Computational illumination for portrait photography and inverse graphics

by

Lukas Murmann

B.Sc., Technical University of Munich, 2011
M.Sc., University College London, 2013
S.M., Massachusetts Institute of Technology, 2017

Submitted to the
Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2021

Author .....

Department of Electrical Engineering and Computer Science
May 20, 2021

Certified by .....

Fredo Durand
Amar Bose Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....

Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Acknowledgments

I want to thank my thesis supervisor Fredo Durand for his unfailing support during my PhD. Fredo's clarity and discipline in the pursuit of science will stay with me long after graduation. I also want to express my gratitude to my thesis committee who provided support and encouragement throughout my entire time at MIT. The work presented in this thesis would not have been possible without the advice and contributions of my co-authors. I was fortunate to complete three summer internships and would like to thank my hosts and coworkers for their profound impact on my research. Being a member of CSAIL's Computer Graphics group and Vision/Graphics neighborhood has been a wonderful experience and my fellow student and post docs continue to be a source of inspiration.

I want to thank my friends, both in Boston and elsewhere, for supporting me throughout this entire journey. I could not have done it without them. I am deeply grateful to my parents for instilling in me a sense of curiosity, and for giving me the freedom to pursue my dreams.

Early during my time in Cambridge, I hit a lucky break and met my wife Alba at an MIT event. Alba has been an inexhaustible source of support during my thesis work and enriches my life beyond measure. Thank you.

Contents

List of Figures

Chapter 1

Introduction

This thesis considers the use of computer-controlled illumination for computational photography and inverse graphics problems. In particular, we will introduce solutions to

- image relighting, lighting estimation, and mixed white balance
- photography at very low light levels
- shape and material reconstruction

.

The above problems are known to be very challenging and in general require both a large number of observations and simplifying assumptions to solve [2, 8, 12, 5]. Simplifying assumptions might appear both in a constrained solution space, for example restricting recovered materials to a small family of parametric models, or in a simplified image formation model, for example one that ignores shadows and global illumination.

The work presented in this thesis presents two approaches to tackle these challenging problems.

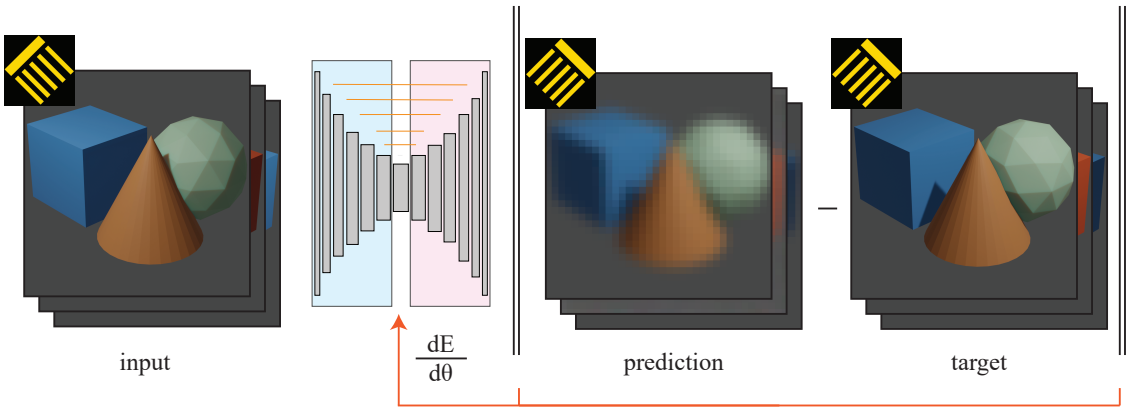

Figure 1-1: Supervised learning is a powerful technique for approximating functions that do not have a known closed form solution, but for which paired input/target training pairs can be collected. In the above example, we train a convolutional network to relight the input image to another fixed light direction. Given a sufficiently large dataset, the network learns to synthesize plausible images without the need for the programmer to explicitly encode domain knowlege.

1.1 Supervised learning for computational photography: Learning image formation and priors from data

This thesis proposes using supervised training of feed-forward convolutional models for computational photography applications. Supervised training removes the need to explicitly encode assumptions about the image formation model, or the need to specify a-priori heuristics about the distribution of output images. Instead, a deep network is trained from labeled data pairs, and learns the distribution of output images from that dataset. Similarly, the network is trained to approximate the image formation model, without explicit assumptions about light transport or imaging hardware. Figure 1-1 shows an example of traning a deep convolutional network for image relighting.

The primary challenge in the supervised learning approach is the need to collect a dataset that (a) is sufficiently large to allow a high-capacity network to be trained without overfitting and (b) comes from the same distribution as the data that is expected at test time. It is difficult to satisfy both of these properties for many relevant

computational photography tasks, including relighting based on multi-illumination image sets.

Multi-illumination image sets are a collection of pictures of a static scene observed under varying lighting conditions. Such image sets are useful for training or validating illumination-related problems, for example image relighting or lighting estimation. Researchers in the computer graphics community have proposed light stages [6] to collect this data, but publicly available datasets are relatively small in size and restricted to subjects that fit into the light stage capture volume.

These limitations, along with the cost of expensive multi-light capture stages has led others to explore using synthetic training data for multi-illumination problems [14]. Large-scale geometry databases [4] and physically-based materials and lighting hold the promise to be indistinguishable from real data, when rendered using high-quality offline rendering algorithms [10]. While this approach offers a promising direction for future research, with current datasets there is a notable domain gap between large synthetic data sets and real data [11]. Beyond image quality, synthetic assets need to be arranged into scenes that contain plausible co-occurrences of individual objects. At present, automatically generating plausible scene layouts remains an active area of research [13].

In chapters 2 and 3, we introduce hardware prototypes that use computer-controlled light sources in portable form factors to collect real-world data sets for supervised learning of computational photography problems. The data collected using these cameras allows us to train deep convolutional networks from scratch. During this process, the networks learn relevant priors about the target distribution from data, rather than encoded as ad-hoc heuristics. Since the data was captured in real scenes, we avoid the domain gap found when training on synthetic data [11]. In the multi-illumination case (Chapter 2), the collected dataset is larger than previously published datasets, and allows us to train state-of-the-art models for several problems. For low-light imaging using near-infrared dark flash (Chapter 3), we are able to collect a unique data set of NIR dark flash pictures in challenging dynamic scenes that allows us to address the problem with a novel "near-infrared colorization" approach.

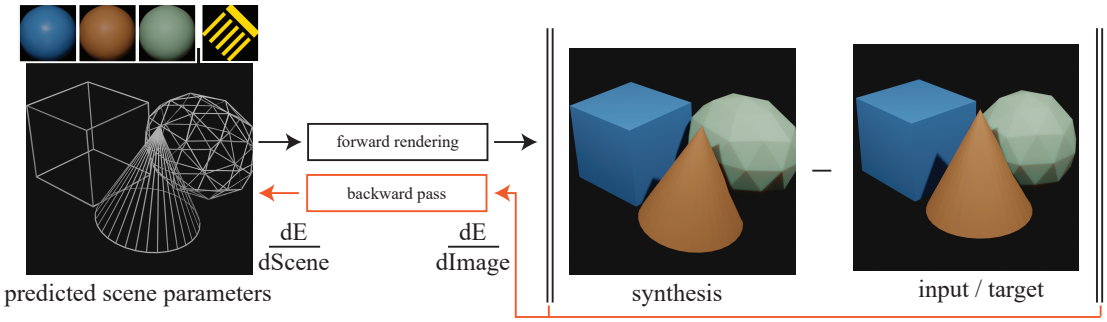

Figure 1-2: Inverse graphics is the problem of reconstructing a 3D representation of a scene (shapes, materials, light sources) from one or more observations of the scene. When solving inverse graphics problems, we can leverage our knowledge of the forward graphics problem in an analysis-by-synthesis setting, where an initial guess of the scene state (left) is rendered using the forward path and compared to the observation (right). A differentiable renderer lets us propagate image-space gradients back to the scene (orange backward path). Scene space gradients are then applied to the current parameter estimate using gradient descent optimization.

1.2 Differentiable rendering for inverse graphics: Using well-understood forward models for inverse problems

For inverse graphics problem (see Figure 1-2), we benefit from a deep understanding of the forward problem, which can be implemented either as offline [10] or real-time [1] rendering algorithms. Forward light transport algorithms are capable of rendering photorealistic images, and the scene parameterizations common in computer graphics are compact and interpretable. Examples for interpretable computer graphics representations include meshes, SDFs, or volumetric occupancy grids for geometry, (SV)BRDFs for materials, and environment lighting or discrete light sources for scene illumination.

Forward rendering can be applied to the inverse problem in an analysis-by-synthesis setting, where the current guess of the scene is rendered using the forward model, after which the rendered image is compared to the known observation. When the renderer is implemented using differentiable programming, image-space gradients can be prop-

agated backwards from the output image to the interpretable scene representation. This makes it possible to minimize the error between synthetic image and observation using gradient descent.

Chapter 4 of this thesis presents an efficient differentiable real-time rendering system. The fundamental difficulty of differentiable rendering, computing the impulse-shaped derivatives that occur along visibility discontinuities [9], is solved through per-pixel visibility analysis that integrates with analytic anti-aliasing of a box-shaped pixel reconstruction filter. The resulting renderer is several times faster than most other differentiable rendering systems and computes gradients that are more accurate than those computed by other differentiable rasterizers. The renderer also supports an arbitrary number of fragments per pixel [3], which makes it an efficient renderer for multi-plane images in inverse problems [7].

Chapters 2 through 4 cover a range of challenging computational photography and inverse graphics problems. The solutions to these problems include both supervised learning using novel datasets, and analysis-by-synthesis through differentiable programming. In the conclusion of this thesis (Chapter 5), we discuss the continuum that exists between these two techniques, and point to future research directions into practical inverse graphics systems that fully utilize both training data and explicitly programmed domain knowledge.

Bibliography

[1] Tomas Akenine-Möller, Eric Haines, Naty Hoffman, Angelo Pesce, Michał Iwanicki, and Sébastien Hillaire. *Real-Time Rendering 4th Edition.* A K Peters/CRC Press, Boca Raton, FL, USA, 2018.

[2] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37(8):1670–1687, 2015.

[3] Loren Carpenter. The a -buffer, an antialiased hidden surface method. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, page 103–108, New York, NY, USA, 1984. Association for Computing Machinery.

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

[5] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 1999.

[6] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. *ACM SIGGRAPH*, 2000.

[7] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Styles Overbeck, Noah Snavely, and Richard Tucker. Deepview: High-quality view synthesis by learned gradient descent. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[8] Samuel W Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 35(6):192:1–192::12, 2016.

[9] 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.

[10] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically Based Rendering, 3rd Edition.* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3nd edition, 2016.

[11] Stephan R. Richter, Zeeshan Hayder, and Vladlen Koltun. Playing for benchmarks. *CoRR*, abs/1709.07322, 2017.

[12] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016.

[13] Kai Wang, Yu-An Lin, Ben Weissmann, Manolis Savva, Angel X. Chang, and Daniel Ritchie. Planit: Planning and instantiating indoor scenes with relation graph and spatial prior networks. *ACM Trans. Graph.*, 38(4), July 2019.

[14] Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. Deep image-based relighting from optimal sparse samples. *ACM Transactions on Graphics (TOG)*, 37(4):126, 2018.

Chapter 2

A Dataset of Multi-Illumination Images in the Wild

published as

Lukas Murmann, Michael Gharbi, Miika Aittala, and Fredo Durand. A multi-illumination dataset of indoor object appearance. In *2019 IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

Abstract: Collections of images under a single, uncontrolled illumination [44] have enabled the rapid advancement of core computer vision tasks like classification, detection, and segmentation [27, 45, 19]. But even with modern learning techniques, many inverse problems involving lighting and material understanding remain too severely ill-posed to be solved with single-illumination datasets. The data simply does not contain the necessary supervisory signals. Multi-illumination datasets are notoriously hard to capture, so the data is typically collected at small scale, in controlled environments, either using multiple light sources [10, 55], or robotic gantries [8, 21]. This leads to image collections that are not representative of the variety and complexity of real-world scenes. We introduce a new multi-illumination dataset of more than 1000 real scenes, each captured in high dynamic range and high resolution, under 25 lighting conditions. We demonstrate the richness of this dataset by training state-of-the-art models for three challenging applications: single-image illumination

Figure 2-1: Using our multi-illumination image dataset of over 1000 scenes, we can train neural networks to solve challenging vision tasks. For instance, one of our models can relight an input image to a novel light direction. Specular highlights pose a significant challenge for many relighting algorithms, but are handled gracefully by our network. Further analysis is presented in Section 2.4.2.

estimation, image relighting, and mixed-illuminant white balance.

2.1 Introduction

The complex interplay of materials and light is central to the appearance of objects and to many areas of computer vision, such as inverse problems and relighting. We argue that research in this area is limited by the scarcity of datasets — the current data is often limited to individual samples captured in a lab setting, e.g. [8, 21], or to 2D photographs that do not encode the variation of appearance with respect to light [44]. While setups such as light stages, e.g. [10], can capture objects under varying illumination, they are hard to move and require the acquired object to be fully enclosed within the stage. This makes it difficult to capture everyday objects in their real environment.

In this paper, we introduce a new dataset of photographs of indoor surfaces under varying illumination. Our goal was to capture small scenes at scale (at least 1,000 scenes). We wanted to be able to bring the capture equipment to any house, apartment or office and record a scene in minutes. For this, we needed a compact setup. This appears to be at odds with the requirement that scenes be illuminated from different directions, since designs such as the light stage [10] have required a large number of individual lights placed around the scene. We resolved this dilemma by using indirect illumination and an electronic flash mounted on servos so that we can

control its direction. As the flash gets rotated, it points to a wall or ceiling near the scene, which forms an indirect "bounce" light source. The reflected light becomes the primary illumination for the scene. We also place a chrome and a gray sphere in the scene as ground truth measurements of the incoming illumination.

Our capture process takes about five minutes per scene and is fully automatic. We have captured over a thousand scenes, each under 25 different illuminations for a total of 25,000 HDR images. Each picture comes segmented and labeled according to material. To the best of our knowledge, this is the first dataset of its kind: offering both everyday objects in context and lighting variations.

In Section 2.4, we demonstrate the generality and usefulness of our dataset with three learning-based applications: predicting the environment illumination, relighting single images, and correcting inconsistent white balance in photographs lit by multiple colored light sources.

We release the full dataset, along with our set of tools for processing and browsing the data, as well as training code and models.

2.2 Related Work

2.2.1 Multi-Illumination Image Sets

Outdoors, the sky and sun are natural sources of illumination varying over time. Timelapse datasets have been harvested both "in the wild" from web cameras [52, 48] or video collections [46], or using controlled camera setups [47, 30, 28].

Indoor scenes generally lack readily-available sources of illumination that exhibit significant variations. Some of the most common multi-illumination image sets are collections of flash/no-flash pairs [41, 12, 2]. These image pairs can be captured relatively easily in a brief two-image burst and enable useful applications like denoising, mixed-lighting white balance [23], or even BRDF capture [1]. Other applications, such as photometric stereo [53] or image relighting [10, 55], require more than two images for reliable results.

Datasets with more than two light directions are often acquired using complex hardware setups and multiple light sources [10, 21]. A notable exception, Mohan *et al.* [36] proposed a user-guided lighting design system that combines several illuminations of a single object. Like us, they acquire their images using a stationary motor-controlled light source and indirect bounce illumination, although within a more restrictive setup, and at a much smaller scale. For their work on user-assisted image compositing Boyadzhiev *et al.* [7] use a remote-controlled camera and manually shine a hand-held flash at the scene. This approach ties down the operator and makes acquisition times prohibitive (they report 20 minutes per scene). Further, hand-holding the light source makes multi-exposure HDR capture difficult. In contrast, our system, inspired by work of Murmann *et al.* [37], uses a motor-controlled bounce flash, which automates the sampling of lighting directions and makes multi-exposure HDR capture straightforward.

2.2.2 Material Databases

To faithfully acquire the reflectance of a real-world surface, one typically needs to observe the surface under multiple lighting conditions. The gold standard in appearance capture for materials is to exhaustively illuminate the material sample and photograph it under every pair of viewpoint and light direction, tabulating the result in a Bidirectional Texture Function (BTF). The reflectance values can then be read off this large table at render-time [8].

A variety of BTF datasets have been published [8, 31, 50], but the total number of samples falls far short of what is typically required by contemporary learning-based algorithms. A rich literature exists on simple, light-weight hardware capture systems [18], but the corresponding public datasets also typically contain less than a few dozen examples. Additionally, the scope, quality and format of these scattered and small datasets varies wildly, making it difficult to use them in a unified manner. Our portable capture device enables us to capture orders of magnitude more surfaces than existing databases and we record entire scenes at once —rather than single objects— "in the wild", outside the laboratory.

Bell *et al.* [5, 6] collected a large dataset of very loosely controlled photographs of materials from the Internet, enriched with crowd-sourced annotations on material class, estimated reflectance, planarity and other properties. Inspired by their approach, we collect semantic material class segmentations for our data, which we detail in section 2.3.3. Unlike ours, their dataset does not contain lighting variations.

Previous works have investigated the use of synthetic image datasets for material estimation [39, 51]. But even carefully crafted synthetic datasets typically do not transfer well to real scenes due remaining differences in scene complexity, object appearance, and image formation [42].

2.3 Dataset

Our dataset consists of 1016 interior scenes, each photographed under 25 predetermined lighting directions, sampled over the upper hemisphere relative to the camera. The scenes depict typical domestic and office environments. To maximize surface and material diversity, we fill the scenes with miscellaneous objects and clutter found in our capture locations. A selection of scenes is presented in Figure 2-2.

In the spirit of previous works [36, 37], our lighting variations are achieved by directing a concentrated flash beam towards the walls and ceiling of the room. The bright spot of light that bounces off the wall becomes a virtual light source that is the dominant source of illumination for the scene in front of the camera.

We can rapidly and automatically control the approximate position of the bounce light simply by rotating the flash head over a standardized set of directions (Figure 2-3). This alleviates the need to re-position a physical light source manually between each exposure [7, 34]. Our camera and flash system is more portable than dedicated light sources, which simplifies its deployment "in the wild".

The precise intensity, sharpness and direction of the illumination resulting from the bounced flash depends on the room geometry and its materials. We record these lighting conditions by inserting a pair of light probes, a reflective chrome sphere and a plastic gray sphere, at the bottom edge of every image [9]. In order to preserve the

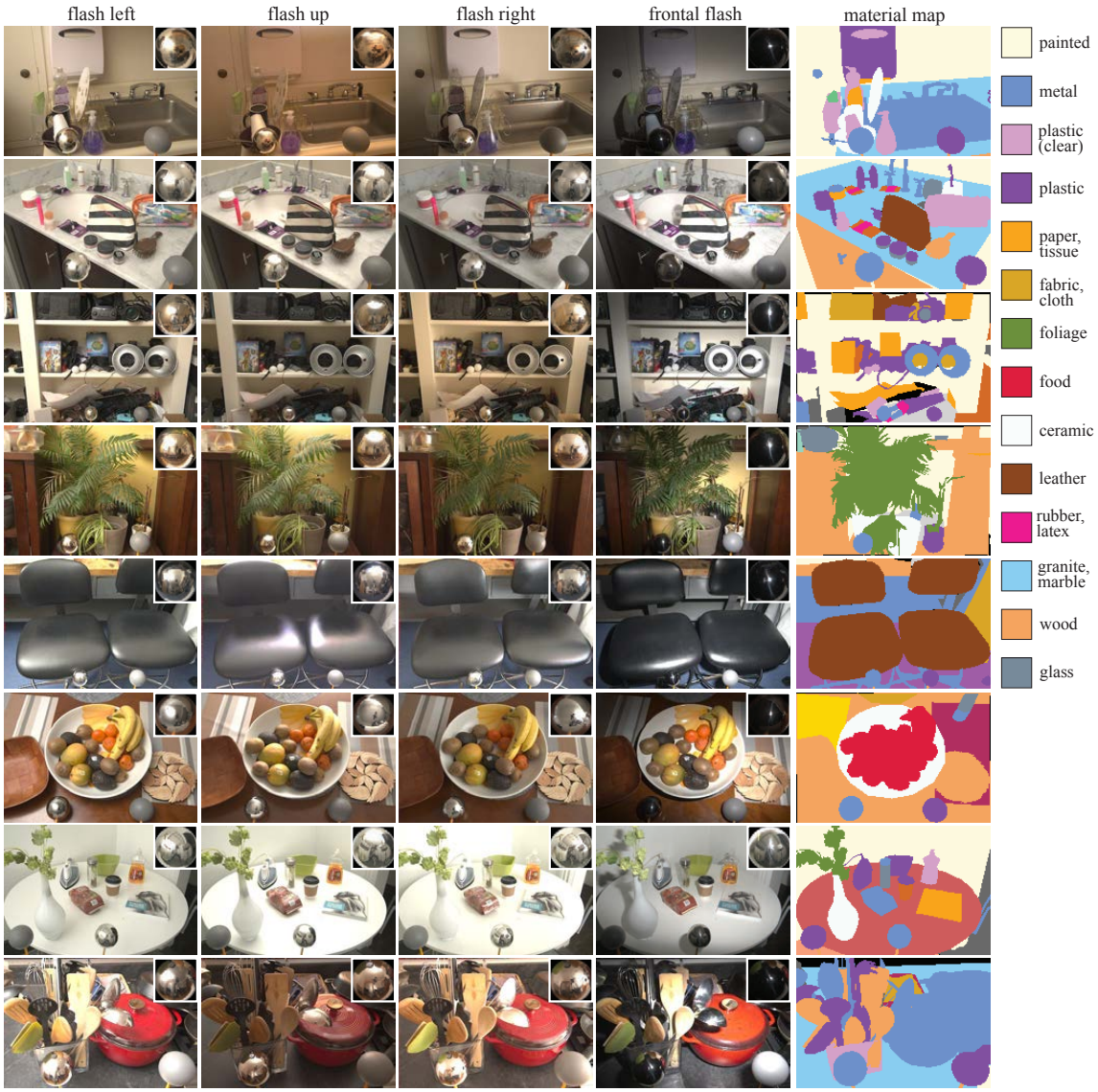

Figure 2-2: Eight representative scenes from our dataset. Each scene is captured under 25 unique light directions, 4 of which are shown in the figure. We strived to include a variety of room and material types in the dataset. Material types are annotated using dense segmentation masks which we show on the right.

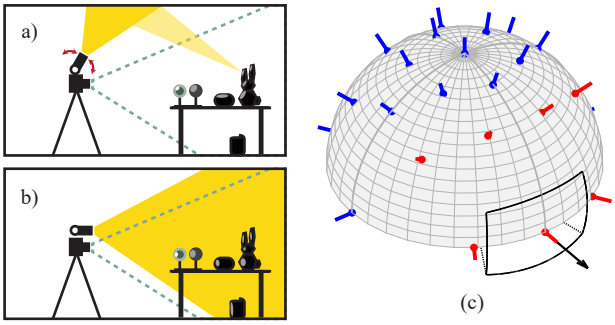

Figure 2-3: a) Most of our photographs are lit by pointing a flash unit towards the walls and the ceiling, creating a virtual bounce light source that illuminates the scene directionally. b) Some of the photographs are captured under direct flash illumination, where the beam of light intersects the field of view of the camera. c) The flash directions used in our dataset, relative to the camera viewing direction and frustum (black). The directions where direct flash illumination is seen in the view are shown in red, and the fully indirect ones in blue.

full dynamic range of the light probes and the viewed scene, all our photographs are taken with bracketed exposures.

As a post-process, we annotate the light probes, and collect dense material labels for every scene using crowd-sourcing, as described in Section 2.3.3.

2.3.1 Image Capture

Our capture device consists of a mirrorless camera (*Sony α6500*), and an external flash unit (*Sony HVL-F60M*) which we equipped with two servo motors. The servos and camera are connected to a laptop, which automatically aims the flash and fires the exposures in a pre-programmed sequence. The 24mm lens provides a 52° horizontal and 36° vertical field of view.

At capture time, we rotate the flash in the 25 directions depicted in Figure 2-3, and capture a 3-image exposure stack for each flash direction. We switch off any room lights and shut window blinds, which brings the average intensity of the ambient light to less than 1% of the intensity of the flash illumination. For completeness, we capture an extra, ambient-only, exposure stack with the flash turned off.

The 25 flash directions are evenly spaced over the upper hemisphere. In 18 of these directions, the cone of the flash beam falls outside the view of the camera, and

consequently, the image is only lit by the secondary illumination from the bounce. In the remaining 7 directions, part or all of the image is lit by the flash directly. In particular, one of the directions corresponds to a typical frontal flash illumination condition.

Capturing a single set (78 exposures) takes about five minutes with our current setup. The capture speed is mostly constrained by the flash's recycling time (around 3.5 seconds at full power). Additional battery extender packs or high-voltage batteries can reduce this delay for short bursts. We found them less useful when capturing many image sets in a single session, where heat dissipation becomes the limiting factor.

2.3.2 HDR processing

The three exposures for each light direction are bracketed in 5-stops increments to avoid clipped highlights and excessive noise in the shadows. The darkest frame is exposed at $f/22$ ISO100, the middle exposure is $f/5.6$ ISO200, and the brightest image is recorded at $f/5.6$ ISO6400. The shutter speed is kept at the camera's fastest flash sync time, $1/160^{\text{th}}$ second to minimize ambient light. The camera sensor has 13 bits of useful dynamic range at ISO100 (9 bits at ISO6400). Overall, our capture strategy allows us to reconstruct HDR images with at least 20 bits of dynamic range.

Using the aperture setting to control exposure bracketing could lead to artifacts from varying defocus blur. We limit this effect by manually focusing the camera to the optimal depth, and by avoiding viewpoints with depth complexity beyond the depth-of-field that is achieved at $f/5.6$.

After merging exposures, we normalize the brightness of the HDR image by matching the intensity of the diffuse gray sphere. The gray sphere also serves as a reference point for white balance. This is especially useful in brightly-colored rooms that could otherwise cause color shifts.

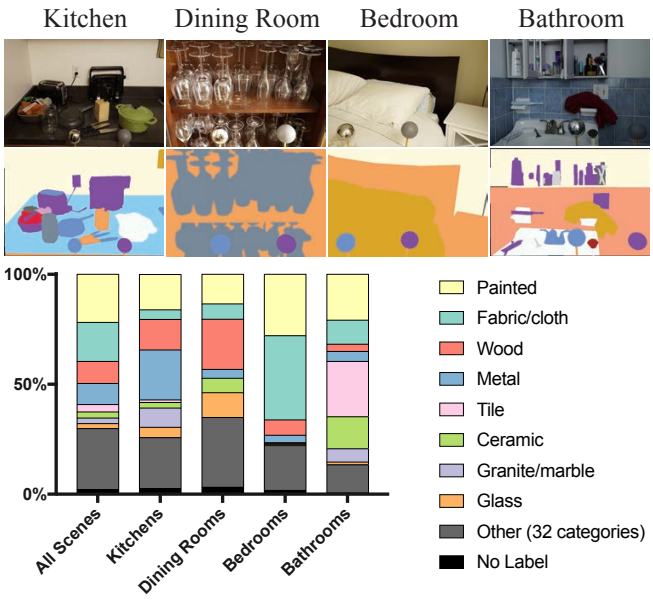

Figure 2-4: Crowd-sourced material annotations show that painted surfaces, fabrics, wood, and metal are the most frequently occurring materials in our dataset, covering more than 60% of all pixels. For some room types, the material distribution is markedly different from the average. For example, in kitchens we frequently encounter wood (shelves) and metal (appliances), bedroom scenes show a high frequency of fabrics, and the material distribution of bathrooms is skewed towards tiles and ceramics.

2.3.3 Dataset Statistics

To ensure our data is representative of many real-world scenes, we collected images in 95 different rooms throughout 12 residential and office buildings, which allowed us to capture a variety of materials and room shapes.

In order to analyze the materials found throughout our dataset, we obtain dense material labels segmented by crowd workers, as shown in Figure 2-2 and 2-4. These annotations are inspired by the material annotations collected by Bell *et al.* [5], whose publicly available source code forms the basis of our annotation pipeline.

Figure 2-4 shows the distribution of materials in our data set. Specific room types have material distributions that differ markedly from the unconditioned distribution. For example, in kitchens we frequently find metal and wooden surfaces, but few fabrics (less than 5% of pixels). Bedrooms scenes on the other hand show fabrics in 38% of the pixels, but contain almost no metal surfaces (less than 4%).

2.4 Applications

In this section we present learning-based solutions to three long-standing vision problems: single-image lighting estimation, single-image relighting and mixed-illuminant white-balance. Our models are based on standard convolutional architectures, such as the U-net [43]. For all experiments, we normalize the exposure and white balance of our input images with respect to the gray sphere. We also mask out the chrome and gray spheres with black squares both at training and test time to prevent the networks from using this information directly.

2.4.1 Predicting Illumination from a Single Image

Single-frame illumination estimation is a challenging problem that arises e.g. when one wishes to composite a computer-generated object into a real-world image [9]. Given sufficient planning (as is common for visual effects in movies), illumination can be recorded at the same time the backdrop is photographed (e.g. by using a light probe). This is rarely the case for a posteriori applications. In particular, with the growing interest in augmented reality and mixed reality, the problem of estimating the illumination in uncontrolled scenes has received increased attention.

Several methods have explored this problem for outdoor images [29, 15, 16, 20, 33] as well as indoor environments [14]. Noting the lack of viable training data for indoor scenes, Gardner *et al.* explicitly detect light sources in LDR panoramas [54]. Our proposed dataset includes HDR light probes in every scene which makes it uniquely suitable for illumination prediction and other inverse rendering tasks [4] in indoor environments.

Model

We approach the single image illumination prediction problem by training a convolutional network on 256×256 image crops from our dataset. We ask the network to predict a 16×16 RGB chrome sphere, that we compare to our ground truth probe using an L_2 loss. The 256×256 input patch is processed by a sequence of convolution,

ReLU, and Max-pooling layers, where we halve the spatial resolution and double the number of feature maps after each convolution. When the spatial resolution reaches 1×1 pixel, we apply a final, fully-connected layer to predict 768 numbers: these are reshaped into a 16×16 RGB light probe image. Exponentiating this images yields the final, predicted environment map. We provide the network details in supplemental material.

Compositing synthetic objects

Figure 2-5 shows some compositing results on a held-out test set. While our model does not always capture the finer color variations of the diffuse global illumination, its prediction of the dominant light source is accurate. Figure 2-7 shows one of the test scenes, with synthetic objects composited. The synthetic geometry is illuminated by our predicted environment maps and rendered with a path-tracer. Note that the ground truth light probes visible in the figure were masked during inference, and therefore *not* seen by our network.

Evaluation

We evaluated our model on a held-out test subset of our data and compared it to a state-of-the-art illumination prediction algorithm by Gardner *et al.* [14]. Compared to their technique, our model more accurately predicts the direction of the bounce light source (see Figure 2-5). In comparison, Gardner *et al.* 's model favors smoother environment maps and is less likely to predict the directional component of the illumination. For visual comparison, we warp the 360° panoramas produced by their technique to the chrome sphere parameterization that is used throughout our paper.

In order to quantify the performance of the chrome sphere predictor, we analyzed the angular distance between the predicted and true center of the light source for 30 test scenes. Our technique achieves a mean angular error of 26.6° (std. dev. 10.8°), significantly outperforming Gardner *et al.* 's method, which achieves a mean error of 68.5° (std. dev. 38.4°). Visual inspection suggests that the remaining failure cases of our technique are due to left/right symmetry of the scene geometry, mirrors, or

Figure 2-5: As the first application of our dataset, we train a deep network to predict environment maps from single input images. Our model consistently predicts the dominant light direction of the ground truth environment map. The model successfully estimates illumination based on shiny objects (a and g) and diffuse reflectors (e.g. row f). Rows h) and i) show failure cases where the network predicts low-confidence outputs close to the mean direction. We compare to Gardner *et al.* 's algorithm [14] which, while predicting visually plausible environment maps, lacks the precise localization of highlights shown by our technique. (Please ignore the vertical seam in Gardner *et al.* 's result. Their model uses a differing spherical parametrization, which we remap to our coordinate system for display.)

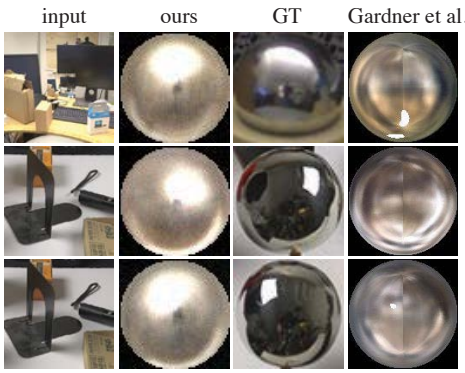

Figure 2-6: We validate our model's ability to generalize beyond bounce flash illumination. The top row show an office scene with regular room lights. The bottom two rows show a scene illuminated by softboxes, first lit from the right and then from the left. The second set of results suggests that our model can aggregate information from shadows to infer the light source position.

Figure 2-7: We use the environment maps predicted by our model to illuminate virtual objects and composite them onto one of our test scenes. The light probe in the bottom of the frame shows the ground truth lighting. (Note that these probes are masked out before feeding the image to the network).

simply lack of context in the randomly chosen input crops (see Figure 2-5 bottom).

We verified that our model generalizes beyond bounce flash light sources using a small test set of pictures taken under general non-bounce flash illumination. The results of this experiment are presented in Figure 2-6.

2.4.2 Relighting

A robust and straightforward method to obtain a relightable model of a scene is to capture a large number of basis images under varying illumination, and render new images as linear combinations of the basis elements. Light stages [10] work according to this principle. With finitely many basis images, representing the high frequency content of a scene's light transport operator (specular highlights, sharp shadow boundaries, etc.) is difficult. Despite this fundamental challenge, prior work has successfully exploited the regularity of light transport in natural scenes to estimate the transport operator from sparse samples [38, 40, 49]. Recent approaches have employed convolutional neural networks for the task, effectively learning the regularities of light transport from synthetic training data and reducing the number of images required for relighting to just a handful [55].

In our work, we demonstrate relighting results from a *single input image* on real-world scenes that exhibit challenging phenomena, such as specular highlights, self-shadowing, and interreflections.

Model

We cast single-image relighting as an image-to-image translation problem. We use a convolutional neural network based on the U-net [43] architecture to map from images illuminated from the left side of the camera, to images lit from the right (see supplemental material for details). Like in Section 2.4.1, we work in the log-domain to limit the dynamic range of the network's internal activations. We use an L_1 loss to compare the *spatial gradients* of our relit output to those of the reference image, lit from the right. We found this gradient-domain loss to yield sharper results. It also allows the network to focus more on fine details without being overly penalized for low-frequency shifts due to the global intensity scaling ambiguity (the left- and right-lit images might not have the same average brightness, depending on room geometry).

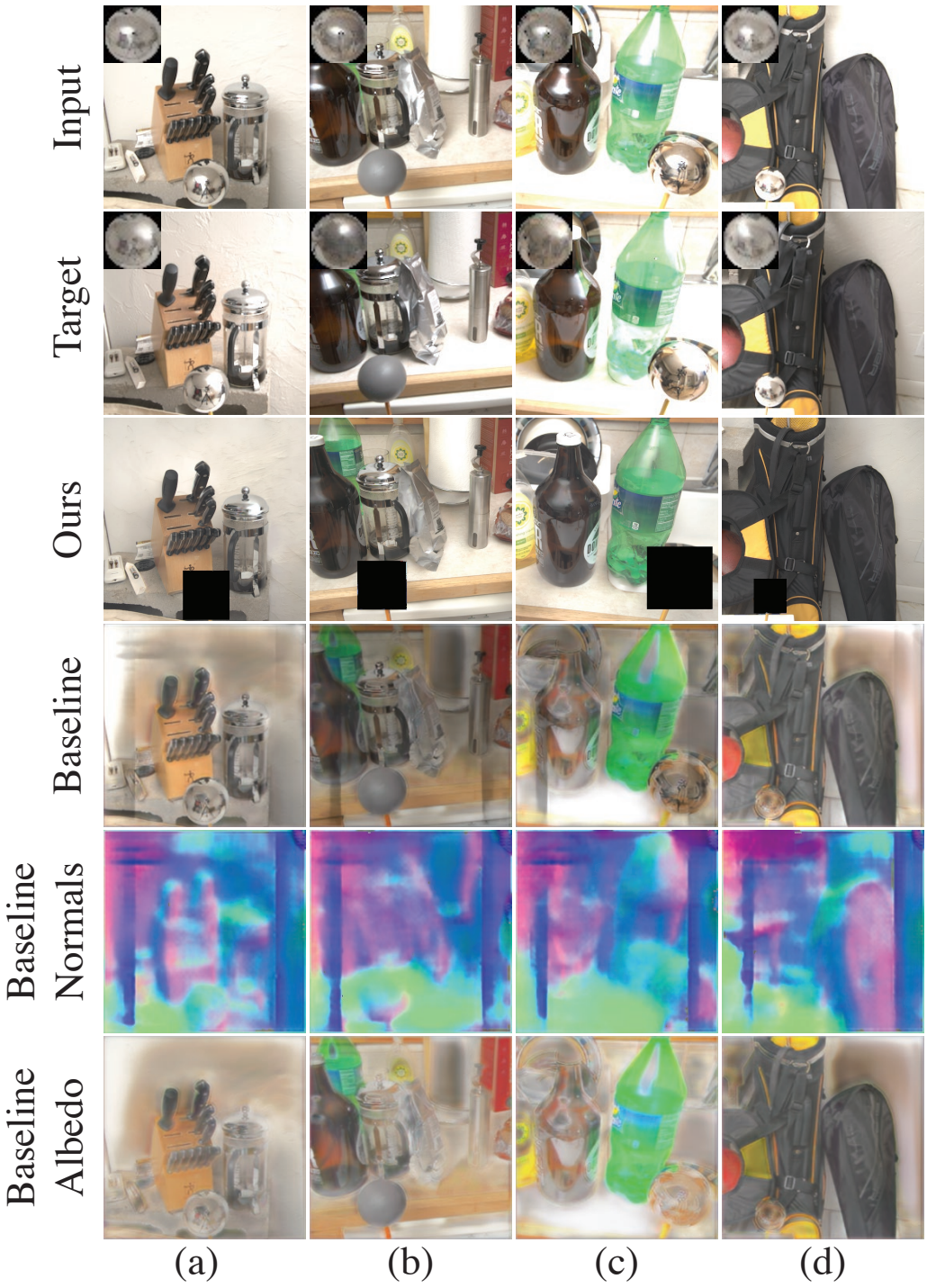

Figure 2-8: The second application of our data is learning image relighting using a single input image. The trained model synthesizes moving specular highlights (a, b, c) and diffuse shading (b, d), and correctly renders shadows behind occluders (d). For the baseline result, we first estimate normals and diffuse albedo using published models, and then re-render the image as lit by the target environment map.

Evaluation

Our model faithfully synthesizes specular highlights, self- and cast-shadows, as well as plausible shading variations. Without a large-scale training dataset for an end-to-end solution, a valid straw man approach would be to use existing techniques and decompose the input into components that can be manipulated for relighting (e.g. normals and albedo). We provide such a baseline for comparison. It uses a combination of deep single-image normal estimation [56] and learned intrinsic image decomposition [25] (see Figure 2-8). Both components are state-of-the-art in their respective fields and have source code and trained models publicly available.

This baseline illustrates the challenges in decomposing the single-image relighting problem into separate inverse rendering sub-problems. Specifically, major sources of artifacts include: incorrect or blurry normals, and incorrect surface albedo due to the overly simplistic Lambertian shading model. The normals and reflectance estimation networks were independently trained to solve two very difficult problems. This is a arguably more challenging than our end-to-end relighting application and, also unnecessary for plausible relighting.

Our end-to-end solution does not enforce this explicit geometry/material decomposition and yields far superior results. More relighting outputs produced by our model are shown in Figure 2-1 and in the supplemental material.

2.4.3 Mixed-Illumination White-Balance

White-balancing an image consists in neutralizing the color cast caused by non-standard illuminants, so that the photograph appears lit by a standardized (typically white) light source. White-balance is under-constrained, and is often solved by modeling and exploiting the statistical regularities in the colors of lights and objects. The most common automatic white balance algorithms make the simplifying assumption that the entire scene is lit by a single illuminant. See [17] for a survey. This assumption rarely holds in practice. For instance, an interior scene might exhibit a mix of bluish light (e.g. from sky illuminating the scene through a window) and warmer tones

(e.g. from the room's artificial tungsten light bulbs). Prior work has formulated a local gray-world assumption to generalize white balance to the mixed-lighting case [11], exploiting the difference in light colors in shadowed vs. sunlit areas for outdoor scenes [26], or flash/no-flash image pairs [35, 22, 24].

Here again, we approach white-balancing as a supervised learning problem. Because our dataset contains high-dynamic range linear images with multiple lighting conditions, it can be used to simulate a wide range of new mixed-color illuminations by linear combinations. We exploit this property to generate a training dataset for a neural network that removes inconsistent color casts from mixed-illumination photographs.

Mixed-illuminant data generation

To create a training set of input/output pairs, we extract 256×256 patches from our scenes at multiple scales. For each patch, we choose a random number of light sources $n \in \{1, \ldots, 4\}$. Each light index corresponds to one of 25 available flash directions, selected uniformly at random without replacement. We denote by $I_1, \ldots, I_n$ the corresponding images.

For each light i, we sample its color with hue in $[0, 360]$, and saturation in $[0.5, 1]$, represented as a positive RGB gain vector α_i, normalized such that $||\alpha_i||_1 = 1$. We randomize the relative power of the light sources by sampling a scalar exposure gain g_i uniformly (in the log domain) between -3 and $+2$ stops. We finally assemble our mixed-colored input patch as the linear combination: $I = \frac{1}{n} \sum_{i=1}^{n} \alpha_i g_i I_i$.

We define the color-corrected target similarly, but without the color gains: $O = \frac{1}{n} \sum_{i=1}^{n} g_i I_i$.

Model

Like for the relighting problem, we use a simple convolutional network based on a U-net [43] to predict white-balanced images from mixed-lighting inputs (details in the supplemental).

To reduce the number of unknowns and alleviate the global scale ambiguity, we

Figure 2-9: The first row shows a mixed white-balance result on an image from a held-out test set. The input (left) is a linear combination of several (two here) illuminations of the same scene under varied color and intensity. The reference image (right) has the same energy but no color cast. Our output (middle) successfully removes the green and magenta shifts. The simple model we trained on our dataset, generalizes well to unseen, real RAW images (second row). The most noticeable failure case are skin tones (third row), which are entirely absent from our data set of static scenes.

take the log transform of the input and target images, and decompose them in 2

chrominance u, v, and a luminance component l [3]:

$$u = \log(I^r + \epsilon) - \log(I^g + \epsilon), \tag{2.1}$$

$$v = \log(I^b + \epsilon) - \log(I^g + \epsilon), \tag{2.2}$$

$$l = \log(I^g + \epsilon), \tag{2.3}$$

where $\epsilon = 10^{-4}$, and the superscripts stand for the RGB color channels.

Our network takes as input u, v, l and outputs two correctly white-balanced chroma components. We assemble the final RGB output from l and the predicted chroma, using the reverse transform. Our model is trained to minimize an L_2 loss over the chroma difference.

Results

Our model successfully removes the mixed color cast on our test set and generalizes beyond, to real-world images. The main limitation of our technique is its poor generalization to skin tones, to which the human eye is particularly sensitive, but which are absent from our dataset of static indoor scenes. We present qualitative results in Figure 2-9 and in the supplemental video.

2.5 Crowd-sourced data annotation

For each scene, we include dense material labels, segmented by crowd workers. These annotations are inspired by the material annotations collected by Bell *et al.* [5], whose publicly available source code forms the basis for our data annotation system. Departing from Bell *et al.* , we strive to densely label each scene with at least 95% coverage. In comparison, the images published by Bell *et al.* have an average coverage of 20%.

Annotations with above 95% coverage also set our dataset apart from semantic segmentation datasets such as Coco [32] and Pascal [13], which generally exhibit many unlabeled background pixels. In particular, the Coco 2014 training set has an average

pixel-level coverage of 29.6%, while Pascal VOC2012 includes annotations for 25.5% of the pixels.

In early crowd sourcing experiments, we found it difficult to achieve high annotation coverage using the polygon-based segmentation user interface from Bell *et al.* [5], which also forms the backbone of the Coco annotations [32]. While segmentation using a single polyline is an effective solution for foreground objects without holes, encouraging a thorough labeling of the background turned out more difficult. We found that background regions must frequently be split into several polygons to avoid accidentally including foreground objects. Further, we observed that labeling both foreground and background would almost double the work required, as each occluding contour must be traced twice.

We overcome these limitations by segmenting objects in order, from front to back. In a typical segmentation session, a worker starts by labeling foreground objects that are not occluded (i.e. objects closest to the camera). Once these occluding objects are labeled, the worker can "extract" them from the image. As background polygons are automatically masked by extracted foreground shapes, the worker can then segment the "second layer" without worrying that their newest polygon might overlap previously segmented areas. Our front-to-back segmentation interface is efficient since occluding contours only have to be traced once, background objects are segmented into a single contiguous polygon, and the front-to-back logic ensures there are neither gaps nor overlap between foreground and background annotations.

We found that crowd workers on Amazon Mechanical Turk were able to reliably segment scenes in front-to-back order after viewing a short tutorial video that introduces our user interface and demonstrates the semantics of front-to-back ordering.

After the segmented shapes are submitted to our server, we add them to a work queue where workers are asked to choose the material for each segment. The choice of material categories and annotation interface follows Bell *et al.* [5]. Each shape is presented to five workers and materials are determined by a simple majority vote. In cases where no material receives a majority, we resolve ties manually. Figure 2-10 compares the segmentations obtained using our technique with thouse obtained using

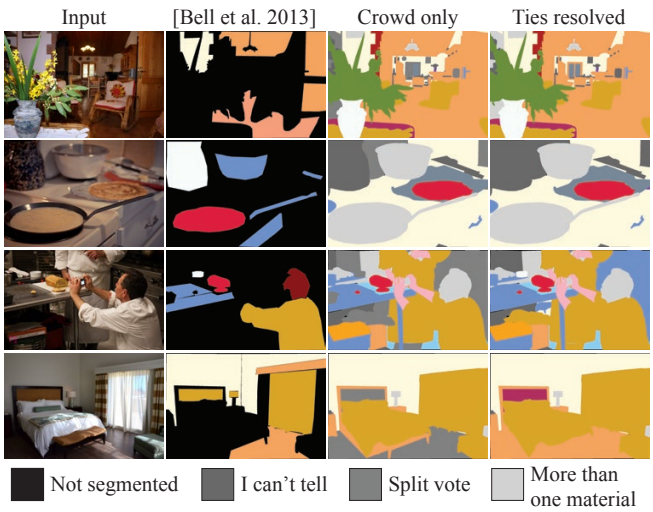

Figure 2-10: Obtaining full-image annotations using an unmodified version of [5] is challenging (column 2). Our modified version of their UI lets workers segment images front-to-back, which allows background objects to be captured in a single contiguous shape (column 3). In some cased (see wooden floor in bottom row), crowd-sourced votes are split, which we resolve manually for the final result (column 4).

the original user interface.

2.6 Network architectures

Let C_k denote a Conv–ReLU layer with k 3×3 kernels. P_k is a $k \times k$ max-pooling operator, and L_k is a 1×1 convolution with k linear outputs (no activation). U_k is a $k \times k$ bilinear upsampling layer.

Illumination prediction Our fully convolutional illumination prediction network takes 256×256 color images as input and produces 16×16 RGB light probe images (i.e. 768 output floats). Its architecture is given by:

$C_{32}P_2C_{64}P_2C_{128}P_2C_{256}P_2C_{512}P_4DC_{512}P_4C_{512}L_{768}$.

Relighting and white-balance Both the relighting and white-balance application use a U-net with 7 downsampling stages and take 3-channel images as input. The relighting output is a color image (3 channels), but the white-balance model produces only the chroma components (2 channels), which are then combined with the input

luminance. The U-net encoder can be described as:

$(C_{64})^2 P_2 (C_{128})^2 P_2 (C_{256})^2 P_2 (C_{512})^2 P_2 (C_{512})^2 P_2$
$(C_{512})^2 P_2 (C_{512})^2 P_2 (C_{512})^2$.

And the decoder is given by:

$U_2 (C_{64})^2 U_2 (C_{128})^2 U_2 (C_{256})^2 U_2 (C_{512})^2 U_2 (C_{512})^2 U_2 (C_{512})^2 U_2 (C_{512})^2 L_{2\text{or}3}$,

with additive skip-connections between matching resolutions.

2.7 Limitations

A limitation of our capture methodology is that it requires good bounce surfaces placed not too far from the scene. This precludes most outdoor scenes and large indoor rooms like auditoriums. Our capture process requires the scene to remain static for several minutes, which keeps us from capturing human subjects. Compared to light stages or robotic gantries, the placement of our bounce light sources has more variability due to room geometry, and the bounce light is softer than hard lighting from point light sources. Finally, we only capture 25 different illuminations, which is sufficient for diffuse materials but under-samples highly specular ones.

2.8 Conclusions

We have introduced a new dataset of indoor object appearance under varying illumination. We have described a novel capture methodology based on an indirect bounce flash which enables, in a compact setup, the creation of virtual light sources. Our automated capture protocol allowed us to acquire over a thousand scenes, each under 25 different illuminations. We presented applications in environment map estimation, single-image relighting, and mixed white balance that can be trained from scratch using our dataset. Code and data are available online at `https://projects.csail.mit.edu/illumination/`.

Bibliography

[1] Miika Aittala, Tim Weyrich, and Jaakko Lehtinen. Two-shot svbrdf capture for stationary materials. *ACM SIGGRAPH*, 2015.

[2] Yağız Aksoy, Changil Kim, Petr Kellnhofer, Sylvain Paris, Mohamed Elgharib, Marc Pollefeys, and Wojciech Matusik. A dataset of flash and ambient illumination pairs from the crowd. *ECCV*, 2018.

[3] Jonathan T. Barron. Convolutional color constancy. *ICCV*, 2015.

[4] Jonathan T. Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(8):1670–1687, Aug 2015.

[5] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. OpenSurfaces: A richly annotated catalog of surface appearance. *ACM SIGGRAPH*, 2013.

[6] Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. Material recognition in the wild with the materials in context database. *CVPR*, 2015.

[7] Ivaylo Boyadzhiev, Sylvain Paris, and Kavita Bala. User-assisted image compositing for photographic lighting. *ACM SIGGRAPH*, 2013.

[8] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 1999.

[9] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. *ACM SIGGRAPH*, 1998.

[10] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. *ACM SIGGRAPH*, 2000.

[11] Marc Ebner. Combining white-patch retinex and the gray world assumption to achieve color constancy for multiple illuminants. In *Joint Pattern Recognition Symposium*, pages 60–67. Springer, 2003.

[12] Elmar Eisemann and Frédo Durand. Flash photography enhancement via intrinsic relighting. *ACM SIGGRAPH*, 2004.

[13] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010.

[14] Marc-André Gardner, Kalyan Sunkavalli, Ersin Yumer, Xiaohui Shen, Emiliano Gambaretto, Christian Gagné, and Jean-François Lalonde. Learning to predict indoor illumination from a single image. *ACM SIGGRAPH Asia*, 2017.

[15] Stamatios Georgoulis, Konstantinos Rematas, Tobias Ritschel, Mario Fritz, Tinne Tuytelaars, and Luc Van Gool. Natural illumination from multiple materials using deep learning. *CoRR*, 2016.

[16] Stamatios Georgoulis, Konstantinos Rematas, Tobias Ritschel, Efstratios Gavves, Mario Fritz, Luc Van Gool, and Tinne Tuytelaars. Reflectance and natural illumination from single-material specular objects using deep learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[17] Arjan Gijsenij, Theo Gevers, and Joost Van De Weijer. Computational color constancy: Survey and experiments. *IEEE Transactions on Image Processing*, 2011.

[18] Dar'ya Guarnera, Giuseppe Claudio Guarnera, Abhijeet Ghosh, Cornelia Denk, and Mashhuda Glencross. BRDF Representation and Acquisition. *Computer Graphics Forum*, 2016.

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *ICCV*, 2017.

[20] Yannick Hold-Geoffroy, Kalyan Sunkavalli, Sunil Hadap, Emiliano Gambaretto, and Jean-François Lalonde. Deep outdoor illumination estimation. *CVPR*, 2017.

[21] Michael Holroyd, Jason Lawrence, and Todd Zickler. A coaxial optical scanner for synchronous acquisition of 3D geometry and surface reflectance. *ACM SIGGRAPH*, 2010.

[22] Zhuo Hui, Aswin C Sankaranarayanan, Kalyan Sunkavalli, and Sunil Hadap. White balance under mixed illumination using flash photography. *IEEE International Conference on Computational Photography*, 2016.

[23] Zhuo Hui, Kalyan Sunkavalli, Sunil Hadap, and Aswin C. Sankaranarayanan. Post-capture lighting manipulation using flash photography. *CoRR*, 2017.

[24] Zhuo Hui, Kalyan Sunkavalli, Sunil Hadap, and Aswin C Sankaranarayanan. Illuminant spectra-based source separation using flash photography. *CVPR*, 2018.

[25] Carlo Innamorati, Tobias Ritschel, Tim Weyrich, and Niloy J. Mitra. Decomposing single images for layered photo retouching. *Computer Graphics Forum*, 2017.

[26] Rei Kawakami, Katsushi Ikeuchi, and Robby T Tan. Consistent surface color for texturing large objects in outdoor scenes. *ICCV*, 2005.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *NIPS*, 2012.

[28] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM SIGGRAPH*, 2014.

[29] 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.

[30] Jean-François Lalonde and Iain Matthews. Lighting estimation in outdoor image collections. *International Conference on 3D Vision*, 2014.

[31] Jason Lawrence, Aner Ben-Artzi, Christopher DeCoro, Wojciech Matusik, Hanspeter Pfister, Ravi Ramamoorthi, and Szymon Rusinkiewicz. Inverse shade trees for non-parametric material representation and editing. *ACM SIGGRAPH*, 2006.

[32] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[33] Stephen Lombardi and Ko Nishino. Reflectance and illumination recovery in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2016.

[34] Vincent Masselus, Philip Dutré, and Frederik Anrys. The free-form light stage. Technical report, Departement of Computer Science, KU Leuven, 2002.

[35] Ryo Matsuoka, Tatsuya Baba, Mia Rizkinia, and Masahiro Okuda. White balancing by using multiple images via intrinsic image decomposition. *IEICE Transactions on Information and Systems*, 2015.

[36] Ankit Mohan, Reynold Bailey, Jonathan Waite, Jack Tumblin, Cindy Grimm, and Bobby Bodenheimer. Tabletop computed lighting for practical digital photography. *IEEE Transactions on Visualization and Computer Graphics*, 2007.

[37] Lukas Murmann, Abe Davis, Jan Kautz, and Frédo Durand. Computational bounce flash for indoor portraits. *ACM SIGGRAPH Asia*, 2016.

[38] Shree K. Nayar, Peter N. Belhumeur, and Terry E. Boult. Lighting sensitive display. *ACM Transactions on Graphics*, 2004.

[39] Keunhong Park, Konstantinos Rematas, Ali Farhadi, and Steven M Seitz. Photoshape: photorealistic materials for large-scale shape collections. *ACM SIGGRAPH Asia*, 2018.

[40] Pieter Peers, Dhruv K. Mahajan, Bruce Lamond, Abhijeet Ghosh, Wojciech Matusik, Ravi Ramamoorthi, and Paul Debevec. Compressive light transport sensing. *ACM Transactions on Graphics*, 2009.

[41] Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. Digital photography with flash and no-flash image pairs. *ACM SIGGRAPH*, 2004.

[42] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. *ECCV*, 2016.

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

[44] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 2015.

[45] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, 2013.

[46] Yichang Shih, Sylvain Paris, Frédo Durand, and William T Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *ACM SIGGRAPH Asia 2013*, 2013.

[47] Jessi Stumpfel, Chris Tchou, Andrew Jones, Tim Hawkins, Andreas Wenger, and Paul Debevec. Direct hdr capture of the sun and sky. *International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction in Africa*, 2004.

[48] Kalyan Sunkavalli, Fabiano Romeiro, Wojciech Matusik, Todd Zickler, and Hanspeter Pfister. What do color changes reveal about an outdoor scene? *CVPR*, 2008.

[49] Jiaping Wang, Yue Dong, Xin Tong, Zhouchen Lin, and Baining Guo. Kernel nyström method for light transport. *ACM SIGGRAPH*, 2009.

[50] Michael Weinmann, Juergen Gall, and Reinhard Klein. Material classification based on training data synthesized using a btf database. In *European Conference on Computer Vision (ECCV)*, pages 156–171, 2014.

[51] Michael Weinmann, Juergen Gall, and Reinhard Klein. Material classification based on training data synthesized using a btf database. *ECCV*, 2014.

[52] Yair Weiss. Deriving intrinsic images from image sequences. *ICCV*, 2001.

[53] Robert J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 1980.

[54] Jianxiong Xiao, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. Recognizing scene viewpoint using panoramic place representation. *CVPR*, 2012.

[55] Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. Deep image-based relighting from optimal sparse samples. *ACM Transactions on Graphics*, 2018.

[56] Yinda Zhang, Shuran Song, Ersin Yumer, Manolis Savva, Joon-Young Lee, Hailin Jin, and Thomas Funkhouser. Physically-based rendering for indoor scene understanding using convolutional neural networks. *CVPR*, 2017.

Chapter 3

Learning Near-Infrared Colorization of Dark Flash Portraits

Authors:

Lukas Murmann[1], Mike Krainin[2], Jiawen Chen[3], Miki Rubinstein[2], William T. Freeman[1,2], Frédo Durand[1]

Author Affiliations:

[1] MIT CSAIL

[2] Google

[3] Adobe

this chapter is unpublished as of May 2021

Abstract: We present an algorithm for capturing portraits in low-light environments that is robust to scene motion and does not require a visible flash. Our system can readily be integrated into modern smartphones, requiring only an unmodified RGB camera, a near-infrared camera, and a wavelength-matched LED flash that is invisible to the human eye. Capturing a time-synchronized "dark flash" pair with our system yields a sharp NIR image lit by the flash and a longer-exposure RGB image illuminated by the ambient environment but with significant blur and noise

and captured from an offset viewpoint. We propose a deep learning algorithm that spectrally transforms the NIR image into the visible range, using the blurry and offset RGB exposure as a guide for visible appearance. By predicting visible-light intensity in addition to chrominance, this formulation generalizes grayscale-to-RGB colorization. The task is challenging because material appearance differ dramatically between near-IR and the visible spectrum. To validate our approach, we develop a hardware prototype and collect a new set of images for training. Unlike previous dark flash approaches, we do not require any modifications to the RGB camera, nor do we impose additional restrictions between the NIR and RGB cameras. We demonstrate the effectiveness of our technique with comparisons to state-of-the-art algorithms and show that it outperforms even commercial solutions under challenging low-light conditions. Finally, we show the applicability of our technique to video inputs.

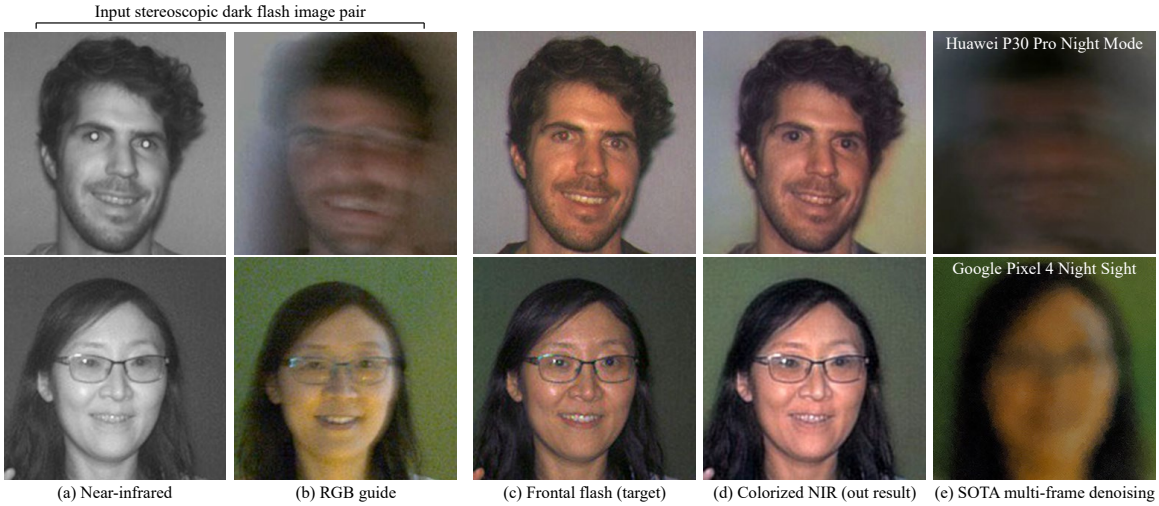

Figure 3-1: Our method captures a synchronized near-infrared (NIR) and RGB image pair from two close viewpoints (a-b), and fuses them to produce a clean RGB image (d) that resembles one captured with visible, frontal flash (c). It uses a novel deep network to predict both intensity and chrominance of the NIR image using the noisy or blurry RGB image as guide. We specifically target portrait photography (non-static scenes, where people might be moving) at very low lux levels—scenarios that pose challenges for state-of-the-art multi-frame denoising algorithms such as Pixel's Night Sight [21] and Huawei's Night Mode (e).

3.1 Introduction

Recent advances in computational techniques have dramatically improved the quality of photographs captured using mobile phone cameras. However, photographing moving subjects in low light remains challenging due to the devices' small lens apertures and sensors, and the resulting images contain significant noise or motion blur (Fig. 3-1). The classical solution employed on today's smartphone cameras is to use an LED flash for additional illumination. However, the bright visible flash dazzles the subject and may be socially unacceptable.

Dark flash photography [16, 34] is a promising alternative because the illumination is not visible to the human eye. Limitations of existing dark flash approaches include impractical restrictions on the hardware configuration (e.g., requiring modifications to the main RGB camera) and their inability to handle motion. We propose a camera configuration where we augment the standard RGB camera with a full-resolution sensor sensitive to near-infrared light and a wavelength-matched LED flash. These components are already present in modern smartphones for facial authentication and depth sensing, and it would be straightforward to replace the NIR sensor with one that has higher resolution to support our technique.

Our proposed setup and capture strategy overcomes the major limitations of previous dark flash approaches. In particular:

- It tolerates camera and scene motion, capturing two images simultaneously with parallax, enabling both still and video capture.
- The cameras need not be geometrically identical, enabling a more flexible choice of sensors and lenses.
- It augments existing well-engineered configurations and does not require modifications in any way.

Our goal is to make it practical to photograph people in low light environments such as a romantic restaurant or a concert using a smartphone—at ranges where a dark flash provides a useful amount of illumination. Our camera captures a pair of

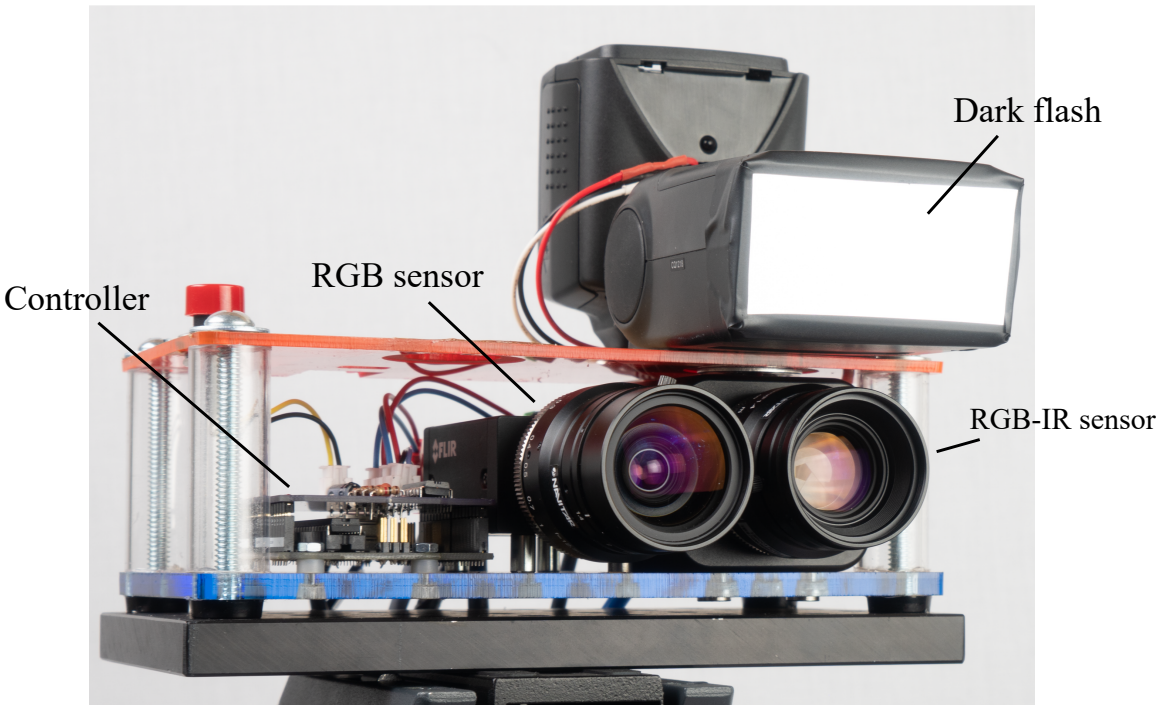

Figure 3-2: Our camera prototype. Notice that when capturing data for training, we remove the flash IR-pass filter (shown mounted in this picture) to permit the capture of full four-channel RGB-NIR images. While we use an RGB-NIR sensor for capturing training data, a monochromatic NIR sensor (augmenting a conventional RGB camera) is all that is required at test time. Similar components already exist in modern smartphones for purposes such as facial authentication.

synchronized images, illuminated by the ambient environment in visible wavelengths but brightly lit by our dark flash in near-IR (NIR) wavelengths. The captured NIR image is sharp with low noise, but also has a characteristic "waxy appearance" on skin tones. In contrast, the RGB image, taken with a longer exposure time, has the right chrominance information but is blurry or noisy. We formulate the combination of sharp near-IR with blurry RGB as an extension to *guided image colorization*, which we solve using a generative adversarial network (GAN) trained on a synthetic dataset and moderately-sized corpus of ground truth images collected with a hardware prototype.

3.2 Related Work

Imaging in low light has been the subject of significant research. Here we provide a short summary of recent literature.

Image Denoising and Deblurring Capturing images in dim environments requires a fundamental tradeoff. Short exposure times capture sharp images that freeze motion but suffer from unavoidable shot and read noise. Integrating over a longer exposure time reduces that noise, but the captured images contain spatially varying blur due to scene motion and hand shake. There are numerous approaches that attempt to remove the corrupting noise or blur post-capture. Traditional denoising techniques like BM3D [5] and Non-Local Means [3] are based on image priors such as sparsity (in a transform domain) and self-similarity. More recently, techniques based on deep learning (e.g., FFDNet [38]) and hybrid approaches (e.g., N^3Net [25]) show impressive results on sRGB images. Recent work applies CNNs to directly to RAW images, denoising images [4] and videos [12] shot in very low light.

The alternative approach of deblurring longer-exposure images is in general a harder problem. Unlike noise, which is a physical process that can be modeled accurately using a small number of parameters, what is observed as blur in an image comes from several spatially-varying, scene-dependent sources. Moreover, the source signal is corrupted by noise prior to integration, and is further processed by a non-linear camera pipeline. Recent techniques [32, 17] have made some progress; however, fully-automatic deblurring remains impractical for consumer applications.

Visible and Dark Flash Photography Our work is not the first to observe that one can enhance images captured in dark environments by combining the sharp details of a flash image with the color and tone of an "ambient" (no-flash) image. Early work [7, 24] captures two images in rapid succession from the same viewpoint, where the flash image is lit by a visible xenon source. Followup work in Dark Flash Photography [16] uses the same capture protocol but leverages IR and UV illumination and a modified camera to avoid dazzling the subject. While these techniques produce

high-quality results, they both require a static scene and either dazzles the subject, or requires modifying the camera hardware to be sensitive to IR and UV.

Previous work on dark flash for video sequences [1] uses NIR frames to aid in RGB denoising through a variety of filtering operations. A specialized hardware configuration is required to ensure the cameras share the same optical path.

Recent work [34] alleviates many of these problems by using a stereo configuration, but the approach imposes other requirements. In particular, the two cameras must be geometrically identical and use sensors that share a green channel but differ in their long (red/IR) and short (blue/UV) wavelengths, making it somewhat impractical. Moreover, their capture protocol requires that the two cameras have identical exposure times for explicit stereo matching, limiting the amount of color information that can be gathered from the RGB camera due to the substantial noise introduced by the short exposure. Our work lifts these restrictions, allowing the use of off-the-shelf components and a more flexible capture protocol.

Recent work in multi-view image fusion [33] may also be applicable to the dark flash setting. The approach features a novel method for computing optical flow with a deep neural network specifically tailored for the applications of color transfer, HDR fusion, and detail transfer. However, it is not trained for our task of RGB-guided NIR colorization and it is unclear how well their formulation of optical flow will work in the presence of cross-modal variations, motion blur, and noise.

Finally, there exists prior research in the area of cross-modal image alignment [28]. This method is hand-tuned to be robust to certain features of NIR images such as gradient reversals. Such alignment techniques could be combined with cross-modal denoising [36] for dark flash imaging. This is analogous to the approach of [34].

Image Colorization Fully-automatic grayscale image colorization is a topic that has received much recent attention [39, 11, 19]. By using an RGB guide image, our work is similar to guided colorization techniques that use sparse annotations [20, 41] or exemplar images [8, 37]. However, these techniques work poorly when directly applied to NIR images because NIR images are *not* grayscale images—they do not

feature many of the properties shared by visible-spectrum grayscale and color images, and they are certainly not in the linear subspace of RGB channels [2]. Materials that appear dark at visible wavelengths may be vibrant in NIR, while visible textures can disappear entirely in the infrared (Figure 3-10).

Recent work in colorizing NIR images [22, 30] focus on well-lit outdoor scenes where the infrared signal comes from the sun. These techniques are trained on RGB-NIR datasets which have per-pixel registration captured using specialized cameras. While these techniques also use modern deep convolutional networks, at test time the NIR image is the only input to colorization. This works well for outdoor environments where scene semantics can be easily inferred from the image but does not consider human subjects in low-light indoor environments.

There are also recent papers based on CycleGAN, which use unpaired training data to learn to colorize NIR images [6, 31]. Like other approaches that colorize only based on NIR, these must cope with the inherent color ambiguities described above. Our setup removes such ambiguities by the acquisition of a (degraded) RGB guide image.

3.3 Proposed Setup

Our method takes as input two images: one reference near-IR (NIR) image illuminated by a dark flash and one noisy and/or blurry RGB image from an offset viewpoint that will act as a color guide. Our goal is to output a sharp RGB image from the viewpoint of the NIR camera. We choose the NIR viewpoint because this is the one where we have sharp information, albeit with an intensity that is very different from grayscale obtained from RGB, which makes our problem harder than traditional colorization. On the other hand, we have some information about color in the guide RGB image, but it has parallax and is blurry, noisy, or both. We seek to train a neural network that combines these two inputs into a sharp RGB image.

We reproduce the illumination conditions of the NIR image because it facilitates the task of the network (no relighting needed), although only up to a point because

near-IR and visible-wavelength reflectance properties can be starkly different. Capturing large amounts of training data with a prototype rig is cumbersome, so we opt instead for a hybrid solution. We first pretrain the network using synthetic data generated from single RGB images, simulating the appearance of NIR intensity, and the degradation of the RGB guide image. We then fine-tune the network with real data obtained from an imaging setup we create. Due to the limited amount of real training data, we have chosen to focus on portraits, as they present a common use-case.

Our proposed camera configuration consists of a conventional RGB camera, augmented by a monochromatic near-infrared camera and a wavelength-matched near-infrared flash (Figure 3-2) whose spectrum is invisible to the human eye. The RGB camera, NIR camera, and flash are offset by a small baseline. The RGB camera has a slightly wider field of view, so that it covers the entire NIR image. At test time, this simple configuration is all that is required. However, to collect a dataset suitable for training an NIR colorization algorithm based on modern deep learning, we need a prototype that can also collect "ground truth".

3.3.1 Prototype camera

For our prototype, we use as the NIR camera a Spectral Devices MSC-RGBN-1-A, which features an RGB-NIR color filter array, enabling us to collect pixel-registered ground truth data. Our RGB camera is a conventional FLIR Grasshopper3 23S6C. As our hardware is a proof of concept designed for gathering training data, we elect to use a standard xenon speedlight for our flash, which emits light in both NIR and visible wavelengths. Although this configuration dazzles the subject, it lets us capture NIR and ground truth RGB from the same viewpoint in a single exposure. The timing of both cameras and the flash is synchronized using a microcontroller.

We also built a second variant for capturing video, replacing the xenon speedlight with an NIR LED array (see Figure 3-8). We use an off-the-shelf assembly of four bright LEDs that is a common accessory for NIR surveillance cameras. Unlike the speedlight, the LEDs emit light continuously, which allows us to capture NIR images at video rate (30 fps). However, since the LED array only emits NIR light, this

configuration does not let us capture ground truth RGB video.

3.3.2 Dataset

Using our camera prototype, we capture low-light portrait photographs of a number of subjects with a diversity of skin color, facial hair, hair style, age, and the presence of accessories such as glasses, earrings, and headphones. During dataset acquisition, subjects naturally vary their amounts of body motion, ranging from "holding still" to rapid movements.

For each scene, we collect four images, with images (1) and (2) acquired concurrently.

1. Input NIR with flash on.
2. Ground truth RGB with flash on.
3. Guide RGB with flash off (blurry long exposure).
4. Guide RGB with flash off (noisy short exposure).

Capture Protocol During a capture session, we continuously present a viewfinder image of the RGB camera, which the operator uses for composition. While in this mode, the shutter time for the long RGB exposure is determined via auto-exposure (AE). We implement AE in software as a PD controller that targets the 90th percentile of the image to be at 60% of the raw range. At our target light levels of 0.1 to 1 lux, the resulting exposure times fall between 100 and 1000 ms.

After shutter release, the RGB exposure time is locked and both cameras wait for a trigger signal from the microcontroller. We first acquire the flash-on NIR and ground truth RGB images in a single exposure with fixed 5 ms exposure time. One millisecond into the exposure, a 5 ms duration flash is triggered via microcontroller. At t = 6 ms, we trigger the RGB long exposure, which guarantees that the flash is no longer illuminating the scene. After the long exposure is completed, we reduce

the exposure time to one tenth of the AE solution and capture another RGB image which trades more noise for reduced blur.

In video mode, the NIR camera is clocked at a constant rate of 30 fps (33 ms duration), leaving 30 ms for exposure and 3 ms for readout. We choose a longer NIR exposure in video mode to compensate for the reduced brightness of the LED compared to the flash. The RGB camera runs at 5 fps (200 ms duration) with a 180 ms exposure time. In video mode, NIR and RGB exposures overlap, with the NIR camera capturing 6 images for every RGB image.

Dataset Details Our full dataset includes 1330 distinct scenes. The RGB-NIR camera has a native raw resolution of 2048x2048 pixels with 12-bit depth. Due to manufacturing tolerances of the RGB-NIR color filter array, only 1024x1024 pixels have well-defined spectral responses. This implicit subsampling introduces aliasing in both NIR and ground truth RGB images. We demosaic the 1024x1024 image into red, green, blue, and NIR channels of size 1024x1024. After this, we run the images through a standard camera pipeline of white balance, exposure, contrast, and saturation adjustment [26], and store the outputs as gamma-encoded 8-bit sRGB images for downstream training and evaluation.

In addition to the still-image dataset above, we also collect two others for evaluation: 21 video sequences, in which we replace the flash with IR LEDs capable of continuous illumination (see Figure 3-8) and low-light images captured by Google Pixel 4 (60 images) and Huawei P30 Pro (38 images) smartphones, which serve as baselines for commercial burst denoising algorithms.

3.4 Method

Our method relies on a deep neural network to reconstruct a sharp RGB image given a near-IR image, using information present in the offset and potentially blurry longer-exposure RGB guide. Modern deep networks require large amounts of representative data to train; yet with only thousands of images collected over a period of months

with our prototype, we need to overcome this data deficit.

Our strategy is to *pretrain* the network on synthetic data generated from a large public RGB portrait dataset. By randomly degrading these images, we can steer the network towards predicting satisfactory results from a "generic" low-quality camera and a grayscale channel that is only loosely correlated with the desired output. In short, the two features we need to replicate in our synthetic data are 1) limited correlation between near-IR and RGB grayscale, and 2) motion blur, noise, and parallax of the RGB guide. Our strategy is not to model these effects accurately (which may overfit a model to a specific setup and calibration), but to instead start from a coarse model using pretraining.

We then fine-tune the pretrained network on actual images from our prototype, adapting it to handle its distinct characteristics.

3.4.1 Synthetic data and pretraining

An appealing aspect of traditional grayscale colorization is the abundance of training data for the problem, as every RGB image can simply be turned into a lightness (input) and two chrominance (output) components. Synthesizing an accurate NIR image using a RGB image is a challenging task since different materials with the same color can produce very different responses in the NIR domain (see Figure 3-9). For pretraining a NIR colorization network, simply using the lightness channel as input is insufficient: the network will quickly learn to pass through the lightness channel to the output feature maps. If we were to apply such a network to actual NIR images, the results will look unnatural due to the aforementioned differences in appearance between NIR and visible intensity images. We narrow this domain gap by approximating the NIR image as semantically weighted averages of the RGB channels.

We use the CelebAHD dataset [14] containing $30,000$ images of faces to curate our synthetic dataset of NIR images, RGB guide images, and target images. To synthesize the NIR image, we semantically blend the color channels. Since different materials have a different response in the NIR modality, we use the semantic maps provided

in the CelebAHD dataset to generate a blending mask where each semantic region has the same blending coefficients. We combine the semantic regions by material (i.e. hair, eyebrows, and facial hair have the same coefficients) to have consistency between same materials. We find this to be a good approximation for a synthetic pretraining task. For the synthetic RGB guide image, we apply motion blur and Gaussian noise with different standard deviation for each of the color channels. We generate a blur kernel of size 101×101 by generating a uniform random walk by integrating a random number of velocities and positions where magnitude and angle of each is uniformly sampled. Motion blur version of the image is generated by convolving the ground truth RGB image with the generated kernel. For the per-channel noise, the standard deviations are sampled from uniform distributions with ranges $(0.003, 0.55)$ for red, $(0.003, 0.27)$ for green, and $(0.003, 1.17)$ for blue. The large range of standard deviations for the blue channel models the comparatively poor SNR of the blue channel in the captured data. Further details and samples of the synthetic dataset are included in the supplementary material and described in Figure 3-3.

3.4.2 Network architecture

We take an image-to-image translation approach to the problem using a conditional GAN model inspired by Pix2Pix and Pix2PixHD [11, 35]. The generator is conditioned on the concatenated input of the NIR and RGB guide image and outputs the sharp RGB image.

Our generator follows a UNet-like architecture as proposed in Pix2Pix [11]. The encoder and decoder have 6 downsampling and upsampling convolutional blocks as shown in Figure 3-4. Each downsampling block increases the number of feature maps by a factor of 2 while reducing the spatial dimension by 2 along each axis. The skip connections concatenate the features from the downsampling blocks in the encoder to the upsampling blocks in the decoder. The upsampling blocks in the decoder reduce the number of input feature maps by a factor of 2 and increase the spatial dimension by a factor of 2 along each axis using a bilinear upsampling layer. As the generator

is fully-convolutional, it suffices to train it at reduced 320×320 resolution.

We use a multi-scale discriminator as proposed in Pix2PixHD [35]. It consists of three discriminators, each with three convolutional blocks. Each of these discriminators are provided with inputs at different scales, specifically $1\times$, $0.5\times$, and $0.25\times$ the original resolution.

We train the proposed model using a weighted combination of L1-loss, least-squares adversarial loss [23], feature matching loss [35], and perceptual loss based on VGG-19 [13, 29]. Following the two time-scale update rule [9], we optimize the generator and discriminator using Adam optimizer [15] with learning rate of `1e-4` and `4e-4` respectively. As an ablation, we train just the generator model using L1, and also L1+LPIPS loss [40]. We observe more realism and contrast in skin tones and hair color when the model was trained with the GAN loss (see Figure 3-5). Please refer to the supplementary material for implementation details.

3.4.3 Training

We first train the model with a batch size of 8 on the synthetic dataset for 16000 steps. The trained model is then finetuned for 2500 steps using our real dataset with ~ 900 samples collected using our proposed camera setup. The training of our model takes approximately 8 hours on a single Titan RTX.

3.5 Results

3.5.1 Low-light portrait photography and comparisons with baselines

Figure 3-6 compares our algorithm to multiple baselines. For all examples, we provide the output of a recent image colorization system [41]. To give the colorization baseline the best chance of succeeding, we chose a method that accepts additional guidance from a color histogram of the guide image.

It is clear from the results that even with additional hints, grayscale colorization

does not generalize well to NIR. Compared to colorization, our algorithm produces more accurate colors, especially when NIR intensity differs substantially from the ground truth grayscale intensity.

We also compare to BM3D [5] for RGB denoising and DeblurGANv2 [17] for RGB deblurring. Since our dataset contains RGB guide images with a range of exposure times, there is a corresponding tradeoff between noise and motion blur.

Figure 3-6 compares our technique to the baselines on images in both regimes, applying denoising where noise is dominant (short exposures) and deblurring where blur is dominant (long exposures). For BM3D we use an empirically determined optimal σ value of 0.03. Neither denoising nor deblurring is able to effectively handle the challenging capture conditions and noise or blur remains visible in the output. In contrast, our results exhibit neither the noise nor the blur of the guide images.

In general we observe that our algorithm performs better with longer exposure guide images than shorter exposure. We hypothesize that this is due to the more reliable color information. Additionally, since chrominance is often low-frequency and we do not rely on precise alignment, they mitigate the negative effects of motion blur on the guide image.

Comparison with burst denoising Modern smartphones use burst denoising techniques to take still photographs in low-light without a flash. Accumulating across a burst of frames allows such algorithms to get similar benefit to long exposures, while also allowing inter-frame alignment to reduce effects of motion blur.

We compare against two such systems: Huawei P30 Pro, running proprietary software, and with Google Pixel 4, running Night Sight [21]. We mount each phone in turn next to the NIR sensor in our rig and place the rig on a tripod. This way the comparison is fair (blur is only caused by scene motion, not by camera shake), and the viewpoint is similar to the one in our result. The phone pictures were each taken within a few seconds of ours. For both phones, capture took around 10 seconds.

A few such results are shown in 3-7 with a larger selection in the supplementary material. These burst denoising algorithms perform poorly in the presence of scene

motion. They also frequently struggle with auto-focus in very low light (Figure 3-1, bottom row), a limitation we sidestep by invisibly illuminating the subject in NIR while being tolerant to a blurry RGB guide.

Finally, it is worth noting that techniques such as burst denoising are complementary to our approach. To the extent that they can improve RGB images in dark conditions, they could be used to improve the quality of guide images for our algorithm.

3.5.2 Low-light video

Our method can also be applied to low-light videos. As shown in Figure 3-8, by replacing the flash in our camera with NIR LEDs that provide continuous illumination. In Figure 3-8 we show this modified camera configuration, along with frames from a representative result.

RGB frames are collected at a lower frame rate to better handle dark conditions, and guide images are selected as the temporally closest RGB frame to each NIR frame. For temporal consistency, we apply the method of [18]. See the supplementary material for the videos.

3.6 Limitations and future work

Our prototype camera is a proof of concept, and image quality falls short of recent smartphone cameras except in extremely dark environments. The biggest limitation to our image quality is the RGB-NIR sensor, which exhibits artifacts stemming from the RGB-NIR color filter array: First, the application of the filter forces a 2x subsampling, causing aliasing. The sensor also shows significant amounts of light leaking between neighboring subpixels, causing desaturated colors even after digital enhancement. As a machine-vision camera, its ISP lacks many modern features (e.g., local tone mapping, edge-enhancement, sharpening) that we have come to expect from well-tuned smartphone camera pipelines or raw workflows .

More fundamentally, there are limitations on how much texture and high-frequency

detail we can recover when it is not visible in infrared. Especially on clothing, patterns may disappear entirely in the NIR image. In this case, blur or noise in RGB guide images can prohibit recovering such details. Figure 3-10 shows a particularly difficult case.

Finally, targeting frontal-flash RGB as the output of our algorithm is not ideal—ambient illumination would be preferable. However, our training set is simply not large enough to learn relighting (a challenging task on its own) in addition to changing the spectral appearance of the NIR image, and extracting color information from the RGB guide. We hope that our work will inspire future progress in this direction.

3.7 Conclusion

We present a dark flash algorithm tailored for portrait photography that overcomes the hardware limitations of previous work [16, 34], making it practical to incorporate into modern smartphones. Our learning-based algorithm requires no modifications to phone RGB cameras, adding only one NIR camera and wavelength-matched LED, and is tolerant to scene and camera motion. We demonstrate the feasibility of our approach with two prototype camera variants compare our results to burst denoising methods used in state-of-the-art smartphones as well as several other relevant baselines.

Bibliography

[1] Eric P Bennett, John L Mason, and Leonard McMillan. Multispectral bilateral video fusion. *IEEE Transactions on Image Processing*, 16(5):1185–1194, 2007.

[2] M. Brown and S. Süsstrunk. Multispectral SIFT for scene category recognition. In *Computer Vision and Pattern Recognition (CVPR11)*, pages 177–184, Colorado Springs, June 2011.

[3] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

[4] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[5] Kostadin Dabov, Alessandro Foi, Vladimir Katkovnik, and Karen Egiazarian. Image denoising by sparse 3-d transform-domain collaborative filtering. *IEEE Transactions on image processing*, 16(8):2080–2095, 2007.

[6] Hao Dou, Chen Chen, Xiyuan Hu, and Silong Peng. Asymmetric cyclegan for unpaired nir-to-rgb face image translation. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1757–1761. IEEE, 2019.

[7] Elmar Eisemann and Frédo Durand. Flash photography enhancement via intrinsic relighting. *ACM transactions on graphics (TOG)*, 23(3):673–678, 2004.

[8] Mingming He, Dongdong Chen, Jing Liao, Pedro V Sander, and Lu Yuan. Deep exemplar-based colorization. *ACM Transactions on Graphics (TOG)*, 37(4):47, 2018.

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017.

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015.

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[12] Haiyang Jiang and Yinqiang Zheng. Learning to see moving objects in the dark. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[13] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[14] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[16] Dilip Krishnan and Rob Fergus. Dark flash photography. In *ACM Transactions on Graphics (TOG)*, volume 28, page 96. ACM, 2009.

[17] Orest Kupyn, Tetiana Martyniuk, Junru Wu, and Zhangyang Wang. Deblurgan-v2: Deblurring (orders-of-magnitude) faster and better. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2019.

[18] Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. Learning blind video temporal consistency. In *European Conference on Computer Vision*, 2018.

[19] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision (ECCV)*, 2016.

[20] Anat Levin, Dani Lischinski, and Yair Weiss. Colorization using optimization. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, page 689–694, New York, NY, USA, 2004. Association for Computing Machinery.

[21] Orly Liba, Kiran Murthy, Yun-Ta Tsai, Tim Brooks, Tianfan Xue, Nikhil Karnad, Qiurui He, Jonathan T Barron, Dillon Sharlet, Ryan Geiss, et al. Handheld mobile photography in very low light. *ACM Transactions on Graphics (TOG)*, 38(6):1–16, 2019.

[22] Matthias Limmer and Hendrik PA Lensch. Infrared colorization using deep convolutional neural networks. In *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 61–68. IEEE, 2016.

[23] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. Least squares generative adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2794–2802, 2017.

[24] Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. Digital photography with flash and no-flash image pairs. *ACM transactions on graphics (TOG)*, 23(3):664–672, 2004.

[25] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. In *Advances in Neural Information Processing Systems*, pages 1087–1098, 2018.

[26] Rajeev Ramanath, Wesley E Snyder, Youngjun Yoo, and Mark S Drew. Color image processing pipeline. *IEEE Signal Processing Magazine*, 22(1):34–43, 2005.

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[28] Xiaoyong Shen, Li Xu, Qi Zhang, and Jiaya Jia. Multi-modal and multi-spectral registration for natural images. In *European Conference on Computer Vision*, pages 309–324. Springer, 2014.

[29] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[30] Patricia L Suárez, Angel D Sappa, and Boris X Vintimilla. Infrared image colorization based on a triplet dcgan architecture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 18–23, 2017.

[31] Tian Sun, Cheolkon Jung, Qingtao Fu, and Qihui Han. Nir to rgb domain translation using asymmetric cycle generative adversarial networks. *IEEE Access*, 7:112459–112469, 2019.

[32] Xin Tao, Hongyun Gao, Xiaoyong Shen, Jue Wang, and Jiaya Jia. Scale-recurrent network for deep image deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8174–8182, 2018.

[33] Marc Comino Trinidad, Ricardo Martin Brualla, Florian Kainz, and Janne Kontkanen. Multi-view image fusion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4101–4110, 2019.

[34] Jian Wang, Tianfan Xue, Jonathan T Barron, and Jiawen Chen. Stereoscopic dark flash for low-light photography. In *2019 IEEE International Conference on Computational Photography (ICCP)*, pages 1–10. IEEE, 2019.

[35] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.

[36] Qiong Yan, Xiaoyong Shen, Li Xu, Shaojie Zhuo, Xiaopeng Zhang, Liang Shen, and Jiaya Jia. Cross-field joint image restoration via scale map. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1537–1544, 2013.

[37] Bo Zhang, Mingming He, Jing Liao, Pedro V. Sander, Lu Yuan, Amine Bermak, and Dong Chen. Deep exemplar-based video colorization. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[38] Kai Zhang, Wangmeng Zuo, and Lei Zhang. Ffdnet: Toward a fast and flexible solution for cnn-based image denoising. *IEEE Transactions on Image Processing*, 27(9):4608–4622, 2018.

[39] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.

[40] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018.

[41] Richard Zhang, Jun-Yan Zhu, Phillip Isola, Xinyang Geng, Angela S Lin, Tianhe Yu, and Alexei A Efros. Real-time user-guided image colorization with learned deep priors. *ACM Transactions on Graphics (TOG)*, 9(4), 2017.

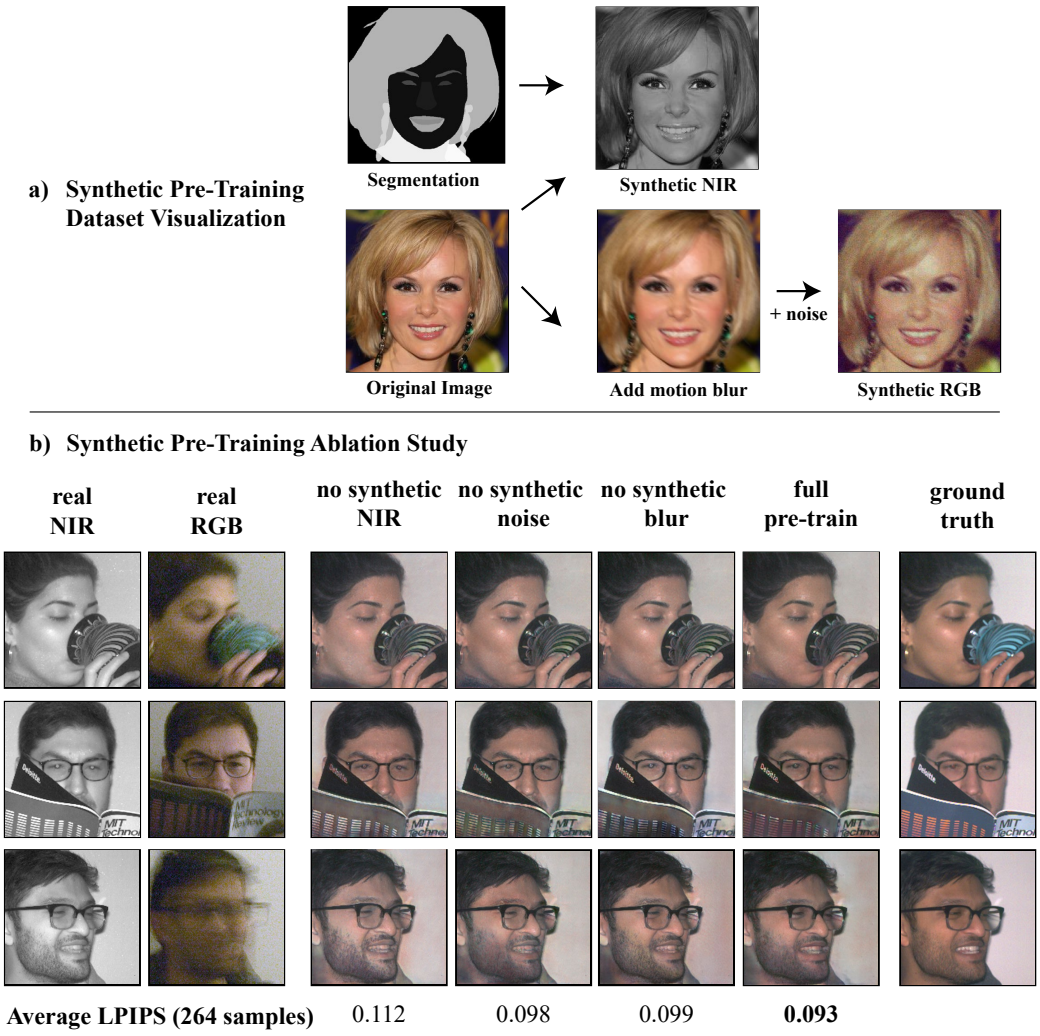

Figure 3-3: **a)** We generate synthetic NIR/RGB training triplets from a single CelebA-HD input image and segmentation mask. For NIR, the RGB color channels of the original are randomly weighted within each segment of the mask and then averaged. For RGB, we first model motion blur using a randomly sampled library of blur kernels. Second, we add Gaussian noise resembling the readout noise of the real data. **b)** We validate the effectiveness of these data augmentations by pretraining a family of models and fine-tuning on real data. We find NIR augmentations to be the most important, as they keep the model from assuming strong correlations between visible and NIR intensity. Synthetic blur and synthetic noise give modest quantitative improvements and perceptually reduce color bleeding.

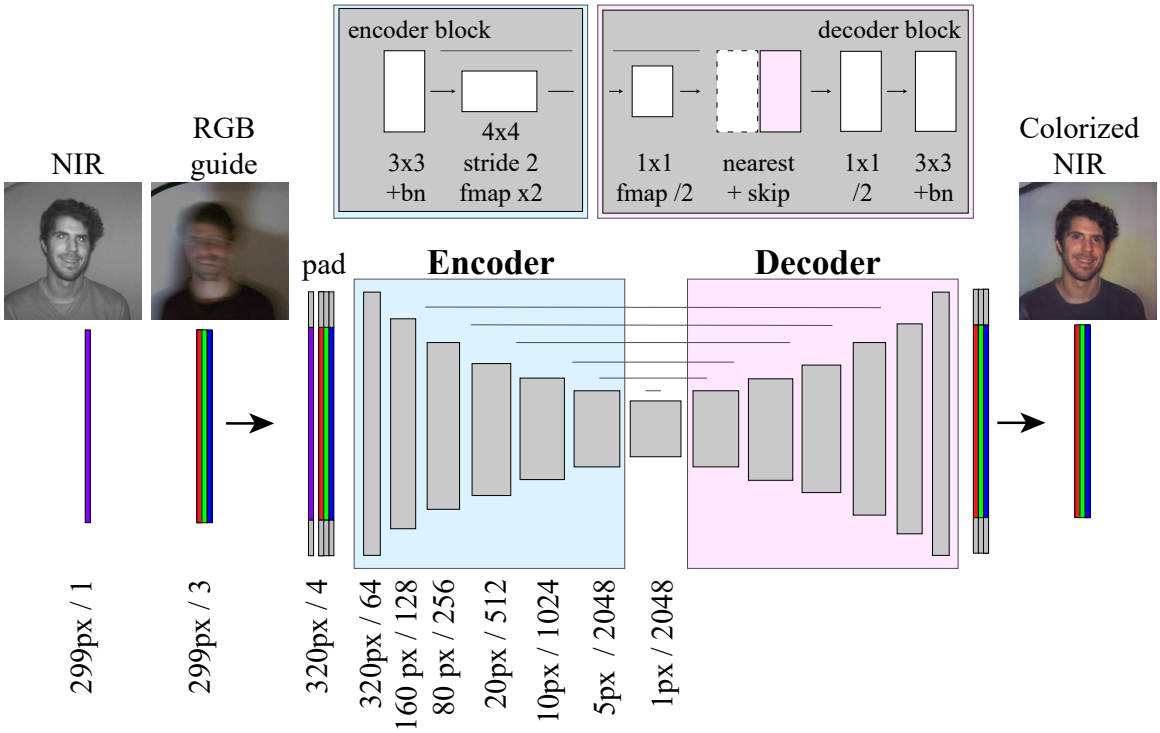

Figure 3-4: Our generator follows an *encoder-decoder with skip connections* architecture inspired by UNet [27]. At each step in the encoder, the spatial size is halved and the number of feature maps doubled. At 5x5 spatial resolution, we use a 5x5 kernel to compression into a bottleneck with 2048 features. Each encoder block (top) alternates size-preserving 3x3 convolutions with batch normalization [10], and size-reducing strided convolutions. In each decoder block, we first halve the number of feature maps with a 1x1 convolution before upsampling and concatenation with the corresponding skip connection. After concatenation, we compress the feature maps and then use a 3x3 convolution that mirrors the corresponding encoder block.

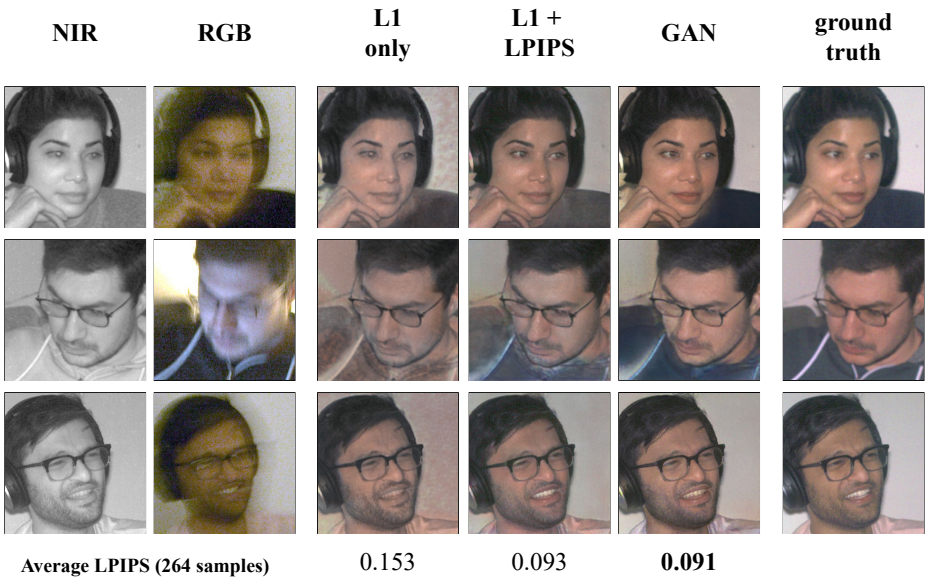

Figure 3-5: Trained with L1 loss, the network fails to output saturated colors [39]. Adding LPIPS gives significant improvements. We achieve the best results with a trained least-squares GAN loss [23] and feature matching loss as proposed in Pix2PixHD [35].

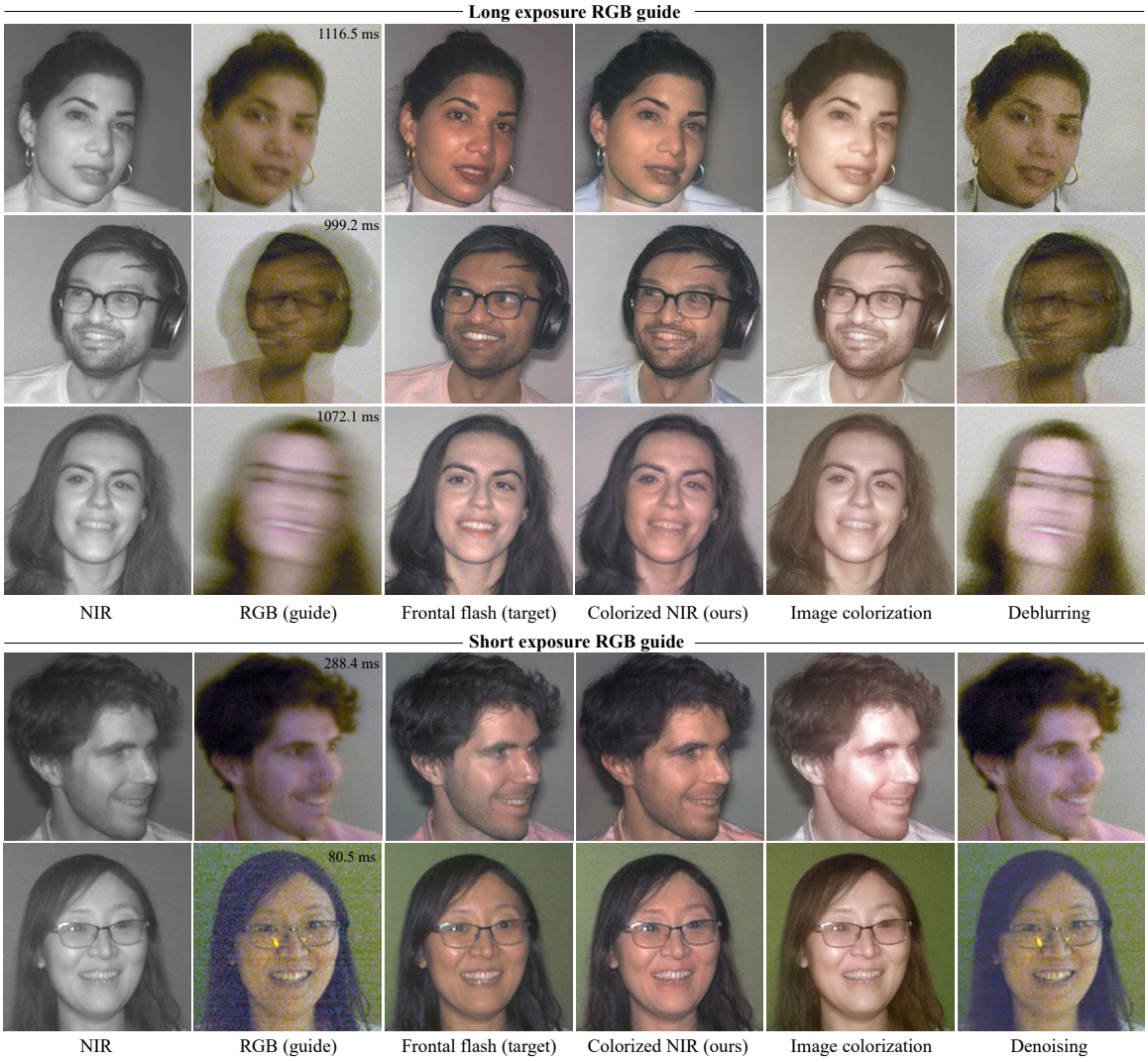

Figure 3-6: More results and comparisons with baselines. While in general we observe better quality results with longer exposures for the RGB guide, we show results for both longer (top) and shorter (bottom) RGB exposures to demonstrate the range of operation of our method. Different scenes may benefit from different exposure times for the RGB guide (e.g., for scenes with large motions, a long exposure RGB guide may be too blurry). The two left-most columns depict the captured NIR and RGB image pair. The exposure time is superimposed on the RGB guide image at the top right corner and determined by AE. The third and fourth columns contain the ground truth and our colorized NIR result, respectively. The fifth column shows the result of state-of-the-art histogram-guided image colorization using the captured RGB image as guide [41] (an upper bound on image colorization). The colorization technique correctly estimates the chrominance of the scene, but is not able to convert NIR intensity to luminance. The sixth column shows: for the long exposure guide: the result of deblurring the RGB image using DeblurGANv2 [17]; and for the short exposure guide, the result of denoising the RGB guide image using BM3D [5]. For all test results, we made sure that no pictures of a given test subject are present in the training set.

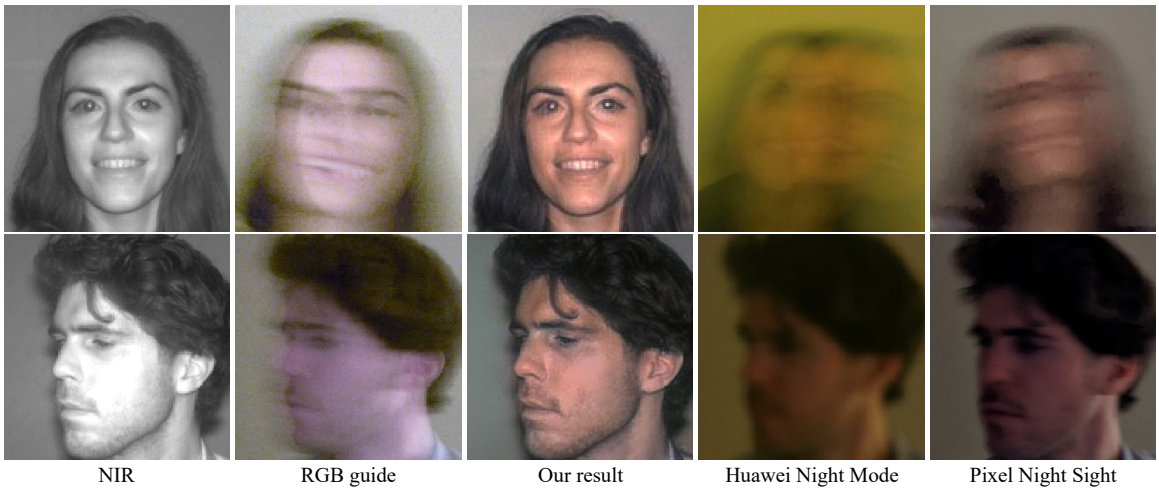

Figure 3-7: Comparisons with state-of-the-art multi-frame denoising methods on mobile phones – Huawei P30 Pro's Night Mode and with Google Pixel 4's Night Sight [21]. More such comparisons can be found it the supplementary material.

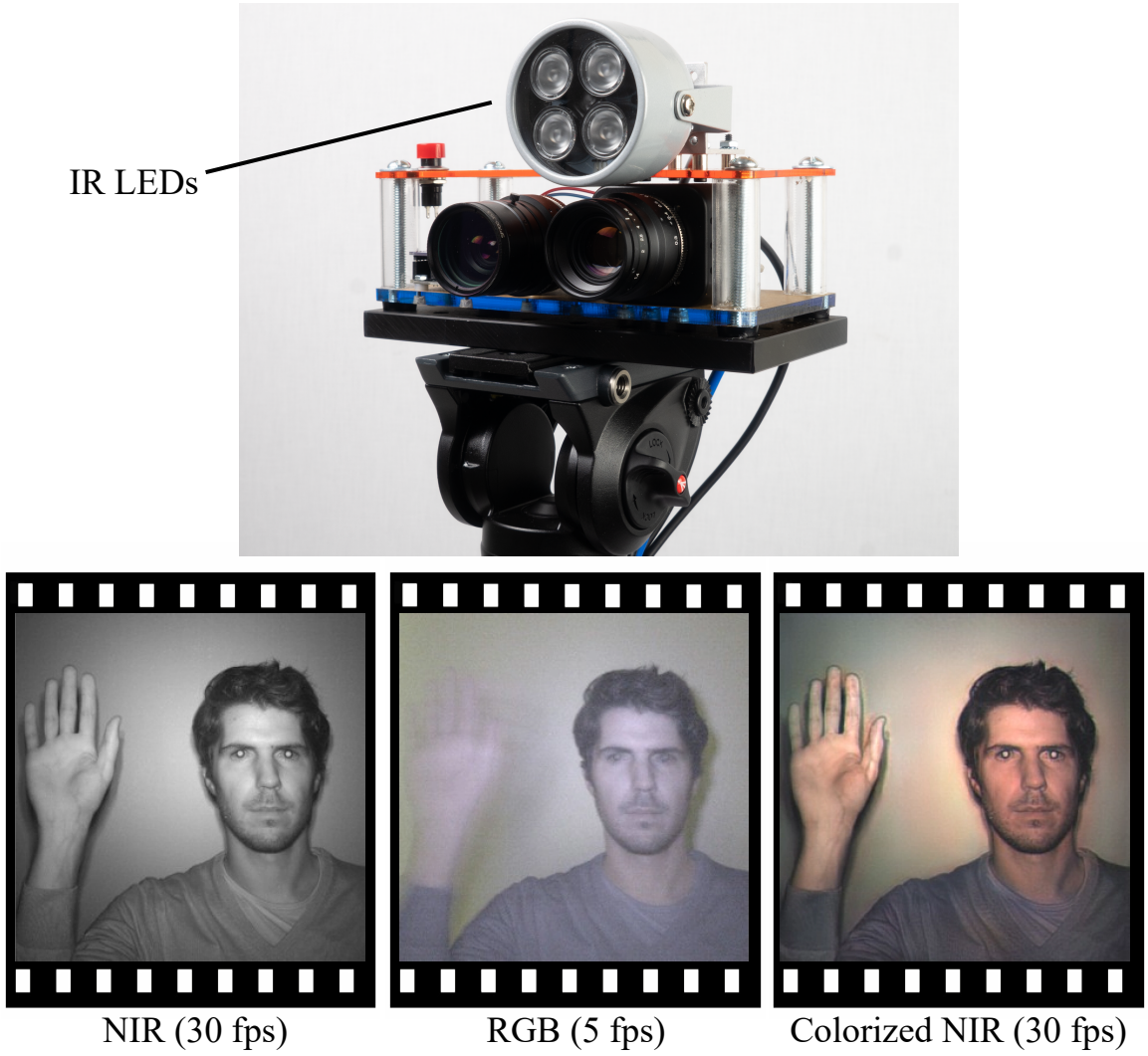

Figure 3-8: Top: Camera configuration for video capture. The flash from Figure 3-2 is replaced with NIR LEDs for continuous illumination. Bottom: One input pair and output frame.

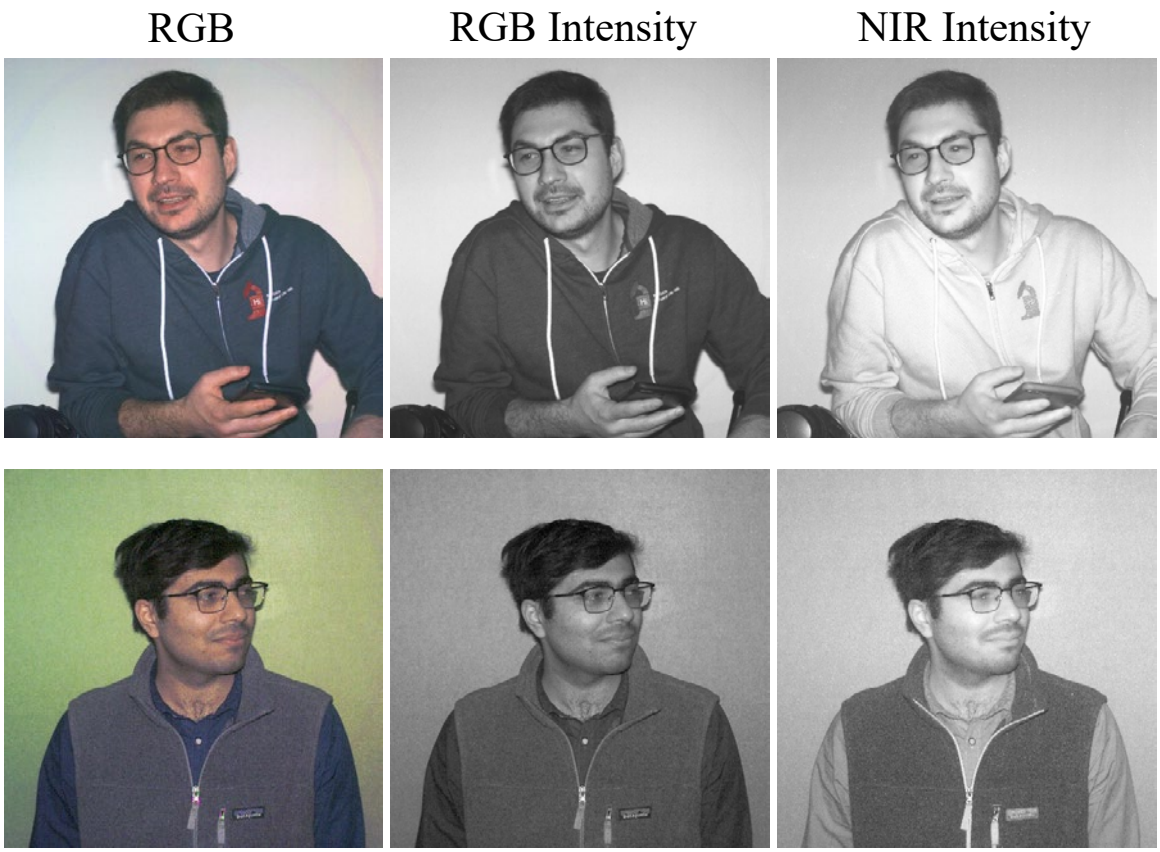

Figure 3-9: Visible surface reflectance and NIR reflectance can differ drastically. Shown in the top (left) is an example of dark blue sweatshirt with correspondingly low intensity in the visible image (center). In NIR, the relative reflectance of this kind of fabric is much higher. However, fabrics are not always brighter in NIR. The zippered vest shown in the bottom is made of a functional fabric with high NIR absorption, leading to a remarkable reversal of relative brightness of shirt and vest.

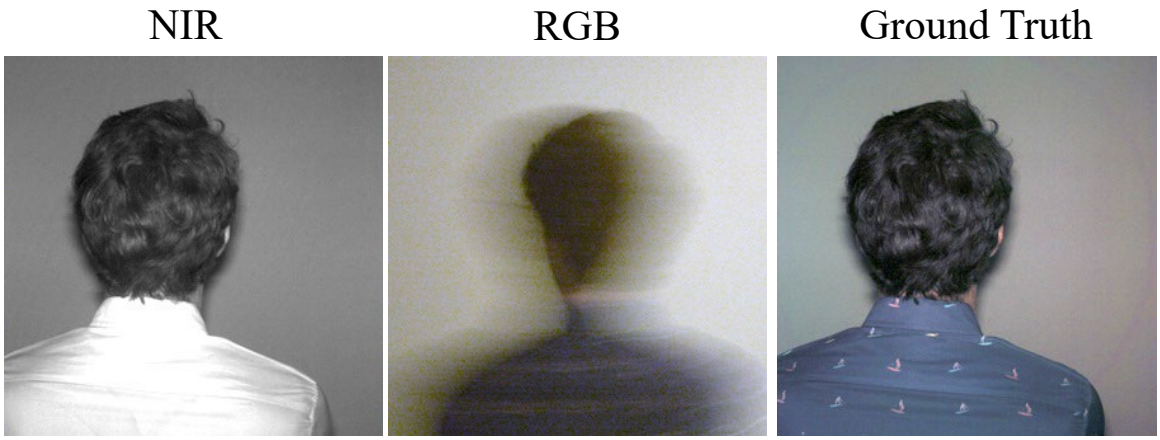

Figure 3-10: Some materials have visible patterns but uniform NIR reflectance. For these materials, the reconstruction problem is equivalent to plain de-noising or de-blurring. Since our network is trained to extract high frequency information from the NIR image, it is not able to reconstruct textures that are neither visible in the NIR nor the RGB image.

Chapter 4

Efficient and Accurate Differentiable Rendering with Analytical Anti-aliasing

Authors:
Lukas Murmann, Tzu-Mao Li, Jonathan Regan-Kelly, Frédo Durand

Author Affiliations:
MIT CSAIL

this chapter is unpublished as of May 2021

Abstract: We propose a new real-time differentiable rendering method that is both more accurate and more efficient than previous rasterization-based methods. Instead of blurring the forward rendering process or using stochastic ray tracing, we closely follow the semantics of conventional real-time renderers and efficiently leverage the rasterization hardware. We use analytic anti-aliasing to ensure that the rasterization step is differentiable. We construct a dynamic deep framebuffer using rasterization hardware to store a per-pixel list of triangles overlapping the pixel filter. We then blend over the triangles using alpha compositing. Our formulation is differentiable,

maps efficiently to the hardware graphics pipeline, and renders high-quality images.

We implement our method using modern graphics APIs and provide a PyTorch interface. Our experiments show that our method is an order of magnitude faster than previous state-of-the-art differentiable rendering methods, achieves high accuracy in various inverse rendering tasks, and is more general than previous rasterization-based methods.

4.1 Introduction

Differentiable rendering is becoming increasingly important in computer vision and machine learning, for both training 3D neural networks and inverse rendering. A central challenge of differentiating rendering is differentiating through the discontinuous visibility function, while maintaining high efficiency of the rendering pipeline. We develop a real-time differentiable rendering pipeline that can correctly differentiate visibility, while generating high-resolution and artifact-free images in milliseconds by utilizing the rasterization hardware.

Current differentiable rendering approaches trade off between accuracy and performance. On the one hand, rasterization-based approaches usually approximate the rendering pipeline, either by softening the forward rendering operator [13, 44, 30, 12, 43] or approximating the gradients [32, 23]. This leads to inferior image or gradient quality, and does not necessarily lead to efficient rendering pipelines. On the other hand, stochastic ray tracing approaches compute accurate gradients by applying Monte Carlo sampling [26, 54, 33, 3]. This allows them to compute high-quality anti-aliased images along with distributed effects and global illumination. However, ray tracing methods require rebuilding acceleration data structures for each gradient iteration, and they are often more computationally expensive. The inherent stochasticity of these techniques also complicates neural network training.

We propose a new differentiable renderer that is both significantly faster than prior systems, and more accurate than previous rasterization-based approaches (Figure 4-1). We achieve this by closely following the rendering model of conventional real-

time renderers, while extending it to be differentiable without sacrificing accuracy or hardware acceleration. Instead of blurring the rendering process or stochastically evaluating gradients, we employ a differentiable analytical anti-aliasing technique. During hardware-accelerated rasterization, we efficiently construct a *deep framebuffer* — a dynamic data structure that stores the list of triangles overlapping each pixel. For each pixel, we then blend all the overlapping triangles using alpha compositing, where the opacity is computed by analytically integrating a triangle with the pixel filter kernel.

Our method is deterministic and parameter-free, produces high-quality images, and can be efficiently implemented on modern graphics APIs. Our method is more flexible and general than previous rasterization-based renderers. It supports common real-time rendering primitives including triangle meshes, transparent billboards, mipmapped textures, and alpha maps. We implement our method in DirectX 12 and provide a PyTorch interface. We experiment with various inverse rendering tasks, and show that our differentiable renderer achieves lower error in significantly shorter time, compared to previous state-of-the-arts.

4.2 Related work

Differentiable rendering. Early computer vision and graphics works developed specialized differentiable renderers that ignore visibility gradients [6, 40, 18, 17, 28, 29, 2]. More recently, general differentiable renderers were developed. To differentiate the discontinuous visibility, some approaches approximate the gradients using post-processing [32] or edge rasterization [23]. Other approaches instead *soften* the forward rendering process to make it differentiable [44, 30, 43, 12].

In contrast to approximated methods, we develop a differentiable rendering pipeline that fits well with the rasterization hardware, is capable of producing high-quality images, while being directly differentiable. We employ a differentiable anti-aliasing technique based on analytical coverage. Anti-aliasing has been used in differentiable rendering methods before. however, previous methods either ignore occlusion [46, 20],

or assume that we are rendering a single, closed object [13]. Instead, we efficiently blend over multiple primitives in a pixel using a visibility-aware alpha compositing, in an order-independent way [50].

Ray tracing based methods [26, 33, 54, 53, 3] compute accurate gradients and high-quality rendering using stochastic sampling. However, they require expensive ray tracing acceleration structures construction. For local shading and 3D scenes with moderate complexity, rasterization is more efficient than ray tracing thanks to the streamlined access to the scene data. In contrast, our method is deterministic and leverage the rasterization hardware.

Other differentiable rendering works target for point clouds [52, 43] or implicit functions [39, 45, 21]. Instead, we focus on primitives that are common in real-time graphics pipelines, including textured meshes and billboards.

Concurrent to our work, Laine et al. [25] developed an efficient differentiable rasterizer. Instead of our analytical anti-aliasing method, which corresponds to a box-shaped reconstruction filter, they approximate box filtering with an edge-aware image-based anti-aliasing method [34, 22]. Their method is efficient thanks to the Z-buffer and hardware occlusion culling. However, their method considers only a single fragment per pixel, which sacrifices silhouette gradient quality (see Figure 4-13), limits gradient propagation to a single triangle per pixel, and does not support order-independent drawing of transparent surfaces.

Real-time rendering and 2D graphics. Analytical integration has been used for solving anti-aliasing since the emergence of computer graphics [10, 15, 35, 1]. Instead of the expensive operations of clipping polygons against each other these methods used, we use a coverage-based blending method that produces high-quality images while avoiding clipping. Our blending method is related to some 2D vector graphics rasterization anti-aliasing approaches [38, 4, 27], but 3D rendering requires per-pixel triangle sorting. Our deep framebuffer is related to the A-buffer [9], deep shadow maps [31], deep compositing in visual effects [16] and order-independent transparency [5, 51, 50].

4.3 Method

Our differentiable rasterizer design is mainly driven by three goals. First, our rasterizer should produce high-quality, anti-aliased, artifact-free images for a well-defined pixel reconstruction filter (box filter). Second, our rasterizer should efficiently leverage the rasterization hardware. Third, our rasterizer should be deterministic and directly differentiable using analytical or automatic differentiation. We focus on local shading and do not specially handle shadows, global illumination, or distributed effects. We make the following design choices:

- To address the discontinuous visibility, we analytically compute the coverage of each triangle, and multiply the coverage with the shading color (Figure 4-3).

- To efficiently rasterize the triangles, we use a deep framebuffer to sort the rasterized triangles (Figure 4-5). This allows us to rasterize the triangles in an order independent way, where each pixel efficiently sort the triangles locally.

- We composite multiple triangles using alpha blending [41], where the opacity of the triangles are determined by their coverage (Figure 4-6).

As a consequence of the analytical coverage computation, our forward model is directly differentiable and can be implemented using rasterization passes and compute shaders (Figure 4-2). We can then efficiently backpropagate the gradients with respect to all the scene parameters. Furthermore, our rendering pipeline allows us to apply flexible shading models, including UV-mapped textures lookup, mip-mapping, alpha maps and transparent objects. In particular, transparency is not allowed in differentiable rendering methods that employ deferred shading [25] with only a single fragment per pixel.

In the following we describe our rendering model and how we efficiently implement both the forward and backpropagation passes.

4.3.1 Computing Fractional Visibility using Analytical Coverage

A key challenge of inverse rendering is to differentiate the discontinuous visibility. In both ray tracing and conventional rasterization, the visibility of surfaces along camera rays is tested only for discrete point samples. The pixel center either intersects the tested geometry (visible) or it doesn't (invisible). While this discontinuous formulation is acceptable for forward rendering, it makes backpropagation infeasible, as visibility gradients are nonzero only along an infinitesimally narrow band along the silhouette of the geometry. Although prior work showed how to estimate high-quality visibility gradients by explicitly sampling the silhouette [26] with ray tracing, it is not applicable to rasterization, where camera rays are constrained to a fixed grid pattern.

We propose a differentiable formulation of rasterization that evaluates fractional visibility within the footprint of a single pixel, rather than binary visibility at just the pixel center (Figure 4-3). We follow the insights of previous work that anti-aliasing makes rendering differentiable, as the pixel color changes smoothly with respect to scene parameters after the anti-aliasing integral [13, 26]. However, solving the full anti-aliasing integral turns out to be intractable, as it requires inter-triangles clipping that is both time-consuming and numerically unstable [10, 48]. Instead, we approximate the anti-aliasing integral in three ways. First, we assume the anti-aliasing filter to be a box filter. Second, we avoid inter-triangles clipping by only clipping triangles against the pixel boundary. Third, we enable complex shading models by decoupling the coverage and shading. Specifically, given a triangle, we write down our anti-aliasing integral as an integral over the visibility V and shading:

$$\begin{aligned}\int_A V(x) \cdot \text{shading}(x) dx &\approx \int_A V(x) dx \cdot \text{shading}(x_s) \\ &= \text{coverage} \cdot \text{shading}(x_s),\end{aligned} \tag{4.1}$$

where A is the pixel filter support, and we compute the *coverage* $= \int_A V(x) dx$ using analytical integration, and approximate the shading integral using a point sample on the triangle x_s. In our implementation, we choose x_s as the point on the triangle that

is closest to the pixel center.

We process triangles overlapping with the pixel from front to back and blend the contribution using the *coverage* (Figure 4-3, right). The current triangle is treated as the foreground, and the triangles behind are treated as the background. To compute the analytical coverage, given a triangle, we first clip the triangle against the pixel filter support. The *coverage* can then be computed as the area of the polygon. In the subsequent subsections, we will discuss how we efficiently enumerate, sort and blend contributions from multiple triangles.

Our analytical anti-aliasing enables high-quality images and gradients compared to the pixel-center sampling or blurring scheme used for previous differentiable rendering works (Figure 4-4), and is completely parameter free. In the Appendix, we show that, in 1D, the spatial smoothing of soft rasterization method [30, 43] can be seen as an analytical integral approximation, with a rather uncommon pixel filter kernel $k(x) = \frac{2e^{-x^2}|x|}{\left(1-e^{-x^2}\right)^2}$ (note that $k(0) = 0$). Furthermore, as shown in Figure 4-4, their depth compositing causes undesired cracking artifacts. Importantly, the same depth compositing strategy is used by many other differentiable rasterizers [12, 43]. Our formulation allows us to better analyze and compute the integrals, enabling higher image quality. Compared to ray tracing approaches, our method is more efficient and deterministic.

4.3.2 Handling Geometric Complexity with a Deep Framebuffer

Rasterization methods using Z-buffering find the triangle that is closest to the camera for each pixel, which is then shaded and blended into the color buffer. Alternatively, shading may be *deferred*, by writing the closest triangle id and barycentric coordinates to a Geometry-Buffer. In a second render pass, the samples in the geometry buffer are shaded and written to a color buffer. Either approach only stores *one sample per pixel*, even if multiple triangles overlap that pixel, as shown in Figure 4-5.[1]

[1]Multi-sample or temporal anti-aliasing capture multiple triangles in a pixel by averaging contribution, but are not directly differentiable.

Rendering with one sample per pixel comes with significant drawbacks for inverse rendering: If only a single triangle contributes to the color of each output pixel, then back-propagation can only yield non-zero gradients for that single triangle. Storing only one sample per pixel is particularly limiting when triangles are small, such as for high-resolution geometry, or along the silhouette boundaries that are crucial for visibility gradients [25].

Our renderer stores all triangles that contribute to a pixel in an irregular *deep framebuffer*. We store the number of overlapping triangles and a pointer to a list for each pixel, where the list stores the ID of the triangles overlapping with the pixel filter. Figure 4-5 shows an example for a simple scene with a tessellated triangle. The deep framebuffer lets us correctly anti-alias fractionally visible geometry, and also allows us to render advanced effects such as transparency. To efficiently leverage the rasterization hardware, we construct the deep framebuffer in a two-pass process, rasterizing the scene twice. In the first pass, we count the number of overlapping triangles per-pixel without storing them. In the second pass, we allocate the buffers based on the counts, and append the triangle IDs and barycentric coordinates into the lists. We use the conservative rasterization feature in modern graphics hardware to include triangles not overlapping the pixel center. Our efficient implementation of the deep framebuffer is inspired by earlier works on A-buffering [9], deep shadow mapping [31], and order-independent transparency [5].

4.3.3 Deep Pixel Compositing Enables High-Quality Anti-Aliasing and Transparent Surfaces

After triangle samples are stored in the deep framebuffer, we independently compute the blended output color of each pixel, using the over compositing (or alpha blending) operation [41]. To handle both transparency and coverage-based anti-aliasing, we keep track of two quantities per sample during the traversal: *alpha* and *visibility*. *alpha* represents the fraction of light or color reflected by a potentially transparent triangle sample in a pixel, where *visibility* represents the fraction of the pixel covered by the

triangle, while taking other triangles' occlusion into account. *visibility* is different from the analytical *coverage* in Equation 4.1, which is the coverage of the individual clipped triangle without taking other triangles into account.

To compute the two quantities, first, we sort a pixel's samples in a front-to-back order. Next, we loop through the fragments front-to-back. The fraction of transmitted light *alpha* by each triangle sample i is computed as the product of the *visibility* and the *opacity* of the sample i:

$$\text{alpha}_i = \text{visibility}_i \cdot \text{opacity}_i. \tag{4.2}$$

Computing the exact fractional *visibility* for a sample can be prohibitively expensive, as it requires inter-triangles clipping. Instead, we make the assumption that background triangles are *as-visible-as-possible*, contributing visibility until a total of 100% visibility is reached for the current pixel. Our approximation allows us to include as many triangles as possible, thus efficiently distributing the gradients:

$$\text{visibility}_i = \min(1 - \sum_{j=0}^{i-1} \text{alpha}_j, \text{coverage}_i), \tag{4.3}$$

where coverage_i is the analytical coverage of sample i (Equation 4.1).

The front-to-back traversal ends once the sum of *alpha* reaches 1.0, which marks the point at which background fragments become fully occluded. If all fragments sum to *alpha* less than one (either due to partial coverage, or due to transparency), we assign the remaining *alpha* to the background color. Figure 4-6 shows a visualization of an example.

Finally, we compute the blended pixel output I as the weighted sum of the shading evaluation and per-sample *alpha*:

$$\text{I} = \sum_{i=0}^{\text{num\_samples}} \text{alpha}_i \cdot \text{shading}_i, \tag{4.4}$$

. All of the sorting and compositing is done in a single compute shader (Figure 4-2),

efficiently caching the intermediate results into registers instead of global memory.

Shading. In principle, our method supports arbitrary differentiable shading, including physics-based bidirectional reflectance distributions [8], convolution of spherical harmonics coefficients between reflectance and environment map interpolated from triangle vertices [42], or neural-network-based reflectance models [37].

Our current implementation includes Lambertian shading with ambient, directional, and point light sources. Surface reflectance and alpha is specified either per-vertex or in mipmapped [49] texture maps. We reconstruct the texture values using the nearest mipmap level with bilinear interpolation.

4.3.4 Differentiating our rendering pipeline

Our forward rendering model is fully differentiable, therefore, in principle, it can be directly differentiated with automatic differentiation [19]. In practice, to better fuse the computation in shaders and reduce global memory usage, we manually implement the gradient passes.

As shown in Figure 4-2, after the forward pass has finished, We hand the computation to PyTorch where the user computes the loss and initiates the backward pass. The goal of the backward pass is to compute the derivatives $\frac{\partial E}{\partial \theta}$ of loss E w.r.t. scene parameters θ. These derivatives factor as $\frac{\partial E}{\partial \theta} = \sum \frac{\partial E}{\partial I} \frac{\partial I}{\partial \theta}$ where I is the pixel color. PyTorch provides us with $\frac{\partial E}{\partial I}$. We need to backpropagate through our renderer to compute $\frac{\partial I}{\partial \theta}$ and accumulate the gradients $\frac{\partial E}{\partial \theta}$ to the corresponding scene parameters.

Backward shading. Backpropagation of the alpha compositing process (Equation 4.4) involves traversing the samples in the deep framebuffer in the back-to-front order. The main non-trivial derivative is the pixel coverage derivative $\frac{\partial \text{coverage}_i}{\partial T_i}$, where T_i is the position of the triangle vertices, since our forward model clips the triangle and introduces dependencies with sparse structure between the clipped polygon vertices and the original triangle vertices. We can exploit the sparse dependency to make the derivative computation efficient. We first compute the derivative of the coverage with

respect to the clipped polygon vertices P_j to compute $\frac{\partial \text{coverage}_i}{\partial P_j}$, then compute the Jacobian $\frac{\partial P_j}{\partial T_i}$ between the the clipped polygons P_j and the original triangle T_i. The Jacobian for the partial derivatives $\frac{\partial P_j}{\partial T_i}$ has a particular sparsity structure that makes evaluating the derivative efficient (Figure 4-7). We can then combine the derivative using standard chain rule: $\frac{\partial \text{coverage}_i}{\partial T_i} = \frac{\partial \text{coverage}_i}{\partial P_j} \frac{\partial P_j}{\partial T_i}$. A sufficiently sophisticated automatic differentiation compiler should be able to efficiently compute the derivatives above, but we leave such compiler as future work.

Gradient Accumulation. To efficiently accumulate the gradients into the corresponding scene parameters, for each triangle sample i, we compute gradients for the three vertices, and write them to per-vertex lists in GPU memory. Similarly, we write four texture gradients to per-texel lists of the active MIP level. After vertex gradient accumulation, we compute the gradients of camera intrinsics and object transforms. For the mipmapping gradients, we backpropagate through the mipmap construction process by upsampling the coarse-level gradients to the fine-levels. All our operations involve minimal atomics and can be efficiently parallelized on modern GPUs.

Mipmapping We show that mipmapping [49] is crucial for high-resolution texture reconstruction. In Figure 4-8, a scene with a texture that has higher-resolution than the rendering resolution is shown. Without mipmapping, only a subset of texels is observed, and even minimal changes of the camera pose reveal cracks in the reconstruction. With mipmaping, the texture receive gradients at the appropriate scale, which are then propagated to the highest resolution layer. This results in artifact-free inverse texturing. See Figure 4-8.

4.4 Results

We implement our differentiable renderer using DirectX 12, and provide a PyTorch interface. Our renderer is capable of rendering 1k resolution images with thousands of triangles in 10 to 20 milliseconds, around 5 to 10 times faster than previous

rasterization-based methods [30] and more than 15 times faster than previous ray-tracing-based methods [26]. Across scenes with different complexities, our renderer typically spends around 60% to 70% of time in the forward pass, which includes setting up the deep framebuffer using rasterization, computing coverage, and blending the contribution, and the rest in the backward pass. We provide a timing breakdown of our method in Figure 4-9.

Inverse rendering. To show the better accuracy and efficiency of our method, we compare it to previous methods in inverse rendering tasks. Figure 4-1 and Figure 4-10 show comparisons between our method, the ray-tracing based differentiable renderer REDNER [26], the rasterization based differentiable renderers *SoftRas* [30] and *PyTorch3D* [43]. We optimize for the target poses from an initial guess using Adam [24] with learning rate 0.02 for all methods. For Figure 4-1 we render the target with the corresponding renderers. For Figure 4-10 we use Blender as the reference. Our method converges to the lowest error for both cases in significantly shorter amount of time. Compared to REDNER, our deterministic gradient leads to slightly better convergence in equal-step-comparison. Compared to SoftRas and PyTorch3D, our gradients are less affected by the artifacts generated in the forward pass (Figure 4-4). For example, at the top region of the can in Figure 4-1, SoftRas generates high-frequency artifacts that hurt convergence. Similarly, both SoftRas and PyTorch3D generate artifacts for the smooth regions in Figure 4-10. We further compare the gradients in Figure 4-11. Our renderer is significantly faster than all three previous methods since we better leverage rasterization hardware. We are capable of rendering and differentiating 1024×1024 images in 21 and 16 milliseconds respectively for the two scenes with thousands of triangles, which is 5 to 8 times faster than SoftRas and more than 15 times faster than both REDNER and PyTorch3D. All the results in the paper were run on a RTX 2060 Super for our method and REDNER, and were run on a Tesla P100 for SoftRas and PyTorch3D.

The tightly packed deep framebuffer allows us to compactly store information. For the scene in Figure 4-10, our method's memory consumption peaks at around 970MB,

while SoftRas takes at around 993MB, and PyTorch3D takes around 11997MB (for PyTorch, we set a maximum of 32 faces per pixel to limit GPU memory consumption).

Pose estimation. Our renderer achieves high performance even for complex models. In this example, we shown an optimization of a model with over 20.000 triangles and significant depth complexity at 512x512 pixel resolution. We use stochastic gradient descent (step size 0.02) for 400 steps (12.5ms per step for a total of 5s until convergence). See Figure 4-16.

Comparison to Laine et al. [25]. Our analytical anti-aliasing and compositing allows us to better sample the silhouette contribution of the visibility gradients. We compare to Laine et al.'s differentiable rendering method [25] based on image-space anti-aliasing in Figure 4-13. Laine et al. only sample a single triangle per-pixel, which misses the boundary gradients for pixels containing small triangles or when rendering low-resolution images, which are important for coarse-to-fine optimization.

Transparency. Our renderer naturally supports rendering with transparency (Figure 4-12). This allows us to optimize over scene representations popular in image based rendering, such as multiplane images [47, 55] or billboards/impostors [14]. Differentiable renderers that use Z-buffering or deferred shading [32, 25] are limited in their support for transparent surfaces, as they assume the closest triangle is the only contributing source and cull all but front-most triangle. SoftRas does not include transparency in their rendering model, instead blending contribution using depth [36]. In contrast, our method supports both opaque and transparent surfaces, including surfaces defined through alpha texture mapping.

Extending multi-plane images with different geometries. The ability to differentiate transparent object rendering allows us to generalize multi-plane images [55] to use different proxy geometry for image-based rendering. In multi-plane images, geometry and shading is represented by layers of alpha-blended planes. This representation, however, cannot effectively reconstruct the viewing directions parallel to the

planes. Using our renderer, we can render *Grid MPI*s (a stack of planes along each of the three axes) and *Sphere MPI*s (concentric spheres) as our proxy geometries. In Figure 4-17, we show results of fitting traditional multi-plane images (*Quad MPI*), as well as the other two types of geometry. Compared to the quad geometry, both the grid and the sphere representations improve the reconstruction quality and generate artifact-free 360 degree views of the scene.

We show results of fitting the renderings of 20 semi-transparent parallel planes (*Quad MPI*), a $10 \times 20 \times 8$ grid (*Grid MPI*), and 16 concentric icosahedrons (*Sphere MPI*, which is subdivided twice and represented with 120 triangles) to 120 views. We map a 256×256 texture onto the parallel plane, three 256×256 textures onto the grid, and 1024×1024 onto the sphere using latitude-longtitude coordinates. Compared to the plane geometry, both the grids and the sphere representations improve the reconstruction quality and generate artifact-free 360 degree views of the scene. For all three setups we run Adam [24] for 50 steps using a learning rate of 0.1, where each step involves rendering and differentiating all 120 views. For the parallel planes optimization takes around 30 seconds, and for the grid and the spheres it takes around 3 minutes. Both the grid and the concentric sphere geometries show significant improvement over the original plane geometry in terms of the reconstruction error.

Non-rigid shape alignment. In Figure 4-15, use our renderer to fit a non-rigid body to a target pose from the FAUST dataset [7].

Photometric object tracking. In Figure 4-14, we show the result of optimizing for the pose and reflectance of a simple object in a photograph. Figure 4-18 shows a result of us tracking the 3D position of a basketball, and that of a cardboard box over multiple frames. At each frame, we run 50 iterations of Adam (lr=0.02, 10ms runtime per step) which is a sufficient number of steps to solve the incremental tracking problem between pose i and pose $i + 1$.

Limitations. While our method can efficiently leverage rasterization hardware, we do not use the Z-buffer for culling occluded triangles. This can lead to inferior per-

formance when the depth complexity of the 3D scenes is high. On the other hand, it is necessary to turn off Z-buffer and occlusion culling for supporting transparency in rasterization. For meshes with hundreds to thousands of triangles, such as the ones in the ShapeNet database [11], our method is very efficient.

4.5 Conclusion

We introduce a new rasterization-based differentiable renderer that is both more accurate and faster than previous methods. We achieve this using a differentiable analytical antialiasing technique, which integrates over triangle area to compute coverage, and blend contributions using a visibility-aware alpha compositing. We leverage rasterization hardware to construct a deep framebuffer data structure for collecting triangles for each pixel. Unlike other rasterization-based differentiable renderers, our renderer has robust support for texture mapping, generates gradients for all triangles overlapping a pixel, and implements efficient support for full order-independent transparency.

Bibliography

[1] Thomas Auzinger, Przemyslaw Musialski, Reinhold Preiner, and Michael Wimmer. Non-Sampled Anti-Aliasing. In *Vision, Modeling & Visualization*. The Eurographics Association, 2013.

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

[3] Sai Praveen Bangaru, Tzu-Mao Li, and Frédo Durand. Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):245:1–245:18, 2020.

[4] Vineet Batra, Mark J. Kilgard, Harish Kumar, and Tristan Lorach. Accelerating vector graphics rendering using the graphics hardware pipeline. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 34(4):146:1–146:15, 2015.

[5] Louis Bavoil, Steven P Callahan, Aaron Lefohn, João LD Comba, and Cláudio T Silva. Multi-fragment effects on the GPU using the k-buffer. In *Symposium on Interactive 3D Graphics and Games*, pages 97–104, 2007.

[6] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999.

[7] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. FAUST: Dataset and evaluation for 3D mesh registration. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Piscataway, NJ, USA, June 2014. IEEE.

[8] Brent Burley. Physically-based shading at Disney. In *SIGGRAPH Course Notes. Practical physically-based shading in film and game production.*, volume 2012, pages 1–7. ACM, 2012.

[9] Loren Carpenter. The a-buffer, an antialiased hidden surface method. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, page 103–108, New York, NY, USA, 1984. Association for Computing Machinery.

[10] Edwin Catmull. A hidden-surface algorithm with anti-aliasing. *Comput. Graph. (Proc. SIGGRAPH)*, 12(3):6–11, 1978.

[11] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository. *arXiv:1512.03012*, 2015.

[12] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3D objects with an interpolation-based differentiable renderer. In *Advances in Neural Information Processing Systems*, pages 9609–9619, 2019.

[13] Martin de La Gorce, David J Fleet, and Nikos Paragios. Model-based 3D hand pose estimation from monocular video. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(9):1793–1805, 2011.

[14] Xavier Décoret, Frédo Durand, François X. Sillion, and Julie Dorsey. Billboard clouds for extreme model simplification. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 22(3):689–696, 2003.

[15] Tom Duff. Polygon scan conversion by exact convolution. In *International Conference On Raster Imaging and Digital Typography*, pages 154–168, 1989.

[16] Tom Duff. Deep compositing using Lie algebras. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 36(3), 2017.

[17] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T. Freeman. Unsupervised training for 3D morphable model regression. In *Computer Vision and Pattern Recognition*, 2018.

[18] Ioannis Gkioulekas, Shuang Zhao, Kavita Bala, Todd Zickler, and Anat Levin. Inverse volume rendering with material dictionaries. *ACM Trans. Graph.*, 32(6):162:1–162:13, nov 2013.

[19] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2008.

[20] André Jalobeanu, Frank O Kuehnel, and John C Stutz. Modeling images of natural 3D surfaces: Overview and potential applications. In *Conference on Computer Vision and Pattern Recognition Workshop*, pages 188–188. IEEE, 2004.

[21] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. SDFDiff: Differentiable rendering of signed distance fields for 3D shape optimization. In *Computer Vision and Pattern Recognition*, pages 1251–1261, 2020.

[22] Jorge Jimenez, Jose I. Echevarria, Tiago Sousa, and Diego Gutierrez. SMAA: Enhanced morphological antialiasing. *Comput. Graph. Forum (Proc. Eurographics)*, 31(2), 2012.

[23] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *Computer Vision and Pattern Recognition*, pages 3907–3916. IEEE, 2018.

[24] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[25] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6), 2020.

[26] 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.

[27] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):193:1–193:15, 2020.

[28] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. Paparazzi: Surface editing by way of multi-view image processing. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):221:1–221:11, 2018.

[29] Hsueh-Ti Derek Liu, Michael Tao, Chun-Liang Li, Derek Nowrouzezahrai, and Alec Jacobson. Beyond pixel norm-balls: Parametric adversaries using an analytically differentiable renderer. In *International Conference on Learning Representations*, 2019.

[30] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. *International Conference on Computer Vision*, 2019.

[31] Tom Lokovic and Eric Veach. Deep shadow maps. In *SIGGRAPH*, pages 385–392, 2000.

[32] Matthew M. Loper and Michael J. Black. OpenDR: An approximate differentiable renderer. In *European Conference on Computer Vision*, volume 8695, pages 154–169. ACM, 2014.

[33] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 38(6):228, 2019.

[34] Hugh Malan. Edge anti-aliasing by post-processing. In *GPU Pro*, pages 265–289. A K Peters, 2010.

[35] Josiah Manson and Scott Schaefer. Wavelet rasterization. *Comput. Graph. Forum (Proc. Eurographics)*, 2011.

[36] Morgan McGuire and Louis Bavoil. Weighted blended order-independent transparency. *Journal of Computer Graphics Techniques*, 2(4), 2013.

[37] 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 *European Conference on Computer Vision*, 2020.

[38] Diego Nehab and Hugues Hoppe. Random-access rendering of general vector graphics. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 27(5):135:1–135:10, 2008.

[39] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *Computer Vision and Pattern Recognition*, pages 165–174, 2019.

[40] Gustavo Patow and Xavier Pueyo. A survey of inverse rendering problems. *Comput. Graph. Forum*, 22(4):663–687, 2003.

[41] Thomas Porter and Tom Duff. Compositing digital images. *Comput. Graph. (Proc. SIGGRAPH)*, pages 253–259, 1984.

[42] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *SIGGRAPH*, pages 497–500. ACM Press/Addison-Wesley Publishing Co., 2001.

[43] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3D deep learning with PyTorch3D. *arXiv preprint arXiv:2007.08501*, 2020.

[44] Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. A versatile scene model with differentiable visibility applied to generative pose estimation. In *International Conference on Computer Vision*, pages 765–773. IEEE, 2015.

[45] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019.

[46] Vadim N Smelyansky, Robin D Morris, Frank O Kuehnel, David A Maluf, and Peter Cheeseman. Dramatic improvements to feature based stereo. In *European Conference on Computer Vision*, pages 247–261. Springer, 2002.

[47] Rick Szeliski and Polina Golland. Stereo matching with transparency and matting. In *Sixth International Conference on Computer Vision (ICCV'98)*, pages 517–524. IEEE Computer Society, January 1998.

[48] Kevin Weiler and Peter Atherton. Hidden surface removal using polygon area sorting. *Comput. Graph. (Proc. SIGGRAPH)*, 11(2):214–222, 1977.

[49] Lance Williams. Pyramidal parametrics. *Comput. Graph. (Proc. SIGGRAPH)*, 17(3), 1983.

[50] Chris Wyman. Exploring and expanding the continuum of OIT algorithms. In *High Performance Graphics*, pages 1–11, 2016.

[51] Jason C. Yang, Justin Hensley, Holger Grün, and Nicolas Thibieroz. Real-time concurrent linked list construction on the gpu. *Comput. Graph. Forum (Proc. EGSR)*, page 1297–1304, 2010.

[52] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 38(6):1–14, 2019.

[53] Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. Path-space differentiable rendering. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 39(6):143:1–143:19, 2020.

[54] Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. A differential theory of radiative transfer. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 38(6):227, 2019.

[55] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 37(4):65:1–65:12, 2018.

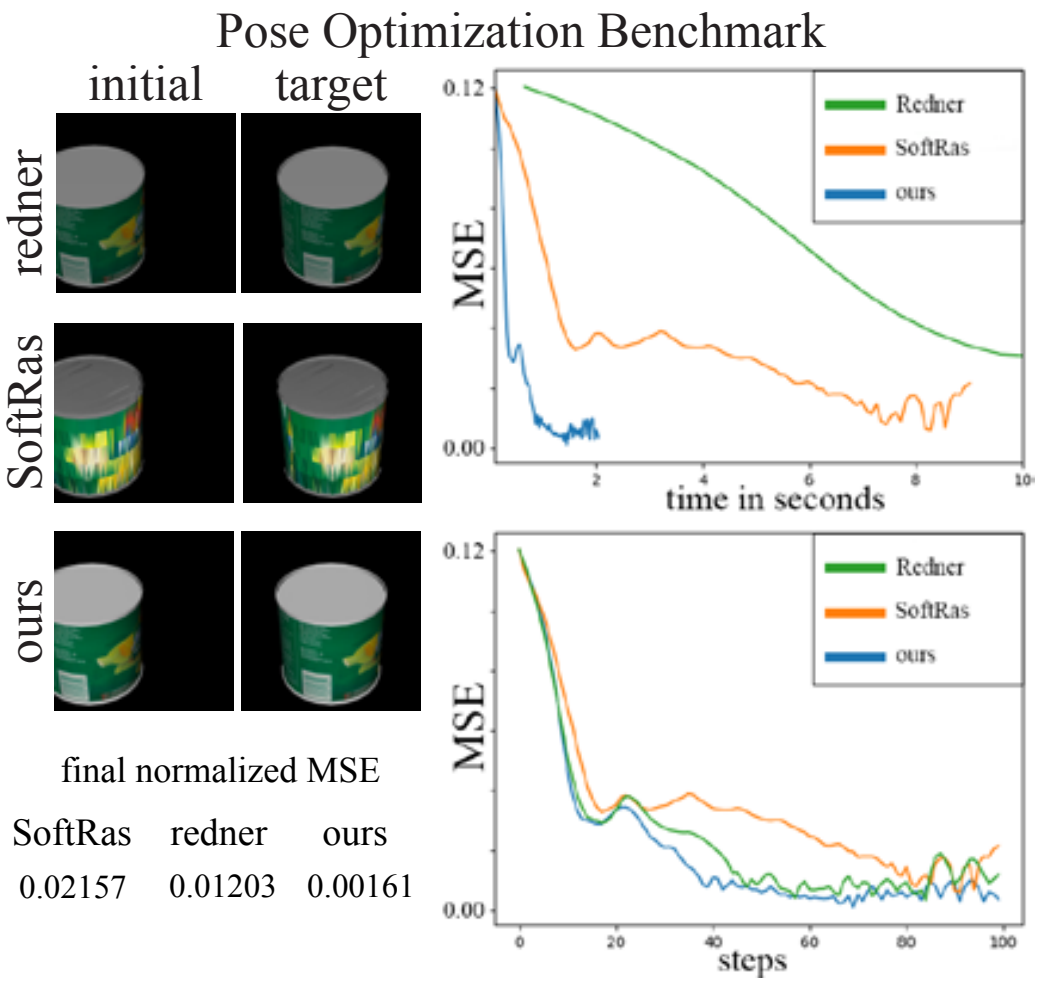

Figure 4-1: We compare our rasterization-based differentiable renderer with previous methods on a pose estimation task with a textured mesh (1432 triangles) on a 1024×1024 image. Our method generates higher-quality gradients while being significantly faster than previous approaches. Our renderer converges within two seconds, compared to 9 seconds for SoftRas [30] and 30 seconds for REDNER [26]. We normalize the mean square errors (MSE) relative to the first frame. SoftRas renders blurry textures due to the lack of UV-mapped texture support. More importantly, it suffers from compositing artifacts on the top of the can, due to its depth-based weighting. Our method is parameter-free and achieves lower error thanks to our high-quality and deterministic gradients, in contrast to the stochastic gradients of the ray tracing methods or the approximated rendering.

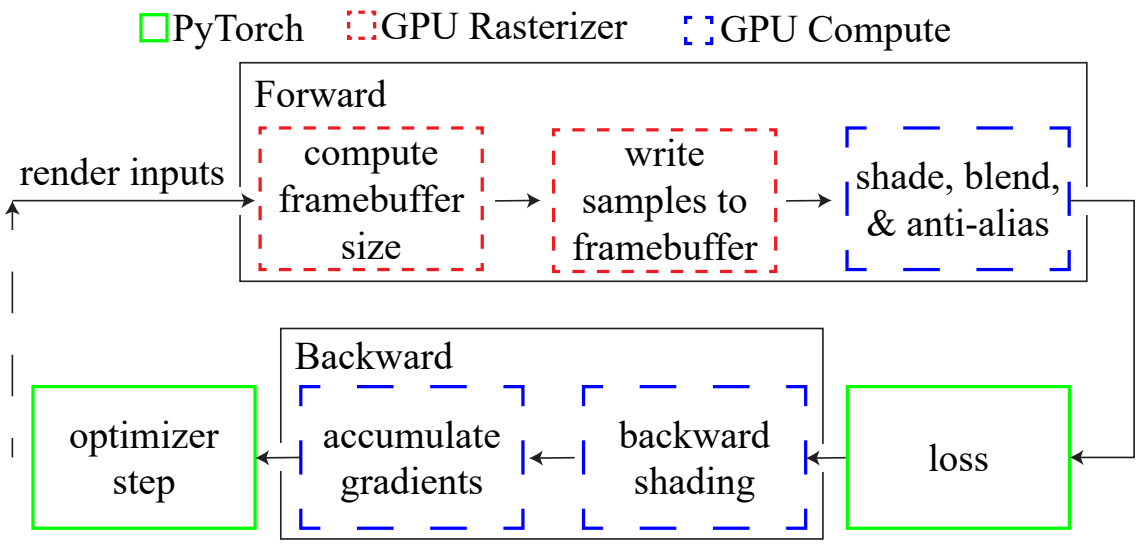

Figure 4-2: Our renderer leverages hardware-accelerated rasterization for the forward pass. The rasterized triangle samples are first written to a dynamically-sized "deep" framebuffer. After all triangle samples are collected, they are sorted by depth and shaded. Multi-sample deferred shading as presented here produces high-quality, differentiable, anti-aliased images and renders advanced effects such as order-independent transparency that are not supported by previous differentiable rasterizers. The backward pass is split between PyTorch and custom ops implemented using the graphics API.

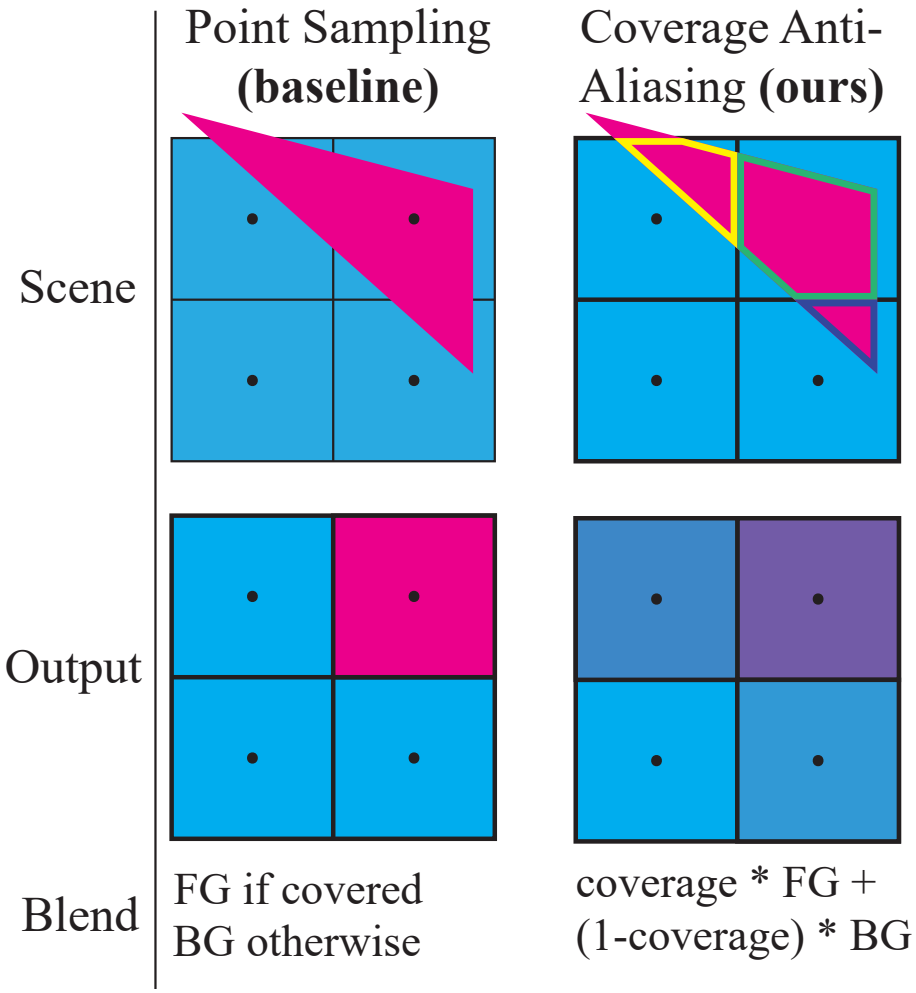

Figure 4-3: Instead of testing the visibility of a triangle at the pixel center (left), our method (right) computes the fractional coverage of a triangle using analytical integration, and blends the contribution between the foreground triangle and background color using the analytical coverage for anti-aliasing. Multiple triangles are blended from front to back, where the current triangle is treated as the foreground, and the unblended triangles are treated as the background. Our analytical anti-aliasing enables high-quality anti-aliasing and makes visibility differentiable with respect to triangle vertices.

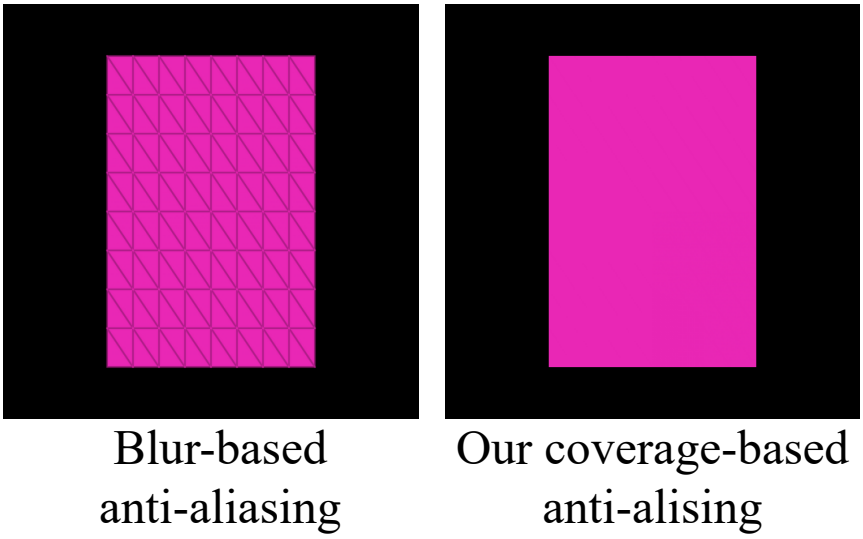

Figure 4-4: Blur-based anti-aliasing [30, 43] is prone to cracking artifacts at polygon edges. The artifacts in this particular figure come from the improper composition of the background color with the foreground triangles. In SoftRas [30], the background is blended with the foreground using softmax with a small fixed weight. Therefore the background would have different weighting for pixels hitting two triangles versus pixels hitting only one triangle. The same depth compositing strategy is used by many other differentiable rasterizers [12, 43]. In contrast, our analytical coverage computation, along with our alpha compositing, produces artifact-free images, and is completely parameter free.

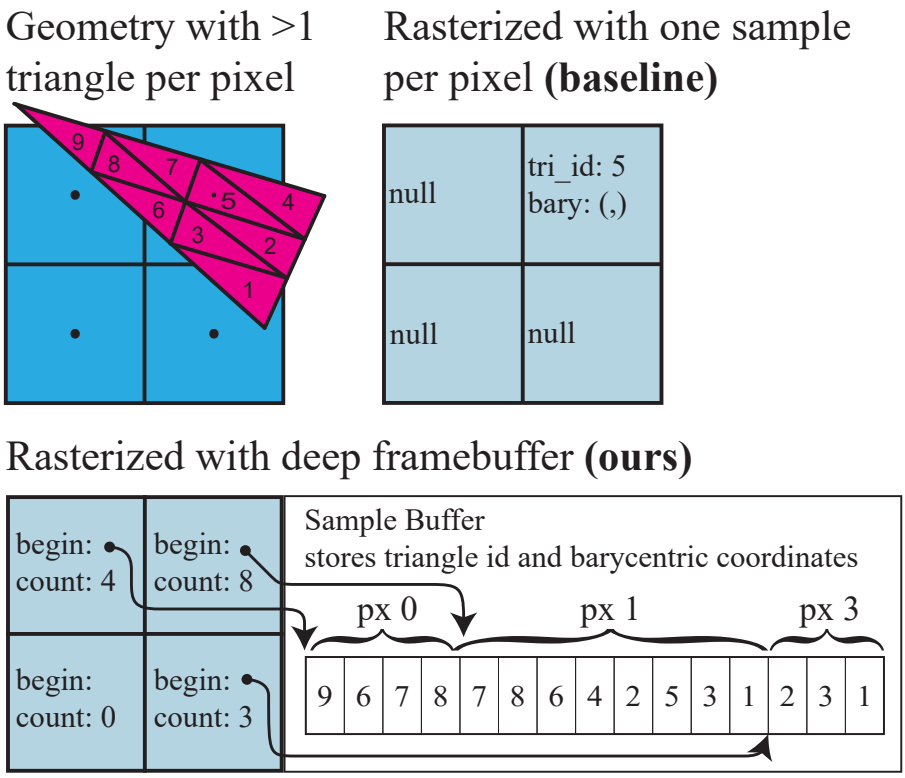

Figure 4-5: In order to efficiently rasterize multiple triangles per pixel, we use a deep framebuffer data structures that store a list of triangles overlapping the pixel's reconstruction filter. For each pixel, we store a pointer to a list of triangle IDs and barycentric coordinates, and the count of triangles. We build the list with two passes: the first pass counts the number of triangles and allocate memory, and the second pass appends the IDs and coordinates into the list. Our deep framebuffer is efficiently implemented using rasterization hardware.

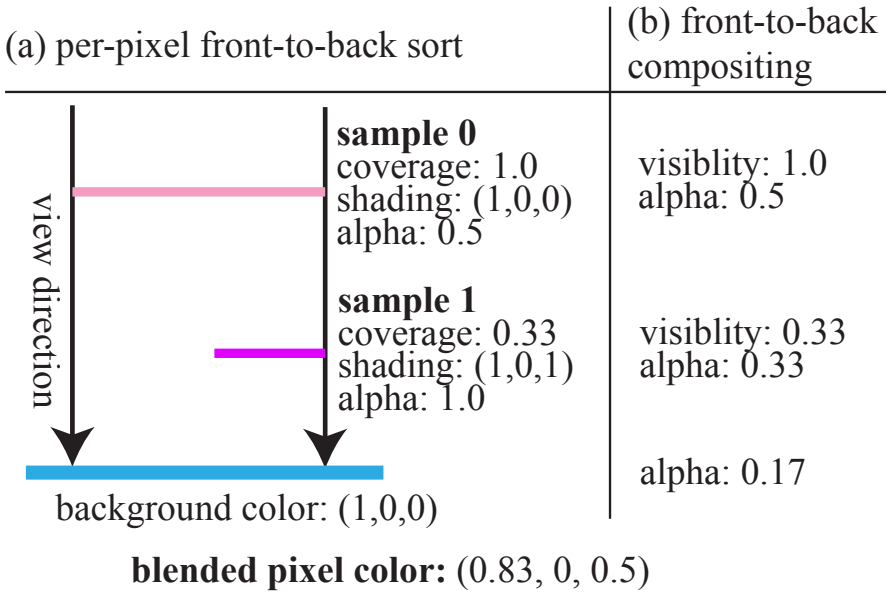

Figure 4-6: We blend multiple samples in a pixel together using a visibility-aware alpha blending. Here we show an example of a pixel with a transparent sample, a opaque but partially visible sample, and a background. We process the samples in a front-to-back order (top-to-down in the figure), tracking the visible portion (*visibility*) and the fraction of reflected color (*alpha*) for each sample. The final color is computed by a sum of the shading color of each segment weighted by their *alpha*.

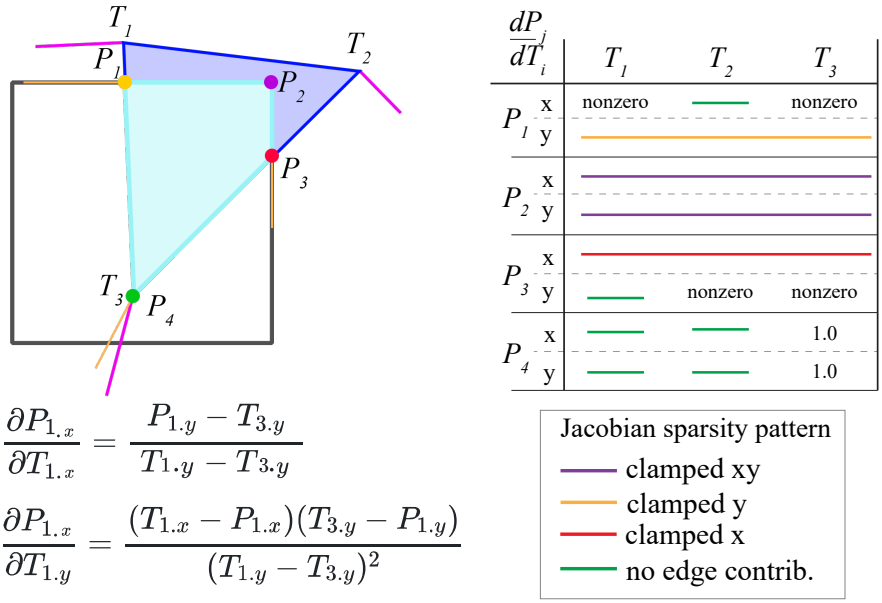

Figure 4-7: To compute the coverage for differentiable anti-aliasing, we clip the triangle T to the pixel filter kernel and compute the area of the clipped polygon P. The figure shows the coverage gradient for vertices of both the clipped polygon and of the original triangle. To compute triangle gradients, we backpropagate through the clipped vertices. The resulting Jacobian $\partial P_j / \partial T_i$ is usually sparse, since clipped vertices either contribute no gradients (clipped to pixel corner like P_2) or only contribute gradients along one axis (clipped to pixel edge, e.g. P_1 and P_3). There are three cases of non-zero entries in the Jacobian: For vertices inside the pixel (P_4), original and clipped vertices are identical and the partial derivatives evaluate to 1. We also list the partial derivatives for vertex motion parallel to and edge, and perpendicular to an edge.

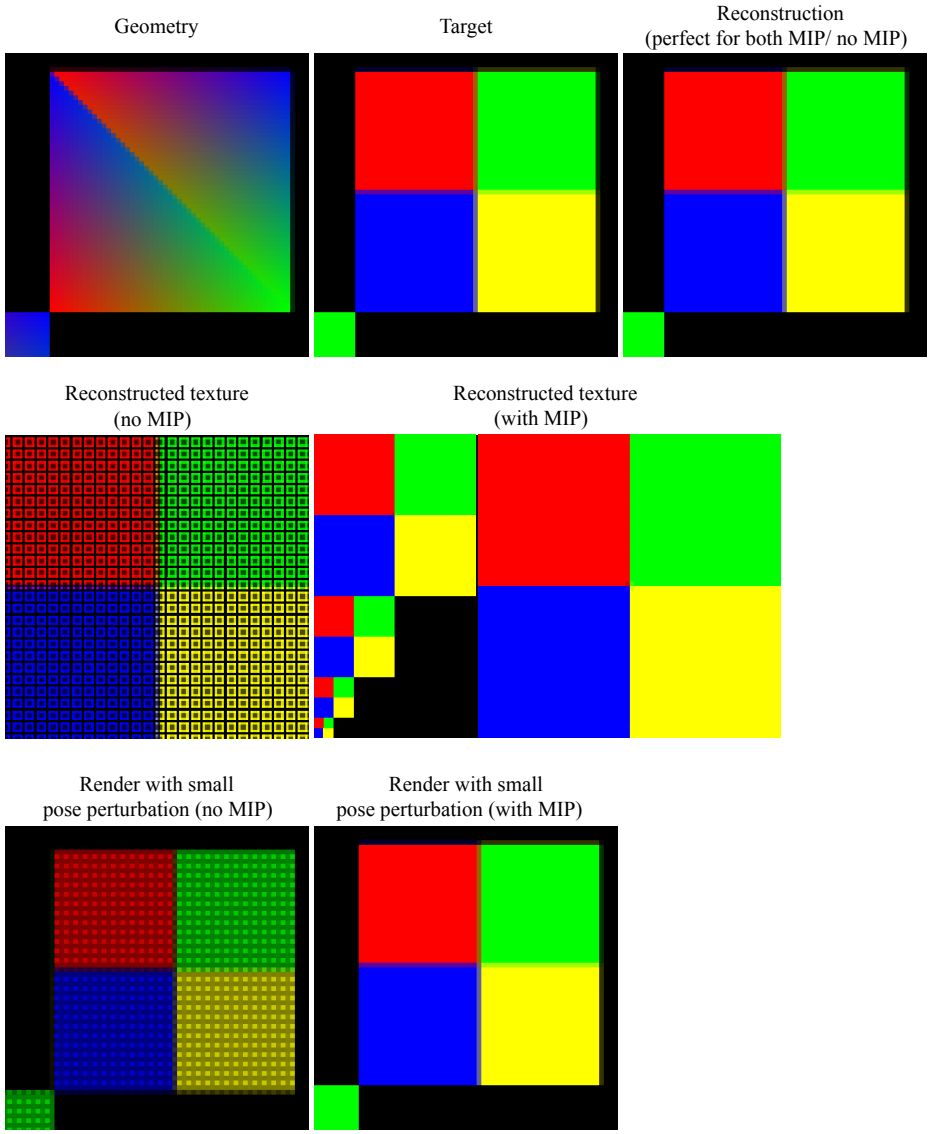

Figure 4-8: Mipmapping [49] is essential for inverse texturing optimizations. Given sufficient texture resolution, we can easily reconstruct a single target image (top row). However, when the camera rays undersample the high-resolution textures, lack of mipmapping leads to a sparse reconstruction. With mipmapping, gradients are written to scale-appropriate MIP levels, leading to dense reconstructions (second row). The difference becomes obvious when we re-render the reconstructed scene from a slightly altered viewpoint (third row).

Total frame time @1024px:	**21ms**
Total Count Pass:	**3.0ms**
Count pass rasterization:	2.1ms
Count pass count accumulation:	0.7ms
Check deep framebuffer size:	0.2ms
Total Forward Rendering:	**9.0ms**
Write samples to deep framebuffer:	2.3ms
Forward shading:	6.7ms
Total Backward Pass:	**6.1ms**
Backward shading	5.0ms
Backward vertex transform	0.9ms
Loss, step, synchronization:	*2-3ms*

Figure 4-9: We break down the frame time of our renderer using the inverse rendering benchmark in Figure 4-1. Our rasterizer spends roughly 57% of time in the forward pass, which includes the rasterization passes and coverage computation. The backward pass takes around 29%, and the remaining 14% is spent in PyTorch or on communication overhead.

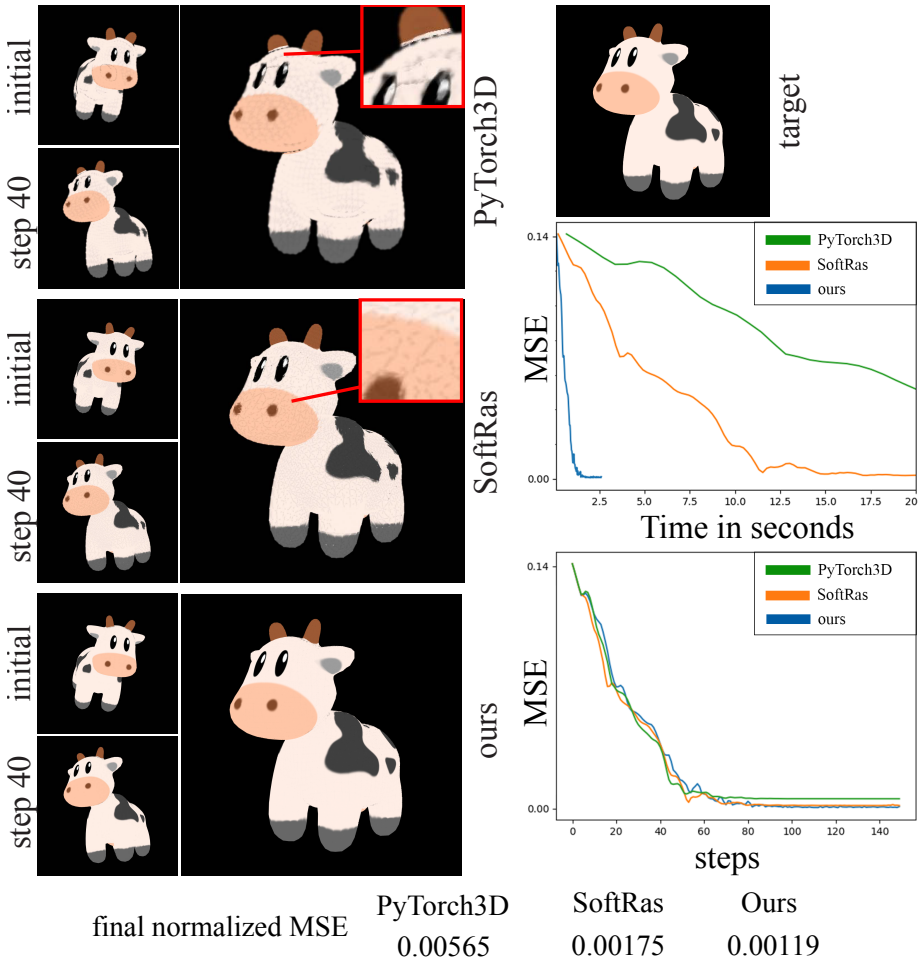

Figure 4-10: We compare our differentiable renderer with SoftRas [30] and PyTorch3D [43] with a 6 degree-of-freedom pose optimization task on a textured mesh (5856 triangles). For PyTorch3D, we set the maximum number of faces per pixel to 32 (setting it to a higher number requires more than 12GB of memory, exceeding our GPU memory). Our method renders and computes gradients more than 8 times faster than SoftRas, and more than 15 times faster than PyTorch3D. Both SoftRas and PyTorch3D show the cracking artifacts similar to the one in Figure 4-4 in the smooth region of the mesh (zoom in for inspection).

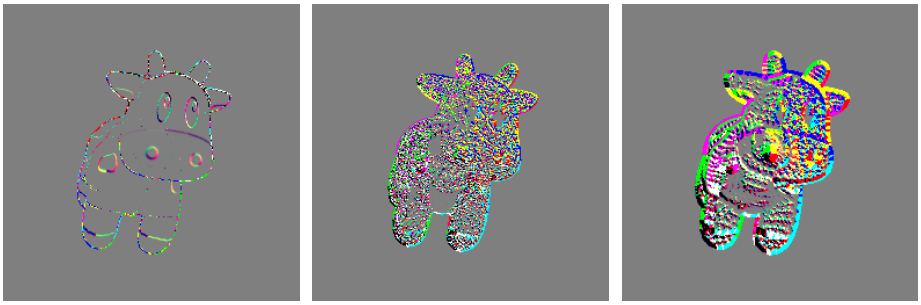

Figure 4-11: We visualize the gradients with respect to 3D translation generated by our method with SoftRas [30] and PyTorch3D [30]. Both SoftRas and PyTorch3D shows spurious gradients at the smooth region of the mesh due to their the depth-based composition. Our visibility-aware alpha blending avoids the issue and generate high-quality gradients.

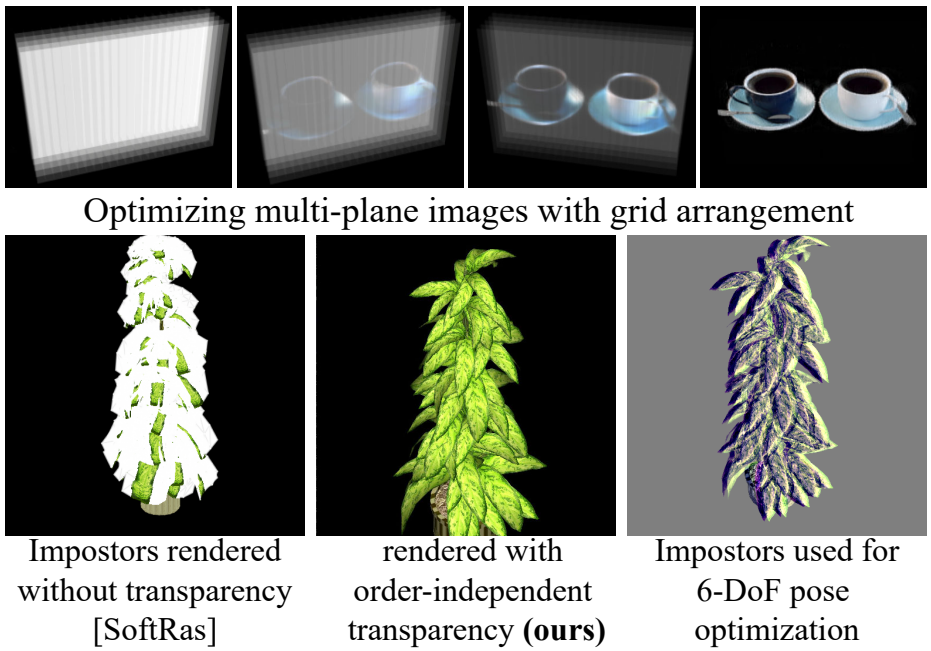

Figure 4-12: Our rendering model enables using transparent surfaces for inverse problems. **Top:** Per-pixel sorting allows our renderer to optimize multiplane images [47, 55] with arbitrary self-intersecting proxy geometry. **Bottom:** *Impostors*, or billboards are a classic technique to reduce geometric complexity of meshes by encoding geometric detail a texture's alpha channel.

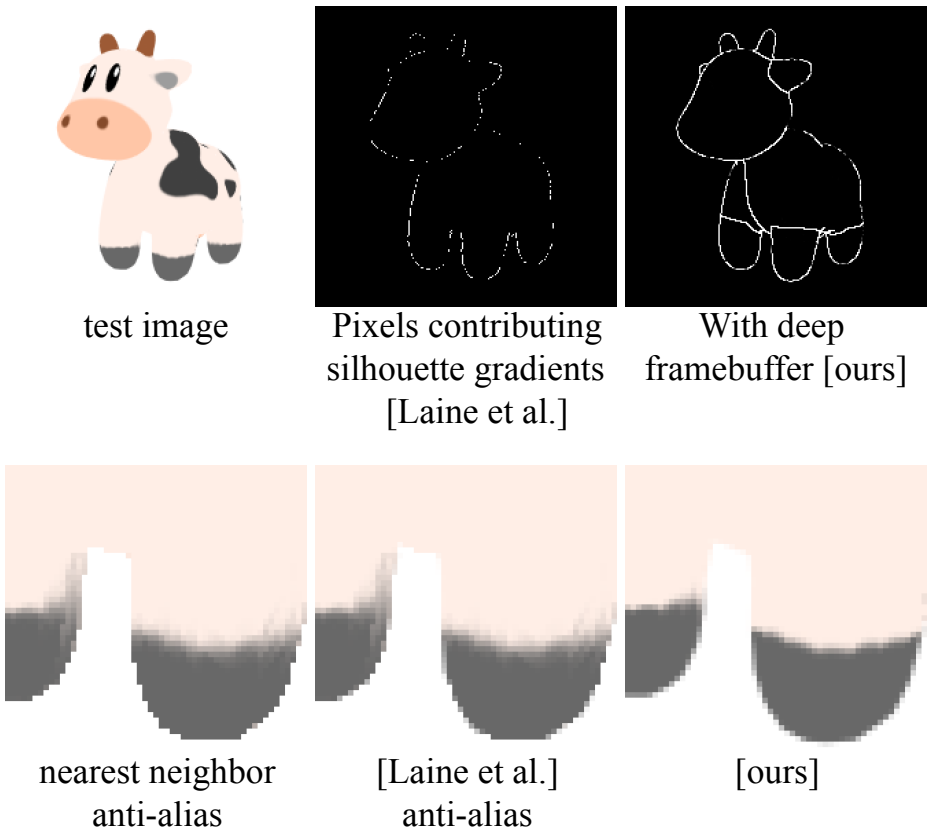

Figure 4-13: Differentiable renderers that rely on image-based anti-aliasing [25] suffer from undersampling of silhouette gradients, when rendering detailed geometry to low-resolution images. We show the non-zero silhouette gradients of Laine et al.'s method, compared to ours. Our multi-sample approach results in higher-quality gradients even at low framebuffer resolutions. This also results in higher anti-aliasing quality from our approach.

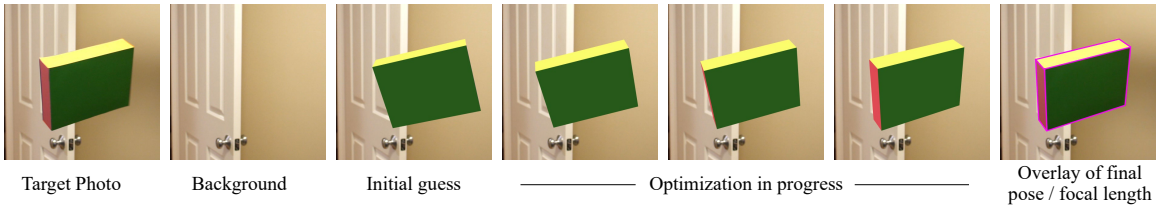

Figure 4-14: 6 degree-of-freedom pose estimation and camera calibration in real photographs. Shown in the left is the target picture of a cardboard box. We take a picture as background for compositing, and initialize the optimization with the measured dimensions of the box, approximate albedo, focal length and guessed position and translation.

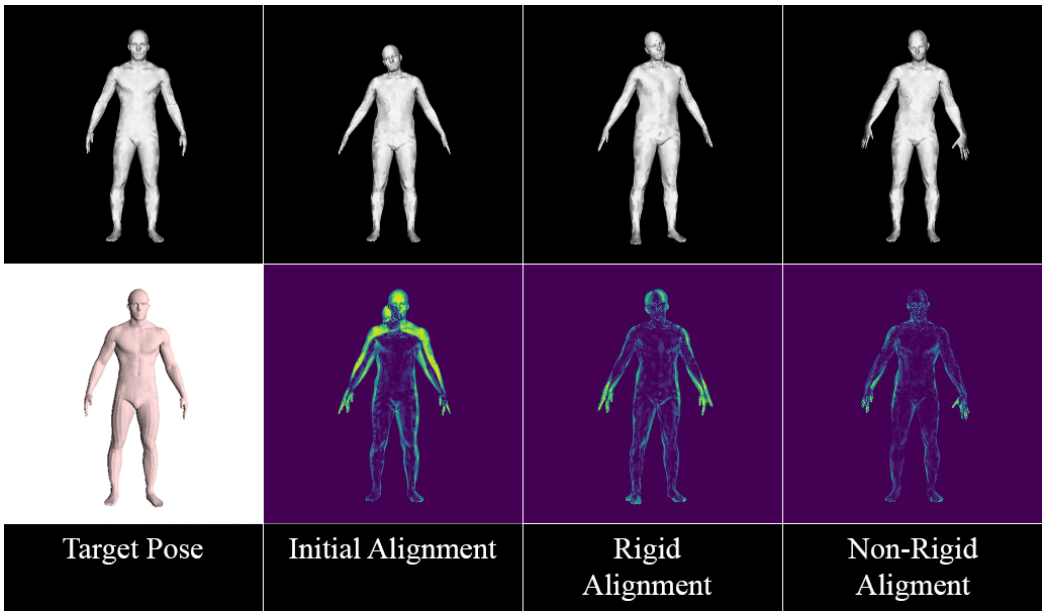

Figure 4-15: We show an application of our renderer optimizing the parameters non-rigid body from the FAUST dataset [7].

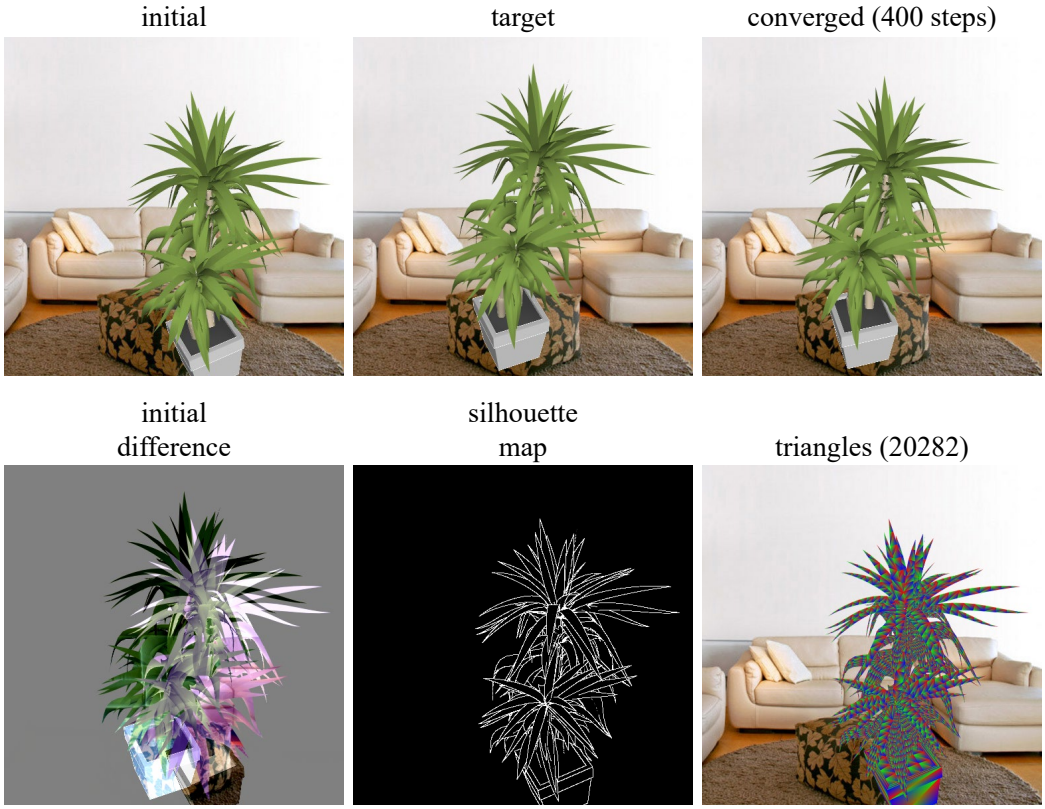

Figure 4-16: 6 DoF pose estimation for a scene with high triangle count and significant depth complexity. A combined forward/backward pass for this model with more than 20,000 triangles takes less than 13ms.

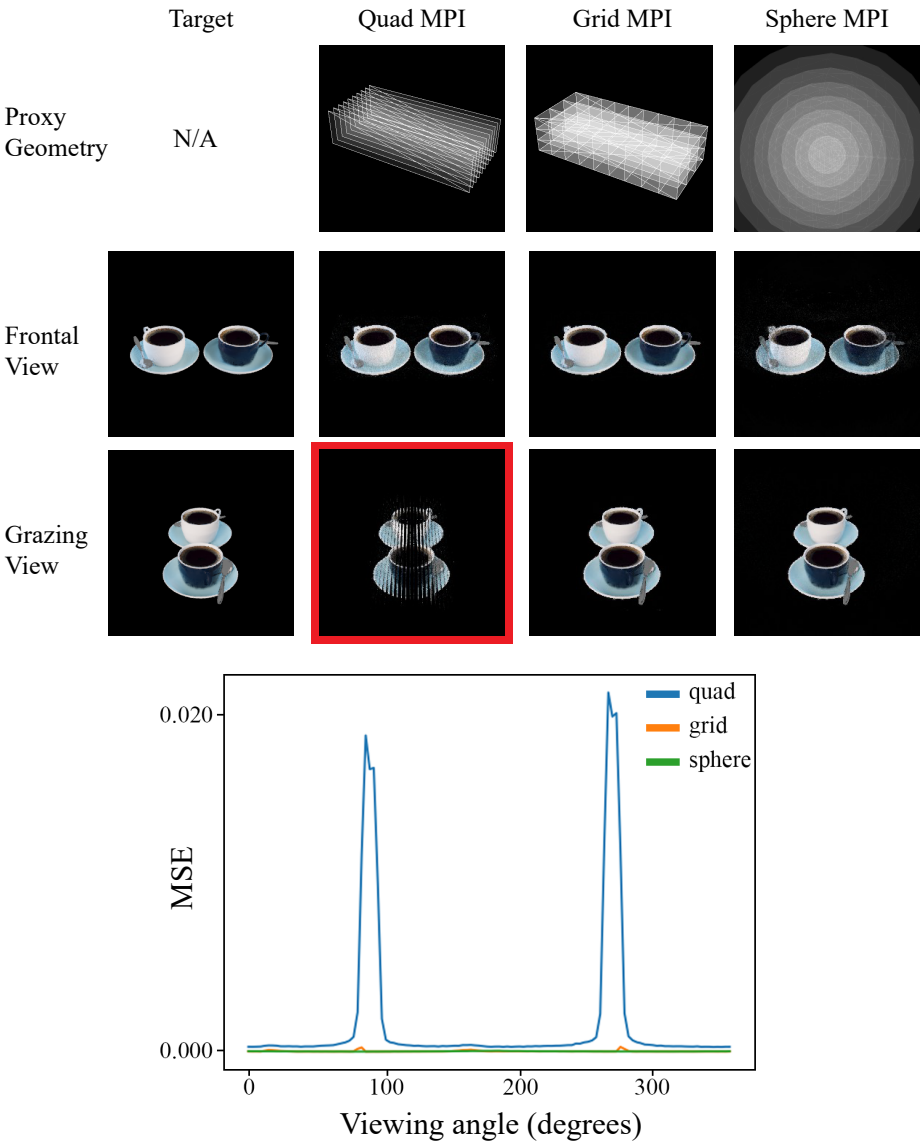

Figure 4-17: Multi-plane images [55] use an alpha-blended stack of quads as a scene representation. They are well-suited for reconstructing and rendering frontal viewing directions. However, they break down for grazing viewing directions that are parallel to the planes, which we show qualitatively on top, and quantitatively in terms of reconstruction mean square error (MSE) below. Since our renderer sorts and blends samples on a per-pixel basis, it enables the use of more general proxy geometry.

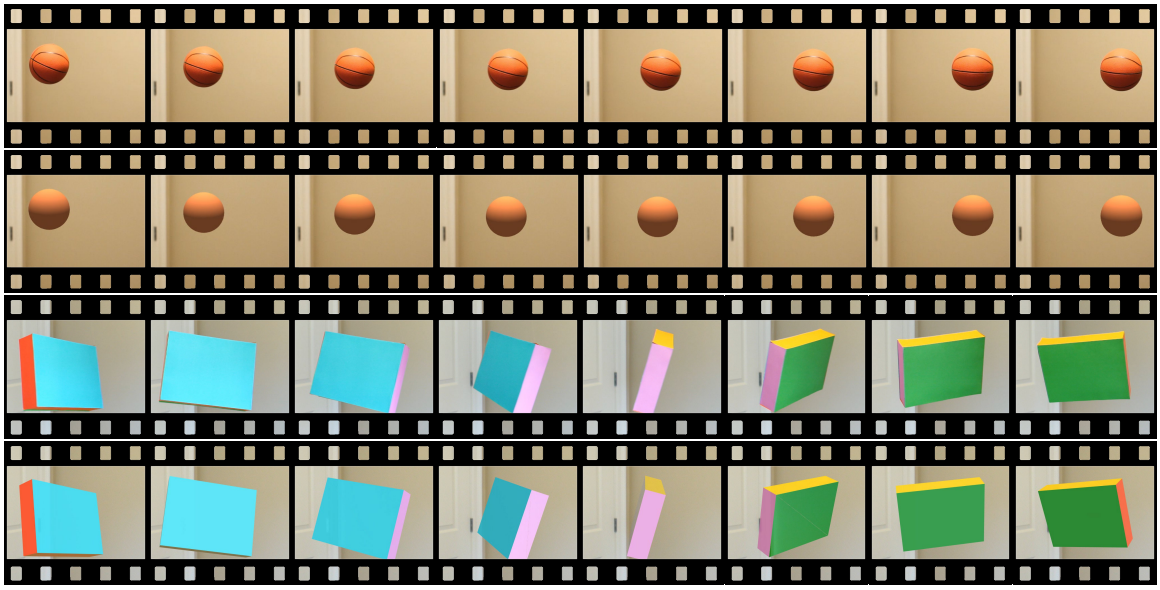

Figure 4-18: We use our renderer to implement a photometric object tracker. Top row is the target and bottom row is our rendering. For the cardboard box scene, we jointly optimize for both pose and surface albedo, which helps track through changes in observed surface brightness.

Chapter 5

Conclusion and Discussion

The preceding chapters presented two distinct approaches for solving computational photography and inverse graphics problems: a feed-forward discriminative approach using supervised learning, and a generative approach based on differentiable programming of a well-understood forward model. We have shown how state-of-the-art results in computational photography can be achieved through targeted collection of training data using custom prototypes. In Chapter 4, we have shown how to use differentiable programming to encode detailed domain knowledge into solvers for inverse problems. In the presented case of differentiable rendering, we showed how to efficiently implement a real-time rasterizer that produces more complete and accurate gradients than other differentiable rasterizers.

5.1 Discussion image-to-image translation using paired training data

In chapters 2 and 3 we implemented supervised learning systems to solve problems ranging from image relighting to low-light photography. Supervised learning relies on large datasets of labeled training pairs and is known to yield poor test result whenever the training and test distributions differ, or when the number of samples in the training set is too small. This thesis proposes portable camera prototypes

with computer-controlled light sources as a means to collect ample high quality training data, and demonstrates the success of such data collection with state-of-the-art results.

Compared to data sourced from consumer internet platforms like photo collections with millions of samples [3], scalable data collection with custom hardware remains a challenging problem. The cameras used in chapters 2 and 3 are unique prototypes, which limits the scalability of capture. For example, with a single, multi-illumination camera, a skilled operator is able to collect around 50 image bursts per day. This number is mostly limited by non-technical factors, such as the need for the operator to ensure access to unique capture locations.

Scaling beyond the amount of data that can reasonably be captured by an individual operator will require a small-scale production run of cameras and a team of hired operators who would capture in a large number of locations. In order to increase the diversity of the data, capture equipment should be shipped to a geographically diverse set of locations. In addition, our experience with a single prototype suggests that ensuring high quality and reliable data capture will require emphasis on the training of operators, as well as improvements to the usability and reliability of the cameras. Despite these operational challenges, the relative affordability of our cameras puts the production of tens to hundreds of cameras within reach of many academic and industrial labs, which would allow them to collect datasets with hundreds of thousands of samples, which can be expected to be a sufficient amount of data for many relevant applications.

Another challenge that arises when capturing with custom prototypes is that of transferability of the trained model to different hardware at test time. A model trained entirely with data from a single camera might be specialized to a single tone curve, lens distortion, or other imaging characteristics. In this case, transferring to a different kind of camera at test time might lead to a drop in test performance. Similarly for computer-controlled illumination, the trained model might specialize to the particular light source beam or emission spectrum of the training rig.

Potential mitigations to these problems include simple data augmentations, such

as additive noise or randomized image warps, or more principled domain adaptation techniques [11]. For applications with known deployment scenarios, for example known cameras and light sources inside a particular smart phone model, a valid strategy might be to build capture hardware with matching components to minimize any domain gap. For scenarios where domain adaptation is insufficient and a large amount of data is captured, a valid strategy might be to manufacture a heterogeneous collection of capture hardware that is more likely to include the sensor types and lighting geometries found at test time.

For problems where a single input image matches to a distribution of outputs, feed forward training only provides us with a single answer, rather than allowing us to predict the entire output distribution. This becomes a deficiency in particular for multi-modal output distributions (for example in image colorization [13]), where shallow image losses steer the network towards predicting mean (L2) or median (L1) estimates of the distribution, which might not at all be representative of any of the distribution's modes. We notice a similar effect in the illumination estimation task of Chapter 2, where these loss functions lead to predicted environment maps that lack high frequencies that are found in the real data. A first approach to overcome these limitations is to use perceptual loss functions [13] which steer the network's prediction away from conservative mean/median estimates. Second, we might use randomized auxiliary inputs which disambiguate the input/output mapping, and provide for an explicit way to draw samples from the output distribution [7]. Finally, we can go beyond point samples by randomizing the initialization of gradient-based search using generative models, as discussed in the next section.

5.2 Discussion of inverse graphics using differentiable programming

Supervised learning as presented in the previous section requires significant amounts of training data, since model behavior or data priors are not explicitly specified by

the researcher, but instead must be learned from data. Differentiable rendering as presented in Chapter 4 is an example of a larger class of systems based on differentiable programming that allow a known forward model (e.g. image processing [8] or physical simulation [6]) to be written by the programmer, while a compiler automatically generates code that propagates gradients through the system [5] back to system parameters or inputs. A forward model implemented in a differentiable framework allows for the inverse problem to be solved by gradient-based search in the input space. For parameter-free models, there is need for training data, since all system behavior is written by the researcher.

In addition to their efficient use of training data, generative models based on differentiable programming also naturally allow for inference of interpretable models of the scene. During the optimization process, gradients are applied to the inputs of the renderer, which usually are vertex attributes, transform matrices, texture maps, light source parameters, and environment maps. All the input have meanings that are well-understood by programmers and existing software systems, and the inferred representations can be transferred to other renderers or content creation tools. This stands in contrast to Neural Rendering [12], where the rendering algorithm itself is learned, and the inferred "neural" representations are not interpretable using conventional tools.

While interpretability is a desirable property for inferred representations, the raw inputs to the renderer, for example raw vertex positions, present a highly non-convex, high-dimensional optimization landscape, where the optimization procedure easily gets trapped in local minima. Because of this, practical inverse rendering systems generally perform optimization in a lower-dimensional parametrization of the shape (e.g. the truncated PCA basis of Blanz and Vetter [2]). We consider the identification of representations that are simultaneously interpretable, sufficiently expressive, and also amenable to optimization, an important future research direction for inverse graphics based on differentiable rendering.

If the loss landscape of the chosen scene representation is sufficiently convex, gradient-based optimization can succeed from a generic initial state. This is the

case with highly constrained scenes based on morphable models [2], and for very soft visibility models based on volumetric rendering [9].

For problems with high-dimensional scene representations or a limited number of observations, a generic starting point might not allow the optimization to converge to a satisfying solution. In these cases, it is possible to leverage a larger corpus of scenes, by training a feed-forward network to predict the initial scene state, which is then refined by gradient-based optimization [4].

5.3 Summary

Supervised learning of computational photography and inverse graphics problems is particularly suitable when the forward model is poorly understood or ambiguous, such as the mapping of near-infrared images to the visible spectrum. The key challenge then becomes collecting sufficient amounts of labeled training data, and minimizing the domain gap between training and test sets. This thesis presented several hardware prototypes that were used to collect datasets, which led to state of the art results in image relighting, lighting estimation, and mixed white balance (Chapter 2), as well as a novel approach to low-light photography based on near-infrared colorization (Chapter 3). Computational illumination also allowed us to propose a automated approach to bounce flash photography [10], and achieve remarkable reconstruction quality in non-line-of-sight imaging [1]. These results suggest that efforts to collect novel, problem-specific datasets will be a fruitful area for future research. In particular, our research suggests that light-weight capture devices using off-the-shelf components will allow for scalable data collection within an affordable budget.

For applications with a well-understood forward model like graphics pipelines (Chapter 4), a differentiable implementation of the forward model imposes structure and reduces the amount of training data required to solve the inverse problem. The search space of probable solutions can be further narrowed by learned priors, for example natural image priors for computational photography problems, or shape, material, and illumination priors over scenes in inverse graphics problems.

The introduction of learned representations into differentiable programming-based solvers points to an exciting direction for future research that incorporates both major topics of this thesis into practical inverse graphics systems. We believe that hardware prototypes similar to the ones presented in this thesis could be constructed to collect large repositories of real-world shapes, materials, light environments, or even entire scene layouts. Scene priors derived from this data would provide an important data-driven foundation for inverse graphics systems. On top of this foundation, differentiable rendering is a crucial building block that contributes both domain knowledge and interpretability. The combined approaches promise to optimally utilize both real-world data and programmer-specified domain knowledge, and together hold significant promise as a future resarch direction for computer vision and inverse graphics.

Bibliography

[1] Miika Aittala, Prafull Sharma, Lukas Murmann, Adam B. Yedidia, Gregory W. Wornell, William T. Freeman, and Frédo Durand. Computational mirrors: Blind inverse light transport by deep matrix factorization. *CoRR*, abs/1912.02314, 2019.

[2] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH*, pages 187–194. ACM Press/Addison-Wesley Publishing Co., 1999.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition*. IEEE, 2009.

[4] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Styles Overbeck, Noah Snavely, and Richard Tucker. Deepview: High-quality view synthesis by learned gradient descent. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[5] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, second edition, 2008.

[6] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, Nathan Carr, Jonathan Ragan-Kelley, and Frédo Durand. Difftaichi: Differentiable programming for physical simulation. *CoRR*, abs/1910.00935, 2019.

[7] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *CoRR*, abs/1912.04958, 2019.

[8] Tzu-Mao Li, Michaël Gharbi, Andrew Adams, Frédo Durand, and Jonathan Ragan-Kelley. Differentiable programming for image processing and deep learning in Halide. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 37(4):139:1–139:13, 2018.

[9] 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 *European Conference on Computer Vision*, 2020.

[10] Lukas Murmann, Abe Davis, Jan Kautz, and Frédo Durand. Computational bounce flash for indoor portraits. *ACM SIGGRAPH Asia*, 2016.

[11] Pedro O Pinheiro. Unsupervised domain adaptation with similarity learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8004–8013, 2018.

[12] 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. *Computer Graphics Forum (Proc. Eurographics STAR)*, 39(2):701–727, 2020.

[13] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *European conference on computer vision*, pages 649–666. Springer, 2016.