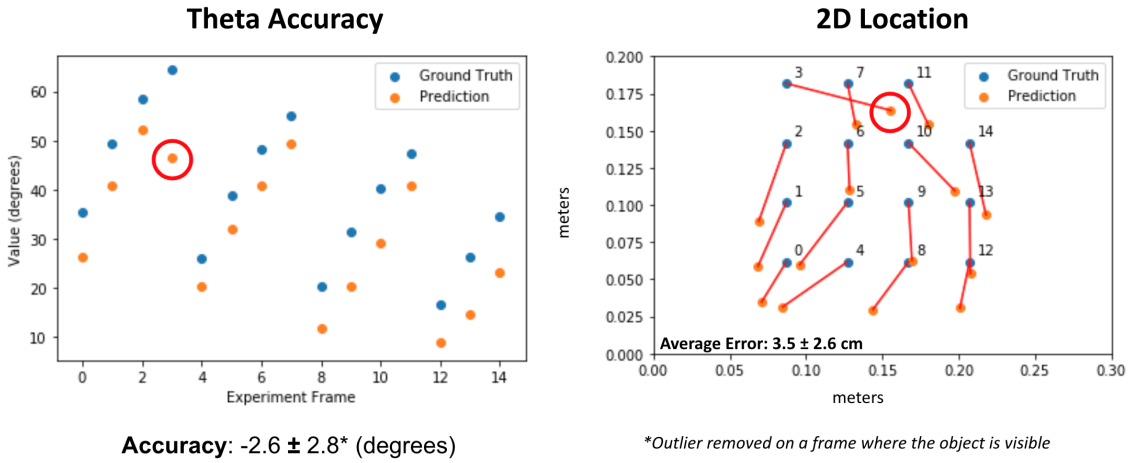


Figure 3-15: The figure to the left shows the ground truth and predicted angles to the object, using the corner and side wall for reference. The absolute position errors often contained difficult to explain bias, but the relative differences were often consistent for a given scene.

extrinsic calibration, and the synthetic training data was sufficient for even real-world scenes and not just table-top poster-board models.

The demo system automatically subtracts frames corresponding to the past when the network is confident a hidden object is not present, this provides a background subtracted image where the hidden object appears like an illumination source.

3.4 Limitations

The key limitations of the suggested approach are summarized below:

1. **Assumes limited prior knowledge of scene geometry:** To train the network we need a parametric model of the scene geometry. Fresh geometries (like an X intersection) will require new rendering and training.

2. **Background subtraction:** Similarly to previous techniques to sense around corners with traditional cameras we assume a background subtracted measurement. This is a relatively strong assumption that limits this and prior works. Another aspect of this limitation is the assumption that the object is well lit.

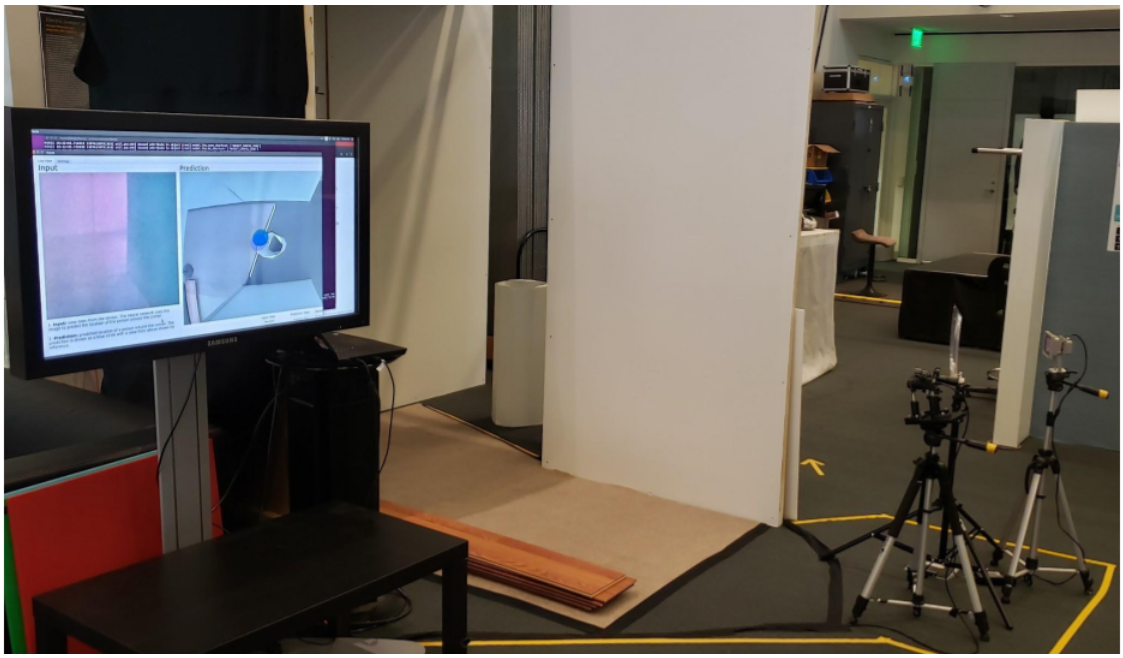


Figure 3-16: The full-scale test scene of a real-time implementation. The screen on the left shows a GUI output showing the background subtracted frame used as input to a pre-trained deep neural network. The network predicts the hidden object location as a distribution in 2D from a top-down perspective relative to the camera. A blue dot is used to mark the mean of a Gaussian fitted to this distribution, with transparency corresponding to the variance of the Gaussian fit. A top-view camera (not used during inference) is super-imposed to provide a view of ground truth. On the right, a camera with a 850nm bandpass filter observed backscattered light. An 850nm flood illumination source is imaged onto the back wall of the scene using a fresnel lens placed in front of it.

3. **Localization recovery:** In this work we focused only on robust localization in novel scenes. We are motivated by prior work [114] that demonstrated other tasks such as identification on a particular corner and leave such demonstrations to future work.

4. **Assumed single object around the corner:** In this work we assume a single occluded object. Future work would explore the possibility of training networks to recover multiple objects and to predict the number of occluded objects.

5. **Assumed diffuse objects:** Our efficient rendering procedure effectively assumes the object is Lambertian. Accounting for objects composed of more complex BRDFs would also have to be dependent on knowledge of the illumination sources in the scene.

3.5 Future Work

3.5.1 Combining Data-driven and model-based approaches

Inspired by [25], the recent availability of differentiable renderers make it possible to combine data-driven and model-based approaches to solve the localization problem. This section describes this idea, with a focus on validating physical plausibility of the prediction made by a pre-trained neural network and the ability to fine-tune the resulting estimate.

While forward models for light transport that are fully differentible have recently become highly capable, there may be instances when the forward model has discontinuities leading to zero-valued gradients. In these cases, a sampling based approach may be well suited, and in the next section we describe the use of variational optimization to address this situation.

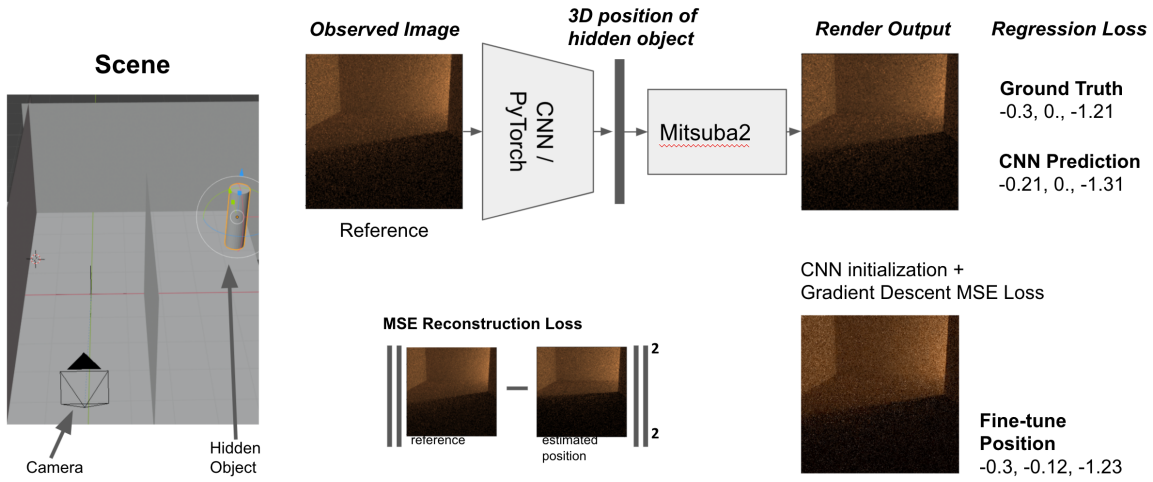


Figure 3-17: (Left): A synthetic scene was constructed that resembles the table top experiments. (Right): A image was rendered and passed to the trained CNN. The predicted position of the hidden object was used to initialize the location of an object in a differentiable renderer. By minimizing the mean squared error reconstruction loss, the object position was fine-tuned to ensure data consistency.

3.5.2 Variational Optimization

One of the challenges with neural networks is the black box nature of their performance. Here, we propose to use the neural network as a hot start for a physically constrained recovery algorithm.

Specifically, our goal is to find the location of the hidden object, such that when we render a measurement given the object location (similarly to our data generation procedure) it will be as close as possible to the actual raw measurement. When this difference is low our confidence in reconstruction quality is high. This is captured by:

$$L_r(z) = \|r(z) - x\|_2^2 \tag{3.2}$$

where x is the vectorized measurement and $r(z)$ is the vectorized rendered output of our rendering engine, given the object location z.

To find z we use variational optimization. The theory of variational optimization procedure is extensively discussed in [110, 72]. For completeness we provide the high-level overview of the variational update procedure. In our case, the variational

upper bound of Eq. 3.2 is:

$$U(z) = [L_r(z)]_{z \sim \mathcal{N}(\mu, \Sigma)} \tag{3.3}$$

where the sampling distribution is a Gaussian defined by $\mathcal{N}(\mu, \Sigma)$. We seek $\bar{z}$ such that:

$$\bar{z} = \arg\min_z U(z) \tag{3.4}$$

To solve Eq. 3.4, we use an iterative algorithm with the following update rule:

$$\begin{aligned} \mu_j &= \mu_{j-1} + \lambda_\mu \partial_\mu U(z) \\ \Sigma_j &= \Sigma_{j-1} + \lambda_\Sigma \partial_\Sigma U(z) \end{aligned} \tag{3.5}$$

here, j is the number of iterations, λ_μ and λ_Σ are the step size for the two parameters, and ∂ is the partial derivative with respect to the sub-script parameter. The iterative solution is initialized with the output of the neural network as described in the previous subsection.

The challenge is of course calculating the gradients in Eq. 3.5, we approximate them using the following steps:

1. Sample points $z_m \sim \mathcal{N}(\mu_j, \Sigma_j)$. Here we sample $M = 5$ points such that $m = 1..M$.

2. Calculate $L_r(z_m)$.

3. Calculate $\partial_\mu \log \mathcal{N}(z_m)$ and $\partial_\Sigma \log \mathcal{N}(z_m)$.

4. The gradients are finally approximated by:

$$\partial_\mu U(z) \approx \frac{1}{M} \sum_{m=1}^{5} L_r(z_m) \partial_\mu \log \mathcal{N}(z_m) \tag{3.6}$$

   with a similar equation for $\partial_\Sigma U(z)$.

In our experiments we found that using the variational optimization procedure alone was not sufficient due to slow convergence and modified mean's gradient such

that:

$$\partial_\mu \hat{U}(z) = \alpha \frac{L_r(z^-) - L_r(\mu)}{z^- - \mu} + (1-\alpha)\partial_\mu U(z) \tag{3.7}$$

where $z^- = \arg\min_z L_r(z_m)$ and α is a weighing term between these two estimates. The left term in the sum is effectively a finite difference computed with respect to the point with smallest cost. In all of our experiments we used a fixed $\alpha = 0.5$ and $\lambda_\Sigma = \lambda_\mu = 0.1$.

Figure 3-18 shows the cost function as a function of number of iterations.

The following points discuss different aspects of the variational optimization algorithm:

Trade off between the two gradient estimates of μ: First we note that even if $\alpha = 1$ the variational optimization still runs to compute $\partial_\Sigma U(z)$. Calculating Σ is essential to adapt the sampling Gaussian according to uncertainty or confidence in different axis and effectively minimizes the variational upper bound. Furthermore, for sufficiently small variance, $p(z|\mu, \Sigma)$ approaches a Dirac delta function and the variational upper bound is equivalent to the minimizer of L_r. Lastly, we note that finding the global minimum of the variational upper bound is probably intractable since L_r is extremely non-convex, and thus finding the minimum relies on a good initialization that is provided by the CNN.

Estimating $r(z)$ and rendering procedure: To calculate $L_r(z_m)$ we must render M measurements for each iteration. We can further accelerate this process and render noisier versions of the measurement. This can be achieved by using a path tracing algorithm with less sampling points. In practice we found that using an efficient Blender Render was sufficient.

Details for calculating gradients

We optimize over the variational upper bound, which follows from Jensen's Inequality:

$$\min_z L_r(z) \leq [L_r(z)]_{z \sim p(z|\beta)} \tag{3.8}$$

where β are parameters of some distribution we use to sample z from $L_r(z)$.

Using the equality:

$$p(z|\beta)\frac{\partial \log p(z|\beta)}{\partial \beta} = \frac{\partial p(z|\beta)}{\partial \beta} \tag{3.9}$$

the gradient of the upper bound with respect to our sampling distribution is then:

$$\frac{\partial}{\partial \beta}[L_r(z)]_{z\sim p(z|\beta)} = [L_r(z)\frac{\partial}{\partial \beta}\log p(z|\beta)]_{z\sim p(z|\beta)} \tag{3.10}$$

which we write $\partial_\beta L_r(z)_{z\sim p_\beta}$ for brevity.

For our experiments, $p(z|\beta)$ is a multivariate gaussian distribution, parameterized by the mean μ and covariance matrix $\mathbf{\Sigma}$, so that $\beta = (\mu, \Sigma)$. We now present the gradient of the log of our sampling distribution with respect to these parameters:

$$\frac{\partial}{\partial \mu}\log p(z|\mu,\Sigma) = \Sigma^{-1}(z-\mu) \tag{3.11}$$

$$\frac{\partial}{\partial \Sigma}\log p(z|\mu,\Sigma) = \frac{1}{2}(\Sigma^{-1}(z-\mu)(z-\mu)^T\Sigma^{-1} - \Sigma^{-1}) \tag{3.12}$$

This implies that we can optimize for μ and Σ via gradient descent, and recover our estimate of θ and ρ using our final μ. We further parameterize Σ, ensuring the diagonal entries are positive and the matrix remains positive semi-definite.

We approximate the expectation by taking the mean of M samples of z from $p(z|\mu,\Sigma)$.

3.6 Conclusion

We introduce a data-driven technique for localizing an object around a corner. The technique is robust for variations in the corner geometry and experimentally demonstrated on three different scenes. We also measured the performance of the system over a range of values for the albedo and size of hidden object and changes to the geometry of the scene. We found that our trained CNN performs well over a wide range of scenes with zero calibration. This robustness is achieved by training the network on a wide range of synthetic scene variants. The main advantage of the

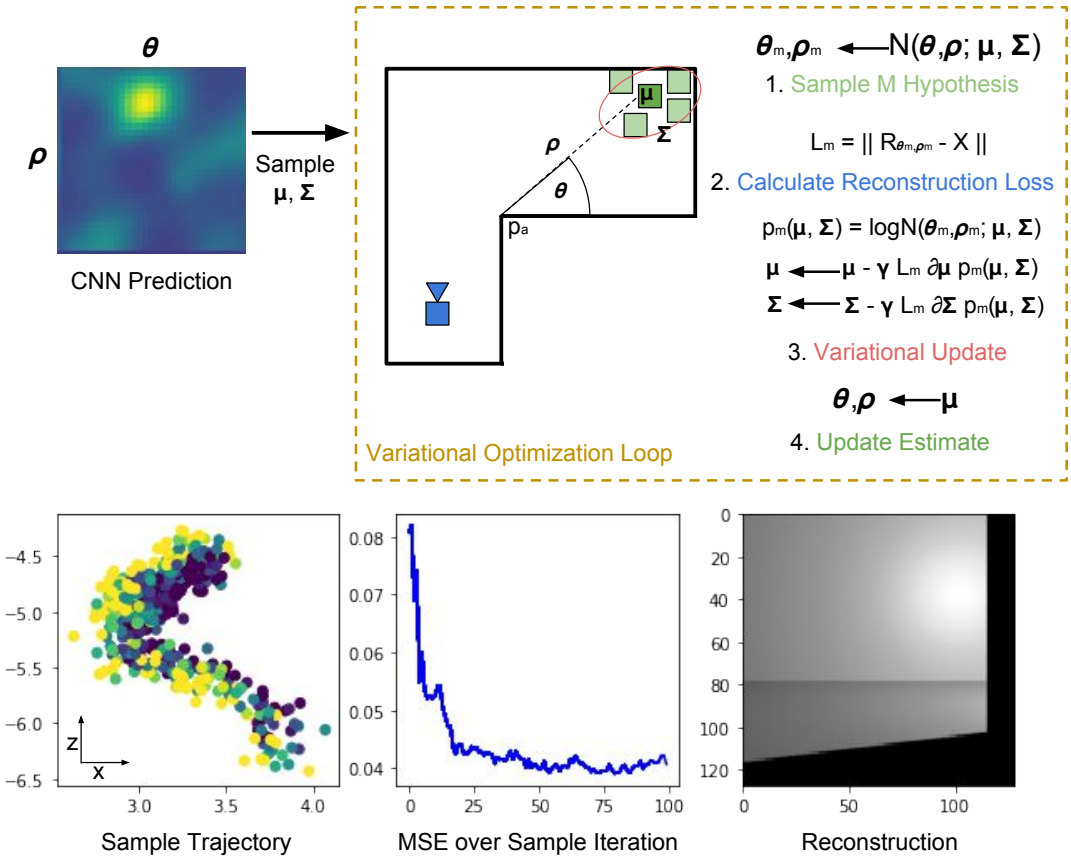


Figure 3-18: An end-to-end refinement approach for object localization around the corner can be applied when using variational optimization. The advantage of such an approach is that the forward model does not need to be differentiable. The bottom row shows the trajectory of the predicted 2D location of the object as updates are applied, along with the MSE loss for each iteration. The bottom right image shows the final reconstruction of the observed image from a simple, but fast, non-differentiable renderer.

proposed data-driven technique for seeing around corners is the inherent ability to leverage complex geometries with the same computational pipeline.

We demonstrated our approach on a full-scale test setup, and found that the network was well-suited for identifying the relative movement of the hidden object. Some obstacles remain, such as overcoming scale ambiguity inherent in using a monocular image source and handling arbitrary scenes. We propose future work that incorporates a rendering engine to make refinements to object positions by optimizing for a data-consistency loss. We believe that such hybrid approaches are promising as data driven methods are more widely used in inverse problems like localizing around the corner and beyond.

Chapter 4

Constraining Light Source Localization using Visible Occlusion Boundaries

This chapter describes the geometric constraints for localizing objects from cast shadow edges. This analysis provides a framework to determine if localization is possible for a given scene using shadow edges.

4.1 Solving for point localization with known shadow edges

If the shadow casting edges and the correspondence to the cast shadow are known, point localization becomes a problem of plane intersection. This section describes this idea, including the relevant geometry and solution to the optimization problem.

Localizing illumination sources from photos has numerous potential applications spanning many industries. In consumer photography, lighting estimation is helpful for image relighting, white balance, and flash removal tasks. Augmented and virtual reality is concerned with realistic object insertion, which requires an accurate incident illumination estimate, and scene capture for representing captured images of the

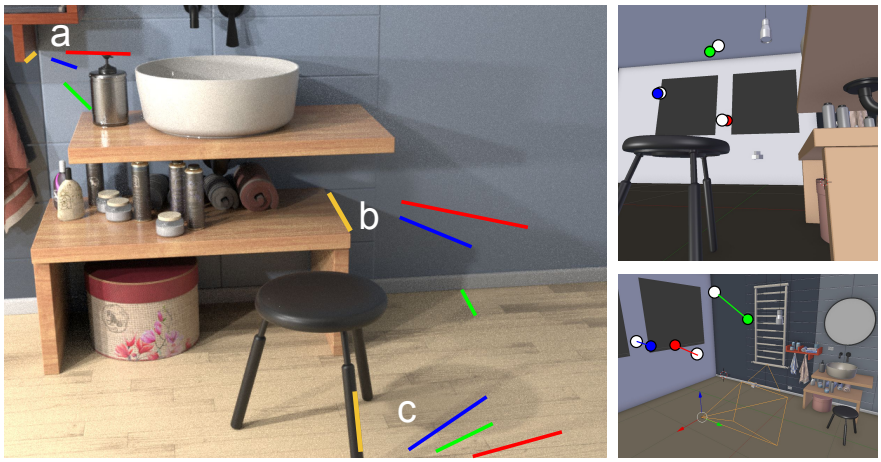


Figure 4-1: *Left*: With a correspondence between edges (a,b,c) and shadows (colored lines), we can estimate the location of multiple hidden point illumination sources. *Right*: We use geometric constraints to validate these shadow proposals and accurately estimate the 3D location of hidden illumination sources.

real world in virtual environments. Self driving cars and autonomous robots rely on accurate 3D reconstruction of the world, often from camera sensors, from which additional lighting cues can help resolve common perception errors. In remote sensing and satellite imaging, better understanding of the interaction of known illumination sources such as the sun could improve understanding of objects on the ground.

One of the strongest cues for illumination source position are cast shadows. Shadows cast by objects illuminated by point sources have been well studied in the context of computer graphics. In this chapter, we investigate how to utilize the geometry of cast shadows to localize point sources from images. Our approach is to model the problem as the *inverse* of shadow volumes, which are typically used in computer graphics to calculate the visibility of scene points to point light sources.

4.1.1 Contributions

Our key insight is that shadow casting edges largely constrain incident illumination recovery. We summarize our contributions as follows:

1. Introduce a framework for establishing the geometric relationships of hidden illumination sources to the observed scene features in the image space.

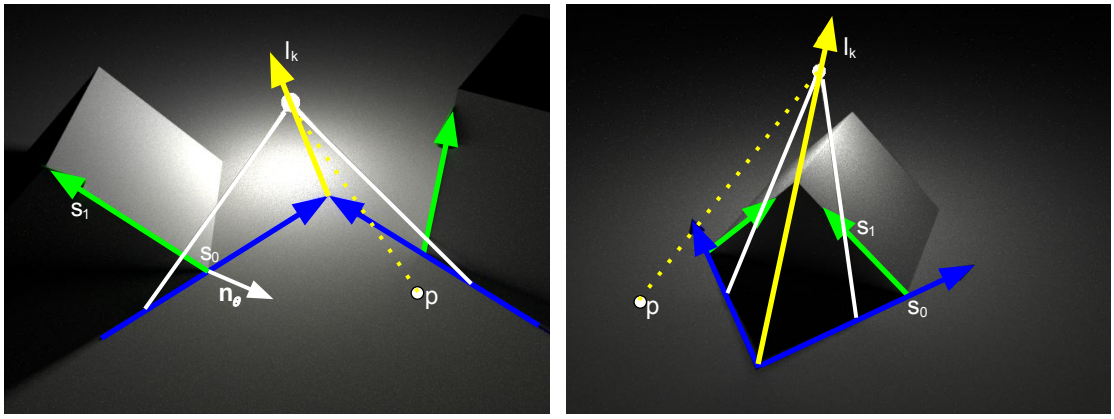


Figure 4-2: Visualization of the geometry for intersecting two planes associated with two edges. *Left*: Unconnected edges can be combined even if they are not parallel, restricting possible point locations to lines in 3D. *Right*: The same geometry can describe connected edges, which also constrain illumination positions to a line in 3D.

2. Identify some common image features and how they impact localization performance.

3. We demonstrate our methodology for localizing a single point source, with extensions to multiple point reconstruction.

4.2 Analyzing shadow edges

We show in this section that edges in the scene geometry can be used to define a scene-specific coordinate system that relates hidden point light source position to shadow edges. We show that with 3 non-parallel edges that cast shadows in the visible part of the scene, we can identify hidden point illumination positions in 3D by finding the intersection of planes. Shadows from 2 edges limits the possible location to a line (Figure 4-2: yellow ray), while a single edge restricts the point location to a plane (Figure 4-2: green and blue arrows).

For simplicity, we assume straight edges formed by the intersection of flat faces, such as edges of a polygonal mesh. We leave the extension of our approach to continuous curved surfaces for future work. We also assume shadows, edges, and their correspondence have been provided, and a per-pixel depth is available at capture time.

We demonstrate in the Results section some results that relax these assumptions, as in some cases very little annotation is required.

4.2.1 Shadows Cast from a Point Source

We begin with a model for localizing a single hidden point source. As in Figure 4-2, we consider a scene point, p, a 3D vector corresponding to any surface position in the visible scene, the illumination source position, l_k, and two points on a scene geometry edge, s_0 and s_1.

In order to describe the position of a hidden light source relative to our scene, we will construct a coordinate system, starting with a single edge, as illustrated in Figure 4-3. Given two points on the edge, the goal is to describe all the planes that intersect those points using a single scalar, θ. As we will see later, two more edges that are not parallel will enable us to describe the position of a point in 3D euclidean space using a set of such scalars: $(\theta_0, \theta_1, \theta_2)$.

Single Edge

We first construct a plane that lies along our edge, parameterized by θ. The plane normal, $\mathbf{n}_\theta$, is defined using Rodrigues' rotation formula:

$$\omega = \frac{s_1 - s_0}{||s_1 - s_0||} \tag{4.1}$$

$$\mathbf{v} = \hat{s}_0 \times \omega \tag{4.2}$$

$$\mathbf{n}_\theta = \cos(\theta)\mathbf{v} + \sin(\theta)(\omega \times \mathbf{v}) + (1 - \cos(\theta))(\omega \cdot \mathbf{v})\omega \tag{4.3}$$

We use a normalized s_0, $\hat{s}_0$, to produce a unit vector, $\mathbf{v}$, orthogonal to ω. For scene measurements that consist of a single depth per pixel, as in our experiments, $|\hat{s}_0 \cdot \omega| \neq 1$ so the cross product produces a non-zero vector. We can then write the plane equation as:

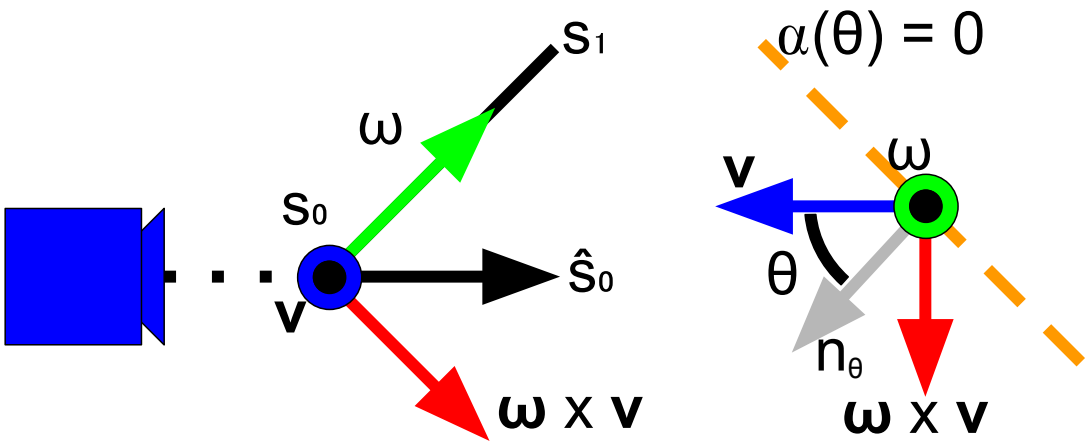


Figure 4-3: We generate a plane parameterized by θ for each chosen edge in the scene. Using a per pixel depth map, an equation of the plane along the edge can be reliably generated for almost any visible edge in the scene.

$$\alpha(p, \theta) = \mathbf{n}_\theta \cdot p - \mathbf{n}_\theta \cdot s_0 \tag{4.4}$$

Separately, we notice that the occluding edge, (s_0, s_1), and the vector pointing towards the light source, $(l_k - s_0)$, lie on a plane with the following normal vector:

$$\mathbf{n}_l = \frac{(l_k - s_0) \times (s_1 - s_0)}{||(l_k - s_0) \times (s_1 - s_0)||} \tag{4.5}$$

Furthermore, we can solve for θ given l_k, observing that $\mathbf{n}_\theta = \mathbf{n}_l$ when l_k is on the plane defined by θ:

$$\theta_l = \tan^{-1}\left(\frac{\mathbf{n}_l \cdot (\omega \times \mathbf{v})}{\mathbf{n}_l \cdot \mathbf{v}}\right) \tag{4.6}$$

While a single edge allows for us to constrain the position of the hidden point to a plane, we can combine multiple edges to uniquely define a point in 3D.

Multiple Edges: Scene Coordinate System

At least 3 edges are required to represent a point in $\mathbb{R}^3$. Our scene can be represented as a set of edge coordinates, $k = [\theta_0, \theta_1, \theta_2] \in \Theta^3$, where $\{\Theta \in [-\pi, \pi]\}$. We fix 3 edges in the scene to represent our coordinate system, and find the values of the remaining

coordinates using Equation 4.6.

Our point location can be determined by finding the intersection of planes parameterized by k., but we may only have access to one or two edges in the scene. For a single edge: $k \in \Theta$, and we can only determine the plane the point lies on. For two edges or all parallel edges: $k \in \Theta^2$, and we can only find a line that the point lies. For at least 3 non-parallel edges: $k \in \Theta^3$, and the point can be recovered in 3D.

From Coordinate to Shadows

Now that we have a suitable coordinate system, we refer back to Figure 4-2, and claim that the planes parameterized by k are also possible positions of shadows in the visible scene. The modeling of shadows from scene edges is well known, typically referred to as "shadow volumes." With our new parameterization, however, we can simply find the values of k that align the planes along the shadows and the corresponding intersection of planes localizes the hidden illumination point. We do this by selecting a point along each shadow and applying Equations 4.5 and 4.6, replacing l_k with any point along the shadow, recovering the values of k.

We point out that Equation 4.4 now has a useful interpretation, a scene point, p, lies on a shadow edge when $\mathbf{n}_\theta \cdot p = \mathbf{n}_\theta \cdot s_0$ since a point on the plane would make $\alpha(p, \theta) = 0$.

4.2.2 Solving for light source location

In general, we can localize a light source by solving a system of linear equations. Finding the intersection of planes is useful in other computer vision tasks, and are often solved in homogenous coordinates [39]. However, we can adapt the coordinate system we've described so far to find the intersection of multiple planes defined by Equation 4.4. Given a set of corresponding points s, and normals n, the intersection can be solved:

$$S = \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_m \end{bmatrix} \tag{4.7}$$

$$N = \begin{bmatrix} n_0 \\ n_1 \\ \vdots \\ n_m \end{bmatrix} \tag{4.8}$$

$$p = (N^\top N)^{-1} N^\top \sum_{k=3}^{3} N_{mk} \cdot S_{mk} \tag{4.9}$$

Here, we solve for the intersection of m shadow planes. Note that in order for this equation to have a solution $(N^\top N)$ must not be singular. Therefore, we need at least 3 planes to localize a point. Furthermore, we find that the intersection of 2 planes yields a line. The constraints on the recovery of the location of the light source using just shadow edge information is thus encoded in the ability to invert N. Naturally, more edges should provide a more accurate result. Equation 4.9 provides the least squares solution to the point source location.

Planar Homologies

While not explored extensively in this work, the relationship between planar objects, or their planar silhouette, that cast shadows on planar surfaces can be described by a homology in image space. There is an interesting example of planar homologies for image editing in [106]. In particular, a planar homology is formed when a planar occluder casts a shadow onto another plane when the scene is illuminated by a point source. The shadow cast by the surface mesh of a triangulated 3D object can be determined by considering the set of triangles making up the mesh as a collection of planar occluders.

The geometry of planar homologies provide constraints in image space, and are

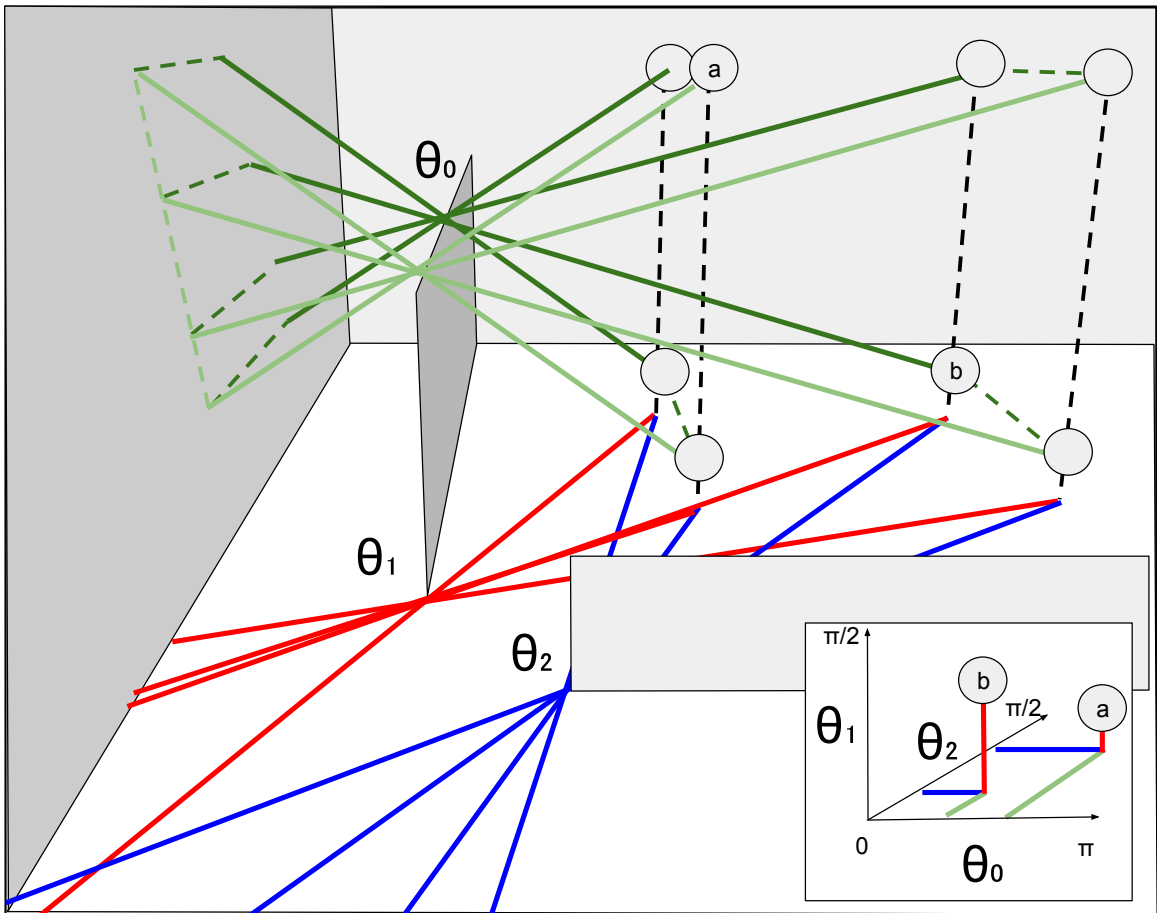


Figure 4-4: 3D points can be described by the intersection of 3 planes formed at 3 edges. There are numerous possible combinations of illumination points, but correspondence between shadows across edges can enable 3D reconstruction of the hidden illumination sources.

useful for determining expected visibility between points in image space and known planar occluders. While the computed visibility terms are useful in many applications, they are not sufficient for localizing point light sources in 3D as they are inherently limited to perspective geometry.

4.2.3 Adding Hidden Points

Additional points to the hidden scene will create more shadows cast by the occluding edges. Figure 4-4 illustrates such a scenario. Reconstructing points by finding all the intersections of these planes will not work because with only 3 edges, there can be M^3 intersections for M shadows per edge. This means that 3D reconstruction is severely

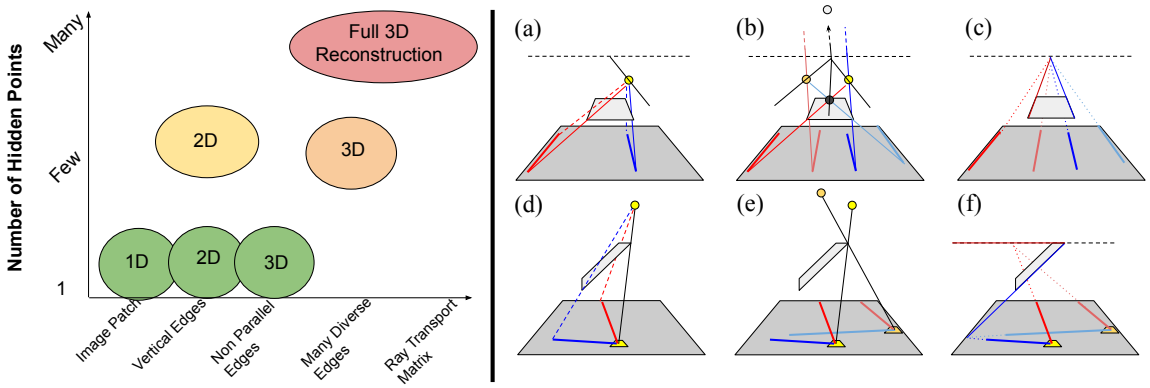


Figure 4-5: *Left*: Image features limit the recoverability of the illumination point position. We show that there's a range of possible features that can be used for recovery of incident illumination, and perfect knowledge of the full light transport may not be required. *Right*: The geometry we describe introduces a few constraints and reveals fundamental ambiguities (a,b). We can use this geometry to associate edges with shadows in image space (c,f), and 3D is a depth map is available (d,e).

ill-posed in some cases. However, combining additional edges can begin to restrict the possible point positions.

4.2.4 Relation between Point Sources and Edges

From Equation 4.9 we can make some important geometric observations. Figure 4-5 illustrates these scenarios as observed by a perspective camera. Figure 4-5(a-c) show two parallel edges that are parallel to each other and also parallel to a plane. Figure 4-5(d-f) illustrates a situation for shadows cast by two non-parallel edges that meet at a corner, one edge that is parallel to a plane, and another edge that is at an angle to the same plane.

2 edges produce set of rays, 3 edges can produce a set of points In Figure 4-5(a,d) we see that two edges can restrict the location of the point to at most a line (1D), as the intersection of two planes is a line. Combining any number of planes that lie along parallel edges also can only intersect along a line. Parallel planes will intersect along a line at infinity.

A natural extension of the first point is to combine three edges with at least one edge not parallel to the others. The intersection of these planes is a point in 3D, as

the intersection of 3 non-parallel planes is a point.

Lack of shadow correspondence can produce ambiguities As show in Figure 4-5(b), by simply observing shadow edges, it is not possible to determine which point cast each shadow. This is fundamentally a correspondence problem, as different combinations of associated shadows can produce phantom points (shades of grey in Figure 4-5(b)). Corner edges, such as in Figure 4-5(e), make this problem better posed, but ambiguities can still exist.

A single patch can not recover depth As show in Figure 4-5(d,e), a single local image patch (show in two shades of yellow) can at best restrict the location of the point source to a ray. Thus, identifying the corner-shaped shadow in the image, and drawing a line from the shadow of the corner to the corner itself forms a ray pointing towards the illumination source. Since all of the rays must pass through the occluding corner point, the distance to the illumination point can not be determined via parallax. This is analogous to the depth ambiguity inherent in pinhole cameras.

Lines along shadow edges and occluding edges intersect in image space We see that in Figure 4-5(c,f), if we draw lines along the occluding edges and extend the lines along cast shadows, they intersect at a point in image space. Occluding edges that are parallel to a plane will intersect at a vanishing point, indicating the lines are parallel in euclidean space. Occluding edges that are not parallel to the plane, such as the edge marked in blue in Figure 4-5(f), intersect shadow edges at the intersection of the extended edge line with the plane. Using this observation, we can associate edges with cast shadows, by ensuring that this intersection occurs. This can also be used to identify which scene edges are casting shadows, by removing all those edges that do not form an intersection point with at least one shadow edge.

Implementation Details

The data we use in our validation consist of images generated using the Blender Cycles path tracer. We use the camera matrix provided by Blender, along with the z-buffer

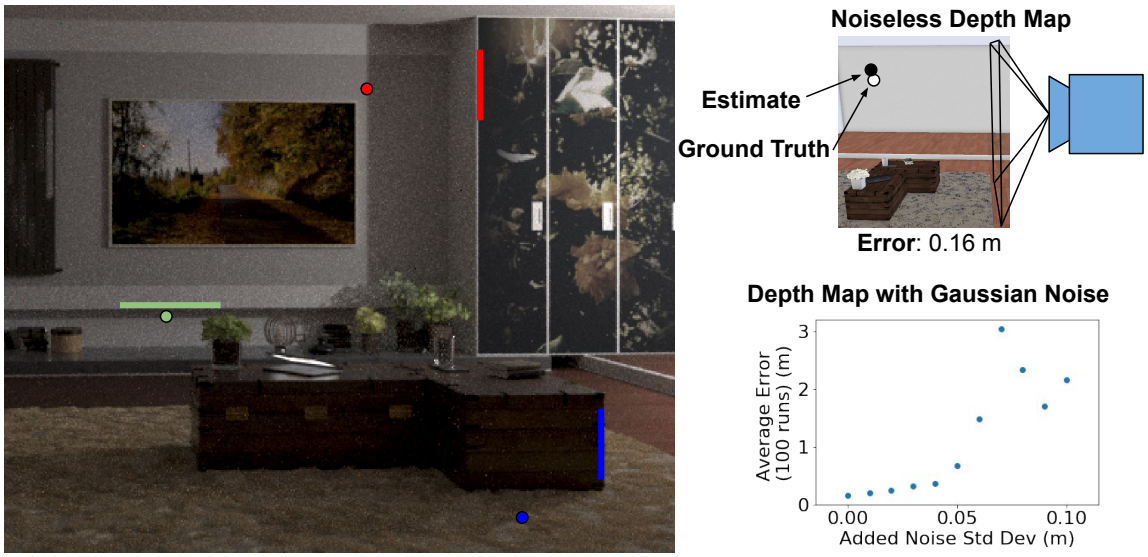


Figure 4-6: Given edge and shadow positions in a realistic synthetic scene, we can localize an illumination source with a high degree of accuracy. Our approach utilizes a depth map to estimate the 3D location of scene points, adding noise to this depth map smoothly degrades localization accuracy, and does not require that the depth maps are noiseless.

to generate a point cloud. This point cloud provides the 3D location, p, of each pixel in the camera's reference frame. We use this point cloud to define edges using (s_0, s_1), and to check pixel points if they lie on a plane for a particular edge-plane angle, θ. We use the world-view matrix to convert estimated points in the camera frame back to world coordinates in Blender to determine error from ground truth.

4.3 Validation

4.3.1 Single Point Localization

In order to demonstrate point localization using our proposed geometric technique, we rendered a photo-realistic scene containing a variety of different materials, textures, and geometry using Blender's Cycles path tracer. An illumination source was placed outside the field of view of the camera, such that visible objects in the scene cast shadows onto other visible objects. After manually labeling three edges and corresponding shadow positions, the point source could be localized within 16cm of the true position.

The z-buffer from Blender was used as a depth map to determine the 3D position of points in the scene. Depth maps captured using RGBD cameras will invariably contain noise, so we also evaluated our method's robustness when using a depth map with additive white gaussian noise. With 1 cm standarrd deviation error applied, the method performed almost the same as in the noiseless case. Adding more noise reduced accuracy gradually, and with 10cm noise localization was much less reliable.

4.3.2 Multi-point Localization

In Figure 5-1, we demonstrate localization errors of 0.48m, 0.65m, and 0.15m for red, blue, and green estimates respectively for a natural scene with multiple point sources. We point out that the majority of the localization error is along the radial distance to the visible scene. This is understandable, as the scene in Figure 5-1 contains less parallax between edge geometry than Figure 4-6. We point out that the observation intersection rule in Figure 4-5(c,f) can be used to verify our edge and shadow associations. For example, the edge at Figure 5-1(b) and the green edge would intersect each other near a vanishing point in image space, while the other edges and shadows obviously intersect in image space. Shadow edges from point sources are easy to identify, and our scene contains a variety of edges to use for localization. However, without a known correspondence between shadows from different edges, there many different possible positions for the hidden points. In practice it may be beneficial to use additional edges to further localize points. To this effect, beyond demonstrating the ability of our method to sufficiently localize hidden points, we conducted an experiment to show 2D reconstruction of multiple points to resolve this ambiguity.

4.3.3 Localizing shadow edges using image gradients

As described in Figure 4-8, combining two edges is not sufficient for 2D reconstruction because there are inherent ambiguities that arise from a lack of known correspondence. In order to investigate this further, we implemented a 2D reconstruction of a hidden

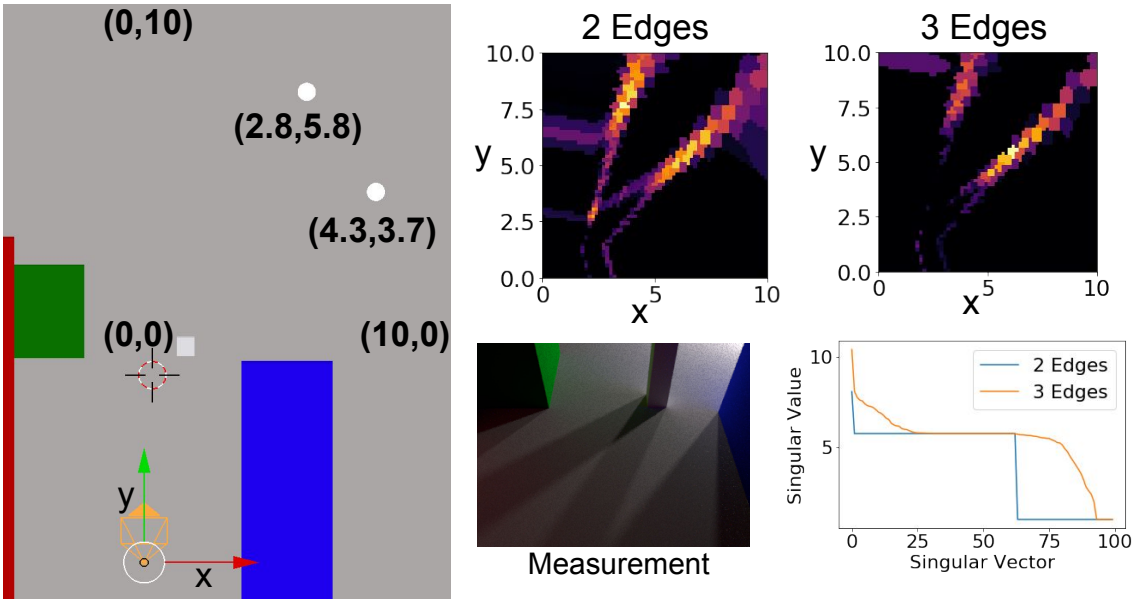


Figure 4-7: One approach to localizing multiple points without a known correspondence is to use additional edges to constrain possible illumination source positions. Using the 1D projections of the scene from each edge, we estimate the location of two light sources in 2D. Observing the singular values of the first 100 singular vectors reveals additional diversity in the visibility matrix when adding a third edge. This seems to resolve ambiguities during estimation, such as the false apparent source at (2.5, 2.5).

scene shown in Figure B-1. First, we annotated the position of occluding edges in the scene as in previous examples, however, instead of specifying shadow locations, we calculated the gradient between slices of pixels we found using Equation 4.4, as in [17]. Taking the gradient in this way produces a 1D projection of the scene, and is easily accomplished using the geometry we propose in the chapter.

For each edge and 32 evenly spaced values of θ, we use n_θ projected in screen space to determine the direction to take the spatial gradient of the image. We point out that this enables direct recovery of the 1D projection without first taking a homography. We then add the value of this intensity gradient to an accumulator, that sums each local intensity gradient with all all other pixels that correspond to θ. As can be seen in the scene in Figure B-1, "thin" column occluders will produce a different shadow pattern than a basic wall edge. We found that simply setting any negative gradients to zero produced reasonable incident illumination recovery.

Given the 1D projection from each edge, b, we attempt to estimate the 2D position

of all hidden sources, r. We do this by solving the linear system $Ar = b$, where r is a flattened 32×32 grid corresponding to each combination of angles from 2 edges (as in Figure 4-8). When adding the third edge, we use the same discrete coordinates from the first two edges, but the 1D projection is shifted due to parallax. The matrix A simply maps each r to the expected 1D projection. We find the solution to this system using a non-negative least squares solver with ridge regression.

As expected, we found that adding an additional edge was able to successfully resolve the ambiguity seen when reconstructing with only 2 edges. The reconstruction of two circular objects is clearly seen, although they appear stretched in the direction moving away from the visible scene, which would be expected from due to the limited parallax available in the visible scene. The first 100 singular values are plotted for the A matrix, for the 2 edge and 3 edge case, further demonstrating the additional information contained in the third edge.

4.4 Discussion

Detecting known failure cases Our proposed technique presents an opportunity to determine when localization or reconstruction is not possible due to limitations of the scene geometry. For example, if we are able to evaluate the scene and determine if scene edges have a diverse orientation, we can make an informed decision about whether a reconstruction of the hidden scene may be possible.

Pre-processing and Annotation For the purposes of this section, we provide manual edge and patch annotation for our scenes. We consider these important sub-problems which are largely perception problems separate from our core contribution. While in this work we consider manual annotations as the "input" to our method, other emerging techniques such as data driven methods may be well suited to automate these tasks in future work.

Non-Emissive Hidden Objects In order to recover a signal from the hidden scene, we can subtract an image of the observed scene before an object was introduced around

the corner. Subtracting this background frame reveals what pixels in the observed scene change. This image can be understood as the "differential image" [30] of the observed scene. In essence, any light paths that interact with the hidden object are modified, and additional light can make it into the visible scene, making the hidden object appear like an illumination source. In practice, the majority of the signal introduced back into the observed scene is additive: the object scatters light from an active light source back into the observed scene. There are some cases where the object could block light paths originating from illumination sources in the hidden scene. In this case the differential image will be negative as the object casts shadows from these illumination sources in the hidden scene.

4.5 Future Work

4.5.1 Coordinate system for discretizing the hidden scene

While the primary focus of this chapter is to describe the shadow-edge geometry in the context of illumination point localization, shadow-edge information may be useful to determine how to discretize the hidden volume when performing full scene reconstruction. This is because shadow edges are an essential cue for the presence and location of illumination in the hidden scene. Discretization of the hidden scene volume is an important part of solving the inverse problem in practice.

Example Discretization

We could discretize the hidden volume in a coordinate system defined by two edges in the scene, as shown in Figure 4-8. In this way, we define a 2D coordinate system $k : (\theta_0, \theta_1)$, where each (θ_0, θ_1) pair corresponds to a line in 3D euclidean space. We define a discrete set of N angles for each of these edges, producing a hidden region of $N \times N$ lines, determined by the intersection of the two planes associated with (θ_0, θ_1).

An area to further explore would be to derive the grid spacing directly from $(J_{M_k}^\top J_{M_k})^{-1}$, which is the inverse Hessian of the coordinate map. The corresponding

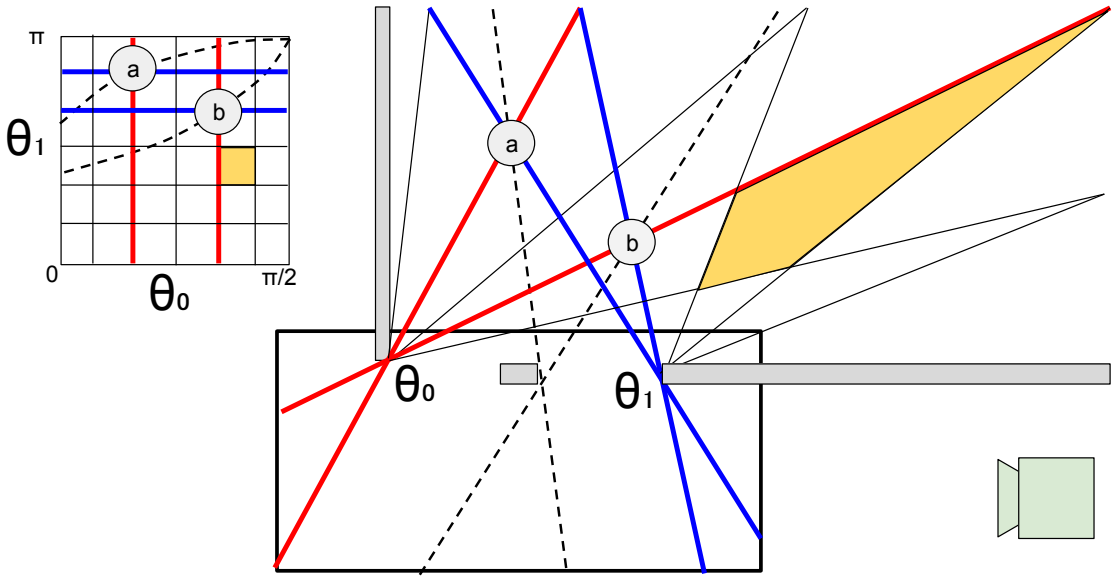


Figure 4-8: A camera observes a field of view shown by solid black box. Defining a coordinate system using the θ associated with each edge, we have a natural way to represent hidden scene coordinates. The yellow square in the coordinate grid is warped in euclidean space, the area can be determined by calculating the magnitude of the determinant of the coordinate system Jacobian.

eigenvectors and eigenvalues would inform the direction and relative spacing between grid points. When the magnitude of the Jacobian determinant is large, it implies that small changes in the shadow angles produce large changes in the location of illumination sources in the hidden scene. As such, in Figure 4-8 we would expect the grid spacing to become larger for points further away from the two occluding edges.

4.5.2 Differentiable Forward Model

So far, we've made use of the geometry of occluding edges to determine shadow edge locations. In this section, we describe how we can think of the shadow-edge planes as surfaces of a shadow volume. If we smooth the shadow edge discontinuity by making the step function in the visibility term a steep sigmoid, we can achieve a simple differentiable renderer where point locations are differentiable with respect to rendered image pixels.

We then demonstrate a single hidden source localization task and solve it using

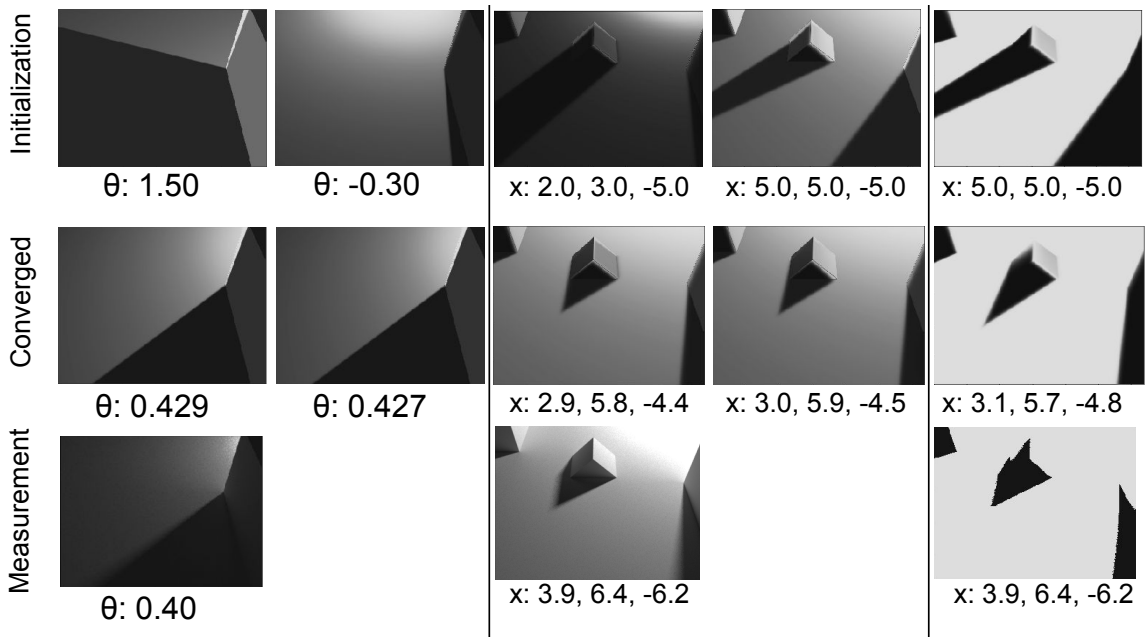


Figure 4-9: With an initial guess of a single light source location, we estimate the source position using a differentiable forward model that naturally arises from our edge-plane parameterization of the hidden point source. By minimizing the mean square reconstruction error using gradient descent, our model converges for different initializations. The last column shows a shadow only model, indicating that the model is not simply using shading and ignoring the shadows to localize the hidden point.

gradient descent. We use gradient descent to find θ that minimizes the mean squared error over M pixels.

$$\arg\min_{\theta} \sum_{i}^{M} ||I_0(p_i) - I(p_i, \theta)||_2^2 \tag{4.10}$$

The full model was implemented using JAX [34], using no momentum and learning rate of 0.01. Equation 4.10 enables optimization of the point light source position, θ, in order to match the observed shadow. Our results for some scenes can be seen in Figure 4-9.

The shading model

We want to estimate an unknown point source location, l_k, from an observed image I_0. We show how to find θ that minimizes the mean squared error over M pixels.

We use the point cloud to estimate surface normals at each point, and use an

initial estimated position of the light source to shade each pixel using a simple diffuse shading model.

$$I(p) = a_P \frac{\frac{l_k - p}{||l_k - p||} \cdot \mathbf{n_p}}{||l_k - p||^2} + g * (l_g \cdot \mathbf{n_p}) \tag{4.11}$$

Where $\mathbf{n_p}$ is the surface normal and a_p is the albedo at point p. To emulate some "ambient" illumination, we add term using the surface normal and a fixed global direction l_g and scaling factor g. In practice, this ambient term had very little effect on the optimization since it is constant, but we kept it as it was useful for visualization. For each iteration of the optimization, we normalized the target and model output such that they were scaled with pixel values $[0, 1]$.

The visibility term, a_P, can be defined using our edge-plane parameterization developed so far:

$$a(p, \theta_0, ..., \theta_N) = \prod_{C}^{i} \left(1 - \prod_{N_i}^{j} \sigma(\alpha_{i,j}(p, \theta_{i,j})) \right) \tag{4.12}$$

Where σ is a sigmoid function $\sigma(x) = (1 + e^{Tx})^{-1}$, $T \approx 100$. The outer product multiplies C convex shadow volume indicator functions. The inner product is composed of all the planes that make up the shadow volume and its purpose is to multiply to 1 only if the point is on the negative side of each plane equation. In this way, we can check if a point is in shadow (returns 0) or not (returns 1). We note that this visibility term in Equation 4.12 can be differentiated with respect to θ.

We shade every image pixel by constructing the plane equation defined by the point source location and a visible scene edge. This plane equation serves as an implicit equation for the shadow volume produced by the occluding edge. This plane equation can be used to weight visible scene pixels by passing it through a sigmoid function, such that any negative values (inside the shadow volume) are scaled by 0, and points with postive values (outside the shadow volume) are scaled by 1.

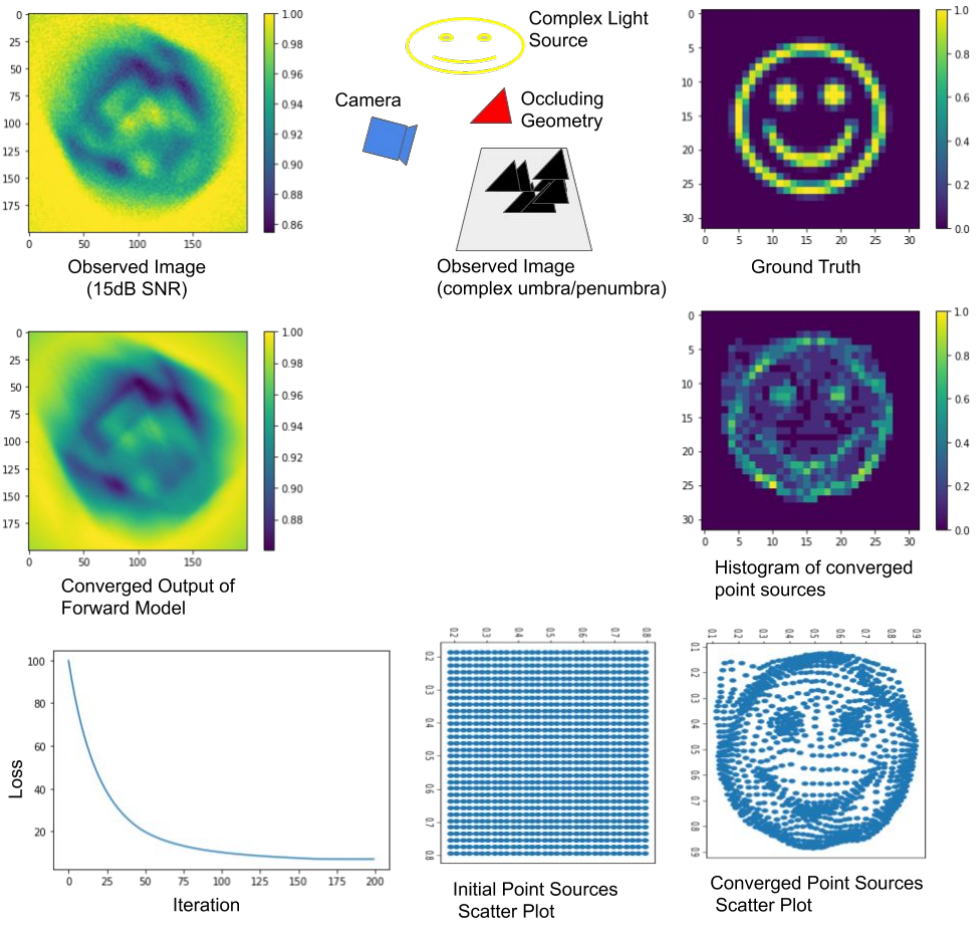


Figure 4-10: Example output of solving for each position of a set of point lights. Each light was initialized on a uniform grid above the scene and the position was optimized in order to minimize the mean squared error between the measurement and output of the differentiable forward model.

Optimizing over multiple points

Since we can optimize for a single light source position using auto-differentiation, it is relatively straightforward to optimize over multiple point source positions simultaneously. The observed image under multiple illumination sources is the sum of all images produced with each individual light on at one time. By adding a sum to Equation 4.11 over multiple lights, we optimize for the position of a collection of light sources initialized as a grid of point sources. In the basic scene in Figure 4-10, a single triangle occluder casts a shadow on a surface observed by a camera.

After a few iterations of gradient descent, the grid of light sources converges toward

the shape of the ground truth complex "smiley" light source. Effectively, the point sources converge to regions of high light flux and remain relatively static in regions of low light flux.

Handling arbitrary meshes

We extended the simple differentiable renderer to handle multiple triangles for complex meshes. This approach is easily accomplished by using Equation 4.12, since each triangle forms a convex shadow volume. While we were able to optimize the renderer for a more complex scene of a few triangles on a mesh, the product over triangles was extremely computationally intensive using the JAX autodifferentiation framework. We also implemented this renderer using the Enoki autodifferentiation library, which significantly improved the computation time, but this approach is fundamentally less efficient than emerging path tracing auto-differentiation frameworks such as Mitsuba2. The reason for this is that calculation of scene points in shadow volumes requires checking the shadow volume of every triangle in the scene that could possibly cast a shadow, which scales proportionally to the product of triangle counts and light sources.

4.6 Conclusion

In this chapter, we propose a method for localizing illumination sources from images. We identified a geometrical constraint that is commonly found in many real-world scenes and demonstrate how these constraints can be used to form a forward model to describe light transport in a scene that contains occlusion. We show that our method can be applied in scenes with arbitrary edge orientation.

We show how our method can be used to localize a single illumination point, and multiple illumination points with known correspondences. We further show the practicality of our approach by implementing two tasks using our framework. First, we demonstrate multiple source localization and the importance of using as many edges in the scene as possible. Second we show how an extension to our point localization

geometry can be used to create a differentiable forward model properly models cast shadows. This enables us to perform single source localization using gradient descent. We hope that in the future, extensions to our approach will foster more direct modeling of shadows in incident illumination estimation. In broader terms, we hope our work will serve as inspiration for combining geometrical understanding of light transport in scenes with perception based tasks.

Chapter 5

Reconstructing the Hidden Scene from Object Shadows

This chapter explores the idea that incident illumination estimation is a method for exploiting hidden cues that can enable the reconstruction of the scene from the viewpoint of an object in the scene.

In the previous chapters, we describe how object localization and point-light localization are highly related. We also show how many point-lights can be used to model extended sources. The common observation used by all point-light based approaches is that inference about the hidden scene is closely related to the problem of incident illumination estimation. As such, we bring our focus to incident illumination estimation directly.

This chapter will build on the principles of coded aperture imaging, where shadows cast from a known mask onto an image sensor are used to reconstruct images of a scene. A key idea from coded aperture imaging is that we can describe a linear image formation model, that can be inverted using standard techniques. We extend this work to arbitrary objects with 3D shape instead of flat masks.

Consider a small object sitting on a desk in your living room. The object is illuminated by light sources from all directions—this includes direct sources such as the sun or overhead lights, but also indirect sources, like the foliage outside that scatters sunlight through your window. The appearance of the object and the surface that it

rests upon results from the complex interaction between the incident illumination and the geometry and material properties of the object and the desk. In this chapter we ask the question—if the geometry and material properties of the observed scene are known, how well can we reconstruct the incident illumination pattern?

If we assume that the illumination sources are distant relative to the size of the observed object, then we can represent this illumination as a hemispherical photograph taken from the perspective of the object. Thus, by using our knowledge about an object to accurately estimate the illumination incident upon it, we effectively turn the object into a camera.

We primarily make use of shadows cast by an object onto nearby surfaces. Cast shadows are particularly easy to interpret when an object is illuminated from a single direction. For example, one can immediately determine the position of the sun by looking at a sundial. Estimating the illumination incident from all directions simultaneously is more challenging, and is a linear but ill-posed inverse problem.

Prior methods are commonly limited to controlled capture methodologies. Some rely upon object's shading or specular highlights and thus require the bidirectional reflectance distribution function (BRDF) or precise surface normal information. Our method relies on cast shadows and requires only the shape of the object and the shape and albedo of the shadowed surface. Previous works that utilize cast shadows require planar objects that cast shadows onto planar surfaces. We aim to support complex 3D objects in a general framework applicable to most natural objects and scenes.

We require partial reconstruction of the visible scene surface, such as from a stereo camera pair, or utilize known object geometry and relative camera orientation. Our estimates are performed by jointly optimizing for scene parameters using this partial set of views to approximate the ray transport matrix. Once we have estimated the ray transport matrix, further estimates only require a single photograph of the scene.

5.0.1 Contributions

- Recover high-frequency environment map from object shadows that does not require a special setup, specular object, or low-frequency assumptions

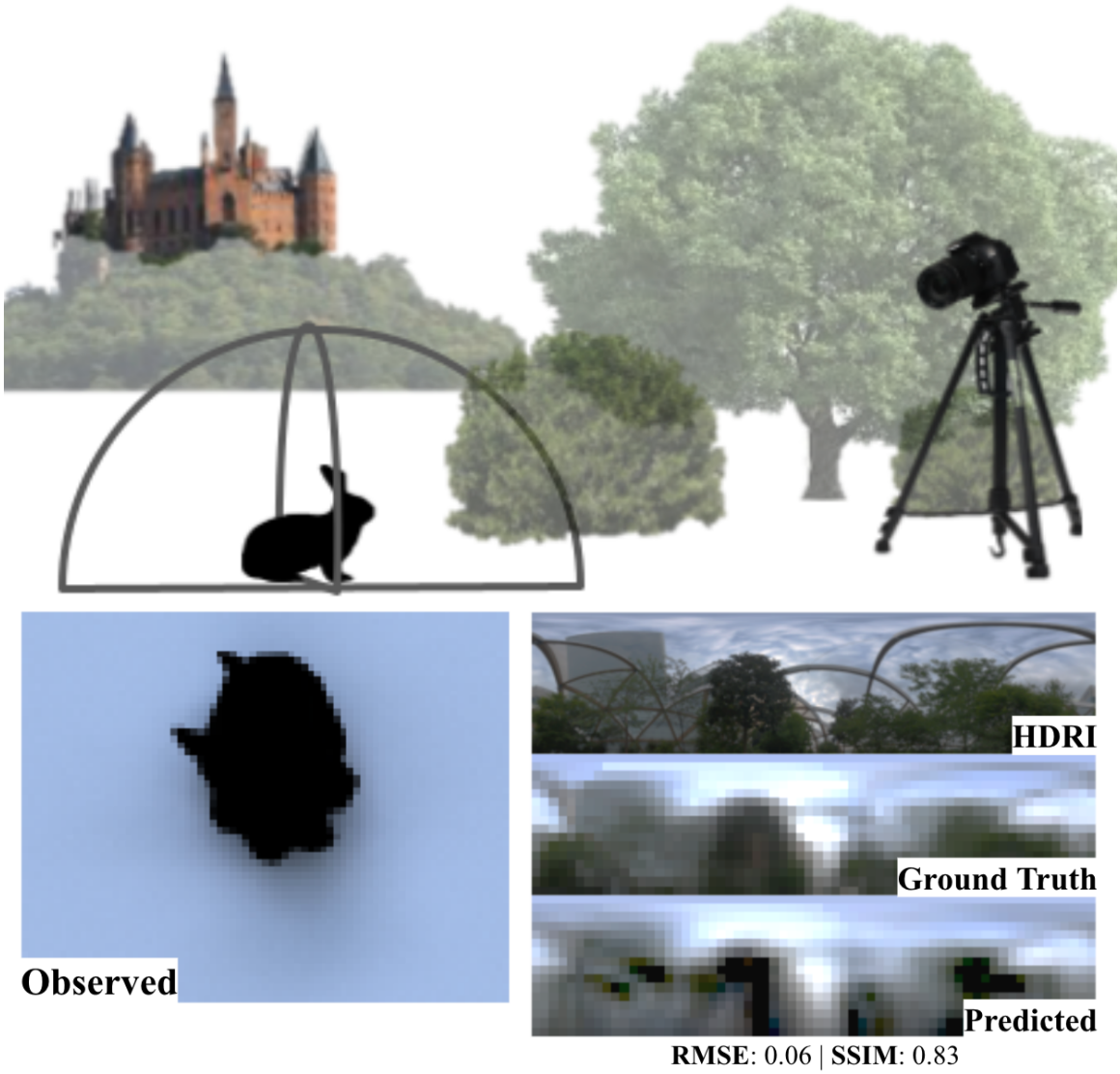


Figure 5-1: (Top) Observing the area around a small, bunny-shaped object (top-left), can we recover occluded viewpoints only visible from the bunny's perspective? (Bottom) Given the surface geometry of an object (bottom-left), we estimate the incident illumination, and in some cases, the unknown diffuse albedo of the surface surrounding the object.

- Propose a practical technique for approximating the light transport of an arbitrary object in a natural setting from a restricted set of partial viewpoints

- Demonstrate a strategy for solving the inverse problem "in the wild" for a scene with unknown surface material and using just a single photograph.

- Examine the structure of the ray transport matrix to determine feasible regions of reconstruction and assess the object-camera performance

5.1 The Object Camera

A perspective camera separates the radiance of light rays that travel through a particular point in space. A pinhole camera's aperture separates rays such that each sensor position receives light from a unique direction. In our proposed setting, a chosen object forms the camera's "aperture" and the surrounding visible surface forms the sensor. Given an image of this object and the shadows that it casts, we hope to "see" what the object can see from its own perspective.

5.1.1 Inverse Rendering Problem

Inverse rendering is an analysis by synthesis approach, where we optimize the parameters of a forward light transport model until the rendered images closely resemble the measured image. In general, the inverse rendering problem is extremely ill-posed: there are some combinations of different scene parameters that render very similar looking images. As such, additional information is required in the form of priors or additional images to resolve ambiguities.

To illustrate the problem, consider a scene with a single object on a flat surface that has an arbitrary albedo and diffuse surface reflectance. We aim to reconstruct an image from the object's perspective, represented by the set of rays extending from the center of the object to a distant hemisphere. If we consider only direct illumination, the observed radiance from a scene point, x, can be written as follows:

$$L_o(x, \omega_o) = \rho(x) \int_\Omega V(x, \omega_i) L_i(\omega_i)(\omega_i \cdot \mathbf{n}) d\omega_i, \tag{5.1}$$

where L_o is the radiance observed by the camera at surface position x with surface normal $\mathbf{n}$, ρ is the spatially varying diffuse albedo, V is the visibility of the surrounding hemisphere, parameterized by ω_i, as seen from surface point x, and L_i is the radiance of every incident ray from the surrounding hemisphere. We note that L_i forms an image of the incident illumination from the perspective of the object. We assume that the illuminating hemisphere is sufficiently far away that we only need to know ω_i. This illumination model is often referred to as an environment map.

If ρ and V are known, solving for L_i is a linear inverse problem. Concretely, if we discretize these spatially varying functions, with incident illumination vector $\mathbf{x}$ corresponding to illumination directions sampled over the environment map, and $\mathbf{y}$ the pixels of the corresponding image, we can write the above equation in matrix form, where "$\odot$" is the Hadamard product:

$$\mathbf{y} = \text{diag}(\rho)(\mathbf{V} \odot \mathbf{C})\mathbf{x} = \mathbf{A}\mathbf{x}, \tag{5.2}$$

Where ρ is a $M \times 1$ vector of diffuse albedos, $\mathbf{V}$ is a $M \times N$ matrix, $\mathbf{C}$ is a $M \times N$ matrix of the cosine factors, and $\mathbf{x}$ is a $N \times 1$ vector of unknown illumination radiance. Combining factors, we obtain a linear system of equations represented by the matrix $\mathbf{A}$, where $\mathbf{A} = \text{diag}(\rho)(\mathbf{V} \odot \mathbf{C})$. We can therefore solve for the environment map, $\mathbf{x}$, by minimizing the mean squared error: $||\mathbf{A}\mathbf{x} - \mathbf{y}||_2^2$.

5.1.2 Solving for Incident Illumination

Overview of Approach Rather than decompose the matrix explicitly for every scene as in Eq. 5.2, we make use of a modern rendering framework to represent the more complex structure of the model matrix, $\mathbf{A}$. Since we model the incident illumination using an environment map, we can generate the rows of $\mathbf{A}$ by rendering an image for each illumination source in the discretized environment map.

Our inverse rendering problem is composed of two steps:

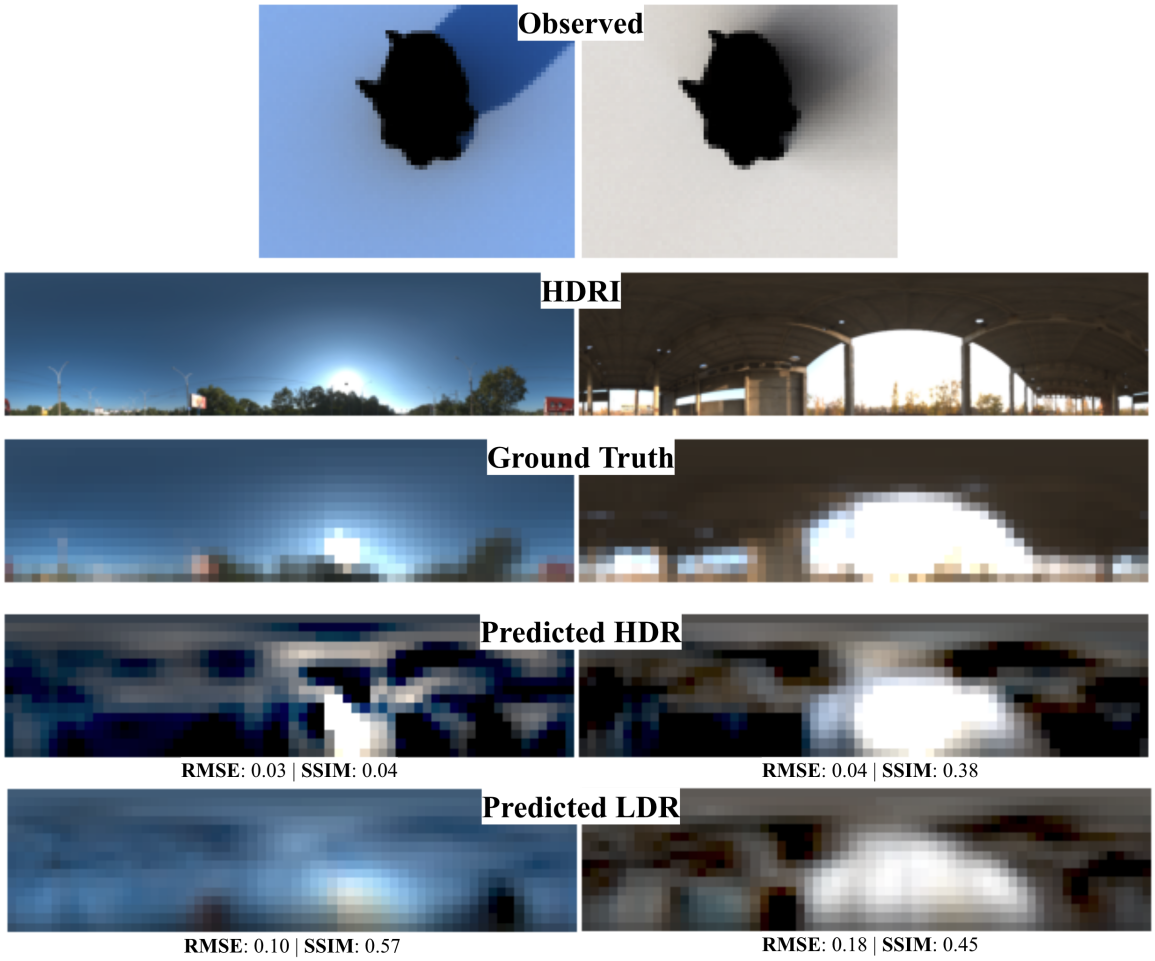


Figure 5-2: Given a known surface geometry and shadow surface albedo, we estimate the environment map illumination using a single observed image by solving Eq. 5.3. For easier viewing, basic tonemapping was applied to all images using gamma correction with clipping ($\gamma = 2.2$, image normalized such that the sun is clipped).

1. Render the approximate light transport matrix using estimated scene parameters
2. Estimate illumination or albedo by solving the corresponding linear system

The scene parameters we estimate at step 1 are the camera extrinsic parameters, scene surface mesh, and diffuse surface albedo. These parameters could be known *a priori*, or obtained using some other 3D reconstruction methodology such as photogrammetry.

Rendering the Ray Transport Matrix We utilize the Mitsuba 2 [85, 71] path tracer to render the ray transport matrix. Each column in the $\mathbf{A}$ matrix is associated to an individual pixel in our environment map, and consists of the image that is rendered when only that single environment map pixel is lit. The rendered images are the columns of $\mathbf{A}$: $\mathbf{A_i} = f_\theta(\mathbf{x_i})$ where $\mathbf{x_i}[i] = 1$ and $\mathbf{x_i}[j] = 0$ when $i \neq j$. $f_\theta(\mathbf{x})$ represents the Mitsuba 2 rendering pipeline given scene parameters θ. In our case, θ are the surface geometry mesh data and uv texture maps with corresponding diffuse albedo or other material parameters. We map $\mathbf{x}$ to the texture data used by the environment map emitter, where image pixels correspond to latitude and longitude in spherical coordinates.

Jointly Solving for Illumination and Albedo With known surface geometry and material parameters, we can solve for the incident illumination. We write the illumination recovery problem as the solution to a linear system, where the rows of the model matrix, $\mathbf{A}$, are comprised of rendered images for each component (e.g. ray) of the illumination model. Since $\mathbf{A}^\top\mathbf{A}$ is not necessarily well conditioned, we add spatial smoothness and L2 regularization, such that we aim to minimize the following objective:

$$\begin{aligned} \arg\min_{\mathbf{x}} \quad & ||\mathbf{Ax} - \mathbf{y}||_2^2 + \lambda_s||\mathbf{Dx}||_2^2 + \lambda_r||\mathbf{x}||_2^2 \\ \text{s.t.} \quad & \mathbf{x} >= 0, \end{aligned} \tag{5.3}$$

where $\mathbf{D}$ is the spatial difference operator and λ_s, λ_r are regularization parameters controlling spatial smoothness and L2 regularization respectively.

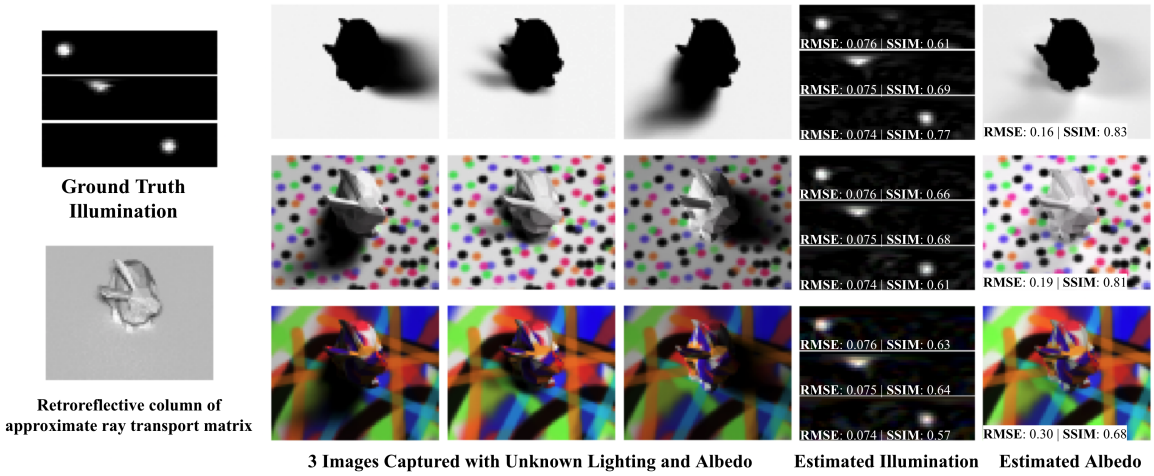


Figure 5-3: Given a ray transport matrix computed assuming all surface albedo are one, and 3 images with unknown surface albedo or illumination, we estimate both using the update procedure described in Equation 5.7. Three different texture maps are shown, the observed images are rendered with global illumination.

Typically, the estimated albedo of the shadow surface is unavailable or not accurate enough to obtain good results. We found that jointly estimating albedo and illumination was essential to make our model robust to poor initialization of mesh albedo, due to occasional artifacts in the surface texture produced by photogrammetry. Given a ray transfer matrix, $\mathbf{A}'$, rendered using Mitsuba with or without an existing albedo texture map, we augment our forward model with a per-pixel scaling factor as a proxy for diffuse albedo: $\mathbf{A} = \text{diag}(\rho)\mathbf{A}'$.

As pointed out in [101], we also observed that adding a pixel-wise scaling term to emulate the diffuse albedo residuals seemed to make the illumination estimates more robust. As such, given the ray transport matrix rendered using Mitsuba, $\mathbf{A}'$, we can solve for the per-pixel scaling term. Since the albedo term is fixed under changing illumination conditions, it is advantageous to write the following objective for L observed images, $\mathbf{y_l}$, and the associated environment map, $\mathbf{x_l}$.

Thus, we minimize the following objective:

$$\begin{aligned} \arg\min_{\rho} \quad & \sum_{l} ||\text{diag}(\mathbf{A}'\mathbf{x_l})\rho - \mathbf{y_l}||_2^2 + \lambda_p ||\mathbf{G}\rho||_2^2 \\ \text{s.t.} \quad & 0 < \rho \leq 1, \end{aligned} \tag{5.4}$$

where $\mathbf{G}$ is the linear inverse operator: $\mathbf{G} = (\mathbf{A'}^{\top}\mathbf{A'} + \lambda_r \mathbf{I})^{-1}\mathbf{A'}^{T}$. This is similar to the regularization term used in [101], and discourages albedo estimates that can be easily reconstructed using the ray transport matrix, such as cast shadows.

In practice, we solve for the albedo term, ρ, and each environment map, $\mathbf{x_l}$, separately. We solve Equations 5.3 and 5.4 using damped Newton's method, with update steps in the direction of $\mathbf{x}^*$ and ρ^*, defined below.

The unconstrained minimum solution of Equation 5.3, $\mathbf{x}^*$, satisfies the linear equation, with $\mathbf{R} = \lambda_s \mathbf{D}^{\top}\mathbf{D} + \lambda_r \mathbf{I}$:

$$(\mathbf{A'}^{\top}\mathrm{diag}(\rho)^2\mathbf{A'} + \mathbf{R})\mathbf{x}^* = (\mathrm{diag}(\rho)) \odot \mathbf{A'})^{\top}\mathbf{y} \tag{5.5}$$

Similarly for the ρ^* and Equation 5.4:

$$(\lambda_p \mathbf{G}^{\top}\mathbf{G} + \sum_l \mathrm{diag}(\mathbf{A'x_l})^2)\rho^* = \mathbf{y_l} \odot \sum_l \mathbf{A'x_l} \tag{5.6}$$

Equations 5.5 and 5.6 can be solved efficiently using a linear solver. With $\mathbf{x}^*$ and ρ^*, we alternately update ρ^{t+1} and $\mathbf{x}^{t+1}$ until convergence using a damped Newton step with damping factor γ, applying a projection H after each step:

$$\begin{aligned} \rho^{t+1} &= H_{0,1}(\rho^t + \gamma(\rho^* - \rho^t)) \\ \mathbf{x}^{t+1} &= H_{0,\infty}(\mathbf{x}^t + \gamma(\mathbf{x}^* - \mathbf{x}^t)) \end{aligned} \tag{5.7}$$

Where H is a simple clipping operator, with $\min \leq H_{\min,\max}(\cdot) \leq \max$.

Combining Multiple Images While this work evaluates data from a single camera pose, our framework can be applied to P camera poses each with L illumination conditions as described below.

Multi-Illumination For a static camera with L different illumination conditions, we solve for each $\mathbf{x_l}^*$ environment map separately. Updating the albedo scaling factor, ρ, only requires one solution once each environment map, $\mathbf{x_l}$, is computed.

Multi-Viewpoint If there are P viewpoints, we find $\mathbf{x}^*$ by solving a larger system of equations, by stacking the light transport matrix associated with each camera

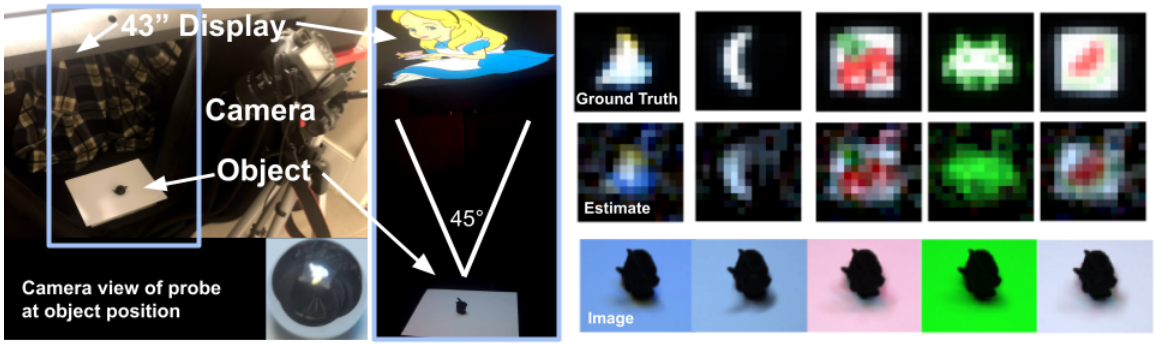


Figure 5-4: An object of known shape (a bunny) is placed in a controlled illumination environment. Illumination patterns are displayed on an LCD display placed above the object, and a camera observes the shadows that the object casts onto a flat surface. From this single image we produce an estimate of the illumination pattern seen by the object. A chrome ball is used to collect ground truth data in the object's position for each test pattern.

pose, $\mathbf{A_p}$, and observed images $\mathbf{y_p}$. We estimate a single environment map while incorporating information from all viewpoints. To find $\rho_\mathbf{p}$ for each camera, the set of $\mathbf{y_p}, \mathbf{A_p}$ are used to solve each $\rho_\mathbf{p}{}^*$ separately.

5.2 Implementation

Image Capture For all of our experiments, we capture a single "target image", $\mathbf{y}$, using a DSLR camera (Canon EOS Rebel T5). For alignment to the object, we use a photogrammetry pipeline that includes structure from motion, mesh generation, and surface albedo estimation. In some of our real experiments, we use the known geometry. For the ground truth environment map, we photograph a chrome ball with exposure bracketing for high-dynamic-range (HDR).

Rendering System For all experiments, we performed the rendering computations on a single Nvidia RTX 2080. Ray transport matrices took between 200-300 seconds (scene dependent) to render 2048 total images with environment map textures sized 32×64. Results in Fig. 5-3 converged in about 5 iterations (∼25 secs/iter) running on the CPU.

5.2.1 Synthetic Results

Realistic HDR Environments We rendered the ray transport matrix with known geometry and albedo for the black albedo bunny supported by a white planar surface. Each row of the ray transport matrix was rendered using 32 bit floats with 768 samples per pixel (spp). Observed images were rendered with 25.6k spp using freely available high dynamic range (HDR) probe images of real environments.[1]. We show the effect of renders with fewer spp and larger variance in the supplement, as well as an analysis of additive noise.

To solve Equation 5.5, each ray transport matrix requires approximately 500 MB of memory. We found that reconstructions took a few seconds on our machine using an Intel Xeon Silver 4210 CPU with 256GB RAM. As we see in Figure 5-2, we achieve reconstructions of the HDR environment maps with a reasonable recovery of the relative intensity of the bright sources.

Unknown Albedo and Illumination Conditions In practical scenarios, the surface albedo of an object may be unknown. We can overcome poor estimates of the surface albedo by jointly recovering the albedo and illumination. In Figure 3, we separate the albedo and illumination for three different textured versions of the bunny under a bright moving illumination source. Our initial ray transport matrix is generated using an all-white albedo for both the bunny and shadow surface. Upon initializing the albedo and environment map to all ones, we apply the update scheme in Equation 5.7 with $\gamma = 0.9$ for 20 iterations. We notice that the first update quickly resolves the albedo, but it often takes an additional iteration before the environment maps begin to converge. While both Equation 5.5 and Equation 5.6 can be solved in one step, we found it useful to reduce the learning rate such that the projection operator could enforce the constraints without collapsing the environment map solution to all zeros.

[1] https://hdrihaven.com

Failure Cases We found that HDR environment maps with extreme dynamic range can pose a challenge for reconstruction. In Figure 5-2, the bright sun has a radiance of $\sim 10000:1$ compared to the sky. Our approach attempts to estimate the HDR scene, but primarily resolves the brightest sources and ignores the detail in darker areas of the scene. When the environment map is clipped, producing a low-dynamic range (LDR) scene (only possible in simulation), our method is able to recover finer details. While we did not investigate further in this work, encouraging the recovery of darker regions is challenging because any HDR compression added to the output of the forward model would make it nonlinear. Similarly, since our method relies on tiny variations in the observed scene, our method is susceptible to errors in the HDR capture process.

The recovered albedo maps contain minor artifacts resembling the shadows in the observed images. We unsuccessfully applied a wide range of values for the regularization proposed in [101]. However, diverse and extended sources help reduce the appearance of these artifacts, but that may be due in part to the softer shadows cast by these objects. We were able to estimate convincing albedo using environment maps of real scenes, but the associated estimated environment maps often contain "retro-reflection" artifacts along the retro-reflective direction from the camera to object. These artifacts only appear after the first few iterations of projected gradient descent, and so reasonable results are achieved using early stopping. We show additional results in the supplement highlighting these failure modes.

Gradient Descent Baseline We compared our method in Table 5.1 to a stochastic gradient descent (SGD) baseline by minimizing reconstruction error using Mitsuba 2's auto-differentiation capabilities. We update both the scene albedo and environment map using Mitsuba 2's Adam optimizer, with learning rate 0.01. For the multi-illumination/unknown albedo scenes, the recovered albedo using Mitsuba 2+Adam was slightly better in terms of RMSE, but remained quite noisy, reducing SSIM. We believe Mitsuba 2's improved albedo estimation is due to the fact that our linear diffuse albedo model does not account for global illumination. As such, applying our

proposed method to quickly solve the linear approximation and then fine-tuning for non-linear effects using a differentiable renderer is a promising direction for future work.

Our approach has a number of advantages over gradient descent. In the simpler case where scene albedo is known, the loss function in Eq. 5.3 can be solved exactly in a single step using a linear solver ($\sim$4.5 secs in our CPU numpy implementation). Additionally, once the albedo has been estimated, the regularized inverse, $\mathbf{G}$, can be applied to new measurements to quickly estimate novel illumination without re-computing $\mathbf{A}$. Furthermore, our analysis in Section 5.3 is made possible by first rendering $\mathbf{A}$. We show in Table 5.1 that our method converges faster than Adam+Mitsuba 2. However, since our method effectively computes the Hessian, $\mathbf{A}^\top\mathbf{A}$, this advantage may not scale to high resolution environment maps. Regardless, faster low resolution estimates, like the proposed method, would remain useful for initialization.

5.2.2 Real-data Results

3D Printed Probes We 3D printed a 5cm tall "Utah Teapot" and low poly "Stanford Bunny" in black filament such that the 3D shape of the object is known, but the surface of the object would make it difficult to use any other cues for incident illumination estimation.

Each object was placed below an LCD panel (43" LG Desktop Monitor), on a white sheet of computer paper, approximately one meter away from the display. The shortest and longest monitor dimensions correspond to a 45 and 90 degree field of view from the perspective of the object respectively. The display was used to show different patterns, and an image of the object was captured with a known camera orientation, about one meter from the 3D printed object. Reconstructions are shown for each test pattern in Figure 5-4.

Figure 5-4 shows cropped and centered environment maps with each side corresponding to a 79 degree field of view. There are some apparent distortions in the reconstructions, such as a missing corner in Figure 5-4(e). We also produced reconstructions using much higher environment map resolutions (up to 256×512) and

Scene / Method	Envmap RMSE	Envmap SSIM	Albedo RMSE	Albedo SSIM	Compute Time (s)
Multi-illumination "dots"	0.417	0.43	0.286	0.78	**327**
+L2	**0.075**	0.62	0.219	**0.81**	**327**
+L2+Smooth (ours)	**0.075**	0.63	0.219	**0.81**	**327**
+L2+Smooth+[101]	**0.075**	0.63	0.218	**0.81**	338
Grad Descent Baseline (early stop)	0.531	0.41	0.202	0.69	**327**
Grad Descent	0.501	**0.68**	**0.172**	0.73	1434
Known albedo "garden"	0.492	0.24	-	-	**225**
+L2	0.078	0.77	-	-	**225**
+L2+Smooth (ours)	**0.063**	**0.83**	-	-	**225**
Grad Descent Baseline (early stop)	0.294	0.31	-	-	**225**
Grad Descent	0.120	0.57	-	-	975

Table 5.1: Ablation study: **Multi-illumination**: RMSE/SSIM relative to the known synthetic ground truth for the unknown albedo "dots" scene computed for 5 iterations for each set of added regularization terms. **Known albedo**: RMSE/SSIM computed for the HDRI "garden" scene for a single iteration. The compute times include pre-rendering of the ray transport matrix (220 secs) for the proposed method. Gradient Descent was stopped early to compare results under similar compute times, corresponding to $\sim$3461 iterations. The learning rate for Adam, $\gamma = 0.01$, for the gradient descent baseline was chosen to maximize SSIM after full convergence, which took significantly more time than the proposed method (+L2+Smooth).

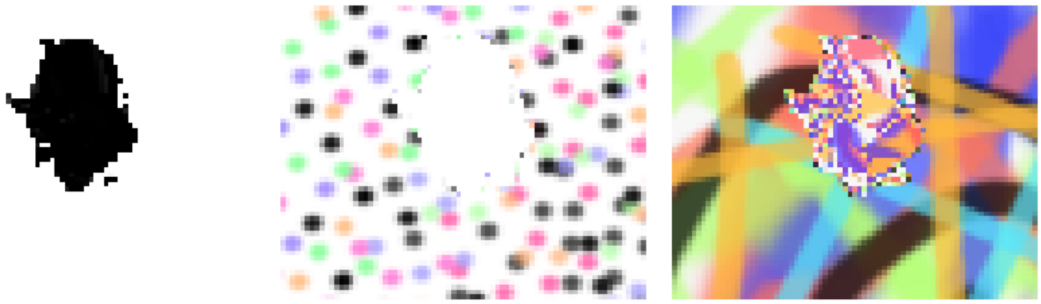


Figure 5-5: Ground truth albedo maps used to compute the RMSE/SSIM metrics in Fig 3 and Table 1 in the paper. Mitsuba2 ‘aov‘ integrator was used to output the per-pixel uv coordinates, which were then read from the texture. This also enabled measurements of the RMSE/SSIM of the recovered albedo texture obtained using Mitsuba2 autodifferentiation and gradient descent in image space for the gradient descent section of Table 1. Note that some artifacts from the Mitsuba2 ‘aov‘ integrator are apparent near the feet and ears of the bunny, potentially artificially raising the RMSE of the estimated albedos.

higher input image resolutions, but found they did not produce significantly improved reconstructions while taking longer to compute.

5.3 Analysis

5.3.1 Robustness to Sensor Noise

Estimates with Simulated Noise Regardless of the accuracy of a given ray transport matrix used for reconstruction, any real image will have some amount of noise from the sensor. This includes shot noise caused by the random arrival time of photons, which is a Poisson process, and noise from the analog electronics in the sensor from thermal effects or dark current, which can be approximated by a Gaussian process. As such, we start with a “No Noise” image obtained from a path tracer with sufficiently high samples per pixel. We then add Poisson noise based on the pixel intensities of the “No Noise” image and add Gaussian noise with some standard deviation to the shot noise image.

We show results in Figure 5-6, where we plot the structural similarity index measure (SSIM) of the ground truth environment map compared to the estimated environment

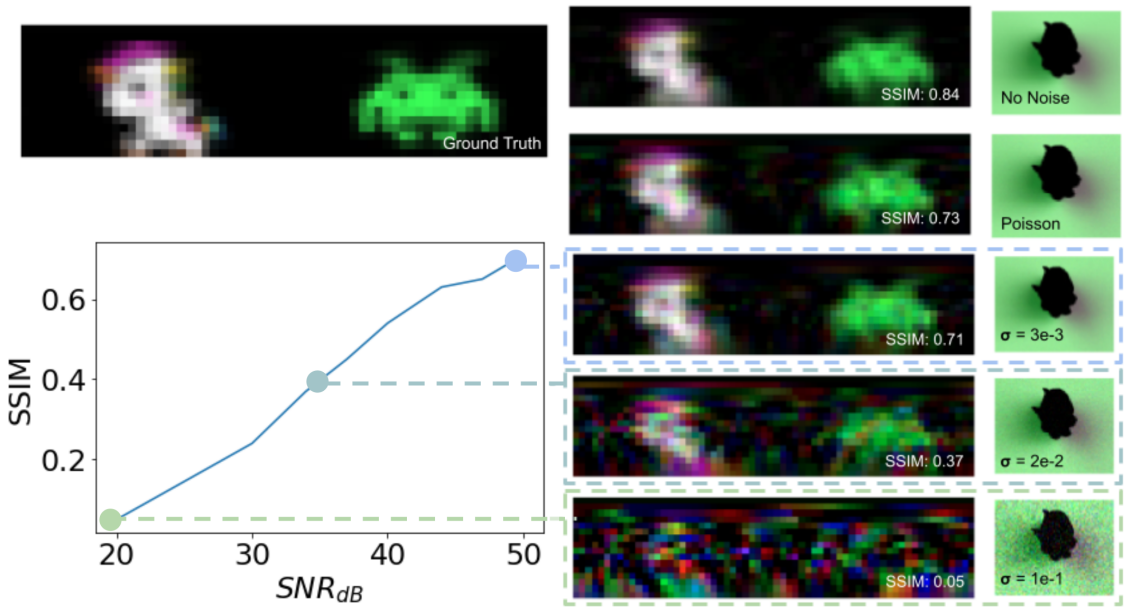


Figure 5-6: Top Left: Amber the unicorn and space invader. Estimated environment maps given noisy observations and their corresponding structure similarity index measure (SSIM). We synthetically added noise to a rendered image with pixel values between [0,1]. (Right column) Estimated environment maps with no noise, Poisson (i.e. shot noise) only, and Poisson + additive Gaussian noise with increasing standard deviation (σ) and zero mean. (Bottom left) The SSIM plotted against SNR ($20\log_{10}(\frac{1}{\sigma})$) in dB.

map given a noisey image. We see that reconstructions only begin to significantly degrade once sufficient Gaussian noise is added to the observed image. This provides some confidence that our method should work under realistic noise environments.

Variance from Path Tracer in Synthetic Results For all of our synthetic results, we render observed images using a path tracer. This is to ensure we are actually solving the inverse problem, as opposed to generating synthetic observed images using the ray transport matrix itself. If the observed images are rendered without enough samples per pixel, estimated environment maps are poor due to the variance of the path tracer's monte carlo estimate of the true image. This variance does not have an obvious physical analog, so we prefer to use sufficient samples to avoid problems. Figure 5-7 shows a range of values for the samples per pixel and the corresponding reconstruction results. We used these reconstruction results to determine how many

samples per pixel to use in our synthetic experiments.

Using a Higher Resolution Ray Transport Matrix In the main text we used environment maps with a resolution of 32×64, which provided a decent trade-off between computation time and resolution. However, there is no reason that our method could not be used with higher resolutions. Unfortunately, the use of higher resolutions comes at a significant computational cost, as doubling the resolution increases the ray transport matrix memory requirements and flops for a forward calculation by $4\times$. Inverting $A^\top A$ has a higher computational cost as the ray transport matrix increases in size. While out of the scope of the paper, the uncertainty maps in Figure 7 of the main text may provide a useful cue for irregular-sampling of the ray transport matrix such that higher resolutions can be preserved while reducing the total size of the matrix. Below we discuss further some of these issues and potential future directions when scaling this method.

Regularization Parameters For all experiments in the paper, we used the following values for the regularization parameters in Equation 6 and Equation 7 in the main text, λ_s: 600, λ_r: 100, λ_p: 1. We used a slightly different set of parameters for Figure 5-14 in the supplement, due to the challenge of model mismatch, λ_s: 1, λ_r: 200, λ_p: 1.

5.3.2 Additional Results

Teapot In addition to the results in Figure 4 in the main text, we show environment map estimates using a Utah Teapot in Figure 5-9. The quality of the reconstructions is not as good as the bunny results shown in the paper, but do appear to be successful. The teapot is rounded and is comparable to the sphere in Figure 7 of the main text, which we expect to produce lower resolution reconstructions than the bunny object. Interestingly, the same bright pixel artifact seen to the top right appears in both the teapot and bunny reconstructions. We hypothesize that this pixel corresponds roughly to the retro-reflective path from the perspective of the camera.

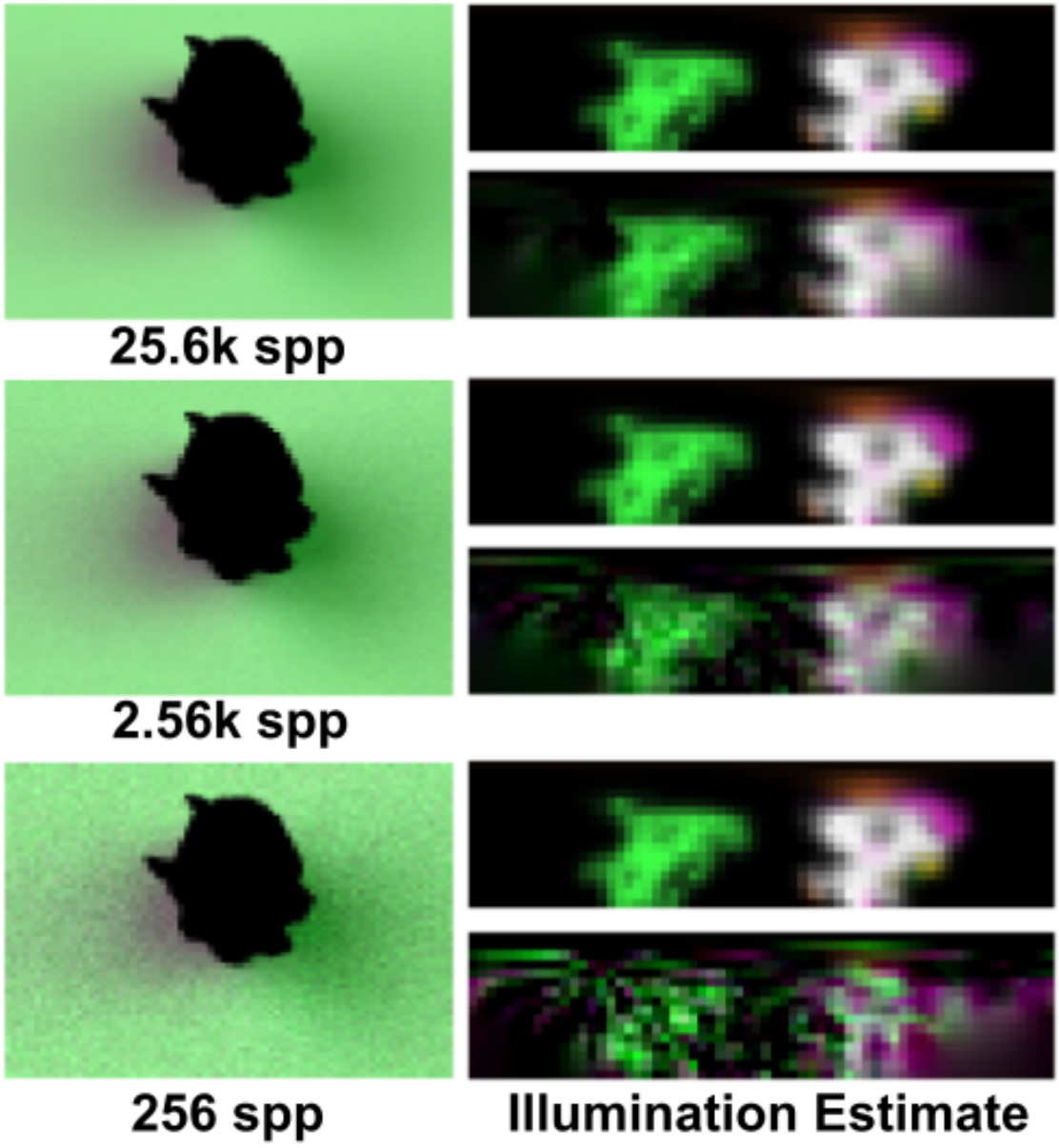


Figure 5-7: Estimated environment maps compared to ground truth given rendered observed images with decreasing number of samples per pixel (spp). The left column shows the rendered observed images with decreasing number of spp, and the right column shows ground truth and the estimated environment map given the corresponding observed image.

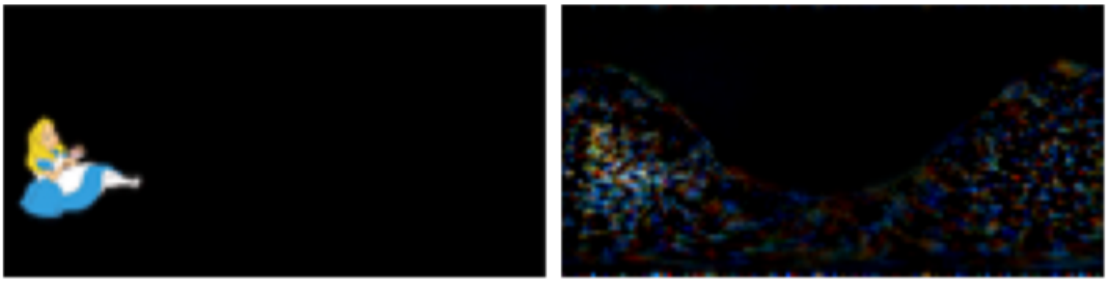


Figure 5-8: Higher resolution estimates. Illumination estimation using a higher resolution ray transport matrix (environment map 64×128), using the same regularization parameters used in the main text. Reconstructions are not significantly better than the lower resolution case and artifacts are apparent. This suggests that tuning regularization parameters could improve results for higher resolution reconstructions.

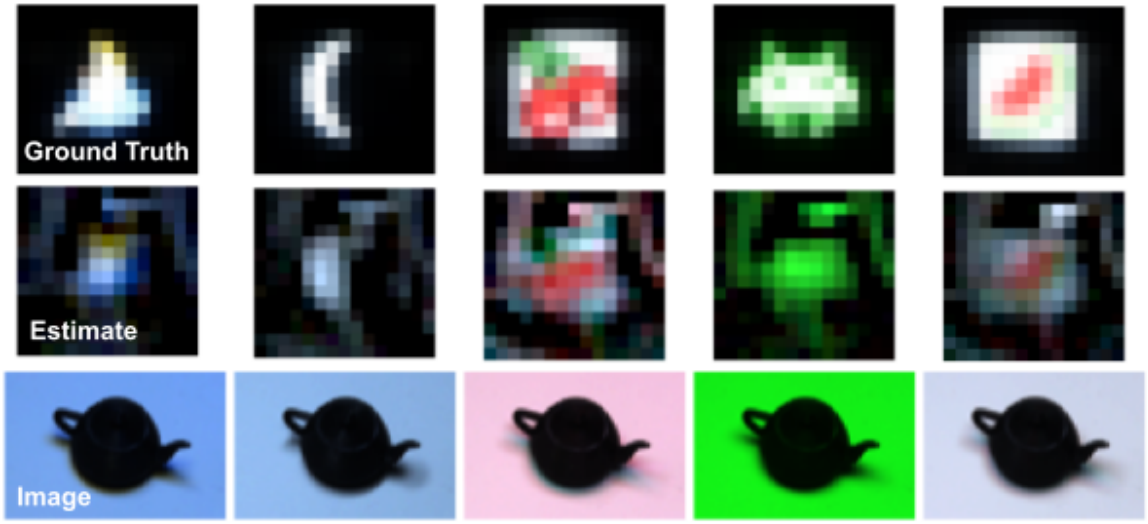


Figure 5-9: Estimated environment maps for a 3D printed teapot. In the top row, we show a cropped ground truth environment map centered on a display placed above the object. The middle row shows the estimated environment map with the same crop applied, and the bottom row shows the observed images captured using the camera.

Real Data using 3D Printed Object The main text shows environment map estimates from rendered models and realistic environment maps. In Figure 5-10, we show estimated environment maps captured using a 3D printed bunny on a white sheet of paper placed in various real world environments. For each estimate, we first determine the pose of the camera using a known printed pattern printed on the white paper near the bunny location, we obtain correspondences from the camera image to the known pattern and use the OpenGV [54] library implementation of the UPnP [55] algorithm to obtain the camera pose. Using the known intrinsic parameters of the camera, we mirror the scene in Mitsuba2 by matching camera parameters to Mitsuba's pinhole camera model. Since the 3D printed object was not affixed to the sheet of paper, we also perform a manual alignment step to ensure the 2D position and 1D rotation of the bunny match the captured image. We point out that much of this pipeline could be automated using standard object detection and pose estimation techniques. We also suggest that future work could utilize Equation 5.8 to refine the object position directly using a renderer capable of calculating derivatives of geometric parameters, such as the pose of a mesh.

Overall, the reconstructions are reasonable, and all major bright sources in the scene are recovered. It may not be surprising that the sun is recovered, but for the more indirectly illuminated scenes, the reconstructions contain more interesting detail. For example, in the second major row of Figure 5-10, most of the blue sky and a specular reflection of the sun from a building surface are recovered. In the first major row, the square shape of the window is recovered, as well as the light source in the upper left corner. There appear to be some artifacts near the horizon, as expected and discussed in relation to Figure 7 in the main text.

Notably, the reconstructions appear to fixate on the brightest sources, but when the brightest sources are blocked, as in the last row of Figure 5-10, the resulting environment map estimates contain less bright sources. It is a topic of future work to explore the effective dynamic range of estimated environment maps.

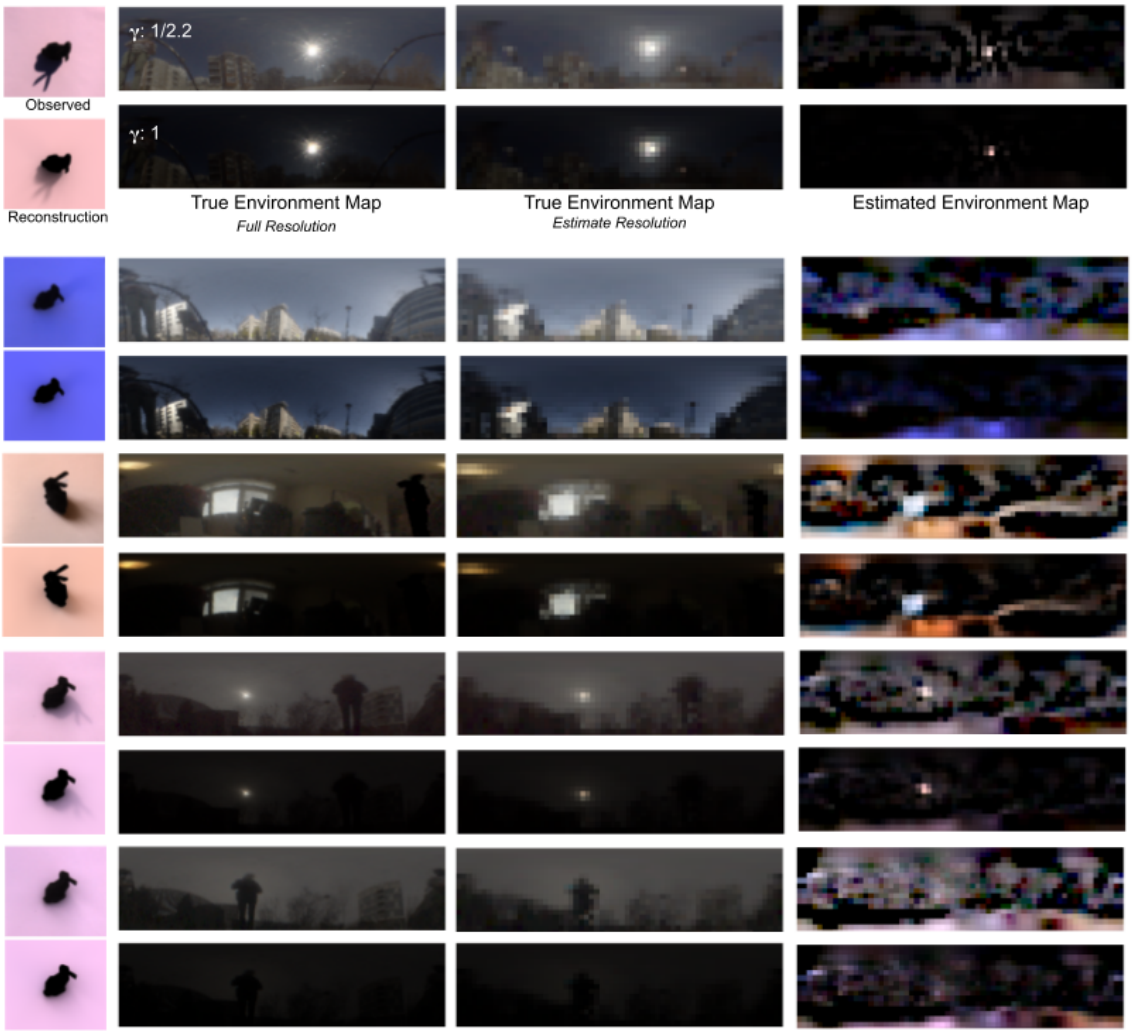


Figure 5-10: Environment map estimates using real data captured using the 3D printed bunny placed on a white sheet of paper and placed in various real environments. The left column shows the observed image captured by the camera and the corresponding reconstruction from the estimated environment map. The middle two columns shows a full-resolution ground truth environment map displayed with two gamma correction factors and the same environment map resized to the dimensions of the estimated environment map. The right column shows estimated environment maps. All environment maps are shown with two gamma correction factors to highlight the brightest sources in the scene.

Differential Illumination Estimation One application of using objects as cameras could be for static surveillance cameras that observe a scene over a period of time. While most of the environment map will remain unchanged, can a person walking by be detected in the subtle changes to the observed image?

In Figure 5-12, we captured a sequence of four images of the bunny, each at a different point in time as a person was walking by. As seen in the top row ground truth environment maps, the person is moving from right to left from the perspective of the object. We preprocess the observed images by subtracting the first frame from the last three frames. As such, our observed images contain both positive and negative values, as seen in the bottom row of Figure 5-12. Given these observed images, we perform the environment map estimate without the non-negativity constraint, so the estimated environment map contains both positive and negative values. We can observe a moving negative region as the person blocks the sky behind them, and a brighter region where the sky was originally blocked in the first frame.

Such an approach could be used to detect subtle variations in the observed scene, and this experiment demonstrates the feasibility of estimating "differential environment maps" that explain how the observed scene changes as the surrounding illumination changes.

Albedo Estimation using Realistic Environment Maps In the main text we demonstrated the simultaneous estimation of albedo and illumination for somewhat simple localized light sources. We show a similar result in Figure 5-13, but in this case use more realistic environment maps. We found that letting the alternating update method run to convergence led to poor estimates of the individual environment maps, while the albedo appeared largely unchanged for much of the optimization. Instead, we show albedo and environment map estimates after only 3 iterations. The resulting albedo looks reasonable, and the environment maps appear to correspond far more closely to ground truth than the results obtained after full convergence.

More analysis is necessary to determine when our alternating update scheme is successful, but this result highlights some of the inherent difficulty of this setting. In

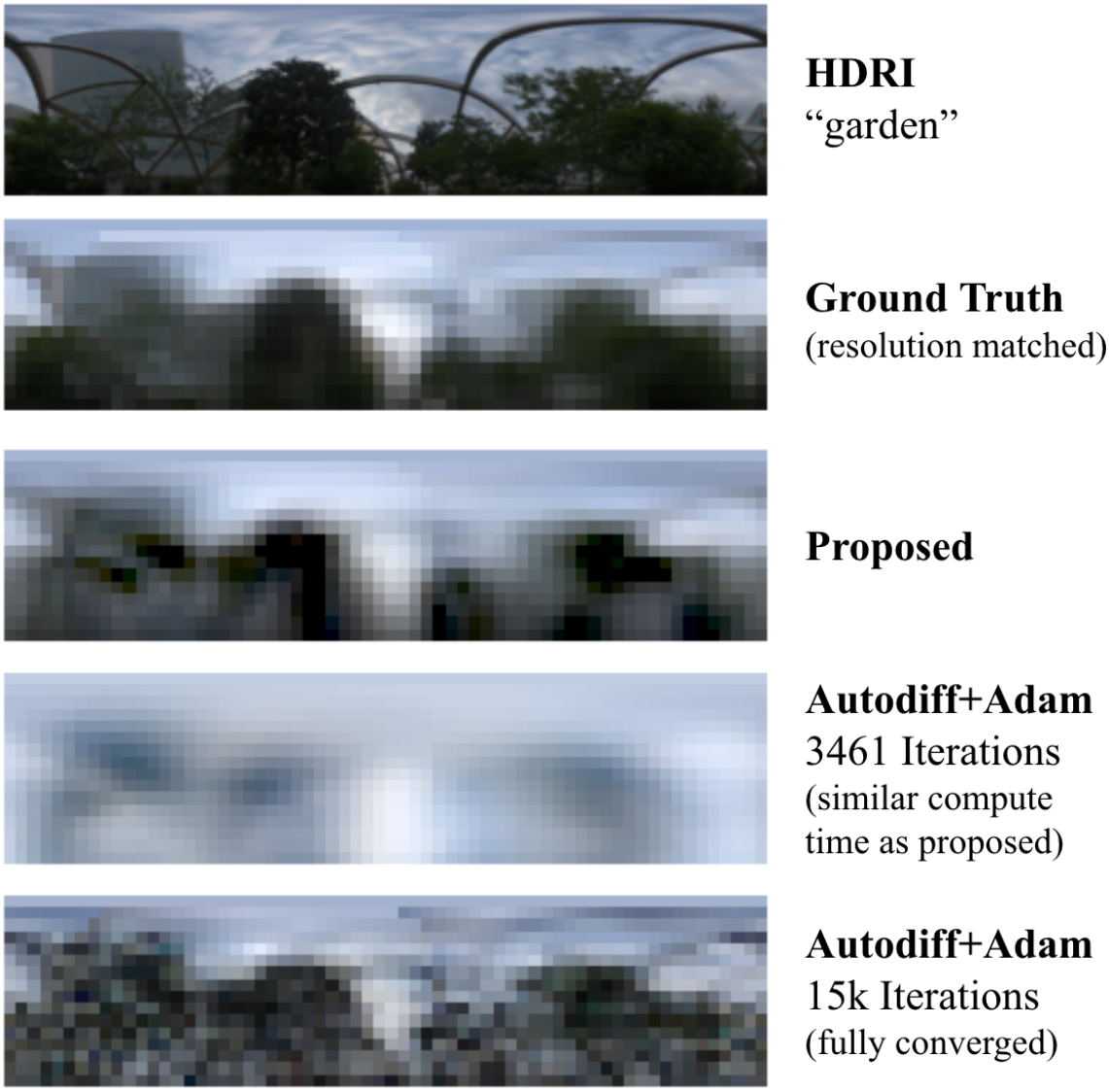


Figure 5-11: Visualization of main text Table 1 results. Predicted environment maps compared to the gradient descent baseline for two stopping criteria. Within a similar computation time as the proposed method gradient descent produced lower quality estimates, requiring a greater number of iterations to converge. While convergence may be accelerated by using a larger learning rate, this led to poorly converged results. Table 1 in the paper contains the RMSE/SSIM values for these images.

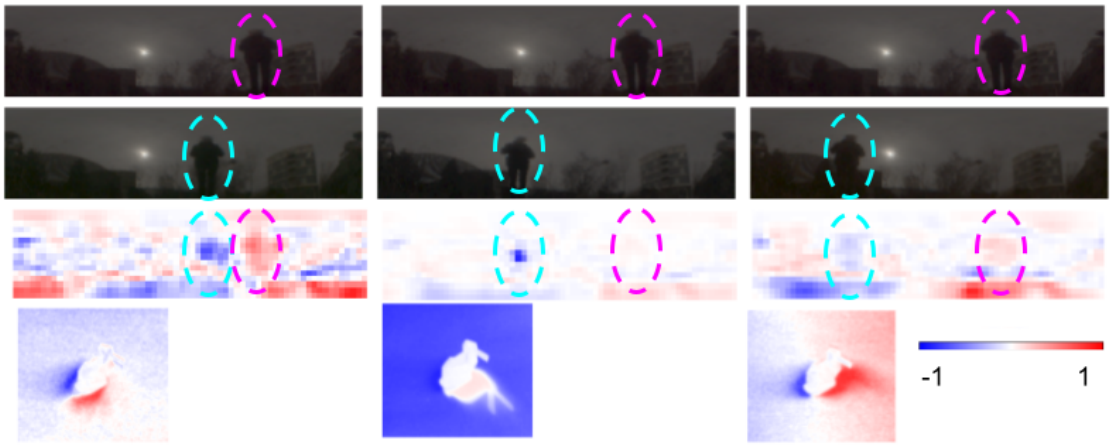


Figure 5-12: Estimated "differential environment maps" given a temporally subtracted image. The top row shows the starting frame from a sequence of 4 images with a human figure (magenta). The second row shows the subsequent 3 images in the sequence as the human figure (cyan) moves from right to left in the frame. The third row shows the estimated environment maps without a non-negativity constraint applied, showing how adding a figure subtracts light (cyan), and removing a figure adds light (magenta) to the scene from particular directions. The bottom row shows the observed differential images taken by subtracting the observed image corresponding to the top row from the observed image corresponding to the second row. All images have been normalized with a maximum value of either [-1,1] for display purposes.

particular, it appears as though an excess of energy was placed in the top center of the fully converged environment map estimates, which corresponds to the retro-reflective light direction. This led to a reduction in residual error but did not improve results.

5.4 Discussion

The paper assumes much of the object and surrounding surface is known, such as the surface geometry and material properties. While we show some ability to handle unknown diffuse albedo, there are other sources of model mismatch that lead to failed illumination estimates. To provide such a case study, we attempted to use photogrammetry to create the surface mesh and approximate material properties from a partial set of views. In this way, the complete model would not be needed *a priori* and could be obtained using standard computer vision tools. Results can be seen in Figure 5-14. We believe these results highlight additional sources of model mismatch

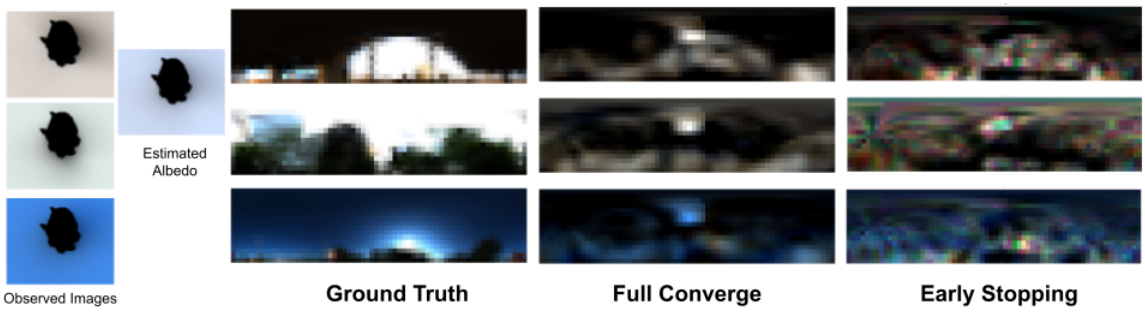


Figure 5-13: Simultaneous estimation of illumination and albedo for realistic outdoor environment maps. Solving the same problem shown in Figure 3 of the main text, but with realistic extended sources. On the left column we see three observed images. Given known geometry but unknown albedo, a ray transport matrix was generated using a white albedo for all surfaces. The second column shows the per-pixel estimated albedo after convergence. The environment maps associated with each observed image are shown in the next 3 columns: (third column) The ground truth environment map for 3 realistic scenes, (fourth column) estimated environment maps after running all iterations, (fifth column) estimated environment maps when running for 3 iterations where more structure is preserved.

that can be addressed in future work. We also propose a simple extension to the paper that may handle some of these sources of error.

Below, we describe our photogrammetry experiments. We make use of an existing photogrammetry pipeline Meshroom [80, 44, 6] to recover camera poses, scene geometry, and diffuse albedo.

Using Meshes captured from Photogrammetry In practical settings, known 3D shape and camera orientation are rarely available. We demonstrate our approach when the visible surface is entirely estimated by photogrammetry. We capture 10 images along a line about 2 meters away from the object, sampling a limited set of viewpoints within about a 20 degree angular fan originating from the object. The recovered surface mesh from photogrammetry does not model the back surfaces of the objects (see Figure 5-14a).

In order to obtain better results of texture estimates of our surface mesh, we found it useful to capture two images at each camera pose. The first image is taken with a flash, and the second with no flash. We subtract out the non-flash image from the flash image to remove all non-flash illumination from the image. The recovered

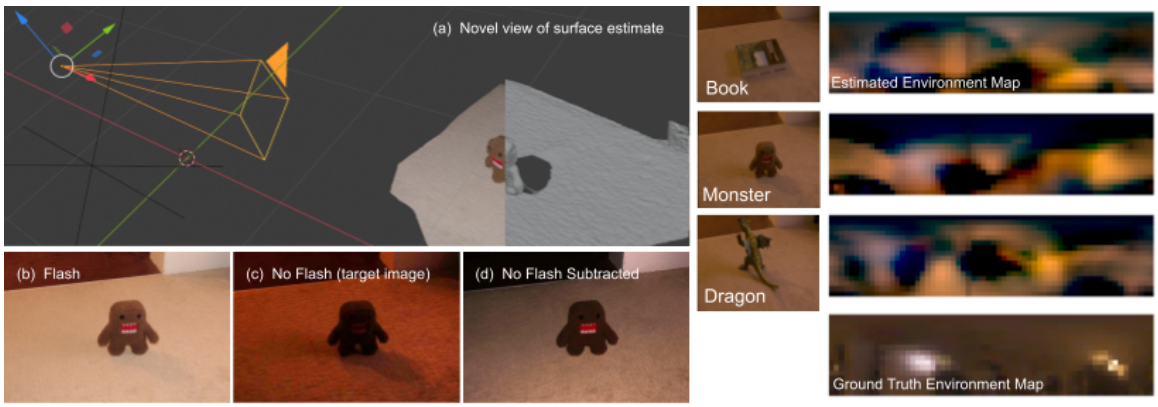


Figure 5-14: Estimation results from meshes obtained using photogrammetry. The left set of panels shows the capture and pre-processing approach: (a) Using photogrammetry, we obtain a camera pose, surface mesh, and diffuse texture. We capture two images (b) a flash image and (c) an image without a flash. (d) We use the flash-only image as input to the texture estimation using photogrammetry. On the right section, we use the approach on the left to recover the 3D geometry and diffuse texture of three objects: a book, monster, and dragon. For each of these objects we generate the corresponding ray transfer matrix and produce an estimated environment map. While the brightest sources can be identified in the estimated environment maps, the estimates appear to suffer from model mismatch, such as the warped recovery due to the larger dragon object and nearby light fixture. Addressing these failure modes is an area for future work.

flash-only images led to much better results in the full surface estimation pipeline.

Masking out Object Surfaces For all of our experiments, we create a segmentation mask to remove the pixels of the object in the scene. We found that the reconstructions still succeed in these cases, but the region of the environment map pointed back towards the camera (i.e. the retro-reflective direction) sometimes contains bright phantom sources. We hypothesize that this is due to the lack of shadow information for these illumination orientations (the object blocks the image of the shadow from the camera perspective). For this reason, object surfaces can help reduce this ambiguity. Adding object surfaces back into the reconstructions reduce these artifacts, but do not seem to significantly improve reconstruction in other incident illumination directions.

While the reconstructions in Figure 5-14 show the two brightest reconstructions (the blue on the left and yellow on the right), the estimated environment map contains

other artifacts. To this end, we hypothesize the following sources of error are highly influential, and could be addressed in future work. Given a ray transport matrix without these sources of model mismatch, the main text demonstrates that high frequency environment maps can be estimated.

Surface Mesh Error In Figure 5-14a, the carpet surface seems well captured. However, the resulting geometric texture produces a complex appearance for grazing illumination angles near the horizon. If the recovered surface mesh is incorrect, a small error in this mesh could cause it to cast a shadow over many observed pixels. Simplifying the resulting mesh and baking in this fine surface detail into normal maps may reduce the effect of false self-shadowing while still maintaining realistic appearance for less extreme incident illuimation angles.

Albedo Estimation Error We use flash-only images (using the processing described above) for photogrammetry and albedo estimation. While the resulting textures appear correct, small stitching artifacts and other artifacts from non-uniform illumination from the flash are apparent upon close inspection. While our method can be somewhat robust to diffuse albedo under certain conditions, reducing these artifacts may be beneficial.

Object Large relative to illumination distance Interestingly, the estimated environment map in Figure 5-14 corresponding the "Dragon" appears compressed relative to ground truth. The two brightest sources appear to be shifted slightly to the left and are closer together than expected. It's important to note that the dragon is actually fairly large, about 30cm tall, while the nearest illumination source (the blue-tinted source) is only about 0.5 meters from the dragon. Furthermore, the other bright source, the yellow-tinted luminaire is only about 2 meters from the dragon center. In this situation, the dragon's size may introduce a warped environment map estimate due to parallax. The use of environment maps may be inappropriate for nearby light sources, and instead a more general 4D light field may be necessary to account for such effects. Regardless, warped estimated environment maps may still be useful for re-lighting or applications where accurate angular recovery is not needed.

5.5 Future work

5.5.1 Updating the Ray Transport Matrix

We might attempt to reduce model-mismatch by optimizing the ray-transport matrix using a differentiable renderer. While we did not make use of the feature in the experiments, Mitsuba2 [85] can calculate gradients of internal scene parameters with respect to rendered pixels. We sketch out a simple way to incorporate our lighting estimates to iteratively update the ray transport matrix. This would be very useful, as effects such as global illumination are not modeled by our linear diffuse albedo model, but could be "baked-in" to the ray transport matrix. In this way, we could alternately solve for illumination and rendering parameters to account for non-linear interactions of parameters such as color bleeding.

Given an initial guess of the scene parameters, θ, and an illumination estimate, $\tilde{\mathbf{x}}$, we can minimize the following objective.

$$\begin{aligned} \arg\min_{\theta} \quad & ||f_\theta(\tilde{\mathbf{x}}) - \mathbf{y}||_2^2 + \lambda_i R_i(\theta) \\ \text{s.t.} \quad & C(\theta) > 0 \end{aligned} \tag{5.8}$$

With regularization terms $R_i(\theta)$ and constraints, $C(\theta)$, which indicates physically valid parameter values, and $\mathbf{y}$, the pixel data for the captured image. Equation 5.8 can be solved using stochastic gradient descent. The resulting scene can be used to render a new transport matrix for environment map estimation using the method described above, alternating until convergence.

5.5.2 Adding robustness to model parameter uncertainty

Estimating scene parameters such as surface geometry using photogrammetry can produce errors in the forward model. If we know that we are uncertain about certain scene parameters such as precise geometry, BRDF, or camera intrinsic parameters, we should use this knowledge to assist in solving the inverse problem. We propose a simple approach that is analogous to domain randomization used in Chapter 1 for

data-driven methods. A detailed overview is provided in Appendix A. The key idea is to minimize the expected error under model uncertainty, which leads to a simple model averaging scheme.

5.5.3 Alternative Forward Models

Neural Ray Transport Model: There is an appeal in combining data-driven methods and the family of model-based linear inversion methods described in Chapter 4. We show that it is possible to train a neural network to reconstruct the ray transport matrix from a dataset of input-output pairs in Appendix A. This is an avenue for future work.

Volumetric Rendering: While we propose to use photogrammetry to estimate model parameters such as surface geometry, volumetric rendering may better model the uncertainty of scene parameters relevant to the problem. Volumetric rendering is also trivially differentiable, making it well suited for both the forward model and to update model parameters. We introduce a simple volumetric renderer which shows promise for future work.

5.5.4 Approximate Inverse of Continuous Linear Operator with Deep Neural Network

A linear operator, which for the discrete case corresponds to a matrix, A, can have a corresponding inverse operator. For the ridge regression case, this would be:

$$G = (A^\top A + \lambda \mathbb{I})^{-1} A^\top \tag{5.9}$$

Where G is the linear inverse operator for L2 regularization with regularization parameter λ.

If we compute G, can we train a neural network to approximate the rows of G?

For illumination estimation, it's more natural to consider A and G as 4D functions of 2D lighting angle, (θ, ϕ), and 2D pixel space, (x, y). In practice we observe that

G is typically sparse and smoothly varying under perturbations of all parameters (x, y, θ, ϕ). This is due to the structure of $(A^\top A + \lambda \mathbb{I})^{-1}$, which is sparse and similarly smoothly varying. In special cases, this inverse is Toeplitz, such that it can be modeled as a convolution.

For the discrete case, we simply generate a dataset from the rows of G, and train a DNN to approximate the row given a scalar corresponding to row number. We sample an illumination direction(θ, ϕ), and the DNN generates an image with dimensions (x, y) that resembles the corresponding reshaped row in G.

Furthermore, while we can sample rows of A, and potentially the rows of G (using a differentiable renderer), can we train a DNN to approximate the continuous linear inverse operator?

For the continuous case, let $f(x, y, \theta, \phi)$ be a function of illumination angle and observed image pixels, $L_i(\theta, \phi)$ corresponds to the continuous illumination function, and $L_o(x, y)$ is the continuous observed image radiance in pixel coords.

Therefore, the linear operator is applied by taking an inner product:

$$L_o(x, y) = \int_\Omega f(x, y, \theta, \phi) L_i(\theta, \phi) d\omega \tag{5.10}$$

The function $f(\cdot)$ is in general smoothly varying in (θ, ϕ) but discontinuities can exist. However, the inverse operator f^{-1} should be sparse and smoothly varying, as its discrete verions, G, is. Thus the inverse operation can be written:

$$L_i(\theta, \phi) = \int_\Omega f^{-1}(x, y, \theta, \phi) L_o(x, y) d\omega \tag{5.11}$$

Can we approximate f^{-1} using a DNN? This would be a function of input (x, y, θ, ϕ) and map to a scalar value. Then, at inference time, the DNN approximator would be queried at samples of the corresponding values for a given image $L_o(x, y)$ and desired environment map resolution for (θ, ϕ). A simple extension of this approach would be for the inverse network approximating $f(x, y, \theta, \phi)^{-1}$ to estimate the 4D light-field instead of just (θ, ϕ). In this way, the network would estimate the corresponding inverse operator over the full spherical light-field, e.g.: (θ, ϕ, u, v), for ray direction

and position along the hemisphere. In this way, (u, v) represents a position on the surface of a sphere surrounding the object and (θ, ϕ) would correspond to a local set of 2D directions, or positions on a hemisphere with center point at (u, v).

It would be more convenient to represent the function as an input of (θ, ϕ), and output as a 2D image with dimensions (x, y):

$$\text{DNN}(\theta, \phi) \rightarrow g_{\theta,\phi} \tag{5.12}$$

At inference time, we sample from the desired set of illumination directions we want to reconstruct and generate the corresponding matrix, G, or perform the inner products directly.

The intuition is that G should be highly compressible given that it is sparse and smoothly varying, so it would be well approximated by a DNN. Furthermore, if we could sample from continuous valued input of illumination direction, (θ, ϕ), then training would be quite easy. Regardless, training on samples of (θ, ϕ) by pre-computing G as a dataset may make the learned DNN able to smoothly interpolate for higher resolution reconstructions. It would also require less memory and reduced amount of computation depending on the necessary complexity of the DNN architecture.

As an added benefit, the architecture itself could introduce regularization, for example if it was a CNN decoder, it may benefit from "Deep Image Prior"-like regularization by exploiting spatial similarity and smoothness in observed image space, (x, y).

How to compute a row of G directly using a differentiable renderer?

With a differentiable renderer, could we render a row of G directly?

I've observed that $(A^\top A + \lambda \mathbb{I})^{-1}$ is sparse, smooth, and appears to be spatially localized. For object shadows, it often resembles a laplacian operator, such that in G, rows of $A^\top$ are selected for such that the image for a ray direction of interest is added to negative images of surrounding ray directions.

This intuitively makes sense, as it is almost a finite difference approximation of

the Hessian? Perhaps the gradient of the image with respect to incident ray angle could be used to calculate this directly? This would correspond to the gradient of the visibility function and other shading terms, and could be computed using a differentiable renderer like Mitsuba2.

5.5.5 Alleviating Storage Requirements for Large Linear Models

One potential drawback of the linear systems approach to high frequency illumination estimation is that the size of the model matrix $\mathbf{A}$ can become quite large for modestly sized problems, and this can lead to memory issues. In the main text we largely avoided this issue by limiting ourselves to low resolution measurement images and environment map estimates. However there are more sophisticated strategies that exploit the peculiar nature of the illumination estimation problem to reduce memory requirements.

The unwieldly size of $\mathbf{A}$ is due to the fact that it is effectively a 4-dimensional structure that maps 2D environment maps to 2D measurement images. However, all of the information required to generate $\mathbf{A}$ can be stored in a lower dimensional surface mesh and texture map. This suggests the possibility of more memory-efficient inversion algorithms that could query a rendering engine to obtain necessary matrix elements rather than forming the entire matrix in memory.

As an example, consider the linear system posed in Eq. 6 of the main text. The system can be written more simply as

$$\mathbf{B}\mathbf{x} = (\mathbf{A}^\top\mathbf{A} + \mathbf{R})\mathbf{x} = \mathbf{A}^\top\mathbf{y} = \mathbf{z} \tag{5.13}$$

We note that $\mathbf{B}$ has dimensions $N \times N$ and $\mathbf{z}$ has dimension N, where N is the size of the environment map vector. These dimensions are independent of the number of pixels (M) in the measurement image. If we could form this sysem directly, without storing $\mathbf{A}$, this would make the use of high resolution measurement images more practical. The elements of $\mathbf{B}$ and $\mathbf{z}$ can be written as

$$B_{ij} = \mathbf{a_i}^\top \mathbf{a_j} + r_{ij} \quad , \quad z_i = \mathbf{a}_i^\top \mathbf{y}. \tag{5.14}$$

Here r_{ij} denotes the ij^{th} element of R, and $\mathbf{a}_i$ is the i^{th} column of $\mathbf{A}$, which can be generated using a rendering engine. We see that each element of $\mathbf{B}$ can be generated by rendering and taking the inner product of two frames. Because $\mathbf{B}$ is symmetric, N^2 total frames (N^2M total pixels) must be rendered to generate the entire matrix. These frames could be recycled to generate $\mathbf{z}$.

Although this method allows us to avoid storing $\mathbf{A}$ directly, we have traded speed for memory. Although $\mathbf{A}$ may be much larger than $\mathbf{A}^\top\mathbf{A}$ when the measurement images are large, $\mathbf{A}$ can be generated from just N rendered frames (NM pixels). As an intermediate option we might instead consider the case where we are allowed to store blocks of $\mathbf{A}$ with a maximum size of s^2 entries. In this case we can compute the product $\mathbf{A}^\top\mathbf{A}$ using block matrix multiplication. By re-using elements in the pre-rendered block, we can reduce the number of pixels that need to be rendered by approximately a factor of s, such that we only need to render $\frac{1}{2s}N(N-1)M$ pixels total.

5.6 Conclusion

Reconstruction using the shadows cast by objects onto their surrounding surface can be practically achieved using tools used commonly in computer vision and graphics research. Shadows are an important cue for high-frequency incident illumination estimation and can be essential for solving an otherwise poorly-conditioned problem.

Chapter 6

Discovering and Exploiting Hidden Cues

6.1 Data-driven Hidden Cue Discovery

The trained networks can be inspected in a number of different ways. One common technique is to calculate the class activation map in input image space. These maps show the gradient of the class prediction with respect to the input pixels. Effectively, this can be interpreted as a kind of sensitivity analysis. We apply this technique to the network trained in Chapter 3 in Figure 6-1. We observe that the network learned to make use of cues in the input image. Perhaps most interesting is that the images resemble what we might expect to see for an inverse operator.

Appendix A explores this idea a bit further, but mostly focuses on creating networks that learn the forward model. If the network is trained using input-output pairs from a simulation or real measurements, it would not be difficult to expand the idea in Appendix A to learn the inverse operator directly.

We also note that the gradient shown in Figure 6-1 can be interpreted as a column of a matrix, G, that is a linear approximation of the neural network. As such, it may be possible to obtain a linearized forward model from the neural network directly: $G = (A^\top A)^{-1} A^\top$, where A could be found by optimizing for its entries that have the corresponding equality. Furthermore, applying physical constraints, such as

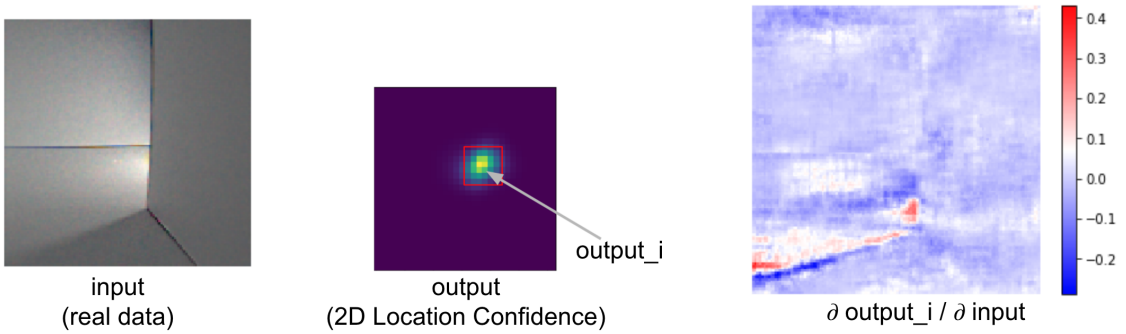


Figure 6-1: (Left): The background subtracted, or differential image used as input to localize objects in Chapter 3. (Middle): The network is trained to predict the likelihood of an object located within a 2D area. Since the network is differentiable, we can show the gradient of the input pixels with respect to the predicted location. (Right): Incredibly, the gradient resembles an edge filter than identifies the shadow edge and also reveals important pixels near the corner-floor and back wall.

non-negativity of A, may be a way of regularizing the neural network.

6.2 Hidden cues from the linear inverse operator

Using the linear inverse operator found in Chapter 5, we can easily extract maps of what input pixels are important to estimate light from different directions. Furthermore, we can utilize Cook's Distance [28] to analyze sensitivity of input pixels for specific scenes. This approach is analogous to the analysis used by the neural network based methods above.

Interestingly, we found that these cues were quite similar to the Laplacian of the rendered image with respect to incident illumination, which could be computed by a differentiable renderer.

Extracting the Shadow's Edge In Figure 6-2 we show that our algorithm implicitly amplifies the edge gradients to determine the intensity of illumination from a particular direction. In the bottom row we plot the rows of the inverse operator matrix $\mathbf{G} = (\mathbf{A}^\top \mathbf{A} + \mathbf{R})^{-1} \mathbf{A}^\top$ (referred to in [17] as "estimation gain images"). These rows are correlated with the input image to estimate the intensity of illumination

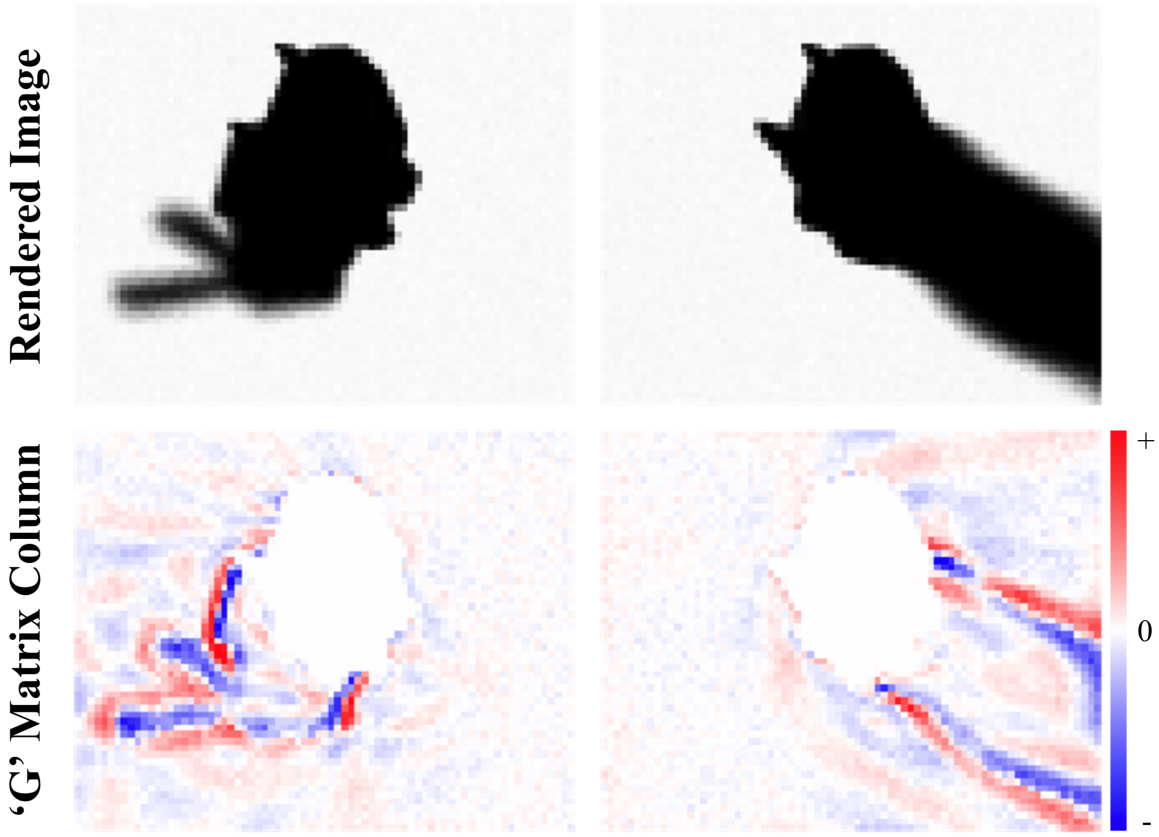


Figure 6-2: Our algorithm implicitly amplifies the edge gradients of shadows cast from a particular direction. (Top Row) Shadows rendered for point illumination above and to the right, and behind and to the left of the bunny-shaped occluder. (Bottom row) Corresponding rows of regularized inverse operator.

originating from a particular direction.

Determining Influential Image Pixels Given a target image measurement, we can quantify the influence that individual pixels have on our estimated environment map using a statistical tool known as Cook's distance [28]. The Cook's distance measures the effect that removing a measurement has on an estimated curve fit, and can be expressed as follows:

$$\mathbf{D_i} = \frac{\sum_{j=1}^{M}(\hat{y}_j - \hat{y}_{j(i)})^2}{ps^2} = \frac{e_i^2}{ps^2}\left[\frac{h_{ii}}{(1-h_{ii})^2}\right] \tag{6.1}$$

Here $\hat{y}_j$ denotes the re-projected curve fit (that is, the image rendered using our environment map estimate $\hat{x}$) obtained when all measurements are used for the fit, and $\hat{y}_{j(i)}$ denotes the fit obtained after the i^{th} data point has been removed from the measurement set. The quantity $s^2 = \frac{\mathbf{e}^T\mathbf{e}}{M-N}$ is the mean square error of the fit $\hat{\mathbf{y}}$ calculated from the residual vector $\mathbf{e} = (\mathbf{y} - \hat{\mathbf{y}})$ and the dimensions of the model matrix. The value h_{ii} is referred to as the *leverage* of the measurement y_i and is defined as the i^{th} diagonal element of the hat matrix $\mathbf{H} = \mathbf{A}(\mathbf{A}^\top\mathbf{A} + \lambda_r\mathbf{I})^{-1}\mathbf{A}^\top$.

In Figure 6-3 we've plotted an image of Cook's Distance corresponding to a specific target image measurement. As expected, we notice that the pixels with the largest Cook's Distance appear to lie within the shadowed regions.

Assessing Object-Camera Performance Although the Cook's Distance is useful for determining which pixels in a specific set of measurements are most influential, we may also want to assess how influential individual pixels are in general, independent of any specific set of measurements. For this purpose, the *leverage* of individual measurement channels can be a useful metric. The leverage of pixel i, previously defined as the i^{th} diagonal component of the hat matrix, can also be defined as follows:

$$h_{ii} = \mathbf{a}_i^*(\mathbf{A}^\top\mathbf{A} + \lambda_r\mathbf{I})^{-1}\mathbf{a}_i^{*\top} = \frac{\partial\hat{y}_i}{\partial y_i} \tag{6.2}$$

Here $\mathbf{a}_i^*$ corresponds to the i^{th} row of the matrix A. Figure 6-4 includes an image of per-pixel leverage values calculated for the bunny-camera. We note that, as

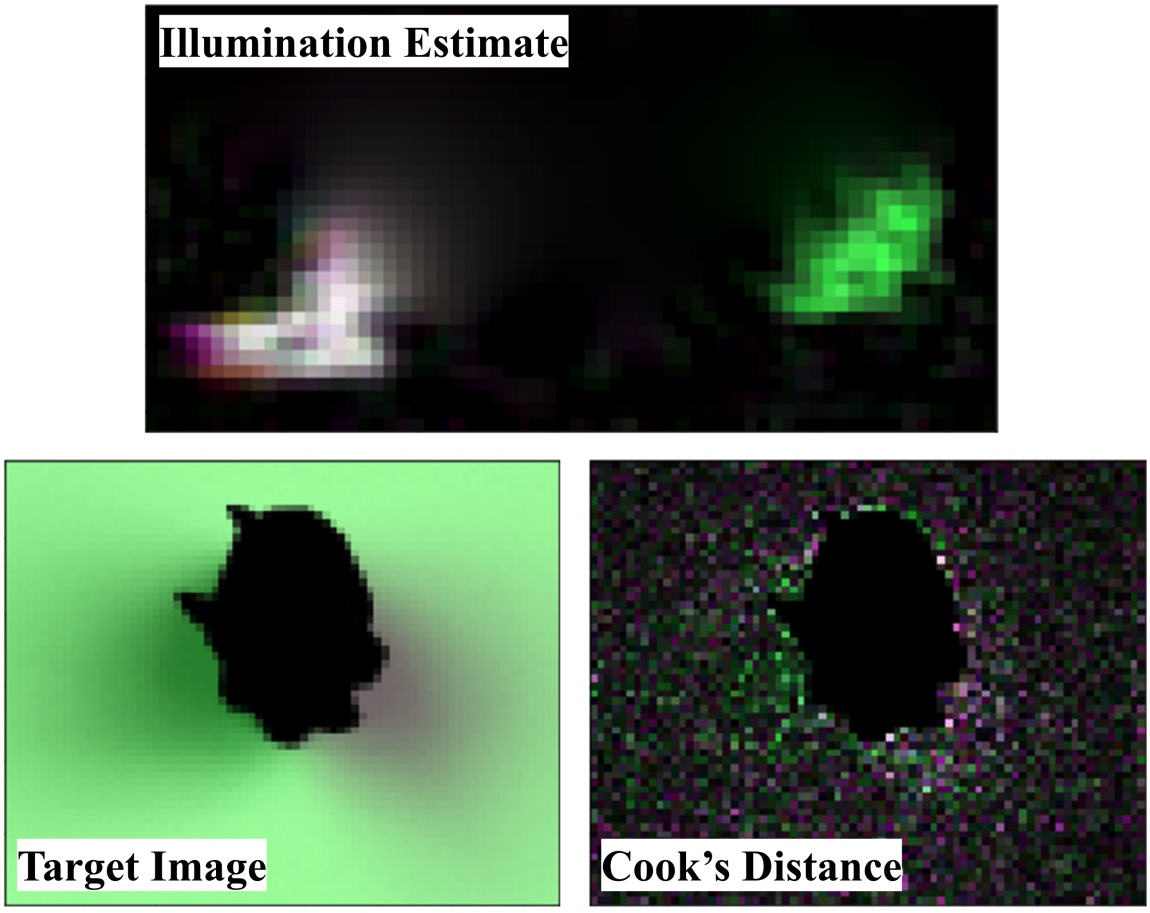


Figure 6-3: We plot an image of the Cook's distance (bottom right) associated with each pixel in the target image shown on the bottom left. The estimated environment map associated with this image is shown in the top row. The Cook's distance image has been compressed with a gamma value of 0.5 to highlight interesting features.

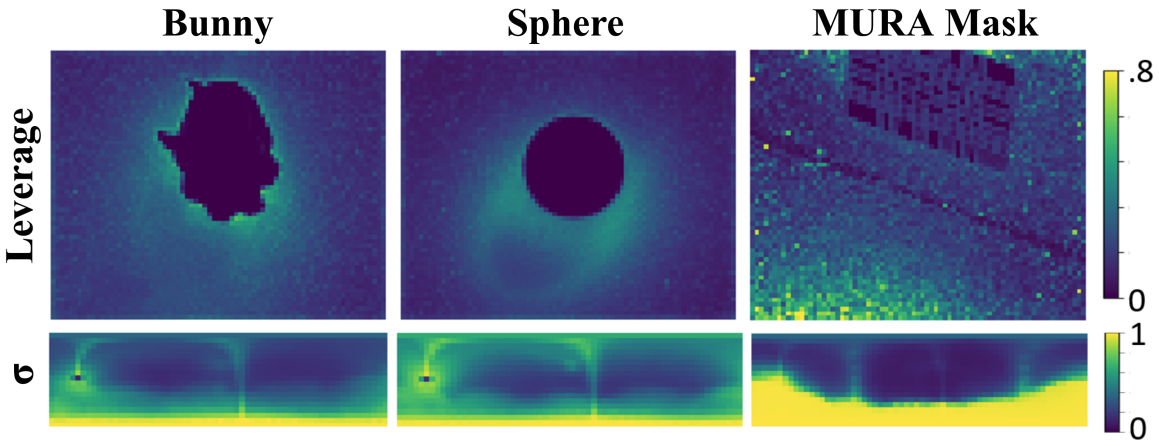


Figure 6-4: We illustrate how occluder shape impacts object-camera performance by generating images of per-pixel leverage (top row) and environment map uncertainties (bottom) for three occluder shapes: a bunny, a sphere, and a coded aperture mask [59].

with Cook's distance, the pixels closest to the base of the bunny appear to be most influential.

Given a particular object-camera configuration, we might also be interested in assessing which entries in an environment map we can expect to reconstruct accurately. We take the square roots of the diagonal entries of the covariance matrix of our least-squares fit: $\Sigma = (\mathbf{A}^\top \mathbf{A} + \lambda_r \mathbf{I})^{-1}$—that is, the inverse of the Hessian of the loss function defined in Eq. 5.3. We ignore the scene smoothness prior for the sake of analyzing the intrinsic properties of the object-camera.

A plot of these relative uncertainty values is also shown in Figure 6-4. From this image, we anticipate that the bunny-camera will be best at reconstructing illumination arriving from above and slightly to the left or right of the bunny.

Effect of Occluder Shape Our analysis of the bunny-camera makes it clear that the shape of the occluder can have a significant effect on the object camera's performance. This has important practical ramifications. For instance, we might choose to opportunistically exploit occluders found "in the wild" that are likely to produce accurate reconstructions of illumination originating from certain directions. Alternatively, we could design an optimized occluder shape that can be 3D printed and used as an object camera in the real world.

We demonstrate the effect of occluder shape in Figure 6-4. We show leverage and uncertainty maps for three different occluder shapes—a bunny, a sphere, and a 2D coded aperture mask. Compared to the bunny-camera, the sphere-camera achieves reconstruction uncertainties that are more uniform across the hemisphere, but that are higher on average. In contrast, the coded aperture mask achieves very low uncertainties when the shadow of the mask falls within the camera field of view, but uncertainty is high when the mask is illuminated edge-on, and very high when light originates near the horizon.

6.3 Good cues are robust to model parameter uncertainty?

We observed that adding certain types of regularization to the objective used to solve our inverse problems produced inverse operators, either as a neural network or matrix, that when analyzed produced intuitive hidden cues. We hypothesize that good cues are thus a product of inverse operators that are robust to model parameter uncertainty.

By solving inverse problems, we might uncover hidden cues present in certain types of environments. Once we extract and characterize these cues, is there an opportunity to develop imaging systems that make use of these cues? This section explores this question in more detail.

We also suggest and end-to-end framework that might be useful for designing imaging systems that exploit hidden cues, without the manual process of characterizing or understanding hidden cues by a human designer.

6.4 Future Outlook: Designing Imaging Systems that Exploit Hidden Cues

Over the course of billions of years, natural organisms have evolved highly effective imaging systems that often surpass the best designs produced by humans. Imaging

systems are ubiquitous in nature, almost every animal capable of movement has some form of image-based perception system. Natural designs are highly efficient and robust, often consisting of hardware and information processing that is well suited for the actions an animal must perform to survive in their ecological niche. Furthermore, the perception systems of natural systems often make use of "unusual cues" present in their environment, leveraging as much of the environment as possible to accomplish necessary tasks for survival.

As in nature, imaging systems are essential in modern robotics. However, modern imaging systems are often designed independently of a particular robot's task, and thus may be limited in their lack of adaptation to a given environment. For example, a robot designed to pick up trash may use an RGB video camera, and fail to detect transparent plastic bottles scattered around it. The robot's designer must identify an opportunity to modify the imaging system: perhaps transparent plastic bottles exhibit florescence when illuminated by UV light. Thus, a more robust imaging system might incorporate UV illumination. A specialized task often provides an opportunity for the imaging system designer. Unfortunately, there is presently no systematic way to discover these unusual cues in the environment. How can we discover how to exploit these cues in order to close the gap between what humans and nature can create?

Given continued trends in physical simulation, falling costs of computation, and general purpose large scale optimization techniques, the question is if these capabilities could enable autonomous "computational discovery" of designs beyond what is possible with human designers today. Such an approach may lead to more rapid development of artificial perception systems that are perfectly adapted to a given environment, making them lower cost and more reliable in accomplishing their tasks.

We propose a computational design framework consisting of an system architecture selection step, followed by parameter tuning for a specific task in a simulated environment. The performance of the tuned system would then inform future system architecture selections and enabling an iterative process of imaging and information processing system design. In order to develop a system that could be implemented in practice, we introduce a number of trainable optical components that can be composed

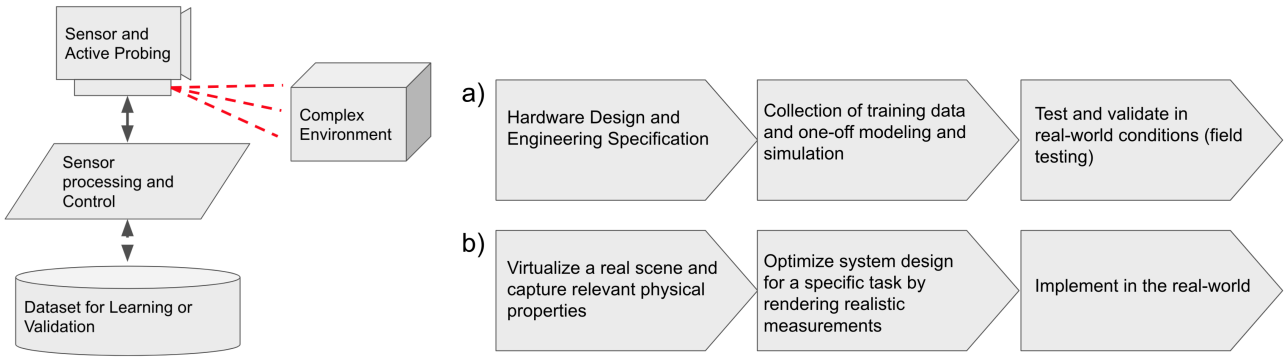


Figure 6-5: **Left**: A typical high-level architecture for a perception stack employed in a real environment. A sensor observes a complex environment, sometimes using active probing (e.g. LiDAR). Data is then processed to provide suitable information (e.g. state estimates) for control or other tasks. The raw data can be saved and labeled in a dataset that can be used for offline training and validation. **Right**: a) The standard design process for such a system utilizes an engineered spec, collection and labeling of data, and then real-world testing. b) The proposed approach makes use of an end-to-end optimization framework to produce the hardware design and engineering specification automatically in a process we call "computational discovery."

and trained end-to-end.

6.5 Computational Discovery of Computational Imaging Platforms

Taking inspiration from nature and the trends in optical design and machine learning, we propose a framework, we call "computational discovery," for using computational methods to design novel perception systems automatically. Since most of the pieces have been developed in the last few years, we believe an integrated design framework is now possible. However, there are some remaining open challenges limiting direct application of these ideas, which we also discuss below.

A high level overview of the approach can be seen in Figure 6-6. This approach is similar to that described in Section 2.4.5, however it adds an initial architecture selection step before optimizing the learnable parameters. In essence, the architecture describes how individual components are combined, with each component containing parameters that may be trained.

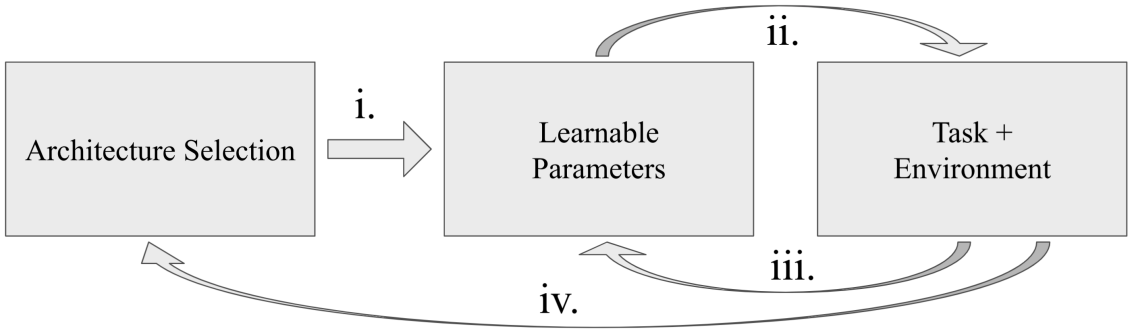


Figure 6-6: A high-level overview of the proposed design framework. i) given a library of components, Architecture Selection proposes a component graph that is used to encode an instance of a parametric imaging system and neural network for information processing. ii) The parameters in these components are initialized and the system is evaluated on a task in a simulated environment. iii) The loss from this task objective is used to calculate the gradient of the parameters which can be updated using gradient descent. The loop between ii. and iii. continues until convergence. iv) The final best task loss value is used to score the selected architecture, which is used to inform better architecture selection. The entire loop can then be iterated until convergence.

6.5.1 Architecture Selection

In architecture selection, we define graph from a library of components (see Figure 6-7), each with a set of tunable parameters. The architecture consists of a graph consisting of components as nodes, and edges describing how each component connects to their neighbors. We start with a given image sensor component as the root node in the graph, and add additional optical components. In this way, the architecture selection step must generate a candidate graph that represents the optical system as a set of connected components.

We also include the architecture of any information processing neural network as part of the architecture selection step. This network would take the image sensor values as input and make predictions depending on the desired task. Importantly, the entire architecture, both optical and neural network, contains parameters that can be trained end-to-end given a task objective.

Generating the Component Graph

Similar to the architecture selection produced by [9], we can define a deep neural network (DNN) that iteratively builds the architecture by predicting what other components to add to a given component. A key feature of this approach is the addition of a "EXIT" component that represents that nothing is connected, ensuring the component graph eventually has the equivalent of an entrance pupil, where light would enter into the optical system.

The architecture selection DNN could be trained using Q-learning or another reinforcement learning technique, where the reward function could be derived similarly to Section 6.5.3.

Traditional Optical Components

Components could consist of traditional optical components. For example, a library could consist of an image sensor, lenses, stand-off distances, and phase masks. Starting from the image sensor, the architecture selection network would choose relevant components. For example, it may select a stand-off distance, followed by a phase mask, a lens, and completed by the EXIT component.

Volumetric Design Component (VDC)

One downside to selecting from a set of traditional optical components is that the space of possible designs is limited to some valid combination of library components. We propose to add a general-purpose component that defines a volume, where each point in the volume contains parameters such as the absorption coefficient, scattering, and index of refraction. We note that the surface refractive optics, such as lenses, are important information, and as such this component would be capable of representing these surfaces using a local normal vector within the volume. We call this a volumetric design component (VDC).

Such a component is highly general purpose and can simplify much of the component graph. Given a graph consisting of an image sensor followed by a VDC, most animal-eye

like optics could be represented.[1]

While the VDC concept is highly desirable, real-life implementations would require a sufficiently advanced 3D printer, or some other way of reproducing the volumetric information. A simple 3D printer with black plastic filament and a clear plastic filament could be used to produce a minimal design. It may be possible to use just black filament, but constraints to ensure the proposed designs are 3D printable would be necessary. With an optically clear and optically absorbing material, any VDC using these two plastics could be printed and the design would be structurally stable.

Learnable parameters

Each optical component described so far would contain a number of parameters that are differentiable with respect to the base image sensor component. Given modern auto-differentiation frameworks, many components and learnable parameter combinations are possible. These parameters could include lens curvature, phase mask codes, or stand-off distances between components. These parameters can be tuned for a particular application using gradient descent.

In the case of VDC, volumetric path tracers are trivially differentiable, making the parameters used by the phase function at each point within the volume part of the set of learnable parameters.

6.5.2 Task-specific Parameter Optimization

Given an architecture consisting of a component graph and the corresponding set of trainable parameters, we can define a objective to optimize for a particular task. We do this by creating a simulation of the component graph in a differentiable renderer, such that we can render the image sensor values given the optical system defined by the component graph. This image is then fed to the DNN which makes a prediction. This stage has been explored in related work (see Section 2.4.5), and can make use of many of the techniques developed so far in the community.

[1]Not including the curved surface of the retina since the image sensor would likely be flat.

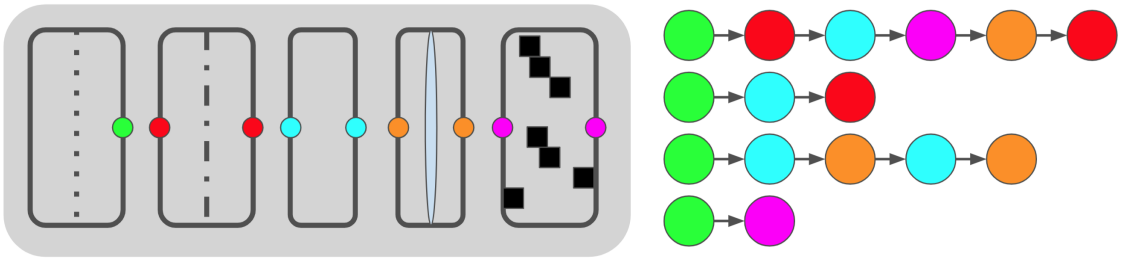


Figure 6-7: In order to define an imaging system, we produce a set of optical components in an optical library. Each component contains an input, and most also contain outputs, which represent how they may be connected with other compatible components. For example, on the left we have library components from left to right: An image sensor, coded aperture, standoff distance or air gap, lens, and the VDC, or volumetric region where each point in the volume corresponds to an optical material (via absorption or refraction) that could be 3D printed. Each of these components contain some parameters that are differentiable with respect to their outputs or inputs, making them well suited for propagating gradients using backpropagation. On the right, we show different instances of component graphs that might be possible. These graphs encode the optical system design. Note that while not shown here, branches are possible if components have multiple possible outputs, such as a beamsplitter.

Simulation of Environment

A particular challenge when using a novel imaging system is the collection of training data. The simplest form of training data is to use existing image datasets. If we assume the entrance pupil of an imaging system is sufficiently small such we can approximate the observed scene as optically far away, we could make use of images taken with a known camera field-of-view (FOV) to obtain the radiance values that enter the imaging system. In a rendering engine, this could be accomplished by creating a rectangular light source sufficiently far away and orthogonal to the imaging system, with a chosen image as a texture to modulate the light source intensity.

While the dataset generation approach enables the use of existing image datasets, it does not properly account for more complex optical effects. A more general purpose approach would be to make use of a full simulation of the environment. Using photogrammetry or other scene reconstruction techniques, a real environment could be scanned to obtain accurate geometry and surface reflectance data. This environment model could then be used by the renderer to produce a close to physically accurate

simulation of the full optical system. While it may require significant effort to obtain high-fidelity models real-world scenes, it would enable the "discovery" of unusual cues within a chosen environment that the optical system could make use of. For example, if the scene contains many reflections from puddles of water along the ground, a perception system may make use of these reflections to improve depth estimates by combining these reflections with direct views to better triangulate an object on the horizon.

Optimization

Given an architecture selection, dataset, and task objective function, the entire set of system parameters can be trained end-to-end using gradient descent. The goal of this optimization would be to discover the best set of parameters to solve a particular task. While DNNs often find very good local minima, it is less likely the parameters corresponding to optical components would avoid local minima and poses a potential challenge for optimization. As such, multiple runs with different initialization would be needed to select from the best set of parameters. An area of future work would be to define optical components that are less susceptible to getting trapped in local minima. Regardless of these challenges, natural evolution appears to have produced the eye after following incremental steps, so perhaps using a highly configurable component like the VDC may avoid these issues.

6.5.3 Improved Architecture Selection

After optimizing for the set of imaging parameters on a task, the best task objective value can be used to score the performance of a particular architecture selection. This score could serve as a reward function for training the architecture selection network. In addition to using something like a architecture selection network like that used by [9], an evolutionary strategy could be employed that would simply perturb the component graph by randomly adding or removing components. We anticipate such an approach would be highly inefficient as many proposed architecture selections would

be unusable or produce poor images, but may serve as a starting point. This step may prove extremely challenging since it may require an enormous amount of computation. For this reason, we are particularly interested in the VDC, as such a general purpose component may make it unnecessary to apply this step at all. In effect, the VDC could become any combination of components in the library for a given task, so an outer loop may not be necessary. Despite the potential of the VDC concept, the exact configuration of the DNN used for information processing and specific shape and resolution of the VDC would still need to be decided, making it likely some level of outer-loop architecture selection will still be necessary.

Regardless of the selection scheme, starting from random guesses for how optical components may be combined, eventually better architectures will be selected that will perform better on a given task.

6.6 Experimental Design

In order to validate the integrated computational discovery framework described above, we envision started with a platform we call "Robobee."

Our approach would incorporate a simulated model of the environment, optics, and information processing system (as a DNN), and our goal would be to optimize a set of parameters that could be easily translated into a manufacturable specification. We also envision the development of a testing platform that contains a multitude of configurable sensor designs that would enable a rapid assessment of performance in a real environment before mass-manufacturing the optimized perception system.

6.6.1 Robobee Platform

We envision Robobee as a small, unmanned aerial vehicle with on-board sensors and computation. Flight is accomplished using motor powered propellers in a quadcopter configuration. This choice of propulsion allows us to use a large quadcopter for testing and more easily scale down the final design.

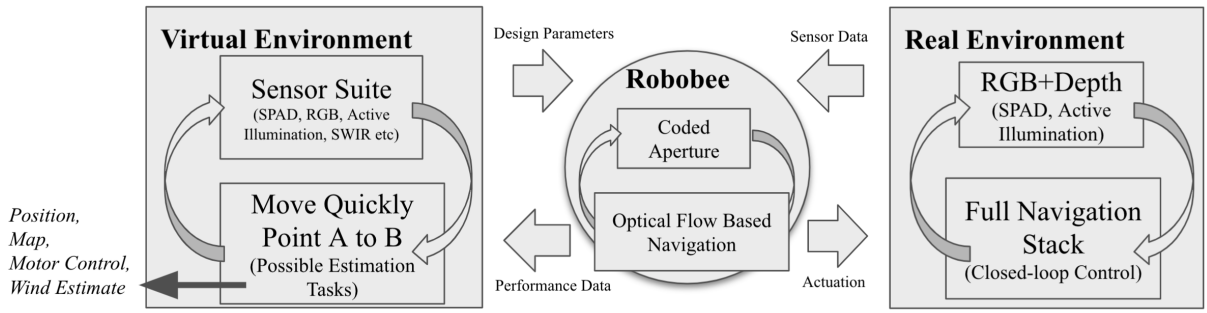


Figure 6-8: The Robobee is a small UAV with a vision system designed to solve a specific task. In order to automatically discover the imaging system, we use a virtual environment to evaluate different combinations of possible sensors for a navigation task. These designs are evaluated in a real platform which is configured with a general purpose navigation stack. Information from the simulated and real validation environments are used to refine the final specification defining the Robobee, which only contains a subset of possible sensors and computation.

6.6.2 Collision-free Movement Task

For all Robobee experiments, we propose a high-level navigation task: move forward through an environment without collisions. The task will be considered solved if the Robobee is capable of travelling forward along an approximate heading while avoiding obstacles. A high-level plan could be defined using GPS or other data to sketch the approximate local headings. In this way, the Robobee will perform local navigation using only visual sensors, which would be highly useful for navigating in cluttered environments.

6.6.3 Experiment 1: Sensor Suite Selection

While full novel imaging system design is an ultimate goal, we first propose to demonstrate the imaging architecture selection using a restricted scenario. In this setting, we define a library of sensor modules such as standard video sensors, SPAD sensors, and active illumination sources. The optical components would enable some level of configuration, such as coded masks, or a discrete set of lens options, but the majority of the optical system would be defined ahead of time. In this way, the architecture selection step would attempt to select the best sensor and optical

configuration given a possible suite of options.

Given sensory input from the selected sensor configuration, a DNN would be trained to predict local control movements that would move the platform forward while avoiding obstacles. This DNN would output high-level controls such as elevation, rotation, and lateral movement, while the quadcopter control system using accelerometer, gyroscope, and barometer would handle the low-level motor control.

Our design pipeline, depicted in Figure 6-8, consists of three steps:

1. **Virtual Environment**: We first create a simulation of a test environment and possible sensor components. Using RL techniques to train the navigation DNN, we iterate over the sensor architectures that produce the best performing control DNN.

2. **Real Environment Validation**: A larger quadcopter capable of carrying all sensors would then be set to match the optimized configuration found in the virtual environment. This test platform would contain its own navigation stack as a backup to intervene in case the test platform is at risk of a crash.

3. **Final Specification Manufacture**: Upon successful validation, a specialized smaller lightweight drone with only the necessary visual components and compute would be constructed. This design is well adapted for the given task, but less general purpose.

If the demonstration of this image architecture selection task is successful, there are a number of key metrics that would be interesting to investigate. First, the total power consumption based on the total weight and compute resources could be compared to the more general purpose navigation stack baseline. A related measure would be the speed with which the perception system could perform local navigation.

It would also be important to measure the number of optical design candidates proposed within each step of the design process. We should be able to evaluate orders of magnitude more candidates for the virtual environment than the real environment, such that the time spent validating designs in the world world should be reduced. The

exact trade-off between simulation performance and modeling compared to real-world validation would be interesting to understand how quickly and reliably a design could be produced.

6.6.4 Experiment 2: VDC for collision-free movement and image sensor

In the second experiment, we would optimize component graph consisting of an image sensor and VDC placed on top of the sensor. This experiment would follow similar steps and evaluation as the first experiment, but would focus on analyzing the patterns of the VDC that emerge in different navigation environments. As such, a fully trained VDC would be produced for a number of navigation environments: forest, indoors, and urban. We hypothesize that some features may be quite similar, while the exact VDC pattern may be different depending on the environment used for training.

In practice, we envision the final Robobee using a VDC as a single image sensor with a 3D printed cube place on top of it, where the cube was printed using 2 plastics: black and transparent. It is possible the optimized design could resemble a pinhole camera, regular pinholes near the sensor, or a more complex coded aperture. In interesting analysis would be to see how the optimized design changes with each iteration of the corresponding control DNN.

6.7 Discussion

6.7.1 Enabling Manufacturing Technologies

The emergence of 3D printing technologies for manufacture of high-performance mechanical parts (using materials such as metal, carbon composites, and plastics) has enabled a large design space of possible part designs. Design tools are increasingly used to propose parts that are stronger, and more lightweight than would be produced by a human designer. Often the proposed designs can only be produced by a 3D printer, but often the improved performance justifies an increased cost of 3D printed

parts.

Our VDC concept making use of more complex optical designs described above would require a 3D printer capable of producing different optical properties within a 3D volume. In particular, special care would need to be taken to ensure optically important boundaries—such as at index of refraction changes—are sufficiently smooth.

6.7.2 Nature's optical design grammar

Natural imaging system are encoded in DNA by a sequence of 4 possible nucleotides, which encode the amino acid sequence used to construct the proteins that are responsible for the shape and function of all of life's designs. Rather than encoding the optical properties directly, the resulting proteins encoded by DNA are more similar to a complex computer program capable of self-reference and recursion. When eyes are "manufactured" in nature, proteins are used as structural material, act as scaffolding, or even trigger the production of other proteins in an unfathomably complex process. This encoding is not only useful for describing the eye design morphology, but also because it appears simple perturbations of the DNA sequence can have complex effects on morphology, such as going from producing only one eye to two. It would be an interesting future direction to capture some advantages of this encoding using an artificial approximation.

While it is clearly infeasible to simulate protein interactions on the scale required to grow an artificial eye, perhaps some other efficient encoding of the desired morphology could be used. The deep image prior (DIP) appears to exhibit self-reference and recursion when generating images, and it may be interesting to consider an analogous approach for producing VDC. A neural network could be trained to output the desired optical properties given an input representing a 3D point within a volume. In this way, the neural network could be queried with enough 3D points to generate a VDC. An advantage of this approach is that the network could efficiently store things like repeating patterns or simple structures within its parameters that would otherwise require a large volumetric data-structure. As such, incremental changes to the network weights may produce more creative and dramatic leaps in design that would otherwise

not arise from perturbing the VDC parameters directly.

6.7.3 Making use of the environment

Another key insight from nature is to consider the environment an important part of the imaging system. While the environment can generally not be modified, it imposes constraints that might lead to efficiency gains, by obviating certain signals or introducing new "unusual cues" or signals that a given imaging system can use rather than directly observing the subject.

In this way, the design optimization should include a realistic environment that contains these cues. By synthetically evolving the structure and information processing in a realistic environment, the resulting design may learn to make use of these environmental cues.

6.7.4 Risks

Complexity of defining task objectives

Initial work would utilize simple task objectives—e.g. “travel fast without colliding with the environment”—that could be easily encoded in the simulation framework using standard reinforcement learning reward objectives. As tasks become more complex or nuanced, it may be useful to develop new ways of encoding task objectives.

Furthermore, there may be certain external considerations, such as power consumption and weight which may be incorporated into the full task objective in order to constrain solutions that accomplish the task objective. These design constraints may also act as regularization, selecting more simple designs that solve the task equally well.

Unfortunately, it is less clear how to take inspiration from nature for defining the task objectives. Natural selection for survival probably does not contain enough information to produce a useful perception system. An interesting way this could produce perception system designs is to create a simulation of “competitive” organisms capable of eliminating each other. In this way, good perception is required for a given

design to survive. It may be possible to define such a synthetic environment that creates a perception system that could be useful for more specialized tasks as well.

Combinatorial explosion of possible designs

One of the most challenging aspects of this proposed work is efficient sampling of component graphs defining new imaging systems. Auto-differentiation allows the parameters of these components (position, size, frequency, etc) to be differentiable with respect to some objective function. However, the ordering of the components into a graph is seemingly non-differentiable or even combinatorial. We believe ultimately this problem can be addressed by learning approximate "component selection functions" learned through experience.

AI safety and role of human designers

If such a design framework were wildly successful, there is a possibility designs could be produced that would be detrimental to human flourishing. It remains an interesting question if a general purpose design framework could incorporate high-level constraints that would prevent the automatic evolution of undesirable behavior. For example, the *Paper-clip Maximizer* by [16], is a well-known theoretical problem where a machine designed to produce as many paperclips as possible consumes all of Earth's resources in order to perfectly solve a given task.

While seemingly far-fetched, it may be useful to consider mitigation strategies that could include humans in the design process if only to ensure designs do not have unintended consequences. For example, human intervention might be useful to ensure perception systems do not unintentionally discriminate based on race, sex, or age in tasks that serve humans. Such risks are present in existing design methodologies, but a *computational discovery* paradigm might present new challenges as humans may not be able to easily analyze the consequence of design decisions.

6.8 Conclusion

We present a computational discovery framework that offers an exciting direction for perception system design. While previous methods have typically optimized parametric imaging modules for a specific task, we propose an additional architecture selection step that has the potential to generate novel imaging systems. We also propose a trainable imaging component we call the VDC, or volumetric design component, that could serve as a general purpose optical component and can be trained in an end-to-end manner. We hypothesize that the combination of the VDC, architecture selection, and differentiable simulation will enable a broad range of possible artificial perception systems that may even rival human designs.

We take inspiration from nature, where astounding optical designs have evolved from incremental improvements. There are some challenges remaining to ensure such a process is computationally tractable and task objectives can be well defined. Despite these hurdles, we believe an initial demonstration of computational discovery for novel imaging platforms is within reach.

Chapter 7

Conclusion and Future Work

7.1 Goals and Research Questions

7.1.1 Exploiting hidden cues in photographs

In this thesis, we demonstrated a number of ways to exploit hidden cues in photographs. We made use of these cues to localize hidden objects and reconstruct images of a scene outside the line of sight of the camera.

What hidden cues are in photographs?

Hidden cues in photographs are small changes to the observed image that indicate changes in the hidden scene. The most common of these cues is cast shadows, which this thesis made extensive use of. There are a number of other cues, such as specular reflections, that can provide very useful cues in certain scenes. The various methodology we explored provides a path to automatic discovery of hidden cues.

When and how can we localize hidden objects outside the field of view?

We showed how to localize objects outside the field of view of a camera using data-driven and model-based approaches. We also provide analysis of the types of scenes that are amenable to localizing hidden objects outside the line of sight.

Paradigm \ Challenge	Computational Complexity	Model Mismatch	Generalization	Priors	Confidence in Solution
Physics Based	- approximations - can be intractable	- calibration - poor noise models + differentiability	+ "extrapolating" + inductive	- simplistic + fundamental constraints	+ detailed theory + data consistency + sensitivity analysis
Data Driven	+ fixed complexity	- data dependent + adaptive + handles noise	- "interpolating" - requires many input-output pairs	+ highly flexible + expressive - domain restricted	- nascent theory - hallucinations

Figure 7-1: Physics-based and data-driven approaches to solving inverse problems are sometimes considered incompatible, but are largely complementary. Physics-based, or model-based, approaches rely on models that are generated using induction, making their generalization excellent, but may be highly sensitive to model parameter error and require extensive calibration. Data-driven methods are typically excellent at interpolating values within the training data domain, but do not make use of epistemological priors such as Occam's razor, potentially producing models that are over parameterized and/or produce hallucinations.

How can machine-learning and physics-based approaches be used?

One of the themes of this work was to explore both data-driven and model-based approaches for making use of hidden cues. Through the process of implementing these various approaches, we observed some common patterns that provide insight into their relative strengths, and sought to combine them when possible. We found that more tightly coupled integration of these approaches is an exciting direction for future work. We summarize these complementary approaches below.

In the follow sections, we consider the use of the two paradigms for solving inverse problems like those investigated in this thesis.[1]

Computational Complexity Physics-based models are highly accurate, but often are computationally expensive, particularly when using them to solve an inverse problem. In order to make the problem tractable, physics-based models often make approximations which might greatly improve their efficiency, but require additional assumptions or restrictions on the scene.

In contrast, data-driven approaches have fixed complexity once they have been

[1]Problems like object classification are difficult to compare to physics-based approaches, and data-driven methods are clearly the only viable option.

trained. Neural networks are surprisingly good at modeling complex phenomena when sufficient input-output pairs are available. The primary advantage of data-driven approaches is their flexibility. While physics-based approaches typically achieve computational efficiency by restricting the problem domain, data-driven approaches can be used in a large variety of situations.

Model Mismatch Physics-based models can be very accurate, but often fall short in real-world systems that contain sources of noise that is difficult to model. Furthermore, the accuracy of these models often depend on precise calibration and as such can be "brittle" to miscalibration or model uncertainty.

Data-driven methods naturally adapt to the statistics of the data they are trained on. This typically makes data-driven methods good at handling real-world sources of noise that are difficult to model using physics-based methods. Furthermore, domain-randomization techniques provide a simple method to ensure the trained systems are less susceptible to miscalibration type effects. That said, we consider generalizability separately—if the data at test time is significantly different than that used in training, data-driven methods generally do not perform well.

Generalization Perhaps the most powerful aspect of physics-based models is their incredible generalization. Assuming they make use of appropriate approximations, physical models should work over a wide range of inputs.

A big challenge for data-driven methods is ensuring generalization. Typically, general function approximators like neural networks perform excellent interpolation, but struggle when extrapolating values outside the domain of their training data. For this reason, domain-randomization can make these models reasonably robust to model-mismatch, but are not effective at describing a system entirely, like physics-based models.

Priors Priors in physics-based models are typically quite basic, consisting of things like minimum norm, smoothness, or sparsity (e.g. total variation). This can be

attractive in the sense that it does not bias the solution towards hallucination, but does not leverage the richness of the prior distribution in many real problems.

Data-driven solutions are reasonably well suited for defining priors if sufficient training data is available. These priors can often be implemented in two ways. First, one could train a generative model as a prior. These approaches loosely restrict the solution to some high-likelihood manifold. The second approach is to train a discriminator to score the likelihood of various solutions, effectively modeling the prior distribution directly. While these approaches can be extremely powerful, they require high-enough capacity and sufficient training data to model the prior distribution.

Confidence in Solution Physics-based models are trustworthy: if a proposed solution is close to the observed measurements after passing through the forward model, we can at least ensure the solution is consistent. For some situations, there may still be an infinite number of possible solutions, but data consistency is powerful evidence that a solution is reasonable.

In contrast, purely data-driven approaches do not guarantee data consistency. That said, given a set of solutions that are data-consistent, data-driven methods may provide better priors. A key part of solution confidence is how this information is communicated to the end user. Neural networks have a reputation for "hallucinating" parts of a solution even if the solution achieve data-consistency. If a confidence map or other visualization of the parts of a solution that are hallucinated by a model were presented to a user, it may alleviate some of these problems.

7.1.2 Automatically discovering hidden cues

What are the opportunities when developing imaging systems that make use of hidden cues?

One lesson learned throughout the work presented as part of this thesis is the importance of understanding model uncertainty. Imaging systems that obtain more information about the visible scene, and thus provide better certainty about the for-

ward model, should be better suited to take advantage of hidden cues. Such imaging systems may probe the visible scene or adaptively capture additional measurements. This additional information would be invaluable when solving the inverse problem.

What are the design principles that could be used to develop imaging systems that make use of hidden cues in the environment?

Designing imaging systems in an end-to-end manner may produce novel optical designs that are well suited to take advantage of hidden cues to solve certain tasks. Most imaging systems trained end-to-end update a fixed set of parameters as specified by the meta-designer. However, we may be able to make use of differentiable volumetric renderers to develop more more general designs. This approach would still optimize over a fixed set of parameters, but could produce a large variety of possible imaging system designs more similar in variety and effectiveness to those produced in nature. Chapter 6 describes this idea as a "Volumetric Design Component" and is an exciting area for future work.

7.2 Overview of Contributions

- We show that under certain assumptions, point-light localization can be used to localize moving objects outside the line-of-sight by treating them as differential illumination sources. We describe the geometry that fundamentally limits localization using shadow edges and relate this to prior work.

- A data-driven method to localize moving objects outside the line-of-sight when only partial information is known about the forward model. We show how this trained neural network can be used to discover hidden cues present in certain environments.

- Given mild assumptions, we show how an image of shadows cast by objects can be used to reconstruct a viewpoint of the scene from the perspective of the object, even when the surrounding surface albedo is unknown.

- Using the resulting inverse operator for estimating incident illumination, we describe a method for highlighting regions of the image that potentially contain hidden cues.

- We show how the respective inverse problems can be made more robust by handling uncertain model parameters using a sampling based approach.

- Propose an approach for the development of imaging systems that utilize hidden cues and are well adapted for a given environment.

7.2.1 Relevant Papers and Presentations

Peer Reviewed Papers

- T Swedish, C Henley, R Raskar. *Objects as Cameras: Estimating High-Frequency Illumination from Shadows.* ICCV, 2021.

- C Henley, T Maeda, T Swedish, R Raskar. *Imaging Behind Occluders Using Two-Bounce Light* European Conference on Computer Vision (ECCV), 573-588, 2020.

- T Swedish and R Raskar. *Deep visual teach and repeat on path networks.* CVPR Workshops, 2018.

White Papers (Arxiv and Non-peer Reviewed Conferences)

- T Swedish, C Henley, R Raskar. *Constraining light source localization using visible occlusion boundaries.* arXiv (to be uploaded), 2022.

- T Swedish, G Satat, R Raskar. *Learning cues to locate hidden objects.* arXiv (to be uploaded), 2022.

- SC Sadhu, A Singh, T Maeda, T Swedish, R Kim, L Sinha, R Raskar. *Automatic calibration of time of flight based non-line-of-sight reconstruction.* arXiv:2105.10603, 2021.

- T Maeda, G Satat, T Swedish, L Sinha, R Raskar. *Recent advances in imaging around corners.* arXiv:1910.05613, 2019.

- M Tancik, T Swedish, G Satat, R Raskar. *Data-driven non-line-of-sight imaging with a traditional camera*, COSI, IW2B 6, 2018.

Presentations and Conference Courses

- R Raskar, A Velten, S Bauer, T Swedish. *Seeing around corners using time of flight.* ACM SIGGRAPH Courses, 1-97, 2020.

- G Satat, T Swedish, V Boominathan, A Veeraraghavan, R Raskar. *Data Driven Computational Imaging.* CVPR Courses, 2019.

7.2.2 Software Implementations

- Probabilistic Rendering Framework for Synthetic Training Data (Blender + Python)

- A Simple Differentiable Renderer for Cast Shadows (Python)

- Variational Optimization Library (Python)

- A Volumetric Renderer with Surface Reflectance Phase Function (Python / C++)

- A direct time-of-flight renderer (Python)

7.2.3 Miscellaneous Contributions

- *Demo* DARPA REVEAL Program: Real-time localization of Hidden Object

- *Blog* "Automatic Differentiation from Scratch: Forward and Reverse Modes". https://medium.com/camera-culture/automatic-differentiation-from-scratch-forward-and-reverse-modes-2dcdb9be8cb

7.3 Future Work

7.3.1 Summary from Chapters 3-6

We proposed a number of exciting areas for future work at the end of Chapters 3-6. In this section, we summarize these ideas.

Combining data-driven and model-based approaches Physics-based and data-driven methods are largely complementary, and more tightly integrating them is an clear next step throughout this work. A broad set of approaches for incorporating data-driven priors should improve many of the reconstruction results shown in this thesis. To this end, a key challenge is finding sufficient training data to learn these priors.

One exciting area is leveraging physics-based models to verify solutions provided by data-driven approaches are physically plausible. In particular, ensuring data-consistency while making use of data-driven priors would provide clear benefits. The current approach in the literature roughly follows that of [25], where a neural network might propose solutions that are then verified by a physics-based forward model. However, this general approach does not directly leverage the ability of data-driven approaches to handle model mismatch. In other words, understanding and visualizing the uncertainty in model parameters in addition to the unknowns would be broadly useful.

Another idea would be to use data-driven methods to learn physical laws directly. A possible benefit would be to learn extremely efficient but reliable approximations of computationally intensive forward models. Utilizing neural networks to interpolate the action of physics-based models, or "lifting" discrete matrix based models to a continuous domain may be one way to achieve this.

Variational Optimization Perhaps a hidden theme of this thesis is the benefit of auto-differentiation for many inverse problems. Unfortunately, discontinuous functions do not have a well defined derivative, and flat areas in an objective can produce

zero-valued gradients. Variational optimization is a sampling based approach for approximating the value of functions that may not be continuous or smooth.

It would be interesting to combine auto-differentiation and variational optimization to perform gradient descent in inverse problems like reconstructing a hidden scene when the model parameters are unknown. In this way, variational optimization could be used to estimate the gradient updates, but also the local curvature or Hessian, of the gradient updates when there is significant uncertainty about the model parameters.

Efficient discretization of the hidden scene It is often required we discretize the forward model when solving an inverse problem. We are often given a discretization of our observations as an image of pixels, but the hidden scene could be subdivided in many different ways. Coming up with automatic and efficient ways to discretize the hidden scene volume may be informed by understanding what cues in the visible scene are possible and how they change with perturbations in the hidden scene.

One way to accomplish this with a differentiable renderer is to calculate the inverse of $J^\top J$, where J is the Jacobian of the image pixels with respect to the location of a point source at a particular choice of coordinates in the hidden scene. In effect, $(J^\top J)^{-1}$ would encode the change in image pixels values when the hidden scene coordinate position changes as a sort of measure of resolution. Given the expected noise of the observed images, one could use this information to sample the hidden scene at discrete points such that each discrete location in the hidden scene would be distinct enough to produce an observable change in the image.

Differentiable forward models Differentiability has become increasingly practical for complex forward models using modern autodifferentiation techniques. Simply applying gradient descent to solve inverse problems is only one application of differentiability. Differentiable forward models could be used to train approximate data-driven models, or used in higher order optimization methods like Gauss-Newton, sensitivity analysis, or to add regularization under model uncertainty.

Updating the ray transport matrix and handling uncertainty Perhaps the most direct extension in Chapter 5 is to add regularization that handles model uncertainty. The sketch of the approach provided in Chapter 5 might be broadly useful and can be implemented using sampling or a differentiable forward model.

This line of work is exciting, as it provides a nice framework for leveraging differentiable forward models to solve inverse problems in imaging. This is highly related to the next area for future work.

Approximating the continuous inverse operator with neural networks Imaging problems can often be best analyzed in terms in continuous space—an integral over some domain. The light transport matrix is a discretization of the continuous light transport operator. It would be interesting to "lift" the light transport matrix and solve for the continuous inverse operator. This may enable better analysis of things like the expected recoverable resolution of hidden scenes. One tool for enabling this is training a neural network to approximate the continuous operator.

7.3.2 Updating the Model Matrix

While we propose some methods of handling model mismatch and model uncertainty in Chapter 5, it still remains a very difficult problem. In particular, if little is known about the scene, the space of possible models and their parameters, θ, is quite large. We have mostly assumed that a model and a suitable parameterization has already been chosen. For example, in Mitsuba2, parameters are defined by an initial scene when is composed of mesh vertices, textures, and light sources. If we are unable to estimate any of these scene properties to provide a reasonable initialization, the space of possible Mitsuba2 scenes is practically infinite. Recently, more general and expressive scene parameterizations, such as voxel-based phase functions or radiant fields, provide a possible path forward to ensure the model parameterization is fully general. These models still have challenges, as they are often over-parameterized and memory-intensive.

Regardless, there are a few concepts and ideas that might be able to expand on

the linearization described in this thesis.

7.3.3 Linearization and tangent spaces of light transport

We often assume we have some model matrix that maps unknowns to our observations. As we are interested in visualizing the hidden scene radiance, this model is linear. For n unknowns and m observations, all possible model matrices are in the set $R^{m \times n}$. This set of possible models is enormous, and so we can restrict ourselves to a "physical model manifold" that is embedded in this set. To extend the idea of parameterization by θ, the set of model matrices on this manifold could be indexed by the set of all θ that are physically possible. Figure 7-2 shows this idea visually.

When solving for our unknowns, x, we may also jointly solve for model parameters, θ, as was done in Chapter 5. There is often a non-linear relationship between the model and its parameters θ. As such, a model is often linearized to account for perturbations in the non-linear model parameters. The linear model with a chosen θ can be thought of as linearization points of the set of physics-based models for the hidden scene reconstruction problem.

Inspiration: Lie Algebra of SO(3) Differentiable renderers provide linearization at given points along the physics-based model manifold. We can use Taylor approximation to create an approximate forward model that is perturbed by a given parameters: $A_\phi \approx A_\theta + (\phi - \theta)A'_\theta$. The problem is that even a small perturbation $(\phi - \theta)$ may change the entries of A_ϕ such that it deviates from the physics-based model manifold.

An inspiration for handling this problem could be the Lie Algebra of SO(3), which is commonly used to generate updates for rotation matrices in bundle adjustment. Gradients can be calculated for the entries of a rotation matrix, but updated matrices must be projected back into SO(3). Projection back onto the manifold is not ideal, as higher-order optimization is made more difficult. It turns out there is a much better way to perform these updates using the matrix exponential map. While I only provide an intuitive sketch, the exponential map embeds SO(3) in a vector space—the tangent space of SO(3) at the identity. Updates to points in this vector space can then be

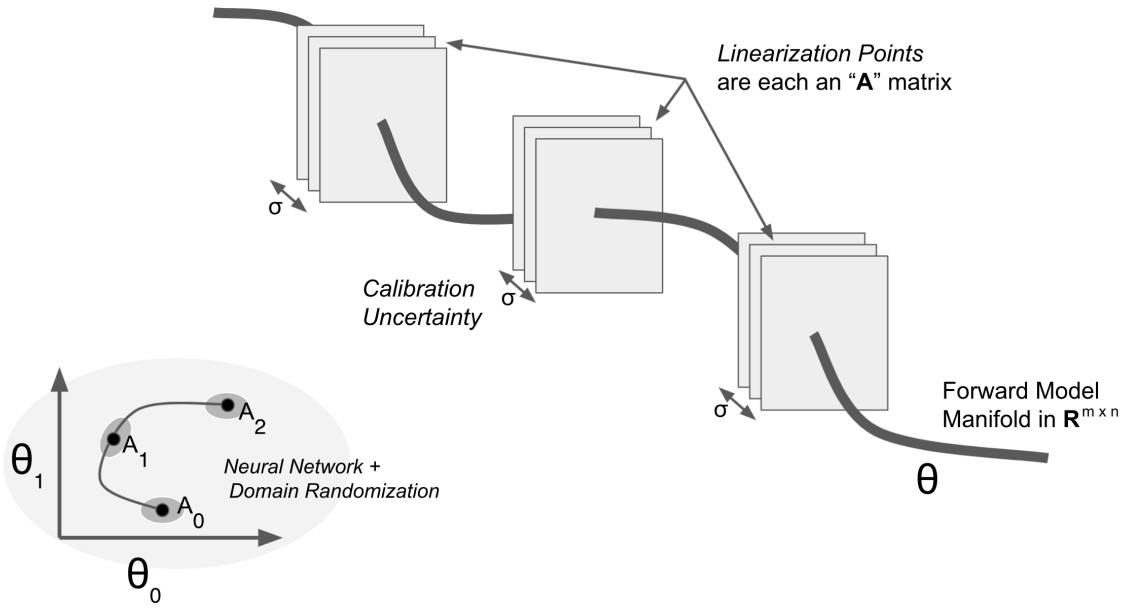


Figure 7-2: Light transport can be described by a linear map from illumination source to observed image. The set of *physically-plausible* linear maps are the subset of all possible linear maps that can be modeled using a physically-based renderer. This subset is at least locally continuous for perturbations of scene parameters that can describe the rendering of a particular scene: changes to material reflectance, geometry, and camera parameters. Loosely, we refer to this as the "Forward Model Manifold." A data-driven model does not have a notion of physical plausibility, but may learn an approximate map from observed image to unknown model parameters given sufficient training data.

mapped back to SO(3) through the logarithmic map.

For perturbations to our physics-based model, is there something like the matrix exponential and logarithmic map in SO(3)?

Data Driven: Domain Randomization

In the data driven case, we can train a neural network by collecting samples generated by a variety points along the physics-based model manifold. In our case, we define a parametric model of the scene geometry, materials, camera, and lighting, and generate a wide variety of possible scenes. We then render input-output pairs for these sampled instances of forward models.

Interestingly, data driven models such as neural networks seem to be robust over a

fairly large domain of θ values when a sufficiently diverse data set is available. For models trained to solve inverse problems, they are explicitly trained to recover x from y, and implicitly must handle the unknown θ in some way. In practice, it is sometimes helpful to add an auxiliary objective at training time such that the network is explicitly trained to estimate θ and x. This training often improves performance, suggesting that the network benefits from some prior over θ to make its estimate.

7.3.4 Types of non-linear effects

In the hidden scene recovery problem, there are a number of non-linear effects that can not be simply incorporated into the linear forward model. Since we are estimating illumination, which is linear, the non-linear effects appear for geometric perturbations and material reflectance changes.

Geometric Perturbations

Geometric perturbations include things like camera motion and changes to the scene geometry such as the vertex position of meshes in the scene. The image gradients for these perturbations has traditionally been difficult to calculate, but recent work has made the calculation of these gradients possible for general purpose path tracers [85, 65].

Optical Flow In practice, the gradients for geometric perturbations often appear like the gradients used to warp an image from optical flow. For example, a black square moving to the right in front of a white background could be warped using a horizontal gradient kernel $I_{t+1} = I_t + [1, 0, -1] * I_t = I_t + I'_t$. If we repeatedly apply this equation, we should see the black square move across the screen from left to right in a smooth motion. In essence, this equation describes a differential equation that updates the position of the square. We calculate the new image by integrating this differential equation for a period of time.

Lucas-Kanade Similarly, a differentiable renderer can provide a gradient at a given linearization point I_t, however it does not describe a suitable differential equation because the update to the image is spatially varied and changes over time. However, if we *infer* the optical flow of the update in screen space, we have a suitable differential equation. This can be accomplished using the Lucas-Kanade (LK) method [73], since the gradient image provided by the differential renderer provides us with the image difference, traditionally I'_t, used in LK methods. We can then warp the image by scaling the optical flow, assuming the screen space motion velocity is constant and we handle the case of low-texture regions—e.g. with total variation regularization.

If we apply these optical flow updates to each column of the linear model matrix A, we can update the model matrix over a wider range of geometric changes. This approximation is not perfect, but may keep the updated model closer to the physics-based manifold. For example, a differentiable renderer could easily return the local gradients for the black square example with respect to a camera panning left. The gradient image could be used to derive the implied optical flow, and the optical flow could then be scaled for much larger pans, while maintaining the appearance of the square moving to the right.

Material Properties

Material properties are another type of parameter that is non-linear with respect to the model matrix. For example, changing the roughness of a microfacet reflectance model will lead to quickly sharpening specular highlights. A basic specular reflection model that has this property but is not physically based is the Phong reflection model [91]. The key part of this effect is that the cosine terms for specular paths are exponentiated to create sharp specular highlights. The specular highlights can be sharpened by increasing this exponentiation factor.

Exponentiation In a similar fashion to using optical flow to stay close to the physics-based model manifold, we may be able to define a differential equation to describe larger changes in parameters that effect the specular reflection of materials.

A differentiable renderer can provide us with the local gradient of a specular reflection with respect to a material property parameter. As such, we may be able to raise this gradient to some power to simulate the effect of increasing ($\gamma > 1$) or decreasing ($\gamma < 1$) the specular reflectance. This is an idea that warrants a more rigorous treatment, solving the implied differential equation using the known reflectance model may be a more general approach.

In fact, it may be possible to assume independence between material property parameters in pixel space, and a differential renderer may provide higher order terms to the Taylor approximation. Independence assumptions do not hold for multi-bounce light paths, however, so in general higher order terms would introduce higher order tensors to approximate the full renderer, which will quickly become impractical.

When to perform re-linearization

While we may be able to extend the region around the current estimate of θ using the techniques described above in order to keep A close to the physics-based model manifold, it will be necessary to select a new linearization point and re-run the forward model and the corresponding derivatives to ensure the model remains physically plausible.

7.4 Conclusion

Images of real-world scenes are the result of a complex interaction of light with the environment before being scattered towards a camera lens. For the purpose of image formation, the computer graphics community has enabled the production of physically-based photo-realistic renders that are consistent with real captured photographs. These renders can be parameterized by scene parameters such as surface geometry and reflectance, and the intensity of a number of illumination sources.

In parallel, the field of inverse graphics has blossomed in recent years as differentiable models have enabled the optimization of scene parameters for renders that match captured photographs. It is likely that this field will continue to "solve for"

the most likely configuration of real-world scene parameters that produced a given photograph. Since light that forms an image is a result of many scattering events outside the field of view of a particular camera, a complete solution for a scene will certainly include information about the scene outside the camera's line-of-sight.

While inverse graphics is an essential part of this work, simply solving for a particular scene does not directly teach us about the principles or patterns that can enable rapid inference of hidden information in general. As such, this thesis explores the idea of the discovery of common cues, such as shadows. While the ideas and techniques discussed in this work might inspire improvements to inverse rendering, such cues may also help inform the design of future perception systems. In a similar fashion, this work could inspire the design of objects or environments that either make inverse rendering easier or more difficult, depending on the application.

Appendix A

Minimizing the Expected Error with Model Uncertainty

Let us assume there is a linear map that describes our forward model, discretized such that we represent it by the matrix A. We generate this linear map using a renderer as described in Chapter 5, with scene parameters θ. In this way, for each set of parameters, θ, we have a corresponding ray transport matrix, A_θ. Adjusting parameters such as position of mesh vertices or material properties will often produce different A_θ.

Given this setting we want to minimize the expected mean squared error to solve for x.

$$\arg\min_x \mathbb{E}_\theta[||A_\theta x - y||_2^2 + \lambda\Gamma(x)] \tag{A.1}$$

Where $\Gamma(\cdot)$ is a regularization/penalty function with regularization factor, λ.

A.1 Sampling Approach

For example, we assume θ are independently drawn from a Gaussian distribution:

$$\theta_i \sim \mathcal{N}(\theta, \mathrm{diag}(\sigma^2)) \tag{A.2}$$

Now we can draw samples of A_θ by drawing N samples, θ_i, and rendering the corresponding ray transport matrix, A_{θ_i}, for each. In order to minimize the expected error, we have the following objective:

$$\arg\min_x \frac{1}{\sum_{i=0}^{N} p(\theta_i)} \sum_{i=0}^{N} p(\theta_i)(||A_{\theta_i}x - y||_2^2 + \lambda\Gamma(x)) \tag{A.3}$$

The constant normalization factor does not change the minimum, and can be removed. By writing out the squared term and setting the derivative with respect to zero, we obtain the following:

$$\frac{\partial}{\partial x}\left(\sum_{i=0}^{N} p(\theta_i)\left[\lambda\Gamma(x) + xA_{\theta_i}^\top A_{\theta_i}x - 2x^\top A_{\theta_i}^\top y + y^\top y\right]\right) = 0 \tag{A.4}$$

$$\frac{\partial}{\partial x}\lambda\Gamma(x) + 2\left[\sum_{i=0}^{N} p(\theta_i)A_{\theta_i}^\top A_{\theta_i}\right]x = 2\left[\sum_{i=0}^{N} p(\theta_i)A_{\theta_i}^\top\right]y \tag{A.5}$$

Where for large enough samples, $\sum_{i=0}^{N} p(\theta_i) = 1$. If $\Gamma(x)$ is a linear function, we can use Tikonov regulation, $||\Gamma x||_2^2$, and with some rearranging, we can write:

$$\left(\left[\sum_{i=0}^{N} p(\theta_i)(A_{\theta_i}^\top A_{\theta_i})\right] + \lambda\Gamma^\top\Gamma\right)x = \left[\sum_{i=0}^{N} p(\theta_i)A_{\theta_i}^\top\right]y \tag{A.6}$$

Therefore, by rendering to obtain many A_{θ_i}, we can generate a solution for x by solving the above linear system.

A.2 Taylor Approximation Approach

Another idea is to assume we can calculate $\frac{\partial}{\partial\theta}A_\theta$ using a differentiable renderer, such that for small perturbation of θ, we can approximate all A_θ around some initial parameter $\theta_{i,0} = \phi_i$:

$$A_{\theta_i} \approx A_{\phi_i} + (\theta_i - \phi_i)A'_{\phi_i} \tag{A.7}$$

For all parameters (over i), θ_i, where $A'_{\phi_i} = \frac{\partial}{\partial\theta_i}A_{\phi_i}$, the partial derivative of A with

respect to some model parameter θ_i, around an initial value ϕ_i. We assume model parameters are independent over i, so we omit the subscript in the following analysis, however, this approach can be applied to each model parameter θ_i.

Now, we solve for the above expectation more efficiently, assuming some "perturbation distribution" around ϕ. We define a Gaussian random variable, $\theta_\epsilon = \theta - \phi$, which is centered around ϕ, such that $\theta_\epsilon \sim \mathcal{N}(0, \sigma_\theta^2)$. Now we write Equation A.3 by using the approximation in Equation A.7, omitting the regularization term for brevity:

$$\arg\min_x \frac{1}{\sum_{i=0}^{N} p(\theta_i)} \sum_{i=0}^{N} p(\theta_i)(||A_{\phi_i} + (\theta_i - \phi_i)A'_{\phi_i}x - y||_2^2 \tag{A.8}$$

Using the same approach as above, by setting the gradient of this objective equal to zero, we obtain the following, noting that $E[\theta_\epsilon] = 0$ and $E[{\theta_\epsilon}^2] = \sigma_\theta^2$:

$$\left(A_\phi^\top A_\phi + \left[\sum_{i=0}^{N} \sigma_i^2 ({A'_{\phi_i}}^\top A'_{\phi_i})\right]\right) x = A_\phi^\top y \tag{A.9}$$

Where σ_i^2 is equal to the i-th σ_θ^2 term. This form is much simpler than the more general version in Equation A.6, but is only valid for small perturbations (small σ_θ). Regardless, it reveals that the Jacobian of the forward model with respect to model parameters can be used to introduce a regularization term that should make the solution more robust to uncertainty. Considering that the resulting inverse will weight the "inverse Gramm gradient matrices", $({A'_{\phi_i}}^\top A'_{\phi_i})^{-1}$, proportional $\frac{1}{\sigma_\theta^2}$, the solution makes intuitive sense. When uncertainty is high about a parameter, pixels in x that are most affected by changes in that parameter are weighted less.

A.3 Non-linear Least Squares to update θ

We can also solve for θ_i directly. Rewriting Equation A.8 to solve over a vector of parameters, θ for a given x, we notice that this reveals a non-linear least squares problem. Given a vector of uncertainties, σ_θ, we can add a weighting term:

$$\left[{T_x}'^\top \text{diag}\left(\frac{1}{\sigma_\theta^2}\right) T_x'\right] \theta_\epsilon = {T_x}'^\top \text{diag}\left(\frac{1}{\sigma_\theta^2}\right) (y - Ax) \tag{A.10}$$

Here, we've abused our notation to interpret θ_ϵ as a vector. We also write ${T_x}'$ to represent a matrix where each column is the per-pixel gradient for each parameter θ_i: $A'_{\phi,i}x$. Note that this makes ${T_x}'$ a Jacobian-vector product, which makes it well suited to be computed using a differentiable renderer directly instead of computing each $A'_{\phi,i}$ matrix and computing the product with x.

Since this approximation is only valid for small perturbations, this update should be scaled by an appropriate learning rate ($<< 1$). As such, this approach is a means of performing gradient descent on θ that is weighted by a Gauss-Newton term that incorporates knowledge about the model parameter uncertainty and current estimate, x.

In this way, θ and x could be updated in an alternating fashion similar to the approach used in Chapter 5 that updated the albedo of the scene and environment map. Furthermore, as for the albedo, it is likely that variation in the environment map would better stabilize the estimate of θ, and the update described in Equation 5.6 could be applied. The parameters we optimize for, θ, could include scene albedo, material reflectance, or position of objects.

Appendix B

Learning Linear Models from Data

One of the most widespread problems in engineering is the linear inverse mapping, where we estimate some latent value x given observations b and a linear measurement model A:

$$Ax = b \tag{B.1}$$

In general, A is not a square matrix, so this problem can be posed using the *Normal Equation*.

$$(A^\top A)x = A^\top b \tag{B.2}$$

B.1 Model Driven Linear Inverse Problem

When the matrix A is large, solving Equation B.1 directly using a closed-form solution can become intractable. Instead, we can solve Equation B.1 using optimization, with many iterative algorithms to choose from, such as gradient descent. The basic idea is to update an initial guess of x in a way that minimizes the difference between Ax and b under some norm (e.g. L_2). We can also include a regularization term, $\Gamma(x)$, to weight more likely values of x.:

$$\hat{x} = \arg\min_x ||Ax - b||_2^2 + \lambda\Gamma(x) \tag{B.3}$$

Using modern deep learning frameworks, even if A is large, we can define a loss from Equation B.3 and perform stochastic gradient descent to calculate gradients for only a batched subset of rows of A and b. In fact, as long as some function f is differentiable (e.g. a neural network), we can solve

$$\hat{x} = \arg\min_x \sum_m ||f_m(x) - b_m||_2^2 + \lambda\Gamma(x) \tag{B.4}$$

, where f_m is some function that mimics the action of row, A_m, on x. How do we ensure *f mimics the action of A*? We write f as a Taylor Series:

$$f(x) = f(a) + f'(x - a) = Aa + A(x - a) \tag{B.5}$$

$$f'(x) = A \tag{B.6}$$

If we want a function, f, that mimics the transformation A, the gradient of f with respect to the input should be, $\nabla_x A_m x = A_m$, where A_m is the mth row of $m \times n$ matrix A.

B.1.1 Learning an Approximate Inverse

In order to find such a function, we define g_θ as some linear neural network and define a new optimization problem:

$$g_\theta^\star = \arg\min_\theta \sum_m ||\nabla_a g_{\theta,m}(a) - A_m||_2^2 \tag{B.7}$$

Using a deep learning framework, we can construct g_θ using linear operations (e.g. linear layers, convolutions, einsums) without nonlinear activation functions or affine bias. Furthermore, since g_θ is a linear function we can set a to any convenient input, such as a vector of zeros.

Due to the architecture of the computational graph defining g_θ, it may be an approximation of the action of A on x, but in some special cases, a g_θ with much fewer parameters (e.g. if A performs a convolution) may be a perfectly good representation of A. Intuitively, we use the rows of A as training data to learn the network g_θ.

Our ultimate goal is to approximate f using a computationally efficient g_θ, thus making Equation B.4 tractable for large problems without losing the physical constraints imposed by A. Therefore, we aim to solve Equation B.3 as:

$$\hat{x} = \arg\min_x \sum_m ||g^\star_{\theta,m}(x) - b_m||_2^2 + \lambda\Gamma(x) \tag{B.8}$$

We note that the architecture of g_θ given a class of matrices A could be automated using neural architecture search (i.e. AutoML), discovering the efficient, often lower dimension, latent structure imposed by the physics of the forward model used to create A.

B.1.2 Extension to Nonlinear Forward Models

We note that this general idea can be applied to any analytic function, not just linear functions, but Equation B.7 would require higher order derivative terms or multiple well chosen a as linearization points.

B.1.3 Comparison to Supervised Learning

We could instead train g_θ using supervised data pairs (x, b). We show that for many model types, models trained in this way are also reasonable solutions to Equation B.7. Using supervised data pairs enables the implicit learning of priors on the input data distribution.

Instead, we propose to sample from the Jacobian of the forward model to serve as the loss function used to train the approximate map. This map is exact for linear models, and in general could be extended to higher order functions by taking higher order derivatives (e.g. fitting the Hessian for quadratic functions).

B.1.4 Learning Linear Maps

The main idea is to use "model action" (via Taylor Series) as a way to create a scalar loss to compare two models. We want a way to compare an arbitrary neural network architecture with the physics based model and ensure they end up doing the same thing to all inputs. This appraoch does note require any training (input/output) pairs. In other words, the Taylor Series creates a standard representation for how input is mapped to output, and thus allows us to compare models via this representation.

B.2 Results

B.2.1 Frontoparallel Ray Transport Matrix

For a pair of frontoparallel planes, x and y, we consider two multiplicative masks between the planes. The input plane, x, emits light in all directions, the two masks block some of these rays, and the observed irradiance pattern on plane y can be described by the light transport tensor, T, which encodes how the two intermediate masks block rays. Writing this using Einstien summation:

$$T_{k,l,i,j}x_{i,j} = y_{k,l} \tag{B.9}$$

Each local patch (i, j) on the 2D plane x contributes to the image formed on plane y, as determined by T. In this case, T is identical to matrix A in Equation B.1 if it were reshaped into a $(k \cdot l) \times (i \cdot j)$ matrix.

We define 3-layer linear neural network with input size $(i \cdot j)$, a hidden layer with 2 units, and a $(k \cdot l)$ size output. The total number of parameters of this network is $2(i \cdot j + \cdot k \cdot l)$, a significant factor reduction from the $i \cdot j \cdot k \cdot l$ entries in T. We fit this randomly initialized network using the objective in Equation B.7, using PyTorch Autograd to compute the gradient for each output pixel, $y_{k,l}$, with respect to the input. Computing the gradient for every pixel of output y generates the Jacobian of the model mapping x to y.

Deep learning libraries like PyTorch typically use reverse mode auto-differentiation,

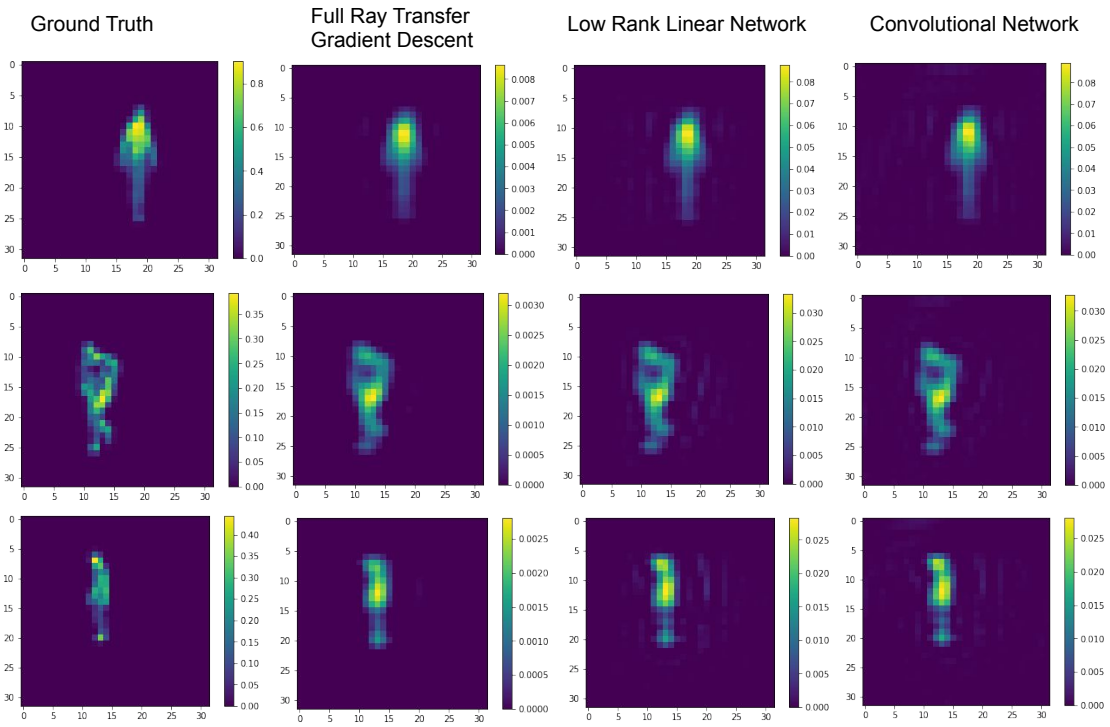


Figure B-1: Reconstruction Results

we effectively calculate the Jacobian by performing backpropagation with respect to each output pixel. If the Jacobian is too large to fit into memory, we can perform batched updates to θ.

We then use PyTorch to update the model weights, θ, of g_θ using the loss function in Equation B.7, using the corresponding (k', l') to represent row A_m as $T_{k',l',i,j}$.

Figure B-3 shows the gradient of two different positions of the output with respect to the input for the ground truth ray transfer matrix, a low rank, and convolutional neural network. This demonstrates that the networks learn to map x to b as well as A (up to a scaling factor). This ray transfer matrix is a challenging deconvolution and reconstructions using all architectures are comparable.

Figure B-1 shows reconstruction results for the full ray transfer matrix and two approximate networks.

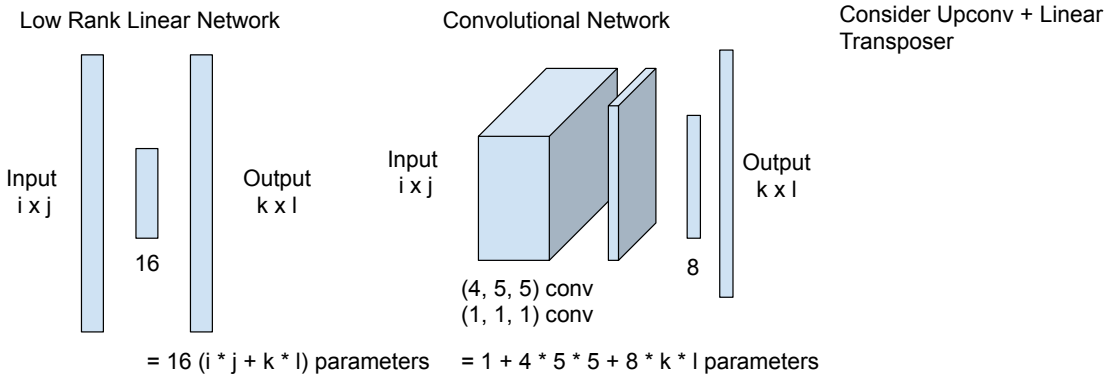


Figure B-2: Network Architecture

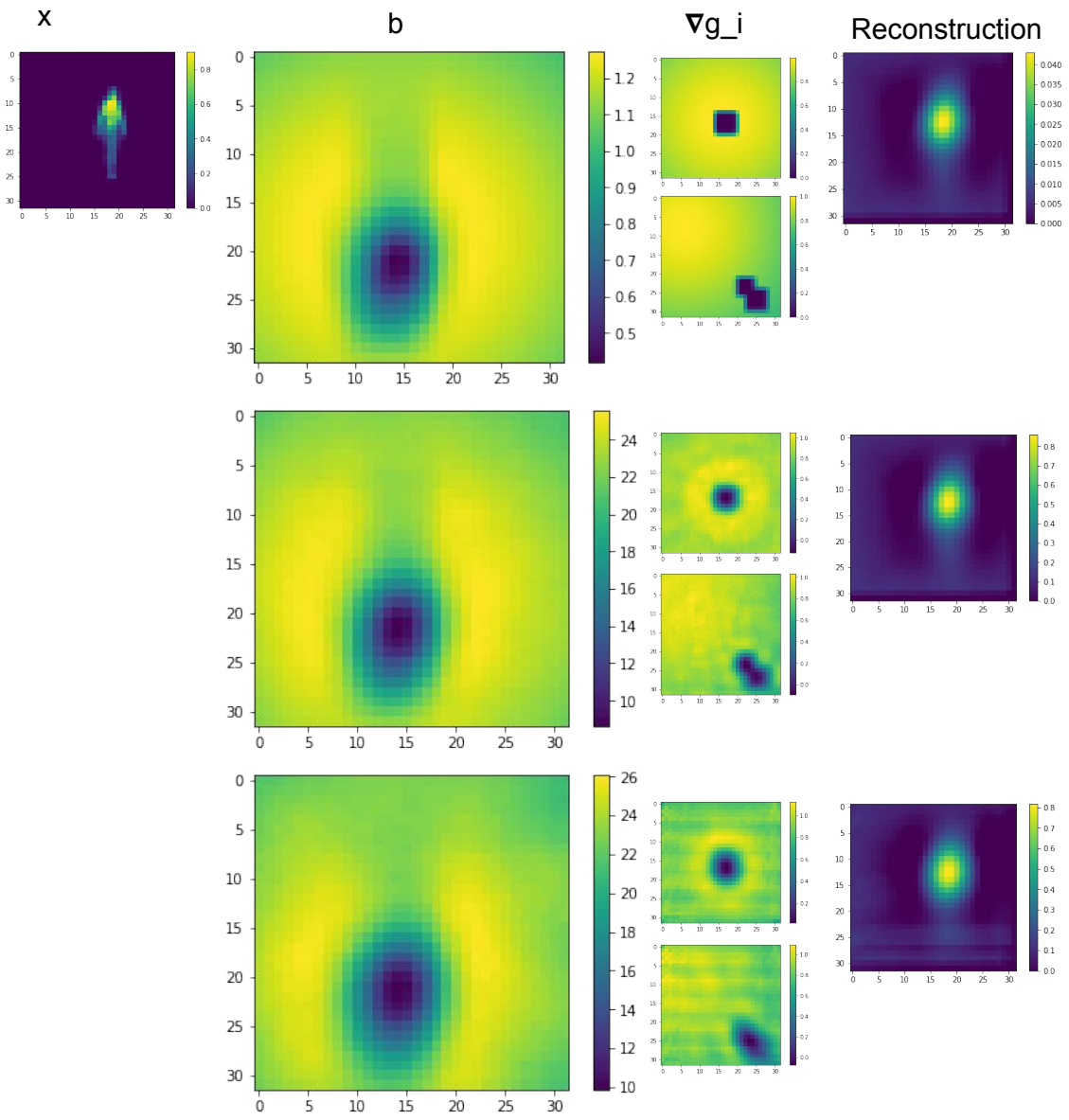


Figure B-3: Networks and their Gradients. The top row is the ground truth, the second row is a low-rank approximation network, and the bottom row corresponds to a convolutional neural network.

Bibliography

[1] Fadel Adib, Zachary Kabelac, and Dina Katabi. Multi-Person localization via RF body reflections. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*, pages 279–292, Oakland, CA, May 2015. USENIX Association.

[2] A. Agrawal, M. Gupta, A. Veeraraghavan, and S. G. Narasimhan. Optimal coded sampling for temporal super-resolution. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 599–606, 2010.

[3] Byeongjoo Ahn, Akshat Dave, Ashok Veeraraghavan, Ioannis Gkioulekas, and Aswin C. Sankaranarayanan. Convolutional approximations to the general non-line-of-sight imaging operator. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[4] Miika Aittala, Prafull Sharma, Lukas Murmann, Adam Yedidia, Gregory Wornell, Bill Freeman, and Fredo Durand. Computational mirrors: Blind inverse light transport by deep matrix factorization. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 14311–14321. Curran Associates, Inc., 2019.

[5] Kazunori Akiyama, Andrew Chael, and Dominic W. Pesce. New views of black holes from computational imaging. *Nature Computational Science*, 1(5):300–303, May 2021.

[6] AliceVision. Meshroom: A 3D reconstruction software., 2018.

[7] Ansys. Perception algorithms are the key to autonomous vehicles safety, 2021.

[8] Dejan Azinović, Tzu-Mao Li, Anton Kaplanyan, and Matthias Nießner. Inverse path tracing for joint material and lighting estimation. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2019.

[9] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*, 2016.

[10] Guha Balakrishnan, Adrian V. Dalca, Amy Zhao, John V. Guttag, Fredo Durand, and William T. Freeman. Visual deprojection: Probabilistic recovery of collapsed dimensions. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[11] M. Baradad, V. Ye, A. B. Yedidia, F. Durand, W. T. Freeman, G. W. Wornell, and A. Torralba. Inferring light fields from shadows. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6267–6275, June 2018.

[12] J. T. Barron and J. Malik. Intrinsic scene properties from a single rgb-d image. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 17–24, June 2013.

[13] P. Bergeron. A general version of crow's shadow volumes. *IEEE Computer Graphics and Applications*, 6(9):17–28, Sep. 1986.

[14] Dana Berman, Tali Treibitz, and Shai Avidan. Non-local image dehazing. *CVPR*, 2016.

[15] Blender Foundation. *Blender - a 3D modelling and rendering package*, 2018.

[16] Nick Bostrom. Ethical issues in advanced artificial intelligence. In Wendell Wallach and Peter Asaro, editors, *Machine Ethics and Robot Ethics*. 2017.

[17] Katherine L. Bouman, Vickie Ye, Adam B. Yedidia, Fredo Durand, Gregory W. Wornell, Antonio Torralba, and William T. Freeman. Turning corners into cameras: Principles and methods. In *ICCV*, 2017.

[18] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, January 2011.

[19] Mauro Buttafava, Jessica Zeman, Alberto Tosi, Kevin Eliceiri, and Andreas Velten. Non-line-of-sight imaging using a time-gated single photon avalanche diode. *Opt. Exp.*, 2015.

[20] Piergiorgio Caramazza, Alessandro Boccolini, Daniel Buschek, Matthias Hullin, Catherine Higham, Robert Henderson, Roderick Murray-Smith, and Daniele Faccio. Neural network identification of people hidden from view with a single-pixel, single-photon detector. *arXiv preprint arXiv:1709.07244*, 2017.

[21] V. Caselles, A. Chambolle, and M. Novaga. *Total Variation in Imaging*, pages 1455–1499. Springer New York, New York, NY, 2015.

[22] Ayan Chakrabarti. Learning sensor multiplexing design through back-propagation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3089–3097, Red Hook, NY, USA, 2016. Curran Associates Inc.

[23] Eric Chan and Frédo Durand. An efficient hybrid shadow rendering algorithm. In *Proceedings of the Eurographics Symposium on Rendering*, pages 185–195. Eurographics Association, 2004.

[24] Julie Chang and Gordon Wetzstein. Deep optics for monocular depth estimation and 3d object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 10193–10202, 2019.

[25] C. Che, F. Luan, S. Zhao, K. Bala, and I. Gkioulekas. Towards learning-based inverse subsurface scattering. In *2020 IEEE International Conference on Computational Photography (ICCP)*, pages 1–12, 2020.

[26] W. Chen, S. Daneau, C. Brosseau, and F. Heide. Steady-state non-line-of-sight imaging. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6783–6792, June 2019.

[27] A. L. Cohen. Anti-pinhole imaging. *Journal of Modern Optics*, 29:63–67, 1982.

[28] R. Dennis Cook. Detection of influential observation in linear regression. *Technometrics*, 19(1):15–18, 1977.

[29] Franklin C. Crow. Shadow algorithms for computer graphics. *SIGGRAPH Comput. Graph.*, 11(2):242–248, July 1977.

[30] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, page 189–198, New York, NY, USA, 1998. Association for Computing Machinery.

[31] Steven Diamond, Vincent Sitzmann, Felix Heide, and Gordon Wetzstein. Unrolled optimization with deep priors. *arXiv preprint arXiv:1705.08041*, 2017.

[32] Sai kiran Doddalla and Georgios C. Trichopoulos. Non-line of sight terahertz imaging from a single viewpoint. In *2018 IEEE/MTT-S International Microwave Symposium - IMS*, pages 1527–1529, 2018.

[33] R.D. Fernald. Evolving eyes. *Int J Dev Biol.*, (48(8-9)):701–5, 2004.

[34] Roy Frostig, Matthew Johnson, and Chris Leary. Compiling machine learning programs via high-level tracing. 2018.

[35] Genevieve Gariepy, Francesco Tonolini, Robert Henderson, Jonathan Leach, and Daniele Faccio. Detection and tracking of moving objects hidden from view. *Nat. Photon.*, 2015.

[36] Mathieu Garon, Kalyan Sunkavalli, Sunil Hadap, Nathan Carr, and Jean-Francois Lalonde. Fast spatially-varying indoor lighting estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[37] Stamatios Georgoulis, Konstantinos Rematas, Tobias Ritschel, Mario Fritz, Tinne Tuytelaars, and Luc Van Gool. What is around the camera? In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[38] Davis Gilton, Greg Ongie, and Rebecca Willett. Neumann networks for linear inverse problems in imaging. *IEEE Transactions on Computational Imaging*, 6:328–343, 2019.

[39] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[40] Felix Heide, Lei Xiao, Wolfgang Heidrich, and Matthias B. Hullin. Diffuse mirrors: 3D reconstruction from diffuse indirect illumination using inexpensive time-of-flight sensors. *CVPR*, 2014.

[41] Roarke Horstmeyer, Richard Y Chen, Barbara Kappes, and Benjamin Judkewitz. Convolutional neural networks that teach microscopes how to image. *arXiv preprint arXiv:1709.07223*, 2017.

[42] Thomas Houllier and Thierry Lépine. Comparing optimization algorithms for conventional and freeform optical design. *Opt. Express*, 27(13):18940–18957, Jun 2019.

[43] Julian Iseringhausen and Matthias B. Hullin. Non-line-of-sight reconstruction using efficient transient rendering. *ACM Trans. Graph.*, 39(1), January 2020.

[44] Michal Jancosek and Tomas Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR 2011*. IEEE, jun 2011.

[45] S. Jiddi, P. Robert, and E. Marchand. Estimation of position and intensity of dynamic light sources using cast shadows on textured real surfaces. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1063–1067, 2018.

[46] Salma Jiddi, Philippe Robert, and Eric Marchand. Detecting specular reflections and cast shadows to estimate reflectance and illumination of dynamic indoor scenes. *IEEE transactions on visualization and computer graphics*, 2020.

[47] Achuta Kadambi, Hang Zhao, Boxin Shi, and Ramesh Raskar. Occluded imaging with time-of-flight sensors. *ACM TOG*, 2016.

[48] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, August 1986.

[49] Ulugbek S Kamilov, Ioannis N Papadopoulos, Morteza H Shoreh, Alexandre Goy, Cedric Vonesch, Michael Unser, and Demetri Psaltis. Learning approach to optical tomography. *Optica*, 2, 2015.

[50] H. Kato, Deniz Beker, M. Morariu, T. Ando, T. Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *ArXiv*, abs/2006.12057, 2020.

[51] Brendan Kelly, Thomas P Matthews, and Mark A Anastasio. Deep learning-guided image reconstruction from incomplete data. *arXiv preprint arXiv:1709.00584*, 2017.

[52] S. S. Khan, V. Sundar, V. Boominathan, A. Veeraraghavan, and K. Mitra. Flatnet: Towards photorealistic scene reconstruction from lensless measurements. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2020.

[53] Jonathan Klein, Christoph Peters, Jaime Martín, Martin Laurenzis, and Matthias B. Hullin. Tracking objects outside the line of sight using 2D intensity images. *Sci. Rep.*, 2016.

[54] L. Kneip and P. Furgale. Opengv: A unified and generalized approach to real-time calibrated geometric vision. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8, 2014.

[55] Laurent Kneip, Hongdong Li, and Yongduek Seo. Upnp: An optimal o(n) solution to the absolute pose problem with universal applicability. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 127–142, Cham, 2014. Springer International Publishing.

[56] Eric I Knudsen. Evolution of neural processing for visual perception in vertebrates. *Journal of Comparative Neurology*, 528(17):2888–2901, 2020.

[57] Kuldeep Kulkarni, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok. Reconnet: Non-iterative reconstruction of images from compressively sensed measurements. In *CVPR*, 2016.

[58] Jean-François Lalonde, Alexei A. Efros, and Srinivasa G. Narasimhan. Estimating the natural illumination conditions from a single outdoor image. *International Journal of Computer Vision*, 2011.

[59] Douglas Lanman, Ramesh Raskar, Amit Agrawal, and Gabriel Taubin. Shield fields: Modeling and capturing 3d occluders. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 27(5), 2008.

[60] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[61] Chloe LeGendre, Wan-Chun Ma, Rohit Pandey, Sean Fanello, Christoph Rhemann, Jason Dourgarian, Jay Busch, and Paul Debevec. Learning illumination

from diverse portraits. In *SIGGRAPH Asia 2020 Technical Communications*, SA '20, New York, NY, USA, 2020. Association for Computing Machinery.

[62] Xin Lei, Liangyu He, Yixuan Tan, Ken Xingze Wang, Xinggang Wang, Yihan Du, Shanhui Fan, and Zongfu Yu. Direct object recognition without line-of-sight using optical coherence. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11729–11738, 2019.

[63] Anat Levin, Rob Fergus, Frédo Durand, and William T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Trans. Graph.*, 26(3):70–es, July 2007.

[64] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 31–42, 1996.

[65] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):222:1–222:11, 2018.

[66] Yuanzhen Li, Hanqing Lu, Heung-Yeung Shum, et al. Multiple-cue illumination estimation in textured scenes. In *Proceedings Ninth IEEE International Conference on Computer Vision*, pages 1366–1373. IEEE, 2003.

[67] Pantelis Linardatos, Vasilis Papastefanopoulos, and Sotiris Kotsiantis. Explainable ai: A review of machine learning interpretability methods. *Entropy*, 23(1), 2021.

[68] David B. Lindell, Gordon Wetzstein, and Vladlen Koltun. Acoustic non-line-of-sight imaging. 2019.

[69] David B. Lindell, Gordon Wetzstein, and Matthew O'Toole. Wave-based non-line-of-sight imaging using fast f-k migration. *ACM Trans. Graph. (SIGGRAPH)*, 38(4):116, 2019.

[70] Xiaochun Liu, Ibón Guillén, Marco La Manna, Ji Hyun Nam, Syed Azer Reza, Toan Huu Le, Adrian Jarabo, Diego Gutierrez, and Andreas Velten. Non-line-of-sight imaging using phasor-field virtual wave optics. *Nature*, 572(7771):620–623, Aug 2019.

[71] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph.*, 38(6), November 2019.

[72] Gilles Louppe and Kyle Cranmer. Adversarial variational optimization of non-differentiable simulators. *arXiv preprint arXiv:1707.07113*, 2017.

[73] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, page 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[74] Tomohiro Maeda, Guy Satat, Tristan Swedish, Lagnojita Sinha, and Ramesh Raskar. Recent advances in imaging around corners, 2019.

[75] Tomohiro Maeda, Yiqin Wang, Ramesh Raskar, and Achuta Kadambi. Thermal non-line-of-sight imaging. In *2019 IEEE International Conference on Computational Photography (ICCP)*, pages 1–11, 2019.

[76] Christopher A. Metzler, Felix Heide, Prasana Rangarajan, Muralidhar Madabhushi Balaji, Aparna Viswanath, Ashok Veeraraghavan, and Richard G. Baraniuk. Deep-inverse correlography: towards real-time high-resolution non-line-of-sight imaging. *Optica*, 7(1):63–71, Jan 2020.

[77] Christopher A. Metzler, Ali Mousavi, and Richard G. Baraniuk. Learned d-amp: Principled neural network based compressive image recovery. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1770–1781, Red Hook, NY, USA, 2017. Curran Associates Inc.

[78] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[79] K. Mitra and A. Veeraraghavan. Light field denoising, light field superresolution and stereo camera based refocussing using a gmm light field patch prior. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 22–28, 2012.

[80] Pierre Moulon, Pascal Monasse, and Renaud Marlet. Adaptive structure from motion with a contrario model estimation. In *Proceedings of the Asian Computer Vision Conference (ACCV 2012)*, pages 257–270. Springer Berlin Heidelberg, 2012.

[81] John Murray-Bruce, Charles Saunders, and Vivek K. Goyal. Occlusion-based computational periscopy with consumer cameras. In Dimitri Van De Ville, Manos Papadakis, and Yue M. Lu, editors, *Wavelets and Sparsity XVIII*, volume 11138, pages 286 – 297. International Society for Optics and Photonics, SPIE, 2019.

[82] F. Naser, I. Gilitschenski, G. Rosman, A. Amini, F. Durand, A. Torralba, G. W. Wornell, W. T. Freeman, S. Karaman, and D. Rus. Shadowcam: Real-time detection of moving obstacles behind a corner for autonomous vehicles. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 560–567, Nov 2018.

[83] Ren Ng. Fourier slice photography. In *ACM SIGGRAPH 2005 Papers*, pages 735–744. 2005.

[84] D. E. Nilsson and S. Pelger. A pessimistic estimate of the time required for an eye to evolve. In *Proc Biol Sci.*, pages 256(1345):53–8, 1994.

[85] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 38(6), December 2019.

[86] Matthew O'Toole, Ramesh Raskar, and Kiriakos N Kutulakos. Primal-dual coding to probe light transport. *ACM Trans. Graph.*, 31(4):39–1, 2012.

[87] Matthew O'Toole, David B Lindell, and Gordon Wetzstein. Confocal non-line-of-sight imaging based on the light-cone transform. *Nature*, 555(7696):338, 2018.

[88] Jeong Joon Park, Aleksander Holynski, and Steven M Seitz. Seeing the world in a bag of chips, 2020.

[89] Gustavo Patow and Xavier Pueyo. A survey of inverse rendering problems. *Computer Graphics Forum*, 22(4):663–687, 2003.

[90] Yifan (Evan) Peng, Ashok Veeraraghavan, Wolfgang Heidrich, and Gordon Wetzstein. Deep optics: Joint design of optics and image recovery algorithms for domain specific cameras. In *ACM SIGGRAPH 2020 Courses*, SIGGRAPH 2020, New York, NY, USA, 2020. Association for Computing Machinery.

[91] Bui Tuong Phong. Illumination for computer generated pictures. *Commun. ACM*, 18(6):311–317, jun 1975.

[92] Louis B. Rall and George F. Corliss. An introduction to automatic differentiation. In Martin Berz, Christian H. Bischof, George F. Corliss, and Andreas Griewank, editors, *Computational Differentiation: Techniques, Applications, and Tools*, pages 1–17. SIAM, Philadelphia, PA, 1996.

[93] Joshua Rapp, Charles Saunders, Julián Tachella, et al. Seeing around corners with edge-resolved transient imaging. *Nature Communications*, 11(5929), November 2020.

[94] Ramesh Raskar, Amit Agrawal, and Jack Tumblin. Coded exposure photography: motion deblurring using fluttered shutter. In *ACM SIGGRAPH 2006 Papers*, pages 795–804. 2006.

[95] Rendered.ai. Data engineering tools for proveable ai, 2021.

[96] Guy Satat, Matthew Tancik, Otkrist Gupta, Barmak Heshmat, and Ramesh Raskar. Object classification through scattering media with deep learning on time resolved measurement. *Optics Express*, 25, 2017.

[97] Guy Satat, Matthew Tancik, and Ramesh Raskar. Towards photography through realistic fog. In *Computational Photography (ICCP), 2018 IEEE International Conference on*, pages 1–10. IEEE, 2018.

[98] Imari Sato, Yoichi Sato, and Katsushi Ikeuchi. Illumination from shadows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(3):290–300, 2003.

[99] Charles Saunders, John Murray-Bruce, and Vivek K Goyal. Computational periscopy with an ordinary digital camera. *Nature*, 565(7740):472–475, January 2019.

[100] I.R. Schwab. The evolution of eyes: major steps. In *The Keeler lecture*, volume 32(2), pages 302–313, 2017.

[101] S. W. Seidel, Y. Ma, J. Murray-Bruce, C. Saunders, W. T. Freeman, C. C. Yu, and V. K. Goyal. Corner occluder computational periscopy: Estimating a hidden scene from a single photograph. In *2019 IEEE International Conference on Computational Photography (ICCP)*, pages 1–9, May 2019.

[102] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[103] Pradeep Sen, Billy Chen, Gaurav Garg, Stephen R. Marschner, Mark Horowitz, Marc Levoy, and Hendrik P. A. Lensch. Dual photography. *ACM Trans. Graph.*, 24(3):745–755, July 2005.

[104] David Shafer. Lens designs with extreme image quality features. *Advanced Optical Technologies*, 2(1):53–62, 2013.

[105] Mark Sheinin and Yoav Y. Schechner. The next best underwater view. *CVPR*, 2016.

[106] Amit Shesh, Antonio Criminisi, Carsten Rother, and Gavin Smyth. 3d-aware image editing for out of bounds photography. In *Proceedings of Graphics Interface 2009*, GI '09, page 47–54, CAN, 2009. Canadian Information Processing Society.

[107] Ayan Sinha, Justin Lee, Shuai Li, and George Barbastathis. Lensless computational imaging through deep learning. *Optica*, 4, 2017.

[108] Vincent Sitzmann, Steven Diamond, Yifan Peng, Xiong Dun, Stephen Boyd, Wolfgang Heidrich, Felix Heide, and Gordon Wetzstein. End-to-end optimization of optics and image processing for achromatic extended depth of field and super-resolution imaging. *ACM Transactions on Graphics (TOG)*, 37(4):1–13, 2018.

[109] 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.

[110] Joe Staines and David Barber. Variational optimization. *arXiv preprint arXiv:1212.4507*, 2012.

[111] Qilin Sun, Jian Zhang, Xiong Dun, Bernard Ghanem, Yifan Peng, and Wolfgang Heidrich. End-to-end learned, optically coded super-resolution spad camera. *ACM Transactions on Graphics (TOG)*, 39(2):1–14, 2020.

[112] Yu Sun, Zhihao Xia, and Ulugbek S. Kamilov. Efficient and accurate inversion of multiple scattering with deep learning. *Opt. Express*, 26(11):14678–14688, May 2018.

[113] Richard Szeliski. Computer vision algorithms and applications, 2011.

[114] Matthew Tancik, Guy Satat, and Ramesh Raskar. Flash photography for data-driven hidden scene recovery. *arXiv preprint arXiv:1810.11710*, 2018.

[115] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B Goldman, and M. Zollhöfer. State of the art on neural rendering. *EG*, 2020.

[116] C. Thrampoulidis, G. Shulkind, F. Xu, W. T. Freeman, J. H. Shapiro, A. Torralba, F. N. C. Wong, and G. W. Wornell. Exploiting occlusion in non-line-of-sight active imaging. *IEEE Transactions on Computational Imaging*, 4(3):419–431, Sep. 2018.

[117] Antonio Torralba and William T. Freeman. Accidental pinhole and pinspeck cameras. *International Journal of Computer Vision*, 110(2):92–112, March 2014.

[118] Neil Vaughan. Evolution of biological eye in computer simulation. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 2537–2543, 2019.

[119] A. Veeraraghavan, R. Raskar, A. Agrawal, A. Mohan, and J Tumblin. Dappled photography: mask enhanced cameras for heterodyned light fields and coded aperture refocusing. *ACM Transactions on Graphics*, 26(3), 2007.

[120] Andreas Velten, Thomas Willwacher, Otkrist Gupta, Ashok Veeraraghavan, Moungi G Bawendi, and Ramesh Raskar. Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging. *Nat. Comms.*, 3, 2012.

[121] Henrique Weber, Donald Prévost, and Jean-François Lalonde. Learning to estimate indoor lighting from 3d objects. *2018 International Conference on 3D Vision (3DV)*, pages 199–207, 2018.

[122] Yicheng Wu, Vivek Boominathan, Huaijin Chen, Aswin Sankaranarayanan, and Ashok Veeraraghavan. Phasecam3d—learning phase masks for passive single view depth estimation. In *2019 IEEE International Conference on Computational Photography (ICCP)*, pages 1–12. IEEE, 2019.

[123] Chris Wyman and Zeng Dai. Imperfect voxelized shadow volumes. In *ACM SIGGRAPH 2013 Talks*, SIGGRAPH '13, New York, NY, USA, 2013. Association for Computing Machinery.

[124] Y. Yang and A. Yuille. Sources from shading. In *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 534–539, June 1991.

[125] A. B. Yedidia, M. Baradad, C. Thrampoulidis, W. T. Freeman, and G. W. Wornell. Using unknown occluders to recover hidden scenes. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12223–12231, June 2019.

[126] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[127] Edward Zhang, Michael F. Cohen, and Brian Curless. Discovering point lights with intensity distance fields. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[128] L. Zhang, Q. Yan, Z. Liu, H. Zou, and C. Xiao. Illumination decomposition for photograph with multiple light sources. *IEEE Transactions on Image Processing*, 26(9):4114–4127, 2017.

[129] Xianan Zhang, Lieke Chen, Mingjie Feng, and Tao Jiang. Toward reliable non-line-of-sight localization using multipath reflections. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 6(1), mar 2022.

[130] Bo Zhu, Jeremiah Z Liu, Stephen F Cauley, Bruce R Rosen, and Matthew S Rosen. Image reconstruction by domain-transform manifold learning. *Nature*, 555(7697):487–492, 2018.