

BOOSTING 3D RECONSTRUCTION WITH DIFFERENTIABLE IMAGING SYSTEMS

by

Wenzheng Chen

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

© Copyright 2023 by Wenzheng Chen

Abstract

Boosting 3D Reconstruction with Differentiable Imaging Systems

Wenzheng Chen
Doctor of Philosophy

Graduate Department of Computer Science
University of Toronto
2023

Imaging and 3D reconstruction are reciprocal procedures. The former employs imaging systems to measure scene properties, where it encodes various 3D-related properties (including geometry, material, and light) into specific measurements. Conversely, the latter aims to solve the inverse problem, *i.e.*, recovering 3D-related properties back from captured data. As the reciprocal step, imaging process provides important cues to 3D reconstruction. Exploring ways to utilize imaging priors is critical to improve the performance of 3D reconstruction.

Recent advances of deep learning (DL) have reshaped 3D reconstruction algorithms. Inspired by it, this dissertation provides a new framework to boost 3D reconstruction with imaging priors. The core idea is to differentiate imaging systems and embed them into reconstruction algorithms, in particular DL-based approaches. Consequently, the proposed framework allows information to be freely propagated between imaging and 3D reconstruction. It supports reconstruction algorithms to utilize inherent geometric and physical rules inside imaging systems for better recovery of 3D properties. Additionally, it enables tuning imaging systems to explore the best settings they should adopt under reconstruction tasks.

We validate the superiority of our framework on two imaging systems: rendering pipelines and structured light systems. First, we present novel differentiable reformulations of rendering pipelines and apply them in unsupervised, single-view 3D object reconstruction. Compared to previous methods, our novel designs demonstrate more accurate shape and texture recovery, as well as enable new capabilities to achieve faithful material and lighting disentanglement. Next, we discover optimal illumination patterns in structured light triangulation. Unlike existing approaches that use predetermined patterns, for the first time, differentiable structured light systems allow us to optimize patterns for the chosen imaging conditions and exhibit substantial improvements in 3D triangulation accuracy.

Acknowledgments

First and foremost, I am forever grateful to my supervisors Kyros Kutulakso and Sanja Fidler. Six years ago, I came to Toronto in a daze and embarked on a wonderful doctoral journey. This journey was a mix of unexpected delights and inevitable challenges. Kyros and Sanja led me through every difficult point during my Ph.D. with their wisdom and erudition. Without their guidance, I can't even begin to imagine how I would have completed my doctoral work.

Next, I want to deeply thank my supervisory committee, Alec Jacobson and Florian Shkurti. Their constructive feedback greatly improved my thesis drafts, and their constant encouragement has been invaluable. I also wish to thank David B. Lindell and Hao (Richard) Zhang for generously dedicating their time to serve on my final examination committee.

My doctoral work has largely been a result of productive collaborations. I've had the privilege to work with some extremely outstanding collaborators, including Jun Gao, Huan Ling, Zian Wang, Frank Shen, Yuxuan Zhang, Parsa Mirdehghan, Joey Litalien, and Edward J. Smith. I also want to extend my thanks to the friends I've met in the DGP and Sanja's group. They accompanied me through the long Ph.D. journey.

Lastly, I wish to express my love and gratitude to my parents and my sister for their unwavering support and unconditional love. I also want to take a moment to acknowledge myself, for the courage I had six years ago and for the persistence throughout the Ph.D. time.

Contents

1	Introduction	1
1.1	Organization	6
1.2	Included Publication	6
2	Background	8
2.1	Imaging Systems	8
2.2	3D Reconstruction Algorithms	9
2.3	Differentiable Imaging Systems	10
2.4	Differentiable Rendering	13
2.4.1	Rendering Equation	14
2.4.2	Monte Carlo Ray-tracing based Rendering Pipeline	16
2.4.3	Rasterization-based Rendering Pipeline	19
2.4.4	Applications & Limitations	21
2.5	Structured Light Systems	22
2.5.1	Structured Light Triangulation	22
2.5.2	Structured Light Patterns	24
2.5.3	Structured Light Decoding Algorithms	25
3	An Interpolation-based Differentiable Renderer	27
3.1	Introduction	27
3.2	Related Work	29
3.3	Differentiable Interpolation-based Renderer	30
3.3.1	Rendering Pipeline	30
3.3.2	Differentiable Rasterization	30
3.3.3	Rendering Models	33
3.3.4	Optimization	35
3.4	Single-view 3D Object Reconstruction	36
3.4.1	Basic Model: Predicting Geometry and Color	36

3.4.2	Lighting Models: Predicting Geometry, Texture, and Light	38
3.5	Experiments	40
3.5.1	Basic Model: Predicting Geometry and Color	40
3.5.2	Lighting Models: Predicting Geometry, Texture and Light	41
3.5.3	Texture and Lighting Results with Adversarial Loss	44
3.5.4	Disentanglement Study	48
3.6	Summary	49
4	A One-bounce Ray-tracing based Renderer	51
4.1	Introduction	51
4.2	Related Work	52
4.3	Differentiable Deferred Rendering	53
4.3.1	Overview	54
4.3.2	Background	54
4.3.3	Two-stage Deferred Rendering	55
4.3.4	Shading Models	56
4.3.5	DIB-R++ Rendering Effects	57
4.3.6	Optimization of Lighting and Material	60
4.4	Single-view 3D Object Reconstruction	64
4.5	Experiments	68
4.5.1	Synthetic Datasets	68
4.5.2	Evaluation Metrics	68
4.5.3	Metallic Surfaces	69
4.5.4	Glossy Surfaces	72
4.5.5	Shape & Material Evaluation	74
4.5.6	Ablation Study	75
4.6	Summary	78
5	Singe-view Real Imagery 3D Object Reconstruction	79
5.1	Introduction	79
5.2	Related Work	81
5.3	Our Approach	82
5.3.1	StyleGAN as Synthetic Data Generator	82
5.3.2	Camera Initialization	86
5.3.3	Training Inverse Graphics Neural Networks	87
5.4	Experiments	89
5.4.1	StyleGAN Dataset	89
5.4.2	Shape & Texture Recovery with DIB-R	89

5.4.3	Lighting & Material Prediction with DIB-R++	97
5.5	Summary	99
6	Differentiable Structured Light Triangulation	102
6.1	Introduction	102
6.2	Optimal Structured Light	105
6.3	Differentiable Epipolar-Only Imaging Formation Model	107
6.3.1	Plausible Parameter Space	107
6.4	Optimal Position Decoding	109
6.5	Optimal Position Encoding	110
6.6	Experiments	113
6.6.1	Ablation Study	116
6.7	Summary	118
7	Optimizing Illuminations for Active Imaging Systems with Optical Stochastic Gradient Descent	119
7.1	Introduction	119
7.2	Differentiable Imaging Systems	122
7.3	Optical SGD Framework	124
7.4	Auto-Tuning Structured Light	127
7.4.1	Efficient Optical-Domain Implementation	130
7.4.2	Optical SGD for Low-SNR Scenes	132
7.4.3	Optical SGD for Scenes with Indirect Light	133
7.4.4	Optical SGD for Color Structured Light Systems	134
7.5	Experiments	135
7.5.1	Quantitative Comparison with State-of-the-art Methods	135
7.5.2	Indirect Light Experiment	135
7.5.3	OpticalSGD for Different Imaging Systems	137
7.5.4	Operating Range of an Auto-Tuned System	138
7.6	Summary	141
8	Conclusion	142
8.1	Learned Lessons	142
8.2	Future Work	144
A	Supplementary Material for DIB-R	146
A.1	Derivation of DIB-R Renderer	146
B	Supplementary Material for DIB-R++	148

B.1	BRDF Model for DIB-R++	148
C	Supplementary Material for OpticalSGD	150
C.1	Proof of Eq.(7.16)	150
C.2	List of Devices & Encoding-Decoding Methods	152

List of Tables

3.1	Differentiable vertex attributes and rendering models supported by DIB-R. With our method, we can differentiate most common attributes (Column 1 & 4, Vertex attributes and Fragment attributes) and apply it into multiple rendering models (Column 5, Rendering Models).	35
3.2	Results on single image 3D object prediction reported with 3D IOU (%) / F-score (%).	41
3.3	Results for texture and light prediction. Texture/Texture+Light shows L_1 loss on the rendered image for texture/texture+lighting. Lighting shows the angle between predicted lighting and GT lighting. Lower is better.	44
4.1	Rendering time comparison. We render a sphere into 256×256 images and compare the rendering time under different settings. In MC shading, more sample count costs higher rendering time, while in SG shading, a different numbers of SG component barely impacts timings.	59
4.2	Comparison with Mitsuba2. We evaluate the running time and memory within ray tracing optimization task. Our method performs faster and requires less memory compared to Mitsuba2.	59
4.3	Quantitative results of single image 3D Reconstruction on synthetic data. While all the methods achieve comparable performance on re-rendered images and 2D IoUs, both MC and SG achieve better results on lighting and texture. MC is particularly better for metallic surfaces, and SG works best for glossy surfaces.	69
4.4	Comparison of shading methods.	76
4.5	Ablation study of number of SG components.	76
4.6	Training variance.	77
5.1	(a): We compare dataset size and annotation time of Pascal3D with our StyleGAN dataset. (b): We evaluate re-projected 2D IOU score of our StyleGAN-model vs the baseline Pascal3D-model on the two datasets. . . .	93

5.2	User study results: (a): Quality of 3D estimation (shape, texture and overall). (b): Annotators agreement analysis. “No agreement” stands for the case where all three annotators choose different options.	94
5.3	Comparison of different camera initialization methods. First table shows annotation time required for the StyleGAN dataset, and training times of the <i>LazyInit</i> -model and <i>SfM</i> -model on the dataset with respective annotations (binned viewpoints or cameras computed with SFM from annotated keypoints). The <i>LazyInit</i> -model requires significantly less annotation time, and its final performance is comparable to the <i>SfM</i> -model. Second table shows the difference of the camera parameters after training both methods (which optimize cameras during training). They converge to very similar camera positions. This shows that coarse view annotation along with camera optimization during training is sufficient in training high accuracy inverse graphics networks.	96
7.1	Devices and penalty functions compatible with our framework. † indicates the choices we validate experimentally.	124
C.1	List of the experimental imaging systems.	152
C.2	List of the coding-decoding methods used for experiments.	152

List of Figures

1.1	Overview. This thesis proposes a new reconstruction framework which reformulates the conventional, non-differentiable imaging systems in a differentiable way and embeds them into 3D reconstruction algorithms. As a consequence, it bridges imaging systems and reconstruction algorithms together, which enables information (<i>e.g.</i> , gradients) to be propagated bidirectionally and brings mutual enhancement. The proposed framework allows reconstruction algorithms to utilize inherent geometric or physical rules inside imaging systems for better recovery of 3D properties. Additionally, it enables tuning imaging systems to explore the best settings they should adopt under specific reconstruction tasks.	3
2.1	BRDF Illustration. The BRDF $f_r(\omega_i, \omega_o)$ from incoming direction ω_i to the outgoing direction ω_o is defined as $f_r(\omega_i, \omega_o) = \frac{dL_o(\omega_o)}{L_i(\omega_i) \mathbf{n} \cdot \omega_i d\omega_i}$	15
2.2	Single-view 3D Object Reconstruction. Given an input image, we could employ a neural network to predict geometry, light, and texture. During training we render the prediction back to a new image. We then compute the 2D image loss between input image and rendered prediction to train the network. Therefore, no 3D supervision is needed.	21
2.3	Structured Light System Overview.	23
2.4	Structured Light Pattern Examples. We show two sets of manually designed structured light patterns ($K = 4$). Left is Micro Phase Shifting (MPS) Patterns [74], where the frequency $\omega = 16$. Right is Hamiltonian [78] designed by maximizing Hamiltonian path.	25
3.1	Illustration of Our Differentiable Rasterization.	31
3.2	Illustration of Our Rendering Pipeline. (a) 3D mesh. (b) Rendered RGB image. (c) Rendered silhouette image with δ as $1.5e-4$. (d) Rendered silhouette with δ as $1.5e-3$. (e) Rendered silhouette with δ as $1.5e-5$. (f) Rendered silhouette from SoftRas[154]	33

3.3	DIB-R Sanity Check. We perform a sanity check for DIB-R by running optimization over several mesh attributes. We optimize w.r.t. different attributes in different rendering models: (a,b) vertex position and color in the vertex color rendering model, (c,d) texture and texture coordinates in the texture rendering model, (e,f) vertex and camera position in Lambertian model, (g) lighting in the Spherical Harmonic model, (h) material in the Phong model.	35
3.4	Full Architecture of Our Approach. Given an input image, we predict geometry, light, and texture. During training we render the prediction with a known camera. We use 2D image loss between input image and rendered prediction to train our prediction networks. Note that the prediction can vary in different rendering models, e.g. texture can be vertex color or a texture map while the lighting can be Lambertian, Phong or Spherical Harmonics.	36
3.5	Qualitative Results on Single-view 3D Object Reconstruction. First and fifth columns are the ground-truth image, the second and sixth columns are the prediction from our model, the third and seventh columns are results from SoftRas [154], the rest two columns are results from N3MR [122]. . .	41
3.6	Qualitative Comparisons on Single-view 3D Object Reconstruction. First column is the ground-truth image, the second and third columns are the prediction from our model, the fourth and fifth columns are results from SoftRas [154], the last two columns are results from N3MR [122].	42
3.7	Qualitative Results on Single-view 3D Object Reconstruction from Different Views. The first column is the ground truth, the rest columns are reconstructed object rendered in different views.	43
3.8	Qualitative Examples for 3D Shape, Texture, and Light Prediction. Col. 1-3: 1) GT rendered image with texture+light, 2) texture only rendered image, 3) light map. Col 4-6: our predictions. Col: 7-9: N3MR [122] . . .	44
3.9	Qualitative Comparison of Texture and Light with N3MR [122]. We show 9 examples. In each example, Purple rectangle: Input image. First Row: Ground Truth. Second Row: Prediction with DIB-Render. Third Row: Prediction of N3MR. First column: Texture and Light. Second column: Texture. Third column: Light. We demonstrate significantly better shape, texture and lighting predictions.	45

3.10	Qualitative Comparison of Texture and Light for the Model Trained w. and w.o. Adversarial Loss. We show two examples. In each example, Purple rectangle: Input image. First Row: Ground Truth. Second Row: Prediction with adversarial loss. Third Row: Prediction without adversarial loss. First column: Texture and Light. Second column: Texture. Third column: Light. Forth to Ninth column: Renderings from different views.	46
3.11	Qualitative Examples for 3D Shape, Texture and Light Prediction for Spherical Harmonic Model. We show three examples. In each example, Purple rectangle: Input image. First Row: Ground Truth. Second Row: Our Predictions. First column: Texture and Light. Second column: Texture. Third column: Light. Forth to Ninth column: Texture from different views.	46
3.12	Qualitative Examples for 3D Shape, Texture and Light Prediction for Phong Model. We show four examples. In each example, Purple rectangle: Input image. First Row: Ground Truth. Second Row: Our Predictions. First column: Texture and Light. Second column: Texture. Third column: Light. Forth to Ninth column: Texture from different views.	47
3.13	Light & Texture Separation Study for Varying Views. We show three sets of cars. In each set, we show predictions from different views. Purple rectangle: Input image. First Row: Ground Truth. Second Row: Our Predictions. First column: Texture and Light. Second column: Light. We show our model have consistent predictions when the input images are with same light and texture but varying views.	48
3.14	Light & Texture Separation Study for Varying Lighting Directions. We show three sets of cars. In each set, we show predictions from different lighting directions. Purple rectangle: Input image. First Row: Ground Truth. Second Row: Our Predictions. First column: Texture and Light. Second column: Light. Third column: Texture. We show our model have consistent predictions when the input images have varying lighting directions.	49
3.15	Shininess Separation Study. We render the same car with two shininess values, as shown in left 3 columns and right 3 columns. Purple rect: Input image. First Row: Ground Truth. Second Row: Our Predictions. First column: Texture and Light. Second column: Texture. Third column: Light. The network predicts incorrect shininess constants and compensates the shininess effects in texture maps, especially in the second example.	49

4.1	Overview. Given a 3D object \mathcal{M} , we employ (a) a rasterization-based renderer to obtain diffuse albedo, surface normals and mask maps. In the shading pass (b), we then use these buffers to compute the incident radiance by sampling or by representing lighting and the specular BRDF using a spherical Gaussian basis. Depending on the representation used in (c), we can recover a wide gamut of specular/glossy appearances (d).	54
4.2	Sample Count vs. Roughness in Monte Carlo Shading. High roughness surfaces require more samples to render into noise-free images.	57
4.3	Number of SG Light Component. More components (larger K) result in better environment map approximation.	58
4.4	Visual Comparisons Between MC and SG Shading. On the left, we show an environment map and its SG-representation ($K = 128$). While losing sharp details, SGs only needs 1% parameters compared to the dense pixel HDR map. On the right, we increase roughness left-to-right, revealing that our spherical basis can correctly approximate direct illumination at moderate-to-high roughness.	58
4.5	Independent Optimization. We optimize complex lighting and material for both MC shading (left) and SG shading (right). In each case, we first optimize lighting and show the light and rendered image in the first block. Next we optimize the surface material and show the surface material value and rendered image in the second block.	60
4.6	Disentanglement Analysis. We render a sphere with 1) large roughness and sharp light (top row) and 2) small roughness and smooth light (bottom row), but their renderings are almost the same. It shows the ambiguity between light and surface material.	61

4.7	<p>Monte Carlo Shading Optimization Results. We fix the shape and optimize the environment map, diffuse albedo and surface roughness jointly from a foreground image loss only. To validate the convergence of different surface materials, we use the same environment map and albedo initialization but employ different roughness initialization and optimize for different GT roughness settings. Top: In the first two rows, we show the GT and initialized environment map, texture and rendering in 6 views. Bottom: In each row we optimize for one specific roughness, and in each column block we start with different roughness initialization. Comments: We successfully optimize all the parameters such that the rendered images (Col. 4 & 7) are very close to the GT rendering (Col. 1). However, due to the ill-posedness of the problem, the converged environment map, diffuse albedo roughness are not exactly the same as GT.</p>	62
4.8	<p>Spherical Gaussian Shading Optimization Results. We fix the shape and optimize the environment map and surface material properties (roughness and specular albedo) jointly from a foreground image loss only. To validate the convergence of different surface material, we use the same environment map and texture initialization but different material initialization and optimize for different GT material settings. Top: In the first two rows we show GT and initialized environment map, diffuse albedo and rendering in 6 views. Bottom: In each row, we optimize for a specific GT material setting while in each column block we start from different surface material initialization. Comments: We successfully optimize all the parameters such that the rendered images (Col. 4 & 7) are very close to the GT rendering (Col. 1). However, due to the problem is ill-posed. the converged environment map, diffuse albedo and roughness are not exactly the same as GT.</p>	63
4.9	<p>Training Pipeline. Given an image, we employ a neural network to predict all 3D attributes. We then adopt DIB-R++ to render them back to the images and supervise via the loss between the rendered image and input image. This model can be trained without any 3D supervision.</p>	64
4.10	<p>Metallic Surface Dataset Overview. We render various cars under both indoor and outdoor environment maps. We set the m as 1 and β as 0 and the rendered images are highly reflective. The reflected environment maps can be observed in the car body, which provide strong signals in lighting estimation.</p>	66

4.11	Glossy Surface Dataset Overview. We render various cars under both indoor and outdoor environment maps. We set the $m = 0$ and vary $\beta \in [0, 0.5]$ so that the rendered images contain view-dependent highlights.	67
4.12	Prediction on the Metallic Car Dataset ($\beta = 0$). While the re-rendered images look similar, the underlying components (material and lighting) are different. A SH basis (DIB-R [38]) cannot recover the high frequency details of the sky light maps. In this case, MC performs best due to low variance in the estimator for mirror-like BRDFs. SGs can recover the overall form and contrast of the light map, but tend to predict incorrect texture maps incorporating the ground dominant color (e.g., brown). Note that GT texture maps are not available as they cannot be compared due to different uv -parameterizations / texture atlases.	70
4.13	Prediction on the Metallic Car Dataset ($\beta = 0$). Top and Middle: We show more comparison results for 3 different shading methods. While all re-rendered images look almost identical, the underlying radiometric components (material and lighting) are different. A SH basis(DIB-R [38]) cannot recover the high frequency details of the sky light maps. In this case, MC performs best due to low variance in the estimator for mirror-like BRDFs. SGs can recover the overall form and contrast of the light map, but tend to predict incorrect texture maps incorporating the ground dominant color. Bottom: We show a failure case in the bottom row. MC shading fails when the environment map contains a lot of high frequency details. It reconstructs low frequency content but also merges the high frequency leaves into the texture.	71
4.14	Prediction on the Glossy Car Dataset ($\beta > 0$). When the objects are glossy but not perfectly specular, SGs can correctly disentangle reflectance from lighting, as evidenced by the absence of white highlights in the predict albedos. DIB-R [38] cannot capture these bright regions due to a diffuse-only shading model, while MC oversmooths the predictions due to noisier pixel gradients. While all methods cannot completely reconstruct the environment map, our method can predict the correct dominant light location and sky color.	72

4.15	Prediction on the Glossy Car Dataset ($\beta > 0$). Top and Middle: We show more comparisons for three different shading methods. When the objects are glossy but not perfectly specular, SGs can correctly disentangle reflectance from lighting, as evidenced by the absence of white highlights in the predict albedos. DIB-R [38] cannot capture these bright regions due to a diffuse-only shading model, while MC oversmooths the predictions due to noisier pixel gradients. While all methods cannot completely reconstruct the environment map, our method can predict the correct dominant light location and sky color. Bottom Row: We show a failure case in the bottom row. All the methods fail when the texture contains a lot of high frequency details. We show SG only reconstructs low frequency content.	73
4.16	Specular effect of SG Shading. We decompose the prediction into diffuse albedo, light and material parameters, then visualize the specular contribution in other views and compare with GT (last 4 columns). Although the predicted material parameters do not match GT, interestingly, the specular renderings align well with each other.	74
4.17	Ablation Study of Shading Methods. Even though we apply the same lighting representation to DIB-R [38], due to the lack of specular transport support, [38] cannot perfectly disentangle the specular light from diffuse albedo. Its window and front cover still “bake in” white specular light. Moreover, the lighting is also blurry, as opposed to DIB-R++. This demonstrates the necessity of explicitly modelling advanced lighting effect in the renderer.	75
4.18	Ablation Study of Number of SG Components. As the number of SG components increase, the predicted lighting is more and more similar to the GT lighting and the color of predicted texture becomes more and more visually uniform, which indicates better disentanglement.	77
4.19	Ablation Study of Loss Terms. Similar to [291], we also find \mathcal{L}_{lap} regularizes the geometry to be smooth while \mathcal{L}_{per} helps create high-frequency, detailed textures.	77
5.1	Overview. We exploit StyleGAN as a synthetic data generator, and label the generated data in an extremely efficient way. This “dataset” is used to train an inverse graphics network that predicts 3D properties from single-view images.	82

5.2	StyleGAN Viewpoint & Content Disentanglement. We show examples of cars (first two rows) synthesized in chosen viewpoints (columns). To get these, we fix the latent code w_v^* that controls the viewpoint (one code per column) and randomly sample the remaining dimensions of (Style)GAN’s latent code (to get rows). Notice how well aligned the two cars are in each column. In the third row we show the same approach applied to horse and bird StyleGAN.	83
5.3	All Viewpoints. We show an example of a car, a bird and a horse synthesized in all of our chosen viewpoints. While shape and texture are not perfectly consistent across views, they are sufficiently accurate to enable training accurate inverse graphics networks in our downstream tasks. Horses and birds are especially challenging due to articulation. One can notice small changes in articulation across viewpoints. Dealing with articulated objects is subject to future work.	84
5.4	Dataset Overview. We synthesize multi-view datasets for three classes: <i>car</i> , <i>horse</i> , and <i>bird</i> . Our datasets contain objects with various shapes, textures and viewpoints. Notice the consistency of pose of object in each column (for each class). Challenges include the fact that for all of these objects StyleGAN has not learned to synthesize views that overlook the object from above due to the photographer bias in the original dataset that StyleGAN was trained on.	85
5.5	Annotation Comparisons. We show two different annotation methods: LazyInit and SfM. Top row shows the classified pose bin annotations, while the images show the annotated keypoints. For 39 viewpoints car class, annotating pose bins took 1 min, while keypoint annotation took 3-4 hours. We empirically find that pose bin annotation is sufficient in training accurate inverse graphics networks (when optimizing camera parameters during training in addition to optimizing the network parameters).	87
5.6	Shape & Texture Reconstruction Results. Given input images (1st column), we predict 3D shape, texture, and render them into the same viewpoint (2nd column). We also show renderings in 3 other views in remaining columns to showcase 3D quality. Our model is able to reconstruct cars with various shapes, textures and viewpoints. We also show the same approach on harder (articulated) objects, i.e., bird and horse.	90
5.7	Comparison on Pascal3D Testing Set. We compare inverse graphics networks trained on Pascal3D and our StyleGAN dataset. Notice considerably higher quality of prediction when training on the StyleGAN dataset.	90

5.8	3D Reconstruction Results for Car, Horse, and Bird Classes. We show car, horse and bird examples tested on the images from the StyleGAN dataset test sets. Notice that the model struggles a little in reconstructing the top of the back of the horse, since such views are lacking in training.	91
5.9	Comparison on PASCAL3D Imagery. We compare PASCAL-model with StyleGAN-model on PASCAL3D test set. While the predictions from both models are visually good in the corresponding image view, the prediction from StyleGAN-model have much better shapes and textures as observed in other views.	92
5.10	User Study Interface (AMT). Predictions are rendered in 6 views and we ask users to choose the result with a more realistic shape and texture that is relevant to the input object. We compare both the baseline (trained on Pascal3D dataset) and ours (trained on StyleGAN dataset). We randomize their order in each HIT.	94
5.11	Ablation Study. We ablate the use of multi-view consistency and perceptual losses by showing results of 3D predictions. Clearly, the texture becomes worse in the invisible part if we remove the multi-view consistency loss (rows 2, 5, denoted by “w.o M. V.”, which denotes that no multi-view consistency was used during training), showcasing the importance of our StyleGAN-multiview dataset. Moreover, the textures become quite smooth and lose details if we do not use the perceptual loss (rows 3, 6, noted by “w.o P.”, which denotes that no perceptual loss was used during training).	95
5.12	Comparison of Different Camera Initialization Methods. The first row shows predictions from <i>SfM</i> -Initialization (cameras computed by running SFM on annotated keypoints) and the second row show results obtained by training with <i>LazyInit</i> -Initialization (cameras are coarsely annotated into 12 view bins). Notice how close the two predictions are, indicating that coarse viewpoint annotation is sufficient for training accurate inverse graphics networks. Coarse viewpoint annotation can be done in 1 minute.	95
5.13	3D Reconstruction Failure Cases. We show examples of failure cases for car, bird and horse. Our method tends to fail to produce relevant shapes for objects with out-of-distribution shapes (or textures).	97
5.14	Results on Real Imagery from the StyleGAN-generated Car Dataset. In Row(1,3), we show DIB-R++ can recover a meaningful decomposition as opposed to DIB-R (Row(2,4)), as shown by cleaner texture maps and directional highlights (e.g., car windshield).	98

5.15	Prediction on LSUN Dataset (Cars). DIB-R++, trained on StyleGAN dataset, can generalize well to real images. Moreover, it also predicts correct high specular lighting directions and usable, clean textures.	99
5.16	Material Editing. Our method allows for artistic manipulation of appearance, such as novel view synthesis, material editing (both diffuse and specular components), and relighting, thanks to our effective disentanglement. . .	99
5.17	Prediction on Real-world Dataset (Cars). DIB-R++ accounts for high specular light and always have cleaner textures compared to DIB-R	100
5.18	Prediction on LSUN Dataset (Cars). Our model, trained on StyleGAN dataset, can be well generalized to real images. Moreover, it also accounts for high specular light and predict correct lighting directions and clean textures.	101
6.1	Overview. <i>Top:</i> A projection pattern is a 1D image projected along a projector’s rows. A sequence of them defines a code matrix, whose columns encode pixel position. We present a framework for computing stereo correspondences using <i>optimal code matrices</i> , which we generate on the fly. These matrices minimize the expected number of stereo errors that occur when the individual matrix columns are not very distinctive (red=similar, blue=dissimilar). <i>Middle:</i> A whole space of optimal matrices exists, for different numbers of projection patterns, image signal-to-noise ratio, spatial frequency content (sample patterns shown above), <i>etc.</i> <i>Bottom:</i> We use two automatically-generated four-pattern sequences to compute the depth map of the object shown on left. Both are optimized for a one-pixel tolerance for stereo errors, without (middle) and with (right) a bounding-box constraint. Both depth maps are unprocessed (please zoom in).	103
6.2	Viewing Geometry. We assume the projector-camera system has been rectified, <i>i.e.</i> , epipolar lines are along rows.	105
6.3	Generative Model of Image Formation for a Single Epipolar Line Across K Images. Each column of matrix \mathbf{O} is an observation vector (red) and each row collects the observations from a single image across all pixels on the epipolar line (yellow). All yellow rows are associated with the same input image and all red columns are associated with the same camera pixel q . The gray column and row are associated with the same projector pixel p	106

6.4 **Geometric Constraints.** (a) Top view of the epipolar plane. (b) \mathbf{T} is always lower triangular because the 3D rays of all other elements intersect behind the camera. (c) \mathbf{T} 's non-zero elements are restricted even further by knowledge of the working volume (*e.g.*, black square in (a)): its depth range (red) and its angular extent from the projector (green) and the camera (blue) define regions in \mathbf{T} whose intersection contains all valid correspondences. . . . 108

6.5 **ZNCC v.s. Native Decoding.** *Left:* We project K micro phase shifting (MPS) patterns [74] of maximum frequency F onto a known planar target and compute correspondence errors using our ZNCC decoder (red) and the one by the MPS authors (black). *Right:* A similar comparison for 10 Gray codes (purple) and 10 XOR-04 codes (green), projected along with their binary complement. We used the binarization technique in [219] for “native” decoding. Since these codes have no frequency bound we plot them against image PSNR. In all cases, ZNCC decoding yields comparable results. . . . 109

6.6 **A Walk in the Space of Optimal Codes.** To better visualize code structure, the pairwise scores $\text{ZNCC}(c_i, c_j)$ of code vectors are shown as a jet-colored matrix (deep red = 1, deep blue = -1). These can be treated as a confusion matrix. *Row 1:* We set the maximum spatial frequency of the patterns to $F = 4$ and the image PSNR to be maximal for our imaging conditions (frame rate=50Hz, camera gain=1, known read noise, pixel intensity that spans the full interval $[0, 1]$). We then compute the optimal code matrix for our 608-pixel projector for different numbers of patterns and no other constraints. *Row 2:* We then choose $K = 4$ (outlined in red in Row 1) and compute optimal matrices for different bounds on the maximum spatial frequency, with everything else fixed as above. *Row 3:* We now set the frequency to 8 (outlined in red in Row 2) and compute optimal matrices for different values of pixel PSNR (*i.e.*, the maximum image intensity gets increasingly smaller), again with everything else fixed as above. *Rows 4 and 5:* We follow the exact same process for different lower bounds on disparity (*i.e.*, the maximum scene depth is increasingly being restricted), and different tolerances in correspondence error. 113

6.7 **Quantitative Evaluation.** *Top row and first two columns of bottom row:* Each data point represents three independent acquisitions with the same pattern sequence (x-axis is frequency). Error bars indicate the smallest and largest fraction of correct correspondences in those runs. We used $\epsilon = 0$ for optimization in the top row and $\epsilon = 1$ in the bottom. Solid lines show results when no geometry constraints are imposed on code optimization and on decoding. Dashed lines show what happens when we use a depth-constrained geometry matrix \mathbf{G} (Figure 6.4c). For EPS and MPS, the constraint is used only for decoding, *i.e.*, we search among the valid correspondences for the one that maximizes the ZNCC score. Our codes, on the other hand, are optimized for that constraint and decoded with it as well. *Bottom row, right: RMSE plots.* 115

6.8 **Qualitative Comparisons.** We acquired depth maps for the scenes on the left using three methods, with the same ZNCC decoder and the same triangular geometry matrix \mathbf{G} (Figure 6.4b). For each method, we reconstructed the scenes for several maximum frequencies in the range $[4, 32]$ and show depth maps for each method’s best-performing frequency. *Top row:* Reconstructing a dark, varnished and sculpted wooden trunk with five patterns. *Middle row:* Reconstructing a scene with significant indirect transport (a bowl, candle, and convex wedge) using conventional imaging and six patterns. *Bottom row:* Depth map acquired with many more patterns, along with cross-sections of the above depth maps (blue points) and a histogram of disparity errors (please zoom in to the electronic copy). For reference, we include the cross-sections of depth maps acquired using epipolar-only imaging [190] with the exact same patterns (green points), as well as of “ground truth” depth maps acquired with 160 shifted cosine patterns of frequencies 16 to 31 using epipolar-only imaging (red points). 116

6.9 **Hyper-parameter Tuning.** We show validation error (on 500 fixed random samples) over iterations in optimization of a sample code matrix of 4 patterns and 608 pixels with maximum frequency 16. 117

6.10	Code Performance as a Function of Camera Noise Model. Each column is patterns optimized under a specific noise model, while each row is a specific pattern setting. We show the <i>raw</i> depth maps in top left and the confusion matrix in the middle right. To assess it quantitatively, we also compare 2D slices of the reconstructed 3D pointset against those computed by the ground-truth sequence, and show the histogram of differences in the computed and ground-truth disparities in the bottom part. Please zoom into the electronic copy for details.	118
7.1	Top: <i>Optimal structured light with smartphones.</i> We placed a randomly-colored board in front of an Optoma 4K projector and a Huawei P9 phone, let them auto-tune for five color-stripe patterns and the 1-tolerance penalty (Table 7.1), and used the resulting patterns (middle) to reconstruct a scene (inset). Middle & Bottom: <i>Auto-tuning systems for 4 patterns on various imaging systems.</i> Note the patterns’ distinct spatial structure and frequency content, especially for Episcan3D [187] which employs a scanning-laser projector.	120
7.2	Algorithm Illustration. Differentiable imaging systems allow us to “probe” their behavior by differentiating them in the optical domain, <i>i.e.</i> , by repeatedly adjusting their control vector, taking images, and computing image differences. Projector-camera systems, as shown above, are one example of a differentiable system where projection patterns play the role of control vectors. Many other combinations of programmable sources and sensors have this property (Table 7.1).	122
7.3	Numerical vs. Optical-domain Implementation of SGD, with Red Boxes Highlighting Their Differences.	125
7.4	Decoder for K-pattern Triangulation.	127
7.5	Image Formation in General Projector-camera Systems. The projector function <code>proj()</code> maps a control vector of digital numbers to a vector of outgoing radiance values. Similarly, the camera function <code>cam()</code> maps a vector of sensor irradiance values to a vector holding the processed image.	128
7.6	Optical-domain Differentiation on an Actual Projector-camera System. (a) 1D plot of a sample projection pattern <code>c</code> and its corresponding 2D projection pattern. (b) a photo of the experimental setup and the corresponding camera image of the scene under the projection pattern depicted in (a). (c) adjustment vector and its corresponding image difference.	132

7.7	Optical SGD in Action for the LG-IDS Pair and the Training Board in Figure 7.1. Top: The red graph shows the progress of the optimization objective (Eq. 7.5) across iterations when auto-tuning on the training board for four patterns, the zero-tolerance penalty, and the ZNCC-NN ₃ decoder. The green graph shows $\ \text{err}(\mathbf{d}, \mathbf{g})\ _1$ as a function of iteration for the previously-unseen (and much more challenging) test scene below. Middle: Visualizing the evolution of pattern \mathbf{c}_1 as a grayscale image whose i -th column is the pattern at iteration i . Bottom: Three snapshots of the optimization, each showing the patterns at iteration i ; the disparity map of the training board (inset) reconstructed from those patterns; and the disparity map of the test scene reconstructed from the same patterns.	133
7.8	Comparison of Different Coding-decoding Methods based on Percentage of Zero-error Pixels. The table lists the no-error reconstruction rates (<i>i.e.</i> the percentage of pixels with zero error) of each pattern, decoded with all the decoding methods. Their corresponding 0-tolerance disparity maps (pixels with zero error are drawn in the figures) are shown with the same layout as the table. The raw disparity maps are also shown as insets, for reference. Table entries and disparity maps marked as blue represent the current state of the art. Entries and disparity maps marked as green indicate the best-performing decoder for 3 previously-proposed pattern sequences (MPS, Hamiltonian, a la carte). Note that the best results of these patterns are obtained with decoders introduced in this paper. The best performance, shown in red, is obtained by auto-tuning with the ZNCC-NN ₅ decoder. . . .	136
7.9	Auto-tuning for Indirect Light. To better visualize reconstruction accuracy, only pixels with error ≤ 2 are shown in the disparity maps above. The total percentage of such pixels is indicated in yellow in the upper left, with the correspondence error map shown as an inset (darkest blue for error=0, darkest red for error ≥ 20).	137
7.10	Auto-tuning Mitsuba CLT for Four Patterns and the 0-tolerance Penalty. Left: Performance of optimized patterns and ZNCC-NN ₃ decoder across iterations of optical SGD. We measure performance by reconstructing the virtual training board (red plot) as well as ModelNet objects (green plot, averaged over 30 models). Optical SGD performs considerably better on ModelNet than state-of-the-art patterns combined with our ZNCC ₃ decoder (dashed lines). Right: Disparity maps for a sample model.	138

7.11	Comparison of Auto-tuned Color Patterns with Gray Patterns. Top: Auto-tuning gray patterns for LG-AVT. (a) scene image captured by the monochrome camera. (b) 0-tolerance ground-truth. (c-d) 0-tolerance disparity map (overlaid with raw disparity map) for $K = 4$ and $K = 3$ with their percentages of zero-error pixels. Bottom: Auto-tuning color patterns for LG-AVT. (a) scene image captured by the color camera. (b) 0-tolerance ground-truth. (c) 0-tolerance disparity map (overlaid with raw disparity map) for $K = 3$ color pattern, auto-tuned on demosaiced images with its percentage of zero-error pixels. (d) 0-tolerance disparity map (overlaid with raw disparity map) for $K = 3$ color pattern, auto-tuned on raw images with its percentage of zero-error pixels.	139
7.12	Comparison of Auto-tuned Patterns Optimized on Different Geometrical Arrangements with Existing Patterns.	140
7.13	Reconstructing a Room Corner from Different Standoff Distances After Auto-tuning for a Specific Distance (or a Range of Distances). We observe that the frequency content of patterns optimized for 0.8m (red) is much higher than those for 0.8-5.8m (green).	141
A.1	Illustration of Our Differentiable Rasterization.	146

Chapter 1

Introduction

3D reconstruction aims at inferring 3D-related properties of a scene, such as geometry, texture, material, and light. As a long-standing goal in computer vision and computer graphics, it serves as key in a variety of important applications like virtual reality (VR), augmented reality (AR), robotics, autonomous driving, *etc.* For example, in AR games, acquiring 3D properties of a scene is the prerequisite for user interaction. Moreover, in more rigorous cases, 3D reconstruction also plays an essential role, *e.g.*, to make robots or autonomous driving cars safely ride on the road, it is necessary to accurately model the surrounding environment.

The input data of 3D reconstruction are often 2D measurements (*e.g.*, images) captured by imaging systems, in a process called *imaging*. In the past centuries, people have developed various imaging systems to facilitate 3D reconstruction. Created in the 1840s [4], cameras are the most popular choice. Later, structured light (SL) [61] and time of flight (ToF) [56] setups emerged to infer scene geometry by incorporating active light sources in image capture. Today, there are continuous efforts to explore new imaging systems like transient imaging or lensless imaging systems [25, 107], which further push the boundaries of 3D reconstruction, allowing occluded object reconstruction or direct 3D imaging [14, 254].

Conceptually, imaging and 3D reconstruction are inverse procedures. The former chooses a specific imaging system to measure scene properties, where it encode various 3D-related properties into measurements. For example, images can be formulated as 2D projections of 3D geometry and material interacting with light. On the contrary, the latter applies 3D reconstruction algorithms to the captured data in order to recover all kinds of 3D properties back from the measurements.

Being the reciprocal step, imaging process contains an abundance of valuable information

that could significantly benefit 3D reconstruction. Such information is commonly referred as *imaging priors*. For instance, image formation models explain how imaging devices record 3D properties of a scene. Imaging settings, like exposure, gain, or tone mapping, influence the quality of captured data [117]. System illuminations, which are the light sources in the active imaging systems, affect the performance of structured light or time of flight systems [61, 56]. Therefore, exploring better ways to utilize imaging priors is critical to improve performance of 3D reconstruction.

A large body of work has been done to address problems of 3D reconstruction, ranging from classical geometrical or optical methods [83, 61, 56] to recent deep learning based approaches [131, 65]. Particularly, the advances of deep learning [127, 111, 228, 48] have ushered in many breakthroughs in 3D reconstruction. For example, detailed 3D shapes, high-fidelity texture maps, accurate lighting and material parameters are all inferable from 2D images via neural networks models [53, 70, 260, 267, 230, 264, 288, 171, 147]. With the help of massive training data, deep learning based methods significantly outperform the classical works [159, 19, 46, 212].

While great progress has been made, imaging priors are still under-explored in 3D reconstruction, in particular deep learning based approaches. As a physical process or a simplified physical simulation [5], imaging process is generally not differentiable. However, the non-differentiable property makes it difficult to be involved in deep learning frameworks. As a consequence, imaging and reconstruction are treated as discrete procedures, which impedes imaging priors propagating from the former to the latter, leading suboptimal designs in both stages. Existing methods either utilize imaging in data preparation [235] or adopt imaging constraints as regularization terms [261, 264]. Still, a large portion of imaging priors are ignored and not fully utilized in deep learning.

The focus of this thesis is on exploring a new way to exploit information of imaging process in deep learning based reconstruction algorithms. To this end, we introduce *differentiable imaging systems*. Specifically, an imaging system is defined as *differentiable*, if its imaging process is differentiable, *i.e.*, we are able to compute the gradients of the imaging measurements with regard to the scene properties or the imaging parameters. While conventional imaging systems are not designed to be differentiable, it is challenging to acquire the analytical gradients. Instead, we construct an differentiable imaging system by either approximating its traditional non-differentiable imaging process with a differentiable surrogate [38, 39, 172], or estimating the gradients with numerical methods [40].

The main advantage of differentiable imaging systems is that they can be embedded into 3D reconstruction algorithms, in particular deep learning based reconstruction approaches.

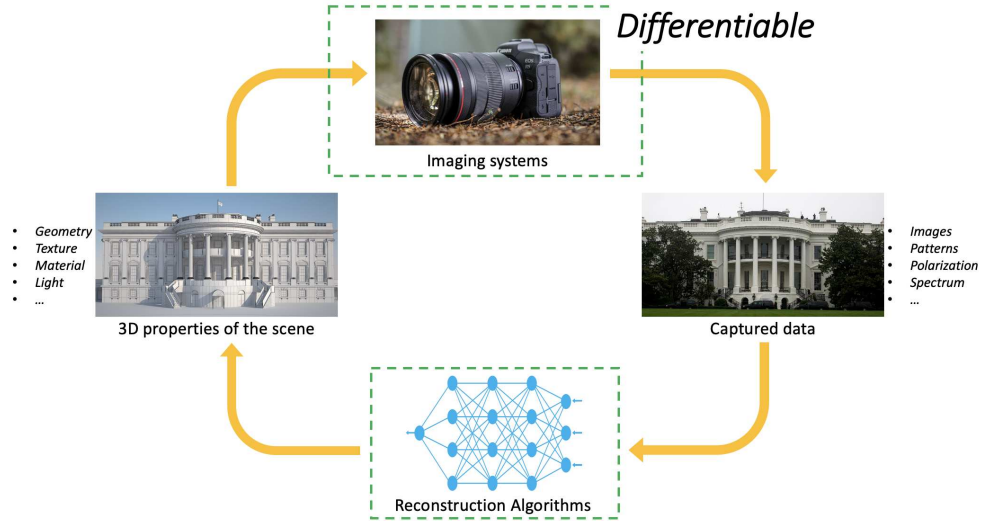


Figure 1.1: **Overview.** This thesis proposes a new reconstruction framework which reformulates the conventional, non-differentiable imaging systems in a differentiable way and embeds them into 3D reconstruction algorithms. As a consequence, it bridges imaging systems and reconstruction algorithms together, which enables information (*e.g.*, gradients) to be propagated bidirectionally and brings mutual enhancement. The proposed framework allows reconstruction algorithms to utilize inherent geometric or physical rules inside imaging systems for better recovery of 3D properties. Additionally, it enables tuning imaging systems to explore the best settings they should adopt under specific reconstruction tasks.

As shown in Fig. 1.1, our method bridges imaging systems and reconstruction algorithms together, which enables information (*e.g.*, gradients) to be propagated bidirectionally and brings mutual enhancement to each other.

On one hand, information from imaging systems is passed to reconstruction algorithms, which helps design algorithms to match the given imaging systems perfectly [249, 278, 230, 58, 280, 38, 39, 171, 155]. The optimized reconstruction approaches, built on top of imaging systems, could utilize the inherent geometric or physical rules inside the imaging systems, resulting in much better performance than the traditional methods which are generally blind to the imaging systems.

On the other hand, information is also propagated from reconstruction algorithms to imaging systems, helping explore the best settings they should adopt under specific reconstruction tasks, *e.g.*, discovering the most appropriate hardware parameters like camera settings [177, 248] or system illuminations [172, 40]. It enables automatically tuning imaging systems to be tailored for corresponding downstream tasks, which generally requires expert knowledge in previous works.

We validate the superiority of our method on two imaging systems: *rendering pipelines* and *structured light systems*. Specifically, in the first part of the thesis, we focus on applying differentiable rendering techniques in unsupervised, single-view 3D object reconstruction. We propose two efficient and photorealistic differentiable rendering frameworks [38, 39] and

incorporate them in neural networks to predict 3D properties from a single image. Compared to previous methods, our novel designs not only predict accurate shape and realistic texture, but also enable new capabilities to achieve faithful material and lighting disentanglement under challenging imaging conditions, as well as generalize to real imagery, predicting reliable 3D properties for real photographs [291].

Next, we discover optimal illumination patterns in structured light triangulation. By differentiating structured light systems and embedding them in reconstruction algorithms [172, 40], we are able to optimize patterns for chosen systems and desired imaging conditions. Unlike existing approaches that use predetermined patterns, which have no optimal guarantees and are blind to structured light system properties, the optimized patterns demonstrate precise match to the system properties and achieve substantial improvements in 3D triangulation accuracy.

Principally, the main advantage of differentiable imaging systems is that it allows information to be propagated bidirectionally. Namely, imaging priors can be utilized to boost reconstruction algorithms while learning constraints can also be applied to enhance imaging procedures. The two differentiable imaging systems discussed in this dissertation, *i.e.*, the differentiable rendering pipelines and the differentiable structured light systems, demonstrate advantages in two complementary directions.

The differentiable rendering part focuses on boosting 3D reconstruction algorithms with imaging priors. Following the principle of “analysis-by-synthesis”, the imaging priors guide the neural networks to predict faithful geometry, texture, material and light from a single image without any 3D supervision [38, 39, 291]. On the other side, the differentiable structured light part explores how to tune optimal imaging settings from learning constraints. Differentiable structured light systems allows themselves to be embedded in learning frameworks, where the gradients of the illumination patterns are adopted to optimize patterns to maximize the 3D reconstruction accuracy [172, 40]. Notably, imaging systems and reconstruction algorithms can be optimized jointly. For example, in [40] we jointly explore the best imaging parameters, as well as train a neural network that perfectly matches the given imaging system. In all cases, we show differentiable imaging systems bring significant performance improvement over prior arts.

In conclusion, this thesis makes the following contributions:

- In Chapter 3, we propose DIB-R [38], an interpolation-based differentiable rendering pipeline that allows gradients computation from 2D images to most 3D attributes, including vertex positions, vertex colors, multiple lighting directions and texture mapping through a variety of lighting models. We also apply DIB-R in single-view

3D object reconstruction tasks, which can be trained exclusively using 2D supervision. Compared to prior works, DIB-R demonstrates more faithful shape and realistic texture recovery, achieving new state-of-the-art results.

- In Chapter 4, we extend DIB-R to DIB-R++ [39], a hybrid differentiable renderer which combines rasterization and ray-tracing to take the advantage of their respective strengths—speed and realism. Being efficient and photorealistic, it enables new capabilities for geometry, reflectance and lighting prediction from a single image without requiring any ground-truth. DIB-R++ achieves superior material and lighting disentanglement compared to existing rasterization-based approaches, especially under challenging imaging conditions.
- In Chapter 5, we investigate how to apply differentiable rendering techniques to predict 3D properties for real imagery [291]. Differentiable rendering based inverse graphics models rely on multi-view imagery, which are not readily available in existing real datasets. Training with synthetic renderings often leads to a gap in performance when tested on real photographs. We propose to exploit StyleGAN [119] as a multi-view data generator, synthesizing infinite multi-view photorealistic images to train inverse graphics networks. Our approach achieves exquisite 3D reconstruction effects for real images and significantly outperforms models trained on existing datasets.
- In Chapter 6, we discover optimal illumination patterns in structured light triangulation [172]. Unlike existing approaches that use predetermined patterns, we propose à la carte, a new paradigm to optimize patterns for user-defined imaging conditions. By defining a simplified linear imaging formation model and embedding it into learning frameworks, à la carte derives objective functions to evaluate performance of the patterns and optimize them by minimizing the reconstruction error. We show the optimized patterns outperform state-of-the-art triangulation techniques.
- Lastly, in Chapter 7, we propose OpticalSGD [40], a computational imaging technique that optimizes the performance of an active imaging system by automatically discovering the illuminations it should use, and the way to decode them. OpticalSGD supports “tuning” the illuminations and decoding algorithm precisely to the system properties, and doing so without any requirement of system. The key idea of OpticalSGD is to formulate a stochastic gradient descent (SGD) optimization procedure that puts the *actual* system in the loop: adopting real devices to illuminate a scene and capture images, and computing the gradient of the reconstruction error and update the system parameters. We verify OpticalSGD in structured light triangulation and show the tuned system substantially boosts 3D reconstruction accuracy over state-of-the-art methods.

1.1 Organization

The chapters in this thesis are organized as follows. In Chapter 2, we provide background knowledge of the discussed imaging systems. In Chapter 3, Chapter 4, and Chapter 5, we investigate how to apply differentiable rendering techniques in unsupervised, single-view 3D object reconstruction. We first present an interpolation-based differentiable renderer, then extend it to a one-bounce ray-tracing based differentiable renderer, and finally talk about how to predict 3D properties from real imagery, respectively.

In Chapter 6 and Chapter 7, we discover optimal illumination patterns in structured light triangulation with differentiable structured lighting systems. Chapter 6 introduces a simplified differentiable structured light image formation model in simulation. Chapter 7 explores pattern optimization with hardware-in-the-loop. Finally, in Chapter 8, we discuss promising future directions and conclude the thesis.

1.2 Included Publication

The thesis mainly contains 5 earlier published papers, with proper re-organization, more technical details and results. The papers and author contributions are listed as follows. * denotes equal contribution:

Wenzheng Chen, Jun Gao, Huan Ling*, Edward J. Smith*, Jaakko Lehtinen, Alec Jacobson, Sanja Fidler: Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer, NeurIPS 2019* is reproduced and adapted in Chapter 3. Author contributions: the author conceived the idea and implemented the prototype. The author, Jun Gao, Huan Ling, and Edward J. Smith did the learning experiments. Prof. Lehtinen, Prof. Jacobson, and Prof. Fidler helped complete the idea. All the authors contributed to the manuscript writing and Prof. Fidler supervised the project.

Wenzheng Chen, Joey Litalien, Jun Gao, Zian Wang, Clement Fuji Tsang, Sameh Khamis, Or Litany, Sanja Fidler: DIB-R++: Learning to Predict Lighting and Material with a Hybrid Differentiable Renderer, NeurIPS 2021 is reproduced and adapted in Chapter 4. Author contributions: the author and Prof. Fidler conceived the idea. The author and Joey Litalien implemented the prototype. Clement Fuji Tsang did acceleration. Jun Gao, Zian Wang, Dr. Khamis, Dr. Litany, and Prof. Fidler helped complete the idea. All the authors contributed to the manuscript writing and Prof. Fidler supervised the project.

Yuxuan Zhang, Wenzheng Chen*, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, Sanja Fidler: Image GANs meet Differentiable Rendering for Inverse Graphics and Inter-*

pretable 3D Neural Rendering, ICLR 2021 is reproduced and adapted in Chapter 5. Author contributions: the author and Prof. Fidler conceived the idea. Yinan Zhang implemented the GAN dataset generation part. Yuxuan Zhang, the author, and Jun Gao implemented the 3D inference part. Yuxuan Zhang and Huan Ling implemented the GAN fine-tuning part. Prof. Torralba and Prof. Fidler helped complete the idea. All the authors contributed to the manuscript writing and Prof. Fidler supervised the project.

Parsa Mirdehghan, Wenzheng Chen, Kyros Kutulakos: Optimal Structured Light a la Carte, CVPR 2018 is reproduced and adapted in Chapter 6. Author contributions: Parsa Mirdehghan and Prof. Kutulakos conceived the idea. Parsa Mirdehghan and the author implemented the prototype. The author did all the captures. All the authors contributed to the manuscript writing and Prof. Kutulakos supervised the project.

Wenzheng Chen, Parsa Mirdehghan*, Sanja Fidler, Kyros Kutulakos: Auto-Tuning Structured Light by Optical Stochastic Gradient Descent, CVPR 2020* is reproduced and adapted in Chapter 7. Author contributions: Prof. Kutulakos conceived the idea. The author implemented the prototype and did all the captures. Parsa Mirdehghan and Prof. Fidler helped complete the idea. All the authors contributed to the manuscript writing and Prof. Kutulakos supervised the project.

Chapter 2

Background

In this section, we provide research background for the thesis. We first have a brief overview of imaging systems and 3D reconstruction algorithms in Sec. 2.1 and Sec. 2.2. Next, we introduce differentiable imaging systems (Sec. 2.3), including their advantages, differentiation and optimization designs. We focus on two specific imaging systems, where we talk about rendering pipelines in Sec. 2.4 and structured light systems in Sec. 2.5.

2.1 Imaging Systems

Optical imaging systems are developed to measure the properties of a scene, including its appearance, geometry, light, material, *etc.* In the past two centuries, various imaging systems have been created. As the most popular choice, digital cameras, such as those integrated on smartphone devices, are used by billions of people to take photos every day [6]. Besides that, numerous imaging systems are created, such as Structured Light (SL) [61], Time of Flight (ToF) [56], LiDAR (Light Detection and Ranging), Single-photon Avalanche Diode (SPAD) [107], so on and so forth.

The concept of imaging systems could be very broad. In addition to actual hardware devices, there are also imaging systems in simulation. For example, in computer graphics we have the rendering pipelines [5, 2] to “take photos” of 3D models, which simulate cameras in the virtual world. Recently, people further extend simulation to other imaging systems like SL [238] or ToF [106]. Imaging is also not limited to visible light spectrum but can be applied to infrared or ultraviolet cases, which is generally used in medical or astronomy imaging like the computed tomography (CT) or radio telescope. In this thesis, we mainly explore two imaging systems: the rendering pipelines and the structured light systems.

Although the measurements of imaging systems describe the scene properties, typically

they are measured after transformation, projection, lossy compression, *etc.* Consequently, the measured data typically records entangled 3D properties of the scene, rather than capturing each property independently. For example, the scene appearance recorded by images is actually decided by the scene geometry projection, lighting condition and surface material [241]. It might even be compressed with information loss when saving in JPEG format [258]. Therefore, recovering and disentangling each 3D property from the entangled measurements would be a challenging task. To achieve this goal, people designed various 3D reconstruction methods. They either follow optical or geometric rules, or adopt learning priors, which we discuss later.

2.2 3D Reconstruction Algorithms

Traditional 3D Reconstruction Methods 3D reconstruction is one of the most fundamental problems in computer vision and computer graphics. As mentioned earlier, the goal of 3D reconstruction is to recover 3D-related properties of a scene from measurements of various imaging systems. In most cases, the measurements refer to images.

Before the era of deep learning, 3D reconstruction is typically solved with optical or geometric prior knowledge. For example, optical methods adopt expensive temporal sensors to measure the time of photon travels (ToF, LiDAR) [56]. Since the light speed is a constant, we can therefore recover the geometry of the scene. On the other hand, geometric methods utilize multi-view stereo matching algorithms (structure from motion, structured light) [83, 241, 61]. If two pixels in two images are the projection of the same 3D point (which is called correspondence), we can compute its 3D location based on stereo geometric constraints. However, the performance of stereo matching is heavily influenced by correspondence matching accuracy [159, 46, 212].

Previous methods also explore single-view 3D reconstruction, which recover the scene geometry from a single-view image. As an ill-posed problem, it is addressed with both geometric constraints and learning priors like vanishing points, T-junction occlusion cues [90, 91, 109, 283]. Moreover, capturing light and material is a more challenging problem, which requires specially-designed, expensive equipment like lighting stage [134, 135, 24].

Deep Learning based 3D Reconstruction Methods Since 2012 [127], 3D reconstruction has been largely boosted by deep learning. It is demonstrated that, with the help of massive training data, neural networks can learn to predict from 2D images accurate 3D properties, including shape, texture, light and material [53, 70, 260, 267, 147, 149].

While great progress has been made, deep learning methods are dominated by supervised

methods which require images together with the corresponding ground truth 3D properties. For example, to recover 3D shapes from images with deep learning approaches, people have to prepare annotated 3D ground truth (GT) shapes for the input images. However, annotating 3D shapes for real images is an extremely challenging task [224]. The largest dataset till now is PASCAL3D [272], which contains only a few thousands of (*real image, 3D shape*) pairs and is not enough to train a high-performant, generalizable neural network. Moreover, the light and material dataset [165] contains even fewer examples, which makes the lighting and material estimation tasks more challenging for deep learning approaches.

Due to the data limitation, people always train 3D prediction models on synthetic datasets, in particular the ShapeNet dataset [33]. ShapeNet contains 51,300 unique 3D models. By applying computer graphics rendering pipelines, people can render 3D models into infinite images with the corresponding 3D property labels. Next, deep learning models can be trained on the rendered images to predict corresponding 3D properties. However, the models trained on synthetic data have a performance gap when applied to real images [70, 260, 267], due to the distribution deviation between real and synthetic images.

2.3 Differentiable Imaging Systems

As mentioned earlier, 3D reconstruction and imaging are reciprocal procedures. As the reciprocal step, imaging process describes how a imaging system records the 3D properties of a scene and produces specific measurements. It contains rich optical and geometric information that can be further utilized to promote the performance of 3D reconstruction.

A branch of works explore how to introduce imaging prior knowledge into deep learning. Some of them focus on applying optical and geometric prior knowledge as regularization terms and object functions [237, 161, 261], which has shown to help regularize the network and achieves strong performance boost. On the other side, instead of applying imaging priors as losses or regularization terms, recent works move one step further by re-formalizing the image formation models or the imaging systems in a differentiable manner and embedding them in learning frameworks [38, 39, 278, 249, 280, 155, 110, 171, 122, 155, 128, 172, 40, 248, 177, 210, 247, 36, 274, 191, 239, 169], which have demonstrated tremendous advantages.

Self-supervised Training with Differentiable Image Formation Models First and foremost, differentiable imaging systems solve the severe data problem in 3D reconstruction tasks. Take the image based 3D reconstruction task as the example. Supervised methods rely on images with 3D ground truth which is expensive to acquire [70, 260, 267]. In contrast, by incorporating

differentiable image formation models into deep learning pipeline, the networks can be trained in a self-supervised manner without any 3D supervision.

For example, given an input image, we could employ a neural network to predict its 3D properties. Next, we adopt the differentiable image formation model to convert the 3D properties back to an image and supervise the network via the loss between the rendered image and the input image. The inherent optical and geometric constraints inside the image formation models will help regularize and refine the network to predict 3D properties to be consistent with the input image. Therefore, the network can be trained totally on 2D images but predict reasonable 3D properties without any 3D supervision.

Recent works have demonstrated this benefit on a variety of 3D reconstruction tasks, including 3D object reconstruction [249, 122, 38, 278, 280, 155, 110], scene reconstruction [171, 242], material and light estimation [288, 39, 255], or even protein recovery [294]. It largely relieves the data-hungry problem in deep learning and can be further applied not only in synthetic domain but also to real data, make it possible to recover 3D properties from real images [291, 150].

Along this direction, we focus on differentiating the rendering pipeline and embedding it into deep learning to predict 3D properties from 2D images. We show detailed shapes, exquisite texture maps, accurate lighting and material parameters are all inferable from 2D photographs without any 3D supervision [38, 39, 291]. We talk about this part in Sec. 3, Sec. 4 and Sec. 5.

Tuning Imaging Parameters for Downstream Tasks Moreover, differentiable imaging systems also support to propagate information (*e.g.*, gradients) to imaging parameters, which allows to explore the best settings they should adopt under specific downstream tasks. It enables automatically tuning the imaging systems, discovering the most appropriate imaging parameters such as camera settings [177, 248] or system illuminations [172, 40], which generally requires expert knowledge in previous works.

As a frontier research topic, recently many works have appeared along this direction [172, 40, 248, 177, 210, 247, 36, 274, 191, 239, 169]. Existing works have spread to varying modalities, including the structured light system [172, 40], camera image signal processor (ISP) [36, 274, 191, 248, 177, 210], diffractive optical element (DOE) [239, 169, 247], or holography [32, 194]. Besides tuning the parameters of 3D imaging systems to boost 3D reconstruction tasks [172, 40], differentiable imaging systems are also adopted in a variety of downstream tasks like dark imaging [36], high dynamic range (HDR) imaging [191, 210, 239], holography display [32, 194], or even high level vision tasks like object detection and

segmentation [177], and have demonstrated significant performance boost.

Along this direction, we explore the optimal system illuminations in the structured light 3D imaging systems [172, 40]. In contrast to previous approaches which use manually designed illuminations, differentiable imaging systems allow us to tune system illuminations under user-specific imaging conditions. We show the optimized illuminations achieve substantial improvements in 3D triangulation accuracy. We discuss this part in Sec. 6 and Sec. 7.

Differentiation and Optimization Design Lastly, we briefly talk about the differentiation and optimization design for imaging systems. Since different imaging systems have distinct imaging formation models, there is no general way to differentiate all of them. Different imaging systems require specific differentiable designs. Roughly speaking, existing works can be categorized into two classes: differentiable approximation and hardware-in-the-loop. The former focuses on how to approximate the imaging process in a differentiable way. For example, designing the differentiable image formation models [171], approximating the gradients [157, 122, 154, 38, 128], or even using a neural network to simulate the imaging process [248, 36, 274, 191]. On the other side, people also propose to directly put the actual imaging systems in the optimization, which is called hardware-in-the-loop [194, 40, 177].

It is worth mentioning that, the goal of hardware-in-the-loop is to explore the best imaging settings under specific downstream tasks, which can also be formalized as an optimal parameters searching problem, *i.e.*, searching optimal parameters to minimize an objective function. Under such a scenario, it is not necessarily to have a differentiable system to apply gradient-based optimization methods. Instead, the imaging systems can be treated as a black box function and a variety of gradient-free optimization [130] can be adopted, such as Bayes optimization [223], or evolutionary algorithms [81]. For example, Mosleh *et al.* [177] optimize camera ISP parameters with Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithm and demonstrate the searched imaging parameters improves the downstream tasks.

However, differentiable imaging systems are more preferable, as the gradient-free optimization methods are hard to scale to high dimensionality, *e.g.*, larger than 1,000 dimensions [204]. As a result, they can hardly be applied to imaging systems with thousands of parameters. In contrast, differentiable imaging systems allow us to conduct optimization with large-scale parameter. For example, in [172, 40] we differentiate structured light systems to optimize the optimal system illuminations, which are composed of more than 10,000 parameters¹. More importantly, differentiable imaging systems is compatible with the deep learning pipeline and can therefore be embedded into neural networks to do joint

¹We optimize system illuminations at the scale of 4096 pixels * 5 patterns

optimization. The gradients can not only explore the best imaging parameters but also refine the network weights to better match the chosen hardware systems. Compared to gradient-free optimization methods which only tune the imaging systems themselves, we show joint optimization of both imaging systems and neural networks brings further performance improvement [40].

Till now we have covered a wide range of imaging systems. However, in this thesis we focus on the two of them: rendering pipelines and structured light systems. In the following sections we briefly introduce the background knowledge of the two systems and talk about the detailed differentiable reformulation in later chapters.

2.4 Differentiable Rendering

Developed in computer graphics, the rendering pipeline can be treated as a virtual camera to “take photos” of 3D models in the virtual world. The format of 3D models is typically triangle meshes, but it could also be in other types like voxel [278], point cloud [280], and signed distance function (SDF) [155]. In this thesis, we focus on triangle mesh based rendering and its differentiable reformulation [157, 122, 38, 154, 128].

The rendering pipeline is not designed to be differentiable in its original form. The technique of *differentiable rendering* aims to re-formulate it to become a differentiable process. Once done, it allows gradient computation from 2D images to various 3D model attributes, including the 3D geometry, texture, light and material. The gradients are very helpful and can be further utilized in a lot of applications like 3D optimization [152, 153], single view 3D reconstruction [122, 154, 38, 39], and intrinsic decomposition [160].

In the below sections, we first describe the general rendering equation (RE) [199], including its formula and explanation of each term. We also provide an overview of bidirectional reflectance distribution function (BRDF) [72]. We then talk about two rendering pipelines: *Monte Carlo ray-tracing* and *rasterization*, which are two different implementations of RE. We briefly introduce Monte Carlo ray-tracing based rendering pipeline and talk about the difficulties in differentiating through Monte Carlo ray-tracing. Next we describe rasterization based rendering pipeline and its differentiable design. We focus on rasterization since it is fast, computationally efficient and can be integrated within deep learning algorithms. Lastly, we talk about important applications and limitations in differentiable rendering.

2.4.1 Rendering Equation

Non-emissive Rendering Equation Let's start from a non-emissive rendering equation (RE) [115]. Assume \mathcal{M} is a 3D object in a virtual scene, the outgoing radiance L_o at any surface point $\mathbf{x} \in \mathcal{M}$ in the camera direction $\boldsymbol{\omega}_o$ is given by

$$L_o(\mathbf{x}, \boldsymbol{\omega}_o) = \int_{\mathcal{H}^2} f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) L_i(\mathbf{x}, \boldsymbol{\omega}_i) |\mathbf{n} \cdot \boldsymbol{\omega}_i| d\boldsymbol{\omega}_i, \quad (2.1)$$

where L_i is the incident radiance, f_r is the (spatially-varying) bidirectional reflectance distribution function (SV-BRDF), \mathbf{n} is the surface normal at \mathbf{x} . The domain of integration is the unit hemisphere \mathcal{H}^2 of incoming light directions $\boldsymbol{\omega}_i$. The BRDF characterizes the surface's response to illumination from different directions and is modulated by the cosine foreshortening term $|\mathbf{n} \cdot \boldsymbol{\omega}|$. Intuitively, Eq. (2.1) captures an energy balance and computes how much light is received and scattered at a shading point in a particular direction.

BRDF Overview While other terms are relatively straightforward to understand, we provide a brief explanation of bidirectional reflectance distribution functions (BRDFs). More details can be founded in [72]. Intuitively, BRDF evaluates the reflectance property of object surface, *i.e.*, given a light source, how much light is reflected at a specific direction. Following the similar notations in [72], BRDF is defined by the ratio from the outgoing radiance to the incoming irradiance:

$$\begin{aligned} f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) &= \frac{dL_o(\boldsymbol{\omega}_o)}{dE(\boldsymbol{\omega}_i)} \\ &= \frac{dL_o(\boldsymbol{\omega}_o)}{L_i(\boldsymbol{\omega}_i) |\mathbf{n} \cdot \boldsymbol{\omega}_i| d\boldsymbol{\omega}_i}, \end{aligned} \quad (2.2)$$

Here, $\boldsymbol{\omega}_i, \boldsymbol{\omega}_o$ are the incoming and outgoing directions². L_o is the surface leaving radiance (the reflected flux per unit area per unit solid angle, with the unit as $\frac{W}{m^2 \cdot sr}$). E is the irradiance (*e.g.*, the incident flux per unit area of the surface, with the unit as $\frac{W}{m^2}$). $dE(\boldsymbol{\omega}_i)$ denotes the small piece of E along $\boldsymbol{\omega}_i$ while $dL_o(\boldsymbol{\omega}_o)$ denotes the small piece of $L_o(\boldsymbol{\omega}_o)$ that comes from $dE(\boldsymbol{\omega}_i)$. Therefore, the unit of BRDF is $\frac{1}{sr}$. The irradiance $dE(\boldsymbol{\omega}_i)$ from direction $\boldsymbol{\omega}_i$ is equal to the incoming radiance $L_i(\boldsymbol{\omega}_i)$ times the cosine term $|\mathbf{n} \cdot \boldsymbol{\omega}_i|$ and the solid angle $d\boldsymbol{\omega}_i$, where: $dE = L_i(\boldsymbol{\omega}_i) |\mathbf{n} \cdot \boldsymbol{\omega}_i| d\boldsymbol{\omega}_i$. We illustrate the BRDF in Fig. 2.1.

The BRDF describes the reflectance characteristics of the surface material. In the simplified case, it is assumed that all the surface points share the same BRDF property (homogeneous BRDF, $f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$). However, in real world surface material is much more complicated and

²More precisely, they are solid angle in BRDF formulas

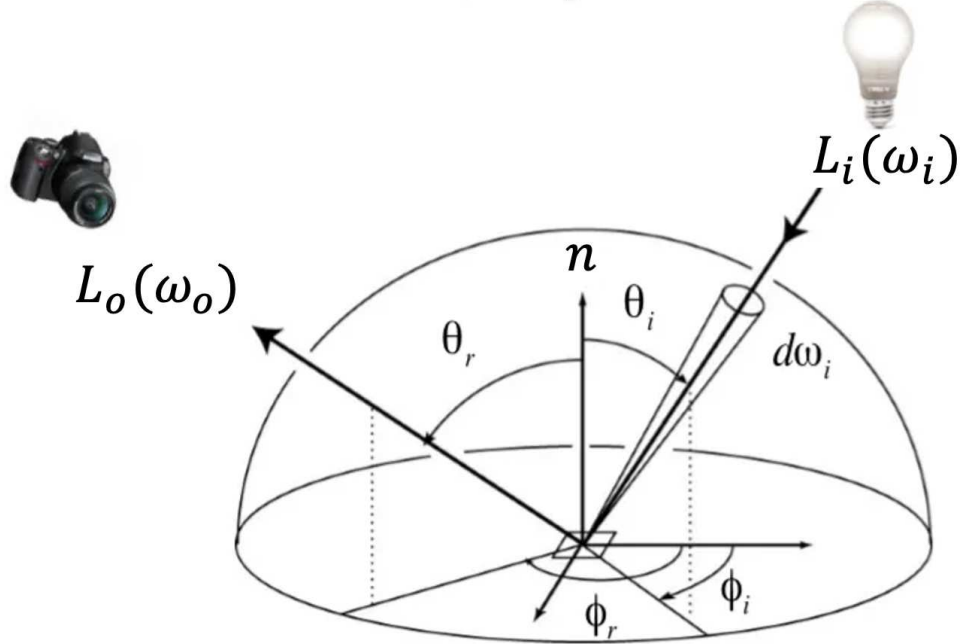


Figure 2.1: **BRDF Illustration.** The BRDF $f_r(\omega_i, \omega_o)$ from incoming direction ω_i to the outgoing direction ω_o is defined as $f_r(\omega_i, \omega_o) = \frac{dL_o(\omega_o)}{L_i(\omega_i)|\mathbf{n} \cdot \omega_i|d\omega_i}$.

might have spatial-varying properties, *e.g.*, the car body is metallic while the tires are diffuse. Therefore, to represent non-homogeneous surface material, we extend homogeneous BRDF by spatially-varying BRDF (SVBRDF $f_r(\mathbf{x}, \omega_i, \omega_o)$). As shown in Eq. (2.1), we introduce an extra parameter \mathbf{x} to take account the different BRDF properties in different locations of the surface. While BRDF only explains the surface material properties, it has limitations to model special materials like transparent objects, scattering media, and non-surface geometry like hair or fur. People also proposed advanced models to represent these special materials, which is beyond the scope of this thesis [108, 114, 196].

The BRDF should satisfy some physical rules. In summary, it should be non-negative, reciprocity and energy conservation.

1. Non-negative: BRDF is a non-negative function. For any incoming irradiance direction and outgoing radiance direction, $f_r(\omega_i, \omega_o) \geq 0$.
2. Reciprocity: Based on Helmholtz reciprocity principle, the light path is reversible. That's to say, $f_r(\omega_i, \omega_o) = f_r(\omega_o, \omega_i)$. However, in some special cases, *e.g.*, non-corresponding states of polarisation for incident and emerging fluxes, this rule might not hold [43].
3. Energy conservation: Overall, the reflected energy cannot exceed the incident energy. Namely, $L_o \leq E$. We can also re-write it in the form of the integral: $\int_{\mathcal{H}^2} f_r(\omega_i, \omega_o)|\mathbf{n} \cdot$

$$\omega_i | d\omega_i \leq 1.$$

However, in practice, BRDFs might violate the 3 rules, due to simplicity or computation efficiency [57, 200]. For example, in Unreal Engine 4 [57], the split-sum BRDF model violates the energy conservation rule and the total energy might be larger than 1.

There are many BRDF models [118, 200, 29, 57, 44] to represent different surface material properties. One of the most simplest BRDF model is Lambertian model, which is used to represent the purely diffuse surface. Here, the BRDF is modelled as a constant value: $f_r(\omega_i, \omega_o) = \frac{a}{\pi}$, where a is the surface albedo. It is largely used in many graphics applications due to its simplicity and high efficiency. However, as a constant, it cannot represent view-dependent effects. Later, Phong model [200] is proposed to represent non-Lambertian surface, where the BRDF is modeled as $f_r(\omega_i, \omega_o) = k_s(\omega_R \cdot \omega_o)^\alpha$. where, k_s and α are: specular reflection and shininess constants. ω_R are the direction of ω_i after being perfectly reflected. In Chapter 3 our rasterization based differentiable rendering discussed these two shading models.

Lambertian and Phong models, while computationally fast and widely used in graphics applications, can only represent diffuse and simple specular rendering effects. Also, they cannot model the realistic view-dependent surface reflectance effects, *e.g.*, the car body might reflect the surrounding environment. Therefore, more physically based BRDF models have been proposed to address the realistic reflectance effects. The Cook-Torrance model [44] is popular in computer graphics, as it distinguishes between metals and dielectrics and provides reasonable results and direction dependencies. The model is defined as $f_r(\omega_i, \omega_o) = \frac{a}{\pi} + \frac{D(\omega_h)F(\omega_o, \omega_h)G(\omega_i, \omega_o)}{4|\mathbf{n} \cdot \omega_i||\mathbf{n} \cdot \omega_o|}$, where the first part is Lambertian diffuse reflection while the second term is specular reflection. As for the specular component, it adopts microfacet surface model and uses the normal distribution function $D(\omega_h)$ and half vector ω_h to represent the surface property. F is the Fresnel term G is the shadowing term which states how many facets are occluded from different view angles. In Chapter 4 our one-bounce ray tracing based differentiable renderer utilizes this model.

2.4.2 Monte Carlo Ray-tracing based Rendering Pipeline

Monte Carlo Ray-tracing Estimating the RE typically requires Monte Carlo (MC) integration [199], which involves tracing rays from the camera into the scene. It also recursively considers rays bounced from one 3D point to another 3D point, until they arrive at light sources or reach the max recursion limit. The rendering process is also called physically based rendering (PBR) when incorporating with physical based BRDFs [118, 29, 57, 44]. In such a case, it satisfies physical rules and could represent realistic, global illuminations and

complex materials such as reflection, shadow, transparency, *etc.* While physically correct, this process is computationally expensive and does not generally yield a closed-form solution. MC estimators can exhibit high variance and may produce noisy pixel gradients at low sample counts, which may significantly impact performance and convergence. To keep the problem tractable, people make several simplifications Eq. (2.1) and develop *rasterization* to accelerate rendering process. In below sections, we briefly introduce how to differentiate the general rendering equation (Eq. (2.1)) through Monte Carlo ray-tracing based rendering pipeline. We discuss rasterization based rendering pipeline in Sec. 2.4.3.

Differentiable Rendering Equation To differentiate the rendering equation (Eq. (2.1)), let's start from the basic Leibniz integral rule [1]:

$$\begin{aligned} \frac{d}{d\pi} \int_{a(\pi)}^{b(\pi)} f(x, \pi) dx &= \int_{a(\pi)}^{b(\pi)} \frac{d}{d\pi} f(x, \pi) dx \\ &+ f(b(\pi), \pi) \frac{db(\pi)}{d\pi} - f(a(\pi), \pi) \frac{da(\pi)}{d\pi} \\ &+ \sum_i (f(c_i^-(\pi), \pi) - f(c_i^+(\pi), \pi)) \frac{dc_i(\pi)}{d\pi}, \end{aligned} \quad (2.3)$$

where given an integral $\int_{a(\pi)}^{b(\pi)} f(x, \pi) dx$, we want to compute its gradients $\frac{d}{d\pi} \int_{a(\pi)}^{b(\pi)} f(x, \pi) dx$ with regard to some parameters π . Leibniz integral rule tells us that in order to compute the derivatives of an integral, we need to consider three parts. The first term is to move the derivatives inside the integral, which is straightforward. However, in addition to that, the second term accounts for the changes in the integral limits while the last term considers all the discontinuities of integrand depend on π . Basically, the second and third terms explain the boundaries and discontinuities cases.

We can apply Leibniz's rule to RE with some generalization (Reynolds Transport Theorem) [284, 285]. To compute the derivatives of the image with regard to the 3D attributes π , we have:

$$\begin{aligned} \frac{dL_o(\mathbf{x}, \boldsymbol{\omega}_o)}{d\pi} &= \int_{\mathcal{H}^2} \frac{d}{d\pi} \{f_r(\mathbf{x}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) L_i(\mathbf{x}, \boldsymbol{\omega}_i) |\mathbf{n} \cdot \boldsymbol{\omega}_i|\} d\boldsymbol{\omega}_i \\ &+ \int_{\partial\mathcal{H}^2} g dl. \end{aligned} \quad (2.4)$$

Here, the first term is described as interior integral while the second term is called boundary integral. The interior integral accounts for the local parameters like BRDF parameters in each local point, while the boundary integral represents all the boundaries and discontinuities

in ray tracing. These cases generally happen in the edges of geometry or lighting. For example, considering two neighbour rays, one hits the foreground while the other hits background. The interior part can be addressed via auto-differentiation [104]. However, the boundary part requires special consideration.

Monte Carlo Ray-tracing based Differentiable Rendering Methods A branch of methods [144, 285, 284, 158, 16] have been proposed to address the boundary integral. Li *et al.* [144] propose the first Monte Carlo ray-tracing based differentiable rendering method. To address the discontinuous parts, it employs a novel edge sampling method to capture the changes at boundaries. Instead of silhouette sampling, Loubet *et al.* [158] propose to reparametrize the integrals to remove the discontinuities. Typically, the boundary integrals happen at the points that depend on the scene parameters. As such, the reparametrization helps remove the dependency and therefore cancels the boundary terms, though at the cost of introducing biased gradients. Bangaru *et al.* [16] propose to fast approximate the boundary terms by sampling the continuous area instead of the silhouette. While above methods address boundary terms in the rendering equation, Zhang *et al.* [285, 284] propose methods to estimate the derivatives in the path integral formulation, which is unbiased and computationally efficient as it does not need to find object silhouette edges explicitly.

On the other side, ray tracing typically involves millions of light path computation and several works [186, 185, 256, 105] focus on address the integral integrals efficiently. Nimier-David *et al.* [186] propose Mitsuba2, a famous graphics rendering library which adopts auto-differentiation design [104] and supports to compute derivatives during rendering process. On top of that, Nimier-David *et al.* [185] and Vicini *et al.* [256] employ adjoint method to reduce the heavy computation memory cost in auto-differentiation. Jakob *et al.* [105] propose a new compiler to further speed up the rendering and its derivative computation process.

Challenges in Differentiating through Monte Carlo Ray-tracing While great progress has been made, currently it is still hard to apply Monte Carlo ray-tracing based differentiable rendering methods in learning pipelines, especially for deep learning tasks. The challenges mainly happen in three aspects: computational efficiency, implementation difficulty, and optimization complexity.

1. Computational Efficiency: Although the above approaches can handle complex light transport effects, the Monte Carlo ray-tracing methods rely on heavy ray samples. As a result, ray tracing based differentiable rendering methods are computationally expensive. For example, rendering one image could take a few seconds to tens of

seconds. Moreover, the memory cost is also huge in ray tracing, which may take a few GB GPU memory. The slow rendering speed and heavy memory cost restrict the ray-tracing based differentiable rendering methods to be mainly adopted in 3D optimization tasks [184], which require only hundreds of iterations. However, they are still not suitable to be applied in general learning tasks that might take several hundred thousand iterations.

2. **Implementation Difficulty:** Monte Carlo ray-tracing based differentiable rendering pipelines are very difficult to implement. The implementation typically involves complex mathematical and coding efforts, *e.g.*, writing various CPP and CUDA programs. Although open-source libraries exist [186, 284], they are very heavy and have a specific data format. Currently, these libraries are still hard to be incorporated within popular deep learning pipelines like PyTorch or TensorFlow.
3. **Optimization Complexity:** Lastly, due to the complex light transport and the high variance in Monte Carlo sampling, ray-tracing based differentiable rendering tasks have fragile gradients which make the optimization process very difficult to converge. In the typically optimization tasks, *e.g.*, optimizing 3D parameters such that the rendered images look similar to the GT images, a good initialization is needed [184]. Recently Vicini *et al.* [257] explore combining ray-tracing based differentiable rendering with implicit representations to optimize from scratch, but optimizing with explicit meshes from scratch is still challenging.

2.4.3 Rasterization-based Rendering Pipeline

Ray tracing considers lighting propagation with multiple bounces, resulting in global illumination effects such as reflection, refraction, shadow, *etc.* However, it generally requires more memory usage and has a much slower rendering speed. In contrast, rasterization-based rendering pipeline simplifies RE to achieve real-time rendering speed. It only considers rays from the camera origin to the object surface without extra bounces, *i.e.*, the primary rays. Moreover, it mainly adopts simplified, local shading models, such as Lambertian, Spherical Harmonic, or Phong models [200, 68]. Instead of sampling rays to compute the integral, these local shading models approximate shading effects with some analytical illumination functions. As a result, it is much faster and suitable for simplified lighting and material conditions, but fails to represent advanced global illumination effects.

Rasterization-based rendering pipeline has been developed for more than 30 years and has already been implemented as standard libraries [5, 2]. The modern pipeline is composed of a series of functional steps called shaders. The 3 most important shaders are vertex shader,

rasterization, and fragment shader. In vertex shader, a 3D object is projected on the 2D image plane. Then, rasterization decides which pixel is covered by which triangle face (only considering rays from camera origin to object surface without extra bounces). Finally, fragment shader decides how to draw the colours for each pixel, based on selected local shading models. For more details, please refer to [11].

Rasterization-based Differentiable Rendering Similarly, the rasterization-based rendering pipeline is also not differentiable. Among the 3 shaders, vertex shader consists of matrix multiplication and fragment shader is composed of some arithmetic operations. Therefore, these two shaders are differentiable. The non-differentiability happens in rasterization, where the rendered edges of 3D models are not continuous. For example, let's consider a foreground pixel nearby edges. Since it is covered by a face, it should have gradients. Nevertheless, if we move outside a little to a background pixel which is not covered by any face, it won't be influenced by any face and thus should not have any gradients. Due to discrete pixels, we have discontinuity in the edges, which makes the rasterization non-differentiable. Therefore, the whole rendering pipeline is not differentiable, either.

As such, differentiable rendering works generally re-formulate the rasterization to make it differentiable in the edges. Back to 2014, Loper and Black [157] propose the first general differentiable rendering framework called OpenDR. The idea is very intuitive: the gradients from 2D images to 3D properties could be computed in a numerical way. For example, we will get a shifted image if the 3D models move towards x or y directions. Thus, the gradients to 3D vertex position can be computed by the image gradients (differences between the shifted image and the original image) dividing the shifting distance. OpenDR simply uses Sobel operator to compute image gradients and restricts the moving distance as 0.5 or 1 pixel. Later, Kato *et al.* [122] extend the movement to arbitrary distance in the image. They propose neural 3D mesh renderer (N3MR), which first combines the differentiable rendering with neural networks to predict 3D meshes from 2D images without 3D supervision.

The gradients in OpenDR and NMR are all computed numerically with limited precision. Next, Liu *et al.* [154] propose to turn the rasterization step from hard selection to soft merging. In the original form of rasterization, each pixel is only covered by one face while in their method, each pixel is influenced by all the faces with the weighted average. This method is called soft rasterizer (SoftRas). By this way, gradients are not restricted in the foreground pixels but happen to all the pixels in the image. The gradients can also be computed analytically since Liu *et al.* [154] choose the soft merging functions to be continuous.

All the previous works focus more on back propagating gradients to 3D vertex position,

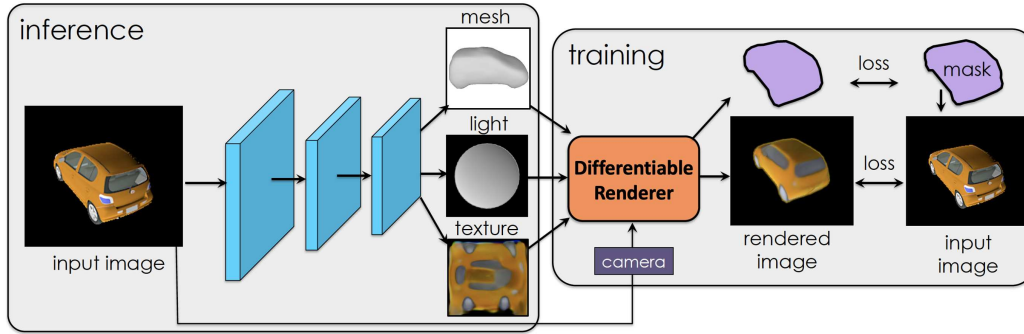


Figure 2.2: **Single-view 3D Object Reconstruction.** Given an input image, we could employ a neural network to predict geometry, light, and texture. During training we render the prediction back to a new image. We then compute the 2D image loss between input image and rendered prediction to train the network. Therefore, no 3D supervision is needed.

with limited support on other 3D properties like texture or lighting. In this thesis, we propose DIB-R [38], which is an interpolation-based differentiable renderer that supports back propagating gradients to all the common properties, including vertex position, vertex color, texture, texture coordinate, lighting, material, etc. The following-up works of DIB-R [291] extend to real imagery shows more vivid 3D reconstruction results, especially in texture prediction. Laine *et al.* [128] propose NVdiffrast, aiming for high performance and supporting more properties in the rendering pipeline, *e.g.* mipmap or depth peeling. Built on top of OpenGL, NVdiffrast achieves much faster rendering speed and supports high resolution meshes with millions of faces.

2.4.4 Applications & Limitations

Image based 3D Optimization Differentiable rendering computes gradients from 2D images to 3D attributes, which allows it to be embedded in the optimization tasks to refine 3D properties with 2D images. For example, given the initial geometry and a GT image, we could render the geometry to an image and evaluate the difference between the rendered image and the GT image. Next, we compute the gradients with regard to the geometry, optimizing the vertex position such that the rendered image is close to the GT. The optimization does not need any 3D supervision, which relieves the data requirement since 3D data is generally expensive to acquire. It can be used in many applications, including 3D design, games, and test-time optimization [160, 184].

Single-view 3D Object Reconstruction Differentiable rendering can further be embedded within deep learning to learn to infer 3D properties without any 3D training data. As shown in Fig. 2.2, given an image, we could employ a neural network to predict its 3D properties, *e.g.* geometry, light and texture. Next, we adopt differentiable rendering to render the properties back to an image and supervise via the loss between the rendered image and

input image. Similarly, the network can be trained totally on 2D images without any 3D supervision. Many differentiable rendering works have demonstrated this application, such as [122, 38, 154].

Limitations Rasterization-based differentiable rendering methods have been demonstrated in single-image 3D reconstruction tasks to recover reasonable shape and texture. However, one limitation is that they can not handle global illuminations. Global illuminations generally come from multiple bounces light propagation and produce advanced lighting effects such as reflection, refraction and transparency. Currently, all rasterization-based rendering methods adopt local shading and do not support secondary lighting effects. Consequently, when the input image has reflection, the recovered texture is always not clean and entangled with lighting effects.

On the other hand, while ray tracing based differentiable rendering methods supports global illumination, they are generally time-consuming and memory inefficient. Current pipelines [144, 186] are very heavy and can hardly be embedded in deep learning. That's to say, we cannot simply solve this problem by replacing rasterization to ray tracing in neural networks.

Since both rasterization and ray-tracing methods have pros and cons, in this thesis we further propose to develop a new hybrid differentiable rendering pipeline that takes advantage of both sides [39]. It is lightweight to be incorporated in networks, as well as supports higher order lighting effects to handle non-Lambertian surface in the images. We will discuss it in Sec. 4

2.5 Structured Light Systems

Structured light triangulation is a widely used 3D scanning technique. Typically, a structured light system is composed of a projector and a camera. As an active imaging system, the projector will send light signals to the environment while the camera will capture images with the active light signals. Such light signals encode spatial information. After capture, we decode the depth of the scene by matching the images and the projected light signals.

2.5.1 Structured Light Triangulation

Fig. 2.3 illustrates how structured light triangulation works. To acquire the depth of the object (e.g., the bunny), we first put it in front of the projector and camera. Next, we prepare multiple images for the projector to project. The projected images are the illuminations of the system, which are called *patterns* or *codes*. Specifically, assume we have K patterns to

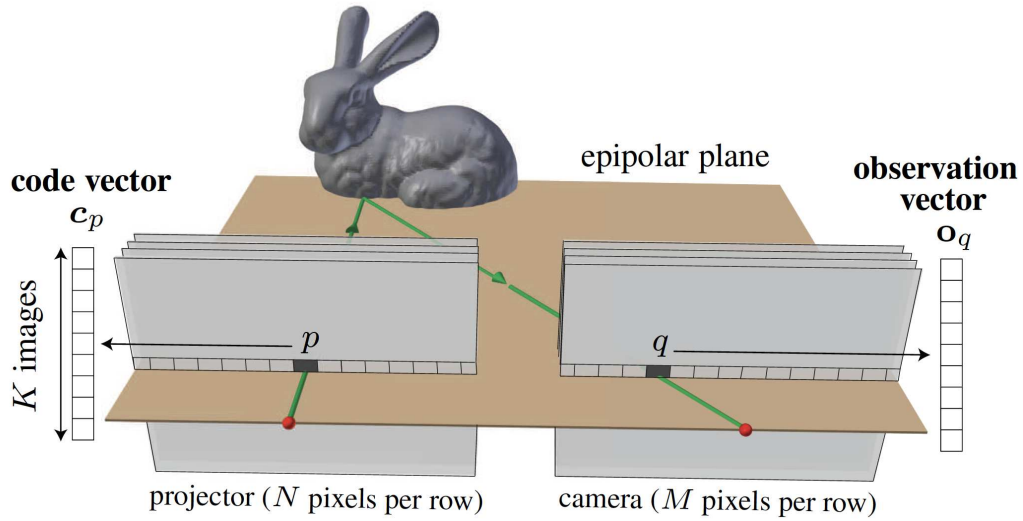


Figure 2.3: Structured Light System Overview.

project (Fig. 2.3 left). We will project each pattern sequentially and capture the corresponding images. K patterns generate K captured images (Fig. 2.3, right). Thus, for the pixel q in the camera, it has K captured intensities and forms a K dimensions feature, which we call observation vector \mathbf{p}_q .

Our goal is to find the correspondence of \mathbf{p}_q in the projector pixels, namely, which pixel of the projector directly illuminates q . In Fig. 2.3 it is pixel p in the projector that illuminates pixel q in the camera. If the correspondence is known, we can then infer the depth of q based on triangulation math. For example, we can shoot two rays from the projector and the camera origins. The two rays will intersect in the 3D space, which is a 3D surface point of the object. The depth can be further computed based on the intrinsic and extrinsic parameters of the projector and the camera [83].

Here, we restrict the path to be direct path. It means the 3D surface point will be directly illuminated by the projector pixel p and it also directly hits the camera pixel q . Only under direct path can we apply the triangulation model. Such a light path is called direct light. Sometimes, the pixel q might be illuminated by multiple pixels in the projector, due to several reasons like camera or projector defocus, inter-reflection geometry, or subsurface scattering. It then becomes a harder problem since we need to distinguish the direct path from other indirect paths [74, 182].

It is also worth mentioning that both the projector and the camera will project and capture 2D images. Nevertheless, it doesn't mean we have to match camera pixel q in the whole projector image. The epipolar geometry [83] constrains the possible searching space of q to be just one line of the projector image. For example, in Fig. 2.3, the projector origin, the

camera origin, and the pixel q define a plane which intersects with the projector plane and forms an *epipolar line*. If we only consider direct light, the possible correspondence pixels of q must lie in that epipolar line [83]. As a result, it reduces the search space from 2D to 1D.

Moreover, it means that instead of projecting 2D patterns, we can simply project 1D patterns since if we know the column correspondence, we can then infer the row correspondence based on the epipolar line. In this thesis we mainly focus on designing 1D patterns, which are the column patterns as they have different values for different columns but the same value for the same column. Designing 2D patterns would be a promising future work.

2.5.2 Structured Light Patterns

Structured light can be treated as a variant of stereo matching. In stereo matching, two or more cameras are adopted to capture images and we do matching between the images. Alternatively, in structured light triangulation, we have a projector and a camera to project patterns and capture images. Thus, we do matching between the patterns and images. Compared to captured images, patterns are totally controllable light sources. That's to say, we can project patterns with distinguishable structures to help matching algorithms.

Moreover, images captured in stereo matching are generally 1-channel gray or 3-channel RGB images, where each pixel contains at most 1-dimension or 3-dimension features. As a result, each pixel contains too little information to find the correspondence. To improve accuracy, people gather neighboring pixels together, conducting stereo matching between patches [17]. In contrast, the number of the projected patterns is not limited. We can project K patterns (for instance, $K > 3$) to collect more information for each pixel, which allows us to achieve dense pixel-size matching.

Clearly, patterns play a vital role in structured light triangulation. The projected patterns should contain rich structure information to maximize the correspondence matching accuracy. Furthermore, we want to project as few patterns as possible to reduce the capture time. Such two goals have derived a series of patterns designed in different ways, ranging from binary, discrete [77, 3] to continuous patterns [293, 74, 175].

Discrete patterns [77, 3] fill each pixel with binary values, but require more patterns to find the correspondence as each pixel contains only 1-bit information. Continuous patterns assign each pixel with continuous values. For example, Phase Shifting (PS) patterns and their variants [293, 74, 175] generally apply sinusoidal curves. We show two examples of continuous patterns in Fig. 2.4. However, previous works generally design patterns based on heuristic rules, which have limited or even no optimal guarantees. In this thesis we

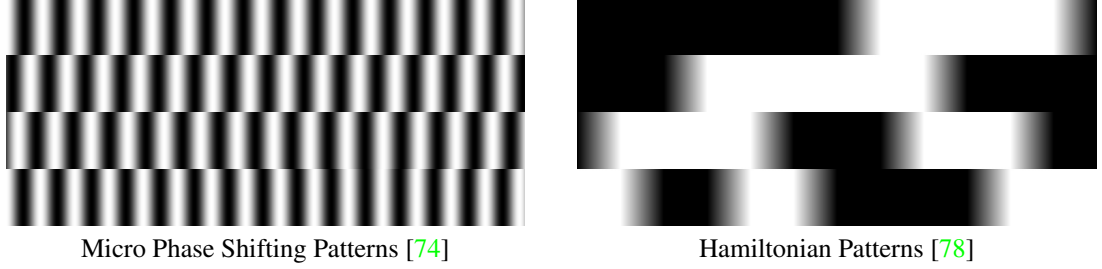


Figure 2.4: **Structured Light Pattern Examples.** We show two sets of manually designed structured light patterns ($K = 4$). Left is Micro Phase Shifting (MPS) Patterns [74], where the frequency $\omega = 16$. Right is Hamiltonian [78] designed by maximizing Hamiltonian path.

show patterns can be automatically optimized instead of manually designed. Moreover, such optimized patterns achieve much better correspondence accuracy than the hand-crafted methods under various scenes and devices

2.5.3 Structured Light Decoding Algorithms

Since the patterns encode spatial information, the correspondence matching can be treated as a decoding process, where we decode the correspondence for each pixel in the camera, figuring out which projector column it comes from. In previous works, people have designed a variety of patterns, and each has an accompanying decoding algorithm that fits the properties of the patterns. Different patterns result in distinct decoding algorithms. Below we introduce a very common phase shifting decoding algorithm for sinusoidal patterns [293].

Sinusoidal patterns are widely used in structured light triangulation. The patterns are composed of sinusoidal curves with frequency ω . Assume we project cosine curves: $y = \cos(2\pi\omega x)$. Here, each projector column has a specific phase. We compute the phase of projector column p by:

$$\phi(p) = 2\pi\omega \frac{p}{N} \quad (2.5)$$

Next, we uniformly shift the patterns K times. As a result, the value of k^{th} pattern in column p is defined by:

$$\mathbf{c}_p^k = \cos\left(\phi(p) + 2\pi \frac{k}{K}\right) \quad (2.6)$$

Assume a pixel q in the camera is corresponded with projector column p , we describe the

imaging formation model of pixel q as:

$$\begin{aligned}\mathbf{p}_q^k &= \mathbf{a}_q \mathbf{c}_p^k + \mathbf{b}_q \\ &= \mathbf{a}_q \cos(\phi(p) + \frac{2\pi k}{K}) + \mathbf{b}_q\end{aligned}\tag{2.7}$$

Here, \mathbf{a} and \mathbf{b} denote the albedo and the ambient light at pixel q . Eq. (2.7) says that if a camera pixel q is corresponded to the projector column p and we project K phase shifted patterns with frequency ω , the image intensity \mathbf{p}_q^k captured for the k^{th} pattern would be the albedo \mathbf{a}_q times the projector column intensity \mathbf{c}_p^k plus the ambient light \mathbf{b}_q .

The unknown variables in this equation are the albedo \mathbf{a}_q , ambient light \mathbf{b}_q , and the phase $\phi(p)$. To recover them, we can solve a linear system when $K > 3$:

$$\mathbf{p}_q = \mathbf{M}\mathbf{u}\tag{2.8}$$

In the linear system, \mathbf{p}_q are the observed values while vector $\mathbf{u} = [\mathbf{b}_q, \mathbf{a}_q \cos(\phi(p)), \mathbf{a}_q \sin(\phi(p))]^T$ is rearranged unknown variables. \mathbf{M} is a $K \times 3$ coefficients matrix and the k^{th} row of \mathbf{M} is $[1, \cos(2\pi \frac{k}{K}), -\sin(2\pi \frac{k}{K})]$. If \mathbf{u} is known, we can further compute \mathbf{a}_q , \mathbf{b}_q , and $\phi(p)$, respectively.

Note that when $\omega > 1$, we cannot recover unique column from phase since more than one column will have the same phase (Recall that $\phi(p) = 2\pi\omega \frac{p}{N}$). To resolve this ambiguity, we project patterns with multiple frequencies to decide column position.

Phase shifting has good performance in the idealized condition, e.g. projector and camera are in focus and there is no indirect light. However, when the camera pixel receives light from multiple projector columns due to defocus or global illuminations, the recovered phase will be different from the actual one. Besides phase shifting patterns, people also develop different decoding algorithms for different patterns. In this thesis, we provide a general, near-optimal decoding algorithm that can be applied for all the patterns, which we detail in Chapter 6.

Chapter 3

An Interpolation-based Differentiable Renderer

Many machine learning models operate on images, but ignore the fact that images are 2D projections formed by 3D geometry interacting with light, in a process called rendering. Enabling ML models to understand image formation might be key for generalization. However, due to an essential rasterization step involving discrete assignment operations, rendering pipelines are non-differentiable and thus largely inaccessible to gradient-based ML techniques. In this chapter, we present DIB-R, a differentiable rendering framework which allows gradients to be analytically computed for all pixels in an image. Key to our approach is to view foreground rasterization as a weighted interpolation of local properties and background rasterization as a distance-based aggregation of global geometry. Our approach allows for accurate optimization over vertex positions, colors, normals, light directions and texture coordinates through a variety of lighting models. We also apply DIB-R in single-view 3D object reconstruction tasks, which can be trained exclusively using 2D supervision. Compared to prior works, DIB-R demonstrates more faithful shape and realistic texture recovery, achieving new state-of-the-art results.

3.1 Introduction

3D visual perception contributes invaluable information when understanding and interacting with the real world. However, the raw sensory input to both human and machine visual processing streams are 2D projections (images), formed by the complex interactions of 3D geometry with light. Enabling machine learning models to understand the image formation process could facilitate disentanglement of geometry from the lighting effects, which is key

in achieving invariance and robustness.

The process of generating 2D images from 3D models is called *rendering*. Rendering is a well understood process in graphics with different algorithms developed over the years. These fall into different categories based on how the light transport is being modelled. *Rasterization*-based techniques are the fastest as they only geometrically project 3D objects onto the image plane but do not model more advanced lighting effects such as shadows and indirect light. *Ray tracing* algorithms are able to handle these effects via more complex optical simulation but require high computation and memory cost.

Making rendering process amenable to deep learning requires us to differentiate rendering pipelines. While ray tracing is too slow, most of learning based differentiable rendering works focus on rasterization-based renderers. Existing approaches typically compute approximate gradients [157, 122] which impacts performance. Furthermore, current differentiable rasterization methods fail to support differentiation with respect to many informative scene properties, such as textures and lighting, leading to low fidelity rendering, and less informative learning signals [122, 154].

In this chapter, we present DIB-R, an approach to differentiable rendering, which, by viewing rasterization as a combination of local interpolation and global aggregation, allows for the gradients of this process to be computed analytically over the entire image. When performing rasterization of a foreground pixel, similar to [62], we define its value as a weighted interpolation of the relevant vertex attributes of the foreground face which encloses it. To better capture shape and occlusion information in learning settings we define the rasterization of background pixels through a distance-based aggregation of global face information. With this definition the gradients of produced images can be passed back through a variety of vertex shaders, and computed with respect to all influencing vertex attributes such as positions, colors, texture, light; as well as camera positions. Our differentiable rasterization’s design further permits the inclusion of several well known lighting models.

More importantly, DIB-R can be embedded in neural networks to predict 3D object properties from a single view 2D image. Traditional supervised methods always require preparing 3D ground truth to supervise the training, which is hard to acquire. Following the “analysis by synthesis” principle, DIB-R is able to render 3D properties back into images and train the networks via minimizing the loss between the rendered image and the input image. Therefore, it supports 3D prediction *without any 3D supervision*, reducing the expensive data demand. We showcase DIB-R in challenging machine learning applications focusing on 3D shape and texture recovery, where we achieve both numerical and visual state-of-the

art results.

3.2 Related Work

OpenDR [157], the first in the series of differentiable rasterization-based renderers, approximates gradients with respect to pixel positions using first-order Taylor approximation, and uses automatic differentiation to back-propagate through the user-specified forward rendering program. In this approach, gradients are non-zero only in a small band around the edges of the mesh faces, which is bound to affect performance. [122] hand-designs an approximate gradient definition for the movement of faces across image pixels. The use of approximated gradients, and lack of full color information results in noisy 3D predictions, without concave surface features. To analytically compute gradients, Paparazzi [152] and its following-up [153], propose to back-propagate the image gradients to the face normals, and then pass them to vertex positions via chain rule. However, their gradient computation is limited to a particular lighting model (Spherical Harmonics), and the use of face normals further prevents their approach to be applied to smooth shading. [198] designs a C^∞ smooth differentiable renderer for estimating 3D geometry, while neglecting lighting and texture. [240] supports per-vertex color and approximates the gradient near boundary with blurring, which produces wired effects and can not cover the full image. [98] focus on rendering of point cloud and adopts a differentiable reprojection loss to constrain the distribution of predicted point clouds, which loses point connectivity and cannot handle texture and lighting.

Concurrent to our work, SoftRas [154] first introduces a probabilistic formulation of rasterization, where each pixel is softly assigned to *all* faces of the mesh. While inducing a higher computational cost, this clever trick allows gradients to be computed analytically. Its shading model incorporates vertex colors and supports texture and lighting theoretically. However, in [154] each pixel would be influenced by all the faces and thus might result into more blurry predictions. The key difference between [154] and ours is that, similarly to [62], we specify each foreground pixel to the most front face and compute analytic gradients of foreground pixels by viewing rasterization as interpolation of *local* mesh properties. This allows our rendering effect the same as OpenGL pipeline and naturally supports optimization with respect to all vertex attributes, and additionally enables the extension of our pipeline to a variety of different lighting models. We demonstrate our design achieves better shape and texture prediction. In contrast to [62], which also uses an interpolation-based approach, but providing no learning signals to the background pixels, our rasterization module allows for soft assignment of background pixels through an aggregation of global features, enabling better supervision signals in shape deformation, especially for the occluded geometry.

3.3 Differentiable Interpolation-based Renderer

In this section, we introduce our DIB-R. Treating foreground rasterization as an interpolation of vertex attributes allows realistic images to be produced, whose gradients can be fully back-propagated through all predicted vertex attributes, while defining background rasterization as an aggregation of global information during learning allows for better understanding of shape and occlusion.

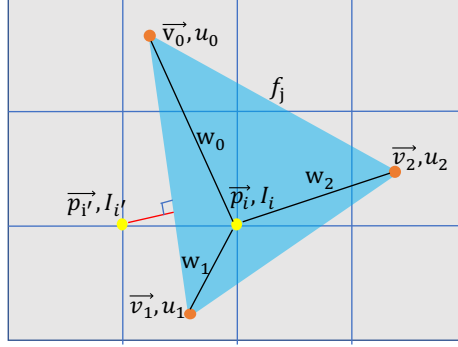
3.3.1 Rendering Pipeline

Many popular rendering APIs, such as OpenGL [5] and DirectX3D [2], decompose the process of rendering 3D scenes into a set of sequential user-defined programs, referred to as *shaders*. While there exist many different shader types, the vertex, rasterization, and fragment shaders the three most important steps for establishing a complete rendering pipeline. When rendering an image from a 3D polygon mesh, first, the vertex shader projects each 3D vertex in the scene onto the defined 2D image plane. Rasterization is then used to determine which pixels are covered and in what manner, by the primitives these vertices define. Finally, the fragment shader computes how each pixel is colored by the primitives which cover it.

The vertex and fragment shaders can easily be defined such that they are entirely differentiable. By projecting 3D points onto the 2D image plane by multiplying with the corresponding 3D model, view and projection matrices, the vertex shader operation is directly differentiable. In the fragment shader, pixel colors are decided by a combination of local properties including assigned vertex colors, textures, material properties, and lighting. While the processes through which this information are combined can vary with respect to the chosen rendering model, in most cases this can be accomplished through the application of fully differentiable arithmetic operations. All that remains for our rendering pipeline is the rasterization shader, which presents the main challenge, due to the inherently non-differentiable operations which it requires. In the following section we describe our method for rasterizing scenes such that the derivatives of this operation can be analytically determined.

3.3.2 Differentiable Rasterization

Consider first only the **foreground pixels** that are covered by one or more faces. Here, in contrast to standard rendering, where a pixel's value is assigned from the closest face that covers it, we treat foreground rasterization as an interpolation of vertex attributes[62]. For every foreground pixel we perform a z-buffering test [69], and assign it to the closest

Figure 3.1: **Illustration of Our Differentiable Rasterization.**

covering face. Each pixel is influenced exclusively by this face. Shown in Fig. 3.1, a pixel at position \vec{p}_i is covered by face f_j with three vertices $\vec{v}_0, \vec{v}_1, \vec{v}_2$, and each vertex has its own attributes: u_0, u_1, u_2 , respectively. \vec{p}_i and $\vec{v}_0, \vec{v}_1, \vec{v}_2$ are 2D coordinates on the image plane while u_0, u_1, u_2 are per-vertex features. We compute the value of this pixel, I_i , using barycentric interpolation of the face’s vertex attributes:

$$I_i = \omega_0 u_0 + \omega_1 u_1 + \omega_2 u_2 \quad , \quad (3.1)$$

where the barycentric weights ω_0, ω_1 and ω_2 are calculated over the vertex and pixel positions using a differentiable functions Ω (provided in Appendix A.1):

$$\begin{aligned} \omega_k &= \Omega_k(\vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{p}_i) \quad , \\ k &= 0, 1, 2 \quad . \end{aligned} \quad (3.2)$$

While barycentric interpolation has been widely used in OpenGL pipeline, we derive its differentiable reformulation. With this approach, it is easy to back-propagate gradients from a loss function \mathcal{L} , defined on the output image, through pixel value I_i to vertex attributes u_k and vertex positions \vec{v}_k via chain rule:

$$\begin{aligned} \frac{\partial I_i}{\partial u_k} &= \omega_k \quad , \\ \frac{\partial \mathcal{L}}{\partial u_k} &= \sum_{i=1}^N \frac{\partial \mathcal{L}}{\partial I_i} \frac{\partial I_i}{\partial u_k} \quad , \end{aligned} \quad (3.3)$$

$$\begin{aligned} \frac{\partial I_i}{\partial \vec{v}_k} &= \sum_{m=0}^2 \frac{\partial I_i}{\partial \omega_m} \frac{\partial \omega_m}{\partial \vec{v}_k} \quad , \\ \frac{\partial \mathcal{L}}{\partial \vec{v}_k} &= \sum_{i=1}^N \frac{\partial \mathcal{L}}{\partial I_i} \frac{\partial I_i}{\partial \vec{v}_k} \quad , \end{aligned} \quad (3.4)$$

where N is the number of pixels covered by the face.

Now consider pixels which no faces cover, which we refer to as **background pixels**. Notice that in the formulation above, the gradients from background pixels cannot back-propagate to any mesh attributes. However, the background pixels provide a strong constraint on the 3D shape, and thus the gradient from them provide a useful signal when learning geometry. Take, for example, pixel $\vec{p}_{i'}$ which lies outside of face f_j , in Fig 3.1. We want this pixel to still provide a useful learning signal. In addition, information from occluded faces are entirely ignored despite their potential future influence.

Inspired by the silhouette rasterization of [154], we define a distance-related probability $A_{i'}^j$, that softly assigns face f_j to pixel $\vec{p}_{i'}$ as:

$$A_{i'}^j = \exp\left(-\frac{d(\vec{p}_{i'}, f_j)}{\delta}\right) , \quad (3.5)$$

where $d(\vec{p}_{i'}, f_j) = \min_{v \in f_j} \|\vec{p}_{i'} - v\|_2^2$ is the distance function from pixel $\vec{p}_{i'}$ to face f_j in the projected 2D space. Similarly, our distance function also adopts square distance from the pixel $\vec{p}_{i'}$ to the closest point in the face f_j . The image coordinate of $\vec{p}_{i'}$ would be normalized into $[-1, 1]$ to avoid bias from different image resolutions. δ is a hyper-parameter that controls the smoothness of the probability. We then combine the probabilistic influence of all faces on a particular pixel in the following way:

$$A_{i'} = 1 - \prod_{j=1}^n (1 - A_{i'}^j) , \quad (3.6)$$

where n is the number of all the faces. The combination of all $A_{i'}$ into their respective pixel positions makes up our alpha channel prediction. With definition, any background pixel can pass its gradients back to positions of all the faces (including those ignored in the foreground pixels due to occlusion) with influence proportional to the distance between them in alpha channel. As all foreground pixels have a minimum distance to some face of 0, they must receive an alpha value of 1, and so gradients will only be passed back through the colour channels as defined above.

We show an rendering example in Fig. 3.2. Given a 3D mesh (Fig. 3.2, a), we render it into an RGBA image, where the RGB image is shown in Fig 3.2, b and the silhouette(alpha channel) is shown in Fig. 3.2, c.

The silhouette is derived from our probabilistic distance function (Eq. (3.5) & Eq. (3.6)). The hyper-parameter δ in Eq. (3.5) is used to control the smoothness of the distance probability, where higher δ (Fig. 3.2, d) makes the silhouette blurry and lower δ (Fig. 3.2, e) will make

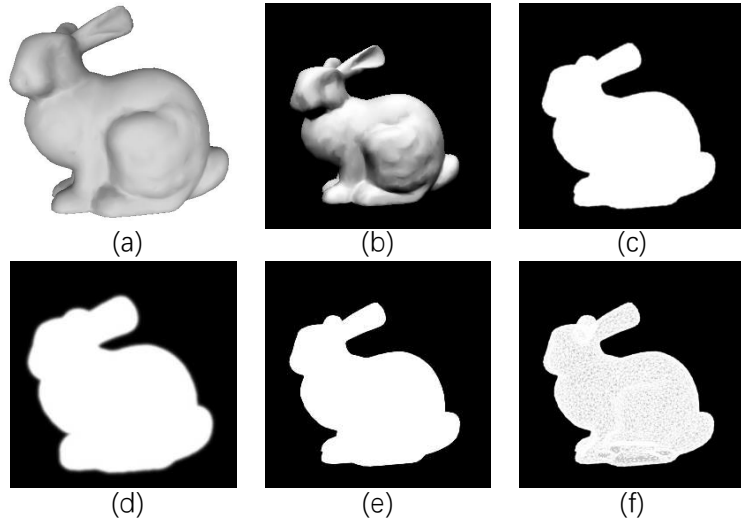


Figure 3.2: **Illustration of Our Rendering Pipeline.** (a) 3D mesh. (b) Rendered RGB image. (c) Rendered silhouette image with δ as $1.5e-4$. (d) Rendered silhouette with δ as $1.5e-3$. (e) Rendered silhouette with δ as $1.5e-5$. (f) Rendered silhouette from SoftRas[154]

the silhouette sharp. With this term we balance the information back propagated to the faces, and the sharpness of the predicted silhouette. In all of our experiments we set δ to $1.5e-4$ (Fig. 3.2, c), as a middle ground between the two.

We also compare our silhouette with SoftRas[154] (Fig. 3.2, f), where a sigmoid function is used to define their probabilistic distance function. However, the sigmoid function outputs 0.5 if a pixel lies on the edge of a face, leading many dim lines near face edges, which provides a poorer learning signal when comparing to ground truth silhouettes. Instead, our solution generates much smooth silhouettes, resulting in better shape prediction.

In summary, foreground pixels, which are covered by a specific face, back-propagate gradients through interpolation while background pixels, which are not covered by any face, softly back propagate gradients to all the faces based on distance. In this way, we can analytically determine the gradients for all aspects of our rasterization process and have achieved a fully differentiable rendering pipeline.

3.3.3 Rendering Models

In Equation (3.1), we define pixel values I_i by the interpolation of abstract vertex attributes u_0 , u_1 and u_2 . As our renderer expects a mesh input, vertex position is naturally one such attribute, but we also simultaneously support a large array of other vertex attributes. In the following section we outline the vertex attributes the rasterization can interpolate over and then back-propagate through, and the rendering models which the support of these attributes allows.

Basic Models Our DIB-R supports basic rendering models where we draw the image directly with either vertex colors or texture. To define the basic colours of the mesh we support vertex attributes as either vertex colour or uv coordinates over a learned or predefined texture map. Pixel values are determined through bi-linear interpolation of the vertex colours, or projected texture coordinates, respectively.

Lighting Models We also support 3 different local illumination models: Phong [200], Lambertian [129] and Spherical Harmonics [206], where the lighting effect is related to normal, light and eye directions.

To unify all the different lighting models, we decompose image color, I , into a combination of mesh color I_c and lighting factors I_d and I_s :

$$I = I_d I_c + I_s \quad . \quad (3.7)$$

I_c denotes the interpolated vertex colour or texture map values extracted directly from the vertex attributes without any lighting effect, I_d and I_s donate the lighting factors decided by specific lighting model chosen, where I_d will be merged with mesh colour and I_s is additional lighting effect that does not rely on I_c . We first interpolate light-related attributes such as normals, light or eye directions in rasterization, then apply different lighting models in our fragment shader.

Phong and Lambertian Models: In the Phong Model, image colour I is decided by vertex normals, light directions, eye directions and material properties through the following equations:

$$\begin{aligned} I_d &= k_d(\boldsymbol{\omega}_L \cdot \mathbf{n}) \\ I_s &= k_s(\boldsymbol{\omega}_R \cdot \boldsymbol{\omega}_V)^\alpha \quad , \end{aligned} \quad (3.8)$$

where, k_d , k_s and α are: diffuse reflection, specular reflection, and shininess constants. $\boldsymbol{\omega}_L$, \mathbf{n} , $\boldsymbol{\omega}_V$ and $\boldsymbol{\omega}_R$ are directions of light, normal, eye and reflectance, respectively, which are all interpolated vertex attributes. This results in the following definition for image colour under the Phong model:

$$I_{Phong} = I_c k_d(\boldsymbol{\omega}_L \cdot \mathbf{n}) + k_s(\boldsymbol{\omega}_R \cdot \boldsymbol{\omega}_V)^\alpha \quad . \quad (3.9)$$

As a slight simplification of full Phong shading we do not adopt ambient light and set light colour at a constant value of 1. The Lambertian model can be viewed as a further simplification of the Phong Model, where we only consider diffuse reflection, and set I_s as zero:

$$I_{Lambertian} = I_c k_d(\boldsymbol{\omega}_L \cdot \mathbf{n}) \quad . \quad (3.10)$$

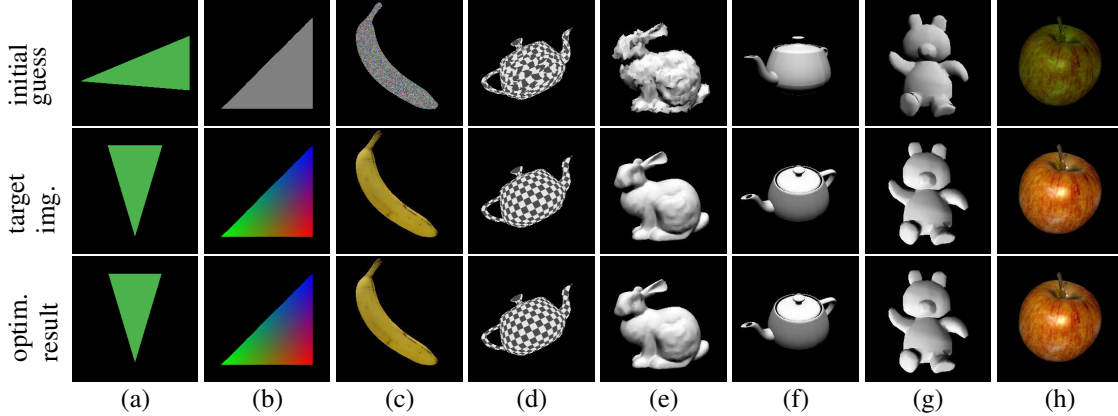


Figure 3.3: **DIB-R Sanity Check.** We perform a sanity check for DIB-R by running optimization over several mesh attributes. We optimize w.r.t. different attributes in different rendering models: (a,b) vertex position and color in the vertex color rendering model, (c,d) texture and texture coordinates in the texture rendering model, (e,f) vertex and camera position in Lambertian model, (g) lighting in the Spherical Harmonic model, (h) material in the Phong model.

Vertex attribute	Vertex Shader	Rasterization	Fragment attribute	Fragment Shader
Vertices				
Vertex Colors	Camera	Differentiable- Rasterizer	Pixel Colors	Color Model
Tex Coords	Model Matrix		Pixel Tex Coords	Texture Model
Vertex Normals	View Matrix		Pixel Normals	Lambertian Model
Light Directions	Projection Matrix		Pixel Light Directions	Spherical harmonic Model
Eye Directions			Pixel Eye Directions	Phong Model

Table 3.1: Differentiable vertex attributes and rendering models supported by DIB-R. With our method, we can differentiate most common attributes (Column 1 & 4, Vertex attributes and Fragment attributes) and apply it into multiple rendering models (Column 5, Rendering Models).

Spherical Harmonic Model: Here, I_d is determined by normals while I_s is set to 0:

$$I_{SH} = I_c \sum_{l=0}^{n-1} \sum_{m=-l}^l w_l^m Y_l^m(\vec{N}) , \quad (3.11)$$

where Y_m^l is an orthonormal basis for spherical functions analogous to a Fourier series where l is the frequency and w_l^m is the corresponding coefficient for the specific basis. To be specific, we set l as 3 and thus predict 9 coefficients in total. By adjusting different w_l^m , different lighting effects are simulated. For more details please refer to [206], Section 2.

Supported Mesh Attributes Our method simultaneously support a large array of vertex attributes. In Table 3.1 we highlight the various rendering settings and scene attributes which our rendering pipeline supports.

3.3.4 Optimization

The design of our differentiable renderer allows for optimization over all defined vertex attributes and a variety of rendering models, which we perform a sanity check for in Fig. 3.3.

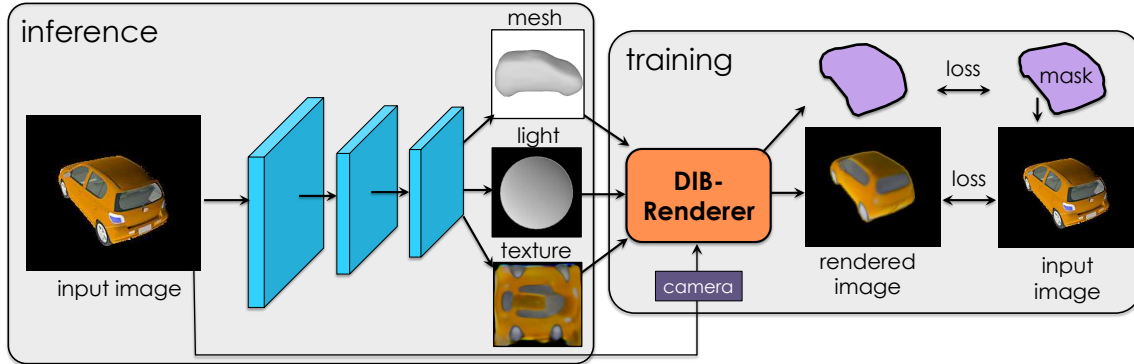


Figure 3.4: **Full Architecture of Our Approach.** Given an input image, we predict geometry, light, and texture. During training we render the prediction with a known camera. We use 2D image loss between input image and rendered prediction to train our prediction networks. Note that the prediction can vary in different rendering models, e.g. texture can be vertex color or a texture map while the lighting can be Lambertian, Phong or Spherical Harmonics.

Here, we optimize the L_1 loss between the target images (second row) and predicted rendered images. Among rasterization-based renderers we are the first to support optimization of all vertex attributes.

3.4 Single-view 3D Object Reconstruction

We demonstrate the effectiveness of our framework through challenging ML applications: 3D shape and texture recovery from a single view 2D image. We validate on different rendering settings, which we detail below.

3.4.1 Basic Model: Predicting Geometry and Color

We first apply our approach to the task of predicting a colored 3D mesh from a single image using only 2D supervision. Taking as input a single RGBA image, with RGB values I and alpha values M (also called silhouette or visibility map), a Convolutional Neural Network F , parameterized by learnable weights ϑ , predicts per-vertex position in a mesh S with a specified topology (sphere in our case), together with per vertex color value C :

$$\{S, C\} = F(I, M; \vartheta) \quad (3.12)$$

Network Structure Specifically, We adopt the encoder-decoder framework as in [122, 154]. The encoder contains 3 convolutional layers and 3 linear layers, each convolutional layer has 64/128/256 channels with kernel size 5 and stride length of 2. The output feature map of the convolutional layers is flattened to 16384- d vector and fed to the first linear layer. Each linear layer has 1024 neurons. We add BatchNorm [99] to all the convolutional layers and

the first and second linear layers. ReLU [127] activation function is used after each layer. We have two decoders, one for vertex position, the other for vertex color. Each decoder has three fully-connected layers, each layer has 1024/2048/1926 neurons, respectively. We directly predict position and color for each vertex. The template sphere has 632 vertices and 1280 faces. We train every model until converge for around 2 days in V100 GPU.

Loss Function We then use a renderer R (specified by shader functions, in the basic model we adopt vertex color shader.) to render the mesh (S, C) predicted by $F(I, M; \boldsymbol{\vartheta})$ to a 2D silhouette \tilde{M} and the colored image \tilde{I} :

$$\{\tilde{M}, \tilde{I}\} = R(S, C) . \quad (3.13)$$

When training this system we separate our losses with respect to the silhouette prediction, \tilde{M} , and the color prediction, \tilde{I} . We use an Intersection-Over-Union (IOU) loss for the silhouette prediction [122]:

$$\mathcal{L}_{IOU}(\boldsymbol{\vartheta}) = 1 - \frac{\|M \odot \tilde{M}\|_1}{\|M + \tilde{M} - M \odot \tilde{M}\|_1} , \quad (3.14)$$

where \odot denotes element-wise product. Note that $\{S, C\} = F(I; \boldsymbol{\vartheta})$, $\{\tilde{I}, \tilde{M}\} = R(\boldsymbol{v}, S, C)$ depend on the network’s parameters $\boldsymbol{\vartheta}$ and render on camera pose \boldsymbol{v} via our DIB-R. We further use an L_1 loss for the colored image:

$$\mathcal{L}_{col}(\boldsymbol{\vartheta}) = \|I - \tilde{I}\|_1 . \quad (3.15)$$

We also regularize the mesh prediction with a smoothness loss [122, 154], \mathcal{L}_{sm} , and a Laplacian loss [154, 232, 260], \mathcal{L}_{lap} , which penalize the difference in normals for neighboring faces and the change in relative positions of neighboring vertices, respectively. Let E be the set of all edges, and θ_i be the angle between two neighbour faces, which share the edge e_i ; the smoothness loss is then:

$$\mathcal{L}_{sm} = \sum_{e_i \in E} (\cos(\theta_i) + 1)^2, \quad (3.16)$$

which regularizes neighboring faces to have the similar normal directions, encouraging a smoother predicted mesh. Our Laplacian loss design follows [260]. For each vertex v , let

$\mathcal{N}(v)$ be the neighbour vertices of v , then the Laplacian loss is defined as:

$$\mathcal{L}_{lap} = (\delta_v - \frac{1}{|\mathcal{N}(v)|} \sum_{\tilde{v} \in \mathcal{N}(v)} \delta_{\tilde{v}})^2, \quad (3.17)$$

where δ_v is the predicted movement of vertex v . The Laplacian loss forces neighborhoods to move consistently [260]. The final loss function is then a weighted sum of these four losses:

$$\mathcal{L}_1 = \mathcal{L}_{IOU} + \lambda_{col} \mathcal{L}_{col} + \lambda_{sm} \mathcal{L}_{sm} + \lambda_{lap} \mathcal{L}_{lap} . \quad (3.18)$$

Multi-view Consistency When rendering our predicted mesh, we not only use the ground truth camera positions and compare against the original image, but also render from a random second view and compare against the ground truth renderings from this new view [122, 154]. This multi-view loss ensures that the network does not only concentrate on the mesh properties in the known perspective.

Mathematically, let us assume we have two images I_1 and I_2 from two different poses (cameras noted as \mathbf{v}_1 and \mathbf{v}_2) of the same car. We can then predict shape and vertex color for each image, noted as S_1, C_1 and S_2, C_2 . The multiview consistency image loss is given by:

$$\begin{aligned} \mathcal{L}_{col}^{MV} = & \|I_1 - R(\mathbf{v}_1, S_1, C_1)\|_1 + \\ & \|I_2 - R(\mathbf{v}_2, S_1, C_1)\|_1 + \\ & \|I_1 - R(\mathbf{v}_1, S_2, C_2)\|_1 + \\ & \|I_2 - R(\mathbf{v}_2, S_2, C_2)\|_1 . \end{aligned} \quad (3.19)$$

We predict shape and vertex color in canonical views. The silhouette loss is applied in the same way. Besides the loss in image space, there are no more consistency regulations, i.e., we did not add any loss term to force S_1, S_2 or C_1, C_2 to be the same. Moreover, while more views would provide more constraints, we find that two views are enough. We thus use two views in all our experiments.

3.4.2 Lighting Models: Predicting Geometry, Texture, and Light

We next apply our method to an extension of the previous task, where a texture map T is predicted instead of vertex colors, and lighting information L is regressed to produce more realistic predictions. Our neural network F is modified to predict vertex positions S , a texture map T , and various lighting information L , depending on the lighting model used. Our full learning pipeline is shown in Fig. 3.4. Note that, our loss is mainly defined in the image space. Therefore, we are able to predict any 3D properties in an unsupervised way, as

long as it is supported by DIB-R. We apply the same losses as in the previous section.

Network Structure We adopt a UNet [211] model for texture prediction, the network architecture is similar to [116], except we add Batch Normalization to every convolutional layer and we use skip connections. We also replace all ReLu layers in UNet [211] to be Leaky Relu [275]. We exploit a shallow fully connected encoder for lighting parameters prediction. The encoder contains 1 linear input layer and 2 1D-residual linear layers [85] and 1 linear output layer. Output image feature for first three linear layers contains 1024/1024/1024 channels. For Phong model, the final linear layer outputs 4 lighting parameters where we try to learn the light direction together with the material shininess constant. For Spherical Harmonic model, the final linear layer outputs 9 coefficients of Spherical Harmonic basis.

Photo-realism Enhancement To increase the photo-realism of our predictions, we also leverage an adversarial framework [66, 173]. This is accomplished by training a discriminator network, $D(\cdot; \phi)$, to differentiate between real images, I , and rendered mesh predictions, \tilde{I} , while our prediction network, F , simultaneously learns to make these predictions. We adopt the W-GAN [15, 73] loss formulation¹:

$$\begin{aligned}\mathcal{L}_{adv}(\vartheta, \phi) &= \mathbb{E}_{\mathbb{I}} \left[D(I; \phi) - D(\tilde{I}; \phi) \right] \\ \mathcal{L}_{gp}(\phi) &= \mathbb{E}_{\mathbb{I}} \left[(\|\nabla_{\tilde{I}} D(\tilde{I}; \phi)\|_2 - 1)^2 \right] .\end{aligned}\quad (3.20)$$

Similar to [101, 262, 279], we additionally use a perceptual loss and discriminator feature matching loss to make training more stable:

$$\mathcal{L}_{per}(\vartheta, \phi) = \mathbb{E}_{\mathbb{I}} \left[\sum_{i=1}^{M_V} \frac{1}{N_i^V} \|V^i(I) - V^i(\tilde{I})\|_1 + \sum_{i=1}^{M_D} \frac{1}{N_i^D} \|D^i(I; \phi) - D^i(\tilde{I}; \phi)\|_1 \right] , \quad (3.21)$$

where V^i denotes the i -th layer of a pre-trained VGG network with N_i^V elements, D^i denotes the i -th layer in the discriminator D with N_i^D elements. The numbers of layers in network V and D are M_V and M_D , respectively. As for the network structure of the discriminator, we inherit basic discriminator architecture from DC-GAN [205]. We use Instance normalization [250], and Leaky-ReLU [163]. Our full objective function for this task is then:

$$\vartheta^*, \phi^* = \underset{\vartheta}{\operatorname{argmin}} \left(\underset{\phi}{\operatorname{argmax}} (\lambda_{adv} \mathcal{L}_{adv} - \lambda_{gp} \mathcal{L}_{gp}) + \lambda_{per} \mathcal{L}_{per} + \mathcal{L}_1 \right) . \quad (3.22)$$

¹We denote $\mathbb{E}_{\mathbb{I}} \triangleq \mathbb{E}_{I \sim \mathcal{P}(I)}$

3.5 Experiments

Dataset As in [122, 154, 260], our dataset comprises 13 object categories from the ShapeNet dataset [33]. We use the same split of objects into our training and test set as [260]. We render each model from 24 different views to create our dataset of RGBA images used for 2D supervision. To demonstrate the multiple rendering models which DIB-R supports, we render each image with 4 different rendering models: 1) basic model without lighting effects, 2) with Lambertian reflectance, 3) with Phong shading, and 4) with Spherical Harmonics. We provide details are in each sections below.

3.5.1 Basic Model: Predicting Geometry and Color

Experimental Settings In our experiments, we set $\lambda_{col} = 1$, $\lambda_{sm} = 0.001$, and $\lambda_{lap} = 0.01$. We train one network on all 13 categories. The network is optimized using the Adam optimizer [124], with $\alpha = 0.0001$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. The batch size is 64, and the dimension of input image is 64×64 . We compare our method with the two most related differentiable renderers, N3MR [122] and SoftRas [154], using the same network configurations, training data split and hyperparameters. For quantitative comparison, we first voxelize the predicted mesh into 32^3 volume using a standard voxelization tool `binvox`² provided by ShapeNet [33] and then evaluate using 3D IOU, a standard metric in 3D reconstruction.

We additionally measure the F-score following [243] between the predicted mesh and ground truth mesh. Specifically, for F-score, we first uniformly sample 2500 points from the predicted mesh and ground truth mesh. For every predicted point, if the minimum distance from it to any ground truth point is less than a threshold, we treat it as a true positive, otherwise, it is false positive. This allows us to compute a precision score, p . For every ground truth point, if the minimum distance from it to every predicted point is less than a threshold, we treat this as a true positive, otherwise, it is a false negative, and so we can compute a recall score r . The F-score is then computed via: $2 * (p * r) / (p + r)$. We set the threshold to 0.02 and normalize the shape to be in $[-0.45, 0.45]$ in our experiments. The tolerance for F-score is set to 0.02, where the 3D model is normalized with radius 0.45.

Dataset Details As mentioned before, we use 13 object categories from the ShapeNet dataset³ [33], version 1. The training set contains 35007 objects, and test set contains 8752 different objects. For each object, we first normalize the object such that the center of the object is in the origin and all the vertexes lie in range $[-0.45, 0.45]$, we then render

²<http://www.patrickmin.com/binvox/>

³www.shapenet.org

Category	Airplane	Bench	Dresser	Car	Chair	Display	Lamp	Speaker	Rifle	Sofa	Table	Phone	Vessel	Mean
N3MR [122]	58.5/80.6	45.7/55.3	74.1/46.3	71.3/53.3	41.4/39.1	55.5/43.8	36.7/46.4	67.4/35.0	55.7/83.6	60.2/39.2	39.1/46.9	76.2/74.2	59.4/66.9	57.0/54.7
SoftR. [154]	58.4/71.9	44.9/49.9	73.6/41.5	77.1/51.1	49.7/40.8	54.7/41.7	39.1/39.1	68.4/29.8	62.0/82.8	63.6/39.3	45.3/37.1	75.5/68.6	58.9/55.4	59.3/49.9
Ours	57.0/75.7	49.8/55.6	76.3/52.2	78.8/53.6	52.7/44.7	58.8/46.4	40.3/45.9	72.6/38.8	56.1/82.0	67.7/43.1	50.8/51.5	74.3/73.3	60.9/63.2	61.2/55.8

Table 3.2: Results on single image 3D object prediction reported with 3D IOU (%) / F-score (%).

Figure 3.5: **Qualitative Results on Single-view 3D Object Reconstruction.** First and fifth columns are the ground-truth image, the second and sixth columns are the prediction from our model, the third and seventh columns are results from SoftRas [154], the rest two columns are results from N3MR [122].

each object using Blender⁴ with 24 different camera views. The camera views are equally distributed in a 360 degree ring around each object. The lighting direction in this dataset is set to uniform light. This results in 840k images for training and 210k images for testing.

Results Table 3.2 provides an evaluation. Our DIB-R significantly outperforms other methods, in almost all categories on both metrics. We surpass SoftRas/N3MR with 1.92/4.23 points and 5.98/1.23 points in terms of 3D IOU and F-score, respectively. As the only difference in this experiment is the renderer, the quantitative results demonstrate the superior performance of our method. Qualitative examples are shown in Fig 3.5. Our DIB-R faithfully reconstructs both the fine-detailed color and the geometry of the 3D shape, compared to SoftRas and N3MR. We provide more results in Fig. 3.6. We further provide more results from different view angles on Figure 3.7, demonstrating the high fidelity our model achieves.

3.5.2 Lighting Models: Predicting Geometry, Texture and Light

Our model supports various lighting models, ranging from Lambertian to Phong and Spherical Harmonic (SH). We first validate the Lambertian model and qualitatively compare with N3MR [122] by measuring the reconstruction error on rendered images under identical settings.

Experimental settings As mentioned before, we adopt a UNet [211] architecture to predict texture maps. Since we deform a mesh from a sphere template, similar to [116], we use 2D spherical coordinates as the UV coordinates. The dimension of the input image and predicted texture is 256×256 . We use a 6-layer ResNet [85] architecture to regress the XYZ directions of light at each vertex from the features at the bottleneck layer of UNet network.

⁴<https://www.blender.org/>

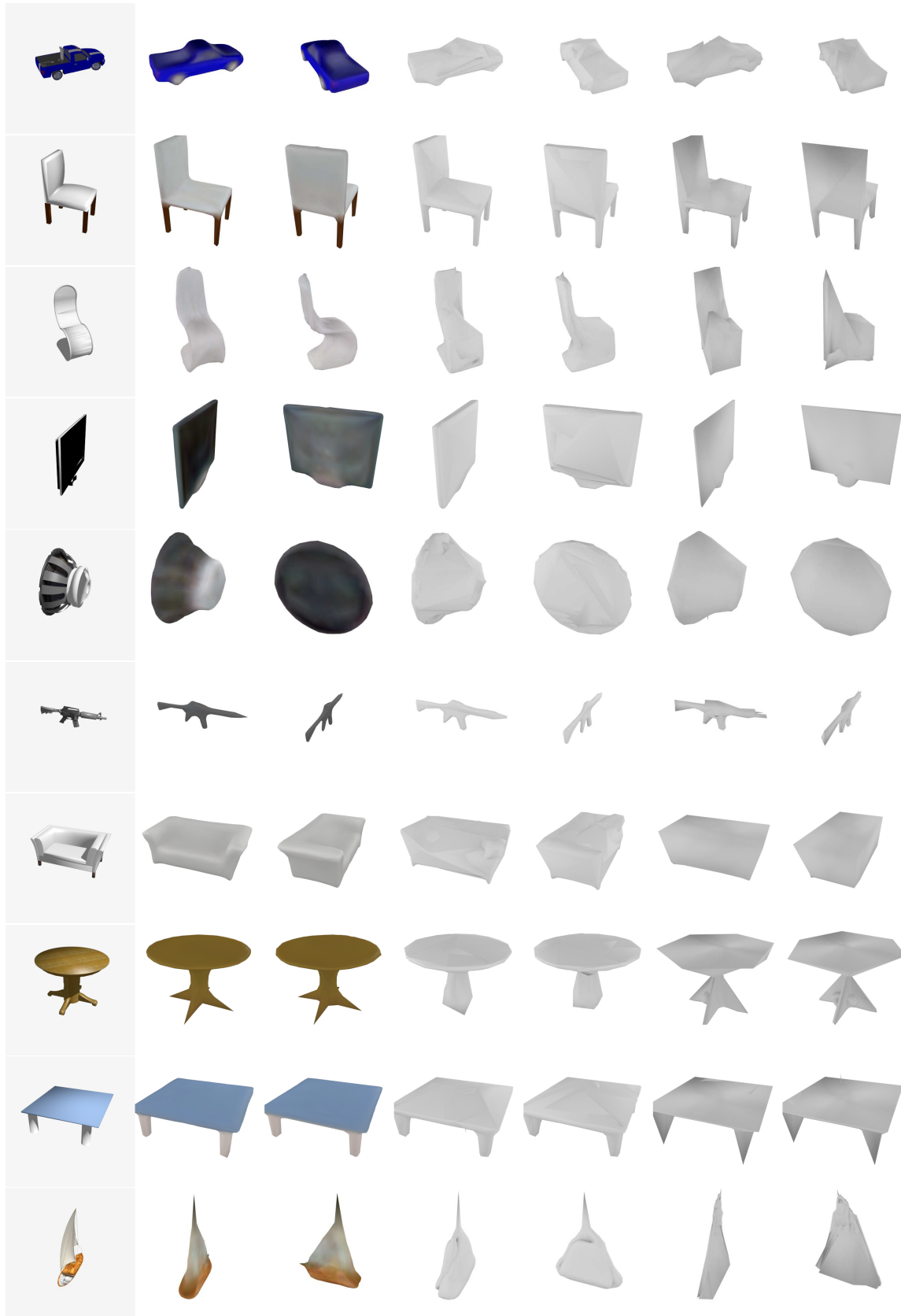


Figure 3.6: **Qualitative Comparisons on Single-view 3D Object Reconstruction.** First column is the ground-truth image, the second and third columns are the prediction from our model, the fourth and fifth columns are results from SoftRas [154], the last two columns are results from N3MR [122].

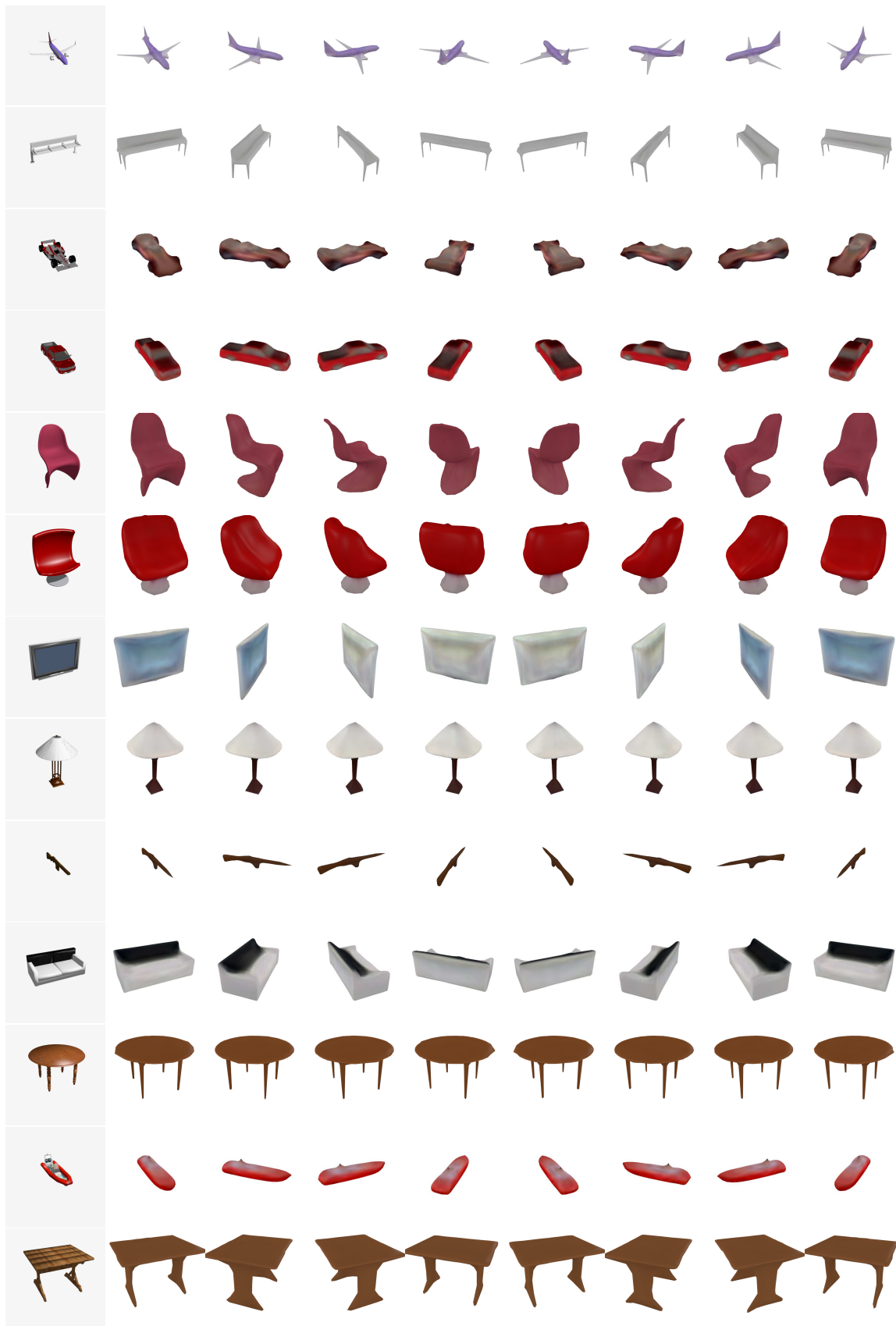


Figure 3.7: **Qualitative Results on Single-view 3D Object Reconstruction from Different Views.** The first column is the ground truth, the rest columns are reconstructed object rendered in different views.

Models	Texture	Lighting	Text. + Light
N3MR [122]	0.03640	23.5585	0.02208
Ours	0.02179	9.7096	0.01362

Table 3.3: Results for texture and light prediction. Texture/Texture+Light shows L_1 loss on the rendered image for texture/texture+lighting. Lighting shows the angle between predicted lighting and GT lighting. Lower is better.



Figure 3.8: **Qualitative Examples for 3D Shape, Texture, and Light Prediction.** Col. 1-3: 1) GT rendered image with texture+light, 2) texture only rendered image, 3) light map. Col 4-6: our predictions. Col: 7-9: N3MR [122]

We use the same learning rate and loss rate as basic model. In the following sections, we only perform experiments on the car class, which has more diverse texture.

Dataset Details To learn diverse texture maps, we select to learn with the car category due to its large diversity of texture. In the experiments where we compare with N3MR, we choose Lambertian reflectance model as our rendering model since N3MR only supports such one lighting model. Instead of Blender, We choose to use OpenGL⁵ to render data since we could easily turn on or turn off light effect in OpenGL shaders to separate the texture and light. We render car models into images both with and without light effect in Lambertian model and randomly sample the lighting direction over a quarter sphere where we constrain the light illuminates the upper part of the objects.

Results We provide results in Table 3.3 and Fig 3.8. As ShapeNet does not provide ground-truth UV texture, we compute the L_1 difference on the rendered image using the predicted texture/texture+lighting and the GT image. Compared to N3MR [122], we achieve significantly better results both quantitatively and qualitatively. We obtain about 40% lower L_1 difference on texture and 60% smaller angle difference on lighting direction than N3MR. We also obtain significantly better visual results, in terms of the shape, texture and lighting. We provide more results in Fig. 3.9.

3.5.3 Texture and Lighting Results with Adversarial Loss

We now evaluate the effect of adding the adversarial loss to the previous experiment. We demonstrate DIB-R with Phong and Spherical Harmonic lighting models. For Phong model we keep diffuse and specular reflectance constant as 1 and 0.4 respectively and predict lighting direction together with shininess constant α while for Spherical Harmonic model we predict 9 coefficients.

⁵<https://www.opengl.org/>

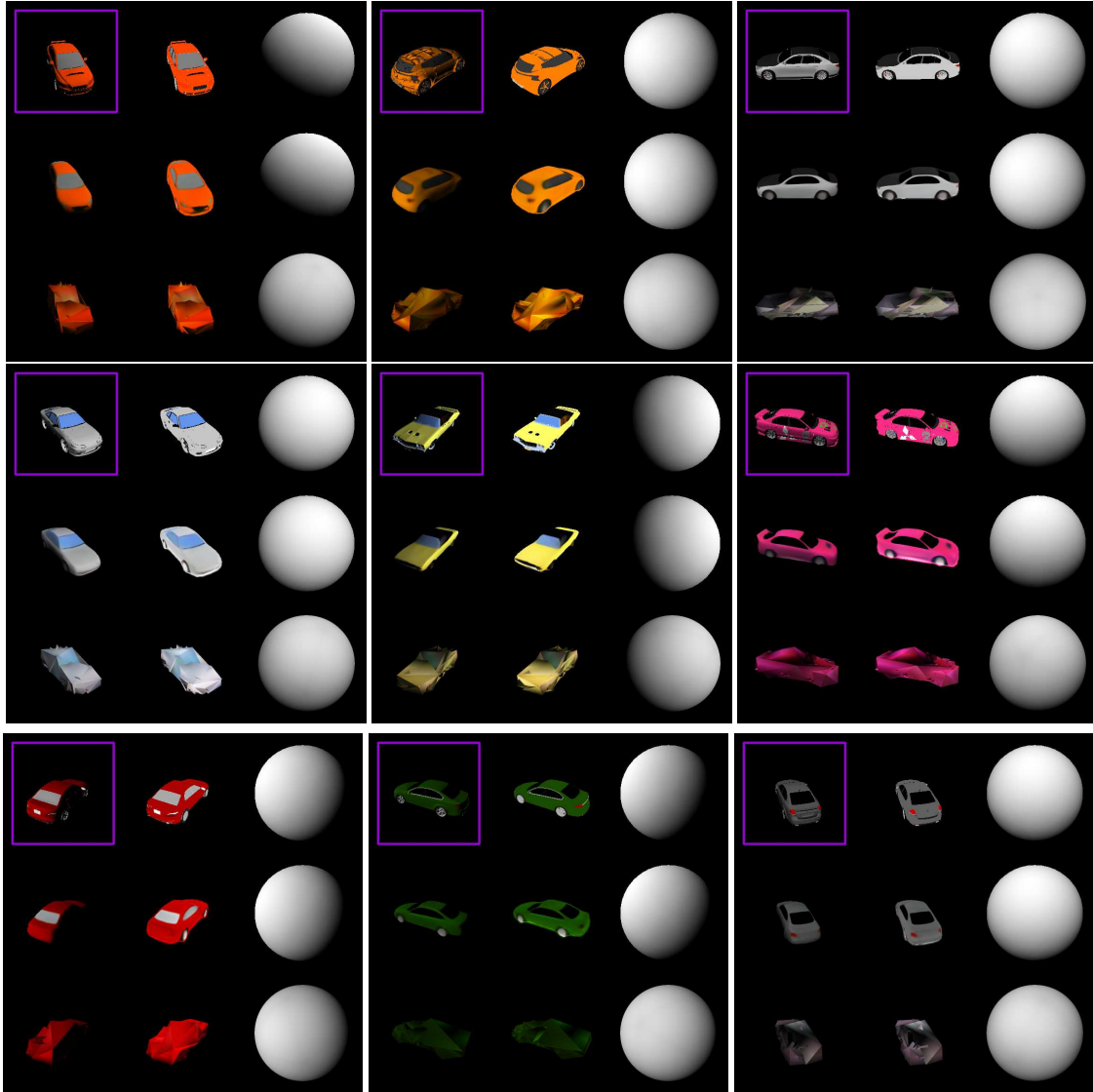


Figure 3.9: **Qualitative Comparison of Texture and Light with N3MR [122].** We show 9 examples. In each example, **Purple rectangle**: Input image. **First Row**: Ground Truth. **Second Row**: Prediction with DIB-Render. **Third Row**: Prediction of N3MR. **First column**: Texture and Light. **Second column**: Texture. **Third column**: Light. We demonstrate significantly better shape, texture and lighting predictions.

Experimental settings We first train the model without adversarial loss for 50000 iterations then fine-tune it with adversarial loss for extra 15000 steps. We set $\lambda_{adv} = 0.5$, $\lambda_{gp} = 0.5$, and $\lambda_{per} = 1$. We fix the learning rate for the discriminator to $1e^{-5}$ and optimize using Adam [124], with $\alpha = 0.0001$, $\beta_1 = 0.5$, and $\beta_2 = 0.999$.

Dataset Details Similarly, for the experiment with adversarial Loss, since we adopting Phong and Spherical Harmonic lighting models in our learning framework, we use OpenGL render to render images with Phong and Spherical Harmonic lighting models respectively and train the neural network with the corresponding dataset.

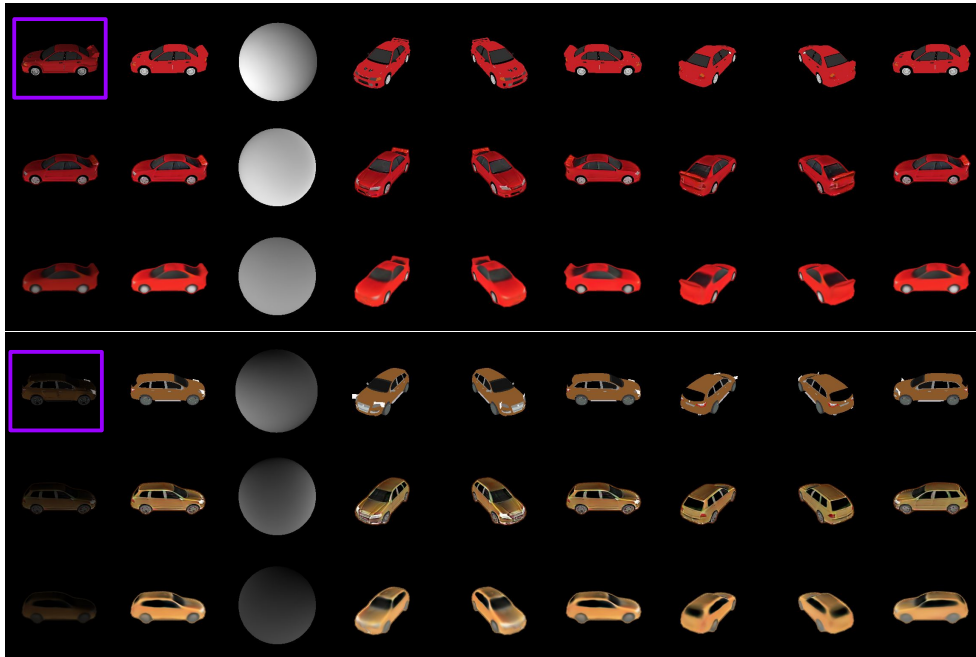


Figure 3.10: **Qualitative Comparison of Texture and Light for the Model Trained w. and w.o. Adversarial Loss.** We show two examples. In each example, **Purple rectangle:** Input image. **First Row:** Ground Truth. **Second Row:** Prediction with adversarial loss. **Third Row:** Prediction without adversarial loss. **First column:** Texture and Light. **Second column:** Texture. **Third column:** Light. **Forth to Ninth column:** Renderings from different views.

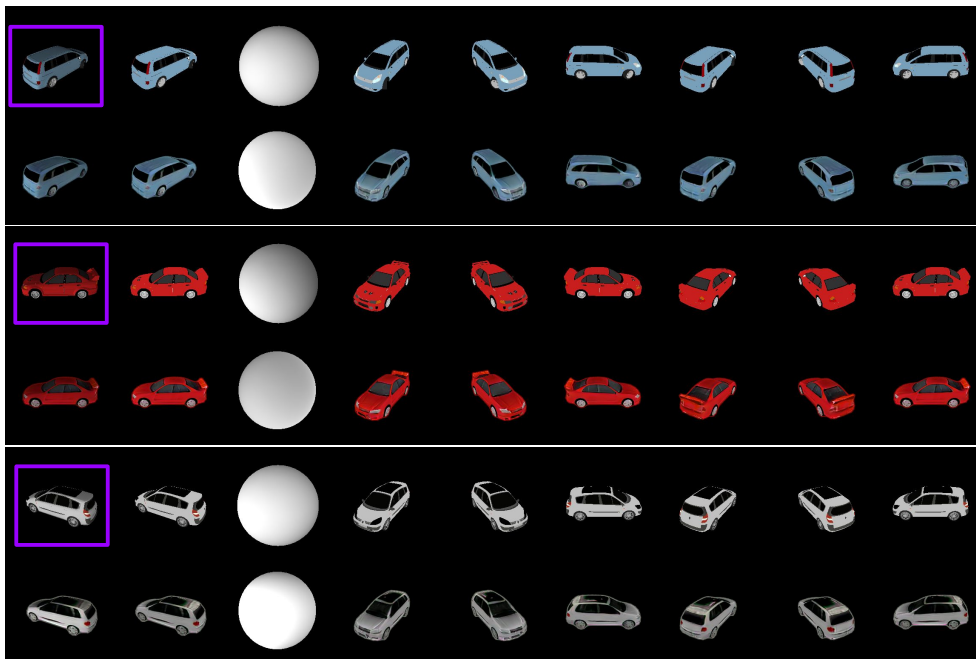


Figure 3.11: **Qualitative Examples for 3D Shape, Texture and Light Prediction for Spherical Harmonic Model.** We show three examples. In each example, **Purple rectangle:** Input image. **First Row:** Ground Truth. **Second Row:** Our Predictions. **First column:** Texture and Light. **Second column:** Texture. **Third column:** Light. **Forth to Ninth column:** Texture from different views.

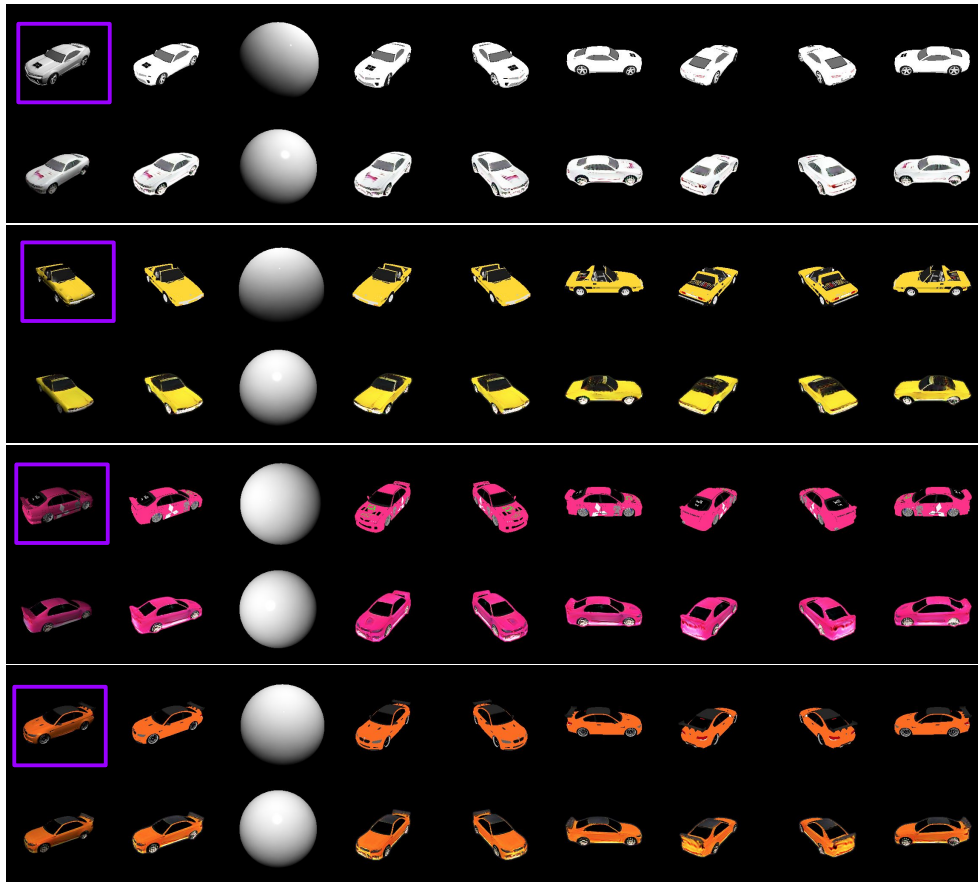


Figure 3.12: **Qualitative Examples for 3D Shape, Texture and Light Prediction for Phong Model.** We show four examples. In each example, **Purple rectangle**: Input image. **First Row**: Ground Truth. **Second Row**: Our Predictions. **First column**: Texture and Light. **Second column**: Texture. **Third column**: Light. **Forth to Ninth column**: Texture from different views.

Adversarial Loss Effect We first show qualitative comparison for model trained with and without adversarial loss in Fig.3.10. We adopt SH lighting models and find the conclusion also applies to Phong model. Clearly, adversarial loss makes the prediction more faithful by introducing more high frequency texture details.

Qualitative Results for SH Lighting Model We show more qualitative results of Spherical Harmonic lighting model in Fig. 3.11. Notice that the network disentangles texture and light quite well, and produces accurate 3D shape.

Qualitative Results for Phong Lighting Model Lastly, we show in qualitative results of Phong lighting model Fig. 3.12. Our predictions recover the correct shape, realistic texture map and reasonable lighting directions.

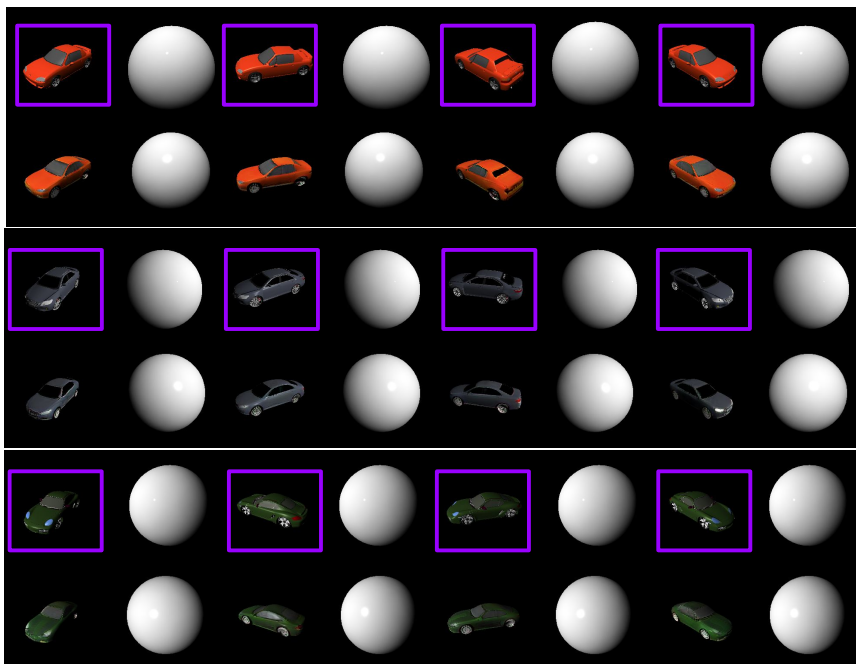


Figure 3.13: **Light & Texture Separation Study for Varying Views.** We show three sets of cars. In each set, we show predictions from different views. **Purple rectangle:** Input image. **First Row:** Ground Truth. **Second Row:** Our Predictions. **First column:** Texture and Light. **Second column:** Light. We show our model have consistent predictions when the input images are with same light and texture but varying views.

3.5.4 Disentanglement Study

Varying views and Lighting Directions To further study texture and light disentanglement, we choose Phong lighting model and render test input images with the same car model but vary the views (Fig. 3.13) or lighting direction (Fig. 3.14).

Fig. 3.13 fixes the lighting and texture but render the car in different camera views, to illustrate consistency of prediction across viewpoints. Our model predicts consistent results under different views. Fig. 3.14 renders images with different lighting directions. Similarly, our model predicts faithful shapes and textures with correct lighting directions.

Shininess Prediction We find our model predict constant shininess α , which didn't reflect the value in GT images. As shown in Fig. 3.12, Fig. 3.13, and Fig. 3.14, almost all the predicted specular lobes have similar shapes(which indicate constant α) and are not similar to the lobe in GT.

Further examples are provided in Fig. 3.15 to evaluate effects of specular. We render images with different shininess constants, but fixed lighting direction, texture and camera view. Here, we find that our model is not able to accurately predict the shininess constant. In this case, the texture map erroneously compensates for the shininess effect. This might be

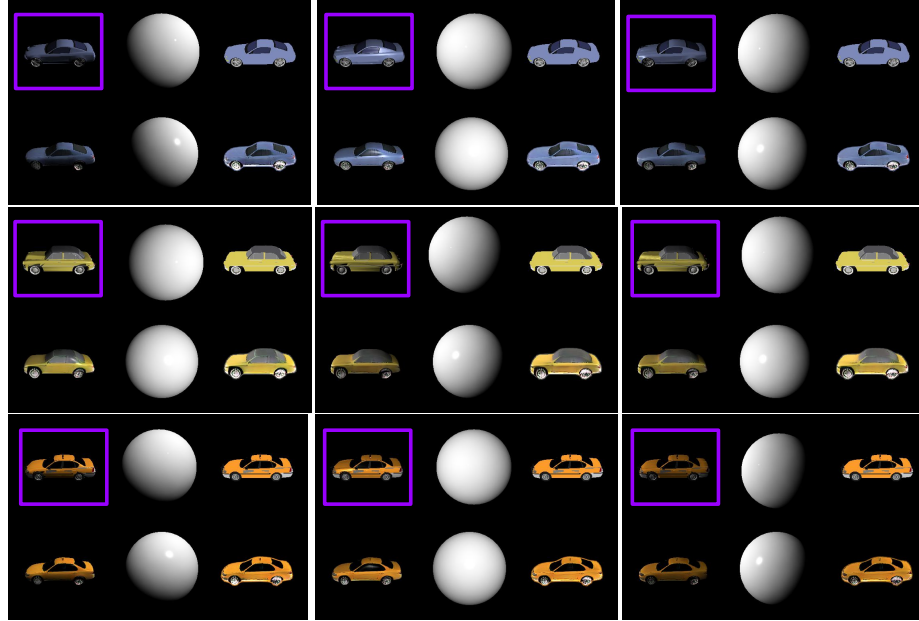


Figure 3.14: **Light & Texture Separation Study for Varying Lighting Directions.** We show three sets of cars. In each set, we show predictions from different lighting directions. **Purple rectangle:** Input image. **First Row:** Ground Truth. **Second Row:** Our Predictions. **First column:** Texture and Light. **Second column:** Light. **Third column:** Texture. We show our model have consistent predictions when the input images have varying lighting directions.

because the shininess effect is not significant enough to be learned by a neural network though 2D supervision. This is one limitation in DIB-R which we address in next chapter.

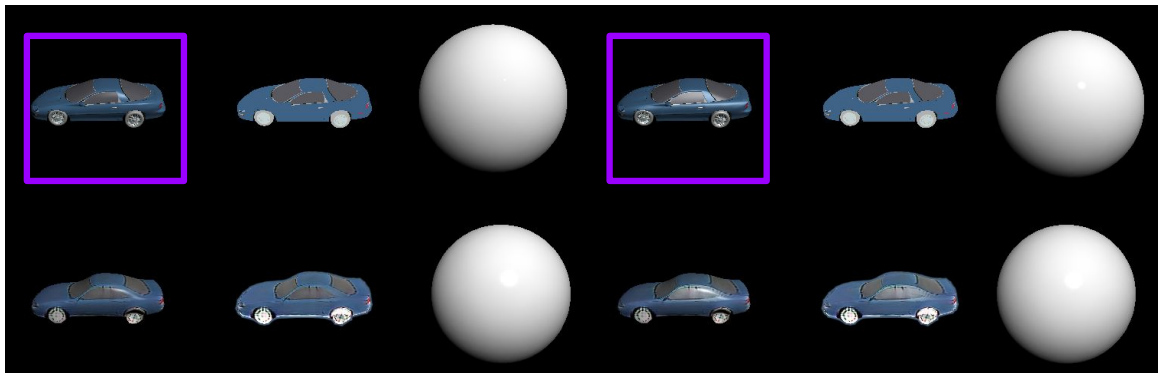


Figure 3.15: **Shininess Separation Study.** We render the same car with two shininess values, as shown in left 3 columns and right 3 columns. **Purple rect:** Input image. **First Row:** Ground Truth. **Second Row:** Our Predictions. **First column:** Texture and Light. **Second column:** Texture. **Third column:** Light. The network predicts incorrect shininess constants and compensates the shininess effects in texture maps, especially in the second example.

3.6 Summary

In this chapter, we proposed a complete rasterization-based differentiable renderer for which gradients can be computed analytically. Our framework, when wrapped around a neural network, learns to predict shape, texture, and light from single images.

As a rasterization-based renderer, DIB-R also has limitations. The most challenge is that it

cannot model advanced lighting effects like specular or surface reflectance. As a result, the secondary lighting effects are entangled in the texture prediction, as shown in Fig. 3.15. We address this issue in next chapter.

Chapter 4

A One-bounce Ray-tracing based Renderer

We consider the challenging problem of predicting intrinsic object properties from a single image by exploiting differentiable renderers. Many previous learning-based approaches for inverse graphics adopt rasterization-based renderers and assume naive lighting and material models, which often fail to account for non-Lambertian, specular reflections commonly observed in the wild. In this work, we propose DIB-R++, a hybrid differentiable renderer which supports these photorealistic effects by combining rasterization and ray-tracing, taking the advantage of their respective strengths—speed and realism. Our renderer incorporates environmental lighting and spatially-varying material models to efficiently approximate light transport, either through direct estimation or via spherical basis functions. Compared to more advanced physics-based differentiable renderers leveraging path tracing, DIB-R++ is highly performant due to its compact and expressive shading model, which enables easy integration with learning frameworks for geometry, reflectance and lighting prediction from a single image without requiring any ground-truth. We experimentally demonstrate that our approach achieves superior material and lighting disentanglement compared to existing rasterization-based approaches.

4.1 Introduction

Inferring intrinsic 3D properties from 2D images is a long-standing goal of computer vision [18]. In recent years, differentiable rendering has shown great promise in estimating shape, reflectance and illumination from 2D images. Differentiable renderers have become natural candidates for learning-based inverse rendering applications, where image synthesis

algorithms and neural networks can be jointly optimized to model physical aspects of objects from posed images, either by leveraging strong data priors or by directly modeling the interactions between light and surfaces.

Not all differentiable renderers are made equal. On the one hand, recent physics-based differentiable rendering techniques [144, 186, 16, 234, 284] try to model the full light transport with proper visibility gradients, but they tend to require careful initialization of scene parameters and typically exhibit high computational cost which limits their usage in larger end-to-end learning pipelines. On the other hand, performance-oriented differentiable renderers [38, 128, 154, 122] trade physical accuracy for scalability and speed by approximating scene elements through neural representations or by employing simpler shading models. While the latter line of work has proven to be successful in 3D scene reconstruction, the frequent assumptions of Lambertian-only surfaces and low-frequency lighting prevent these works from modeling more complex specular transport commonly observed in the real world.

In this work, we consider the problem of *single-view 3D object reconstruction* without any 3D supervision. To this end, we propose DIB-R++, a hybrid differentiable renderer that combines rasterization and ray-tracing through an efficient deferred rendering framework. Our framework builds on top of DIB-R [38] and integrates physics-based lighting and material models to capture challenging non-Lambertian reflectance under unknown poses and illumination. Our method is versatile and supports both single-bounce ray-tracing and a spherical Gaussian representation for a compact approximation of direct illumination, allowing us to adapt and tune the shading model based on the radiometric complexity of the scene. We validate our technique and demonstrate superior performance on reconstructing realistic materials BRDFs and lighting configurations over prior rasterization-based methods.

4.2 Related Work

Differentiable Rendering Research on differentiable rendering can be divided into two categories: physics-based methods focusing on photorealistic image quality, and approximation methods aiming at higher performance. The former differentiates the forward light transport simulation [144, 186, 16, 234, 284] with careful handling of geometric discontinuities. While capable of supporting global illumination, these techniques tend to be relatively slow to optimize or require a detailed initial description of the input in terms of geometry, materials, lighting and camera, which prevents their deployment in the wild. The latter line of works leverages simpler local shading models. Along this axis, rasterization-based differentiable renderers [38, 128, 154, 122, 58] approximate gradients by generating derivatives

from projected pixels to 3D parameters. These methods are restricted to primary visibility and ignore indirect lighting effects by construction, but their simplicity and efficiency offer an attractive trade-off for 3D reconstruction. We follow this line of work and build atop DIB-R [38, 102] by augmenting its shading models with physics-based ones.

Learning-based Inverse Graphics Recent research on inverse graphics targets the ill-posed problem of jointly estimating geometry, reflectance and illumination from image observations using neural networks. For single image inverse rendering, one dominant approach is to employ 2D CNNs to learn data-driven features and use synthetic data as supervision [180, 226, 149, 147, 264], but these methods do not always generalize to complex real-world images [21]. To overcome the data issue, a recent body of work investigates the use of self-supervised learning to recover scene intrinsics [12], including domain adaptation from synthetic reflectance dataset [156], object symmetry [269, 268], or multi-illumination images depicting the same scene [137, 146, 162, 282]. However, these methods either rely on specific priors or require data sources tedious to capture in practice. Some works tackle the subtask of lighting estimation only [60, 59, 92], but still need to carefully utilize training data that are hard to capture. Most similar to us, DIB-R [38] tackles unsupervised inverse rendering in the context of differentiable rendering. Zhang *et al.* [291] further combines DIB-R with StyleGAN [119] generated images to extract and disentangle 3D knowledge. These works perform inverse rendering from real image collections without supervision, but may fail to capture complex material and lighting effects—in contrast, our method models these directly. Several techniques also try to handle more photorealistic effects but typically require complex capturing settings, such as controllable lighting [134, 135], a co-located camera-flashlight setup [179, 50, 148, 22, 23, 27, 216, 160], and densely captured multi-view images [51, 271, 26, 288] with additional known lighting [63] or hand-crafted inductive labels [184]. In our work, we propose a hybrid differentiable renderer and learn to disentangle complex specular effects given a single image. Similar to the recent NeRD [26] and PhysSG [288] which recover non-Lambertian reflectance and illumination with a spherical Gaussian (SG) basis [259], we also employ SGs to model the SV-BRDF and incident lighting, but apply this representation to mesh-based differentiable rendering with direct access to the surface.

4.3 Differentiable Deferred Rendering

In this section, we introduce DIB-R++, our differentiable rendering framework based on deferred shading [49]. DIB-R++ is a hybrid differentiable renderer that can efficiently approximate direct illumination and synthesize high-quality images. Concretely, our renderer

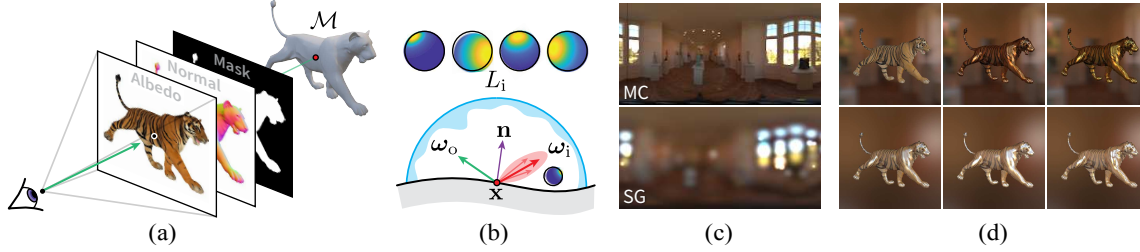


Figure 4.1: **Overview.** Given a 3D object \mathcal{M} , we employ (a) a rasterization-based renderer to obtain diffuse albedo, surface normals and mask maps. In the shading pass (b), we then use these buffers to compute the incident radiance by sampling or by representing lighting and the specular BRDF using a spherical Gaussian basis. Depending on the representation used in (c), we can recover a wide gamut of specular/glossy appearances (d).

leverages the differentiable rasterization framework of DIB-R [38] to recover shape attributes and further employs physics-based material and lighting models to estimate appearance.

4.3.1 Overview

We provide an overview of our technique in Fig. 4.1. We first rasterize a 3D mesh to obtain diffuse albedo and material maps, surface normals, and a silhouette mask. This information is deferred to the shading pass, where outgoing radiance is either estimated stochastically or approximated using a spherical Gaussian basis. The rasterizer and shader are differentiable by design, allowing gradients to be propagated to lighting, material and shape parameters for downstream learning tasks.

4.3.2 Background

Our goal is to provide a differentiable formulation of the rendering process to enable fast inverse rendering from 2D images. Let \mathcal{M} be a 3D object in a virtual scene. We start from the (non-emissive) rendering equation (RE) [115], which states that the outgoing radiance L_o at any 3D surface point $\mathbf{x} \in \mathcal{M}$ in the camera direction ω_o is given by

$$L_o(\mathbf{x}, \omega_o) = \int_{\mathcal{H}^2} f_r(\mathbf{x}, \omega_i, \omega_o) L_i(\mathbf{x}, \omega_i) |\mathbf{n} \cdot \omega_i| d\omega_i, \quad (4.1)$$

where L_i is the incident radiance, f_r is the (spatially-varying) bidirectional reflectance distribution function (SV-BRDF) and \mathbf{n} is the surface normal at \mathbf{x} . The domain of integration is the unit hemisphere \mathcal{H}^2 of incoming light directions ω_i . The BRDF characterizes the surface's response to illumination from different directions and is modulated by the cosine foreshortening term $|\mathbf{n} \cdot \omega|$. Intuitively, Eq. (4.1) captures an energy balance and computes how much light is received and scattered at a shading point in a particular direction.

Estimating the RE typically requires Monte Carlo (MC) integration [199], which involves tracing rays from the camera into the scene. Albeit physically correct, this process is

computationally expensive and does not generally admit a closed-form solution. MC estimators can exhibit high variance and may produce noisy pixel gradients at low sample count, which may significantly impact performance and convergence. To keep the problem tractable, we thus make several approximations of Eq. (4.1), which we detail in the next section.

4.3.3 Two-stage Deferred Rendering

We now describe our rendering framework (Fig. 4.1). We start by defining three families of parameters, where $\boldsymbol{\pi} \in \mathbb{R}^{d_\pi}$ encodes the shape attributes (e.g., vertex positions), $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$ describes the material properties, and $\boldsymbol{\gamma} \in \mathbb{R}^{d_\gamma}$ captures the illumination in the scene¹. In what follows, we shall only consider a single pixel, indexed by p , within an RGB image $I \in \mathbb{R}_+^{3 \times h \times w}$ for notational simplicity.

Stage 1: Rasterization Pass. We first employ a differentiable rasterizer R [38] to generate primary rays $\boldsymbol{\omega}_o \in \mathcal{S}^2$ from a camera viewpoint \boldsymbol{v} and render our scene \mathcal{M} into geometry buffers (commonly called *G-buffers*) containing the surface intersection point $\mathbf{x}_p \in \mathbb{R}^3$, the surface normal $\mathbf{n}_p \in \mathcal{S}^2$, and the spatially-varying material parameters $\boldsymbol{\theta}_p$ (e.g., diffuse albedo). This rendering pass also returns a visibility mask $v_p \in \{0, 1\}$ indicating whether pixel p is occupied by the rendered object, separating the foreground object I_f from its background environment I_b so that $I = I_f + I_b$. We have:

$$R(p, \boldsymbol{\omega}_o; \boldsymbol{v}, \boldsymbol{\pi}, \boldsymbol{\theta}) = (\mathbf{x}_p, \mathbf{n}_p, \boldsymbol{\theta}_p, v_p) . \quad (4.2)$$

Stage 2: Shading Pass. Given surface properties and outgoing direction $\boldsymbol{\omega}_o$, we then approximate the outgoing radiance $L_o(\mathbf{x}_p, \boldsymbol{\omega}_o)$ through several key assumptions. First, we restrict ourselves to direct illumination only (i.e. single-bounce scattering) and assume that the incoming radiance is given by a distant environment map $L_i : \mathcal{S}^2 \rightarrow \mathbb{R}_+^3$. Therefore, we do not model self-occlusion and $L_i(\mathbf{x}_p, \boldsymbol{\omega}_i) \equiv L_i(\boldsymbol{\omega}_i; \boldsymbol{\gamma})$. Such simplification largely reduces computation and memory costs and is trivially differentiable. Second, we assume that the material parameters $\boldsymbol{\theta}$ can model both diffuse and specular view-dependent effects. At a high level, we define our shading model S so that:

$$S(\mathbf{x}_p, \mathbf{n}_p, \boldsymbol{\omega}_o, \boldsymbol{\theta}_p; \boldsymbol{\gamma}) \approx L_o(\mathbf{x}_p, \boldsymbol{\omega}_o). \quad (4.3)$$

¹In Chapter 3, we denote \mathcal{M} as $\mathcal{M} = (S, T)$, where S is shape, T is texture map. Here we slightly abuse notation \mathcal{M} such that $\mathcal{M} = (\boldsymbol{\pi}, \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents SVBRDF for explicit, which contains both surface diffuse albedo map(texture) and reflectance parameters.

Importantly, a differentiable parameterization of S enables the computation of pixel gradients with respect to all scene parameters $\Theta = (\pi, \theta, \gamma)$ by differentiating $I_p(\Theta) = (S \circ R)(p, \omega_o; \mathbf{v}, \pi, \theta, \gamma)$. Given a scalar objective function defined on the rendered output I , $\partial I / \partial \pi$ is computed using DIB-R [38]. In what follows, we thus mainly focus on formulating $\partial I / \partial \{\theta, \gamma\}$ so that all gradients can be computed using the chain rule, allowing for joint optimization of geometry, material and lighting parameters. We assume henceforth a fixed pixel p for conciseness, and remove the subscript.

4.3.4 Shading Models

Since our primary goal is to capture a wide range of appearances, we provide two simple techniques to approximate Eq. (4.1): Monte Carlo (MC) and spherical Gaussians (SG). The former targets more mirror-like objects and can better approximate higher frequencies in the integrand, but is more expensive to compute. The latter is more robust to roughness variations but is limited by the number of basis elements. To model reflectance, we choose to use a simplified version of the isotropic Disney BRDF [29, 57] based on the Cook–Torrance model [44], which includes diffuse albedo $\mathbf{a} \in [0, 1]^3$, specular albedo $s \in [0, 1]$, surface roughness $\beta \in [0, 1]$ and metalness $m \in [0, 1]$. Metalness allows us to model both metals and plastics in a unified framework. We let the diffuse albedo vary spatially ($\mathbf{a} = \mathbf{a}(\mathbf{x})$) and *globally* define all other attributes to restrict the number of learnable parameters.

Monte Carlo Shading Given a 3D surface point $\mathbf{x} \in \mathcal{M}$ to shade, we importance sample the BRDF to obtain N light directions ω_i^k and compute the BRDF value. We represent the incident lighting $L_i^{(\text{MC})}$ as a high-dynamic range image $\gamma \in \mathbb{R}_+^{3 \times h_l \times w_l}$ using an equirectangular projection, which can be queried for any direction via interpolation between nearby pixels. The final pixel color is then computed as the average over all samples, divided by the probability of sampling ω_i^k :

$$S^{(\text{MC})}(\mathbf{x}, \mathbf{n}, \omega_o; \theta, \gamma) = \frac{1}{N} \sum_{k=1}^N \frac{f_r(\mathbf{x}, \omega_i^k, \omega_o; \theta) L_i^{(\text{MC})}(\omega_i^k; \gamma) |\mathbf{n} \cdot \omega_i^k|}{p(\omega_i^k)}. \quad (4.4)$$

Note that this shading function is differentiable. Namely, for each sampled ray, we can compute the gradients from the pixel color $S^{(\text{MC})}$ to the rendering parameters π, θ, γ . The full gradients can be computed by averaging all the sampled rays. When the surface is near-specular (e.g., a mirror), one can efficiently estimate the RE as reflected rays are concentrated in bundles (e.g., to satisfy the law of reflection). However, this estimator can suffer from high variance for rougher surfaces; a higher number of samples may be necessary to produce usable gradients. While this can be partially improved with multiple importance sampling [253], emitter sampling would add a significant overhead due to the

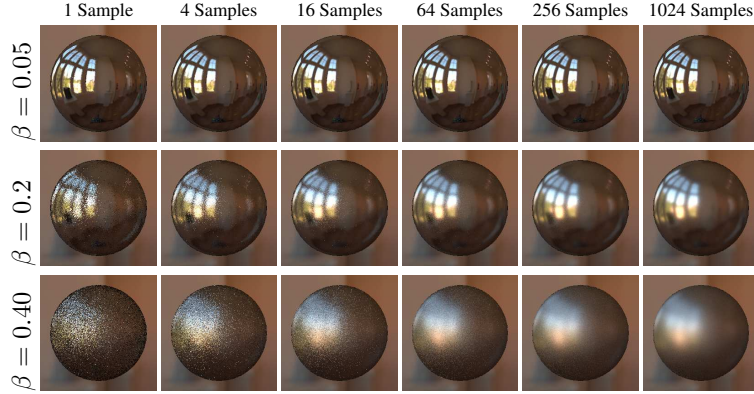


Figure 4.2: **Sample Count vs. Roughness in Monte Carlo Shading.** High roughness surfaces require more samples to render into noise-free images.

environment map being updated at every optimization step. This motivates the use of a more compact representation.

Spherical Gaussian Shading To further accelerate rendering while preserving expressivity in our shading model, we use a spherical Gaussian (SG) [259] representation. Projecting both the cosine-weighted BRDF and incident radiance into an SG basis allows for fast, analytic integration within our differentiable shader, at the cost of some high frequency features in the integrand. Concretely, an SG kernel has the form $\mathcal{G}(\omega; \xi, \lambda, \mu) = \mu e^{\lambda(\xi \cdot \omega - 1)}$, where $\omega \in \mathcal{S}^2$ is the input spherical direction to evaluate, $\xi \in \mathcal{S}^2$ is the axis, $\lambda \in \mathbb{R}_+$ is the sharpness, and $\mu \in \mathbb{R}_+$ is the amplitude of the lobe. We represent our environment map using a mixture of K lighting SGs \mathcal{G}_l , so that:

$$L_i^{(\text{SG})}(\omega_i; \gamma) \approx \sum_{k=1}^K \mathcal{G}_l^k(\omega_i; \xi_l^k, \lambda_l^k, \mu_l^k) , \quad (4.5)$$

where $\gamma := \{\xi_l^k, \lambda_l^k, \mu_l^k\}_k$. For the BRDF, we follow Wang *et al.* [259] and fit a single, monochromatic SG to the specular lobe so that $f_r^{(\text{SG})}$ is a sum of diffuse and specular lobes. The full derivation can be found in Appendix B.1. Finally, we approximate the cosine foreshortening term using a single SG $|\mathbf{n} \cdot \omega_i| \approx \mathcal{G}_c(\omega_i; \mathbf{n}, 2.133, 1.17)$ [166]. Regrouping all terms, the final pixel color can be computed as:

$$S^{(\text{SG})}(\mathbf{x}, \mathbf{n}, \omega_o; \theta, \gamma) = \int_{\mathcal{S}^2} f_r^{(\text{SG})}(\mathbf{x}, \omega_i^k, \omega_o; \theta) L_i^{(\text{SG})}(\omega_i; \gamma) \mathcal{G}_c(\omega_i) d\omega_i , \quad (4.6)$$

which has an analytic form that can be automatically differentiated inside our renderer. All parameters of the SGs, as well as the BRDF parameters, are learnable.

4.3.5 DIB-R++ Rendering Effects

In this section, we analyze the rendering effects under different hyperparameters, including sampling count for MC shading and number of SG light components in SG shading. We

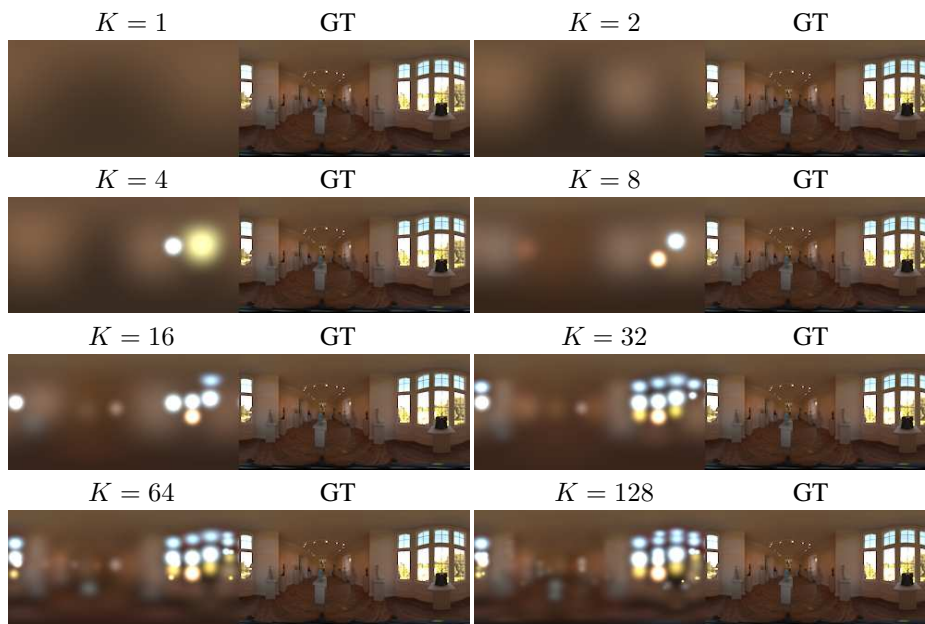


Figure 4.3: **Number of SG Light Component.** More components (larger K) result in better environment map approximation.

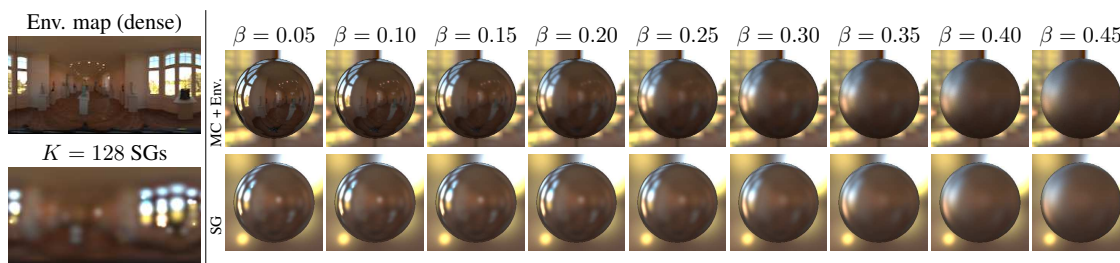


Figure 4.4: **Visual Comparisons Between MC and SG Shading.** On the left, we show an environment map and its SG-representation ($K = 128$). While losing sharp details, SGs only needs 1% parameters compared to the dense pixel HDR map. On the right, we increase roughness left-to-right, revealing that our spherical basis can correctly approximate direct illumination at moderate-to-high roughness.

also compare the rendering effect of MC shading and SG shading and evaluate the rendering speed of the two methods.

Sampling Count As shown in Fig. 4.2, we render a unit sphere with different roughness and sample count. When β is close to 0, a small sample count (e.g. $N = 4$ samples) can render realistic images. However, as β increases, more samples are needed to reduce noise. When β is 0.2, we need $N = 256$ samples. When β is 0.4, even $N = 1024$ samples may still generate noisy results.

Number of SG Light Component As shown in Fig. 4.3, we approximate the same environment map with different numbers of SG components in the mixture. Clearly, the more components we use, the more details we can represent. When we use 128 SGs ($128 \times 7 = 896$ parameters), we can approximate most details of the environment map with only 1% parameters of the whole image ($256 \times 128 \times 3 = 98\,304$ parameters).

Rendering Effects Comparison To visually compare our two shading techniques and understand their limitations, we render a unit sphere under the same lighting (represented differently) in Fig. 4.4. To do so, we first fit a HDR environment map with $K = 128$ SGs

using an equirectangular projection. As shown on the left, SGs smooth out high frequency details and sharp corners but require much fewer parameters to reconstruct incident lighting (896 vs. 98 304). On the right, we visualize the effect of increasing the surface roughness β under the corresponding light representation.

Intuitively, this point of diminishing return indicates that MC is only so useful when the surface reflects most of the incoming light (e.g., a mirror). Indeed, when β is small enough ($\beta \rightarrow 0$) and we deal with a highly non-Lambertian surface, a small number of MC samples are enough to estimate direct illumination, which in turn implies faster render speed and low memory cost. On the other end of the spectrum ($\beta \rightarrow 1$), significantly more samples (e.g., $N > 1000$) are needed to accurately integrate incident light, resulting in longer inference times. In such a case, SGs should be favored since they offer a significant improvement. In the absence of any prior knowledge on the material type, SG shading is preferred. This is reflected in our experiments in Sec. 4.5.

Rendering Speed We further evaluate the rendering performance of different shading methods. The results are shown in Tab. 4.1. We implement both two shading methods in PyTorch [193]. Here, we only consider the time cost for shading, without counting the cost of rasterization. For MC shading, time increases as sample count increases. For example, when the sample count changes from 256 to 1024, the rendering time becomes almost $10\times$ slower. Conversely, SG shading cost is nearly constant. One possible explanation for this is that the number of parameters of SG light is small (no more than 1000 parameters even for 128 SGs), and thus leads to almost constant time cost when computed in parallel with PyTorch. However, as more SGs can significantly impact memory cost, we choose $K = 32$ in the learning tasks.

Comparison with Mitsuba 2 We further compare the rendering performance with Mitsuba2 [186] in Tbl. 4.2. Since both our method and Mitsuba 2 support ray tracing, we evaluate the running times and memory for the same ray tracing optimization task. We restrict the ray samples to be the same (4 samples) and run 300 iterations to optimize an environment map while fixing everything else. Our method performs faster (5.4sec v.s. 9.8sec) and requires less memory(1475MB v.s.

Shading	Sample Count (N)							
MC	1	4	16	64	256	1024		
Time (ms)	4.0	3.7	5.0	9.1	28.3	292.5		
Shading	Number of SG Component (K)							
SG	1	2	4	8	16	32	64	128
Time (ms)	2.6	2.8	2.6	2.6	2.9	2.9	2.5	2.7

Table 4.1: Rendering time comparison. We render a sphere into 256×256 images and compare the rendering time under different settings. In MC shading, more sample count costs higher rendering time, while in SG shading, a different numbers of SG component barely impacts timings.

	DIB-R++	Mitsuba 2
Memory (MB)	1475	3287
Time (s)	5.4	9.8

Table 4.2: Comparison with Mitsuba2. We evaluate the running time and memory within ray tracing optimization task. Our method performs faster and requires less memory compared to Mitsuba2.

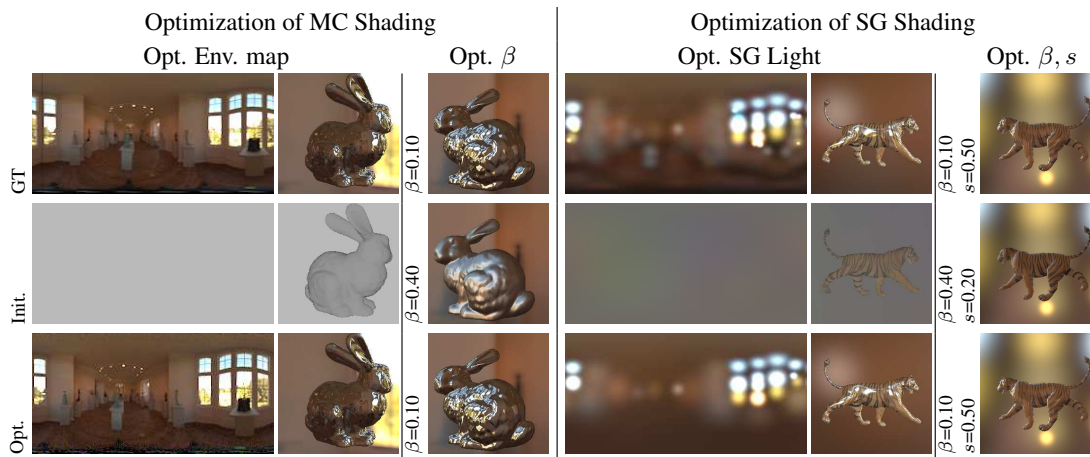


Figure 4.5: **Independent Optimization.** We optimize complex lighting and material for both MC shading (left) and SG shading (right). In each case, we first optimize lighting and show the light and rendered image in the first block. Next we optimize the surface material and show the surface material value and rendered image in the second block.

3287MB), which demonstrates DIB-R++ is highly performant. In contrast, Mitsuba 2 is a more powerful and general differentiable rendering library. However, it is also very heavy, therefore the speed and memory cost is limited.

4.3.6 Optimization of Lighting and Material

The design of our differentiable renderer allows for optimization over geometry, lighting and material parameters. In this section we focus on lighting and material optimization and disentanglement. To validate our technique, we render a unit sphere into 24 views, fixing the shape and optimizing light and surface material from 24 multi-view 2D images. We first showcase independent optimization, where we optimize only one variable and keep the rest the same as GT. We further study joint optimization with more than one variables, which is a more challenging ill-posed problem.

Separate Optimization We perform a sanity check on our renderer in Fig. 4.5 by optimizing for lighting and reflectance properties from a multi-view image L_1 -loss with fixed geometry. We show the ground-truth (GT) parameters and rendered images in the first row, along with initial parameters (Init.) in the second row. Here, we optimize parameters separately using gradient-descent while the others are kept fixed to validate each component of our shading model. DIB-R++ can successfully estimate material and lighting parameters, including the environmental lighting, surface roughness and specular albedo of the object. We find that the converged material parameters closely match GT, while the optimized environment map loses some details due to gradients coming entirely from surface reflections (foreground supervision only). In particular, surface highlights are well captured by our technique.

Disentanglement Analysis Light and material are entangled during the complex image formation process. Thus, jointly recover all of them is ill-posed as there might be several solutions for the same input image. As shown in Fig. 4.6, we can get the same rendering with sharp

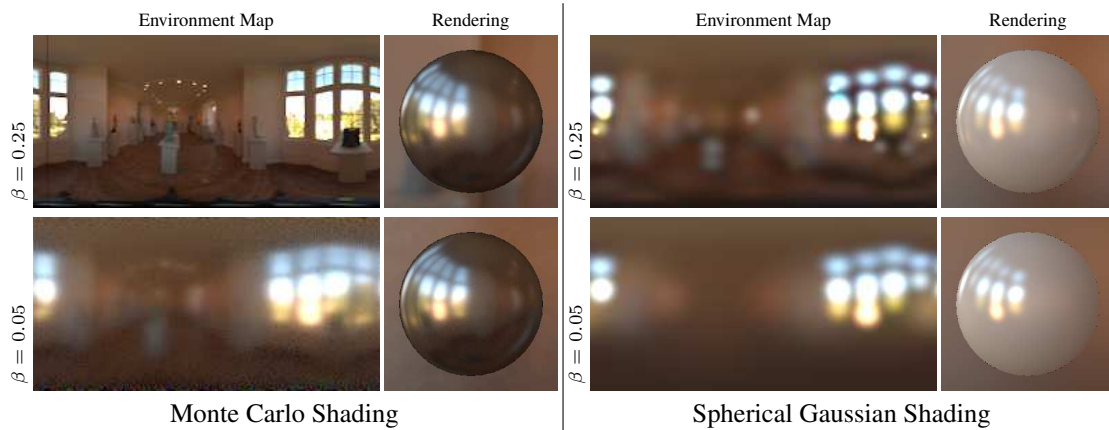


Figure 4.6: **Disentanglement Analysis.** We render a sphere with 1) large roughness and sharp light (top row) and 2) small roughness and smooth light (bottom row), but their renderings are almost the same. It shows the ambiguity between light and surface material.

lighting and large roughness, or smooth lighting and small roughness. Such ambiguities occur regardless of MC and SG shading.

Joint Optimization We explore the extent to which our DIB-R++ can separate lighting and material (including diffuse albedo and surface material such as roughness and specular albedo) from only the rendered images. We jointly optimize light and surface material from 24 multi-view 2D images. We show Monte Carlo-based shading optimization in Fig. 4.7 and Spherical Gaussian-based shading optimization in Fig. 4.8.

MC Shading Several interesting properties can be found in Fig. 4.7. First, DIB-R++ successfully optimizes the light and surface material such that the rendered images (Col. 4 & 7) are close to the GT images (Col. 1) after convergence. Moreover, the converged parameters are also reasonable, e.g. the optimized light recovers most high frequency details, especially in small roughness cases. Second, the converged parameters are not exactly the same as GT settings. For instance, compared to GT, the environment map is always brighter while the diffuse albedo is always darker. As shown in Fig. 4.6, the joint inverse problem is ill-posed so there exists no unique solutions. Third, when the surface is primarily Lambertian, the rendered images can lose high frequency information. As a result, the recovered light map is also blurry. The converged roughness tends to be smaller than GT to compensate for this blurriness.

SG Shading On the other hand, Spherical Gaussian shading has similar properties. We show optimization results in Fig. 4.8 and all the observations in MC shading can be applied to SG shading. More precisely, the optimized light always looks blurry while the optimized surface roughness is smaller compared to GT. This is also reflected in the learning task, which we detail in Sec. 4.6.

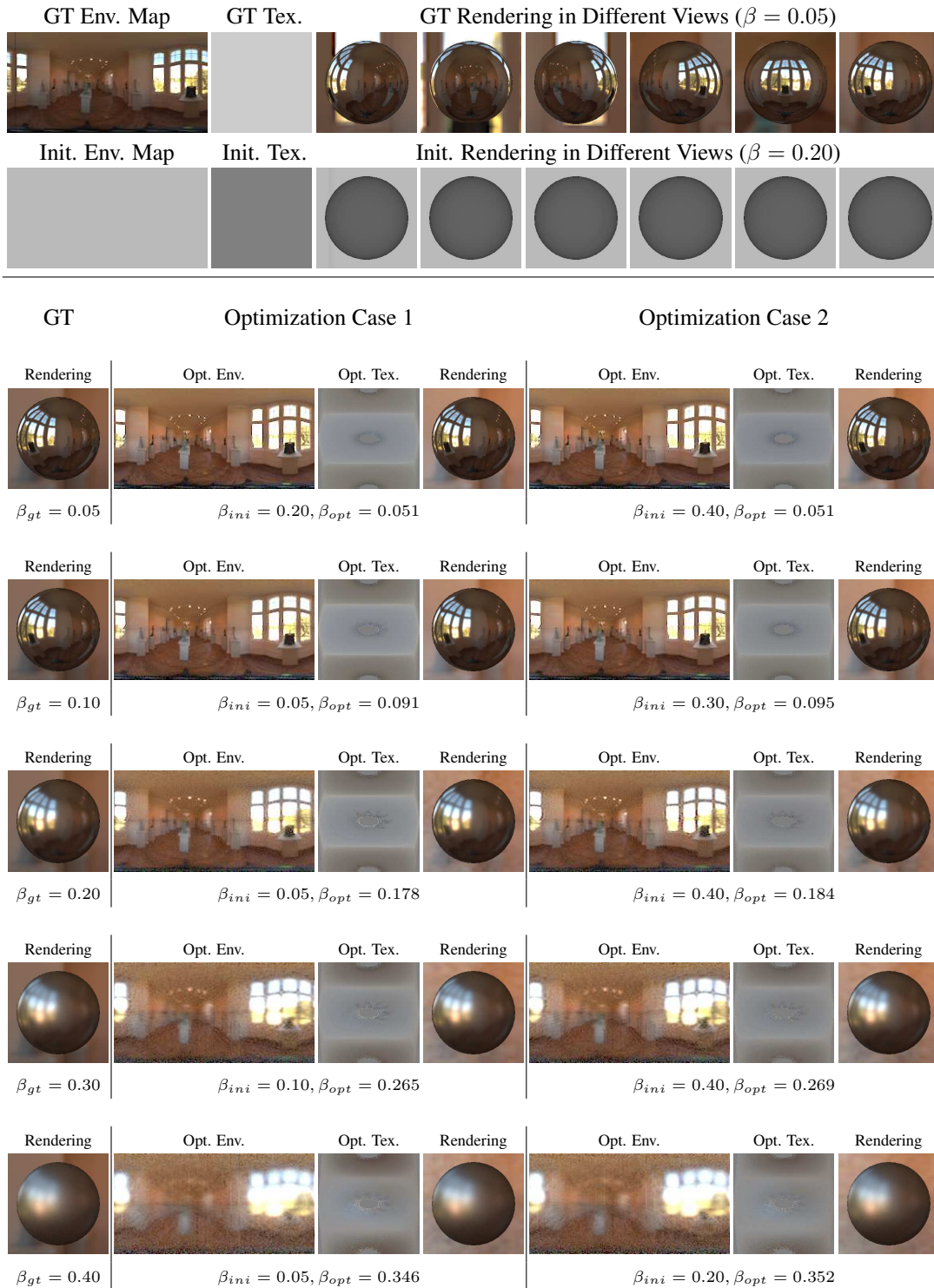


Figure 4.7: Monte Carlo Shading Optimization Results. We fix the shape and optimize the environment map, diffuse albedo and surface roughness jointly from a foreground image loss only. To validate the convergence of different surface materials, we use the same environment map and albedo initialization but employ different roughness initialization and optimize for different GT roughness settings. **Top:** In the first two rows, we show the GT and initialized environment map, texture and rendering in 6 views. **Bottom:** In each row we optimize for one specific roughness, and in each column block we start with different roughness initialization. **Comments:** We successfully optimize all the parameters such that the rendered images (Col. 4 & 7) are very close to the GT rendering (Col. 1). However, due to the ill-posedness of the problem, the converged environment map, diffuse albedo roughness are not exactly the same as GT.

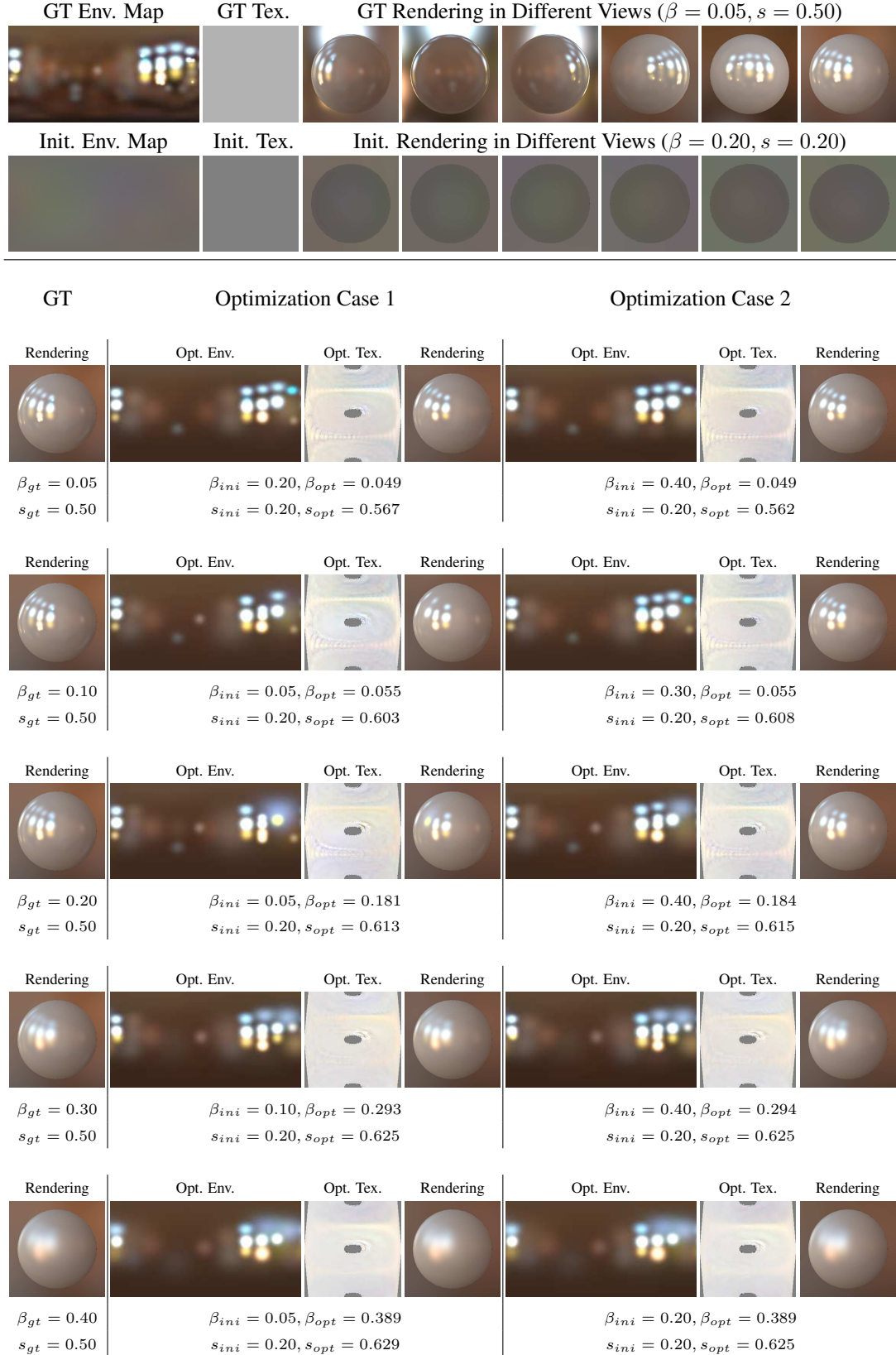


Figure 4.8: **Spherical Gaussian Shading Optimization Results.** We fix the shape and optimize the environment map and surface material properties (roughness and specular albedo) jointly from a foreground image loss only. To validate the convergence of different surface material, we use the same environment map and texture initialization but different material initialization and optimize for different GT material settings. **Top:** In the first two rows we show GT and initialized environment map, diffuse albedo and rendering in 6 views. **Bottom:** In each row, we optimize for a specific GT material setting while in each column block we start from different surface material initialization. **Comments:** We successfully optimize all the parameters such that the rendered images (Col. 4 & 7) are very close to the GT rendering (Col. 1). However, due to the problem is ill-posed, the converged environment map, diffuse albedo and roughness are not exactly the same as GT.

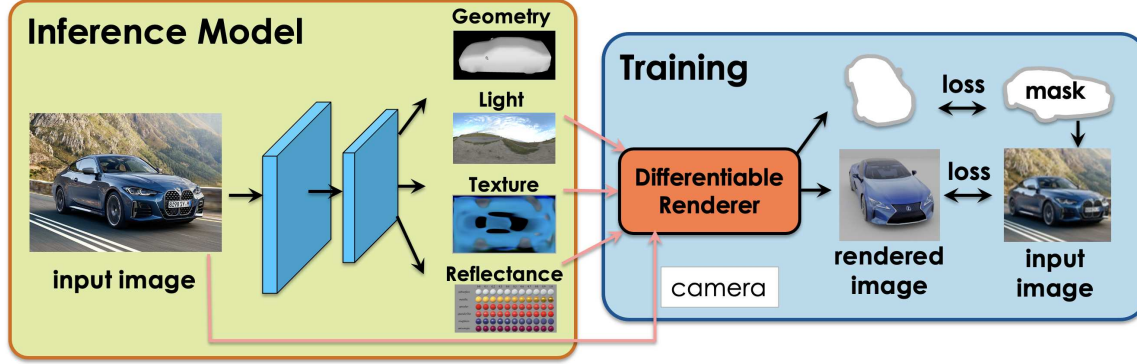


Figure 4.9: **Training Pipeline.** Given an image, we employ a neural network to predict all 3D attributes. We then adopt DIB-R++ to render them back to the images and supervise via the loss between the rendered image and input image. This model can be trained without any 3D supervision.

4.4 Single-view 3D Object Reconstruction

We demonstrate the effectiveness of our hybrid framework through the learning-based problem of single image 3D reconstruction without supervision. We show a pipeline figure in Fig. 4.9, where we embed DIB-R++ into a learning framework and adopt neural networks to predict 3D properties without 3D training data.

Specifically, given an image, we first employ a neural network to predict all 3D attributes, including the geometry (sphere deformation, parameterized by π), light (environment map, parameterized by γ), and BRDF (a spatial varying diffuse albedo map and a global surface reflectance material, parameterized by θ). We can then adopt DIB-R++ to render them back to images and supervise via a loss between the rendered image and input image. This model can be trained without any 3D supervision, assuming good priors on geometry, light and materials. While previous works [38, 291] generally focus on diffuse illumination only, DIB-R++ further generalizes to images containing strong specular lighting effects, disentangling lighting from material and resulting in much cleaner texture and accurate lighting prediction.

We adopt the U-Net [211] architecture of the original DIB-R [38] and modify its output to also predict the appropriate BRDF attributes θ and light parameters γ (pixel colors or SG coefficients) so that $F(I; \vartheta) = (\pi, \theta, \gamma)$. We then render these parameters back to an image \tilde{I} using our differentiable renderer and apply a loss \mathcal{L} on the RGB output to compare the input image I and the rendered image \tilde{I} , where:

$$\mathcal{L}(\vartheta) = \lambda_{col} \mathcal{L}_{col}(\tilde{I}, I) + \lambda_{IOU} \mathcal{L}_{IOU}(\tilde{M}, M) + \lambda_{lap} \mathcal{L}_{lap}(\pi) + \lambda_{per} \mathcal{L}_{per}(\tilde{I}, I) . \quad (4.7)$$

Similar to DIB-R [38], we combine multiple consistency losses with regularization terms: \mathcal{L}_{col} is an image loss computing the L_1 distance between the rendered image and the input im-

age(Eq. (3.15)), \mathcal{L}_{IOU} is an Intersection-over-Union (IoU) loss of the rendered silhouette \tilde{M} and the input mask M of the object(Eq. (3.14)), \mathcal{L}_{per} is a perceptual loss [101, 290] computing the L_1 distance between the pre-trained AlexNet [127] feature maps of rendered image and input image, and \mathcal{L}_{lap} is a Laplacian loss [154, 122] to penalize the change in relative positions of neighboring vertices. Compared the the loss equations(Eq. (3.18),Eq. (3.22)) in DIB-R, we find adversarial loss and smooth loss is not necessary and remove them. We set $\lambda_{col} = 20$, $\lambda_{IOU} = 5$, $\lambda_{per} = 0.5$, $\lambda_{lap} = 5$, which we empirically found worked best.

Multi-view Consistency Since we have multi-view images for the same car, similar to [291, 38], we apply a multi-view consistency loss during training, which is the key to force the lighting to be separated from the texture. Specifically, we predict geometry/light/material in one view and supervise it in another view. When the input image has strong directional lighting effects, the optimization understands it should come from lighting instead of the texture, otherwise it would be wrong in the second view.

Mathematically, let us assume we have two images I_1 and I_2 from two different poses (cameras noted as v_1 and v_2) of the same car. We can then predict shape, surface material and lighting for each image, noted as $\pi_1, \theta_1, \gamma_1$ and $\pi_2, \theta_2, \gamma_2$. The multiview consistency loss is given by:

$$\begin{aligned} \mathcal{L}_{col}^{MV} = & \|I_1 - S \circ R(v_1, \pi_1, \theta_1, \gamma_1)\|_1 + \\ & \|I_2 - S \circ R(v_2, \pi_1, \theta_1, \gamma_2)\|_1 + \\ & \|I_1 - S \circ R(v_1, \pi_2, \theta_2, \gamma_1)\|_1 + \\ & \|I_2 - S \circ R(v_2, \pi_2, \theta_2, \gamma_2)\|_1 . \end{aligned} \quad (4.8)$$

We predict shape and material in canonical views while predicting light in image space. Besides the image loss, there are no more consistency regulations, i.e., we did not add any loss term to force π_1, π_2 or θ_1, θ_2 to be the same. Moreover, while more views would provide more constraints, we find that two views are enough. We thus use two views in all our experiments.

Training Time DIB-R++ contains a second shading stage. As a result, it is slower than [38] and [291] which only adopt rasterization. We find all the methods converge in 200 000 iterations, where [38, 291] cost 57 hours while DIB-R++ cost 77 hours. While longer, it is still affordable.

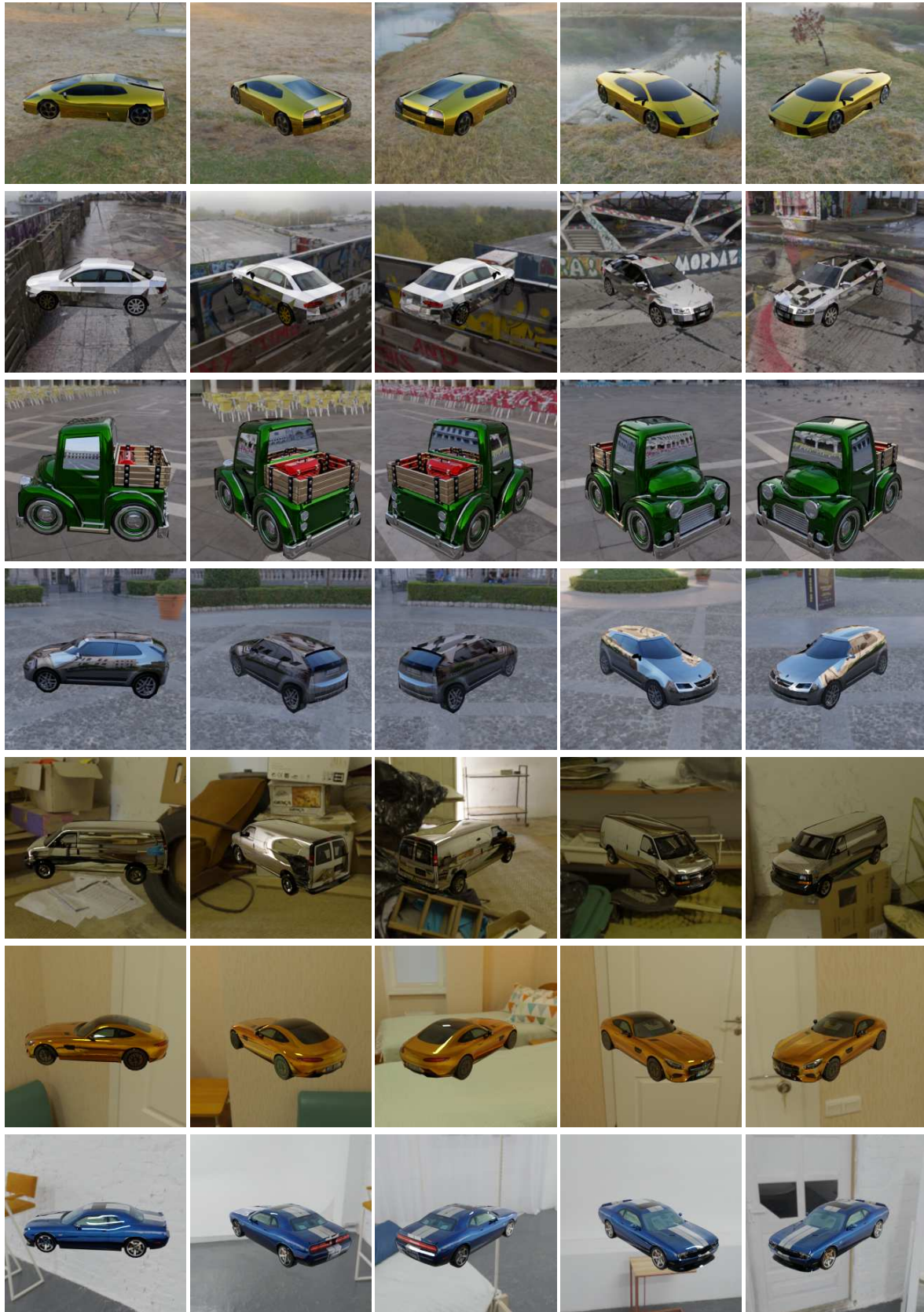


Figure 4.10: **Metallic Surface Dataset Overview.** We render various cars under both indoor and outdoor environment maps. We set the m as 1 and β as 0 and the rendered images are highly reflective. The reflected environment maps can be observed in the car body, which provide strong signals in lighting estimation.

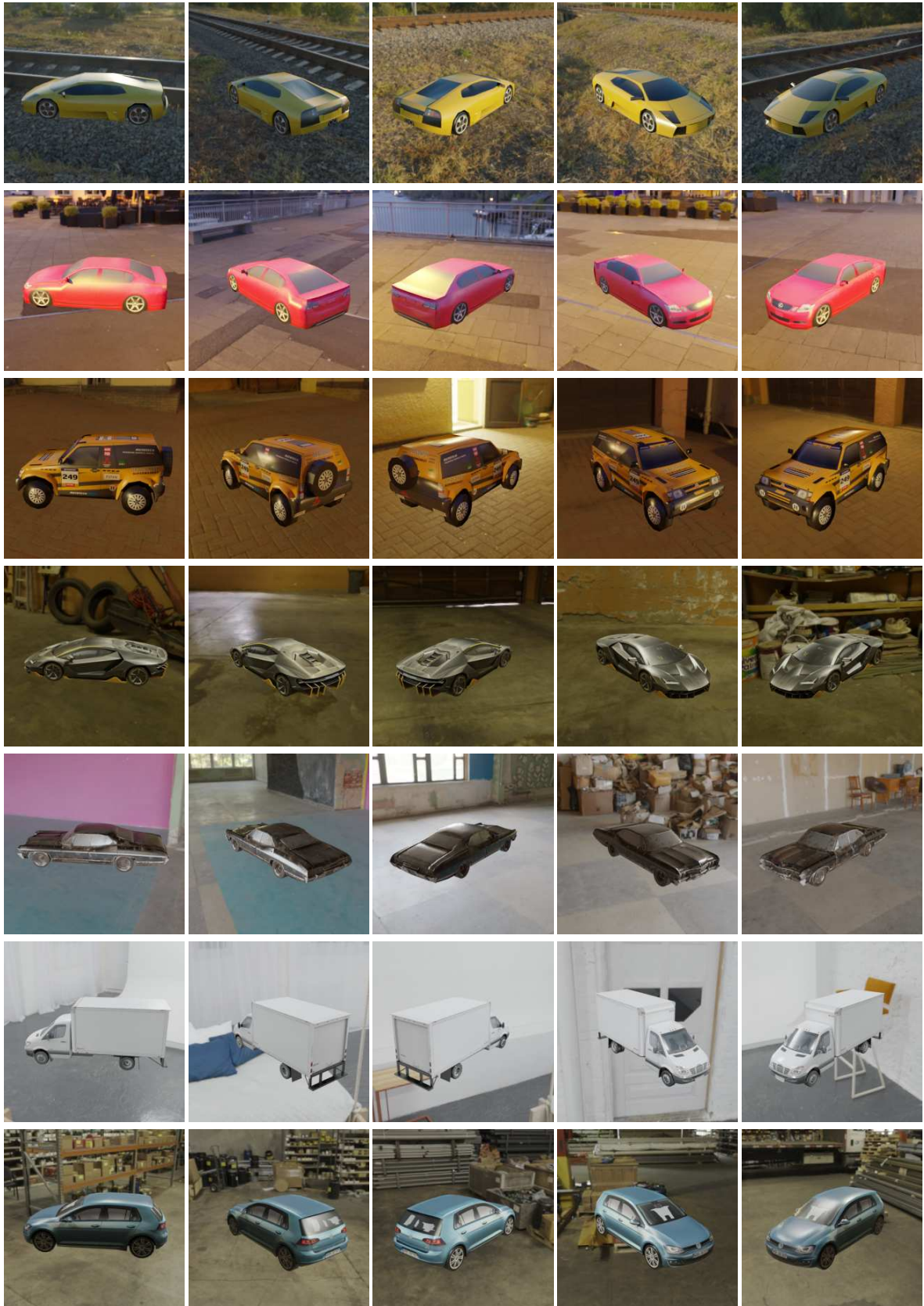


Figure 4.11: **Glossy Surface Dataset Overview.** We render various cars under both indoor and outdoor environment maps. We set the $m = 0$ and vary $\beta \in [0, 0.5]$ so that the rendered images contain view-dependent highlights.

4.5 Experiments

We conduct extensive experiments to evaluate the performance of DIB-R++. We quantitatively evaluate on synthetic data where we have access to ground-truth geometry, material and lighting. Since MC and SG shading have individual pros and cons, we validate them under different settings. In particular, we generate separate datasets with two different surface materials: purely **metallic** surfaces with no roughness, and **glossy** surfaces with random positive roughness. We compare the performance of both shading models against the baseline method, which is DIB-R [38].

4.5.1 Synthetic Datasets

We chose 485 different car models from *TurboSquid*² to prepare data for metallic and glossy surfaces. We also collected 438 freely available high-dynamic range (HDR) environment maps from *HDRI Haven*³ to use as reference lighting, which contain a wide variety of illumination configurations for both indoor and outdoor scenes. For both two datasets we split the training set and testing set to around 4:1 (400 cars for training, 85 cars for testing and 358 envmaps for training, 80 envmaps for testing).

To render all 3D models, we use Blender’s Cycles⁴ path tracer with the Principled BRDF model [29]. We created two datasets on car models: Metallic-Surfaces and Glossy-Surfaces, as shown in Fig. 4.10 and Fig. 4.11, respectively. For metallic surfaces, we set $\beta = 0$ and $m = 1$. Conversely, we set $m = 0$, $s = 1$ and randomly pick $\beta \in [0, 0.4]$ to generate images for glossy surfaces. For each dataset, the car is rendered in 24 fixed views, where we uniformly rotate the camera around. All the 3D models are normalized to be $[-18, 18]$ with camera distance and elevation to be 60 and 27° . We randomly combine 3D models and light and render each car with 3 different environment maps.

4.5.2 Evaluation Metrics

Since GT lighting γ is represented by an environment map while the predicted lighting may be in the form of spherical harmonics [38], spherical Gaussians or HDR texture envmaps with varying resolutions, we cannot directly compute L_1 or L_2 losses. Therefore, we evaluate them with normalized cross correlation (NCC). In particular, we convert the predicted lighting profile to an RGB image $\tilde{\gamma}$ using an equirectangular projection and

²<https://turbosquid.com>. We obtain consent via agreement with TurboSquid, following their license at <https://blog.turbosquid.com/turbosquid-3d-model-license>.

³<https://hdrihaven.com>. We follow the CCO license at <https://hdrihaven.com/p/license.php>.

⁴<https://blender.org>

Shading	Metallic surfaces (\downarrow)				Glossy surfaces (\downarrow)			
	Image	2D IoU	Light	Tex.	Image	2D IoU	Light	Tex.
MC	0.019	0.062	0.074	0.152	0.024	0.061	0.106	0.142
SG	0.019	0.069	0.095	0.218	0.024	0.057	0.091	0.140
SH [38]	0.019	0.056	0.220	0.206	0.024	0.062	0.131	0.152

Table 4.3: Quantitative results of single image 3D Reconstruction on synthetic data. While all the methods achieve comparable performance on re-rendered images and 2D IoUs, both MC and SG achieve better results on lighting and texture. MC is particularly better for metallic surfaces, and SG works best for glossy surfaces.

calculate the loss as:

$$\text{NCC}(\gamma, \tilde{\gamma}) = \frac{\sum \gamma \odot \tilde{\gamma}}{\|\gamma\|_2 \|\tilde{\gamma}\|_2} \quad (4.9)$$

$$\mathcal{L}_{NCC} = 1 - \text{NCC}(\gamma, \tilde{\gamma}) ,$$

Similarly, for material predictions, different uv -parameterizations between GT and our predictions prevents direct comparison. Moreover, the predicted texture might be of different intensity scales compared to the GT. Thus, we first render the GT 3D models and predicted 3D models into images without lighting, and then apply NCC loss to the rendered albedo images as well.

In summary, the image loss is computed via the L_1 loss between the GT images and predicted images. 2D IoU loss is computed via the GT mask and predicted silhouettes [122]. We apply \mathcal{L}_{NCC} to the GT lighting and predicted lighting as the light loss and apply \mathcal{L}_{NCC} to the GT albedo and predicted albedo as the texture loss.

Baseline We compare our method with the rasterization-based baseline DIB-R [38], which supports spherical harmonics (SH) lighting. While the original lighting implementation in DIB-R is monochromatic, we extend it to RGB for a fairer comparison. For quantitative evaluation, we report the common L_1 pixel loss between the re-rendered image using our predictions and ground-truth (GT) image ($\mathcal{L}_{col} = \|\tilde{I} - I\|_1$), and 2D IoU loss between rendered silhouettes and ground-truth masks ($\mathcal{L}_{IOU} = 1 - \frac{\tilde{M} \odot M}{M + M - \tilde{M} \odot M}$). Moreover, we further evaluate the quality of diffuse albedo and lighting predictions using normalized cross correlation (NCC).

4.5.3 Metallic Surfaces

Experimental Settings We first apply all methods to the metallic car dataset. Since this surface property is known a priori, we relax the task for MC shading by setting $\beta = 0$ and only predict geometry, diffuse albedo and lighting from the input image. This allows us to render MC at a low sample count ($N = 4$), achieving higher rendering speed and a lower memory cost. In particular, we predict the relative offset for all $|\mathcal{M}| = 642$ vertices in a mesh and a 256×256 texture map, following the choices in DIB-R [38]. We also predict a 256×256 RGB environment map. For SG shading, we predict all parameters. While shape

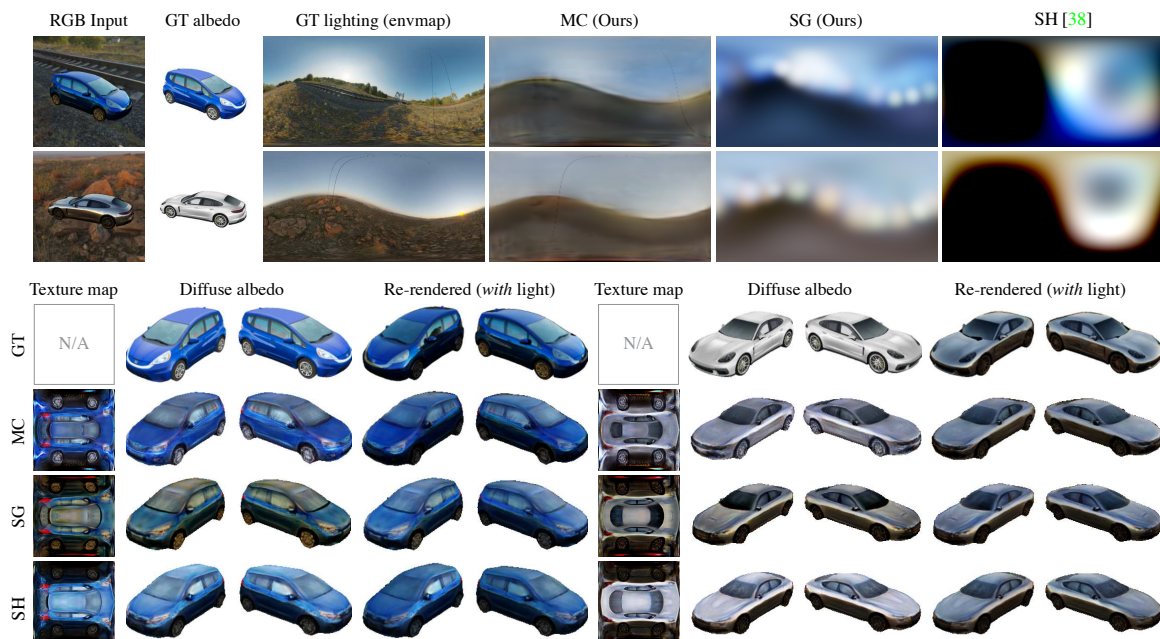


Figure 4.12: **Prediction on the Metallic Car Dataset** ($\beta = 0$). While the re-rendered images look similar, the underlying components (material and lighting) are different. A SH basis (DIB-R [38]) cannot recover the high frequency details of the sky light maps. In this case, MC performs best due to low variance in the estimator for mirror-like BRDFs. SGs can recover the overall form and contrast of the light map, but tend to predict incorrect texture maps incorporating the ground dominant color (e.g., brown). Note that GT texture maps are not available as they cannot be compared due to different uv -parameterizations / texture atlases.

and texture are the same as MC shading, we adopt $K = 32$ for SG and predict two global parameters β and s for the specular BRDF. This keeps the number of parameters relatively low while providing enough flexibility to capture different radiometric configurations.

Experimental Results Quantitative and qualitative results are shown in Fig. 4.12 and Table 4.3 (Left), respectively. Since the main loss function comes from the difference between GT and re-rendered images, we find the re-rendered images (with light) from the predictions are all close to the GT image for different methods, and quantitatively, the image loss and 2D IoU loss are also similar across different models. However, we observe significant differences on the predicted albedo and lighting. Specifically, in Fig. 4.12, MC shading successfully predicts cleaner diffuse albedo maps and more accurate lighting, while DIB-R [38] “bakes in” high specular effects into the texture. Quantitatively, DIB-R++ outperform DIB-R with a $3\times$ improvement in terms of NCC loss for lighting, demonstrating the effectiveness of our DIB-R++. We further compare MC shading with SG shading. While SG shading achieves reasonable lighting predictions, it fails to reconstruct the high frequency details in the lighting and has circular spot effects caused by the isotropic SGs. Finally, we note that due to the ambiguity of the learning task, the overall intensities of all predicted texture maps can largely vary. Still, we observe that MC can better recover fine details, such as the wheels’ rims. More results and are shown in Fig. 4.13.

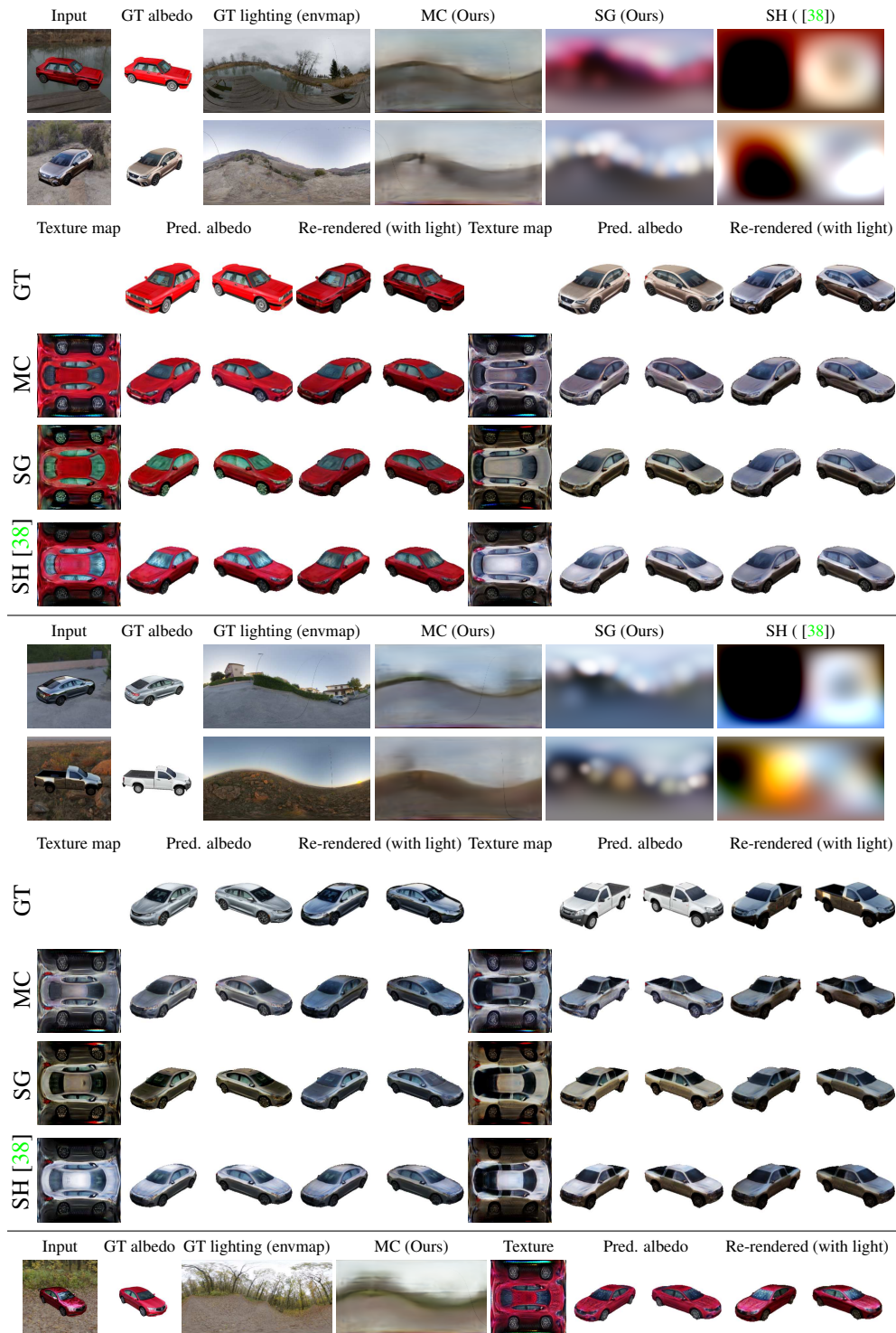


Figure 4.13: **Prediction on the Metallic Car Dataset ($\beta = 0$).** **Top and Middle:** We show more comparison results for 3 different shading methods. While all re-rendered images look almost identical, the underlying radiometric components (material and lighting) are different. A SH basis (DIB-R [38]) cannot recover the high frequency details of the sky light maps. In this case, MC performs best due to low variance in the estimator for mirror-like BRDFs. SGs can recover the overall form and contrast of the light map, but tend to predict incorrect texture maps incorporating the ground dominant color. **Bottom:** We show a failure case in the bottom row. MC shading fails when the environment map contains a lot of high frequency details. It reconstructs low frequency content but also merges the high frequency leaves into the texture.

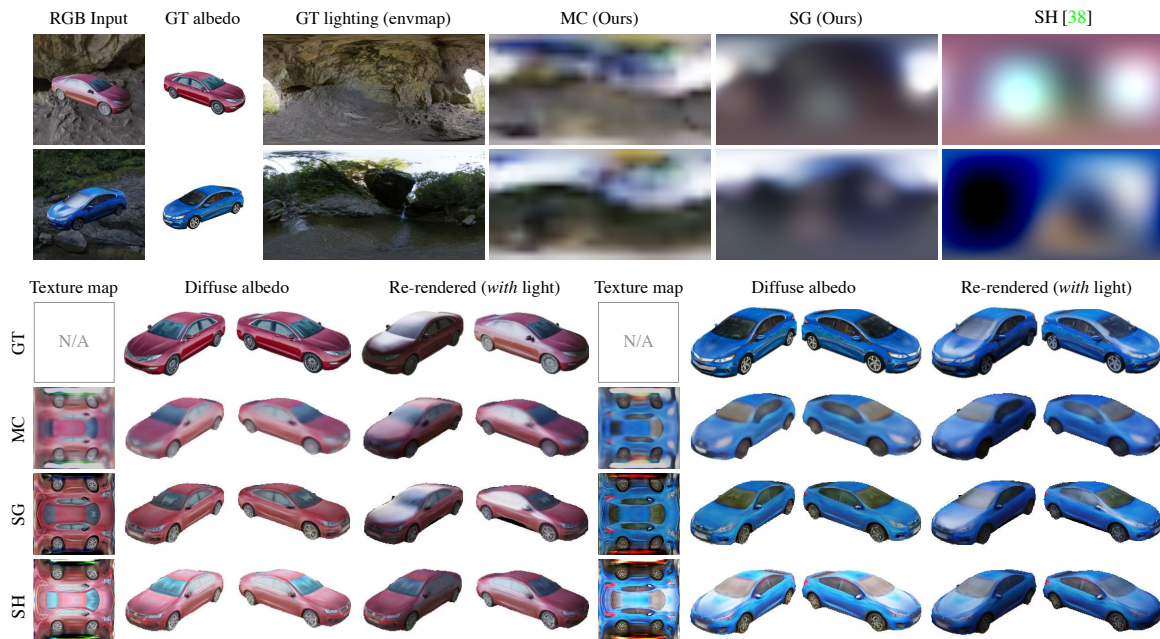


Figure 4.14: **Prediction on the Glossy Car Dataset** ($\beta > 0$). When the objects are glossy but not perfectly specular, SGs can correctly disentangle reflectance from lighting, as evidenced by the absence of white highlights in the predict albedos. DIB-R [38] cannot capture these bright regions due to a diffuse-only shading model, while MC oversmooths the predictions due to noisier pixel gradients. While all methods cannot completely reconstruct the environment map, our method can predict the correct dominant light location and sky color.

4.5.4 Glossy Surfaces

Experimental Settings We further apply our model to synthetic images rendered with positive roughness (glossy). To apply MC shading in such a case, we assume no prior knowledge for material and use a high sample count ($N = 1024$) to account for possibly low roughness images in the dataset. Due to high rendering time and memory cost, we subsample 4% of the pixels and apply \mathcal{L}_{im} to those pixels only in each training iteration. As such, we do not use the perceptual loss \mathcal{L}_{per} as it relies on the whole image. We predict a 32×16 environment map for lighting and predict global β and m for the specular BRDF. For SG shading, we use the same settings as for the metallic surfaces.

Experimental Results We apply both MC and SG shading and compare with DIB-R. Results are shown in Fig. 4.14 and Table 4.3 (Right). Qualitatively, SG shading has better lighting predictions with correct high-luminance regions. The specular highlights in the images successfully guide SG shading, while the bright reflection on the car window and front cover are fused with texture map in [38]. MC also has reasonable lighting predictions, but the predicted light map lacks structure due to weak surface reflections. Without a perceptual loss term, the predicted textures also tend to be blurrier.

Quantitatively, SG shading significantly outperforms DIB-R [38] on lighting prediction in terms of NCC (0.078 vs. 0.127) and improves on texture prediction in terms of

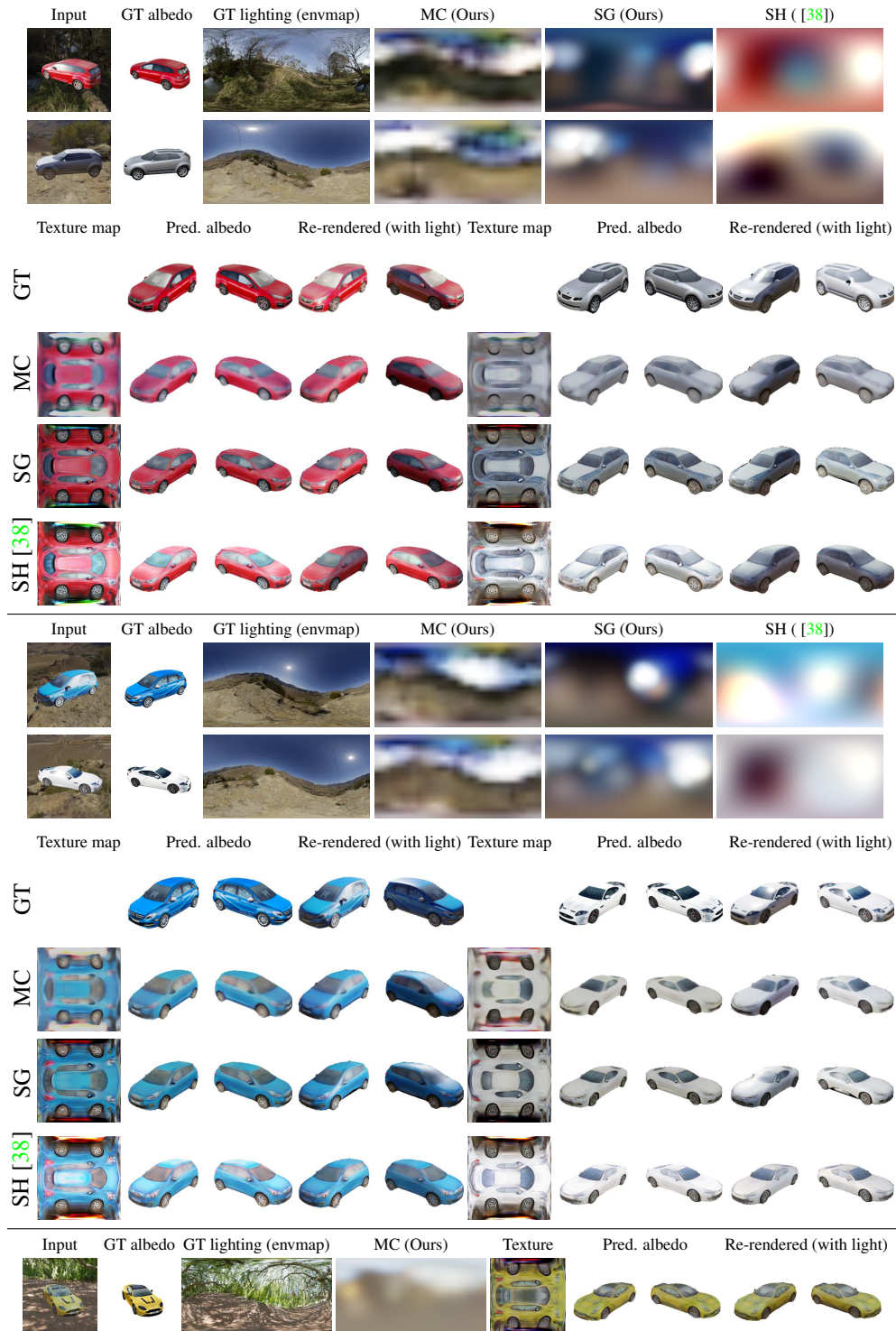


Figure 4.15: **Prediction on the Glossy Car Dataset ($\beta > 0$).** **Top and Middle:** We show more comparisons for three different shading methods. When the objects are glossy but not perfectly specular, SGs can correctly disentangle reflectance from lighting, as evidenced by the absence of white highlights in the predict albedos. DIB-R [38] cannot capture these bright regions due to a diffuse-only shading model, while MC oversmooths the predictions due to noisier pixel gradients. While all methods cannot completely reconstruct the environment map, our method can predict the correct dominant light location and sky color. **Bottom Row:** We show a failure case in the bottom row. All the methods fail when the texture contains a lot of high frequency details. We show SG only reconstructs low frequency content.

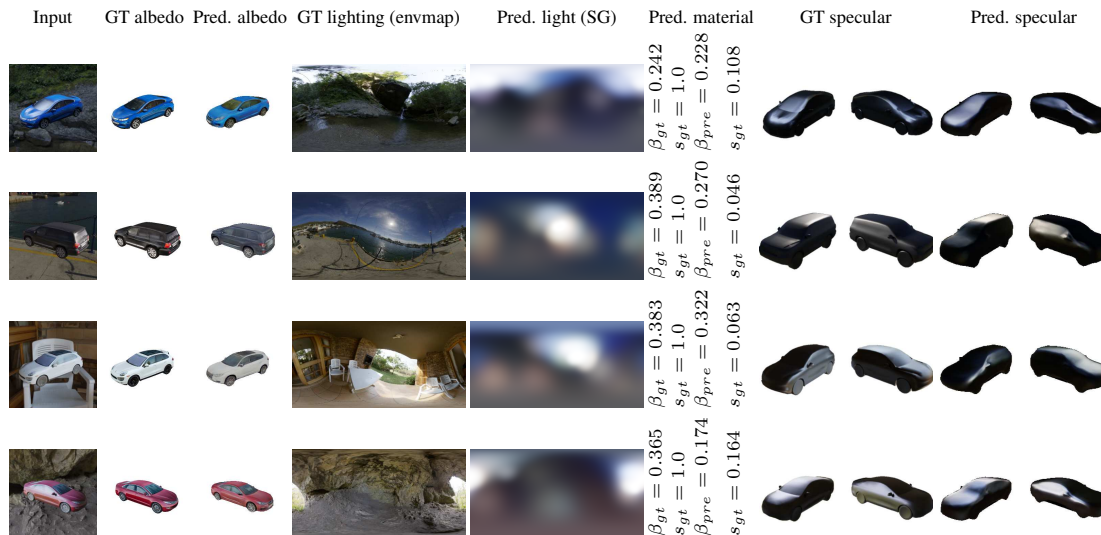


Figure 4.16: **Specular effect of SG Shading.** We decompose the prediction into diffuse albedo, light and material parameters, then visualize the specular contribution in other views and compare with GT (last 4 columns). Although the predicted material parameters do not match GT, interestingly, the specular renderings align well with each other.

BRDF/lighting disentanglement. We also compare with MC shading, where MC achieves slightly worse results on lighting predictions compared to SG, but is still much better than DIB-R. More results are shown in Fig. 4.15.

Failure Cases We also provide failure cases in the last row of Fig. 4.13 and Fig. 4.15. We find that when the GT environment map or the base texture map contains high frequency patterns, the problem becomes harder and our neural networks can only reconstruct low frequency content. This problem might be addressed with recent advances of position encoding and we leave it for future work.

4.5.5 Shape & Material Evaluation

Shape Evaluation 2D IoU loss measures the projection of the predicted shapes. As shown in Tbl. 4.3, the scores of all the methods are pretty close, which indicates DIB-R and DIB-R++ have similar performance for the geometry. We also evaluate the Chamfer distance between the predictions and GT meshes (normalized as one) in our glossy dataset predictions, where DIB-R++ is 0.036 while DIB-R is 0.037. It further demonstrates the both methods can recover good geometry.

Material Evaluation We also check the predicted surface reflectance materials and GT and found little correlation. We then decompose the prediction into the albedo map, lighting and material parameters, and display the specular effect in other views in Fig. 4.16. Compared to GT, we find the specular effects align very well with each other, even though the predicted

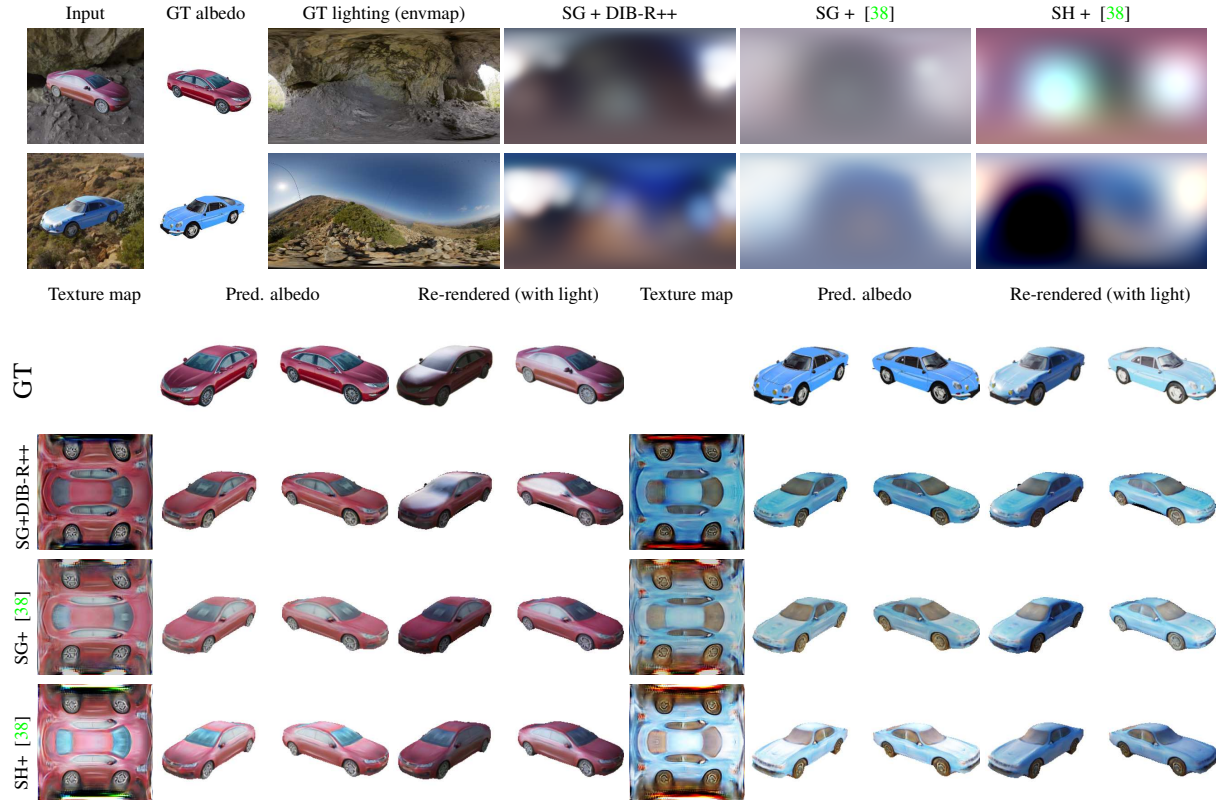


Figure 4.17: **Ablation Study of Shading Methods.** Even though we apply the same lighting representation to DIB-R [38], due to the lack of specular transport support, [38] cannot perfectly disentangle the specular light from diffuse labedo. Its window and front cover still “bake in” white specular light. Moreover, the lighting is also blurry, as opposed to DIB-R++. This demonstrates the necessity of explicitly modelling advanced lighting effect in the renderer.

material parameters has little correlation with GT. It might indicates the material and light disentanglement may have more than one solution, as also evidenced in Fig. 4.8.

4.5.6 Ablation Study

In this section, we provide ablation study to evaluate the performance of our DIB-R++. We first analyze the choice of rasterization renderer, then compare SH [38] and SG shading, then explore the effects of numbers of SG components. We also ablate the influence of all the loss terms. Finally, we run our model three times and report the training variance to show that our method is insensitive to random seed.

Choice of Differentiable Rasterizer Our DIB-R++ rendering framework is quite flexible. In the first stage, it supports an arbitrary differentiable rasterization method, e.g. NVDiffRast [128] or DIB-R [38]. Compared to DIB-R, NVDiffRast is more efficient. However, the way NVDiffRast handles occlusion relies on anti-aliasing, while DIB-R uses soft rasterization. Theoretically, anti-aliasing only influences edge pixels while soft rasterization computes

the soft probability of all pixels, providing more gradient signals in shape deformation. We experimentally found that the predicted shape from DIB-R is better than NVDiffRast so we chose DIB-R.

Comparison of SH and SG Shading DIB-R++ achieves much better disentanglement for reflective surfaces compared to DIB-R [38], since it can account for surface reflections while [38] only supports naive diffuse surfaces low-frequency lighting. We have shown quantitatively and qualitatively comparisons with [38] in Sec. 4.5.3 and Sec. 4.5.4, where [38] uses Spherical Harmonic (SH) lighting while we use Spherical Gaussian lighting. However, among these results, we use the default lighting for [38], which uses a 2-order SH basis and contains 27 parameters ($9 \times 3 = 27$, 9 for 2nd SH and 3 for RGB). On the other hand, DIB-R++ applies SG shading and 32 components SG light, which is composed of 224 parameters ($32 \times 7 = 224$). As such, DIB-R++ differs from [38] in two key aspects: shading method and light representations. Thus, we further apply the same SG lighting to [38] and compare with DIB-R++ again, where the only difference is the shading method to perform a more fair comparison.

We show the comparison results in Fig. 4.17 and Tbl. 4.4. After applying SG lighting to [38], it is still incapable of disentangling strong lighting from the base texture. As a result, the car’s window and front cover still “bake in” white specular light. Moreover, the lighting prediction of [38] is blurrier. Quantita-

tively, the lighting loss of [38] with SG is better than that of [38] with SH, probably due to more parameters (224 vs. 27). However, it is still worse than DIB-R++ with SG. This demonstrates the necessity of explicitly modeling advanced lighting effects in the renderer.

Shading	Glossy surfaces (\downarrow)			
	Image	2D IoU	Light	Tex.
DIB-R++ w/ SG	0.024	0.057	0.091	0.140
[38] w/ SG	0.023	0.063	0.098	0.147
[38] w/ SH	0.024	0.062	0.131	0.152

Table 4.4: Comparison of shading methods.

Number of SG Components We study the influence of the number of SG components K and show results in Fig. 4.18 and Tbl. 4.5. When SG components are less than 4, the representation ability is limited. Consequently, the predicted lighting only has one mode. It is also reflected in Tbl. 4.5: less components generally result in higher lighting prediction error.

Shading K SGs	Glossy surfaces (\downarrow)			
	Image	2D IoU	Light	Tex.
4	0.025	0.067	0.118	0.147
8	0.025	0.063	0.093	0.145
16	0.024	0.063	0.092	0.143
32	0.024	0.057	0.091	0.140

Table 4.5: Ablation study of number of SG components.

On the other side, when the SG components are larger than 16, the performance becomes similar (Tbl. 4.5, 16 vs. 32). However, more SG components have higher representation ability and predicted visually better results, as shown in Fig. 4.18. Thus, we generally

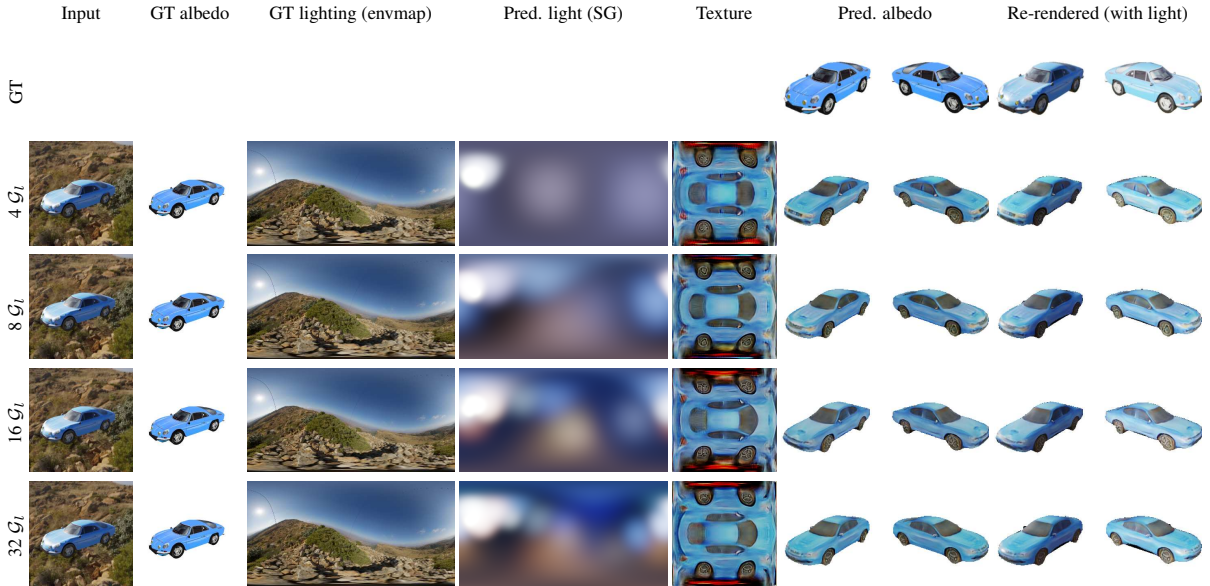


Figure 4.18: **Ablation Study of Number of SG Components.** As the number of SG components increase, the predicted lighting is more and more similar to the GT lighting and the color of predicted texture becomes more and more visually uniform, which indicates better disentanglement.

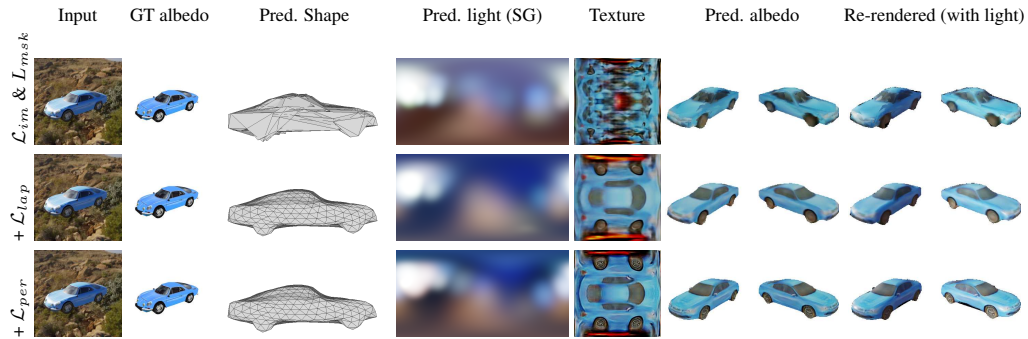


Figure 4.19: **Ablation Study of Loss Terms.** Similar to [291], we also find \mathcal{L}_{lap} regularizes the geometry to be smooth while \mathcal{L}_{per} helps create high-frequency, detailed textures.

choose $K = 32$ in the experiments to make full use of a 16GB memory GPU.

Ablation Study of Loss Terms We also ablate the loss terms in our method in Fig. 4.19 and get similar observations as in [291]. Specifically, we find \mathcal{L}_{lap} regularizes the geometry to be smooth while \mathcal{L}_{per} helps in creating high-frequency, detailed textures.

Training Variance Lastly, we train our model three times on glossy surfaces and report the performance in Tbl. 4.6. Our model converges well, and random seeds had little effect on performance. The models have very similar predicted results and close scores.

Shading	Glossy surfaces (\downarrow)			
	Image	2D IoU	Light	Tex.
DIB-R++ w. SG round 1	0.024	0.057	0.091	0.140
DIB-R++ w. SG round 2	0.025	0.061	0.092	0.143
DIB-R++ w. SG round 3	0.025	0.062	0.091	0.143
Mean	0.025	0.060	0.091	0.142
Variance	0.000	0.002	0.000	0.001

Table 4.6: Training variance.

The variance is small, which demonstrates the robustness of our model.

4.6 Summary

As shown in our previous two experiments, Monte Carlo shading works best under a metallic assumption ($\beta = 0$), in which case the rendered images can have rich details at low sample count ($N \leq 4$). However, when the surface is more Lambertian (i.e., when β is becoming larger), we have to compensate with a larger N to produce noise-free renderings, which impacts learning both time- and memory-wise. As a consequence, we recommend applying MC shading to metallic surfaces only, and default to SGs otherwise.

Our spherical Gaussian shading pipeline provides an analytic formulation for estimating the rendering equation, which avoids the need of tracing ray samples, largely accelerating the rendering process. While SGs can be blurry on metallic surfaces, in most case (e.g., when $\beta \geq 0.2$) it can model similar rendering effects at a fraction of the cost, achieving better results than MC shading and [38].

After inspecting the predicted surface material properties (β, s, m) and diffuse albedo with the ground-truth parameters in Blender, we find the materials contain little correlation and the intensities of diffuse albedo might change. As for SG, we are using only 32 basis elements to simulate a complex, high definition environment map (2K). Since SGs can only represent a finite amount of details, we find the predicted global β tends to be too small. One hypothesis for this is that the optimizer artificially prefers more reflections (and thus lower roughness) to be able to estimate at least some portions of the environment map. On the other hand, in MC, due to the absence of a perceptual loss, the predicted texture is too blurry and cannot represent GT to a high detail. We find that the predicted β and m do not have strong correlation with the GT material. Lastly, we note that the predicted texture map has to change its overall intensity to accommodate for other parameters to ensure the re-rendered images are correct, which leads to some differences with GT. More analysis can be found in Sec. 4.3.6 and Sec. 4.5.5.

In summary, when we have no prior knowledge about the material, our re-rendered images can be very close to the input images but the predicted material parameters are not always aligned with the GT materials. We believe this problem can be relieved by incorporating additional local constraints, e.g., part-based material priors, or by leveraging anisotropic SGs [276]. For instance, a car body is metallic while its wheels are typically diffuse; predicting different parameters for each region has the potential of improving disentanglement and interpretability. We leave this as future work.

Chapter 5

Singe-view Real Imagery 3D Object Reconstruction

Differentiable rendering has paved the way to training neural networks to perform “inverse graphics” tasks such as single-view 3D object reconstruction. To train high performant models, most of the current approaches rely on multi-view imagery, which are not readily available in existing real datasets. Applying renderings of synthetic CAD models often leads to a gap in performance when tested on real photographs. In this chapter, we aim to extract 3D knowledge from real imagery by utilizing differentiable renderers. We observe that recent Generative Adversarial Networks (GANs) not only synthesize photorealistic images, but also acquire 3D knowledge implicitly during training: object viewpoints can be adjusted by simply manipulating the latent codes. Therefore, we propose to exploit GANs as a multi-view data generator, synthesizing infinite multi-view photorealistic images to train inverse graphics networks using off-the-shelf differentiable renderers. We show that our approach achieves exquisite 3D reconstruction effects for real images. It significantly outperforms state-of-the-art inverse graphics networks trained on existing datasets, both quantitatively and via user studies. By incorporating more expressive rendering equations, it further acquires faithful material and lighting disentanglement, enabling artistic applications including material editing and relighting.

5.1 Introduction

The ability to infer 3D properties such as geometry, texture, material, and light from photographs is key in many domains such as AR/VR, robotics, architecture, and computer vision. Interest in this problem has been explosive, particularly in the past few years, as evidenced by

a large body of published works and several released 3D libraries (TensorflowGraphics [251], Kaolin [102], PyTorch3D [208]).

The process of going from images to 3D is often called “inverse graphics”, since the problem is inverse to the process of rendering in graphics in which a 3D scene is projected onto an image by taking into account the geometry and material properties of objects, and light sources present in the scene. Supervised works [260, 168, 70, 263, 42] require 3D labels during training, which are very expensive to acquire. Recent works have explored an alternative way to train inverse graphics networks that sidesteps the need for 3D ground-truth during training. The main idea is to make graphics renderers differentiable which allows one to infer 3D properties directly from images using gradient based optimization, [122, 154, 152, 38]. These methods employ a neural network to predict geometry, material, and light from images, by minimizing the difference between the input image with the image rendered from these properties.

While impressive results have been obtained in [154, 231, 153, 89, 38, 39, 116], most of these works still require some form of implicit 3D supervision such as multi-view images of the same object with known cameras. Thus, most results have been reported on the synthetic ShapeNet dataset [33], or the Pascal3D dataset [272] annotated with keypoints from which cameras can be accurately computed using structure-from-motion techniques. However, neither of them are perfect. The former brings a domain gap whereas the latter contains too few single-view images to train generalizable, high performant models. As a result, they still struggle with real photographs.

In this chapter, we aim to extract 3D knowledge from *real imagery* by utilizing differentiable graphics renderers. We observe that Generative Adversarial Networks (GANs) have shown the ability of generating images of high photorealism. Recent work [119] further demonstrates it is able to learn 3D information implicitly, e.g., by manipulating the latent code, it can produce images of the same scene from a different viewpoint. As such, we propose to exploit a GAN, specifically StyleGAN [119], as a generator to synthesize multi-view photorealistic images, building a dataset containing infinite amount of data which is suitable for inverse graphics tasks.

We create a large scale dataset, containing more than 10 times more images compared to the existing Pascal3D dataset. We then employ it to train inverse graphics neural networks using differentiable renderers [38, 39]. Our approach presents exquisite 3D reconstruction from real imagery, significantly outperforming inverse graphics networks trained on existing datasets. By incorporating more expressive rendering equations [39], we also demonstrate faithful material and lighting disentanglement and showcase several artistic applications

including material editing and relighting.

5.2 Related Work

3D from 2D Reconstructing 3D objects from 2D images is one of the mainstream problems in 3D computer vision. We here focus our review to single-image 3D reconstruction which is the domain of our work. Most of the existing approaches train neural networks to predict 3D shapes from images by utilizing 3D labels during training, [260, 168, 42, 192]. However, the need for 3D training data limits these methods to the use of synthetic datasets. When tested on real imagery there is a noticeable performance gap.

Newer works propose to differentiate through the traditional rendering process in the training loop of neural networks, [157, 122, 154, 38, 39]. Differentiable renderers allow one to infer 3D from 2D images without requiring 3D ground-truth. However, in order to make these methods work in practice, several additional losses are utilized in learning, such as the multi-view consistency loss whereby the cameras are assumed known. Impressive reconstruction results have been obtained on the synthetic ShapeNet dataset. While CMR by [116] and DIB-R by [38] show real-image 3D reconstructions on CUB [266] and Pascal3D [272] datasets, they rely on manually annotated keypoints, while still failing to produce accurate results.

A few recent works, [269, 145, 64, 121], explore 3D reconstruction from 2D images in a completely unsupervised fashion. They recover both 3D shapes and camera viewpoints from 2D images by minimizing the difference between original and re-projected images with additional unsupervised constraints, e.g., semantic information ([145]), symmetry ([269]), GAN loss ([121]) or viewpoint distribution ([64]). Their reconstruction is typically limited to 2.5D ([269]), and produces lower quality results than when additional supervision is used ([64, 145, 121]). In contrast, we utilize GANs to generate multi-view realistic datasets that can be annotated *extremely efficiently*, which leads to more accurate 3D results.

GAN Manipulation GANs [66, 119] take a latent code as input and synthesize an image. However, the latent code is sampled from a predefined prior and lacks interpretability. Exploring how to manipulate the latent code to generate an image with desired properties has been widely explored [132, 151, 195]. Pioneering work is InfoGAN [41], which tries to maximize the mutual information between the prior and the generated image distribution. [244] transfers face rigging information from an existing model to control face attribute disentanglement in the StyleGAN latent space. [225] aims to find the latent space vectors that correspond to meaningful edits, while [82] exploits PCA to disentangle the latent space.

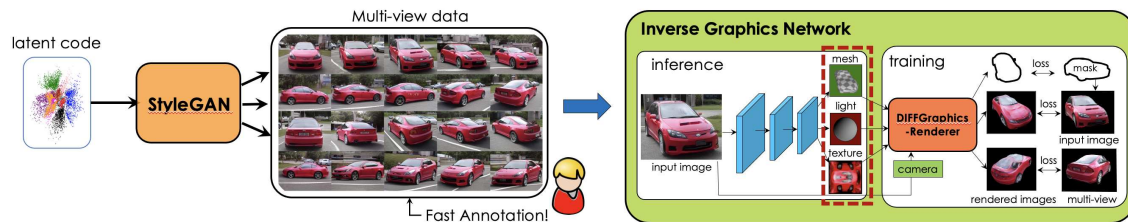


Figure 5.1: **Overview.** We exploit StyleGAN as a synthetic data generator, and label the generated data in an extremely efficient way. This “dataset” is used to train an inverse graphics network that predicts 3D properties from single-view images.

In our work, we manipulate the latent space to synthesize multi-view images suitable for training inverse graphics techniques.

5.3 Our Approach

We start by providing an overview of our approach (Fig. 5.1), and describe the individual components in more detail in the following subsections. Specifically, we leverage the fact that the recent state-of-the-art GAN architecture StyleGAN [119] learns to produce highly realistic images of objects, and allows for a reliable control over the camera. We manually select a few camera views that covers all common viewpoints of objects, and use StyleGAN to generate a large number of examples per view, which we explain in Sec. 5.3.1. While training inverse graphics models typically requires known camera poses for images, in Sec. 5.3.2 we provide an efficient way to annotate camera poses for the created StyleGAN dataset. In Sec. 5.3.3, we exploit this dataset to train inverse graphics networks utilizing the state-of-the-art differentiable renderers, DIB-R and DIB-R++ [38, 39]. We show our created StyleGAN dataset significantly improves real imagery 3D reconstruction results. Incorporating expressive rendering equations [39] further allows us to recover advanced lighting and surface material from real images, which enables many artistic applications including material editing and relighting.

5.3.1 StyleGAN as Synthetic Data Generator

We first aim to utilize StyleGAN to generate multi-view imagery. StyleGAN is a 16 layers neural network that maps a latent code $z \in Z$ drawn from a normal distribution into a realistic image. The code z is first mapped to an intermediate latent code $w \in W$ which is transformed to $w^* = (w_1^*, w_2^*, \dots, w_{16}^*) \in W^*$ through 16 learned affine transformations. We call W^* the transformed latent space to differentiate it from the intermediate latent space W . Transformed latent codes w^* are then injected as the style information to the StyleGAN Synthesis network.

Different layers control different image attributes. As observed in [119], styles in early

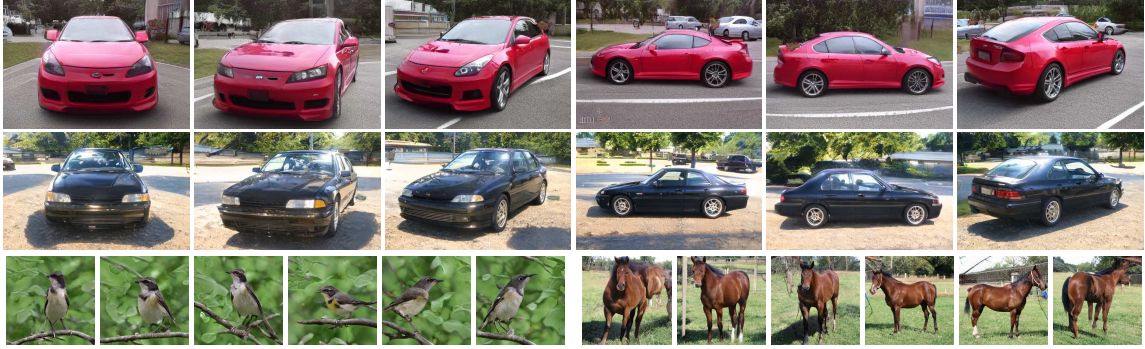


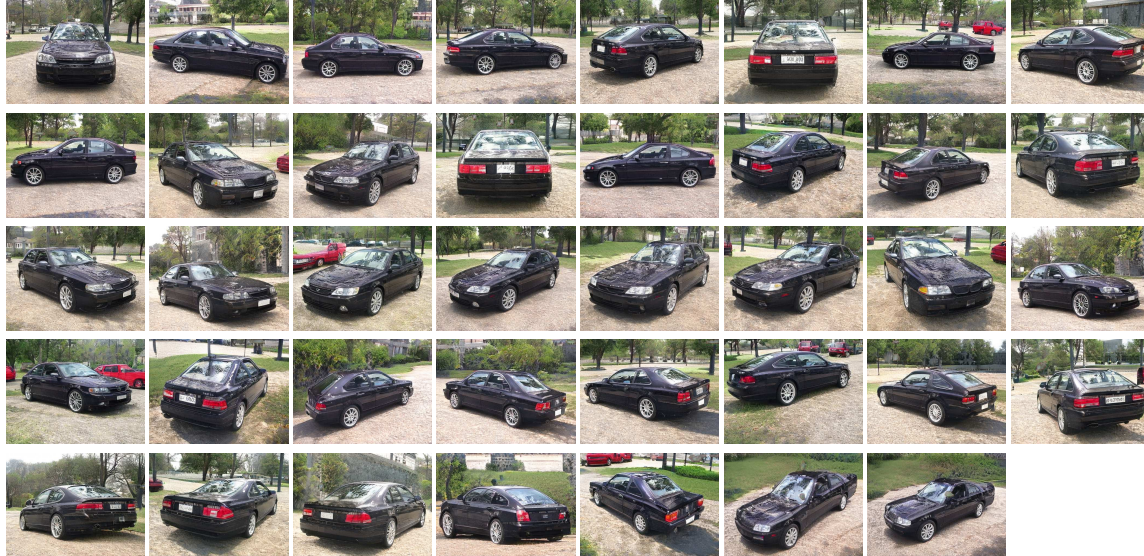
Figure 5.2: **StyleGAN Viewpoint & Content Disentanglement.** We show examples of cars (first two rows) synthesized in chosen viewpoints (columns). To get these, we fix the latent code w_v^* that controls the viewpoint (one code per column) and randomly sample the remaining dimensions of (Style)GAN’s latent code (to get rows). Notice how well aligned the two cars are in each column. In the third row we show the same approach applied to horse and bird StyleGAN.

layers adjust the camera viewpoint while styles in the intermediate and higher layers influence shape, texture and background. We empirically find that the latent code $w_v^* := (w_1^*, w_2^*, w_3^*, w_4^*)$ in the first 4 layers controls camera viewpoints. That is, if we sample a new code w_v^* but keep the remaining dimensions of w^* fixed (which we call the content code), we generate images of the same object depicted in a different viewpoint. Examples are shown in Fig. 5.2.

We further observe that a sampled code w_v^* in fact represents a fixed camera viewpoint. That is, if we keep w_v^* fixed but sample the remaining dimensions of w^* , StyleGAN produces imagery of different objects in the same camera viewpoint. This is shown in columns in Fig. 5.2. Notice how aligned the objects are in each of the viewpoints. This makes StyleGAN a *multi-view* data generator!

“StyleGAN” multi-view dataset We manually select several views, which cover all the common viewpoints of an object ranging from 0-360 in azimuth and roughly 0-30 in elevation. We pay attention to choosing viewpoints in which the objects look most consistent. We show all of our selected viewpoints in Fig. 5.3. Specifically, we demonstrate on car, bird and horse StyleGAN models. The created car, bird and horse training datasets contain 39, 22 and 8 views, respectively. We find that these views are sufficient to learn accurate 3D inverse graphics networks. Notice the high consistency of both the car shape and texture as well as the background scene across the different viewpoints.

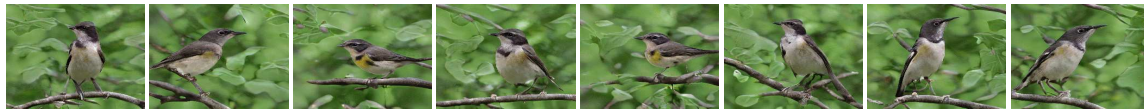
We further show examples from our StyleGAN-generated dataset in Fig. 5.4. We sample w^* such that our dataset contains objects with various shapes, textures and viewpoints. In particular, in the first six rows, one can notice a diverse variants of car types (Standard Car, SUV, Sports car, Antique Car, etc) . We find that StyleGAN can also produce rare car shapes like trucks, but with a lower probability.



Car Viewpoints



Horse Viewpoints



Bird Viewpoints

Figure 5.3: **All Viewpoints.** We show an example of a car, a bird and a horse synthesized in all of our chosen viewpoints. While shape and texture are not perfectly consistent across views, they are sufficiently accurate to enable training accurate inverse graphics networks in our downstream tasks. Horses and birds are especially challenging due to articulation. One can notice small changes in articulation across viewpoints. Dealing with articulated objects is subject to future work.

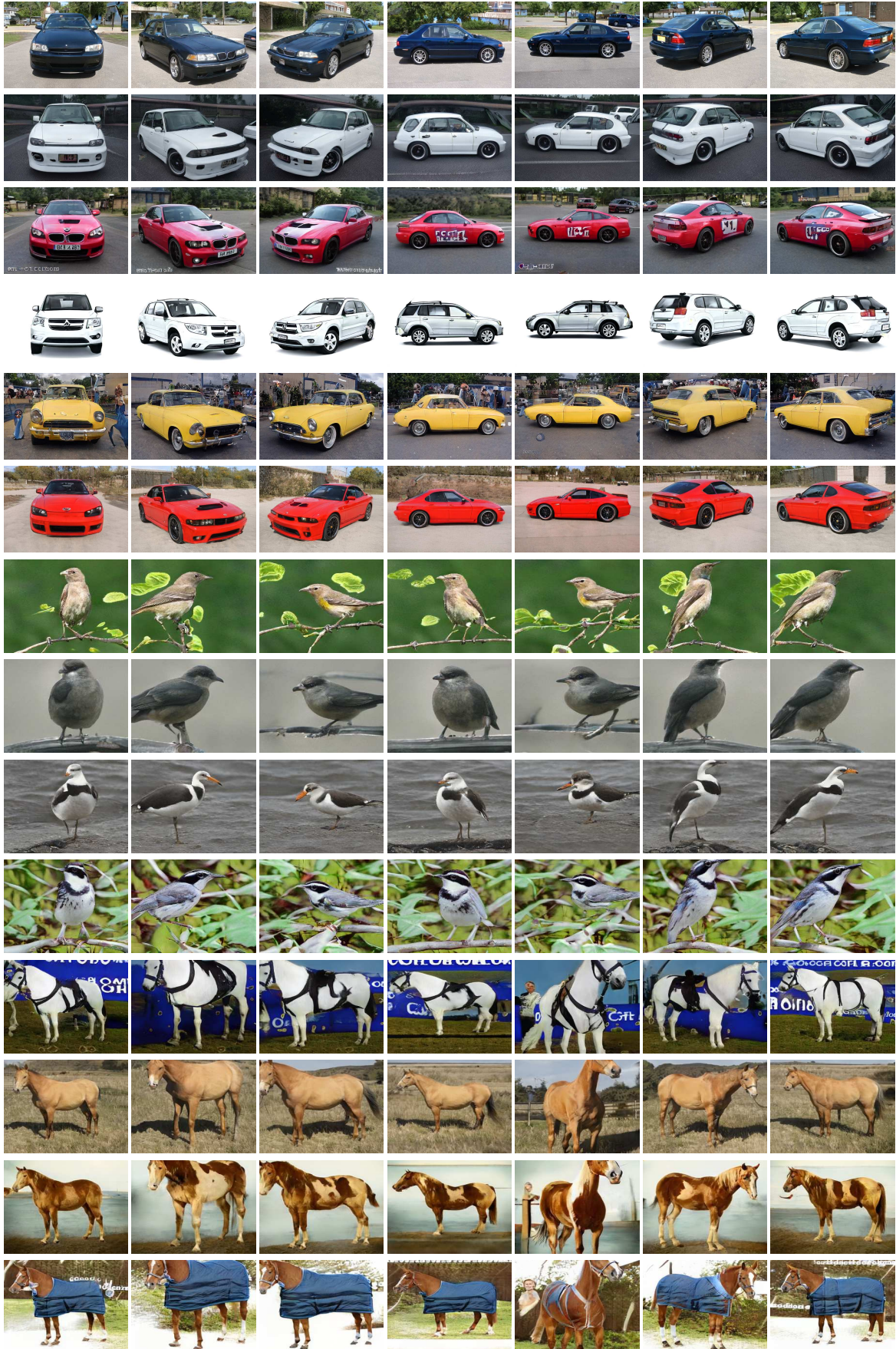


Figure 5.4: **Dataset Overview.** We synthesize multi-view datasets for three classes: *car*, *horse*, and *bird*. Our datasets contain objects with various shapes, textures and viewpoints. Notice the consistency of pose of object in each column (for each class). Challenges include the fact that for all of these objects StyleGAN has not learned to synthesize views that overlook the object from above due to the photographer bias in the original dataset that StyleGAN was trained on.

Segmentation Since differentiable renderers also utilize segmentation masks during training, we further apply MaskRCNN [84] to get instance segmentation in our generated dataset. As StyleGAN sometimes generates unrealistic images or images with multiple objects, we filter out “bad” images which have more than one instance, or small masks (less than 10% of the whole image area).

Discussion StyleGAN datasets have two limitations. First, we could not find views that would depict the object from a higher up camera, i.e., a viewpoint from which the roof of the car or the back of the horse would be more clearly visible. This is mainly due to the original dataset on which StyleGAN was trained on, which lacked such views. This leads to challenges in training inverse graphics networks to accurately predict the top of the objects.

Secondly, for articulated objects such as the horse and bird classes, StyleGAN does not perfectly preserve object articulation in different viewpoints, which leads to challenges in training high accuracy models using multi-view consistency loss. We leave further investigation of articulated objects to future work.

5.3.2 Camera Initialization

Inverse graphics tasks require camera pose information during training, which is challenging to acquire for real imagery. Pose is generally obtained by annotating keypoints for each object and running structure-from-motion (SfM) techniques [266, 272] to compute camera parameters. However, keypoint annotation is quite time consuming – requiring roughly 3-5minutes per object which we verify in practice using the LabelMe interface [213]. In our work, we utilize StyleGAN to significantly reduce annotation effort since samples with the same w_v^* share the same viewpoint. Therefore, we only need to assign a few selected w_v^* into camera poses, instead of annotating all the images.

Although StyleGAN datasets largely reduce the amount of images that require labelling, annotation of keypoints is still time-consuming. We further propose a more efficient way to accelerate it. Specifically, instead of accurate keypoints annotation, we annotate the chosen viewpoint codes with a rough absolute camera pose. To be specific, we classify each viewpoint code into one of 12 azimuth angle bins, uniformly sampled along 360 deg. We assign each code a fixed elevation (0°) and camera distance. These camera poses provide a very coarse annotation of the actual pose – the annotation serves as the initialization of the camera which we will optimize during training. This allows us to annotate all views (and thus the entire dataset) in **only 1 minute** – making annotation effort negligible. We call our method *LazyInit*.



Figure 5.5: **Annotation Comparisons.** We show two different annotation methods: LazyInit and SfM. Top row shows the classified pose bin annotations, while the images show the annotated keypoints. For 39 viewpoints car class, annotating pose bins took 1 min, while keypoint annotation took 3-4 hours. We empirically find that pose bin annotation is sufficient in training accurate inverse graphics networks (when optimizing camera parameters during training in addition to optimizing the network parameters).

Discussion We visualize our two different annotation types in Fig 5.5. We show annotated bins in the top. We also show the annotated keypoints for the (synthesized) car example in the image, based on which we can compute camera poses using SfM. LazyInit provides very coarse annotations while SfM results in more accurate camera poses. However, it is still noisy due to the view inconsistency in StyleGAN images. When training inverse graphics models we will propagate gradients to camera poses as well to improve the pose annotation in each view. In practice, we find starting from either SfM or LazyInit, the converged camera poses are very similar to each other. Since LazyInit requires much less time, we adopt it in all our inverse graphics experiments. Comparison between two methods are shown in Sec. 5.4.2.

5.3.3 Training Inverse Graphics Neural Networks

Now we discuss how to train inverse graphics networks with StyleGAN dataset. We verify two rendering equations in training. We first employ the rasterization based renderer, DIB-R, to recover shape and texture from images. We further apply more expressive rendering equations, utilizing DIB-R++ to predict advanced lighting and material parameters from real images.

Shape and Texture Recovery Following CMR [116], we first adopt DIB-R as the differentiable graphics renderer, training a 3D prediction network to infer 3D shape and texture from images. We turn off lighting and surface material in this setting.

Let I denote an image captured in camera viewpoint \mathbf{v} from our StyleGAN dataset, and M is its corresponding object mask. The inverse graphics network F , parameterized by ϑ , makes a prediction as follows: $\{S, T\} = F(I; \vartheta)$, where S is the shape while T is the texture. To train the network, it takes the prediction as input and produces rendered image and mask $\{\tilde{I}, \tilde{M}\} = R(\mathbf{v}, S, T)$. We slightly modify the loss function in [38] to fit for the new dataset:

$$\mathcal{L} = \lambda_{\text{col}} \mathcal{L}_{\text{col}}(\tilde{I}, I) + \lambda_{\text{IOU}} \mathcal{L}_{\text{IOU}}(\tilde{M}, M) + \lambda_{\text{lap}} \mathcal{L}_{\text{lap}}(\boldsymbol{\pi}) + \lambda_{\text{per}} \mathcal{L}_{\text{per}}(\tilde{I}, I) + \lambda_{\text{mov}} \mathcal{L}_{\text{mov}}(S) \quad (5.1)$$

Here, \mathcal{L}_{col} is the standard L_1 image reconstruction loss defined in the RGB color space. \mathcal{L}_{IOU} computes the intersection-over-union between the ground-truth mask and the rendered mask. \mathcal{L}_{lap} is the commonly used regularization loss to ensure the shape is well behaved. \mathcal{L}_{per} is the perceptual loss that helps the predicted texture look more realistic. Finally, following CMR [116], we add \mathcal{L}_{mov} to further regularize the shape deformation to be uniform and small. Its key idea is to learn a mean shape, where the final shape is equal to the mean shape plus predicted deformation. \mathcal{L}_{mov} tries to minimize the difference between the mean shape and final shape, which helps stabilize the shape prediction.

Training with Multi-view Consistency and Noisy Camera Multi-view consistency loss is typically used with the synthetic Shapenet dataset [33, 38, 39] but rarely adopted for real-image inverse graphics tasks due to the lack of available data. Thanks to StyleGAN dataset, we have access to multi-view images for each object so we apply it during training. We find that this loss is key in obtaining highly accurate results. In particular, we follow the multi-view loss defined in Eq. (3.19). While more views provide more constraints, empirically, two views have been proven sufficient. We randomly sample views for efficiency.

When we use the above loss functions, we will jointly train the neural network F and optimize viewpoint cameras (which were fixed in DIB-R). We assume that different images generated from the same w_v^* correspond to the same viewpoint. Optimizing the camera jointly with the weights of the network allows us to effectively deal with noisy initial camera annotations.

Predicting Advanced Lighting and Material StyleGAN have shown powerful ability to generate photorealistic images. The synthesized images contain not only delicate texture, but also exquisite lighting effects. By incorporating more expressive rendering equations, we can further infer advanced lighting and surface material parameters. We modify the network to jointly predict geometry π , lighting γ and SVBRDF θ ¹ from the input images. We then choose DIB-R++ to render the 3D properties back to images and adopt the same loss, minimizing the input images and rendered images to train the network.

While DIB-R can also be employed to predict lighting, we show the rasterization based renderer only supports low-frequency lighting, e.g., Spherical Harmonics, which fails to account for high order lighting effects. As a result, it sometimes “bakes in” lighting in texture, especially for the images with strong reflection or specular effects. In contrast, DIB-R++ supports more expressive rendering equations to represent secondary lighting

¹To distinguish two settings, we use $\{S, T\}$ to represent shape and texture in the first setting while $\{\pi, \gamma, \theta\}$ to represent shape, material and lighting in the second case. θ contains both surface albedo map and surface reflectance parameters.

effects, disentangling accurate material and faithful lighting which can further be used in material editing and relighting applications.

5.4 Experiments

In this section, we showcase our approach on inverse graphics tasks (single-view 3D object reconstruction). We first describe our StyleGAN dataset in Sec. 5.4.1, then show the results of shape and texture recovery in Sec. 5.4.2. Lastly, we exhibit lighting and material prediction as well as material editing and relighting application in Sec. 5.4.3.

5.4.1 StyleGAN Dataset

StyleGAN Models We choose three category-specific StyleGAN models, one representing a rigid object class, and two representing articulated (and thus more challenging) classes. We use the official car and horse model from StyleGAN2 [120] repo which are trained on LSUN Car and LSUN Horse with 5.7M and 2M images. We also train a bird model on NABirds [252] dataset, which contains 48k images.

Our “StyleGAN” Dataset We first randomly sample 6000 cars, 1000 horse and 1000 birds with diverse shapes, textures, and backgrounds from StyleGAN. After filtering out images with bad masks as described in Sec. 5.3.1, 55429 cars, 16392 horses and 7948 birds images remain in our dataset which is significant larger than the Pascal3D car dataset [272] (4175 car images). Note that nothing prevents us from synthesizing a significantly larger amount of data, but in practice, this amount turned out to be sufficient to train good models. We provide all views and car, bird and horse examples in Fig. 5.3 and Fig. 5.4.

5.4.2 Shape & Texture Recovery with DIB-R

Experimental Details We first train a inverse graphics network with DIB-R to predict shapes and texture maps from images. Our DIB-R based inverse graphics model was trained with Adam ([124]), with a learning rate of $1e-4$. We set λ_{IOU} , λ_{col} , λ_{lap} , and λ_{mov} to 3, 20, 5, and 2.5, respectively. We first train the model with L_{col} loss for 3K iterations, and then fine-tune the model by adding L_{percept} to make the texture more realistic. We set λ_{percept} to 0.5. The model converges in 200K iterations with batch size 16. Training takes around 120 hours on four V100 GPUs.

Results on StyleGAN Dataset We show 3D reconstruction results in Fig. 5.6. Notice the quality of the predicted shapes and textures, and the diversity of the 3D car shapes we obtain.

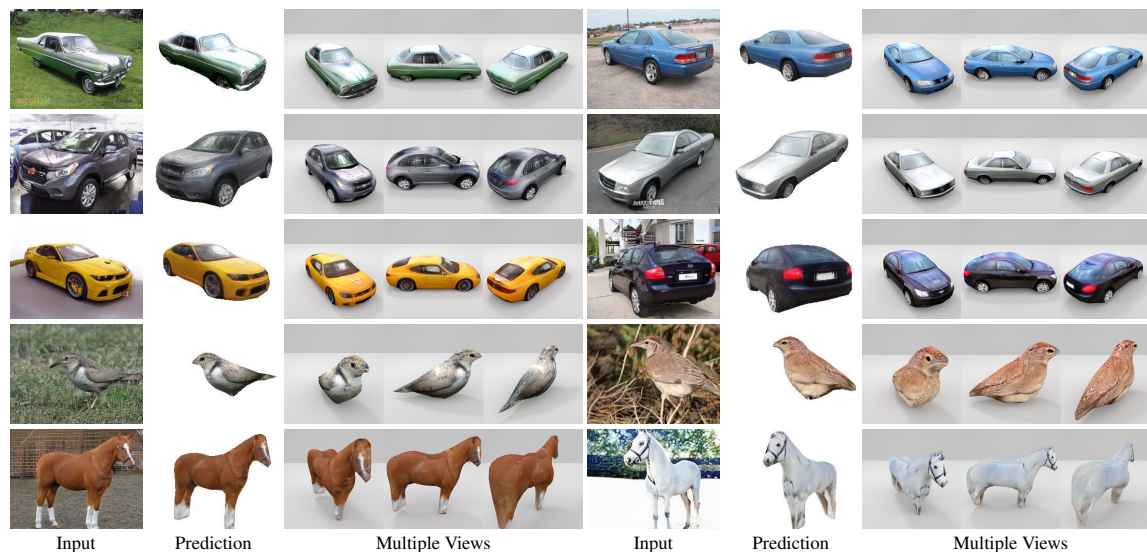


Figure 5.6: **Shape & Texture Reconstruction Results.** Given input images (1st column), we predict 3D shape, texture, and render them into the same viewpoint (2nd column). We also show renderings in 3 other views in remaining columns to showcase 3D quality. Our model is able to reconstruct cars with various shapes, textures and viewpoints. We also show the same approach on harder (articulated) objects, i.e., bird and horse.

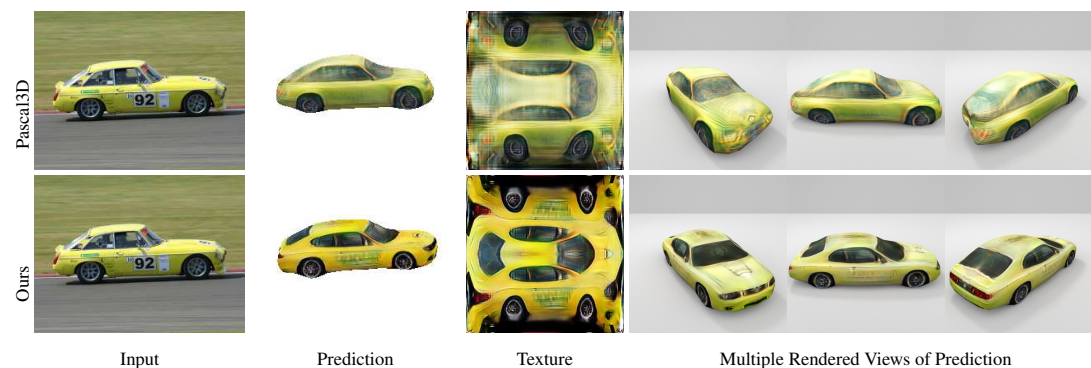


Figure 5.7: **Comparison on Pascal3D Testing Set.** We compare inverse graphics networks trained on Pascal3D and our StyleGAN dataset. Notice considerably higher quality of prediction when training on the StyleGAN dataset.

Our method also works well on more challenging (articulated) classes, e.g. horse and bird. We provide additional examples in Fig. 5.8.

Evaluation on Pascal3D Dataset We test our model on Pascal3D test set and show qualitative results in Fig. 5.7. Additional results are provided in Fig. 5.9. Note that the images from Pascal3D dataset are different from those our StyleGAN-model was trained on. Our model is trained on GAN images generated by StyleGAN [119], thanks to this powerful generative model, the distribution of GAN images is similar to the distribution of real images, allowing our model to generalize well.

To evaluate our dataset, we also train exactly the same inverse graphics model on the Pascal3D car dataset [272] and compare with the model trained on our StyleGAN dataset.

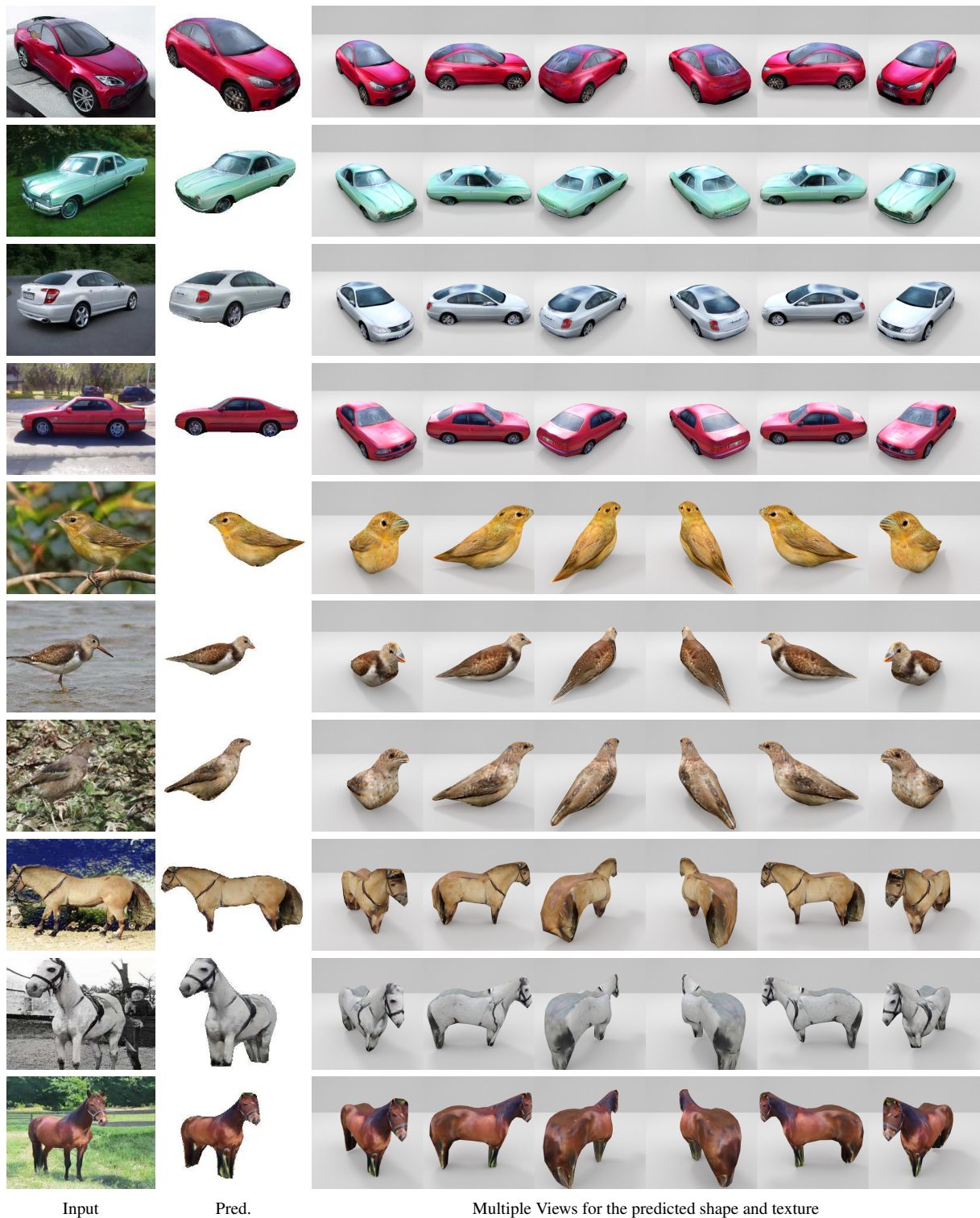


Figure 5.8: **3D Reconstruction Results for Car, Horse, and Bird Classes.** We show car, horse and bird examples tested on the images from the StyleGAN dataset test sets. Notice that the model struggles a little in reconstructing the top of the back of the horse, since such views are lacking in training.



Figure 5.9: **Comparison on PASCAL3D Imagery.** We compare PASCAL-model with StyleGAN-model on PASCAL3D test set. While the predictions from both models are visually good in the corresponding image view, the prediction from StyleGAN-model have much better shapes and textures as observed in other views.

Pascal3D dataset has annotated keypoints, which we utilize to train the baseline model, termed as Pascal3D-model. As shown in Fig. 5.7 and Fig. 5.9, although the Pascal3D-model’s prediction is visually good in the input image view, rendered predictions in other views are of noticeably lower quality than ours, which demonstrates that StyleGAN dataset helps recover 3D geometry and texture better than Pascal3D dataset.

Dataset	Size	Annotation	Model	Pascal3D test	StyleGAN test
Pascal3D	4K	200-350h	Pascal3D	0.80	0.81
StyleGAN	50K	~1min	Ours	0.76	0.95

(a) Dataset Comparison

(b) 2D IOU Evaluation

Table 5.1: (a): We compare dataset size and annotation time of Pascal3D with our StyleGAN dataset. (b): We evaluate re-projected 2D IOU score of our StyleGAN-model vs the baseline Pascal3D-model on the two datasets.

We also quantitatively evaluate the two models in Table 5.1 for the car class. We report the estimated annotation time in Table. 5.1 (a) to showcase efficiency behind our StyleGAN dataset. It takes 3-5 minutes to annotate keypoints for one object, which we empirically verify. Thus, labeling Pascal3D required around 200-350 hours while ours takes only 1 minute to annotate a 10 times larger dataset.

In Table 5.1 (b), we evaluate shape prediction quality by the re-projected 2D IOU score. Our model outperforms the Pascal3D-model on the SyleGAN test set while Pascal3D-model is better on the Pascal test set. This is not surprising since there is a domain gap between two datasets and thus each one performs best on their own test set. Note that this metric only evaluates quality of the prediction in input view and thus not reflect the actual quality of the predicted 3D shape/texture.

Human Study To analyze the quality of 3D prediction, we conduct an AMT user study on the *Pascal3D test set* which contains 220 images. We implement our user interface, visualized in in Fig. 5.10, on Amazon Mechanical Turk. For each example, we show the input image and predictions rendered in 6 views such that users can better judge the quality of 3D reconstruction. We show results for both our inverse graphics network (trained on the StyleGAN dataset) and the one trained on the Pascal3D dataset. We also compare shape reconstruction and textured models separately, such that users can judge the quality of both, shape and texture, more easily.

We randomize the order of ours vs baseline in each HIT to avoid any bias. We ask users to choose results that produce more realistic and representative shape, texture and overall quality with respect to the input image. We separate judgement of quality into these three categories to disentangle effects of 3D reconstruction from texture prediction. We also provide “no preference” options in case of ties. Our instructions emphasize that more

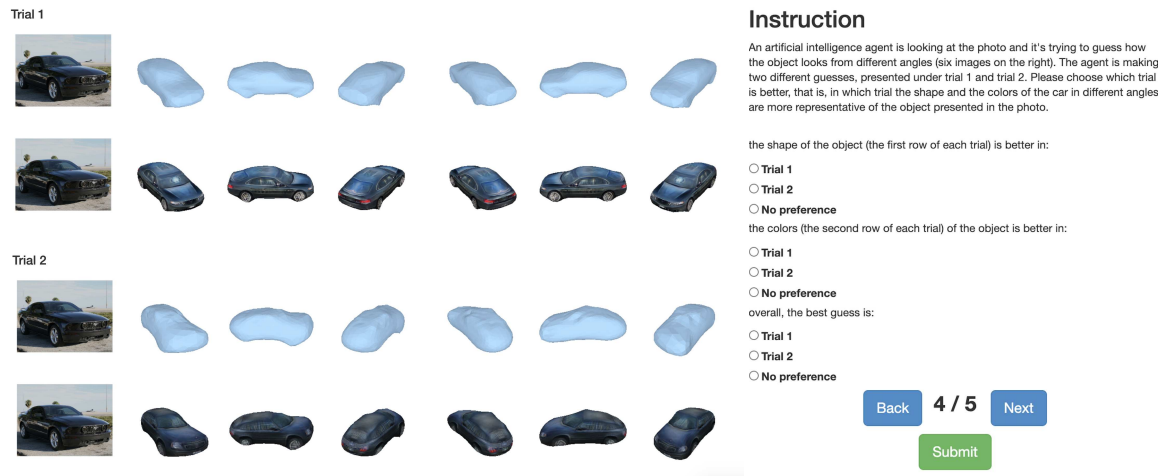


Figure 5.10: **User Study Interface (AMT)**. Predictions are rendered in 6 views and we ask users to choose the result with a more realistic shape and texture that is relevant to the input object. We compare both the baseline (trained on Pascal3D dataset) and ours (trained on StyleGAN dataset). We randomize their order in each HIT.

	Overall	Shape	Texture		Overall	Shape	Texture
Ours	57.5%	61.6%	56.3%	All Agree	26.1%	29.6%	27.1%
Pascal3D-model	25.9%	26.4%	32.8%	Two Agree	61.1%	58.6%	62.1%
No Preference	16.6%	11.9%	10.8%	No Agreement	12.8%	11.8%	10.8%

(a) 3D Quality Study

(b) Annotator Agreement

Table 5.2: User study results: (a): Quality of 3D estimation (shape, texture and overall). (b): Annotators agreement analysis. “No agreement” stands for the case where all three annotators choose different options.

“representative” results of the input should be selected, to avoid users being biased by good looking predictions that are not consistent with the input (e.g., such as in the case of overfit networks).

We evaluate the two networks on all 220 images from the Pascal3D test set (which are “in-domain” for the Pascal3D-trained network). For each image we ask three users to perform evaluation, which results in 660 votes in total. We report the average of all votes as our final metric and report annotator agreement analysis in Table 5.2. Users show significant preference of our results versus the baseline, which confirms that the quality of our 3D estimation. Moreover, for shape, texture, and overall evaluation, there are 88.2%, 89.2%, and 87.2% cases where at least two out of three users choose the same option.

Ablation Study In Fig 5.11 we ablate the importance of using multiple views in our dataset, i.e., by encouraging multi-view consistency loss during training. We compare predictions from inverse graphics networks trained with and without the losses and find significant differences in quality. Absence of multi-view consistency loss results in strong degradation in new views. This further indicates the superiority of our StyleGAN dataset compared

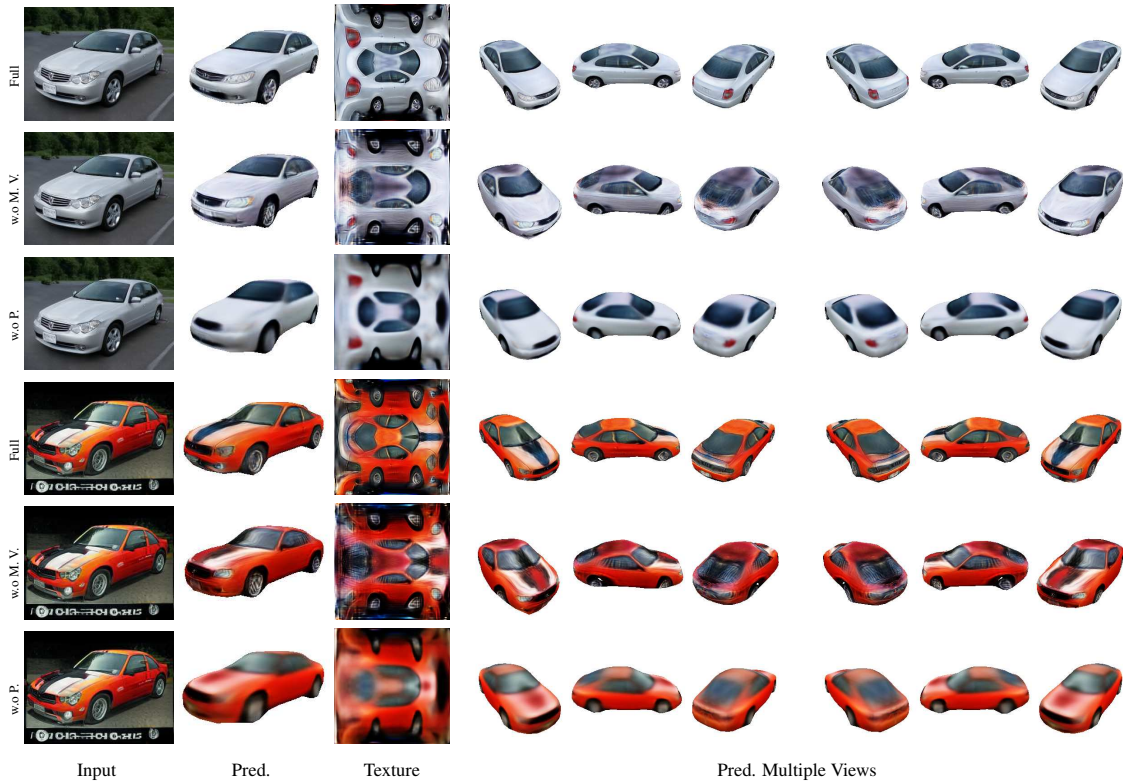


Figure 5.11: **Ablation Study.** We ablate the use of multi-view consistency and perceptual losses by showing results of 3D predictions. Clearly, the texture becomes worse in the invisible part if we remove the multi-view consistency loss (rows 2, 5, denoted by “wo M. V.”, which denotes that no multi-view consistency was used during training), showcasing the importance of our StyleGAN-multiview dataset. Moreover, the textures become quite smooth and lose details if we do not use the perceptual loss (rows 3, 6, noted by “wo P.”, which denotes that no perceptual loss was used during training).

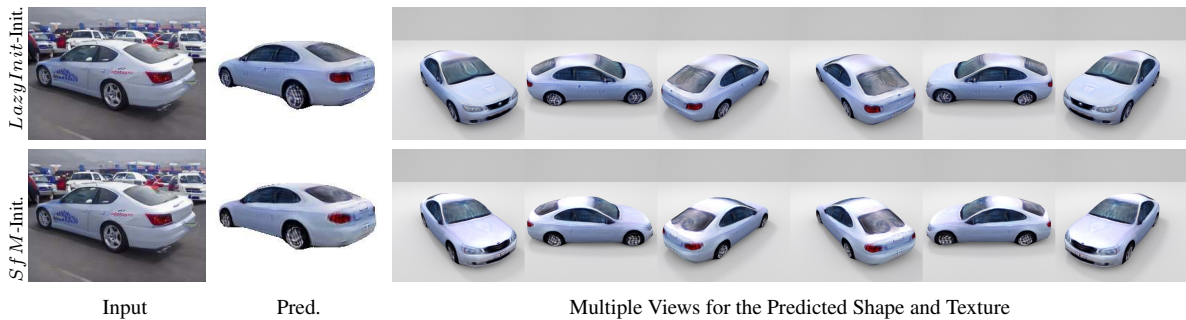


Figure 5.12: **Comparison of Different Camera Initialization Methods.** The first row shows predictions from *SfM*-Initialization (cameras computed by running SFM on annotated keypoints) and the second row show results obtained by training with *LazyInit*-Initialization (cameras are coarsely annotated into 12 view bins). Notice how close the two predictions are, indicating that coarse viewpoint annotation is sufficient for training accurate inverse graphics networks. Coarse viewpoint annotation can be done in 1 minute.

Annotation Type	Annotation Time	Training Time	2D IOU	Quaternion	Mean	Max
<i>SfM</i>	3-4h	60h	0.953	q_{xyz}	1.43°	2.95°
<i>LazyInit</i>	1min	60h	0.952	q_w	0.42°	1.11°

(a) Time & Performance

(b) Camera Difference after Training

Table 5.3: Comparison of different camera initialization methods. First table shows annotation time required for the StyleGAN dataset, and training times of the *LazyInit*-model and *SfM*-model on the dataset with respective annotations (binned viewpoints or cameras computed with SfM from annotated keypoints). The *LazyInit*-model requires significantly less annotation time, and its final performance is comparable to the *SfM*-model. Second table shows the difference of the camera parameters after training both methods (which optimize cameras during training). They converge to very similar camera positions. This shows that coarse view annotation along with camera optimization during training is sufficient in training high accuracy inverse graphics networks.

to Pascal3D dataset, which provides only single-view images and has strong performance degradation in new views. We also investigate the effects of perceptual loss and find it also play an import role in training. Clearly, it helps keep texture more realistic, where the models without it will results very blurry results.

We further conduct an ablation study to demonstrate the effectiveness of our *LazyInit* camera initialization. We train another inverse graphics network with a more *accurate* SfM camera initialization. Such an initialization is done by manually annotating object keypoints in each of the selected views (w_v^*) of a single car example, which takes about 3-4 hours (around 200 minutes, 39 views). Note that this is still a significantly lower annotation effort compared to 200-350 hours required to annotate keypoints for every single object in the Pascal3D dataset. We then compute the camera parameters using SfM. We refer to the two inverse graphics networks trained with different camera initializations as *LazyInit*-model and *SfM* -model, respectively.

We quantitatively evaluate two initialization methods in Table. 5.3. We first compare the annotation and training times. While it takes the same amount of time to train, *view*-model saves on annotation time. The performance of *view*-model and *keypoint* -model are comparable with almost the same 2D IOU re-projection score on the StyleGAN test set. Moreover, during training the two camera systems converge to the same position. We evaluate this by converting all the views into quaternions and compare the difference between the rotation axes and rotation angles. Among all views, the average difference of the rotation axes is only 1.43° and the rotation angle is 0.42°. The maximum difference of the rotation axes is only 2.95° and the rotation angle is 1.11°.

We further qualitatively compare the two methods in Fig. 5.12, showing that they perform very similarly. Both, qualitative and quantitative comparisons, demonstrated that *LazyInit*-camera initialization is sufficient for training accurate inverse graphics networks and no additional annotation is required. This demonstrates a scaleable way for creating multi-view datasets with StyleGAN, with roughly a minute of annotation time per class.



Figure 5.13: **3D Reconstruction Failure Cases.** We show examples of failure cases for car, bird and horse. Our method tends to fail to produce relevant shapes for objects with out-of-distribution shapes (or textures).

Failure Cases We find that our inverse graphics network fails on out-of-distribution images/shapes, as shown in Fig. 5.13. For example, the reconstruction results for Batmobile and Flinstone cars are not representative of the input cars. We anticipate that this issue can be addressed by augmenting the dataset on which StyleGAN is trained with more diverse objects. Part of the issue is also caused by GANs not capturing the tails of the distribution well, which is an active area of research.

5.4.3 Lighting & Material Prediction with DIB-R++

Experimental Settings We further apply DIB-R++ to learn to predict advanced lighting and materials from StyleGAN dataset. We train the inverse graphics models on the car images, which contain various lighting conditions, ranging from high specular paint to nearly diffuse. We apply SG shading and jointly predict shape π , material θ and lighting γ . Here, π is represented by a sphere topology and we predict 642 vertex movements. θ contains a 256×256 diffuse texture map and two global β and s . γ contains $K = 32$ SG bases. To evaluate the role of the more expressive rendering equations, we also run DIB-R as the baseline, predicting the same shape and texture but with Spherical Harmonic light on the same dataset by using the same training procedure.

Results on StyleGAN dataset We qualitatively evaluate DIB-R++ results and compare with DIB-R in Fig. 5.14. DIB-R++ reconstructs more faithful material and lighting components, producing an interpretable decomposition. Specifically, it can represent the dominant light

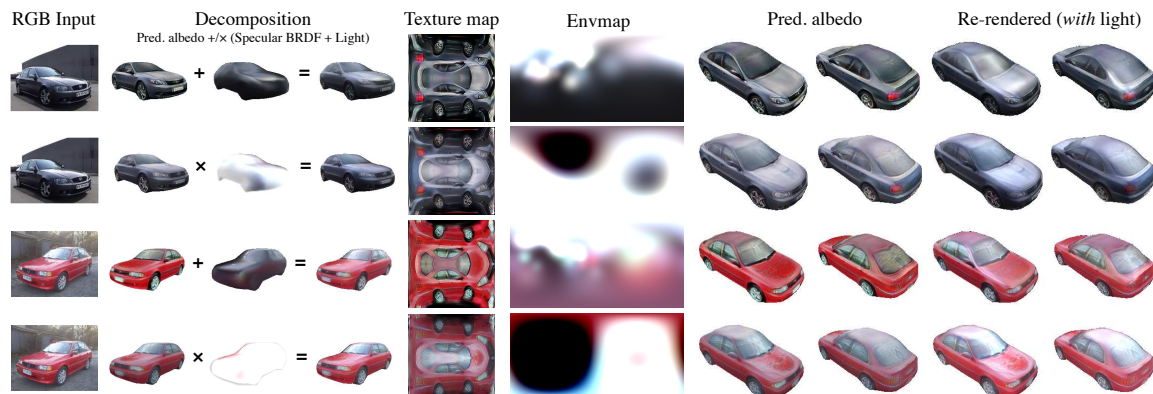


Figure 5.14: **Results on Real Imagery from the StyleGAN-generated Car Dataset.** In Row(1,3), we show DIB-R++ can recover a meaningful decomposition as opposed to DIB-R (Row(2,4)), as shown by cleaner texture maps and directional highlights (e.g., car windshield).

direction more accurately, while naive Spherical Harmonic shading from DIB-R tends to merge reflectance with lighting. More comparisons can be found in Fig. 5.17.

Evaluation on LSUN Dataset Since images in Pascal3D dataset are captured from far camera poses, they are generally of low resolution and contain mostly diffuse appearance. Thus, we show reconstruction results on real images from LSUN [281] in Fig. 5.15, which contains images with more lighting variations. More comparisons can be found in Fig. 5.18.

During inference, we do not need any camera pose and predict shapes in canonical view. However, camera poses are needed to re-render the shape. Since ground-truth camera poses are not available for real images in LSUN dataset, we manually adjust the camera poses in Fig. 5.15 and Fig. 5.18. As a result, the re-rendered images are slightly misaligned with GT. However, DIB-R++ still accounts for specularities and predicts correct predominant lighting directions and clean textures.

Material Editing and Relighting Finally, we demonstrate some applications of DIB-R++ to artistic manipulation in Fig. 5.16. On the left, we show examples of editing the diffuse albedo, where we can insert text, decals or modify the base tint. Since our textures are not contaminated by lighting, clean texture maps can be easily edited by hand and the re-rendered images look natural. On the right, we show examples of editing lighting and surface materials, where we rotate the light (top) or increase glossiness (bottom). We also showcase results where we change lighting orientation or modify the object’s glossiness with consistent shading, which is not feasible with a naive, Lambertian-only shading model.

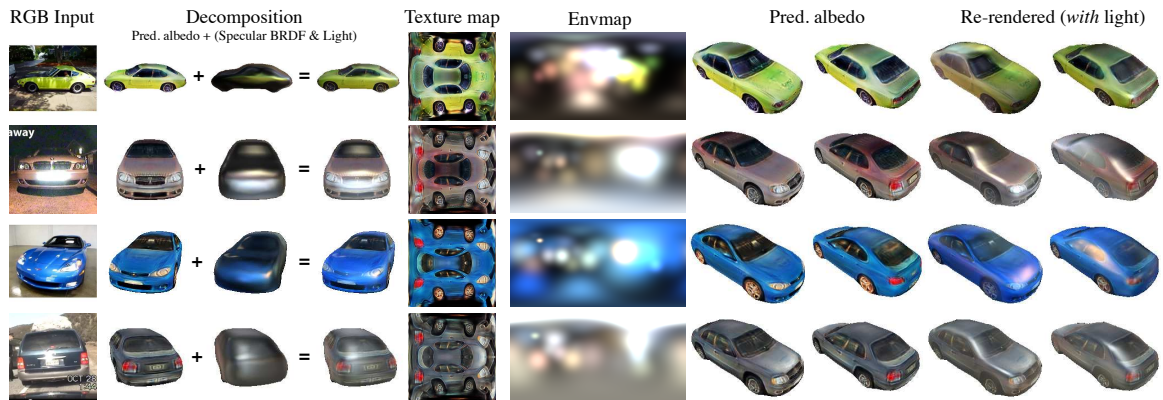


Figure 5.15: **Prediction on LSUN Dataset (Cars)**. DIB-R++, trained on StyleGAN dataset, can generalize well to real images. Moreover, it also predicts correct high specular lighting directions and usable, clean textures.



Figure 5.16: **Material Editing**. Our method allows for artistic manipulation of appearance, such as novel view synthesis, material editing (both diffuse and specular components), and relighting, thanks to our effective disentanglement.

5.5 Summary

In this chapter, we introduced a new powerful way to train inverse graphics models. We propose a state-of-the-art image synthesis approach, generating training data to train inverse graphics networks. We showcased our approach to obtain significantly higher quality 3D reconstruction results while requiring $10,000\times$ less annotation effort than standard datasets. By incorporating with more expressive rendering equations, we can also effectively disentangle material and lighting, enabling applications such as material editing and relighting.



Figure 5.17: **Prediction on Real-world Dataset (Cars)**. DIB-R++ accounts for high specular light and always have cleaner textures compared to DIB-R .



Figure 5.18: **Prediction on LSUN Dataset (Cars)**. Our model, trained on StyleGAN dataset, can be well generalized to real images. Moreover, it also accounts for high specular light and predict correct lighting directions and clean textures.

Chapter 6

Differentiable Structured Light Triangulation

We consider the problem of designing sequences of structured-light patterns for active stereo triangulation of a static scene. Unlike existing approaches that use predetermined patterns and reconstruction algorithms tied to them, we generate patterns on the fly in response to generic specifications: number of patterns, projector-camera arrangement, workspace constraints, spatial frequency content, etc. Our pattern sequences are specifically optimized to minimize the expected rate of correspondence errors under those specifications for an unknown scene, and are coupled to a sequence-independent algorithm for per-pixel disparity estimation. We achieve this by introducing a differentiable structured light imaging formation model and embedding it into learning frameworks. We design patterns from a machine learning perspective, deriving an objective function to evaluate performance of the patterns and optimizing patterns to minimize it. We demonstrate automatically optimized pattern sequences, in under three minutes on a laptop, that can outperform state-of-the-art triangulation techniques.

6.1 Introduction

A key tenet in structured-light triangulation is that the choice of projection patterns matters a lot. Over the years, the field has seen significant boosts in performance—in robustness, 3D accuracy, speed and versatility—due to new types of projection patterns, and new vision algorithms tailored to them [215, 214]. These advances continue to this day, for improved robustness to indirect light [74, 175, 77, 71, 79, 45, 37, 277]; computational efficiency [175, 54, 55]; high-speed imaging [126]; outdoor 3D scanning [76, 167]; and for 3D imaging

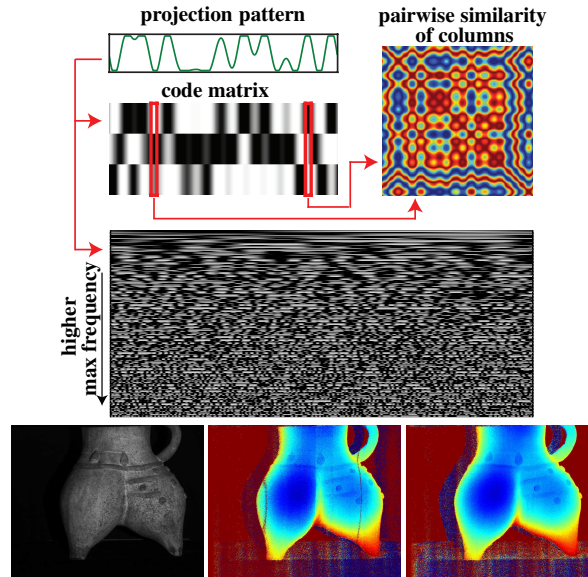


Figure 6.1: **Overview.** *Top:* A projection pattern is a 1D image projected along a projector’s rows. A sequence of them defines a code matrix, whose columns encode pixel position. We present a framework for computing stereo correspondences using *optimal code matrices*, which we generate on the fly. These matrices minimize the expected number of stereo errors that occur when the individual matrix columns are not very distinctive (red=similar, blue=dissimilar). *Middle:* A whole space of optimal matrices exists, for different numbers of projection patterns, image signal-to-noise ratio, spatial frequency content (sample patterns shown above), *etc.* *Bottom:* We use two automatically-generated four-pattern sequences to compute the depth map of the object shown on left. Both are optimized for a one-pixel tolerance for stereo errors, without (middle) and with (right) a bounding-box constraint. Both depth maps are unprocessed (please zoom in).

with specialized computational cameras [187], consumer-oriented devices [174, 52] and time-of-flight cameras [113, 20, 218, 10].

Underlying all this work is a fundamental question: what are the optimal patterns to use and what algorithm should process the images they create? This question was originally posed by Horn and Kiryati twenty years ago [94] but the answer was deemed intractable and not pursued. Since then, pattern design has largely been driven by practical considerations [219, 201, 202, 289, 93] and by intuitive concepts borrowed from many fields (*e.g.*, communications [203], coding theory [214], number theory [201], numerical analysis [136], *etc.*)

In this chapter we present the first computational approach to optimal design of patterns for structured light. We focus on the oldest, most accuracy-oriented version of this task: projecting a sequence of patterns one by one onto a static scene and using a camera to estimate per-pixel depth by triangulation. We view this problem from a machine learning perspective, deriving an objective function over the space of pattern sequences that quantifies the expected number of incorrect stereo correspondences. We introduce a differentiable structure light triangulation imaging formation model and embed it into learning frameworks which allows us to optimize patterns by minimizing the objective function using standard

tools [124].

Our optimization takes as input the projector’s resolution and the desired number of projection patterns. In addition to these parameters, however, it can generate patterns that are precisely optimized for 3D accuracy on the system at hand (Figure 6.1): for the specific arrangement of projector and camera; the shape and dimensions of the 3D scanning volume; the noise properties and peak signal-to-noise ratio of the overall imaging system; the defocus properties of the projector lens; a desired upper bound on the patterns’ spatial frequency; and any unknown scene geometry. Thus, in contrast to prior work, we do not provide a closed-form expression or “codebook” for a one-size-fits-all pattern sequence; we give a way to generate scene-independent pattern sequences on the fly at near-interactive rates—less than three minutes on a standard laptop—so that the patterns and the associated reconstruction algorithm can be easily and automatically adapted for best performance. We call this paradigm *structured light à la carte*.

At the heart of our approach lies an extremely simple differentiable structured light imaging formation model, from which we derive a decoding algorithm for computing stereo correspondences independently of projection pattern. It also makes the pattern optimization problem itself tractable: by giving us a way to quantify the expected errors a pattern sequence will cause, it leads to an objective function over sequences that can be optimized numerically.

From a conceptual point of view our work makes three important contributions over the state of the art. First and foremost, our optimization-based approach turns structured-light imaging from a problem of *algorithm design* (for creating patterns [215], unwrapping phases [97, 100, 197, 74], computing correspondences [219], handling projector defocus [9, 79]) into one of *problem specification* (how many patterns, what working volume, what imaging system, *etc.*). The rest is handled automatically by pattern sequence optimization framework. Second, we demonstrate optimized pattern sequences that can outperform state-of-the-art encoding schemes on hard cases: low numbers of patterns, geometrically-complex scenes, low signal-to-noise ratios. These are especially important in settings where speed and low-power imaging are of the essence. Third, our formulation gives rise to new families of pattern sequences with unique properties, including (1) sequences designed to recover approximate, rather than exact, correspondences and (2) sequences designed with information about free space and stereo geometry already built in. This encodes geometric scene constraints directly into the optical domain for added reliability—via the patterns themselves—rather than enforcing them by post-processing less reliable 3D data.

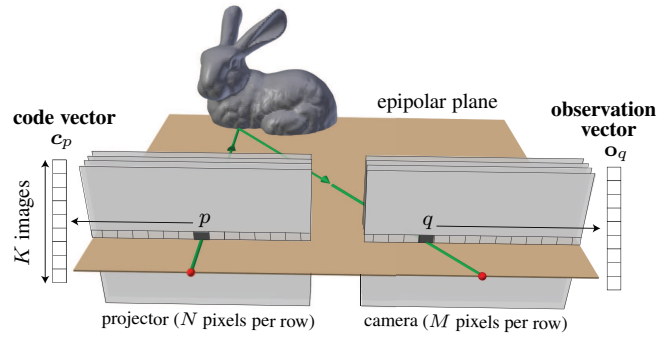


Figure 6.2: **Viewing Geometry.** We assume the projector-camera system has been rectified, *i.e.*, epipolar lines are along rows.

6.2 Optimal Structured Light

Fundamentally, structured-light triangulation requires addressing two basic questions: (1) what patterns to project onto the scene and (2) how to compute projector-camera stereo correspondences from the images captured. Specifying a “good” set of projection patterns can be thought of as solving a one-dimensional *position encoding* problem for pixels on an epipolar line. Conversely, computing the stereo correspondence of a camera pixel can be thought of as a *position decoding* problem. We begin by formally defining both problems, then talk about how to solve them with a differentiable structured light image formation model under learning frameworks.

The Code Matrix A set of K projection patterns implicitly assigns a K -dimensional *code vector* \mathbf{c}_p to each pixel p on the epipolar line (Figure 6.2). The elements of \mathbf{c}_p are the pixel’s intensity in the individual patterns, they can be non-binary, and must be chosen so that each code vector is as distinctive as possible. This becomes harder to do as K decreases (*i.e.*, vectors with fewer dimensions are less distinctive) and as the number of pixels increases (*i.e.*, there are more vectors to be distinguished). We represent the code vectors of an epipolar line with a *code matrix* \mathbf{C} . This matrix has size $K \times N$ for an epipolar line with N pixels.

Position Decoding Consider a camera pixel q . The K intensities observed at that pixel define a K -dimensional observation vector \mathbf{o}_q . Given this vector and the code matrix \mathbf{C} , the goal of position decoding is to infer its corresponding projector pixel p^* :

$$p^* = \text{Decode}(\mathbf{o}_q, \mathbf{C}) . \quad (6.1)$$

Note that p^* may or may not agree with the true correspondence p (Figure 6.2), as finding accurate correspondence is a difficult problem because observations are corrupted by measurement noise and because the relation between observation vectors and code vectors can be highly non-trivial for general scenes.

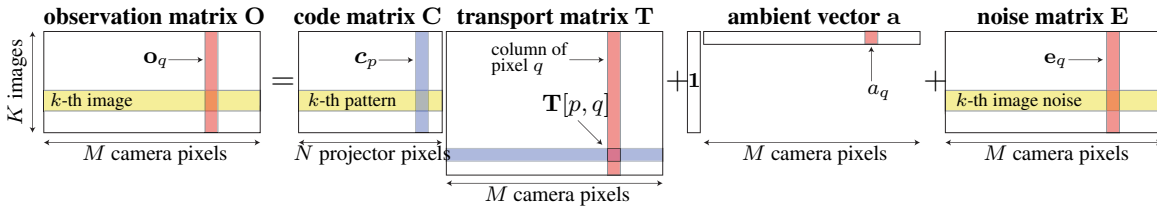


Figure 6.3: **Generative Model of Image Formation for a Single Epipolar Line Across K Images.** Each column of matrix \mathbf{O} is an observation vector (red) and each row collects the observations from a single image across all pixels on the epipolar line (yellow). All yellow rows are associated with the same input image and all red columns are associated with the same camera pixel q . The gray column and row are associated with the same projector pixel p .

Position Encoding The code matrix \mathbf{C} should be chosen to minimize decoding error. For a given projector-camera system and a specific scene, we quantify this error by counting the incorrect correspondences produced by decoder:

$$\text{Error}(\mathbf{C}, \epsilon) \stackrel{\text{def}}{=} \sum_{q=1}^M \mathbb{1} \left(|\text{Decode}(\mathbf{o}_q, \mathbf{C}) - \text{Match}(q)| > \epsilon \right), \quad (6.2)$$

where $\text{Match}(q)$ is the true stereo correspondence of image pixel q ; ϵ is a tolerance threshold that permits small correspondence errors; $\mathbb{1}(\cdot)$ is the indicator function; and the summation is over all pixels on the epipolar line.

We now formulate optimal position encoding as the problem of finding a code matrix \mathbf{C}_ϵ^* that minimizes the expected number of incorrect correspondences:

$$\mathbf{C}_\epsilon^* = \arg \min_{\mathbf{C}} \mathbb{E}[\text{Error}(\mathbf{C}, \epsilon)] \quad (6.3)$$

where $\mathbb{E}[\cdot]$ denotes expectation over a user-specified domain of plausible scenes and imaging conditions. We call \mathbf{C}_ϵ^* the *optimal code matrix* for tolerance ϵ .

Key Objective We seek an efficient solution to the nested optimization problem in Eq. (6.3). To do this, we introduce a differentiable structured light image formation model defined with the epipolar geometry (Section 6.3). This leads to a correlation-based decoder for structured-light reconstruction that is nearly optimal in low, Gaussian distributed noise settings (Section 6.4). Using this decoder, we derive a softmax-based approximation to the objective function of Eq. (6.2). We then embed everything inside learning frameworks and optimize patterns that minimize the expected number of stereo mismatches (Section 6.5).

6.3 Differentiable Epipolar-Only Imaging Formation Model

In this section, we introduce our differentiable structured light imaging formation model. To simplify our formal analysis we assume that all light transport is *epipolar*. Specifically, we assume that observation vectors depend only on code vectors on the corresponding epipolar line. This condition applies to conventionally-acquired images when global light transport, projector defocus and camera defocus are negligible.¹ It also applies to all images captured by an epipolar-only imaging system regardless of scene content—even in the presence of severe global light transport [190].

When epipolar-only imaging holds and the system has been calibrated radiometrically, the relation between code vectors and observation vectors is given by (Figure 6.3):

$$\underbrace{\begin{bmatrix} \mathbf{o}_1 & \cdots & \mathbf{o}_M \end{bmatrix}}_{\text{observation matrix } \mathbf{O}} = \underbrace{\begin{bmatrix} \mathbf{c}_1 & \cdots & \mathbf{c}_N \end{bmatrix}}_{\text{code matrix } \mathbf{C}} \mathbf{T} + \mathbf{1} \underbrace{\begin{bmatrix} a_1 & \cdots & a_M \end{bmatrix}}_{\text{ambient vector } \mathbf{a}} + \mathbf{E} \quad (6.4)$$

where $\mathbf{o}_1, \dots, \mathbf{o}_M$ are the observation vectors of all pixels on an epipolar line; $\mathbf{a} = [a_1, \dots, a_M]$ is the contribution of ambient illumination vector to these pixels; $\mathbf{1}$ is a column vector of all ones; matrix \mathbf{E} is the observation noise; and \mathbf{T} is the $N \times M$ epipolar transport matrix. Element $\mathbf{T}[p, q]$ of this matrix describes the total flux transported from projector pixel p to camera pixel q by direct surface reflection, global transport, and projector or camera defocus.

The epipolar-only model of Eq. (6.4) encodes the geometry and reflectance of the scene as well as the scene’s imaging conditions. Note that although it is very simple, this formula is differentiable and the gradients from the observation \mathbf{O} w.r.t to the code \mathbf{C} is naturally the light transport matrices \mathbf{T} . Such differentiability allows us to embed the image formation model into learning frameworks to optimize the optimal code matrix \mathbf{C} . More importantly, by adjusting epipolar transport matrices \mathbf{T} , ambient vectors \mathbf{a} , and noise matrices \mathbf{E} , we can represent various imaging conditions and specifically optimize \mathbf{C} for them. Next, we describe the parameter space of \mathbf{T} , \mathbf{a} , and \mathbf{E} under specific imaging conditions.

6.3.1 Plausible Parameter Space

Epipolar Transport Matrices Let us first consider the space of plausible matrices \mathbf{T} . Even though the space of $N \times M$ matrices is extremely large, the matrices relevant to structured-light imaging belong to a much smaller space. This is because the elements of \mathbf{T} associated with indirect light generally have far smaller magnitude than direct elements—and can thus be ignored [190]. This in turn makes the observation matrix \mathbf{O} very efficient to compute. In

¹See Figure 6.8 for experiments with scenes with significant indirect light, where this condition does not strictly hold.

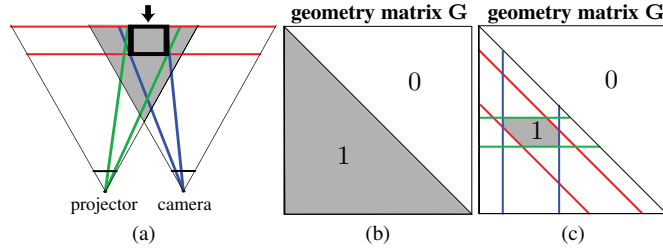


Figure 6.4: **Geometric Constraints.** (a) Top view of the epipolar plane. (b) \mathbf{T} is always lower triangular because the 3D rays of all other elements intersect behind the camera. (c) \mathbf{T} 's non-zero elements are restricted even further by knowledge of the working volume (e.g., black square in (a)): its depth range (red) and its angular extent from the projector (green) and the camera (blue) define regions in \mathbf{T} whose intersection contains all valid correspondences.

particular, we consider imaging conditions for the following two families:

(A) *Direct-only \mathbf{T} , unconstrained:* The non-zero elements of \mathbf{T} represent direct surface reflections and each camera pixel receives light from at most one projector pixel. It follows that each column of \mathbf{T} contains at most one non-zero element. Moreover, the location of that element is a true stereo correspondence. The observation vector is therefore a noisy scaled-and-shifted code vector:

$$\mathbf{o}_q = \mathbf{T}[p, q] \cdot \mathbf{c}_p + a_q + \mathbf{e}_q \quad (6.5)$$

where vector \mathbf{e}_q denotes noise. We assume that the location of the non-zero element in each column of \mathbf{T} is drawn randomly from the set $\{1, \dots, N\}$ and its value, $\mathbf{T}[p, q]$, is a uniform i.i.d random variable over $[0, 1]$. This amounts to being completely agnostic about the location and magnitude of \mathbf{T} 's non-zero elements.

(B) *Direct-only \mathbf{T} with geometry constraints* We now restrict the above family to exclude geometrically-implausible stereo correspondences. These are elements of \mathbf{T} whose associated 3D rays either intersect behind the image plane or outside a user-specified working volume (Figure 6.4a). We specify these invalid elements with a binary indicator matrix \mathbf{G} (Figure 6.4b, 6.4c). Given this matrix, we assume that the location of the non-zero element in each column of \mathbf{T} is drawn uniformly from the column's valid elements.

Observation Noise and Ambient Vector The optimality of our position decoder (Section 6.4) relies on noise being signal independent and normally distributed. The position encoder of Section 6.5, on the other hand, can accommodate any model of sensor noise as long as its parameters are known. We assume that the elements of the ambient vector \mathbf{a} follow a uniform distribution over $[0, a_{\max}]$, where a_{\max} is the maximum contribution of ambient light expressed as a fraction of the maximum pixel intensity.

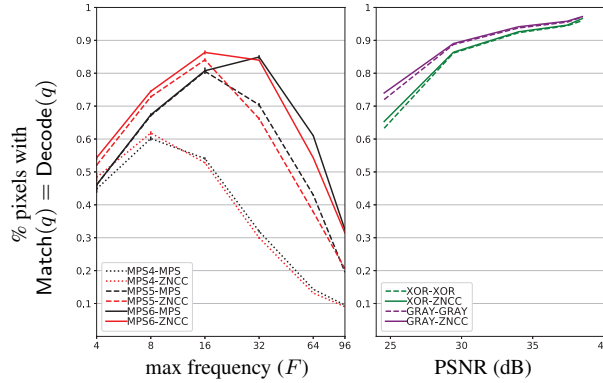


Figure 6.5: **ZNCC vs. Native Decoding.** *Left:* We project K micro phase shifting (MPS) patterns [74] of maximum frequency F onto a known planar target and compute correspondence errors using our ZNCC decoder (red) and the one by the MPS authors (black). *Right:* A similar comparison for 10 Gray codes (purple) and 10 XOR-04 codes (green), projected along with their binary complement. We used the binarization technique in [219] for “native” decoding. Since these codes have no frequency bound we plot them against image PSNR. In all cases, ZNCC decoding yields comparable results.

6.4 Optimal Position Decoding

Now we talk about our decoder derived from the epipolar-only image formation model. Suppose we are given a code matrix \mathbf{C} and an observation vector \mathbf{o}_q that conforms to Eq. (6.4). Our task is to identify the stereo correspondence of pixel q . We seek a generic solution to this problem that does not impose constraints on the contents of the code matrix: it can contain code vectors defined *a priori*—such as MPS [74] or XOR [77] codes—or be a general matrix computed automatically through optimization.

Fortunately, from the differentiable imaging formation model, we derive an extremely simple and near-optimal algorithm to do this: just compute the zero-mean normalized cross-correlation (ZNCC) [164] between \mathbf{o}_q and the code vectors, and choose the one that maximizes it. This algorithm becomes optimal as noise goes to zero and belongs to Gaussian distribution:²

Recall the differentiable image formation model in Sec. 6.3. To be simple, we derive it from the pixelwise image formation model (Eq. (6.5)). Assume the true correspondence of camera pixel q is projector column p , we first compute the mean value of the observation vector \mathbf{o}_q and code vector \mathbf{c}_p :

$$\text{mean}(\mathbf{o}_q) = \mathbf{T}[p, q] \cdot \text{mean}(\mathbf{c}_p) + a_q + \text{mean}(\mathbf{e}_q) . \quad (6.6)$$

Here, $\text{mean}()$ is over the elements of a code vector. $\mathbf{T}[p, q]$ and a_q are scalars, so they will

²ZNCC is a direct generalization of the widely known correlation-based decoder for communication channels corrupted by additive white Gaussian noise [203].

not change. Simply minus Eq. (6.5) with Eq. (6.6), we immediately get rid of the ambient term:

$$\mathbf{o}_q - \text{mean}(\mathbf{o}_q) = \mathbf{T}[p, q] \cdot (\mathbf{c}_p - \text{mean}(\mathbf{c}_p)) + \mathbf{e}_q - \text{mean}(\mathbf{e}_q) . \quad (6.7)$$

When the noise is in Gaussian distribution and it goes to zero, $\mathbf{e}_q - \text{mean}(\mathbf{e}_q)$ also goes to zero. It indicates that if q and p are true correspondence, the normalized correlation between $\mathbf{o}_q - \text{mean}(\mathbf{o}_q)$ and $\mathbf{c}_p - \text{mean}(\mathbf{c}_p)$ should be close to one. Therefore, given an observation vector \mathbf{o}_q and code matrix \mathbf{C} , we propose to use ZNCC to evaluate the probability of correspondence matching between \mathbf{o}_q and each code vector \mathbf{c}_p , then pick up the maximum one:

$$\text{ZNCC}(\mathbf{o}_q, \mathbf{c}_p) = \frac{\mathbf{o}_q - \text{mean}(\mathbf{o}_q)}{\|\mathbf{o}_q - \text{mean}(\mathbf{o}_q)\|} \cdot \frac{\mathbf{c}_p - \text{mean}(\mathbf{c}_p)}{\|\mathbf{c}_p - \text{mean}(\mathbf{c}_p)\|} . \quad (6.8)$$

$$\text{Decode}(\mathbf{o}_q, \mathbf{C}) = \arg \max_{1 \leq p \leq N} \text{ZNCC}(\mathbf{o}_q, \mathbf{c}_p) . \quad (6.9)$$

The proposed ZNCC decoder has two important implications. First, as a sequence-independent decoder, ZNCC performs comparable or even better than those specifically designed ones (Figure 6.5). It suggests that there is potentially no accuracy advantage to be gained by designing decoding algorithms tailor-made for specific codes³. Second, it allows us to transform the nested position-encoding optimization of Eq. (6.3) into a conventional non-linear optimization. This opens the door to automatic generation of optimized code matrices, discussed next.

6.5 Optimal Position Encoding

With the differentiable imaging formation model and the derived sequence-independent position decoder, now we can embed them inside learning frameworks to automatically optimize patterns for the chosen imaging conditions. We begin by developing a continuous approximation to the function $\text{Error}()$ in Eq. (6.2). This function counts the decoding errors that occur when a given code matrix \mathbf{C} is applied to a specific scene and imaging condition, *i.e.*, a specific transport matrix \mathbf{T} , observation noise \mathbf{E} , and ambient vector \mathbf{a} . To evaluate the position-encoding objective function on matrix \mathbf{C} , we draw S fair samples over \mathbf{T} , \mathbf{E}

³Strictly speaking this applies to decoders that estimate depth at each pixel independently. Note that ZNCC decoding does incur a computational penalty: it requires $O(N)$ operations versus $O(K)$ of most specialized decoders.

and \mathbf{a} :

$$\mathbb{E}[\text{Error}(\mathbf{C}, \epsilon)] = (1/S) \sum_{\mathbf{T}, \mathbf{E}, \mathbf{a}} \text{Error}(\mathbf{T}, \mathbf{E}, \mathbf{a}, \mathbf{C}, \epsilon) . \quad (6.10)$$

Softmax Approximation of Decoding Errors Consider a binary variable that tells us whether or not the optimal decoder matched camera pixel q to a projector pixel p . We approximate this variable by a continuous softmax function using Eqs. (6.12)-(6.13): Equation (6.12) states that in order for projector pixel p to be matched to q , the ZNCC score of p 's code vector must be greater than all others. Eq. (6.13) approximates the indicator variable with softmax function. It also introduces a softmax ratio μ , as the scalar μ goes to infinity, the ratio tends to 1 if pixel p 's ZNCC score is the largest and tends to 0 otherwise:

$$\mathbb{1}\left(\left|\text{Decode}(\mathbf{o}_q, \mathbf{C}) - p\right| = 0\right) \quad (6.11)$$

$$= \mathbb{1}\left(\arg \max_{1 \leq r \leq N} \text{ZNCC}(\mathbf{o}_q, \mathbf{c}_r) = q\right) \quad (6.12)$$

$$\stackrel{\mu \rightarrow \infty}{=} \frac{\exp\left(\mu \cdot \text{ZNCC}(\mathbf{o}_q, \mathbf{c}_p)\right)}{\sum_{r=1}^N \exp\left(\mu \cdot \text{ZNCC}(\mathbf{o}_q, \mathbf{c}_r)\right)} \quad (6.13)$$

$$\stackrel{\text{def}}{=} f_{\mu}(\mathbf{C}, \mathbf{o}_q, p) . \quad (6.14)$$

To count all correct matches on an epipolar line, we evaluate the softmax value at the true stereo match of every pixel q , and then compute their sum. Using the notation in Eq. (6.14):

$$\text{Correct}(\mathbf{T}, \mathbf{E}, \mathbf{a}, \mathbf{C}) = \sum_{q=1}^M f_{\mu}(\mathbf{C}, \mathbf{o}_q, \text{Match}(q)) . \quad (6.15)$$

Finally, incorporating the tolerance parameter ϵ to permit small errors in stereo correspondences we get:

$$\text{Correct}(\mathbf{T}, \mathbf{E}, \mathbf{a}, \mathbf{C}, \epsilon) = \sum_{q=1}^M \sum_{r=-\epsilon}^{\epsilon} f_{\mu}(\mathbf{C}, \mathbf{o}_q, \text{Match}(q) + r) \quad (6.16)$$

$$\text{Error}(\mathbf{T}, \mathbf{E}, \mathbf{a}, \mathbf{C}, \epsilon) = M - \text{Correct}(\mathbf{T}, \mathbf{E}, \mathbf{a}, \mathbf{C}, \epsilon) . \quad (6.17)$$

Forward and Backward Steps Now we can combine the differentiable image formation model, the ZNCC decoder and the objective function together, optimizing patterns with gradients

descent algorithms. The optimization procedure contains forward and backward steps, where the former evaluate the performance of the patterns while the latter update the patterns to minimize the error. Specifically, in the forward process, given a randomized code \mathbf{C} , we sample the light transport matrices \mathbf{G} ⁴, observation noise \mathbf{E} , and ambient vector \mathbf{a} to synthesize the observation matrix \mathbf{O} via the image formation model (Eq. (6.4)). We then decode the position and calculate the error with the defined error matrix (Eq. (6.17)). Conversely, in the backward process, thanks to all the differentiable operations, we compute the gradients of the patterns and update them to minimize the error. During updating, we constrain the patterns by user-defined frequency and values, *e.g.*, setting the maximum frequency as 16 and the value range in $[0, 1]$. We repeat the loop until it converges.

Sampling Scenes and Imaging Conditions Constructing fair samples of the observation noise \mathbf{E} and ambient vector \mathbf{a} is straightforward and omitted. To construct a direct-only matrix whose geometric constraints are a matrix \mathbf{G} , we proceed as follows. We first randomly assign a valid stereo correspondence to each camera pixel according to \mathbf{G} . This specifies the location of the single non-zero element in each column of \mathbf{T} (Figure 6.3). We then assign a random value⁵ to each of those elements independently. The result is a valid direct-only transport matrix, *i.e.*, a sample from family (B) in Section 6.4. Note such an image formation model is done in simulation. We investigate how to optimize the real-world projector-camera systems in next chapter.

Optimization Settings We use the Adam optimizer [124] to perform stochastic gradient descent on the objective function in Eq. (6.10) with a fixed learning rate of 0.01. The user-specified parameters are (1) the number of projector pixels N ; (2) the number of camera pixels M ; (3) the number of projection patterns K ; (4) the desired tolerance parameter ϵ ; and (5) the geometric constraint matrix \mathbf{G} . The result of the optimization is a code matrix \mathbf{C}_ϵ^* .

We initialize the optimization with a random $K \times N$ code matrix \mathbf{C} and draw a total of $S = 500$ samples $(\mathbf{T}, \mathbf{E}, \mathbf{a})$ at iteration 1 to define the objective function of Eq. (6.10). These samples act as a “validation set” and remain fixed until convergence. For gradient calculations we use a minibatch containing two new randomly-drawn samples per iteration. Optimization converges in around 250 iterations (152 seconds on an 8-core 2.3GHz MacBook Pro laptop for a six-pattern matrix). We found that increasing the number of samples had no appreciable effect on the quality of \mathbf{C}_ϵ^* (*i.e.*, the number of decoding errors on other randomly-generated scenes and imaging conditions). What does make a big difference,

⁴Recall that we use \mathbf{G} to denote light transport matrices with geometric constraints.

⁵The random value is drawn from albedo distributions, which we set it as uniform distribution over $[0, 1]$

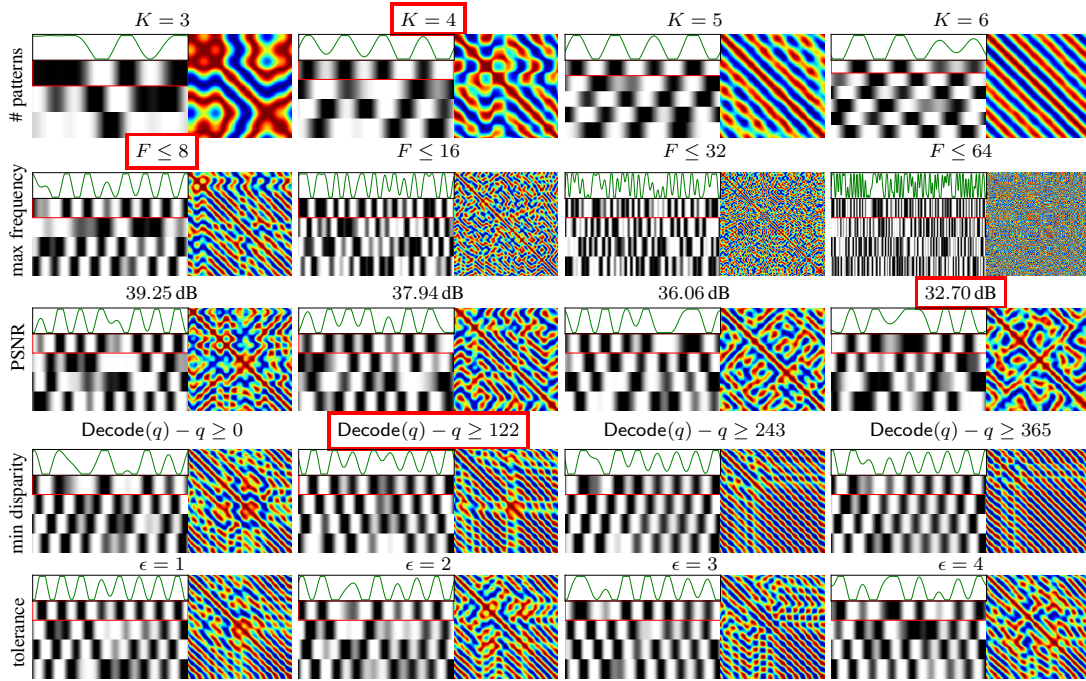


Figure 6.6: **A Walk in the Space of Optimal Codes.** To better visualize code structure, the pairwise scores $\text{ZNCC}(c_i, c_j)$ of code vectors are shown as a jet-color-mapped matrix (deep red = 1, deep blue = -1). These can be treated as a confusion matrix. *Row 1:* We set the maximum spatial frequency of the patterns to $F = 4$ and the image PSNR to be maximal for our imaging conditions (frame rate=50Hz, camera gain=1, known read noise, pixel intensity that spans the full interval $[0, 1]$). We then compute the optimal code matrix for our 608-pixel projector for different numbers of patterns and no other constraints. *Row 2:* We then choose $K = 4$ (outlined in red in Row 1) and compute optimal matrices for different bounds on the maximum spatial frequency, with everything else fixed as above. *Row 3:* We now set the frequency to 8 (outlined in red in Row 2) and compute optimal matrices for different values of pixel PSNR (*i.e.*, the maximum image intensity gets increasingly smaller), again with everything else fixed as above. *Rows 4 and 5:* We follow the exact same process for different lower bounds on disparity (*i.e.*, the maximum scene depth is increasingly being restricted), and different tolerances in correspondence error.

however, is the value of the softmax multiplier μ : there is significant degradation in quality for $\mu < 300$, but increasing it beyond that value has little effect. We use $\mu = 300$ for all results shown. See Sec. 6.6 for more details.

Advanced Sensor Noise Modeling Although the ZNCC decoder is optimal only for additive Gaussian noise, the objective function in Eq.(6.10) can incorporate any sensor noise model: we simply draw samples of \mathbf{E} from the camera’s noise distribution. We found that this improves the real-world performance of the optimized codes. Sec. 6.6 for more details.

6.6 Experiments

The Space of Optimal Code Matrices Figure 6.6 shows several code matrices generated by our optimizer. It is clear by inspection that the codes exhibit a very diverse structure that adapts significantly in response to user specifications. Increasing the frequency content (Row 2)

produces confusion matrices with much less structure—as one would intuitively expect from vectors that are more distinctive. Interestingly, codes adapted to lower peak signal-to-noise ratio (PSNR) conditions have confusion matrices with coarser structure. We did not, however, observe an appreciable difference in the real-world performance of those matrices. Row 3 of Figure 6.6 illustrates the codes’ adaptation to geometric constraints. Specifically, only points on the plane at infinity can have $\text{Decode}(q)=q$ and for 3D points that are closer, a camera pixel can only be matched to a projector pixel on its right (Figure 6.4b). Comparing the code matrix for an unrestricted \mathbf{T} (red box on Row 3) to that of a lower-triangular \mathbf{T} (first column in Row 4) one sees significant re-organization in the confusion matrix; the optimization effectively “focuses” the codes’ discriminability to only those code vectors that yield valid 3D points. On the other hand, code matrices that compute approximate, rather than exact correspondences, exhibit coarser structure in their confusion matrix (Row 4).

Experimental System We further validate the optimized patterns with real devices. We acquired all images at 50Hz and 8 bits with a 1280×1024 monochrome camera supplied by IDS (model IDS UI-3240CP-M), fitted with a Lensation F/1.6 lens (model CVM0411). For pattern projection we used a 100-lumen DLP projector by Keynote Photonics (model LC3000) with a native resolution of 608×684 and only the red LED turned on. We disabled gamma correction, verified the system’s linear radiometric response, and measured the sensor’s photon transfer curve. This made it possible to get a precise measure of PSNR independently for each pixel on the target. We experimented with three different models of pixel noise for our position-encoding optimization: (1) additive Gaussian, (2) Poisson shot noise with additive read noise, and (3) exponential noise [233] with additive read noise.

Ground Truth We printed a random noise pattern of bounded frequency onto a white sheet of paper and placed it on a planar target 60cm away from the stereo pair (Figure 6.7, bottom row, third column). We used two different pattern sequences to obtain “ground-truth” disparity maps: 160 conventional phase-shifted patterns (10 shifts of 16 sinusoidal patterns, with frequencies 1 through 16) and 20 XOR patterns (including the complement codes). We adjusted the aperture so that the maximum image intensity was 200 for a white projection pattern (*i.e.*, a high-PSNR regime at the brightest pixels) and focused the lens on the target. We decode the captured images by ZNCC decoding to assign a depth to each pixel and further validate the depth consistency between two codes. For 97% of pixels the disparities were identical in the two maps; the rest differed by ± 1 disparity. Thus, correctness above 97% against these maps is not significant. We optimize all of our code matrices for these high-PSNR conditions with the exponential-plus-read-noise model.

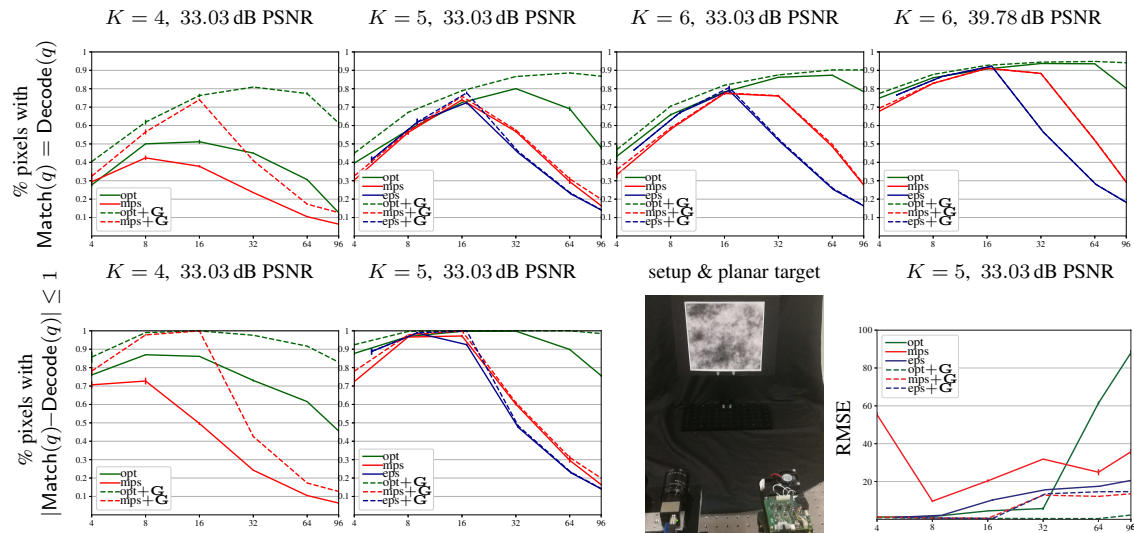


Figure 6.7: **Quantitative Evaluation.** *Top row and first two columns of bottom row:* Each data point represents three independent acquisitions with the same pattern sequence (x -axis is frequency). Error bars indicate the smallest and largest fraction of correct correspondences in those runs. We used $\epsilon = 0$ for optimization in the top row and $\epsilon = 1$ in the bottom. Solid lines show results when no geometry constraints are imposed on code optimization and on decoding. Dashed lines show what happens when we use a depth-constrained geometry matrix \mathbf{G} (Figure 6.4c). For EPS and MPS, the constraint is used only for decoding, *i.e.*, we search among the valid correspondences for the one that maximizes the ZNCC score. Our codes, on the other hand, are optimized for that constraint and decoded with it as well. *Bottom row, right: RMSE plots.*

Quantitative Evaluation We focus here on the most challenging cases: very small number of patterns and low PSNR. To evaluate low-PSNR performance, we reduced the aperture so that the brightest pixel intensity under a white projection pattern is 60, and counted the pixels whose correspondences are within ϵ of the ground truth. Figure 6.7 compares our optimized code matrices against those of MPS and EPS, using the same ZNCC decoder for all codes. Several observations can be made from these results. First, our code matrices outperform MPS and EPS—which represent the current state of the art—in all cases shown. This performance gap, however, shrinks for larger numbers of patterns. Second, our codes perform significantly better than EPS and MPS at higher spatial frequencies. This is despite the fact that those coding schemes were specifically designed to produce high-frequency patterns. It is also worth noting that the performance degradation of MPS and EPS at high frequencies cannot be explained by camera defocus because the camera’s aperture was small in these experiments (*i.e.*, large depth of field). Third, geometric constraints confer a major performance advantage to all codes at low pattern counts. The gain, however, is higher for our codes since they are optimized precisely for them. Fourth, code matrices that are geometry-constrained and optimized for a small error tolerance tend to produce low root-mean-squared errors (RMSE) for most frequencies.

Qualitative Results Reconstructions of several objects are shown in Figure 6.1 (using four patterns) and Figure 6.8 (using five and six patterns). The comparison in Figure 6.1 indicates

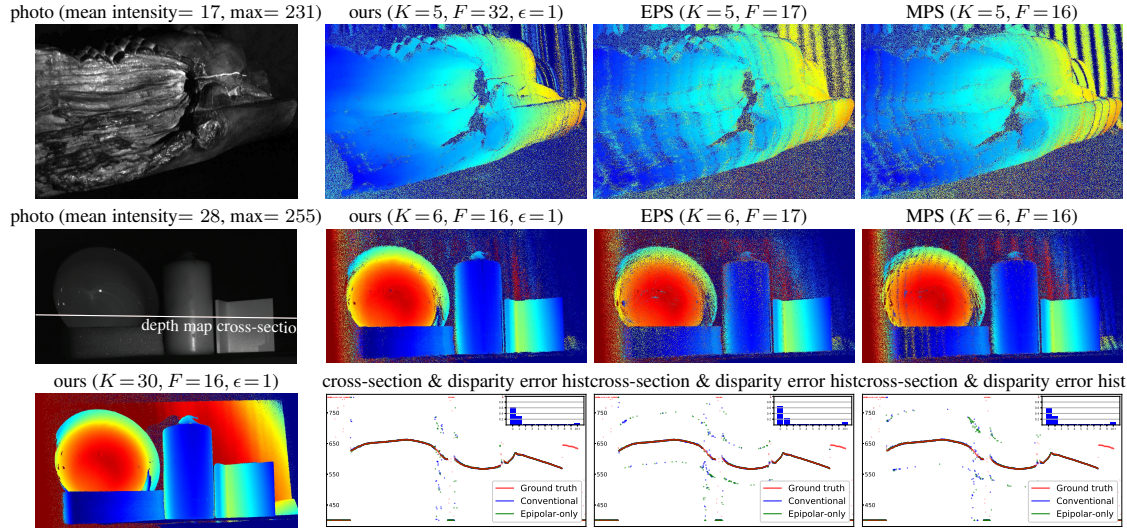


Figure 6.8: **Qualitative Comparisons.** We acquired depth maps for the scenes on the left using three methods, with the same ZNCC decoder and the same triangular geometry matrix \mathbf{G} (Figure 6.4b). For each method, we reconstructed the scenes for several maximum frequencies in the range $[4, 32]$ and show depth maps for each method’s best-performing frequency. *Top row:* Reconstructing a dark, varnished and sculpted wooden trunk with five patterns. *Middle row:* Reconstructing a scene with significant indirect transport (a bowl, candle, and convex wedge) using conventional imaging and six patterns. *Bottom row:* Depth map acquired with many more patterns, along with cross-sections of the above depth maps (blue points) and a histogram of disparity errors (please zoom in to the electronic copy). For reference, we include the cross-sections of depth maps acquired using epipolar-only imaging [190] with the exact same patterns (green points), as well as of “ground truth” depth maps acquired with 160 shifted cosine patterns of frequencies 16 to 31 using epipolar-only imaging (red points).

that computing geometry-constrained codes has a clear effect on the quality of the results—a trend observed in our quantitative comparisons as well. In Figure 6.8 we specifically chose to reconstruct a dark scene as well as a scene with significant indirect light to compare performance under low-PSNR conditions and general light transport. We observe that our depth maps have significantly fewer outliers than EPS and MPS and are less influenced by depth discontinuities. Moreover, despite not being specifically optimized for indirect light, we obtain better depth maps there as well.

6.6.1 Ablation Study

Optimization Hyper-parameters Two hyper-parameters require tuning in our framework: (1) the multiplier μ in the softmax approximation of Eq. 6.13, and (2) the mini-batch size for performing stochastic gradient descent. Figure 6.9 shows the effect of these two parameters in optimizing a code matrix with 4 patterns and maximum frequency 16, without any geometry constraints.

As can be seen from Figure 6.9 (left), when the multiplier μ decreases, the softmax approximation of the objective function is less accurate. As a result, the optimization does not lead to a good local minimum. On the other hand, increasing the multiplier beyond 300 does not have any noticeable impact on minimizing the objective function. Thus, to avoid any floating-point arithmetic issues and to have more stable gradients, we picked $\mu = 300$ for

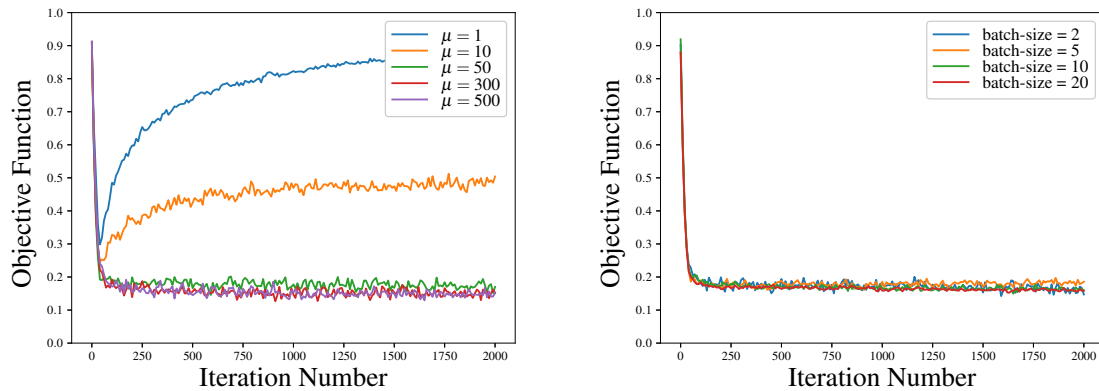


Figure 6.9: **Hyper-parameter Tuning.** We show validation error (on 500 fixed random samples) over iterations in optimization of a sample code matrix of 4 patterns and 608 pixels with maximum frequency 16.

all of our code optimizations.

To perform the gradient descent, in each iteration we draw random samples of \mathbf{T} , \mathbf{E} , and \mathbf{a} , where each sample includes all the valid camera pixels. It can be seen from Figure 6.9 (right) that increasing the sample size does not affect the attained minimum of the objective function over the pre-drawn validation set. Therefore, to speed up the optimization process we used 2-sample mini-batches for all our code optimizations.

To gain some intuition about why the mini-batch size does not affect the results, consider the following. Since in each iteration we draw new samples of \mathbf{T} , \mathbf{E} , and \mathbf{a} , we are essentially training on an infinite-sized dataset, which helps avoid overfitting. As a result, even small mini-batch sizes can still converge to good local minima.

Together, the fast convergence rate of our optimization—less than 250 iterations—and the ability to use very small mini-batch sizes for gradient computation, allow us to optimize code matrices at near-interactive rates.

Sensor Noise Lastly, we performed a new experiment of reconstructing the object with patterns optimized for different camera noise models, and comparing the depth maps to each other. Figure 6.10 shows 3D acquisition results using codes that were optimized for three different noise models. Of the three, the exponential-plus-additive-Gaussian model is the most pessimistic about the effect of signal-dependent noise: unlike Poisson shot noise whose variance increases linearly with the signal, noise variance in the exponential model increases quadratically with the signal. Nevertheless, despite the very significant difference between the three noise models, reconstruction results for all three noise-optimized codes were very similar. While the Poisson-plus-additive-Gaussian codes yielded error histograms which

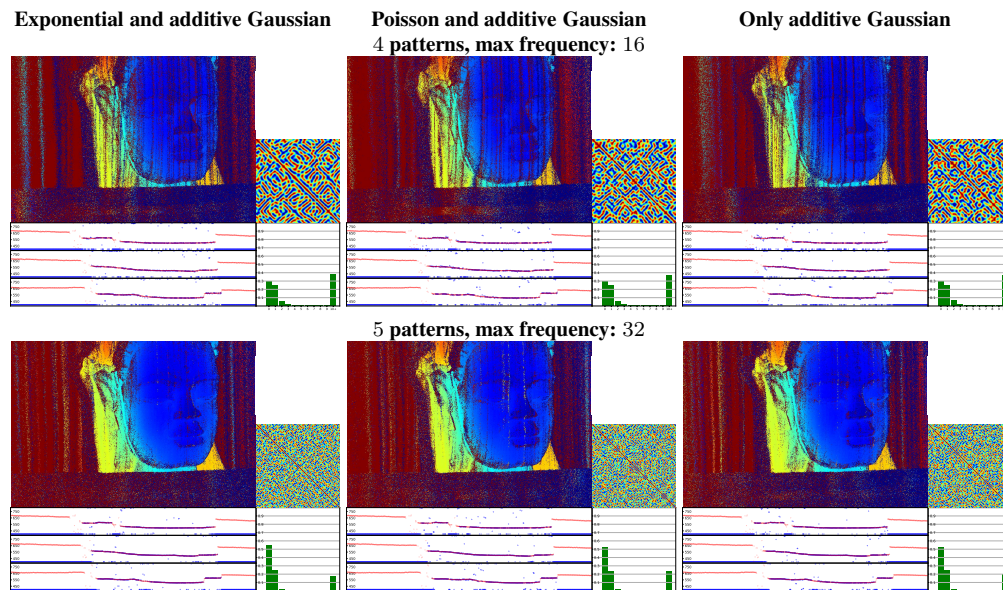


Figure 6.10: **Code Performance as a Function of Camera Noise Model.** Each column is patterns optimized under a specific noise model, while each row is a specific pattern setting. We show the *raw* depth maps in top left and the confusion matrix in the middle right. To assess it quantitatively, we also compare 2D slices of the reconstructed 3D pointset against those computed by the ground-truth sequence, and show the histogram of differences in the computed and ground-truth disparities in the bottom part. Please zoom into the electronic copy for details.

were marginally better for four patterns—as would be expected from a model that captures the actual noise statistics of a conventional CMOS sensor under incoherent illumination—the effect is relatively small. We decided to fix the model to exponential-plus-additive-Gaussian for all subsequent code optimizations.

6.7 Summary

We believe this is just a small first step in designing optimized codes for structured light. The specific imaging regime we chose to study—small numbers of patterns, low-PSNR conditions—leaves a lot of room for further exploration. On the high-accuracy end of the spectrum, the ability to quickly optimize patterns after an initial 3D scan could lead the way to new adaptive 3D scanning techniques [125, 74].

Although derived from differentiable imaging formation model, our position-encoding objective function can be viewed as an extremely simple one-layer neural network. Understanding how to best exploit the power of deeper architectures for active triangulation is an exciting direction for future work, which we further explore in next chapter.

Chapter 7

Optimizing Illuminations for Active Imaging Systems with Optical Stochastic Gradient Descent

We consider the problem of optimizing the performance of an active imaging system by automatically discovering the illuminations it should use, and the way to decode them. Our approach tackles two seemingly incompatible goals: (1) “tuning” the illuminations and decoding algorithm precisely to the devices at hand—to their optical transfer functions, non-linearities, spectral responses, image processing pipelines—and (2) doing so without modeling or calibrating the system; without modeling the scenes of interest; and without prior training data. The key idea is to formulate a stochastic gradient descent (SGD) optimization procedure that puts the actual system in the loop: projecting patterns, capturing images, and calculating the gradient of expected reconstruction error. We apply this idea to structured-light triangulation to “auto-tune” several devices—from smartphones and laser projectors to advanced computational cameras. Our experiments show that despite being model-free and automatic, optical SGD can boost system 3D accuracy substantially over state-of-the-art coding schemes.

7.1 Introduction

Fast and accurate structured-light imaging on your desk—or in the palm of your hand—has been getting ever closer to reality over the last two decades [28, 219, 174, 52]. Already, the high pixel counts of today’s smartphones and home-theater projectors theoretically allow 3D accuracies of 100 microns or less. Similar advances are occurring in the domain of time-of-

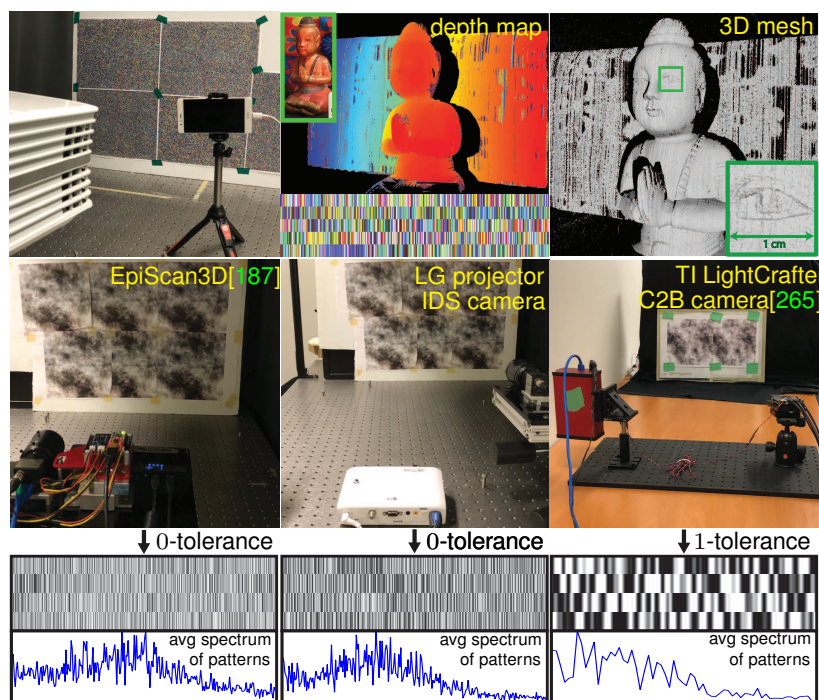


Figure 7.1: **Top:** *Optimal structured light with smartphones.* We placed a randomly-colored board in front of an Optoma 4K projector and a Huawei P9 phone, let them auto-tune for five color-stripe patterns and the 1-tolerance penalty (Table 7.1), and used the resulting patterns (middle) to reconstruct a scene (inset). **Middle & Bottom:** *Auto-tuning systems for 4 patterns on various imaging systems.* Note the patterns’ distinct spatial structure and frequency content, especially for EpiScan3D [187] which employs a scanning-laser projector.

flight (ToF) imaging as well, with inexpensive continuous-wave ToF sensors, programmable lasers, and spatial modulators becoming increasingly available [227, 20, 8, 113, 112, 141, 30, 142, 87]. Unfortunately, despite the wide availability of all these devices, achieving optimal performance with a given hardware system is still an open problem whose theoretical underpinnings have only recently attracted attention [80, 78, 172, 75, 94, 201, 10].

To address this challenge, we introduce *optical SGD*, a computational imaging technique that learns on the fly (1) a sequence of optimized illuminations for multi-shot depth acquisition with a given system, and (2) an optimized reconstruction function for depth map estimation. Optical SGD achieves this by controlling in real-time the system it is optimizing, and capturing images with it. The only inputs to the optimization are the number of shots and a function to penalize depth error at a pixel.

To prepare a system for optical SGD, we adjust its settings for the desired imaging conditions (*e.g.*, exposure time, light source brightness, *etc.*) and place a randomly-textured “training board” in its field of view (Figure 7.1). The process runs automatically after that, minimizing a rigorously-derived estimate of the expected reconstruction error for the system at hand. Optical SGD requires no radiometric or geometric calibration; no manual initialization; no

prior training data; and most importantly, no precise image formation model for the system or the scenes of interest.

The key idea behind our approach is to push the hardest computations in this optimization—*i.e.*, calculating derivatives that depend on an accurate model of the system—to the *optical domain*, where they are easy to do (Figure 7.2). Intuitively, optical SGD treats the imaging system as a perfect “end-to-end model” of itself—with realistic noise and optical imperfections all included.

Using this idea as a starting point, we develop an optimization procedure that runs partly in the numerical and partly in the optical domain. It begins with a random set of K illuminations; uses them to illuminate the training board; captures real images to estimate the gradient of the expected reconstruction error; and updates its illuminations by stochastic gradient descent [124, 246]. Applying this procedure to a given system requires (1) a way to repeatedly acquire higher-accuracy (but still noisy) depth maps of the training board, and (2) programmable light sources that allow small adjustments to their illumination.

At a conceptual level, optical SGD is related to three lines of recent work. First, the end-to-end optimization of computational imaging systems is becoming increasingly popular [123, 95, 31, 229, 236, 248]. These methods train deep neural networks and require precise models of the system or extensive training data, whereas our approach needs neither. Second, the principle of replacing “hard” numerical computations with “easy” optical ones goes back several decades to the field of optical computing [13, 67, 133]. It has been revived recently for calculations such as optical correlation [34], hyperspectral imaging [217] and light transport analysis [189] but we are not aware of any attempts to implement SGD in the optical domain, as we do. Third, optical SGD can also be thought of as training a small, shallow neural network with a problem-specific loss; noisy labels [35, 286, 273] and noisy gradients [183]; and with training and data-augmentation strategies [127, 85] that are implemented partly in the optical domain.

We believe our work represents the first attempt to reduce illumination coding—a hard problem with a rich history [94, 214, 215, 201, 202, 79, 77, 74, 175, 71, 37, 277, 220, 207, 221]—to an online procedure akin to self-calibration [140, 143]. In addition to this basic contribution, we introduce two important new elements to the optimization of structured-light triangulation systems: neighborhood decoding and neural network decoding. The former generalizes the recently-proposed ZNCC decoder [172] to take into account a tiny neighborhood at each pixel (3×1 or 5×1). This seemingly straightforward extension more than doubles per-pixel disparity accuracy in our tests, highlighting the hitherto unnoticed role the decoder can play in per-pixel depth estimation. On the other side, we change the

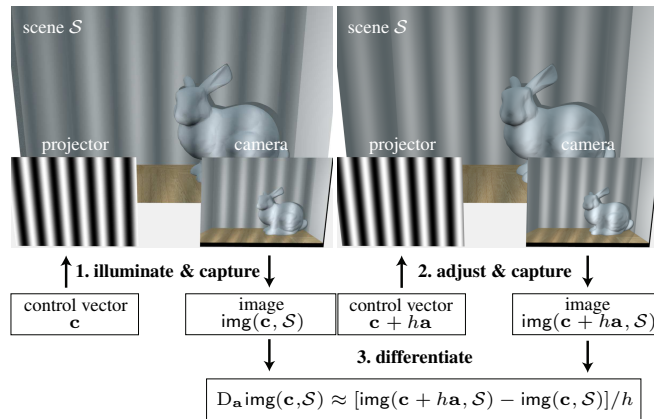


Figure 7.2: **Algorithm Illustration.** Differentiable imaging systems allow us to “probe” their behavior by differentiating them in the optical domain, *i.e.*, by repeatedly adjusting their control vector, taking images, and computing image differences. Projector-camera systems, as shown above, are one example of a differentiable system where projection patterns play the role of control vectors. Many other combinations of programmable sources and sensors have this property (Table 7.1).

fixed ZNCC decoder to a learnable decoder by inserting a 2-layer neural network and convert the raw intensity to high-dimension feature. We find that the decoding accuracy can be further improved when applying ZNCC in feature space.

We first develop our approach in the context of more general 3D imaging systems, and focus specifically on structured-light triangulation in Section 7.4.

7.2 Differentiable Imaging Systems

Many devices available today allow us to control image formation in an extremely fine-grained—almost continuous—manner: off-the-shelf projectors can adjust a scene’s illumination at the resolution of individual gray levels of a single projector pixel; spatial light modulators can do likewise for phase [170] or polarization [176]; programmable laser drivers can smoothly control the temporal waveform of a laser at sub-microsecond scales [113]; and sensors with coded-exposure [287, 265, 188] or correlation [75, 88, 112] capabilities can adjust their spatio-temporal responses at pixel and microsecond scales.

Our focus is on the optimization of programmable imaging systems that rely on such devices for fine-grained control of illumination and sensing. In particular, we restrict our attention to systems that approximate the idealized notion of a *differentiable imaging system*. Intuitively, differentiable imaging systems have the property that a small adjustment to their settings will cause a small, predictable change to the image they output (Figure 7.2):

Definition 1 (Differentiable Imaging System) An imaging system is differentiable if the

following two conditions hold ¹:

- the behavior of its sources, sensors, and/or optics during the exposure time is governed by a single N -dimensional vector, called a *control vector*, that takes continuous values;
- for a stationary scene \mathcal{S} , the directional derivatives of the image with respect to the system's control vector, *i.e.*,

$$D_{\mathbf{a}} \text{img}(\mathbf{c}, \mathcal{S}) \stackrel{\text{def}}{=} \lim_{h \rightarrow 0} \frac{\text{img}(\mathbf{c} + h\mathbf{a}, \mathcal{S}) - \text{img}(\mathbf{c}, \mathcal{S})}{h}, \quad (7.1)$$

are well defined for any control vector \mathbf{c} and unit-length adjustment \mathbf{a} , where $\text{img}(\mathbf{c}, \mathcal{S})$ is the noise-less image.

As we will see in Section 7.3, differentiable imaging systems open the possibility of *optical SGD*—iteratively adjusting their behavior in real time via optical-domain differentiation—to optimize performance on a given task.

The specific task we consider in this chapter is depth imaging. More formally, we seek a solution to the following general optimization problem:

Definition 2 (System Optimization for Depth Imaging) Given

- a *differentiable imaging system* that outputs a noisy intensity image \mathbf{i}_k in response to a control vector \mathbf{c}_k ;
- a *differentiable decoder* that estimates a depth map \mathbf{d} from a sequence of $K \geq 1$ images acquired with control vectors $\mathbf{c}_1, \dots, \mathbf{c}_K$:

$$\mathbf{d} = \text{rec}(\mathbf{i}_1, \mathbf{c}_1, \dots, \mathbf{i}_K, \mathbf{c}_K, \theta) \quad (7.2)$$

where θ is a vector of additional tunable parameters; and

- a *pixel-wise penalty function* $\rho()$ that penalizes differences between the estimated depth map \mathbf{d} and the ground-truth depth map \mathbf{g} ,

compute the settings that minimize expected reconstruction error:

$$\hat{\mathbf{c}}_1, \dots, \hat{\mathbf{c}}_K, \hat{\theta} = \arg \min_{\mathbf{c}_1, \dots, \mathbf{c}_K, \theta} \mathbb{E}_{\text{scenes, noise}} \left[\sum_{m=1}^M \rho(\mathbf{d}[m] - \mathbf{g}[m]) \right] \quad (7.3)$$

where the index m ranges over image pixels and expectation is taken over noise and a space of plausible scenes.

Different combinations of light source, sensor, decoder and penalty function lead to different

¹Here, we specifically define differentiable imaging system for the active imaging system, *i.e.*, the structured light system.

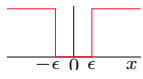
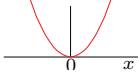
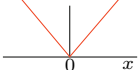
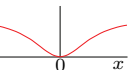
Light source	Camera sensor	Decoder	
†DMD projector [126]	†Grayscale	†max-ZNCC [172]	
†Laser projector [187]	†RGB filter	†max-ZNCC _p (Sec. 7.4)	
LCoS projector [47]	†Coded-exposure [287]	†max-ZNCC _p -NN (Sec. 7.4)	
Projector array [138]	Correlation ToF [75]	deep neural net [55]	
MHz laser [113]	ToF sensor array [227]		
MHz laser + DMD [188]	Light field [139]		
MHz laser array [245]			
† ϵ -tolerance [172]	L_2 [94]	† L_1 [78]	M-estimator [292]
			

Table 7.1: Devices and penalty functions compatible with our framework. † indicates the choices we validate experimentally.

instances of the system optimization problem (Table 7.1). Correlation time-of-flight (ToF) systems, for example, capture $K \geq 3$ images of a scene, and vectors $\mathbf{c}_1, \dots, \mathbf{c}_K$ control their associated laser modulation and pixel demodulation functions [75, 113]. In active triangulation systems that rely on K images to compute depth, the control vectors are simply the projection patterns (Figure 7.2). In both cases, the decoder maps the K observations at each pixel to a depth (or stereo disparity) value. The common penalty function choices could be ϵ -tolerance [172], L_2 [94], L_1 [78], and M-estimator [292]. Each optimizes reconstruction error under different metric. For instance, ϵ -tolerance tries to minimize correspondence error less than ϵ while L_1 minimizes average correspondence error. In the following we use a vector-valued function $\text{err}(\mathbf{d}, \mathbf{g})$ to collect all pixel-wise penalties into a single vector:

$$\text{err}(\mathbf{d}, \mathbf{g})[m] = \rho(\mathbf{d}[m] - \mathbf{g}[m]) . \quad (7.4)$$

7.3 Optical SGD Framework

Suppose for a moment that we have a *perfect* forward model for the image formation process, *i.e.*, we have a perfect model for (1) the system’s light sources, optics, and sensors, (2) the scenes to be imaged, and (3) the light transport between them.

In that case, the widespread success of optimization techniques such as Stochastic Gradient Descent (SGD) [209, 246, 124] suggest a way to minimize our system-optimization objective numerically: approximate it by a sum of reconstruction errors for a large set of fairly-drawn, synthetic training scenes, and for realistic noise; find a way to efficiently evaluate its gradient with respect to the unknowns $\theta, \mathbf{c}_1, \dots, \mathbf{c}_K$; and apply SGD to (locally) minimize it.

Model-driven Optimization by Numerical SGD Approximating the expectation in Eq. (7.3)

Numerical SGD:	Optical SGD:
Input: scene generator, noise generator, evaluator of $\text{img}(\mathbf{c}, \mathcal{S})$, $\mathbf{J}(\mathbf{c}, \mathcal{S})$	Input: $\langle \text{none} \rangle$
Output: optimal $\theta, \mathbf{c}_1, \dots, \mathbf{c}_K$	Output: optimal $\theta, \mathbf{c}_1, \dots, \mathbf{c}_K$
initialize with random $\theta, \mathbf{c}_1, \dots, \mathbf{c}_K$	initialize with random $\theta, \mathbf{c}_1, \dots, \mathbf{c}_K$
generate scenes $\mathcal{S}^1, \dots, \mathcal{S}^T$	position in front of system a scene \mathcal{S}
while not converged do	while not converged do
choose random mini-batch of scenes	choose random mini-batch of image rows compute their ground-truth depth map \mathbf{g}
for each scene \mathcal{S} in mini-batch do	for each control vector \mathbf{c}_k do
for each control vector \mathbf{c}_k do	supply control vector \mathbf{c}_k to system capture image & store it in \mathbf{i}_k
synthesize image \mathbf{i}_k by evaluating $\text{img}(\mathbf{c}_k, \mathcal{S})$ & adding noise	estimate \mathbf{d} from $\mathbf{i}_1, \dots, \mathbf{i}_K$ evaluate $\text{err}(\mathbf{d}, \mathbf{g})$ on mini-batch evaluate $\nabla_{\theta} \text{err}(\mathbf{d}, \mathbf{g})$ on mini-batch
estimate \mathbf{d} from $\mathbf{i}_1, \dots, \mathbf{i}_K$	for all k , compute $\mathbf{J}(\mathbf{c}_k, \mathcal{S})$ optically & use it to evaluate $\nabla_{\mathbf{c}_k} \text{err}(\mathbf{d}, \mathbf{g})$
evaluate $\text{err}(\mathbf{d}, \mathbf{g})$	evaluate total gradient using Eq.(7.5) update $\theta \leftarrow \theta + \Delta\theta, \mathbf{c}_k \leftarrow \mathbf{c}_k + \Delta\mathbf{c}_k$
evaluate $\nabla_{\theta} \text{err}(\mathbf{d}, \mathbf{g})$	apply constraints to $\theta, \mathbf{c}_1, \dots, \mathbf{c}_K$
for all k , evaluate $\nabla_{\mathbf{c}_k} \text{err}(\mathbf{d}, \mathbf{g})$	return $\theta, \mathbf{c}_1, \dots, \mathbf{c}_K$
evaluate total gradient using Eq.(7.5)	
update $\theta \leftarrow \theta + \Delta\theta, \mathbf{c}_k \leftarrow \mathbf{c}_k + \Delta\mathbf{c}_k$	
apply constraints to $\theta, \mathbf{c}_1, \dots, \mathbf{c}_K$	
return $\theta, \mathbf{c}_1, \dots, \mathbf{c}_K$	

Figure 7.3: Numerical vs. Optical-domain Implementation of SGD, with Red Boxes Highlighting Their Differences.

with a sum we get:

$$\mathbb{E}_{\text{scenes, noise}} \left[\sum_{m=1}^M \rho(\mathbf{d}[m] - \mathbf{g}[m]) \right] \approx \frac{1}{T} \sum_{t=1}^T \|\text{err}(\mathbf{d}^t, \mathbf{g}^t)\|_1 \quad (7.5)$$

where $\|\cdot\|_1$ denotes the L_1 norm of a vector and $\mathbf{d}^t, \mathbf{g}^t$ are the reconstructed shape and ground-truth shape for the t -th training sample, respectively. Each training sample consists of a scene \mathcal{S}^t and the noise present in the images $\mathbf{i}_1, \dots, \mathbf{i}_K$ acquired for that scene. Figure 7.3 (left) outlines the basic steps of the resulting numerical SGD procedure.

Optical Computation of the Image Jacobian What if we do not have enough information about the imaging system and its noise properties to reproduce them exactly, or if the forward image formation model is too complex or expensive to simulate? Fortunately, differentiable imaging systems allow us to overcome these limitations by implementing the difficult gradient calculations directly in the optical domain.

More specifically, SGD requires evaluation of the gradient with respect to θ and $\mathbf{c}_1, \dots, \mathbf{c}_K$

of vector $\text{err}(\mathbf{d}^t, \mathbf{g}^t)$:

$$\nabla_{\theta} \text{err} = \frac{\partial \text{err}}{\partial \text{rec}} \frac{\partial \text{rec}}{\partial \theta} \quad (7.6)$$

$$\nabla_{\mathbf{c}_k} \text{err} = \frac{\partial \text{err}}{\partial \text{rec}} \frac{\partial \text{rec}}{\partial \mathbf{c}_k} + \frac{\partial \text{err}}{\partial \text{rec}} \frac{\partial \text{rec}}{\partial \mathbf{i}_k} \frac{\partial \mathbf{i}_k}{\partial \mathbf{c}_k} \quad (7.7)$$

$$\approx \frac{\partial \text{err}}{\partial \text{rec}} \frac{\partial \text{rec}}{\partial \mathbf{c}_k} + \frac{\partial \text{err}}{\partial \text{rec}} \frac{\partial \text{rec}}{\partial \mathbf{i}_k} \underbrace{\left(\frac{\partial \text{img}}{\partial \mathbf{c}} \right)_{\substack{\mathbf{c}=\mathbf{c}_k \\ \mathcal{S}=\mathcal{S}^t}}}_{\text{image Jacobian } \mathbf{J}(\mathbf{c}, \mathcal{S}) \text{ for } \mathbf{c}_k \text{ and } \mathcal{S}^t} \quad (7.8)$$

with points of evaluation omitted for brevity. Eq. (7.8) is obtained by approximating \mathbf{i}_k with its noise-less counterpart. Of all the individual terms in Eqs. (7.6)-(7.8), only one depends on a precise model of the system and scene: the *image Jacobian* $\mathbf{J}(\mathbf{c}, \mathcal{S})$.

For a system that captures an M -pixel image in response to an N -element control vector, $\mathbf{J}(\mathbf{c}, \mathcal{S})$ is an $M \times N$ matrix. Intuitively, element $[m, n]$ of this matrix tells us how the intensity of image pixel m will change if element n of the control vector is adjusted by an infinitesimal amount. As such, it is related to the system's directional image derivatives (Eq. (7.1)) by a matrix-vector product:

$$\mathbf{D}_a \text{img}(\mathbf{c}, \mathcal{S}) = \mathbf{J}(\mathbf{c}, \mathcal{S}) \mathbf{a} . \quad (7.9)$$

It follows that if we have physical access to both a differential imaging system and a scene \mathcal{S} , we can compute individual columns of this matrix without having any computational model of the system or the scene. All we need is to implement a discrete version of Eq. (7.9) in the optical domain, as illustrated in Fig. 7.2 with a projector-camera system. This leads to the following “optical subroutine:”

Optical-domain computation of n -th column of $\mathbf{J}(\mathbf{c}, \mathcal{S})$

Input: control vector \mathbf{c} , adjustment magnitude h

Output: noisy estimate of the column

step 0: position scene \mathcal{S} in front of system

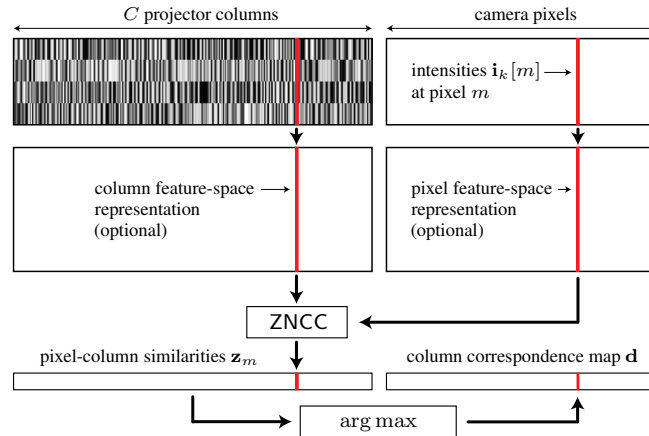
step 1: set control vector to \mathbf{c} and capture noisy image \mathbf{i}

step 2: set control vector to $\mathbf{c} + h\mathbf{a}$, where \mathbf{a} is the unit vector along dimension n , and capture new image \mathbf{i}'

step 3: return $(\mathbf{i}' - \mathbf{i})/h$

step 4: (optional) repeat steps 1 & 2 to get multiple samples of \mathbf{i} and \mathbf{i}' & return the empirical distribution of $(\mathbf{i}' - \mathbf{i})/h$

Optical SGD The above subroutine makes it possible to turn numerical SGD—which depends on system and scene models—into an optical algorithm that is model free. To do this, we replace with image-capture operations all steps in Figure 7.3 (left) that require

Figure 7.4: **Decoder for K -pattern Triangulation.**

modeling of systems and scenes.²

Practical implementations of optical SGD face three challenges: (1) a closed-form expression must be derived for a scene’s expected reconstruction error (Eq. (7.5)) in order to evaluate its gradient, (2) imaging a large set of real-world training scenes is impractical, and (3) the image Jacobian is too large to acquire by brute force. Below we address these challenges by exploiting the structure of the system-optimization problem specifically for triangulation-based systems. The resulting optical SGD procedure is shown in Figure 7.3 (right).

7.4 Auto-Tuning Structured Light

We now turn to the problem of optimizing projector-camera systems for structured-light triangulation (Figure 7.2). In this setting, c_1, \dots, c_K represent 1D patterns projected sequentially onto a scene and the reconstruction task is to compute, independently for every camera pixel, its stereo correspondence on the projector plane. This task is equivalent to computing the *pixel-to-column correspondence map* \mathbf{d} , where $\mathbf{d}[m]$ is the projector column that contains the stereo correspondence of camera pixel m (Figure 7.4). We thus optimize a projector-camera system by minimizing errors in \mathbf{d} .³ Furthermore, we define the *disparity* of pixel m to be the difference of $\mathbf{d}[m]$ and the pixel’s column on the image plane.

Image Jacobian of Projector-camera Systems We treat projectors and cameras as two non-linear “black-box” functions $\text{proj}()$ and $\text{cam}()$, respectively (Figure 7.5). These account for device non-linearities as well as internal low-level processing of patterns and images (*e.g.*,

²Since optical-domain Jacobian estimation relies on noisy images, it introduces an additional source of stochasticity in the SGD procedure [124, 246, 273, 181, 183, 35].

³The pixel-to-column correspondence map requires no knowledge of a system’s epipolar geometry, radial distortion or Euclidean calibration. As a result, optical SGD can be applied even without this information.

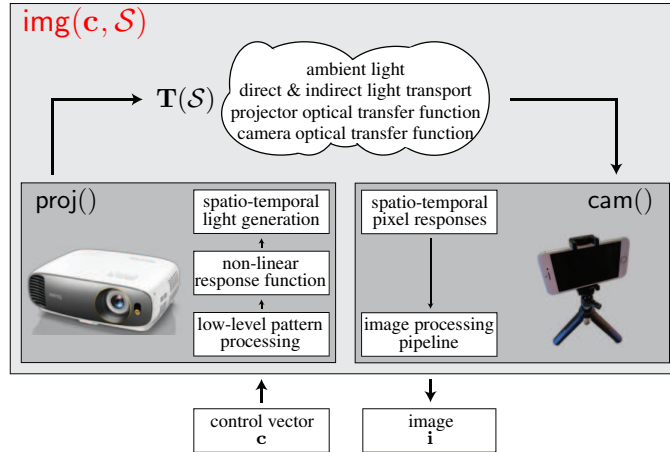


Figure 7.5: **Image Formation in General Projector-camera Systems.** The projector function $\text{proj}()$ maps a control vector of digital numbers to a vector of outgoing radiance values. Similarly, the camera function $\text{cam}()$ maps a vector of sensor irradiances to a vector holding the processed image.

non-linear contrast enhancement, color processing, demosaicing, *etc.*).

Between the two, light propagation is linear and can thus be modeled by a transport matrix $\mathbf{T}(\mathcal{S})$. This matrix is unknown and generally depends on the scene's shape and material properties, as well as the system's optics [79, 172]. It follows that the image and its Jacobian are given by

$$\mathbf{i} = \underbrace{\text{cam}(\mathbf{T}(\mathcal{S}) \text{proj}(\mathbf{c}) + \text{ambient})}_{\text{img}(\mathbf{c}, \mathcal{S})} + e \quad (7.10)$$

$$\mathbf{J}(\mathbf{c}, \mathcal{S}) = \underbrace{\frac{\partial \mathbf{c}}{\partial \text{irr}}}_{\substack{\text{camera} \\ \text{non-linearities} \\ (M \times M)}} \quad \mathbf{T}(\mathcal{S}) \quad \underbrace{\frac{\partial \text{proj}}{\partial \mathbf{c}}}_{\substack{\text{projector} \\ \text{non-linearities} \\ (N \times N)}} \quad (7.11)$$

optics, 3D shape, reflectance, *etc.* $(M \times N)$

where noise may include a signal-dependent component and irr denotes the vector of irradiances incident on the camera's pixels. Thus, auto-tuning a system without indirect light forces it to account for its non-linearities and end-to-end optical transfer function.

Neighborhood Decoding For perfectly linear systems and low signal-independent noise, a very simple correspondence-finding algorithm was recently shown to be optimal in a maximum-likelihood sense [172]: (1) treat the intensities $\mathbf{i}_1[m], \dots, \mathbf{i}_K[m]$ observed at pixel m as a K -dimensional “feature vector,” (2) compare it to the vector of intensities at each projector column, and (3) choose the column that is most similar according to the zero-mean

normalized cross-correlation (ZNCC) score⁴ (Figure 7.4):

$$\mathbf{z}_m[n] \stackrel{\text{def}}{=} \text{ZNCC}([\mathbf{i}_1[m], \dots, \mathbf{i}_K[m]], [\mathbf{c}_1[n], \dots, \mathbf{c}_K[n]]) \quad (7.12)$$

$$\mathbf{d}[m] = \arg \max_{1 \leq n \leq N} \mathbf{z}_m[n] . \quad (7.13)$$

Here we generalize this decoder in three ways. First, we expand feature vectors to include their $1 \times p$ neighborhood (on the same image row as pixel m , in the case of images). We use small, 3- or 5-pixel neighborhoods in our experiments, making it possible to exploit intensity correlations that may exist in them:

$$\text{(ZNCC}_p \text{ similarity)} \quad \mathbf{z}_m[n] = \text{ZNCC}(\mathbf{f}_m, \hat{\mathbf{f}}_n) \quad (7.14)$$

where $\mathbf{f}_m, \hat{\mathbf{f}}_n$ are vectors collecting these intensities. Second, we model the projector’s response curve as an unknown monotonic, scalar function $g(\cdot)$ consisting of 32 linear segments [96]. This introduces a learnable component to the decoder, whose 32-dimensional parameter vector θ is optimized by optical SGD along with $\mathbf{c}_1, \dots, \mathbf{c}_K$. Third, we add a second learnable component to better exploit neighborhood correlations, and to account for noise and system non-linearities that cannot be captured by the scalar response $g(\cdot)$ alone. This consists of two ResNet blocks [85, 86], for the camera and projector, respectively:

$$\text{(ZNCC-NN}_p \text{ similarity)} \quad \mathbf{z}_m[n] = \text{ZNCC}(\mathbf{f}_m + \mathcal{F}(\mathbf{f}_m), g(\hat{\mathbf{f}}_n) + \hat{\mathcal{F}}(g(\hat{\mathbf{f}}_n))) \quad (7.15)$$

where $\mathcal{F}(\cdot)$ and $\hat{\mathcal{F}}(\cdot)$ are neural nets with two fully-connected layers of dimension $(pK) \times (pK)$ and a ReLU in between. Thus the total number of learnable parameters in the decoder—and thus in vector θ —is $4p^2K^2 + 32$.⁵

Optimization with Penalty Functions Optimizing the expected reconstruction error of Eq. (7.5) requires a differentiable estimate of the total penalty, $\|\text{err}(\mathbf{d}, \mathbf{g})\|_1$, incurred on a given scene. A tight closed-form approximation can be expressed in terms of the scene’s ground-truth correspondence map \mathbf{g} , the ZNCC score vectors of all pixels, and the vector of pixel-wise penalties:

$$\|\text{err}(\mathbf{d}, \mathbf{g})\|_1 \approx \sum_{m=1}^M \text{softmax}(\mu \mathbf{z}_m) \cdot \text{err}(\text{index} - \mathbf{g}[m], \mathbf{0}) \quad (7.16)$$

where \cdot denotes dot product; μ is the softmax temperature; \mathbf{z}_m is given by Eqs. (7.12)-(7.15); $\mathbf{0}$ is the zero vector; and **index** is a vector whose i -th element is equal to its index i . Proof is provided in the Appendix Sec. C.1.

⁴For two vectors $\mathbf{v}_1, \mathbf{v}_2$, their ZNCC score is the normalized cross correlation of $\mathbf{v}_1 - \text{mean}(\mathbf{v}_1)$ and $\mathbf{v}_2 - \text{mean}(\mathbf{v}_2)$.

⁵Strictly speaking, ZNCC’s optimality does not carry over to ZNCC_p, ZNCC-NN_p or general non-linear systems. Nevertheless, we use them for optical SGD as we found these similarities to be very effective empirically.

7.4.1 Efficient Optical-Domain Implementation

Different Rows \Leftrightarrow Different Training Scenes Suppose we place an object in front of the system whose ground-truth correspondence map, \mathbf{g} , is known. In principle, since the column correspondence of each camera pixel must be estimated independently of all others, each pixel can be thought of as a separate instance of the depth estimation task. To reduce correlations between these instances we use randomly-textured boards for training (Figure 7.1). This allows us to treat each camera row as a different “training scene” that consists of points with randomly-distributed albedos.

Circular Pattern Shifts \Leftrightarrow Different Scene Depths While the albedo of scene points in the system’s field of view may be random, their depth is clearly not: since our training boards are nearly planar and (mostly) stationary, the pixel-to-column correspondence map varies smoothly across rows and is fixed in time. To break their temporal continuity we move the patterns instead of the scene: we apply the same randomly-chosen circular shift to all K projection patterns prior to projection and image capture, and alter that shift every few iterations. This changes the pixel-to-column correspondence map, and results in images that would have been obtained had the *scene* moved in depth.⁶ It also allows optimization of patterns that span all columns of a projector even when the training scene does not.

Acquisition of Ground-truth Correspondences Optical SGD hinges on being able to compute ground truth far more accurately than the procedure it is optimizing. Since our focus is on optimizing systems for minimal numbers of patterns, we use the same system for ground-truth estimation but with many more patterns. We first assess a system’s maximum attainable accuracy and precision by reconstructing a training board repeatedly with two independent coding schemes—160 phase-shifted patterns [214] and 30 patterns optimized for the 0-tolerance penalty [172]—and cross-validating both across runs and across coding schemes.

Specifically, We acquire the ground truth with the following routine:

1. Project 30 a la carte (0-tolerance) [172] patterns onto the scene, and use the ZNCC decoder to compute a pixel-to-column correspondence map.
2. Repeat the previous step 10 times to capture 10 maps.
3. Fuse the captured maps into a single map \mathbf{g}_1 by majority vote.

To verify the captured ground truth, we compare it with another map acquired with a different family of patterns:

1. Project 160 shifted cosine patterns (10 shifts for frequencies in the range of [16-31]) along with 11 shifted cosine patterns of frequency $\frac{N}{11}$, where N is the number of

⁶Note that this “simulation” does not account for signal-to-noise ratio reductions caused by the squared-distance falloff of irradiance.

projector columns.

2. Compute a sub-pixel map [74].
3. Round the fractional correspondences to the nearest integer, to obtain an integer pixel-to-column correspondence map \mathbf{g}_2 .

We say that the pixel-to-column correspondence map \mathbf{g}_1 captured by the first routine is an ϵ -tolerance ground truth if no pixel in \mathbf{g}_1 and \mathbf{g}_2 differs by more than ϵ ; *i.e.* for each image pixel, the column correspondences captured by these two routines are at most ϵ columns apart from each other. In our experiments, we were able to acquire 0-tolerance ground truth for the training board and our test scenes with all the imaging systems used in this work except the indirect light experiment in Fig. 7.9. For that experiment only 1-tolerance ground truth was possible.

Although an accurate ground truth is required for evaluation of different systems and for comparison to prior art, we do not need the most accurate ground truth while auto-tuning a system. This is because small errors can introduce an additional source of stochasticity, and helps the optimization not to overfit. Therefore, during training, the optical SGD (outlined in Figure 7.3) uses just one set of 30 a la carte patterns. We re-compute the ground truth every 50 iterations to account for minor disturbances (*e.g.*, slight motions of the board or the camera).

Efficient Acquisition of Image Jacobians Although the Jacobian is large, it is usually very sparse for scenes without indirect light transport (*e.g.*, our training boards). This makes it possible to acquire several columns of the Jacobian at once from just one invocation of the optical-domain subroutine of Section 7.3. In particular, an adjustment vector with N/B equally-spaced non-zero elements will yield the sum of N/B columns of the Jacobian. If B is large enough to avoid overlap between the non-zero elements of these columns, exact recovery is possible.

Figure 7.6 (third column) illustrates one sample adjustment vector and its corresponding image difference, where the adjustment vector has N/B equally-spaced non-zero elements. We empirically set $B = 7$ and fix the adjustment magnitude h to 0.15. This adjustment magnitude provides enough change in the image while being small enough to approximate the directional derivatives.

Frequency Constraint We always enforce a weak max-frequency constraint to pattern optimization because we found that it leads to a more stable optimization. When auto-tuning on board, in particular, we band-limit patterns to frequency $2^{\lfloor \log_2 N/4 \rfloor}$, which is the closest power of 2 to half the Nyquist rate. This limit is fixed in all our experiments, with the exception of auto-tuning for indirect light (see Fig. 7.9 and Sec. 7.5.2). Enforcing the max

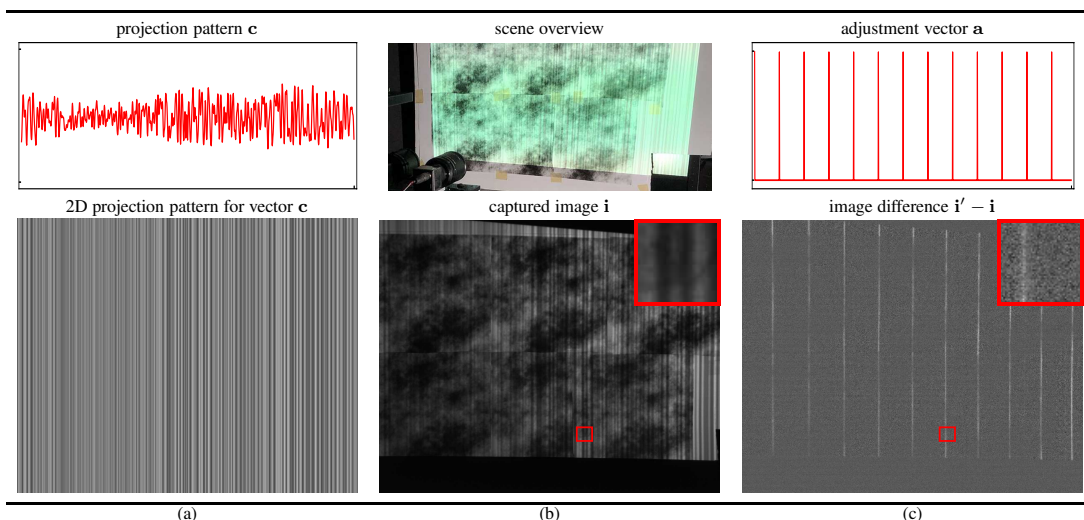


Figure 7.6: **Optical-domain Differentiation on an Actual Projector-camera System.** (a) 1D plot of a sample projection pattern c and its corresponding 2D projection pattern. (b) a photo of the experimental setup and the corresponding camera image of the scene under the projection pattern depicted in (a). (c) adjustment vector and its corresponding image difference.

frequency constraint is done in the Fourier domain; at the end of each optical SGD iteration, we set Fourier coefficients higher than the specified max frequency to zero.

Numerical Considerations We adopt RMSprop [246] and Tensorflow [7] for the numerical loop of Optical SGD. The learning rate is set to 0.001, and allowed to decay by 50% every 350 iterations. We use a softmax temperature of $\mu = 200$, a step size of $B = 7$ for Jacobian acquisition, and initialize patterns with uniform noise in the range $[0.45, 0.55]$. Mini-batches are created by randomly choosing 15% of image rows in each SGD iteration. We estimate ground-truth correspondences and the image Jacobian every 50 and 15 iterations, respectively. To ensure a stable optimization, we band-limit the patterns' frequency to $1/2$ Nyquist for the projector being used. Optimization typically converges in 1000 iterations and takes approximately one hour. The main bottleneck is image acquisition, which runs at 15Hz for our unsynchronized HDMI-driven devices.

Figure 7.7 illustrates how the patterns (and decoder) evolve in a sample auto-tuning run—with marked improvements in reconstruction performance over time. Crucially, performance on a much darker scene with lots of depth discontinuities shows a similar trend, suggesting lack of over-fitting.

7.4.2 Optical SGD for Low-SNR Scenes

Auto-tuning a system for low-SNR conditions (*e.g.* low projector brightness, distant scene and/or high ambient light) introduces a major challenge: the image difference $i' - i$ due to adjustment (Figure 7.6(c)) may be below the image quantization level. This will lead to incorrect gradient estimates that can cause optical SGD to fail. To cope with this challenge,

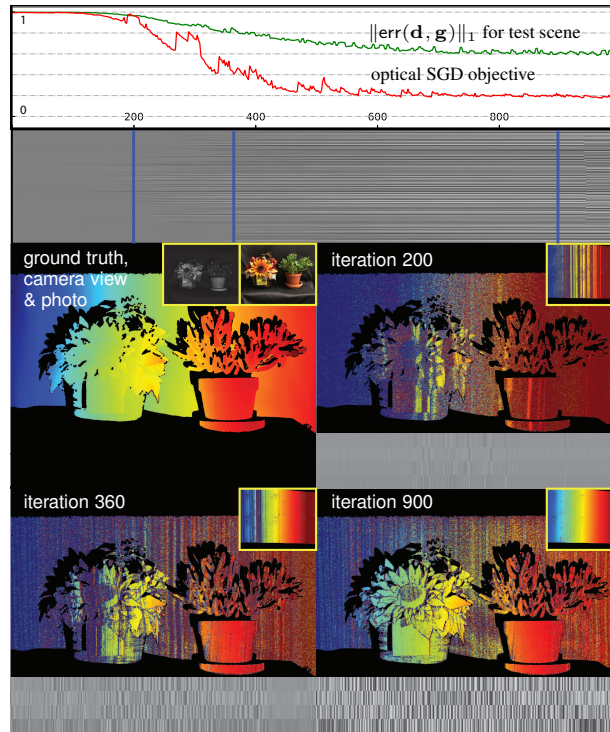


Figure 7.7: **Optical SGD in Action for the LG-IDS Pair and the Training Board in Figure 7.1.** **Top:** The red graph shows the progress of the optimization objective (Eq. 7.5) across iterations when auto-tuning on the training board for four patterns, the zero-tolerance penalty, and the ZNCC-NN₃ decoder. The green graph shows $\|\text{err}(\mathbf{d}, \mathbf{g})\|_1$ as a function of iteration for the previously-unseen (and much more challenging) test scene below. **Middle:** Visualizing the evolution of pattern \mathbf{c}_1 as a grayscale image whose i -th column is the pattern at iteration i . **Bottom:** Three snapshots of the optimization, each showing the patterns at iteration i ; the disparity map of the training board (inset) reconstructed from those patterns; and the disparity map of the test scene reconstructed from the same patterns.

we do not auto-tune systems under these conditions. Instead we always capture images under conditions where the image difference $\mathbf{i}' - \mathbf{i}$ can be computed reliably, and then corrupt these images by (1) down-scaling them and adding zero-mean Gaussian noise to simulate reduced signal level, and (2) adding Poisson noise to simulate high ambient-light contributions. These unquantized images are then used to compute the gradients required by optical SGD.

Intuitively, this approach leads to images with lower quantization error, so that very small image gradients can still be estimated. We use this method only for auto-tuning systems for far-field imaging (Fig. 7.13, and Sec. 7.5.4). In all other cases, auto-tuning is performed without any image corruption.

7.4.3 Optical SGD for Scenes with Indirect Light

Auto-tuning a system for settings with significant indirect light requires specific considerations as well.

Capturing the Ground Truth It is well-known that indirect light can corrupt shape estimates obtained by structured light [175, 74]. In such cases, simply increasing the number of structured light patterns does not guarantee that correct correspondences will be acquired. To mitigate the influence of indirect light as much as possible, we used the same method as described in Sec. 7.4.1 to obtain ground truth, but captured the images by operating EpiScan3D in epipolar-only imaging mode [187]. Note that this mode was employed only for acquiring the ground truth.

Optimization Parameters Indirect light leads to a less-sparse Jacobian. Consequently, we use larger spacing between non-zero elements of the adjustment vector. Furthermore, we observed that using a lower max-frequency for patterns yields a more stable optimization. We empirically set F to $\frac{1}{8}$ of Nyquist limit, and chose $B = 23$ for translucent training scene (refer to Sec. 7.5.2).

Hadamard Multiplexing for Image Jacobian Acquisition Indirect light reduces the SNR of direct surface reflections and therefore the SNR of optically-computed gradients. To improve the gradient estimation, we acquired the image Jacobian by Hadamard multiplexing [221] which is known to improve performance in low-SNR conditions. Specifically, instead of using adjustment vectors that have a single non-zero element, we use those defined by the S-matrix of appropriate size, and demultiplex the captured images after acquisition. This leads to higher-SNR gradients and improves the performance of optical SGD considerably (Fig. 7.9 and Sec. 7.5.2).

7.4.4 Optical SGD for Color Structured Light Systems

To apply the optical auto-tuning framework to color structured light systems, we followed two approaches:

Approach 1: Color Structured Light with Demosaiced RGB Images. First, we use demosaiced RGB images as the input to the algorithm ⁷. For such a system running on K' color patterns, there are $K = 3K'$ control vectors, and $3K'$ corresponding images. With this modification, the same methods as Eq. (7.12)-(7.16), can be applied for both decoding and auto-tuning a system.

Approach 2: Color Structured Light with Raw Single-channel Images. As another option, we simply use raw single-channel Bayer image as the input. Operating on Bayer mosaic introduces a mismatch between the number of control vectors and the number of patterns: for K' RGB patterns, the number of captured images is K' but the number of control vectors is $K = 3K'$. This size mismatch between “image feature vector” and “projector feature vector” (as defined in section 4) prevents direct application of the ZNCC or ZNCC_p decoders

⁷We capture raw single-channel images with camera, and use standard OpenCV algorithm for demosaicing.

(Eq. (7.12)-(7.14)). Here we simply modify ZNCC-NN_p for this purpose, by slightly changing the neural network architecture for image feature vector $\mathcal{F}()$. Specifically, we define $\mathcal{F}()$ to be a neural net with two fully connected layers of dimensions $(pK) \times (3pK) \times (3pK)$. Moreover, since each pixel in a 2×2 Bayer tile has a different spectral response and a different local neighbourhood, we use a distinct neural network for each pixel in a Bayer tile. This means the total number of learnable parameters for image features is $4(3p^2K^2 + 9p^2K^2)$. Eq. (7.16) then can be utilized to auto-tune the patterns and decoder.

7.5 Experiments

7.5.1 Quantitative Comparison with State-of-the-art Methods

Experimental Setup We used the LG-IDS pair as our imaging system. The scene was placed approximately 1m from the system. We chose a scene with both high- and low-albedo surfaces, different geometries, texture, and depth discontinuities. An image of the scene under all-white projection pattern is shown in Figures 7.8.

We used all the decoding methods with $K = 4$ to compare their performance. Fig. 7.8 shows the comparisons for patterns auto-tuned with 0-tolerance. In Figure 7.8, we used percentage of pixels with no error as the evaluation metric, which is equivalent to 0-tolerance penalty. In both figures, The improvements along the columns (marked with green) show the impact of our proposed decoding algorithms on existing encoding methods, while the last row showcases this work’s contributions in both decoding and pattern optimization.

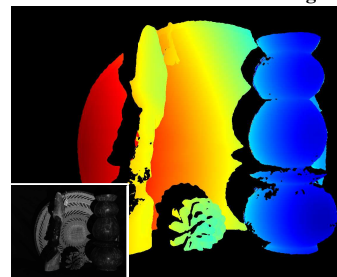
Three observations can be made about these results. First, despite being automatic and calibration free, optical SGD yields state-of-the-art performance. Second, comparing the second columns of Figures 7.8 with their first columns indicates neighbourhood decoding has a drastic impact on the performance of system for all the patterns. For instance, it improves the percentage of zero-error pixels for MPS, Hamiltonian, and a la carte by 89%, 56%, and 86%, respectively. Furthermore, the neural net decoder provides another level of improvement in the performance for auto-tuned patterns, but has almost no effect on patterns which are not optimized with auto-tuning. This suggests that the neural network decoder is most useful when it is optimized along with the patterns. It also justifies our default choice of ZNCC_p decoder for state-of-the-art patterns used in Section 7.5.

7.5.2 Indirect Light Experiment

Experimental Setup We used EpiScan3D [187] (operated as a conventional projector-camera system) to reconstruct a scene made of beeswax and other translucent materials, approximately 80cm away. Ground truth for auto-tuning was acquired as described in Sec. 7.4.3.

	% of pixels with no error		
	ZNCC	ZNCC ₅	ZNCC-NN ₅
MPS [74]	27.42	51.70	49.53
Hamiltonian [78]	9.46	14.72	15.03
a la carte [172]	33.14	61.78	61.76
auto-tuned	32.29	67.33	72.24

Ground Truth & Camera Image



visualization: 0-tolerance disparity maps overlaid with raw disparity

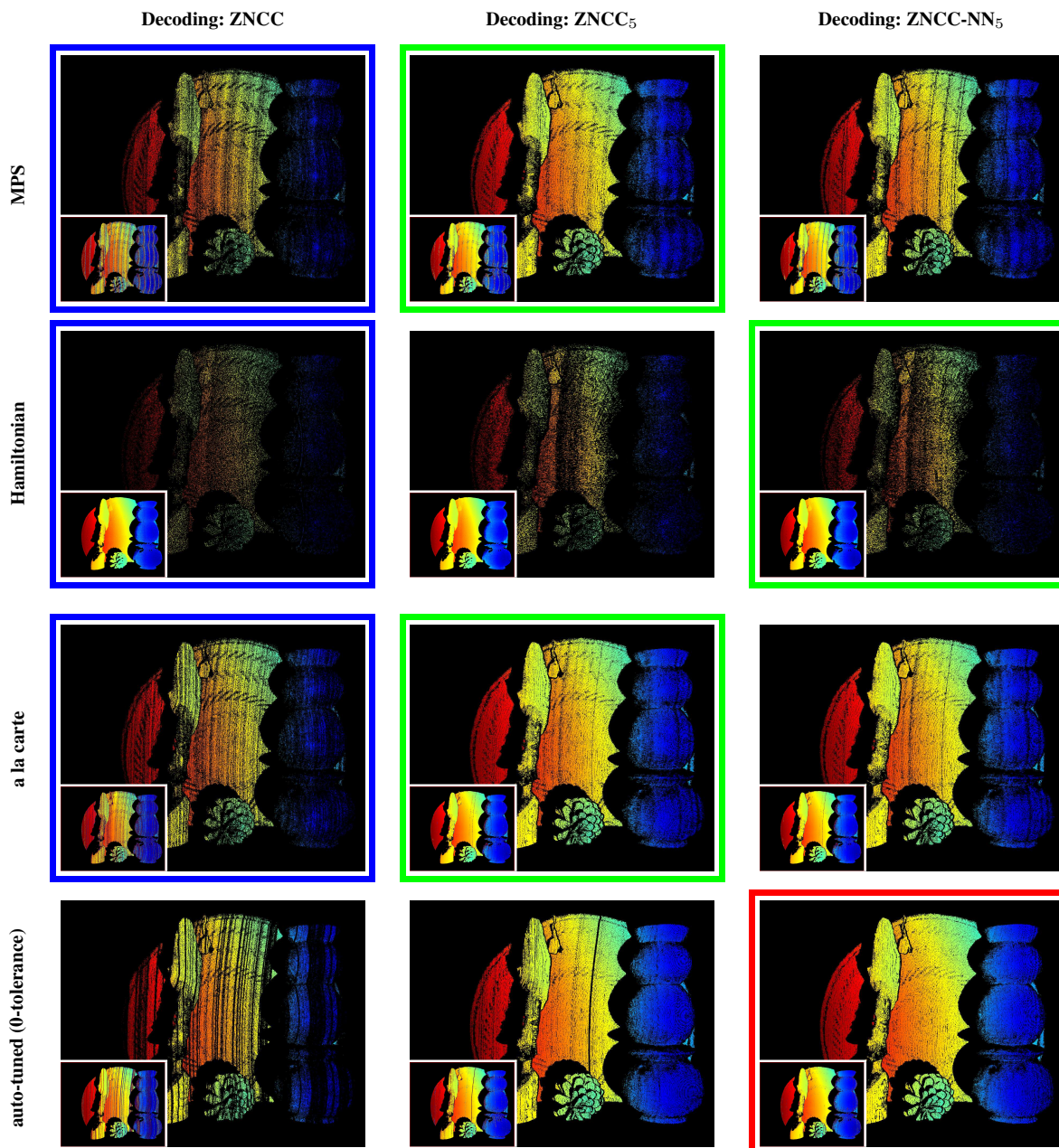


Figure 7.8: Comparison of Different Coding-decoding Methods based on Percentage of Zero-error Pixels. The table lists the no-error reconstruction rates (*i.e.* the percentage of pixels with zero error) of each pattern, decoded with all the decoding methods. Their corresponding 0-tolerance disparity maps (pixels with zero error are drawn in the figures) are shown with the same layout as the table. The raw disparity maps are also shown as insets, for reference. Table entries and disparity maps marked as blue represent the current state of the art. Entries and disparity maps marked as green indicate the best-performing decoder for 3 previously-proposed pattern sequences (MPS, Hamiltonian, a la carte). Note that the best results of these patterns are obtained with decoders introduced in this paper. The best performance, shown in red, is obtained by auto-tuning with the ZNCC-NN₅ decoder.

As a baseline, we auto-tuned with the training board for the 2-tolerance penalty and ZNCC- NN_5 decoder, and used it to reconstruct the scene. This produced a result considerably worse than the MPS16-ZNCC $_5$ decoder combination. (Figure 7.9). Auto-tuning with a beeswax training scene at a similar distance improved performance significantly (75% of pixels with error ≤ 2) but did not outperform MPS16. We then made three small changes to the auto-tuning procedure: (1) bringing the training scene closer (40cm); (2) using Hadamard multiplexing [221] for Jacobian acquisition during optical SGD; and (3) refining the auto-tuned patterns and decoder by running additional Optical SGD iterations with a higher softmax temperature ($\mu = 1000$). This provides an even tighter continuous approximation to the total penalty, $\|\text{err}(\mathbf{d}, \mathbf{g})\|_1$, at the expense of a harder optimization landscape. Figure 7.9 shows the performance improvements caused by individual modifications to our basic optical SGD procedure. Upon convergence, this yielded patterns and a decoder that performed well above MPS16 on the test scene 80cm away.

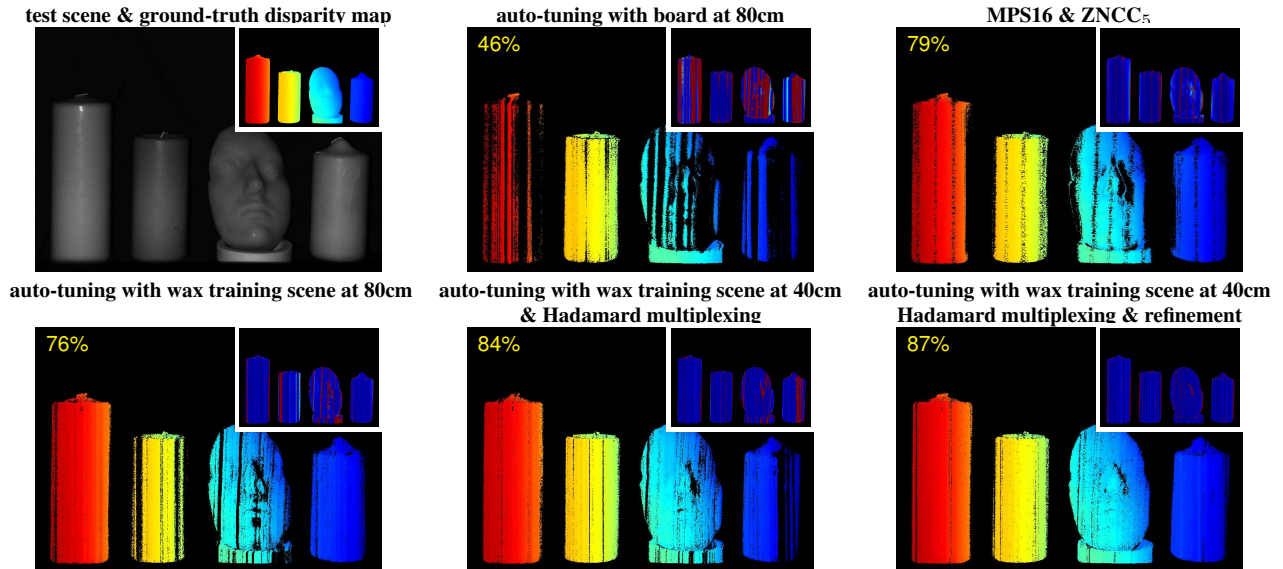


Figure 7.9: **Auto-tuning for Indirect Light.** To better visualize reconstruction accuracy, only pixels with error ≤ 2 are shown in the disparity maps above. The total percentage of such pixels is indicated in yellow in the upper left, with the correspondence error map shown as an inset (darkest blue for error=0, darkest red for error ≥ 20).

7.5.3 OpticalSGD for Different Imaging Systems

This section explores the applicability of optical auto-tuning framework to a broad range of structured light imaging systems.

Simulations with Mitsuba CLT [238, 103] and ModelNet [270] To assess how well an auto-tuned system can perform on other scenes, we treated the Mitsuba CLT renderer as a black-box projector-camera system and auto-tuned it using a virtual training board similar to those in Figure 7.1(middle). We then used the optimized patterns and optimized ZNCC- NN_3 decoder

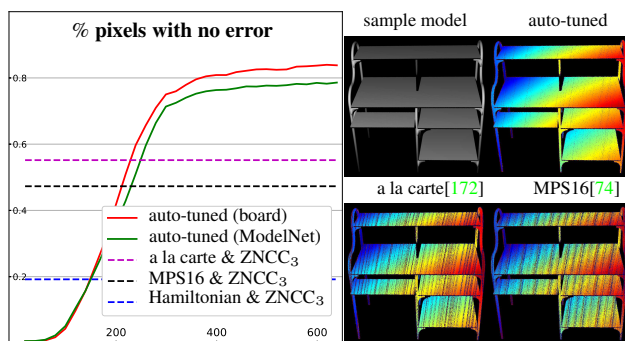


Figure 7.10: **Auto-tuning Mitsuba CLT for Four Patterns and the 0-tolerance Penalty.** **Left:** Performance of optimized patterns and ZNCC- NN_3 decoder across iterations of optical SGD. We measure performance by reconstructing the virtual training board (red plot) as well as ModelNet objects (green plot, averaged over 30 models). Optical SGD performs considerably better on ModelNet than state-of-the-art patterns combined with our ZNCC $_3$ decoder (dashed lines). **Right:** Disparity maps for a sample model.

to reconstruct a set of 30 randomly-selected models from the ModelNet dataset [270]. The results in Figure 7.10 show no evidence of over-fitting to the virtual training board, and mirror those of Figure 7.7.

Auto-Tuning Color Projector-Camera Systems We used the methods described in section 7.4.4 to auto-tune two different projector-camera systems for color structured light.

Color structured light with cellphone. The results are shown in Figure 7.1 (top row). In this experiment, we auto-tuned the system for $K = 5$ patterns, the ZNCC- NN_5 decoder, and 1-tolerance penalty function.

Color vs. monochrome structured light experiment. To do so, we used the same projector (LG) and base camera system (Prosilica AVT with Schneider lens), with the only difference being that the color version of the system had a Bayer filter fitted onto the sensor (models Prosilica 1920c and Prosilica 1920, respectively). Thus, the color camera provided color information at the expense of lower quantum efficiency for its pixels. For both systems, we auto-tune for the 0-tolerance penalty and the ZNCC- NN_5 decoder.

Figure 7.11 compares disparity maps obtained with color patterns and gray patterns, using the approach in Section 7.4.4 to handle color. Two observations can be made about the results. First, despite their lower quantum efficiency and lower number of patterns, 3 color patterns perform comparably to 4 gray patterns. Second, auto-tuning on demosaiced images works better than using the raw images. Intuitively, auto-tuning on demosaiced images benefits from the inference taking place in the demosaicing procedure.

7.5.4 Operating Range of an Auto-Tuned System

Experimental Setup We used the LG-IDS system in Figure 7.12 to reconstruct the training board in geometrical arrangements that are different from those used for auto-tuning. We

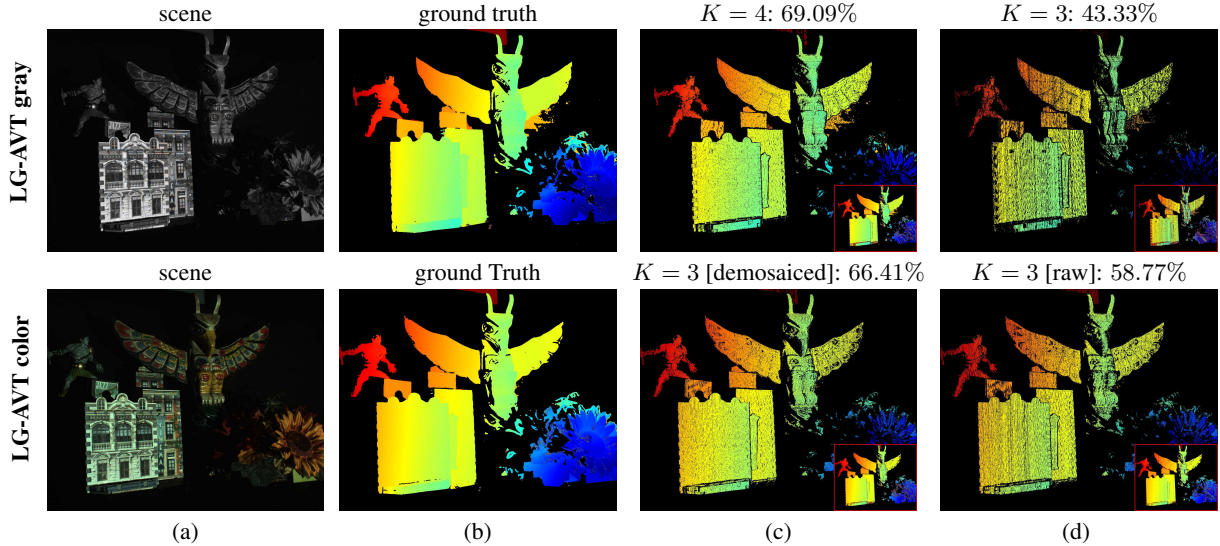


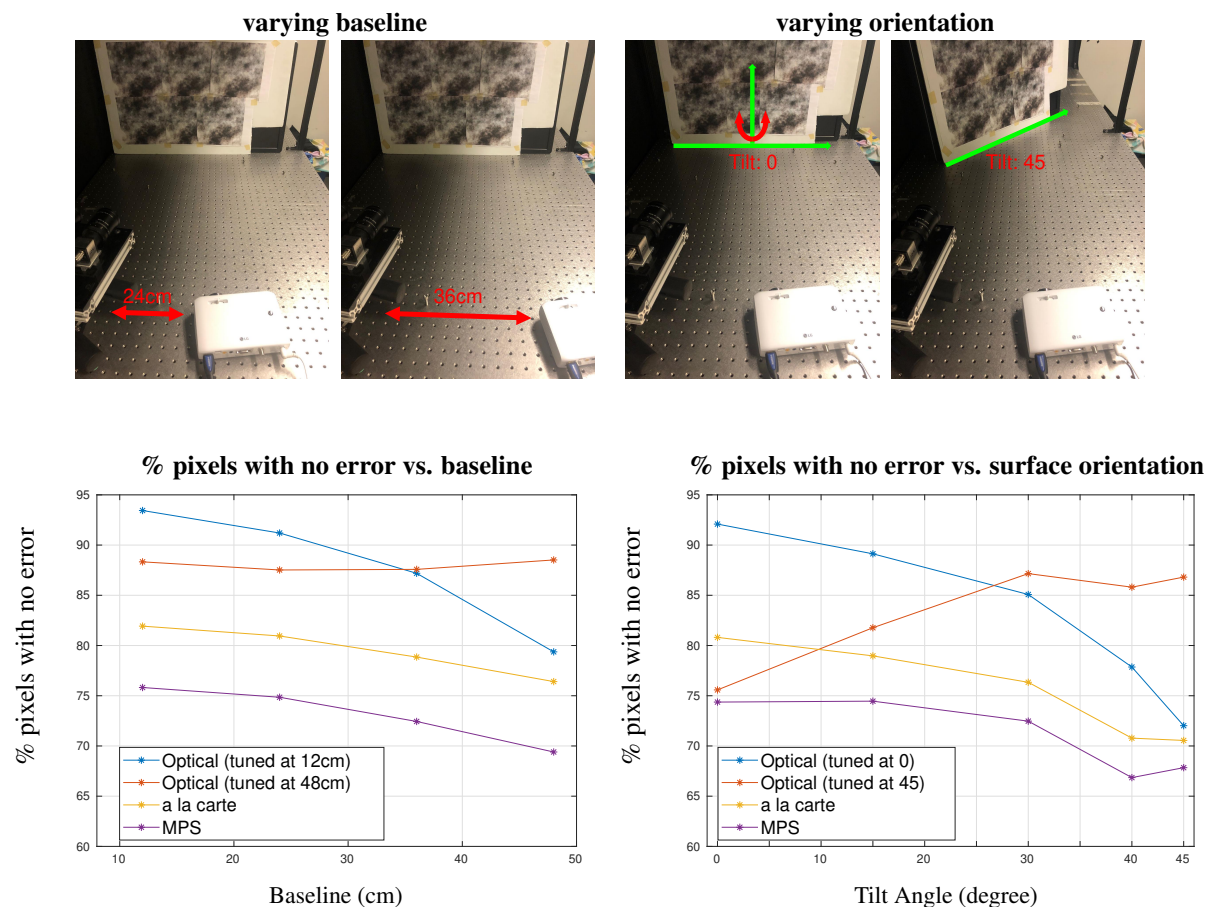
Figure 7.11: **Comparison of Auto-tuned Color Patterns with Gray Patterns.** **Top:** Auto-tuning gray patterns for LG-AVT. **(a)** scene image captured by the monochrome camera. **(b)** 0-tolerance ground-truth. **(c-d)** 0-tolerance disparity map (overlaid with raw disparity map) for $K = 4$ and $K = 3$ with their percentages of zero-error pixels. **Bottom:** Auto-tuning color patterns for LG-AVT. **(a)** scene image captured by the color camera. **(b)** 0-tolerance ground-truth. **(c)** 0-tolerance disparity map (overlaid with raw disparity map) for $K = 3$ color pattern, auto-tuned on demosaiced images with its percentage of zero-error pixels. **(d)** 0-tolerance disparity map (overlaid with raw disparity map) for $K = 3$ color pattern, auto-tuned on raw images with its percentage of zero-error pixels.

compare the performance of $K = 4$ sequences of MPS, Hamiltonian, a la carte patterns decoded with ZNCC_5 (which gives the best performance for the patterns), and patterns auto-tuned for 0-tolerance penalty with ZNCC-NN_5 decoder. .

Varying Stereo Baseline First, we analyze the performance of patterns on the same system but with different baselines. For this case, we fix the camera’s and the training board’s position and orientation, and move the projector to change the stereo baseline. The camera-to-board distance was approximately 80cm and we tested five baselines (12cm, 24cm, 36cm, and 48cm). We compare the performance of MPS, Hamiltonian, and a la carte with the performance of a system auto-tuned to one of two different baselines (12cm and 48cm). Figure 7.12 (left) shows the experimental setup, and patterns’ performance according to the evaluation metrics corresponding to 0-tolerance penalty.

Discussion. Three observations can be made about the results: First, by increasing the baseline the performance of all the patterns drops. Second, training on a particular baseline provides the best performance on that baseline (regardless of penalty function). Third, although optimizing on a particular baseline gives the best performance, auto-tuning a system on one baseline continues to perform well in other baselines as well, outperforming existing methods.

Varying Surface Orientation Second, we analyze the effect of surface orientation on reconstruction performance. Here, we fix the baseline (24cm), and the training board’s distance with respect to the system (0.8m); and rotate the board (0, 15, 30, and 45 degree). We



average error vs. baseline

average error vs. surface orientation

Figure 7.12: Comparison of Auto-tuned Patterns Optimized on Different Geometrical Arrangements with Existing Patterns.

auto-tune the system at one of two board orientations, and compare their performance with other existing patterns. Figure 7.12 (right) shows the experimental setup, and the patterns' performance at different orientations.

Discussion. The results show similar behavior to the case of varying baseline. First, almost all the methods' performance drops when the surface tilts away from the baseline; the only exception is the case of auto-tuning at 45 degree which performs better as tilt increases. Second, systems auto-tuned on a specific orientation perform best at that orientation. Third, a system tuned for zero tilt, generalizes well to other tilts, outperforming other existing methods at different angles. This does not always hold for a system tuned for 45 degree tilt. This suggests systems tuned with a board parallel to the baseline perform well for a larger range of orientations.

Varying Distance Finally, we evaluate the influence of object-to-system distance on the performance of patterns. For this experiment, we used a room corner as the test scene (Figure 7.13, top right). While keeping the baseline (20cm) and scene fixed, we translated

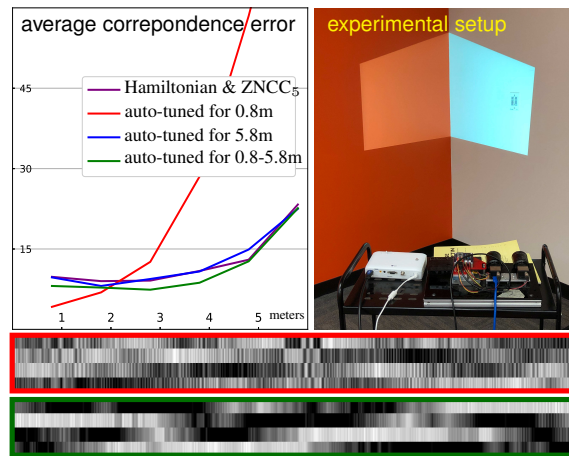


Figure 7.13: **Reconstructing a Room Corner from Different Standoff Distances After Auto-tuning for a Specific Distance (or a Range of Distances)**. We observe that the frequency content of patterns optimized for 0.8m (red) is much higher than those for 0.8-5.8m (green).

the camera-projector pair backward from the corner (0.8m, 1.8m, 2.8m, 3.8m, 4.8m, and 5.8m) and auto-tune the system for 3 distances using the L_1 penalty: one for a training board at 0.8m, one for a training board at 5.8m, and one for the range of 0.8 to 5.8 meters (by drawing random distance samples).

Discussion. We use the approach described in section 7.4.2 to auto-tune for distances beyond 0.8m because the projection patterns are too dim at distances beyond 2m to compute gradients reliably. We also choose L_1 penalty since it produces more reasonable results for far distance. Figure 7.13 shows an overview of the experimental setup, and the average disparity error as a function of depth. Similarly, systems auto-tuned on a specific distance perform best at that distance range.

7.6 Summary

Our optical-domain implementation of SGD offers an alternative way to solve optimal coding problems in imaging, that emphasizes real-time control—and learning by imaging—over modeling. Although we have shown that very competitive coding schemes for structured light can emerge on the fly with this approach, the question of how a system can be tuned *even further*—for specific materials, for specific families of 3D shapes, for complex light transport, *etc.*—remains wide open. Another interesting direction would be extend OpticalSGD to other imaging systems to discover their optimal imaging parameters. We leave it as future work.

Chapter 8

Conclusion

In this thesis, we propose to differentiate imaging systems and embed them into learning based reconstruction frameworks to boost 3D reconstruction. The thesis conducts novel differentiable reformulation on rendering pipelines and structured light systems, where each demonstrates new capabilities in 3D reconstruction, achieving large-margin performance improvement compared to prior works. In this chapter, we first introduce key insights we learned from the accomplished works and their limitations in Sec. 8.1. We then talk about promising future directions in Sec. 8.2.

8.1 Learned Lessons

Single-view 3D Object Reconstruction with Differentiable Rendering Single-view 3D object reconstruction is a challenging problem, where our goal is to infer 3D properties from a single image. It becomes even more difficult when we want to jointly recover shape, texture, light and material which are highly entangled together in the image. Traditional supervised methods rely on images accompanied with 3D ground truth while differentiable rendering techniques relax the data requirement. In Chapter 3, we propose DIB-R, an interpolation-based renderer that supports to compute gradients from 2D images to most 3D attributes, including vertex positions, vertex colors, multiple lighting models, texture mapping, *etc.* It shows faithful shape and texture prediction under low-frequency lighting assumption.

However, DIB-R is still limited in disentangling clean texture from high-frequency lighting effects. This is the main reason why DIB-R++ came out in Chapter 4. By explicitly modelling the high-frequency lighting models, DIB-R++ demonstrates new capabilities to jointly predict shape, texture, light and material from a single image in an unsupervised way. It presents clean texture and accurate lighting estimation, especially for the images captured

with secondary lighting effects like specular or reflection.

By comparing these two methods, the lesson we learned is that differentiable rendering based single-view 3D object reconstruction cannot recover 3D properties beyond its rendering equations. In other words, if the rendering effects cannot be expressed by the differentiable rendering pipeline, it is impossible to recover the corresponding 3D properties in the learning tasks. It also indicates the limitation of DIB-R++: as a one-bounce ray tracing based renderer, it fails to account for more advanced rendering effects like shadows or scatter media which require at least two bounces. To further disentangle them, we need a new renderer that takes more than one bounce into consideration, as well as being fast enough to be embedded in learning frameworks.

On the other side, single-view 3D object reconstruction problem is highly ill-posed, where another key to disentangling everything lies in the training data. We find successful inverse graphics models rely on multi-view images. Therefore, in Chapter 5 we propose to use StyleGAN to create high-quality, multi-view training images and demonstrate that the neural networks trained on StyleGAN dataset can be well applied to real images to predict reliable 3D properties, since they have similar distributions. The benefits of multi-view images are two folds: On one hand, the shape is partially observed in the image. Similar to shape curving, novel views help regularize the shape to be correct for both visible and invisible parts. Moreover, texture, material and light are generally merged together to generate the final object appearance. Luckily, lighting effects are view-dependent so multi-view data could help decouple it from texture. Interestingly, even with multi-view images, we find that the predicted texture, material and lighting parameters, though exhibiting very close rendering effects to the input image, are not exactly the same from the ground truth parameters. It indicates that even multi-view data is not sufficient to find a unique solution. We might need multi-view, multi-illumination data to further constrain this problem.

Structured Light Triangulation with Optimized Patterns In Chapter 6 and 7, we move to structured light systems. While all the previous works define system illuminations with manually designed patterns, in Chapter 6 we propose à la carte, a new paradigm where by differentiating structured light imaging formal models and embed them into learning frameworks, it is able to design patterns from a machine learning perspective and automatically optimize them by minimizing the expected disparity loss. We show the optimized patterns significantly outperform previous hand-crafted ones.

While à la carte considers a simplified imaging formation model in simulation and ignores realistic hardware properties, in Chapter 7 we further propose OpticalSGD to embed the real structured light systems in learning frameworks, optimizing patterns for the actual hardware

devices, and doing so without any requirement of system modelling. The strong performance boost from *à la carte* to OpticalSGD indicates the huge domain gap between the simulated imaging formation models and the actual devices. *À la carte* adopts a simplified linear model, which fails to account for more practical optical phenomena like camera lens blur, radiometric response functions, as well as the camera and projector noise. Hardware-in-the-loop can inherently take them into consideration during training, generating patterns that perfectly match the hardware device properties.

OpticalSGD differentiates hardware systems by computing gradients in optical domain. However, currently they are computed in a numeric way. The efficient gradient computation is implemented with spatial spacing, which relies on the sparse property of the light transport matrices. As a result, it can hardly be extended to other systems without such properties. Developing general and fast gradient computation techniques might be the key to generalize to more broader imaging systems.

8.2 Future Work

Conceptually, differentiating and embedding imaging systems in learning frameworks can be treated as a new way of applying imaging priors to learning systems. The biggest advantage is that it could jointly optimize imaging settings as well as reconstruction algorithms, where they boost each other, achieving huge improvement compared to previous methods. In practice, differentiable imaging systems lie in the intersection of two important research areas: machine learning and computational imaging, where promising future research happens in both sides. We propose to continue to push the boundaries of 3D reconstruction in both areas, with special focus on the below directions.

3D Representations Choosing proper 3D representations that perfectly match imaging systems is critical in 3D reconstruction, which influences the design of differentiable imaging systems and reconstruction algorithms, as well as their usage in downstream applications. Our differentiable rendering works [38, 39, 291] utilize explicit mesh representations. Composed of explicit geometry and SVBRDF, they are readily usable in graphics engines like Maya or Blender. However, meshes generally require pre-defined topology which are limited in representing complex objects, *e.g.*, shapes beyond 0-genus. On the other side, implicit representations in Nerf-series works [171] demonstrate ability to represent complex geometry and exhibit excellent effects in novel view synthesis. However, they typically lack interpretability and editability, and are not compatible with graphics engines.

As such, exploring how to develop hybrid representations that leverage the complementary

benefits of explicit and implicit representations is an important direction. As a first try, our recent work [178] combines explicit tetrahedral mesh and implicit SDF field together, achieving promising effects in complex object reconstruction. We demonstrate detailed shapes, exquisite texture maps, accurate lighting and material parameters are all inferable from 2D photographs only. Moreover, the predicted 3D attributes can be directly exported to any graphics engine for artists to edit. Next, we will continue to explore proper 3D representations for more challenging cases, *e.g.*, discovering suitable representations in large-scale scene or video reconstruction tasks.

Universal Differentiable Framework Differentiate imaging systems enable joint optimization for reconstruction algorithms and imaging settings, which demonstrate huge improvements over prior works. However, different imaging systems have distinct imaging formation models. Currently, each imaging system requires a specific differentiable design. For example, in my previous works, rendering pipelines, structured light systems have totally different designs.

Therefore, it is also important to develop universal differentiable frameworks that can be applied to all the imaging systems, back propagating gradients to their imaging parameters efficiently, no matter which imaging formation models they have. Existing gradient descent based methods can hardly handle the non-differentiable, physical procedures in imaging process. We plan to investigate advanced optimization algorithms like Bayesian optimization or reinforcement learning to solve the problem. We anticipate the frameworks can be potentially applied to a wide range of imaging systems, from conventional cameras to cutting-edge medical imaging or astronomy imaging.

Appendix A

Supplementary Material for DIB-R

A.1 Derivation of DIB-R Renderer

In this section we show how to back propagate gradients from barycentric weights to the vertex positions via differentiable functions Ω . As shown in Fig. A.1, the pixel at position \vec{p}_i is covered by the face f_j with three vertices $\vec{v}_0, \vec{v}_1, \vec{v}_2$, where \vec{p}_i and \vec{v}_k are 2D coordinates on the image plane and ω_k is corresponding barycentric weight. Here, $k = 0, 1, 2$. We show how to compute $\omega_0 = \Omega_0(\vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{p}_i)$ below, and weights ω_1 and ω_2 can be calculated similarly.

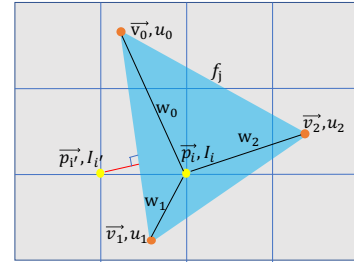


Figure A.1: Illustration of Our Differentiable Rasterization.

With the sum of barycentric weights being equal to 1, we have:

$$\omega_0 + \omega_1 + \omega_2 = 1 . \quad (\text{A.1})$$

We can rewrite it in matrix form as:

$$\omega_0 = 1 - \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} . \quad (\text{A.2})$$

The Barycentric weights are calculated via pixel position \vec{p}_i and face vertex positions \vec{v}_0, \vec{v}_1 and \vec{v}_2 :

$$\vec{p}_i = \omega_0 \vec{v}_0 + \omega_1 \vec{v}_1 + \omega_2 \vec{v}_2 , \quad (\text{A.3})$$

where we can also rewrite it as:

$$\begin{bmatrix} \vec{v}_1 - \vec{v}_0 & \vec{v}_2 - \vec{v}_0 \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} = \begin{bmatrix} \vec{p}_i - \vec{v}_0 \end{bmatrix} . \quad (\text{A.4})$$

Thus we have:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \end{bmatrix} = \begin{bmatrix} \vec{v}_1 - \vec{v}_0 & \vec{v}_2 - \vec{v}_0 \end{bmatrix}^{-1} \begin{bmatrix} \vec{p}_i - \vec{v}_0 \end{bmatrix} . \quad (\text{A.5})$$

If we merge Equation (A.2) and Equation (A.5), we can easily derive that:

$$\omega_0 = 1 - \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} \vec{v}_1 - \vec{v}_0 & \vec{v}_2 - \vec{v}_0 \end{bmatrix}^{-1} \begin{bmatrix} \vec{p}_i - \vec{v}_0 \end{bmatrix} . \quad (\text{A.6})$$

In this way, ω_0 can be treated as a output of a function while the input variables are pixel coordinate \vec{p}_i and vertex positions \vec{v}_0, \vec{v}_1 and \vec{v}_2 . We rewrite the weight ω_0 as the Ω_0 function and thus gradients can be back propagated from ω_0 to vertex positions:

$$\omega_0 = \Omega_0(\vec{v}_0, \vec{v}_1, \vec{v}_2, \vec{p}_i) \quad (\text{A.7})$$

$$\frac{\partial \omega_0}{\partial \vec{v}_k} = \frac{\partial \Omega_0}{\partial \vec{v}_k} \quad (\text{A.8})$$

Appendix B

Supplementary Material for DIB-R++

B.1 BRDF Model for DIB-R++

For the BRDF, we use a simplified version of the isotropic Disney BRDF [29]:

$$f_{\text{r}}(\mathbf{x}, \boldsymbol{\omega}_{\text{i}}, \boldsymbol{\omega}_{\text{o}}) = \frac{\mathbf{a}}{\pi} + \frac{D(\boldsymbol{\omega}_{\text{h}})F(\boldsymbol{\omega}_{\text{o}}, \boldsymbol{\omega}_{\text{h}})G(\boldsymbol{\omega}_{\text{i}}, \boldsymbol{\omega}_{\text{o}})}{4|\mathbf{n} \cdot \boldsymbol{\omega}_{\text{i}}||\mathbf{n} \cdot \boldsymbol{\omega}_{\text{o}}|}, \quad (\text{B.1})$$

where \mathbf{a} is the diffuse albedo, D is the (micro-)normal distribution function (NDF), F is the Fresnel term, G is a geometry or shadowing factor, and $\boldsymbol{\omega}_{\text{h}} = (\boldsymbol{\omega}_{\text{i}} + \boldsymbol{\omega}_{\text{o}})/\|\boldsymbol{\omega}_{\text{i}} + \boldsymbol{\omega}_{\text{o}}\|$ is the half-vector.

MC Shading For the NDF, we use the GGX/Trowbridge–Reitz distribution parameterized by a roughness parameter $\beta > 0$, which we map to $\alpha = (\beta + 1)^2/8$ to allow for more perceptually linear change [57]:

$$D^{(\text{MC})}(\boldsymbol{\omega}_{\text{h}}; \alpha) = \frac{\alpha^2}{\pi((\mathbf{n} \cdot \boldsymbol{\omega}_{\text{h}})^2(\alpha^2 - 1) + 1)^2}. \quad (\text{B.2})$$

We importance sample this term directly using inverse transform sampling [199, 29]. For the geometry term, we use the Schlick model [222], which is given by

$$\begin{aligned} G(\boldsymbol{\omega}_{\text{i}}, \boldsymbol{\omega}_{\text{o}}) &= G_1(\boldsymbol{\omega}_{\text{i}})G_1(\boldsymbol{\omega}_{\text{o}}) \\ G_1(\boldsymbol{\omega}; \mathbf{n}, \alpha) &= \frac{\mathbf{n} \cdot \boldsymbol{\omega}}{(\mathbf{n} \cdot \boldsymbol{\omega})(1 - \alpha) + \alpha}. \end{aligned} \quad (\text{B.3})$$

Finally, for the Fresnel reflection coefficient, we use the SG approximation¹ [57] given by:

$$F(\boldsymbol{\omega}_{\text{o}}, \boldsymbol{\omega}_{\text{h}}) = s + (1 - s)2^{(-5.55473(\boldsymbol{\omega}_{\text{o}} \cdot \boldsymbol{\omega}_{\text{h}}) - 6.98316)(\boldsymbol{\omega}_{\text{o}} \cdot \boldsymbol{\omega}_{\text{h}})}, \quad (\text{B.4})$$

¹Note that this is not related to our SG model. It is common to replace the $1 - (\boldsymbol{\omega}_{\text{o}} \cdot \boldsymbol{\omega}_{\text{h}})^5$ term in Schlick’s model by an SG to remove the power and improve efficiency.

where s is the specular albedo at normal incidence, which we define as a linear combination between relative IORs and albedo using a *metallic* parameter $m \in [0, 1]$:

$$F_0 = \text{Lerp}(|1 - \eta|^2 / |1 + \eta|^2, \mathbf{a}, m) . \quad (\text{B.5})$$

Here, $\eta = 1.5$ (vector) and m is the blending weight. This interpolation allows us to treat conductors and dielectric with the same approximation. Intuitively, if a material is dielectric we use the IOR, otherwise we use the albedo to “tint” the reflection.

SG Shading Similar to Wang et al. [259], we approximate the NDF D using a single SG lobe

$$D^{(\text{SG})}(\boldsymbol{\omega}_h; \alpha) \approx \mathcal{G}_d \left(\boldsymbol{\omega}_h; \mathbf{n}, \frac{2}{\alpha^2}, \frac{1}{\pi\alpha^2} \right) , \quad (\text{B.6})$$

and apply a spherical warp [259] to orient the distribution lobe about the reflected view direction:

$$D^{(\text{SG Warped})}(\boldsymbol{\omega}_h; \alpha) \approx \mathcal{G}_d \left(\boldsymbol{\omega}_h; 2(\boldsymbol{\omega}_o \cdot \mathbf{n})\mathbf{n} - \boldsymbol{\omega}_o, \frac{\lambda_d}{4|\mathbf{n} \cdot \boldsymbol{\omega}_o|}, \boldsymbol{\mu}_d \right) , \quad (\text{B.7})$$

where λ_d and $\boldsymbol{\mu}_d$ are those of Eq. (B.6). Since the Fresnel and geometry terms cannot be approximated with SGs, we assume that their values are constant across the entire BRDF lobe and pull them out of the integral. The cosine foreshortening term is approximated with a single SG as $|\mathbf{n} \cdot \boldsymbol{\omega}_i| \approx \mathcal{G}_c(\boldsymbol{\omega}_i; \mathbf{n}, 2.133, 1.17)$ [166]. The outgoing radiance can finally be evaluated in closed form by integrating against the environment map, also given as a mixture of SGs.

Appendix C

Supplementary Material for OpticalSGD

C.1 Proof of Eq.(7.16)

The total penalty of a correspondence map \mathbf{d} with ground truth map \mathbf{g} is defined as:

$$\|\text{err}(\mathbf{d}, \mathbf{g})\|_1 = \sum_{m=1}^M \rho(\mathbf{d}[m] - \mathbf{g}[m]) \quad (\text{C.1})$$

Using ZNCC-based decoders (Eqs. (7.12)-(7.15)) for acquiring correspondence map \mathbf{d} , we have:

$$\text{err}(\mathbf{d}, \mathbf{g})[m] = \rho(\arg \max_{1 \leq n \leq N} \mathbf{z}_m[n] - \mathbf{g}[m]) \quad (\text{C.2})$$

Now, we define a max-indicator vector \mathbf{i}_m as follows:

$$\mathbf{i}_m[j] = \begin{cases} 1 & \text{if } j = \arg \max_{1 \leq n \leq N} \mathbf{z}_m[n] \\ 0 & \text{otherwise} \end{cases}$$

Using \mathbf{i}_m , we can rewrite Eq. (C.2):

$$\text{err}(\mathbf{d}, \mathbf{g})[m] = \rho((\mathbf{i}_m \cdot \mathbf{index}) - \mathbf{g}[m](\mathbf{i}_m \cdot \mathbf{1})) \quad (\text{C.3})$$

where \mathbf{index} is a vector whose i -th element is equal to its index i ; $\mathbf{1}$ represents an all-one vector; and \cdot denotes the dot product.

We can now further simplify Eq. (C.3) by factoring \mathbf{i}_m :

$$\text{err}(\mathbf{d}, \mathbf{g})[m] = \mathbf{i}_m \cdot [\rho(\mathbf{index}[0] - \mathbf{g}[m]), \dots, \rho(\mathbf{index}[n] - \mathbf{g}[m])] . \quad (\text{C.4})$$

On the other hand, it is straightforward to show:

$$\lim_{\mu \rightarrow \infty} \text{softmax}(\mu \mathbf{z}_m) = \mathbf{i}_m \quad (\text{C.5})$$

By combining Eq. (C.4) and Eq. (C.5), we get:

$$\text{err}(\mathbf{d}, \mathbf{g})[m] = \lim_{\mu \rightarrow \infty} \text{softmax}(\mu \mathbf{z}_m) \cdot \underbrace{[\rho(\mathbf{index}[0] - \mathbf{g}[m]), \dots, \rho(\mathbf{index}[n] - \mathbf{g}[m])]}_{\text{err}(\mathbf{index} - \mathbf{g}[m], \mathbf{0})} , \quad (\text{C.6})$$

Therefore, for sufficiently large μ , we can conclude:

$$\|\text{err}(\mathbf{d}, \mathbf{g})\|_1 \approx \sum_{m=1}^M \text{softmax}(\mu \mathbf{z}_m) \cdot \text{err}(\mathbf{index} - \mathbf{g}[m], \mathbf{0}) \quad (\text{C.7})$$

QED. \square

projector	projector response function	projector columns	pattern spatial frequency limit	camera	image exposure time	camera response function
LG-PH550	non-linear	1280	256	IDS-UI324x	33msec	linear
LightCrafter	linear	400	64	C2B [265]	17msec	linear
Optoma 4K	non-linear	3840	512	Huawei P9	33msec	non-linear
LG-PH550	non-linear	1280	256	AVT-1920c	17msec	linear
LG-PH550	non-linear	1280	256	AVT-1920	17msec	linear
PicoPro	non-linear	1280	256	IDS-UI324x	17msec	linear

Table C.1: List of the experimental imaging systems.

encoding scheme	decoder	optimized for penalty	Reference	radiometrically calibrated projector?	auto-tuning applied?
Hamiltonian	ZNCC	L_1	[78]	Yes	No
MPS	ZNCC	none	[74]	Yes	No
A la carte	ZNCC	0-tolerance	[172]	Yes	No
Hamiltonian	ZNCC ₅	L_1	this thesis	Yes	No
MPS	ZNCC ₅	none	this thesis	Yes	No
A la carte	ZNCC ₅	0-tolerance	this thesis	Yes	No
Hamiltonian	ZNCC-NN ₅	0-tolerance & L_1	this thesis	Yes	decoder only
MPS	ZNCC-NN ₅	0-tolerance & L_1	this thesis	Yes	decoder only
A la carte	ZNCC-NN ₅	0-tolerance & L_1	this thesis	Yes	decoder only
auto-tuned	ZNCC	0-tolerance & L_1	this thesis	No	patterns only
auto-tuned	ZNCC ₅	0-tolerance & L_1	this thesis	No	patterns only
auto-tuned	ZNCC-NN ₅	0-tolerance & L_1	this thesis	No	patterns & decoder

Table C.2: List of the coding-decoding methods used for experiments.

C.2 List of Devices & Encoding-Decoding Methods

To better reproduce our method, we list all the devices and encoding-decoding algorithms we use in Chapter 7. Table C.1 lists all the devices we used while Table C.2 shows all the encoding-decoding methods used in the experiments.

Bibliography

- [1] CVPR 2021 Tutorial on Physics-Based Differentiable Rendering. <https://www.diff-render.org/>.
- [2] Direct3D. <https://docs.microsoft.com/en-us/windows/win32/direct3d>.
- [3] Gray code. https://en.wikipedia.org/wiki/Gray_code.
- [4] History of the camera. https://en.wikipedia.org/wiki/History_of_the_camera.
- [5] OpenGL. <https://www.opengl.org/>.
- [6] Photo statistics. <https://photutorial.com/photos-statistics/>.
- [7] Martin Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Man e, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Vi e gas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. Technical report, 2015.
- [8] Supreeth Achar, Joseph R Bartels, William L 'Red' Whittaker, Kiriakos N Kutulakos, and Srinivasa G Narasimhan. Epipolar time-of-flight imaging. *ACM TOG (SIGGRAPH)*, 36(4), 2017.
- [9] Supreeth Achar and Srinivasa G Narasimhan. Multi Focus Structured Light for Recovering Scene Shape and Global Illumination. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 205–219, 2014.

- [10] Amit Adam, Christoph Dann, Omer Yair, Shai Mazor, and Sebastian Nowozin. Bayesian Time-of-Flight for Realtime Shape, Illumination and Albedo. *IEEE T-PAMI*, 39(5):851–864, 2017.
- [11] Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. *Real-time rendering*. AK Peters/crc Press, 2019.
- [12] Hassan Alhaija, Siva Mustikovela, Varun Jampani, Justus Thies, Matthias Niessner, Andreas Geiger, and Carsten Rother. Intrinsic autoencoders for joint neural rendering and intrinsic image decomposition. In *International Conference on 3D Vision (3DV)*, 2020.
- [13] P Ambs. A short history of optical computing: rise, decline, and evolution. In *Proc. SPIE*, 2009.
- [14] Nick Antipa, Grace Kuo, Reinhard Heckel, Ben Mildenhall, Emrah Bostan, Ren Ng, and Laura Waller. Diffusercam: lensless single-exposure 3d imaging. *Optica*, 5(1):1–9, Jan 2018.
- [15] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.
- [16] Sai Bangaru, Tzu-Mao Li, and Frédo Durand. Unbiased warped-area sampling for differentiable rendering. *ACM Trans. Graph.*, 39(6):245:1–245:18, 2020.
- [17] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24, 2009.
- [18] Harry Barrow, J. Tenenbaum, A. Hanson, and E. Riseman. Recovering intrinsic scene characteristics. *Comput. Vis. Syst*, 2:3–26, 1978.
- [19] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.
- [20] Ayush Bhandari, Achuta Kadambi, Refael Whyte, Christopher Barsi, Micha Feigin, Adrian Dorrington, and Ramesh Raskar. Resolving multipath interference in time-of-flight imaging via modulation frequency diversity and sparse regularization. *Optics Letters*, 39(6):1705–1708, 2014.
- [21] Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. Deep hybrid real and synthetic training for intrinsic decomposition. In *Eurographics Symposium on Rendering: Experimental Ideas & Implementations (EGSR)*, 2018.

- [22] Sai Bi, Zexiang Xu, Pratul Srinivasan, Ben Mildenhall, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Neural reflectance fields for appearance acquisition. *arXiv preprint arXiv:2008.03824*, 2020.
- [23] Sai Bi, Zexiang Xu, Kalyan Sunkavalli, Miloš Hašan, Yannick Hold-Geoffroy, David Kriegman, and Ravi Ramamoorthi. Deep reflectance volumes: Relightable reconstructions from multi-view photometric images. *arXiv preprint arXiv:2007.09892*, 2020.
- [24] Oliver Bimber and Ramesh Raskar. *Spatial augmented reality: merging real and virtual worlds*. CRC press, 2005.
- [25] Vivek Boominathan, Jacob T Robinson, Laura Waller, and Ashok Veeraraghavan. Recent advances in lensless imaging. *Optica*, 9(1):1–16, 2022.
- [26] Mark Boss, Raphael Braun, Varun Jampani, Jonathan T. Barron, Ce Liu, and Hendrik P.A. Lensch. Nerd: Neural reflectance decomposition from image collections. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.
- [27] Mark Boss, Varun Jampani, Kihwan Kim, Hendrik Lensch, and Jan Kautz. Two-shot spatially-varying BRDF and shape estimation. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3982–3991, 2020.
- [28] J Y Bouguet and P Perona. 3D photography on your desk. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 43–50. Narosa Publishing House, 1998.
- [29] Brent Burley. Physically-based shading at Disney. In *Practical Physically-Based Shading in Film and Game Production (SIGGRAPH Course)*, pages 10:1–10:7, 2012.
- [30] Clara Callenberg, Felix Heide, Gordon Wetzstein, and Matthias B Hullin. Snapshot difference imaging using correlation time-of-flight sensors. *ACM TOG*, 36(6), November 2017.
- [31] Ayan Chakrabarti. Learning Sensor Multiplexing Design through Back-propagation. In *Advances In Neural Information Processing Systems*, pages 3081–3089, 2016.
- [32] Praneeth Chakravarthula, Ethan Tseng, Tarun Srivastava, Henry Fuchs, and Felix Heide. Learned hardware-in-the-loop phase retrieval for holographic near-eye displays. *ACM Transactions on Graphics (TOG)*, 39(6):186, 2020.

- [33] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.
- [34] Julie Chang, Vincent Sitzmann, Xiong Dun, Wolfgang Heidrich, and Gordon Wetstein. Hybrid optical-electronic convolutional neural networks with optimized diffractive optics for image classification. *Sci. Rep.*, 8(1), 2018.
- [35] Binghui Chen, Weihong Deng, and Junping Du. Noisy Softmax: Improving the Generalization Ability of DCNN via Postponing the Early Softmax Saturation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5372–5381, 2017.
- [36] Chen Chen, Qifeng Chen, Jia Xu, and Vladlen Koltun. Learning to see in the dark. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [37] T Chen, Hans-Peter Seidel, and Hendrik P A Lensch. Modulated phase-shifting for 3D scanning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.
- [38] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaako Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *Advances In Neural Information Processing Systems*, 2019.
- [39] Wenzheng Chen, Joey Litalien, Jun Gao, Zian Wang, Clement Fuji Tsang, Sameh Khalis, Or Litany, and Sanja Fidler. DIB-R++: Learning to predict lighting and material with a hybrid differentiable renderer. In *Advances in Neural Information Processing Systems*, 2021.
- [40] Wenzheng Chen, Parsa Mirdehghan, Sanja Fidler, and Kiriakos N. Kutulakos. Auto-tuning structured light by optical stochastic gradient descent. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [41] Xi Chen, Yan Duan, Rein Houthoofd, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.
- [42] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

- [43] F.J.J. Clarke and D.J. Parry. Helmholtz reciprocity: its validity and application to reflectometry. *Lighting Research & Technology*, 17(1):1–11, 1985.
- [44] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Trans. Graph.*, 1(1):7–24, January 1982.
- [45] Vincent Couture, Nicolas Martin, and Sébastien Roy. Unstructured Light Scanning Robust to Indirect Illumination and Depth Discontinuities. *Int. J. Computer Vision*, 108(3):204–221, 2014.
- [46] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 886–893 vol. 1, 2005.
- [47] Gerwin Damberg, James Gregson, and Wolfgang Heidrich. High Brightness HDR Projection Using Dynamic Freeform Lensing. *ACM TOG*, 35(3):1–11, 2016.
- [48] DeepMind. AlphaStar: Mastering the real-time strategy game StarCraft II, 2019. <https://deepmind.com/blog/article/alphastar-mastering-real-time-strategy-game-starcraft-ii>.
- [49] Michael Deering, Stephanie Winner, Bic Schediwy, Chris Duffy, and Neil Hunt. The triangle processor and normal vector shader: A vlsi system for high performance graphics. In *Proc. of the 15th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '88*, page 21–30, 1988.
- [50] Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. Single-image svbrdf capture with a rendering-aware deep network. *ACM Transactions on Graphics (SIGGRAPH Conference Proceedings)*, 37(128):15, aug 2018.
- [51] Yue Dong, Guojun Chen, Pieter Peers, Jiawan Zhang, and Xin Tong. Appearance-from-motion: Recovering spatially varying surface reflectance under unknown lighting. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 33(6), November 2014.
- [52] Matea Donlic, Tomislav Petkovic, and Tomislav Pribanic. On Tablet 3D Structured Light Reconstruction and Registration. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 2462–2471, 2017.
- [53] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

- [54] Sean Ryan Fanello, Christoph Rhemann, Vladimir Tankovich, Adarsh Kowdle, Sergio Orts Escolano, David Kim, and Shahram Izadi. HyperDepth: Learning Depth from Structured Light without Matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5441–5450, 2016.
- [55] Sean Ryan Fanello, Julien Valentin, Christoph Rhemann, Adarsh Kowdle, Vladimir Tankovich, Philip Davidson, and Shahram Izadi. UltraStereo: Efficient Learning-Based Matching for Active Stereo Systems. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6535–6544, 2017.
- [56] Sergi Foix, Guillem Alenya, and Carme Torras. Lock-in time-of-flight (tof) cameras: A survey. *IEEE Sensors Journal*, 11(9):1917–1926, 2011.
- [57] Brian Karis (Epic Games). Real shading in Unreal Engine 4. In *ACM SIGGRAPH 2013 Courses*, 2013.
- [58] Jun Gao, Wenzheng Chen, Tommy Xiang, Clement Fuji Tsang, Alec Jacobson, Morgan McGuire, and Sanja Fidler. Learning deformable tetrahedral meshes for 3d reconstruction. In *Advances In Neural Information Processing Systems*, 2020.
- [59] Marc-André Gardner, Yannick Hold-Geoffroy, Kalyan Sunkavalli, Christian Gagné, and Jean-François Lalonde. Deep parametric indoor lighting estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7175–7183, 2019.
- [60] 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. *arXiv preprint arXiv:1704.00090*, 2017.
- [61] Jason Geng. Structured-light 3d surface imaging: a tutorial. *Adv. Opt. Photon.*, 3(2):128–160, Jun 2011.
- [62] Kyle Genova, Forrester Cole, Aaron Maschinot, Aaron Sarna, Daniel Vlasic, and William T Freeman. Unsupervised training for 3d morphable model regression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8377–8386, 2018.
- [63] Purvi Goel, Loudon Cohen, James Guesman, Vikas Thamizharasan, James Tompkin, and Daniel Ritchie. Shape from tracing: Towards reconstructing 3d object geometry and svbrdf material from images via differentiable path tracing. In *2020 International Conference on 3D Vision (3DV)*, pages 1186–1195. IEEE, 2020.

- [64] Shubham Goel, Angjoo Kanazawa, and Jitendra Malik. Shape and viewpoints without keypoints. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [65] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [66] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.
- [67] J W Goodman. *Introduction to Fourier Optics*. Roberts & Company Publishers, 3rd edition, 2005.
- [68] Robin Green. Spherical harmonic lighting: The gritty details. In *Archives of the game developers conference*, volume 56, page 4, 2003.
- [69] Ned Greene, Michael Kass, and Gavin Miller. Hierarchical z-buffer visibility. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 231–238. ACM, 1993.
- [70] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan C. Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [71] Jinwei Gu, Toshihiro Kobayashi, Mohit Gupta, and Shree K Nayar. Multiplexed illumination for scene recovery in the presence of global illumination. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 691–698, 2011.
- [72] D. Guarnera, G.C. Guarnera, A. Ghosh, C. Denk, and M. Glencross. Brdf representation and acquisition. *Computer Graphics Forum*, 35(2):625–650, 2016.
- [73] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.
- [74] M. Gupta and S.K. Nayar. Micro Phase Shifting. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Jun 2012.
- [75] M Gupta, A Velten, S.K Nayar, and Eric Breitbach. What are optimal coding functions for time-of-flight imaging? *ACM TOG*, 37(2), 2018.

- [76] M Gupta, Qi Yin, and S.K Nayar. Structured Light in Sunlight. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 545–552, 2013.
- [77] Mohit Gupta, Amit Agrawal, Ashok Veeraraghavan, and Srinivasa G Narasimhan. A Practical Approach to 3D Scanning in the Presence of Interreflections, Subsurface Scattering and Defocus. *Int. J. Computer Vision*, 102(1-3):33–55, 2012.
- [78] Mohit Gupta and Nikhil Nakhate. A geometric perspective on structured light coding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.
- [79] Mohit Gupta, Yuandong Tian, Srinivasa G Narasimhan, and Li Zhang. A Combined Theory of Defocused Illumination and Global Light Transport. *Int. J. Computer Vision*, 98(2), 2011.
- [80] Felipe Gutierrez-Barragan, Syed Azer Reza, Andreas Velten, and Mohit Gupta. Practical Coding Function Design for Time-Of-Flight Imaging. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1566–1574, 2019.
- [81] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016.
- [82] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. *arXiv preprint arXiv:2004.02546*, 2020.
- [83] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, USA, 2 edition, 2003.
- [84] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.
- [85] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [86] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity Mappings in Deep Residual Networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 630–645, 2016.
- [87] Felix Heide, Wolfgang Heidrich, Matthias Hullin, and Gordon Wetzstein. Doppler Time-of-Flight Imaging. *ACM TOG*, 34(4):–36:11, August 2015.

- [88] Felix Heide, Matthias B Hullin, James Gregson, and Wolfgang Heidrich. Low-budget Transient Imaging Using Photonic Mixer Devices. In *ACM TOG (SIGGRAPH)*. ACM Request Permissions, 2013.
- [89] Paul Henderson and Vittorio Ferrari. Learning to generate and reconstruct 3d meshes with only 2d supervision. *arXiv preprint arXiv:1807.09259*, 2018.
- [90] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Automatic photo pop-up. *ACM Trans. Graph.*, 24(3):577–584, 2005.
- [91] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Putting objects in perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2137–2144. IEEE Computer Society, 2006.
- [92] Yannick Hold-Geoffroy, Akshaya Athawale, and Jean-François Lalonde. Deep sky modeling for single image outdoor lighting estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6927–6935, 2019.
- [93] Michael Holroyd, Jason Lawrence, and Todd Zickler. A Coaxial Optical Scanner for Synchronous Acquisition of 3D Geometry and Surface Reflectance. In *Proc. ACM SIGGRAPH Asia*, 2010.
- [94] E Horn and N Kiryati. Toward optimal structured light patterns. In *Proc. IEEE 3DIM*, pages 28–35, 1997.
- [95] Roarke Horstmeyer, Richard Y Chen, Barbara Kappes, and Benjamin Judkewitz. Convolutional neural networks that teach microscopes how to image. *arXiv*, 2017.
- [96] Yuanming Hu, Hao He, Chenxi Xu, Baoyuan Wang, and Stephen Lin. Exposure: A White-Box Photo Post-Processing Framework. *ACM TOG*, 37(2):1–17, 2018.
- [97] J M Huntley and H Saldner. Temporal phase-unwrapping algorithm for automated interferogram analysis. *Appl Optics*, 32(17):3047–3052, 1993.
- [98] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *Advances in Neural Information Processing Systems*, pages 2802–2812, 2018.
- [99] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

- [100] Rui Ishiyama, Shizuo Sakamoto, Johji Tajima, Takayuki Okatani, and Koichiro Deguchi. Absolute phase measurements using geometric constraints between multiple cameras and projectors. *Appl Optics*, 46(17):3528–3538, 2007.
- [101] 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.
- [102] Krishna Murthy J., Edward Smith, Jean-Francois Lafleche, Clement Fuji Tsang, Artem Rozantsev, Wenzheng Chen, Tommy Xiang, Rev Lebededian, and Sanja Fidler. Kaolin: A pytorch library for accelerating 3d deep learning research. *arXiv:1911.05063*, 2019.
- [103] Wenzel Jakob. Mitsuba renderer, 2010. <http://www.mitsuba-renderer.org>.
- [104] Wenzel Jakob. Enoki: structured vectorization and differentiation on modern processor architectures, 2019. <https://github.com/mitsuba-renderer/enoki>.
- [105] Wenzel Jakob, Sébastien Speierer, Nicolas Roussel, and Delio Vicini. Dr.jit: A just-in-time compiler for differentiable rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4), July 2022.
- [106] Adrian Jarabo, Julio Marco, Adolfo Muñoz, Raul Buisan, Wojciech Jarosz, and Diego Gutierrez. A framework for transient rendering. *ACM Transactions on Graphics (SIGGRAPH Asia 2014)*, 33(6), 2014.
- [107] Adrian Jarabo, Belen Masia, Julio Marco, and Diego Gutierrez. Recent advances in transient imaging: A computer graphics and vision perspective. *Visual Informatics*, 1(1):65–79, 2017.
- [108] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A practical model for subsurface light transport. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '01*, page 511–518, New York, NY, USA, 2001. Association for Computing Machinery.
- [109] Zhaoyin Jia, Andrew Gallagher, Yao-Jen Chang, and Tsuhan Chen. A learning-based framework for depth ordering. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 294–301. IEEE, 2012.
- [110] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

- [111] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, Aug 2021.
- [112] Achuta Kadambi and Ramesh Raskar. Rethinking Machine Vision Time of Flight With GHz Heterodyning. *IEEE Access*, 5:26211–26223, 2017.
- [113] Achuta Kadambi, Refael Whyte, Ayush Bhandari, Lee Streeter, Christopher Barsi, Adrian Dorrington, and Ramesh Raskar. Coded time of flight cameras: sparse deconvolution to address multipath interference and recover time profiles. In *Proc. ACM SIGGRAPH Asia*, 2013.
- [114] J. T. Kajiya and T. L. Kay. Rendering fur with three dimensional textures. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '89, page 271–280, New York, NY, USA, 1989. Association for Computing Machinery.
- [115] James T Kajiya. The rendering equation. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 143–150, 1986.
- [116] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 371–386, 2018.
- [117] Hakki Can Karaimer and Michael S. Brown. A software platform for manipulating the camera imaging pipeline. In *European Conference on Computer Vision (ECCV)*, 2016.
- [118] Brian Karis. Real Shading in Unreal Engine 4, 2013.
- [119] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.
- [120] 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.
- [121] Hiroharu Kato and Tatsuya Harada. Self-supervised learning of 3d objects from natural images. *arXiv preprint arXiv:1911.08850*, 2019.
- [122] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3d mesh renderer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3907–3916, 2018.
- [123] Michael Kellman, Emrah Bostan, Nicole Repina, and Laura Waller. Physics-based Learned Design: Optimized Coded-Illumination for Quantitative Phase Imaging. *IEEE TCI*, 2019.
- [124] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations (ICLR)*, 2015.
- [125] T Koninckx and L Van Gool. Real-time range acquisition by adaptive structured light. *IEEE T-PAMI*, 28(3):432–445, 2006.
- [126] Sanjeev J Koppal, Shuntaro Yamazaki, and Srinivasa G Narasimhan. Exploiting DLP Illumination Dithering for Reconstruction and Photography of High-Speed Scenes. *Int. J. Computer Vision*, 96(1):125–144, 2012.
- [127] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [128] Samuli Laine, Janne Hellsten, Tero Karras, Yeongho Seol, Jaakko Lehtinen, and Timo Aila. Modular primitives for high-performance differentiable rendering. *ACM Transactions on Graphics*, 39(6), 2020.
- [129] J.H. Lambert. *Photometria*. 1760.
- [130] Jeffrey Larson, Matt Menickelly, and Stefan M Wild. Derivative-free optimization methods. *Acta Numerica*, 28:287–404, 2019.
- [131] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015.
- [132] Wonkwang Lee, Donggyun Kim, Seunghoon Hong, and Honglak Lee. High-fidelity synthesis with disentangled representation. *arXiv preprint arXiv:2001.04296*, 2020.

- [133] EN Leith. The evolution of information optics. *IEEE J. Select Topics in Quantum Electronics*, 6(6):1297–1304, 2000.
- [134] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatially varying materials. In *Proc. of the 12th Eurographics Conference on Rendering*, page 103–114, 2001.
- [135] Hendrik P.A. Lensch, Jochen Lang, Asla M. Sa, and Hans-Peter Seidel. Planned sampling of spatially varying BRDFs. *Computer Graphics Forum*, 2003.
- [136] Paul Leopardi. *Distributing points on the sphere*. PhD thesis, University of New South Wales, April 2007.
- [137] Louis Lettry, Kenneth Vanhoey, and Luc Van Gool. Unsupervised deep single-image intrinsic decomposition using illumination-varying image sequences. In *Computer Graphics Forum*, volume 37, pages 409–419, 2018.
- [138] Marc Levoy, B Chen, Vaibhav Vaish, Mark Horowitz, Ian Mcdowall, and Mark Bolas. Synthetic aperture confocal imaging. In *ACM SIGGRAPH*, pages 825–834, 2004.
- [139] Ameng Li, Bruce Z Gao, Jiachen Wu, Xiang Peng, Xiaoli Liu, Yongkai Yin, and Zewei Cai. Structured light field 3D imaging. *Opt Express*, 24(18):20324–20334, 2016.
- [140] Baoxin Li and I Sezan. Automatic keystone correction for smart projectors with embedded camera. In *Proc. IEEE ICIP*, pages 2829–2832. IEEE, 2004.
- [141] F Li, J Yablon, Andreas Velten, Mohit Gupta, and Oliver S. Cossairt. High-depth-resolution range imaging with multiple-wavelength superheterodyne interferometry using 1550-nm lasers. *Appl Optics*, 56(31):H51–H56, November 2017.
- [142] Fengqiang Li, Huaijin Chen, Chiakai Yeh, Ashok Veeraraghavan, and Oliver Cossairt. High spatial resolution time-of-flight imaging. In Amit Ashok, Jonathan C Petrucci, Abhijit Mahalanobis, and Lei Tian, editors, *Computational Imaging III*, pages 7–14. SPIE, May 2018.
- [143] Francis Li, Hicham Sekkati, Jason Deglint, Christian Scharfenberger, Mark Lamm, David Clausi, John Zelek, and Alexander Wong. Simultaneous Projector-Camera Self-Calibration for Three-Dimensional Reconstruction and Projection Mapping. *IEEE TCI*, 3(1):74–83, 2017.

- [144] 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.
- [145] Xueting Li, Sifei Liu, Kihwan Kim, Shalini De Mello, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. Self-supervised single-view 3d reconstruction via semantic consistency. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [146] Zhengqi Li and Noah Snavely. Learning intrinsic image decomposition from watching the world. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9039–9048, 2018.
- [147] Zhengqin Li, Mohammad Shafiei, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2475–2484, 2020.
- [148] Zhengqin Li, Kalyan Sunkavalli, and Manmohan Chandraker. Materials for masses: Svbrdf acquisition with a single mobile phone image. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 72–87, 2018.
- [149] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. In *SIGGRAPH Asia 2018 Technical Papers*, page 269. ACM, 2018.
- [150] Chen-Hsuan Lin, Chaoyang Wang, and Simon Lucey. Sdf-srn: Learning signed distance 3d object reconstruction from static images. In *Advances in Neural Information Processing Systems*, 2020.
- [151] Zinan Lin, Kiran Koshy Thekumparampil, Giulia Fanti, and Sewoong Oh. Infogan-cr: Disentangling generative adversarial networks with contrastive regularizers. *arXiv preprint arXiv:1906.06034*, 2019.
- [152] Hsueh-Ti Derek Liu, Michael Tao, and Alec Jacobson. Paparazzi: Surface editing by way of multi-view image processing. *ACM Transactions on Graphics*, 2018.
- [153] 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 (ICLR)*, 2019.

- [154] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft Rasterizer: A Differentiable Renderer for Image-based 3D Reasoning. *arXiv:1904.01786 [cs]*, April 2019. arXiv: 1904.01786.
- [155] Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3d supervision. *Advances in Neural Information Processing Systems*, 32, 2019.
- [156] Yunfei Liu, Yu Li, Shaodi You, and Feng Lu. Unsupervised learning for intrinsic image decomposition from a single image. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [157] Matthew M. Loper and Michael J. Black. OpenDR: An approximate differentiable renderer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 8695 of *Lecture Notes in Computer Science*, pages 154–169. Springer International Publishing, September 2014.
- [158] Guillaume Loubet, Nicolas Holzschuch, and Wenzel Jakob. Reparameterizing discontinuous integrands for differentiable rendering. *ACM Trans. Graph.*, 38(6):228:1–228:14, 2019.
- [159] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.
- [160] Fujun Luan, Shuang Zhao, Kavita Bala, and Zhao Dong. Unified shape and svbrdf recovery using differentiable monte carlo rendering. *ArXiv*, 2021.
- [161] Wenjie Luo, Alexander G. Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [162] Wei-Chiu Ma, Hang Chu, Bolei Zhou, Raquel Urtasun, and Antonio Torralba. Single image intrinsic decomposition without a single intrinsic image. In *Proc. of the European Conference on Computer Vision (ECCV)*, pages 201–217, 2018.
- [163] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning*, volume 30, page 3. Atlanta, Georgia, USA, 2013.
- [164] Jerome Martin and James L Crowley. Experimental Comparison of Correlation Techniques . In *Int. Conf. on Intelligent Autonomous Systems*, 1995.

- [165] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 22(3):759–769, July 2003.
- [166] Julian Meder and Beat Bruderlin. Hemispherical Gaussians for accurate light integration. *Computer Vision and Graphics*, 2018.
- [167] Christoph Mertz, Sanjeev Jagannatha Koppal, Solomon Sia, and Srinivasa G Narasimhan. A low-power structured light sensor for outdoor scene reconstruction and dominant material identification. In *IEEE PROCAMS*, pages 15–22, 2012.
- [168] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.
- [169] C. Metzler, H. Ikoma, Y. Peng, and G. Wetzstein. Deep optics for single-shot high-dynamic-range imaging. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [170] Christopher A Metzler, Manoj K Sharma, Sudarshan Nagesh, Richard G Baraniuk, Oliver Cossairt, and Ashok Veeraraghavan. Coherent inverse scattering via transmission matrices: Efficient phase retrieval algorithms and a public dataset. In *International Conference on Computational Photography (ICCP)*. IEEE, 2017.
- [171] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [172] Parsa Mirdehghan, Wenzheng Chen, and Kiriakos N. Kutulakos. Optimal structured light à la carte. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [173] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.
- [174] Daniel Moreno, Fatih Calakli, and Gabriel Taubin. Unsynchronized structured light. In *Proc. ACM SIGGRAPH Asia*, pages 178–111, 2015.
- [175] Daniel Moreno, Kilho Son, and Gabriel Taubin. Embedded phase shifting: Robust phase shifting with embedded signals. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

- [176] Ignacio Moreno, Jeffrey A Davis, Travis M Hernandez, Don M Cottrell, and David Sand. Complete polarization control of light from a liquid crystal spatial light modulator. *Opt Express*, 20(1):364–376, 2012.
- [177] Ali Mosleh, Avinash Sharma, Emmanuel Onzon, Fahim Mannan, Nicolas Robidoux, and Felix Heide. Hardware-in-the-loop end-to-end optimization of camera image processing pipelines. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [178] Jacob Munkberg, Jon Hasselgren, Tianchang Shen, Jun Gao, Wenzheng Chen, Alex Evans, Thomas Müller, and Sanja Fidler. Extracting Triangular 3D Models, Materials, and Lighting From Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8280–8290, June 2022.
- [179] Giljoo Nam, Joo Ho Lee, Diego Gutierrez, and Min H Kim. Practical SVBRDF acquisition of 3D objects with unstructured flash photography. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 37(6):1–12, 2018.
- [180] Takuya Narihira, Michael Maire, and Stella X Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *Proc. of the IEEE International Conference on Computer Vision (ICCV)*, pages 2992–2992, 2015.
- [181] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with Noisy Labels. In *Advances In Neural Information Processing Systems*, pages 1196–1204, 2013.
- [182] Shree K Nayar, Gurunandan Krishnan, Michael D Grossberg, and Ramesh Raskar. Fast separation of direct and global components of a scene using high frequency illumination. In *ACM SIGGRAPH 2006 Papers*, pages 935–944. 2006.
- [183] Arvind Neelakantan, Luke Vilnis, Quoc V Le, Ilya Sutskever, Lukasz Kaiser, Karol Kurach, and James Martens. Adding Gradient Noise Improves Learning for Very Deep Networks. In *International Conference on Learning Representations (ICLR)*, 2017.
- [184] Merlin Nimier-David, Zhao Dong, Wenzel Jakob, and Anton Kaplanyan. Material and Lighting Reconstruction for Complex Indoor Scenes with Texture-space Differentiable Rendering. In Adrien Bousseau and Morgan McGuire, editors, *Eurographics Symposium on Rendering - DL-only Track*. The Eurographics Association, 2021.
- [185] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. Radiative backpropagation: An adjoint method for lightning-fast differentiable rendering.

- Transactions on Graphics (Proceedings of SIGGRAPH)*, 39(4), July 2020.
- [186] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 38(6), November 2019.
- [187] Matthew O’Toole, Supreeth Achar, Srinivasa G Narasimhan, and Kiriakos N Kutulakos. Homogeneous codes for energy-efficient illumination and imaging. *ACM TOG (SIGGRAPH)*, 34(4), 2015.
- [188] Matthew O’Toole, Felix Heide, Lei Xiao, Matthias B Hullin, Wolfgang Heidrich, and Kiriakos N Kutulakos. Temporal frequency probing for 5D transient analysis of global light transport. *ACM TOG (SIGGRAPH)*, 33(4), 2014.
- [189] Matthew O’Toole and Kiriakos N Kutulakos. Optical computing for fast light transport analysis. *ACM TOG (SIGGRAPH Asia)*, 29(6), 2010.
- [190] Matthew O’Toole, John Mather, and Kiriakos N Kutulakos. 3D Shape and Indirect Appearance by Structured Light Transport. *IEEE T-PAMI*, 38(7):1298–1312, 2016.
- [191] Hao Ouyang, Zifan Shi, Chenyang Lei, Ka Lung Law, and Qifeng Chen. Neural camera simulators. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7700–7709, June 2021.
- [192] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [193] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
- [194] Y. Peng, S. Choi, N. Padmanaban, and G. Wetzstein. Neural Holography with Camera-in-the-loop Training. *ACM Trans. Graph. (SIGGRAPH Asia)*, 2020.
- [195] Guim Perarnau, Joost Van De Weijer, Bogdan Raducanu, and Jose M Álvarez. Invertible conditional gans for image editing. *arXiv preprint arXiv:1611.06355*, 2016.

- [196] K. Perlin and E. M. Hoffert. Hypertexture. In *Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '89, page 253–262, New York, NY, USA, 1989. Association for Computing Machinery.
- [197] Christoph Peters, Jonathan Klein, Matthias B Hullin, and Reinhard Klein. Solving trigonometric moment problems for fast transient imaging. *ACM Trans. Graphics*, 34(6):220, 2015.
- [198] Felix Petersen, Amit H Bermano, Oliver Deussen, and Daniel Cohen-Or. Pix2vex: Image-to-geometry reconstruction using a smooth differentiable renderer. *arXiv preprint arXiv:1903.11149*, 2019.
- [199] Matt Pharr, Wenzel Jakob, and Greg Humphreys. *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.
- [200] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.
- [201] Tomislav Pribanic, Hrvoje Džapo, and Joaquim Salvi. Efficient and Low-Cost 3D Structured Light System Based on a Modified Number-Theoretic Approach. *EURASIP J. Adv. Signal Process.*, 2010.
- [202] Tomislav Pribanic, Saša Mrvoš, and Joaquim Salvi. Efficient multiple phase shift patterns for dense 3D acquisition in structured light scanning. *Image and Vision Computing*, 28(8):1255–1266, 2010.
- [203] John G Proakis. *Digital Communications*. McGraw-Hill, 2001.
- [204] Hong Qian, Yi-Qi Hu, and Yang Yu. Derivative-free optimization of high-dimensional non-convex functions by sequential random embeddings. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 1946–1952. AAAI Press, 2016.
- [205] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.
- [206] Ravi Ramamoorthi and Pat Hanrahan. An efficient representation for irradiance environment maps. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, pages 497–500, New York, NY, USA, 2001. ACM.

- [207] Netanel Ratner and Yoav Y Schechner. Illumination Multiplexing within Fundamental Limits. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2007.
- [208] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Pytorch3d. <https://github.com/facebookresearch/pytorch3d>, 2020.
- [209] Herbert Robbins and Sutton Monro. A Stochastic Approximation Method. *Ann. Math. Stat.*, 22(3):400–407, 1951.
- [210] Nicolas Robidoux, Eduardo Luis Capel García, Dong-eun Seo, Avinash Sharma, Federico Ariza, and Felix Heide. End-to-end high dynamic range camera pipeline optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021.
- [211] 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.
- [212] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011.
- [213] Bryan Russell, Antonio Torralba, Kevin Murphy, and William Freeman. Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 77(05), 2008.
- [214] J Salvi, J Pages, and J Battle. Pattern codification strategies in structured light systems. *Pattern Recogn*, 37(4):827–849, 2004.
- [215] Joaquim Salvi, Sergio Fernandez, Tomislav Pribanic, and Xavier Llado. A state of the art in structured light patterns for surface profilometry. *Pattern Recogn*, 43(8):2666–2680, 2010.
- [216] Shen Sang and M. Chandraker. Single-shot neural relighting and svbrdf estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [217] Vishwanath Saragadam and Aswin C Sankaranarayanan. KRISM—Krylov Subspace-based Optical Computing of Hyperspectral Images. *ACM TOG*, 38(5), 2019.
- [218] Guy Satat, Matthew Tancik, and Ramesh Raskar. Lensless imaging with compressive ultrafast sensing. *IEEE Trans. Comput. Imaging*, 3(3):398–407, 2017.

- [219] D Scharstein and R Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 195–202, 2003.
- [220] Yoav Y Schechner, Shree K Nayar, and Peter N Belhumeur. Multiplexing for optimal lighting. *IEEE T-PAMI*, 29(8):1339–1354, 2007.
- [221] Y.Y Schechner, S.K Nayar, and P N Belhumeur. Multiplexing for Optimal Lighting. *IEEE T-PAMI*, 29(8):1339–1354, 2007.
- [222] Christophe Schlick. A customizable reflectance model for everyday rendering. In *4th Eurographics Workshop on Rendering*, pages 73–84, June 1993.
- [223] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2016.
- [224] Tianchang Shen, Jun Gao, Amlan Kar, and Sanja Fidler. Interactive annotation of 3d object geometry using 2d scribbles. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.
- [225] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. Interfacegan: Interpreting the disentangled face representation learned by gans. *arXiv preprint arXiv:2005.09635*, 2020.
- [226] Jian Shi, Yue Dong, Hao Su, and Stella X. Yu. Learning non-Lambertian object intrinsics across ShapeNet categories. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1685–1694, 2017.
- [227] Shikhar Shrestha, Felix Heide, Wolfgang Heidrich, and Gordon Wetzstein. Computational imaging with multi-camera time-of-flight systems. *ACM TOG (SIGGRAPH)*, 35(4), 2016.
- [228] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- [229] Vincent Sitzmann, Steven Diamond, Yifan Peng, Xiong Dun, Stephen Boyd, Wolfgang Heidrich, Felix Heide, and Gordon Wetzstein. End-to-end optimization of optics

- and image processing for achromatic extended depth of field and super-resolution imaging. *ACM TOG (SIGGRAPH)*, 37(4), 2018.
- [230] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhöfer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [231] 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.
- [232] Edward Smith, Scott Fujimoto, Adriana Romero, and David Meger. GEOMETRICS: Exploiting geometric structure for graph-encoded objects. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5866–5876, Long Beach, California, USA, 09–15 Jun 2019. PMLR.
- [233] F Jiménez Spang, I Rosenberg, E Hedin, and G Royle. Photon small-field measurements with a CMOS active pixel sensor. *Phys. Med. Biol.*, 60(11):4383–4398, May 2015.
- [234] Jos Stam. Computing light transport gradients using the adjoint method. *CoRR*, abs/2006.15059, 2020.
- [235] Hao Su, Charles R. Qi, Yangyan Li, and Leonidas J. Guibas. Render for cnn: Viewpoint estimation in images using cnns trained with rendered 3d model views. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.
- [236] Shuochen Su, Felix Heide, Gordon Wetzstein, and Wolfgang Heidrich. Deep end-to-end time-of-flight imaging. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6383–6392, 2018.
- [237] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [238] Jiatian (Caroline) Sun and Ioannis Gkioulekas. Mitsuba CLT renderer. https://github.com/cmu-ci-lab/mitsuba_clt.
- [239] Qilin Sun, Ethan Tseng, Qiang Fu, Wolfgang Heidrich, and Felix Heide. Learning rank-1 diffractive optics for single-shot high dynamic range imaging. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

- [240] Attila Szabó and Paolo Favaro. Unsupervised 3d shape learning from image collections in the wild. *arXiv preprint arXiv:1811.10519*, 2018.
- [241] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.
- [242] Matthew Tancik, Vincent Casser, Xinchun Yan, Sabeek Pradhan, Ben Mildenhall, Pratul Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable large scene neural view synthesis. *arXiv*, 2022.
- [243] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3405–3414, 2019.
- [244] Ayush Tewari, Mohamed Elgharib, Gaurav Bharaj, Florian Bernard, Hans-Peter Seidel, Patrick Pérez, Michael Zöllhofer, and Christian Theobalt. Stylerig: Rigging stylegan for 3d control over portrait images, cvpr 2020. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, june 2020.
- [245] Changpeng Ti, Ruigang Yang, James Davis, and Zhigeng Pan. Simultaneous Time-of-Flight Sensing and Photometric Stereo With a Single ToF Sensor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4334–4342, 2015.
- [246] Tijmen Tieleman and Geoffrey E Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. In *COURSERA: Neural Networks for Machine Learning*, 2012.
- [247] Ethan Tseng, Shane Colburn, James Whitehead, Luocheng Huang, Seung-Hwan Baek, Arka Majumdar, and Felix Heide. Neural nano-optics for high-quality thin lens imaging. *Nature Communications*, 12(1):6493, Nov 2021.
- [248] Ethan Tseng, Felix Yu, Yuting Yang, Fahim Mannan, Karl St. Arnaud, Derek Nowrouzezahrai, Jean-Francois Lalonde, and Felix Heide. Hyperparameter optimization in black-box image processing using differentiable proxies. *ACM Transactions on Graphics (TOG)*, 38(4), 7 2019.
- [249] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

- [250] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.
- [251] Julien Valentin, Cem Keskin, Pavel Pidlypenskyi, Ameesh Makadia, Avneesh Sud, and Sofien Bouaziz. Tensorflow graphics: Computer graphics meets deep learning. <https://www.tensorflow.org/graphics>, 2019.
- [252] G. Van Horn, S. Branson, R. Farrell, S. Haber, J. Barry, P. Ipeirotis, P. Perona, and S. Belongie. Building a bird recognition app and large scale dataset with citizen scientists: The fine print in fine-grained dataset collection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 595–604, 2015.
- [253] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for Monte Carlo rendering. In *Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '95*, pages 419–428, 1995.
- [254] Andreas Velten, Thomas Willwacher, Otkrist Gupta, Ashok Veeraraghavan, Mounsi G. Bawendi, and Ramesh Raskar. Recovering three-dimensional shape around a corner using ultrafast time-of-flight imaging. *Nature Communications*, 3(1):745, Mar 2012.
- [255] Dor Verbin, Peter Hedman, Ben Mildenhall, Todd Zickler, Jonathan T. Barron, and Pratul P. Srinivasan. Ref-NeRF: Structured view-dependent appearance for neural radiance fields. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [256] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Path replay backpropagation: Differentiating light paths using constant memory and linear time. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 40(4):108:1–108:14, August 2021.
- [257] Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Differentiable signed distance function rendering. *Transactions on Graphics (Proceedings of SIGGRAPH)*, 41(4):125:1–125:18, July 2022.
- [258] Gregory K Wallace. The jpeg still picture compression standard. *IEEE transactions on consumer electronics*, 38(1):xviii–xxxiv, 1992.
- [259] Jiaping Wang, Peiran Ren, Minmin Gong, John Snyder, and Baining Guo. All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 28(5):1–10, December 2009.

- [260] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.
- [261] Qianqian Wang, Xiaowei Zhou, Bharath Hariharan, and Noah Snavely. Learning feature descriptors using camera pose supervision. In *Proc. European Conference on Computer Vision (ECCV)*, 2020.
- [262] 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.
- [263] Weiyue Wang, Xu Qiangeng, Duygu Ceylan, Radomir Mech, and Ulrich Neumann. Disn: Deep implicit surface network for high-quality single-view 3d reconstruction. *arXiv preprint arXiv:1905.10711*, 2019.
- [264] Zian Wang, Jonah Philion, Sanja Fidler, and Jan Kautz. Learning indoor inverse rendering with 3d spatially-varying lighting. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2021.
- [265] Mian Wei, Navid Sarhangnejad, Zhengfan Xia, Nikita Gusev, Nikola Katic, Roman Genov, and Kiriakos N Kutulakos. Coded Two-Bucket Cameras for Computer Vision. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 54–71, 2018.
- [266] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.
- [267] Chao Wen, Yinda Zhang, Zhuwen Li, and Yanwei Fu. Pixel2mesh++: Multi-view 3d mesh generation via deformation. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2019.
- [268] Shangzhe Wu, Ameesh Makadia, Jiajun Wu, Noah Snavely, Richard Tucker, and Angjoo Kanazawa. De-rendering the world’s revolutionary artefacts. *arXiv preprint arXiv:2104.03954*, 2021.
- [269] Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–10, 2020.

- [270] Zhirong Wu, Shuran Song, Aditya Khosla, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [271] Rui Xia, Yue Dong, Pieter Peers, and Xin Tong. Recovering shape and spatially-varying surface reflectance under unknown illumination. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 35(6), 2016.
- [272] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014.
- [273] Lingxi Xie, Jingdong Wang, Zhen Wei, Meng Wang, and Qi Tian. DisturbLabel: Regularizing CNN on the Loss Layer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4753–4762, 2016.
- [274] Yazhou Xing, Zian Qian, and Qifeng Chen. Invertible image signal processing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6287–6296, June 2021.
- [275] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.
- [276] Kun Xu, Wei-Lun Sun, Zhao Dong, Dan-Yong Zhao, Run-Dong Wu, and Shi-Min Hu. Anisotropic spherical gaussians. *ACM Trans. Graph. (Proc. SIGGRAPH)*, 32(6), November 2013.
- [277] Yi Xu and Daniel G Aliaga. Robust pixel classification for 3d modeling with structured light. In *Proc. Graphics Interface*, pages 233–240, 2007.
- [278] Xinchun Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. *Advances in neural information processing systems*, 29, 2016.
- [279] Shunyu Yao, Tzu Ming Hsu, Jun-Yan Zhu, Jiajun Wu, Antonio Torralba, Bill Freeman, and Josh Tenenbaum. 3d-aware scene manipulation via inverse graphics. In *Advances in Neural Information Processing Systems*, pages 1887–1898, 2018.
- [280] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH ASIA)*, 38(6), 2019.

- [281] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015.
- [282] Ye Yu and William AP Smith. Inverserendernet: Learning single image inverse rendering. In *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [283] Qiong Zeng, Wenzheng Chen, Huan Wang, Changhe Tu, Daniel Cohen-Or, Dani Lischinski, and Baoquan Chen. Hallucinating stereoscopy from a single image. In *Computer Graphics Forum*, volume 34, pages 1–12. Wiley Online Library, 2015.
- [284] Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. Path-space differentiable rendering. *ACM Trans. Graph.*, 39(4):143:1–143:19, 2020.
- [285] Cheng Zhang, Lifan Wu, Changxi Zheng, Ioannis Gkioulekas, Ravi Ramamoorthi, and Shuang Zhao. A differential theory of radiative transfer. *ACM Trans. Graph.*, 38(6):227:1–227:16, 2019.
- [286] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.
- [287] Jie Zhang, Ralph Etienne-Cummings, Sang Chin, Tao Xiong, and Trac Tran. Compact all-CMOS spatiotemporal compressive sensing video camera with pixel-wise coded exposure. *Opt Express*, 24(8):9013–9024, 2016.
- [288] Kai Zhang, Fujun Luan, Qianqian Wang, Kavita Bala, and Noah Snavely. Physg: Inverse rendering with spherical gaussians for physics-based material editing and relighting. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [289] Li Zhang, Brian Curless, and Steven M Seitz. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *Proc. IEEE 3DPVT*, pages 24–36, 2002.
- [290] 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 (CVPR)*, 2018.
- [291] Yuxuan Zhang, Wenzheng Chen, Huan Ling, Jun Gao, Yinan Zhang, Antonio Torralba, and Sanja Fidler. Image gans meet differentiable rendering for inverse graphics

- and interpretable 3d neural rendering. In *International Conference on Learning Representations (ICLR)*, 2021.
- [292] Zhengyou Zhang. Parameter estimation techniques: a tutorial with application to conic fitting. *Image and Vision Computing*, 15(1):59–76, 1997.
- [293] Hong Zhao, Wenyi Chen, and Yushan Tan. Phase-unwrapping algorithm for the measurement of three-dimensional object shapes. *Appl. Opt.*, 33(20):4497–4500, Jul 1994.
- [294] Ellen D. Zhong, Tristan Bepler, Bonnie Berger, and Joseph H. Davis. Cryodrgn: reconstruction of heterogeneous cryo-em structures using neural networks. *Nature Methods*, 18(2):176–185, Feb 2021.