University of York
Department of Computer Science

Physics-based Vision Meets Deep Learning

Ye Yu

March 2021

Abstract

Physics-based vision explores computer vision and graphics problems by applying methods based upon physical models. On the other hand, deep learning is a learning-based technique, where a substantial number of observations are used to train an expressive yet unexplainable neural network model. In this thesis we propose the concept of a model-based decoder, which is an unlearnable and differentiable neural layer being designed according to a physics-based model. Constructing neural networks with such model-based decoders afford the model strong learning capability as well as potential to respect the underlying physics.

We start the study by developing a toolbox of differentiable photometric layers ported from classical photometric techniques. This enables us to perform the image formation process given geometry, illumination and reflectance function. Applying these differentiable photometric layers into a bidirectional reflectance distribution function (BRDF) estimation network training, we show the network could be trained in a self-supervised manner without the knowledge of ground truth BRDFs.

Next, in a more general setting we attempt to solve inverse rendering problems in a self-supervised fashion by making use of model-based decoders. Here, an inverse rendering network decomposes a single image into normal and diffuse albedo map and illumination. In order to achieve self-supervised training, we draw inspiration from multiview stereo (MVS) and employ a Lambertian model and a cross-projection MVS model to generate model-based supervisory signals.

Finally, we seek potential hybrids of a neural decoder and a model-based decoder on a pair of practical problems: image relighting, and fine-scale depth prediction and novel view synthesis. In contrast to using model-based decoders to only supervise the training, the model-based decoder in our hybrid model serves to disentangle the intricate problem into a set of physically connected solvable ones. In practice, we develop a hybrid model that can estimate a fine-scale depth map and generate novel view synthesis from a single image by using a physical subnet to combine results from an inverse rendering network with a monodepth prediction network. As for neural image relighting, we propose another hybrid model using a Lambertian renderer to generate initial estimates of relighting results followed by a neural renderer performing corrections over deficits in initial renderings.

We demonstrate the model-based decoder can significantly improve the quality of results and relax the demands for labelled data.

Acknowledgements

I want to express my appreciation and gratitude to the people who physically, mentally, academically and domestically support my PhD journey.

First, I am sincerely grateful to my PhD supervisor, Dr. William Smith for all his priceless and innovative advice on my research and career. Throughout my PhD study, Will constantly provides me necessary and helpful guidance whenever I come across problems no matter in my study or outside of school. Whenever my research is trapped into a bottleneck, his resourceful knowledge and endless ideas could immediately get me back on track. He also provides me precious opportunities to build connections with specialists in my research topic and encourages me to join collaborative projects with other reputed researchers. His broad research interests motivates and drives me to be always curious about and willing to learn new knowledge and subjects.

I also want to take the opportunity to thank researchers I have collaborated with during my study. Prof. Christian Theobalt invited and hosted me for a short visit in Max Planck Institute (University of Saarland). During my visit to MPI, Dr. Mohamed Elgharib and Dr. Abhimitra Meka provided me with kind help for my life and valuable assistance for research progress. For the collaborative project, Dr. Owais Mehmood could always swiftly reply to my data requests and resolve my confusions regarding the data. As my internal assessor, Prof. Richard Wilson would always propose inspiring and insightful questions in our assessment talk.

My friends and peers working around me teach and inspire me a lot. Dizhong, Sarah, Anil, Jianjia, Bruce, Ibrahim, Chao, Guoxi, Tatsuro, Enes - I would like to express my special thanks to them for forging a wonderful working and researching environment.

To my parents, it was their unconditional support and encouragement that drove me to pursue my PhD, and their unchanging love and care carried me to this point. I truly appreciate their understanding when I could not go home frequently during the study and their massive but under-appreciated contributions in this whole journey.

Finally, I would like to thank my wife, Qianru Yang for supporting my research life during the last four years. From the first day of my PhD, she has been devoting great patience and care to me especially when I struggled with negative emotions and unhealthy conditions. Her optimism always encourages me to confront difficulties. Her company forms a warm and comfortable home for me, without which

my PhD life would be much tougher and tedious.

Declaration

I declare that this thesis is a presentation of original work and I am the sole author. This work has not previously been presented for an award at this, or any other, University. The contributions contained in this thesis are my own work. Main contents have been published in the following papers. For all these works I made the major contribution in design, implementation and experiments. All sources are acknowledged as References.

- Ye Yu and William AP Smith. "PVNN: A neural network library for photometric vision." Proceedings of the IEEE International Conference on Computer Vision Workshops. 2017.
- Ye Yu and William AP Smith. "InverseRenderNet: Learning single image inverse rendering." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019.
- Ye Yu, William AP Smith, "Depth estimation meets inverse rendering for single image novel view synthesis." European Conference on Visual Media Production. 2019.
- Ye Yu, Abhimitra Meka, Mohamed Elgharib, Hans-Peter Seidel, Christian Theobalt and William AP Smith, "Self-supervised Outdoor Scene Relighting." Proceedings of the European Conference on Computer Vision. 2020.
- Ye Yu and William AP Smith, "Outdoor inverse rendering from a single image using multiview self-supervision.", in IEEE Transactions on Pattern Analysis & Machine Intelligence.

Ye Yu

March 2021

Contents

List of Tables

List of Figures

Chapter 1

Introduction

RGB images are formed by complicated interactions between lighting, scene and camera. Based on our understandings of lighting transportation in the space, this image formation process can be described by the underlying physical models. Physics-based vision represents a class of researches using such image formation physical models to estimate intrinsic properties of the captured scene. Practical applications regarding this topic includes reflectance model estimation, illumination estimation and so on. For example, the Lambertian reflectance can model perfect diffuse reflections of a matte surface. More generally, the Bidirectional Reflectance Distribution Function (BRDF) of an object explains the directional dependence of light reflecting from the surface and can be estimated given a collection of images captured in calibrated experimental environments. Under this definition, the employed physical models are only related to image formation process. However, in this thesis, we would like to explore the potential benefits when deep learning model meets physical model, which should include the models not only simulating the image formation process but also being exploited in more general vision methods. For this reason, we firstly need to broaden the definition of physics-based vision by including more vision techniques built from underlying physical models. For example, multi-view stereo (MVS), which is a very robust 3D reconstruction vision technique, relies on the physics of 3D geometry and camera perspective projection. Through performing optimisation over such physical models, the objective semantic properties of captured scenes can be inferred. Under our new definition, however, we can find physics-based vision methods are often inherently imperfect, like

the Lambertian reflectance model ignores the effects of specularities. MVS suffers from requiring a large number of input images and fails in untextured regions devoid of matchable features. Thus, this thesis presents a study on a promising candidate for alleviating potential issues in physics-based vision methods, which is the combination of deep learning framework and differentiable physics-based network modules ported from the physical vision models.

Over the last decade, deep learning has become the most powerful tool in computer vision and computer graphics. Learning-based algorithms built on deep neural networks have achieved unprecedented successes in many fundamental problems in these fields, for example monocular depth estimation [84, 26, 25, 29, 152, 39], semantic image segmentation [169, 90, 55, 48], image and video rendering [106, 66, 119, 105, 142], etc. The explosion of deep learning related algorithms initiated from ones attacking recognition problems [72], where the model was trained in an end-to-end setting while assuming that the database along with ground truth is large and diverse enough for training. Such a CNN, however, contains no semantic or interpretable operational kernels inside the network but is only a problem-oriented black box. More importantly, the prerequisite of large-scale labelled data restricts the scope of its applicability. To address these inherent limitations in deep learning method, combining model- and learning-based vision could provide us with a better solution than simply learning a black-box model with annotations. One way to do this is to incorporate physical models into a CNN by introducing layers that explicitly implement physical models. This potentially offers a route to unsupervised learning of physics-based vision tasks with CNNs. For example, a black box encoder that transforms images to physically meaningful parameters can be trained by pairing it with a physics-based decoder that renders an image from the estimated parameters allowing a self-supervised loss to be computed between the original and rendered images. Many other architectures are also possible that exploit these physics-based layers.

More recently, increasing attempts to embed physics-based vision techniques into deep neural networks have been proposed. Compared with preceding CNN-based methods, the embedded physical model introduces the interpretability of underlying physical processes so as to regularise the learned model, and it provides a physics-based supervisory signal to achieve self-supervised learning which relaxes the need of data labels. This family of works inspires us to study the potential that incorporating classical physics-based vision techniques into CNN can improve the performance over the

existing physics-based vision and deep learning methods.

An intuitive way of combining physics-based vision and deep learning is to construct a subnet inside the existing CNN that simulates the physical process. In BRDF modelling, for example, a subnet reproducing observed images with BRDF function estimated by a deep BRDF modelling network enables self-supervised learning. To achieve similar self-supervised training, another example could be the Lambertian reflectance model being used as an image formation decoder in image decomposition networks. There are many more possibilities in combining two models due to the versatility of CNNs and the great variation of physics-based vision algorithms. In common practice, physics-based neural embedding layer is structured subsequently to an encoder or a decoder, and it works as an untrainable decoder, so we call it model-based decoder as opposed to neural decoder, the latter consisting of trainable parameters.

In this thesis, we will first seek the insight of the definition of model-based decoder by introducing differentiable neural network layers for photometric vision and simple Lambertian model, and we will show that they are essential components for self-supervised BRDF estimation and inverse rendering networks. Next, we present a self-supervised inverse rendering network as well as a model-based decoder implementing physical model of multi-view stereo that serves to enable self-supervised training. To demonstrate their effectiveness, both qualitative and quantitative evaluation is performed on the BRDF modelling network and inverse rendering network. Here, model-based decoders are used in the training phase to grant the training process a self-supervisory loss signal, therefore allowing our proposed network to be trained without ground truth labels. The learned model itself, however, remains structurally free from model-based decoders. Instead of only applying model-based decoders in training, an alternative is to embed model-based decoders into neural networks such that the learned model for inference is a union of two types of decoders. Also, from a modelling perspective, the model-based decoder has degraded performance compared to learnable neural decoder yet retaining the physical properties over neural decoder, which encourages us to build neural networks with both model-based decoder and neural decoder. To this end, we present another category of work focusing on the hybrid model of model-based and neural decoder. For this, we show a depth estimation network yielding detailed depth predictions by imposing a derivative relationship between deep depth and normal estimate, and an image relighting renderer employing a rendering network to transfer a

Lambertian rendering exhibiting deficits to a photorealistic rendering. Their impressive performance again confirms the significant advantages conferred by model-based decoders.

We develop our study for the topic of "Physics-based vision meets deep learning" by approaching different vision or graphic problems mentioned above using varying combinations of the model-based decoder and neural networks. To better illustrate them, the detailed problem specific contexts are introduced in the rest of this chapter.

1.1 Self-supervised neural network for BRDF estimation

The key to modelling photometric image formation is the knowledge of reflectance properties of the target object, namely Bidirectional Reflectance Distribution Functions (BRDF). A BRDF is a 4D function describing the ratio relationship between exitant radiance and incident irradiance, where function inputs are incident and exitant directions represented by two individual 3D vectors (each with 2 angular degrees of freedom, hence the BRDF is a 4D function. The level of fidelity of BRDFs employed in the rendering process considerably affects the level of realism of the rendered images. To capture high-fidelity BRDFs, standardised capturing devices have been proposed in some early works, which are prohibitively high-end and complicated. Then more efforts have been made to simplify the cumbersome BRDF measuring procedure by using image-based methods, spherical gantries etc. These methods proposed to efficiently capture BRDF data followed by fitting the raw data to BRDF models so as to achieve efficient storage and computation.

Unlike classical methods outlined above, learning-based approaches make use of deep learning techniques to directly estimate BRDFs of captured objects by one or a few unstructured input images. This branch of work provides faster performance and greater generality, while strictly requiring sufficient data for training network parameters. The challenge in such learning-based approaches is the acquisition of training data since acquiring images with accurate reflectance functions is itself an open problem. One way around this is to use synthetic training data, where the images are rendered from models. However, the ability of CNNs to generalise to real world data is limited by the realism of the synthetic training data. Moreover, such a black box reveals nothing about the underlying physical

processes, and the model must be retrained in order to estimate different physical quantities or solve various problems. In Chapter 3, we present a differentiable renderer supporting tabulated BRDF representation and show how to use the renderer to overcome both problems of necessitating labelled training data and of lacking interpretability in terms of the physical process. As a result, we show a BRDF estimation network trained in a self-supervision regime.

1.2 Deep inverse rendering

Inverse rendering is the problem of estimating one or more of illumination, reflectance properties and shape from observed appearance (i.e. one or more images). In Chapter 4, we tackle the most challenging setting of this problem; we seek to estimate all three quantities from only a single, uncontrolled image of an arbitrary scene. Specifically, we estimate a normal map, diffuse albedo map, cast shadow map and spherical harmonic lighting coefficients. This subsumes, and is more challenging than, several classical computer vision problems: (uncalibrated) shape-from-shading, intrinsic image decomposition, shadow removal and lighting estimation.

Classical approaches [7, 79] cast these problems in terms of energy minimisation. Here, a data term measures the difference between the input image and the synthesised image that arises from the estimated quantities. We approach the problem as one of image to image translation and solve it using a deep, fully convolutional neural network. However, inverse rendering of uncontrolled, outdoor scenes is itself an unsolved problem, and so labels for supervised learning are not available. Instead, we use the data term for self-supervision via a differentiable renderer.

Single image inverse rendering is an inherently ambiguous problem. For example, any image can be explained with zero data error by setting the albedo map equal to the image, the normal map to be planar and the illumination such that the shading is unity everywhere. Hence, the data term alone cannot be used to solve this problem. For this reason, classical methods augment the data term with generic [7] or object-class-specific [5] priors. Likewise, we also exploit a statistical prior on lighting. However, our key insight that enables CNN to learn good performance is to introduce additional supervision provided by an offline multiview reconstruction or from a time-lapse video.

While photometric vision has largely been confined to restrictive lab settings, classical geometric methods are sufficiently robust to provide multiview 3D shape reconstructions from large, unstructured datasets containing very rich illumination variation [40, 31]. This is made possible by local image descriptors that are largely invariant to illumination. However, these methods recover only geometric information and any recovered texture map has illumination "baked in" and so is useless for relighting. We exploit the robustness of geometric methods to varying illumination to supervise our inverse rendering network. We apply a multiview stereo (MVS) pipeline to large sets of images of the same scene. We select pairs of overlapping images with different illumination, use the estimated relative pose and depth maps to cross-project photometric invariants between views and use this for supervision via Siamese training. Here, MVS geometry enables us to align invariants in images with different illuminations and from different viewpoints, and then compute and minimise the distance between aligned invariants. The similar methods is usually used in training networks on time-lapse images. In other words, geometry provides correspondence that allows us to simulate varying illumination from a fixed viewpoint. Finally, the depth maps from MVS provide coarse normal map estimates that can be used for direct supervision of the normal map estimation. This is important for our training schedule but note that the subsequent self-supervised learning allows the network to perform significantly better normal map estimation than simply training a network in a supervised fashion with MVS normal maps. Time-lapse video provides another useful constraint since albedo and surface normal maps remain constant while lighting (and therefore shading and shadowing) vary.

1.3 Depth estimation meets inverse rendering

Depth estimation from a single image is usually learnt in a supervised fashion, where the training examples come from a depth sensor [153] or multiview stereo reconstructions from image collections [84]. These methods are now capable of providing robust performance on real world images and the recovered depth often correctly estimates ordinal relationships between objects and components within a scene. By texture-mapping the depth map with the original image, it is possible to synthesise new images with a novel viewpoint. However, the depth maps estimated by these approaches do not capture fine-scale local detail, and they do not provide any reflectance or lighting estimates. So, the

output cannot be used for rendering novel illumination conditions, i.e. for *relighting*.

In a different direction, inverse rendering (or *intrinsic image decomposition*) provides a decomposition of the image into shading and reflectance, perhaps with shading further decomposed into a normal map and lighting estimate. Here, the challenge is that no existing method can provide training examples for real world scenes since inverse rendering in the wild is still an open problem. For this reason, state-of-the-art methods use self-supervision (see Chapter 4). Estimated surface normal and albedo maps are sufficient for relighting and often capture fine-scale, high frequency shape details. However, with no absolute geometric information, viewpoints cannot be edited and cast shadows cannot be predicted.

In Chapter 5, we take a model-based decoder to refine geometric related predictions from deep depth estimation and deep inverse rendering networks, which yields depth predictions that capture fine surface shape detail.

1.4 Single-image neural relighting

Virtual relighting of real world outdoor scenes is an important problem that has wide applications. For example, synthesised relighting images could be used in film creations. Image relighting technique can also be used in content creations for virtual reality applications and video games. The image relighting technique can generate time-lapse videos of the target scene, which is an inspiring and efficient tool for architectural designer. Performing such a relighting task involves correctly estimating and editing the various scene components – geometry, reflectance and the direct and indirect lighting effects. Measuring these high-dimensional parameters traditionally required the use of instruments such as LIDAR scanners and gonioreflectometers and extensive manual effort [145, 148].

Multi-view and multi-illumination constraints have proved to be effective in solving this problem [75, 24, 122]. 2D images of a scene from different viewpoints and under different lighting conditions provide the necessary constraints to reconstruct the geometry of the scene and disambiguate the lighting from the reflectance. For example, the method of Laffont *et al*. [75], along with multi-view 3D

reconstruction, also uses manual interactions to perform an intrinsic decomposition of the scene images into reflectance and shading layers. By reprojecting the reflectance layer from one viewpoint to another and recombining with the original shading image, lighting conditions of one image of a scene can be transferred to another. While this technique is effective, it is limited in its relighting capability because it cannot relight the scene under an arbitrary lighting condition of choice. The method of Duchêne *et al*. [24] also performs a similar intrinsic decomposition of multi-view images, and additionally estimates the shadows and the parameters of a sun-lighting model for the scene. These parameters are then modified in a geometrically accurate way to achieve scene relighting. Philip *et al*. [122] similarly estimate shadows and sun-light model parameters but skip the inverse rendering process and instead use a deep neural network to generate relighting results directly. Their network takes as input several 'illumination buffers' that are rendered using the reconstructed geometry and estimated sun-light model parameters. This method relies on high-quality ground-truth renderings of synthetic 3D models of outdoor scenes, requiring the availability of high-end computational hardware. While these techniques have been shown to generate high-quality relighting results on real scenes, they are limited by the availability of multi-view images of the scene. They also rely on a sun-lighting model that only works for bright sunlight conditions and does not generalise to cloudy overcast skies, night-time lighting, or other desired target illumination conditions.

Another class of methods circumvent the problem of estimating the scene parameters by achieving relighting directly through lighting style transfer. These methods [136, 77] change the lighting in a scene by learning the colour characteristics of images at different times of the day. Another set of methods [71, 93] learn a more general class of style-transfer in which characteristics of a reference image are transferred to a target image, including the scene lighting. Such methods are not physically based and are limited in relighting a scene either based on a reference image or a particular time of the day.

In contrast, the method of inverse rendering network (Chapter 4) proposes a novel formulation for the problem that allows for fully controlled relighting based on a single image of the scene. It demonstrates a learning method that at training time uses the constraints available from multi-view casual images of outdoor scenes sourced from the internet, to learn to estimate the scene appearance parameters. The network can then at test time estimate these parameters from a single image. By

modifying the lighting to the desired lighting environment, the image can be relit. While this method enables relighting of a scene from a single 2D image to any arbitrary lighting, it is also limited by the low-frequency lighting model used in the decomposition that leads to non-photorealistic relighting results.

In Chapter 5, we present a hybrid model combining a Lambertian rendering model with a neural decoder. In this hybrid model, the embedded Lambertian model provides us with the controllability of new lighting, and the learned neural decoder is able to generate high quality relighting results that introduce non-Lambertian effects being missed by pure Lambertian model.

1.5 Outline

We study the theme of "Physics-based vision meets deep learning" under different frameworks and applications, which are organised as follows:

- Chapter 2: We present related work regarding the general discussion of the model-based decoder and specific applications relevant to our proposed frameworks.
- Chapter 3: A set of concrete model-based decoders are defined in this chapter. For qualifying these model-based decoders, we present experiments and an exemplar BRDF estimation network, empirically showing their contributions in deep learning.
- Chapter 4: We show how to use a model-based decoder to achieve self-supervised training in learning a deep inverse rendering network.
- Chapter 5: We broaden our discussion onto the hybrid model of model-based and neural decoder and demonstrate its effectiveness on two applications, one each for neural networks of detailed depth prediction and self-supervised single image relighting renderer.
- Chapter 6: A discussion of conclusions and future works based on each topic presented in this thesis is provided in this chapter.

Chapter 2

Related work

In this chapter, we will first review related work on differentiable renderers derived from physical models inside a deep neural network. We take the perspective from this branch of work for analysing model-based decoders as it has been an important and well-studied field. As a comparison, recent works about neural renderers will be reviewed in the same section. The main theme of this thesis is organised by reviewing how to leverage the problem-orientated model-based decoder in different practical problems: BRDF estimate, inverse rendering, depth estimate refinement and image relighting. Hence a thorough review covering these practical topics is provided in the rest of this chapter.

2.1 Model-based and neural decoders

In order to include physics-based models in a CNN, the key component is a differentiable layer that can integrate specific physical models into a feed-forward neural network whilst enabling back-propagation during training. Hence, we begin by reviewing related work that utilises trainable or fixed, differentiable renderers for vision tasks.

2.1.1 Differentiable renderer

In energy minimisation approaches to inverse graphics problems, there is a need for differentiable renderers that model the physical process of image intensity formation. This allows minimisation of an appearance error using gradient descent or other first-order optimisation algorithms. Loper *et al*. [91] developed differentiable renderer package called OpenDR as a framework. OpenDR could reproduce the observed images with latent variables like appearance, geometry and camera, then the error between reproduced image intensity map and the observation can be minimised. Zienkiewicz *et al*. [175] applied a similar idea for estimating height maps in a real-time robotic system. The rendering objective is height map rather than RGB intensities in their case.

2.1.2 Trainable render layer

Some work has investigated the idea of a trainable renderer. Dosovitskiy *et al*. [21] proposed a neural network for image reconstruction from extracted feature maps resulting from different hand-crafted or learned descriptors like HOG, SIFT or trained deterministic CNN. They trained an up-convolutional network against the loss function comparing original inputs and reconstruction. Similarly, using differentiable rendering layers inside a network to perform image reconstruction were implemented by [22, 44, 35, 171, 73]. All trainable rendering networks introduced by these works are generative models simulating the probabilistic process of RGB image synthesis. In contrast, our network models a discriminative process on statistical BRDF parameter estimation problem and uses a fixed rendering architecture to formulate the loss function.

Goodfellow *et al*. [43] proposed the idea of Generative Adversarial Nets (GANs), which trains a generative model together with a discriminator. The discriminator is trained to classify real and generated images, and generator is trained to fool the discriminator. This adversarial training mechanism encourages the generative model to generate realistic images capturing the hidden features in the input images. Radford *et al*. [123] applied the adversarial training with convolutional neural networks in their DCGANs. By employing CNN-based generative and discriminative models, their generative model could be trained in an unsupervised fashion to generate realistic images of different domains.

The trained generator in GANs, however, can only take input of random noise vectors but cannot perform rendering based on semantic inputs. Jaderberg *et al*. [53] proposed a plugin renderer layer to seamlessly work along with existing CNN architectures, in order to activate the network's capability on recognising images with spatial transformation. Hinton *et al*. [50] proposed the idea of capsules which are designed to capture features along with their positions, and used these capsules to construct their transforming auto-encoders. The capsule unit enables neural networks to extract translation invariant features while remaining the position information. Following the similar idea, another work that considers positional information is Spatial Transformer Networks presented by Jaderberg *et al*. [53], which introduces spatial transformation unit into regular convolutional neural networks. This new method provides the networks with the ability to perform image transformation, such that convolutional layers can localise the region of interest.

Nalbach *et al*. [109] explored the graphical shading problem by using rendering layers. Specifically, their Deep Shading net explicitly takes as input the meaningful scenery attributes defined in common shaders and use well-trained neural network to act as a self-taught shader as opposite to manually designed shader. Gregor *et al*. [44] employed an recurrent architecture in their DRAW net that learns to generate images in an iterative manner. They applied attention modules to make their network only focus on part of the image canvas at each step, and hence accumulating and gathering the generated pieces in the final results. In each step, the encoder generates latent codes conditioned on the input image and previous outputs from decoder, which is combined with attention model to achieve impressive performance. Gatys *et al*. [35] tackled the problem of auto-generate images with respect to the specified artistic style. The fundamental insight supporting the method is that the representations of content and style are independently captured by hierarchically different sub-networks inside CNNs. Thus the target image could be generated by minimising the content matching loss and style matching loss derived from a pre-trained VGG net [140]. Dosovitskiy *et al*. [22] developed a model to render images given scenes description parameters, like camera position and appearance style. But the effectiveness of the generator is only validated by synthetic laboratory images. Zhmoginov and Sandler [171] proposed a face image rendering network being able to construct face image from a pair of guiding image and identity embedding vector. Similar to Gatys *et al*. [35], Zhmoginov and Sandler [171] formulated network training losses by measuring the feature distance computed by the

pre-trained FaceNet, which is proposed by Schroff *et al*. [130].

2.1.3 Render layers in CNNs

Another group of differentiable rendering layer embedded in CNN are untrainable rendering layers that usually make their network attack a specific problem. The BRDF estimation network proposed in Chapter 3 falls into this group of work. The attractive feature of such a fixed rendering layer is that the decoder acts as prior based on assumed physical model that can provide constraint during unsupervised training. Richardson *et al*. [126] facilitated such a hand-crafted rendering layer for face images. Their fixed differentiable rendering layer works on finetuning results estimated from a preceding discriminative model, and it plays the role of encouraging similarity between rendered images and original input face images. Tung *et al*. [150] concentrated on a similar idea of inverse graphics and incorporate an adversarial architecture into the network. By employing physics-based rendering layers in conjunction with GANs architecture [43], their network can be trained in a self-supervised way to deal with many different vision problems ranging from human post estimation, structure from motion (SfM), super-resolution and inpainting. This paper introduces a general discussion about how physics-based neural network layer can be used to train network on various physics-based problems. A demonstration diagram is visualisation in Figure 2.1. Specifically, the human skeleton projection layer is employed in human pose estimation network, a multi-view stereo cross-projection layer facilitates self-supervised SfM network, a downsamping layer provides self-supervision for learning super-resolution network, and a rendering layer that applies binary masking operation is used to train an inpainting network. Their network setups are too simple to provide strong and reliable supervision signals, hence their training scheme heavily relies on the adversarial loss. Along with this direction, we are exploring the similar topics like using physical model behind MVS and Lambertian reflectance to supervise our network training. But we propose to use multiple models together to better constraint the learning problem rather than heavily depending on adversarial loss.

A variety of works [33, 38, 47, 146, 161] focused on specific physics-based vision problems and realised unsupervised learning by utilising deep learning in conjunction with a problem-oriented rendering layer. Some efforts has been made to apply unsupervised methods to train monodepth prediction

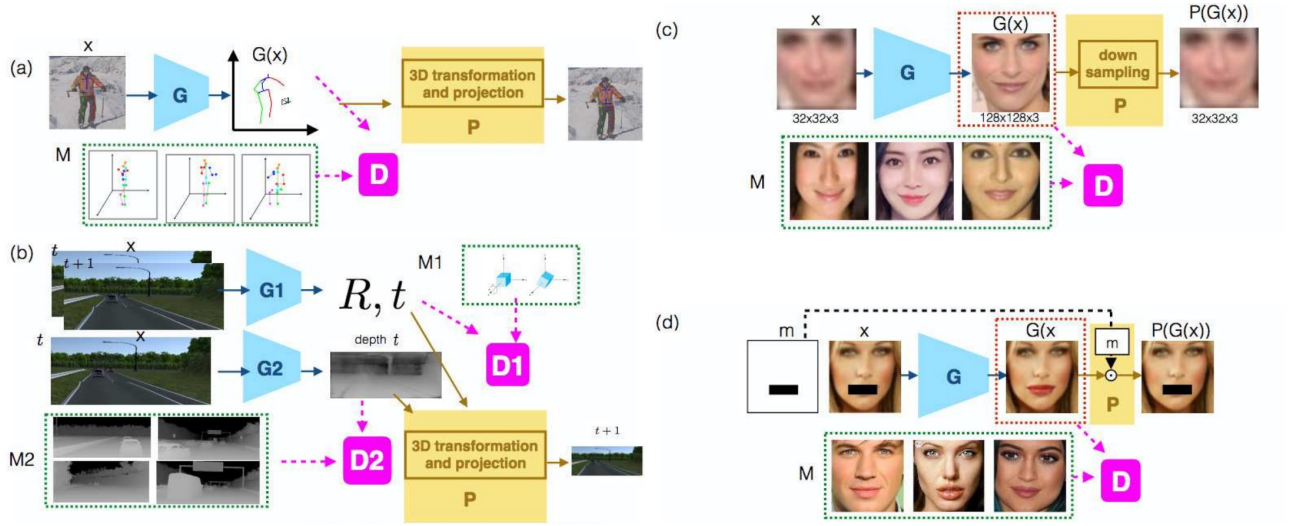


Figure 2.1: Samples of different model-based decoders in neural network training. 3D Human pose estimation can be projected to the original image to formulate self-supervised network training (a). This idea can be extended to other applications like depth estimation network supervised by stereo image pair (b), super-resolution image generation network supervised by downsampling based self-reconstruction (c), and image inpainting network supervised by self-reconstruction combined with binary masking layer. Image taken from [150].

networks, like Garg *et al*. [33] and Godard *et al*. [38]. These two works share the same fundamental motivation that the correct depth prediction allows a pair of images with known baseline and focal length to align with each other. Godard *et al*. [38] outperforms Garg *et al*. [33] by introducing additional left-right consistency loss and structural appearance loss. Handa *et al*. [47] introduced a library of differentiable neural network layers providing implementations for frequently used operations defined in the field of geometric computer vision. Namely, the library includes 3D rotation, translation and scaling, perspective projection for 3D points, pixel-wise affine transformation, and loss functions based on different M-estimators. A practical application is shown to validate their library, where they trained a network to learn odometry from a pair of images in an unsupervised manner. In an attempt of using unsupervised training to learn an inverse rendering networks for face images, Tewari *et al*. [146] proposed to decompose a facial image into model-based geometry, diffuse reflectance and low-frequency lighting. The model-based decoder in their model performs physics-based image formation to reconstruct the input image, hence providing supervisory signals without ground truth labels. Another unsupervised training attempt to estimate 3D object skeletons was proposed by Wu *et al*. [161]. A projection layer mapping the 3D skeleton estimate to the input 2D image is used in their network, which bridges the gap between the skeleton in 3D and the keypoint annotations in 2D.

Other than using rendering layers as a tool for learning in an unsupervised setting, Cole *et al*. [17] achieved great frontal-face rendering results by using a warping layer with the landmark and texture extraction networks to generate input image reconstruction. Their loss function consists of two portions: errors of landmark and texture predictions and error of rendered frontal-face image and both of them need ground truth in training. Tang *et al*. [144] assumed Lambertian reflectance in their render layer and perform a multiplicative rendering process with the outputs from their generative model. Tulsiani *et al*. [149] presented an unsupervised learning scheme to address the problem of geometry estimation from a single image through their special ray tracing loss function, which is similar in function to a render layer. The geometry is represented by occupancy probability distribution over voxel grids, and the differentiable ray tracing layer models the interactions between geometry prediction and observation related rays in a probabilistic manner. This differentiable ray tracing layer formulates the training loss by measuring the consistency between observations and its 'rendered' reconstructions. Liu *et al*. [89] simulated the rasterisation and z-buffering operation as a series of probabilistic processes, hence enabling gradient flows from loss function defined in 2D image domain to intrinsics and extrinsics (geometry, camera etc.) defined in 3D space. Kato *et al*. [63] instead leveraged the standard forward rasterisation operation, while replacing the original derivative formulations with approximated ones induced by continuous samplings. More recently Kato *et al*. [62] proposed a survey of differentiable rendering that summarised important works in the field.

2.2 BRDF estimation and modelling

Traditional BRDF estimation and modelling often places much greater restrictions on data capture. Nielsen *et al*. [117] proposed a statistical model of BRDFs, but their BRDF reconstruction requires images under constrained, calibrated capture conditions. Just like Nielsen's work, traditional BRDF estimation methods heavily relies on sampling algorithms and the reconstruction quality is determined by sampling results. In early work, White *et al*. [159] set up a gonioreflectometer gantry to capture the 4D BRDF in a brute force way. Foo [28] optimised the gantry for measuring isotropic BRDFs by dropping one axis of movement. The exhaustive sampling is time consuming and unwieldy, so some other lightweight sampling tricks have been proposed. Ward *et al*. [157] designed an imaging

gonioreflectometer consisting of a fixed fish-eye camera, a movable illuminant and a half-silvered plastic hemisphere, which enable users to capture light reflections in all hemispherical directions at once. Dana *et al*. [18] built a measuring apparatus comprising a fixed lighting source, a camera with limited mobility and a sample plane being available for limited tilting movements. Marschner *et al*. [96] performed sampling on a sphere or curved object to simplify the gonioreflectometer measurement. Following the similar idea, Ngan *et al*. [116] measured anisotropic BRDFs by wrapping the flat object stripes towards varying directions onto the cylindrical template, and they introduced a precision motor for an additional degree of measurement. Ghosh *et al*. [37] proposed to simplify the process of reflectance sampling by using spherical zonal basis function. Ben-Ezra *et al*. [10] used 151 calibrated cameras along with their flash lights to construct their capturing rig. Naik [108] proposed to use scattered light source and light reflections off the target object to capture BRDFs information with only a fixed laser emitter and a fixed camera. Fuchs *et al*. [30] applied adaptive sampling over their setup to overcome the most common aliasing problems. Tunwattanapong *et al*. [151] applied lighting conditions fitted to spherical harmonic patterns through a rotating LED arm, such that they can efficiently estimate diffuse and specular components of the target BRDFs from captured reflection patterns. Very close to the statistical model proposed by Nielsen *et al*. [117], Matusik *et al*. [100] derived similar BRDF model from MERL dataset but with linear PCA method that is less able to capture the distinctive features of a BRDF.

BRDF estimation using CNNs has recently been considered. Kim *et al*. [69] assumed the Ward BRDF model and trained their network to predict the Ward model parameters of objects. One of their proposed networks contains a fixed layer to perform pre-processing on voxel grids which are inputs for the following CNN architecture. Georgoulis *et al*. [36] used their DeLight-Net to estimate Phong model parameters and environment maps given a reflectance map of a captured material and illumination. Rematas *et al*. [125] tried to extract the reflectance map from a single RGB image through their deep network. They proposed to use a fixed sampling layer and domain conversion layer bridging two subnets that work on surface normal prediction and dense reflectance map prediction respectively. Meka *et al*. [103] used a neural network to estimate BRDF represented by the Blinn-Phong model [14] for the captured material from a single image while performing in real time. Through using a cascaded network model, they tackled this problem by decoupling the full task into a sequence

of simpler ones, namely segmentation, specular estimation, albedo estimation and mirror reflection estimation. For training this intricate model, synthetic data with ground truth is required. Li *et al*. [86, 85] also proposed to use a cascaded network architecture to approach the reflectance estimation problem. Instead of only estimating uniform BRDF, their networks tackles a more challenging problem, which decomposes a single input image to SVBRDF, normal and illumination. Likewise, other recent efforts has been made on modelling SVBRDF by CNNs, for example works from Gao *et al*. [32] and Li *et al*. [81]. Here, their models separate BRDF as a diffuse component being represented by Lambertian model and a specular component that is approximated by a phenomenological model or a physically based model. The direct supervision for learning BRDF predictions leads to the necessity of considerable labelled data, which is the key problem their methods attempted to solve.

2.3 Inverse rendering

Inverse rendering tackles the problem of factoring captured scenes into reflectance, geometry and environmental illumination. Instead of considering the full inverse rendering problem, there has been many efforts on optimising the performance of sub-problem, for example depth estimate and intrinsic image decomposition, where geometry and illumination are treated as a joint quantity of shading. In this section, we will review recent work on classical and learning-based inverse rendering approaches, and relevant techniques on learning-based depth prediction and intrinsic image decomposition.

2.3.1 Classical approaches

Classical methods estimate intrinsic properties by fitting photometric or geometric models using optimisation. Most methods require multiple images. From multiview images, a structure-from-motion/multiview stereo pipeline enables recovery of dense mesh models [64, 31] though illumination effects are baked into the texture. From images with fixed viewpoint but varying illumination, photometric stereo can be applied. More sophisticated variants consider statistical BRDF models [6], the use of outdoor time-lapse images [79] and spatially-varying BRDFs [41]. Attempts to combine geometric and photometric methods are limited. Haber *et al*. [45] assumed known geometry (which can

be provided by MVS) and inverse render reflectance and lighting from community photo collections. Kim *et al*. [68] represented the state-of-the-art and again use a MVS initialisation for joint optimisation of geometry, illumination and albedo. Some methods consider a single image setting. Jeson *et al*. [56] introduced a local-adaptive reflectance smoothness constraint for intrinsic image decomposition on texture-free input images which are acquired with a texture separation algorithm. Barron *et al*. [7] presented SIRFS, a classical optimisation-based approach that recovers all of shape, illumination and albedo using a sophisticated combination of generic priors. Chen *et al*. [15] performed intrinsic image decomposition from an RGB-D image whose additional depth channel enabled them to better model shading.

2.3.2 Deep depth prediction

Direct estimation of shape alone using deep neural networks has attracted a lot of attention. Eigen *et al*. [26, 25] were the first to apply deep learning in this context. Subsequently, performance gains were obtained using improved architectures [76], post-processing with classical CRF-based methods [156, 87, 163] and using ordinal relationships for objects within the scenes [29, 84, 16]. Zheng *et al*. [170] used synthetic images for training but improve generalisation using a synthetic-to-real transformatoin GAN. However, all of this work requires supervision by ground truth depth. An alternative branch of methods explores using self-supervision from augmented data. For example, binocular stereo pairs can provide a supervisory signal through consistency of cross projected images [65, 33, 38]. Alternatively, video data can provide a similar source of supervision [173, 154, 155, 39]. Tulsiani *et al*. [149] used multiview supervision in a ray tracing network. While all these methods take single image input, Ji *et al*. [57] tackled the MVS problem itself using deep learning.

2.3.3 Deep intrinsic image decomposition

Intrinsic image decomposition is a partial step towards inverse rendering. It decomposes an image into reflectance (albedo) and shading but does not separate shading into shape and illumination. Even so, the lack of ground truth training data makes this a hard problem to solve with deep learning.

Recent work either uses synthetic training data and supervised learning [111, 46, 80, 13, 27] or self-supervision/unsupervised learning. Very recently, Li *et al*. [83] used uncontrolled time-lapse images allowing them to combine an image reconstruction loss with reflectance consistency between frames. This work was further extended using photorealistic, synthetic training data [82]. Ma *et al*. [94] also trained on time-lapse sequences and introduced a new gradient constraint which encourage better explanations for sharp changes caused by shading or reflectance. Baslamisli *et al*. [8] applied a similar gradient constraint while they used supervised training. Shelhamer *et al*. [134] proposed a hybrid approach where a CNN estimates a depth map which is used to constrain a classical optimisation-based intrinsic image estimation.

2.3.4 Deep inverse rendering

To date, solving the full inverse rendering problem using deep learning has received relatively little attention. One line of work simplifies the problem by restricting to a single object class, e.g. faces [146, 67], meaning that a statistical face model can constrain the geometry and reflectance estimates. This enables entirely self-supervised training. Shu *et al*. [138] extended this idea with an adversarial loss. Sengupta *et al*. [132] on the other hand, initialised with supervised training on synthetic data, and fine-tuned their network in an unsupervised fashion on real images. Nestmeyer *et al*. [114] targeted realistic face relighting, combining an inverse rendering network with a renderer that combines both physics-based and learnt elements. Going beyond faces, Kanamori and Endo [60] considered whole body inverse rendering for relighting. Here, occlusion of the illumination environment becomes important, so they show how to infer a light transport map. Aittala *et al*. [4] restricted geometry to almost planar objects and lighting to a flash in the viewing direction , and under these assumptions they can obtain impressive results. Gao *et al*. [32] considered the same planar scenario but with multiple images, enabling recovery of spatially varying reflectance properties. More general settings have been considered including natural illumination [81]. Philip *et al*. [122] focused on relighting of outdoor scenes but require multiple images and a 3D geometric proxy. Kulkarni *et al*. [73] showed how to learn latent variables that correspond to extrinsic parameters allowing image manipulation. The only prior work we are aware of that tackles the full inverse rendering problem requires direct supervision

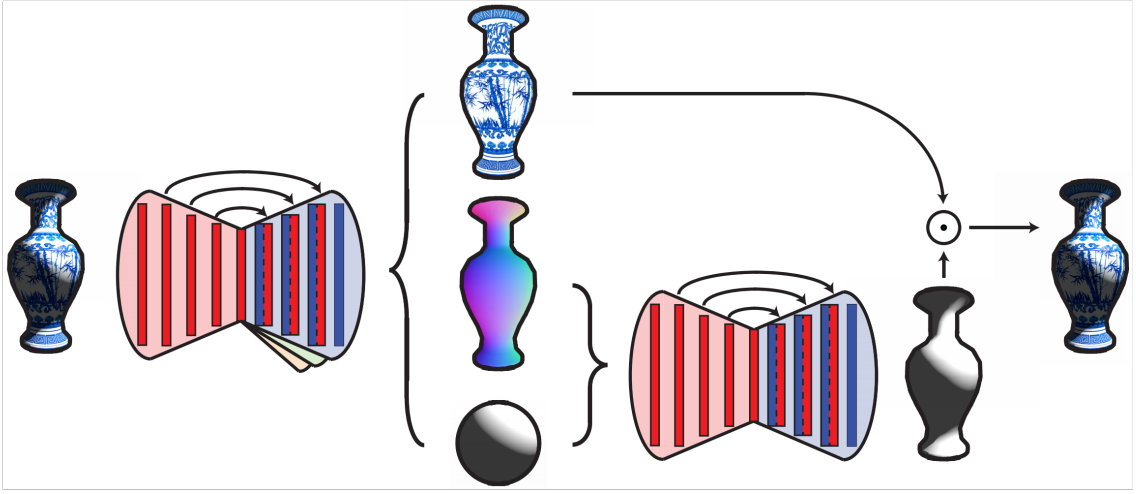


Figure 2.2: Self-supervised inverse rendering network proposed by Janner *et al*. [54]. The first network (on the left) performs inverse rendering on input image, and the second network (on the right) computes shading map from lighting and normal estimate. Finally, a pixel-wise multiplication is performed between shading and reflectance map to form self-reconstruction of the input image. Image taken from [54].

[54, 88, 86]. Hence, it is not applicable to scene-level inverse rendering, only objects, and relies on synthetic data for training, limiting the ability of the network to generalise to real images. Of these object-level inverse rendering networks, the self-supervised inverse rendering work proposed by Janner *et al*. [54] shares similar ideas with us. They proposed to combine predicted reflectance map and shading map in a multiplicative way, and used the outcome of this multiplicative combination to formulate self-reconstruction error. Unlike our model-based shader, they trained a separate neural shader to better capture shading effects. However, their model decomposes lighting by one extra sub-network which is less efficient and stable while requiring more parameters compared with our model. Their model is shown in Figure 2.2. Like discussed above, they would need synthetic data to train the neural shader, and so it only works in object-level settings. Very recently, Sengupta *et al*. [131] combined an inverse rendering network with a trainable, residual rendering network. By pre-training both on synthetic data in a supervised fashion, they can subsequently finetune the inverse rendering network on real data using self-supervision (with the residual rendering network kept fixed). The need for synthetic training data limited their approach to indoor scenes.

2.4 Merging depth and normals

Different shape estimation techniques deliver shape in different representations. For example, photometric methods naturally estimate surface orientation and hence deliver a surface normal map. Stereo methods directly compute scene depth and so deliver a depth map. Moreover, these techniques often have complimentary strengths and weaknesses, for example photometric methods often recover fine-scale detail but contain low frequency bias whereas multiview techniques better capture gross structure but contain high frequency noise. For this reason, there has been interest in techniques that can merge position and surface normal information. Nehab *et al*. [112] proposed an efficient method based on linear least squares that can work with both depth maps and meshes. They also proposed a low pass correction procedure, similar to the work proposed by Zivanov *et al*. [176] who posed the merging process as a nonlinear optimisation problem. These approaches were extended to multiple viewpoints by Berkiten *et al*. [12] who also avoided the linearisation assumptions, though at the cost of an optimisation problem of increased complexity. In deep depth prediction, the idea of separately estimating both depth and normals within the same network has been considered [25]. Here, parts of the network for predicting the two representations are shared but the geometric relationship between them is never explicitly enforced.

2.5 Image relighting

Relighting a scene is a complex task. In order to perform physically accurate relighting, all components of light-transport in the scene need to be measured and modified, in a process known as inverse rendering [120]. Traditionally, this involved using special optical equipment to measure the geometry [166, 92, 101], surface reflectance [20, 158, 98, 148] and environmental illumination [19, 51, 78, 141], while also inverting the global illumination within the scene [165]. Image-based relighting techniques have attempted to simplify the problem by using only 2D images for the task. But using only 2D images makes the problem highly under-constrained and ambiguous.

Due to the ambiguous nature of the problem, recently there has been a lot of interest in applying

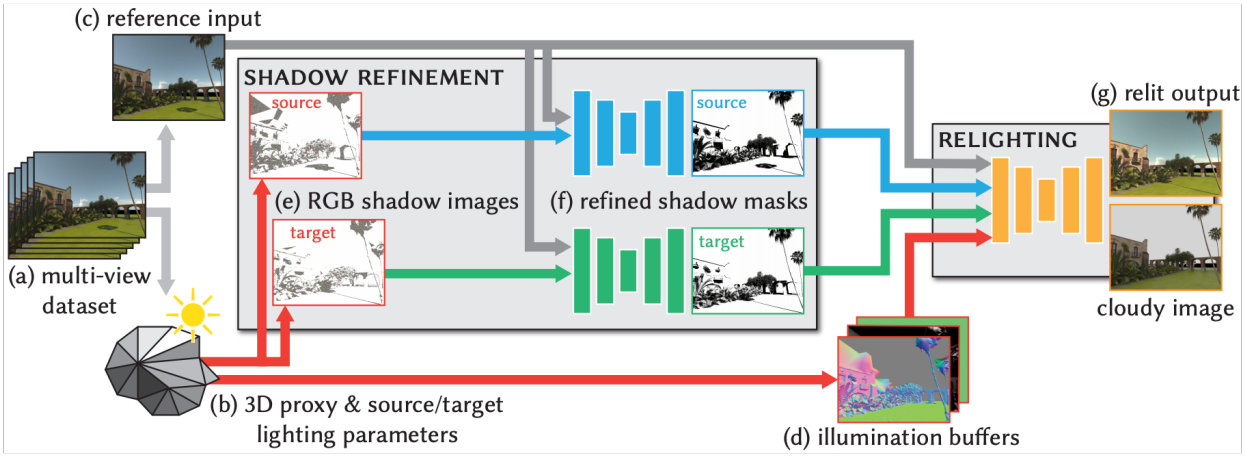


Figure 2.3: Outdoor scene relighting network proposed by Philip *et al*. [122]. Image taken from [122].

learning-based methods to solving it [172, 138, 34, 110]. We restrict our discussion to methods that perform scene level relighting. Relighting of indoor scenes has received greater attention due to the relatively easier acquisition of geometry and illumination data [101]. Due to the very different nature of geometry and illumination in indoor and outdoor scenes, the two have often been treated as separate classes of inverse rendering problems. Inverse rendering in outdoor scenes has usually dealt with specific illumination models for natural illumination. [162] proposed a single-image approach that accounts for environment lighting in outdoor scenes. Collections of photographs of a scene have been used to provide better constraints for relighting [45, 133]. While we also use a dataset of casual photography of particular scenes to learn to perform inverse rendering and relighting, at test time, we only rely on a single image of a scene to perform photorealistic relighting. The method of Shih *et al*. [136] performs lighting transfer by matching a single image to a large database of timelapses but cannot treat cast shadows. Alternatively, online digital terrain and urban models registered to images can be used for approximate relighting [71].

Several methods for multi-view image relighting have been developed, both for the case of multiple images sharing single lighting conditions [24], and for images of the same location with multiple lighting conditions (typically from internet photo collections) [75]. For the single lighting condition, Duchêne *et al*. [24], first performed shadow classification and intrinsic decomposition using separate optimisation steps. Despite impressive results, artifacts remain especially around shadow boundaries and the relighting method fails beyond limited shadow motion.

More recently, several learning-based methods have been suggested to perform relighting for indoor

and outdoor scenarios [164, 122, 131]. A deep relighting neural network for object-level images was proposed by Xu *et al*. [164]. The proposed model is driven by the idea from traditional image-based relighting method, and the model only requires 5 input images to produce realistic relighting output by using the powerful deep learning models. In this work, they proposed a relighting rendering network, and an automatically way to learn a simple yet effective Sample-Net to define the sparse set of input sample directions, which can drive the optimal performance of rendering network. Philip *et al*. [122] proposed a cascaded network model to attack the relighting problem for outdoor scenes, and it is shown in Figure 2.3. To generate realistic and sharp shadows, their relighting process necessitates the accessibility of the 3D geometry information, based on which their shadow sub-networks can accurately estimate the shadow presented in input and the shadow under novel lighting condition. Their intricate shadow handling method involves a mechanism referred to as RGB shadow image, which is a colour shadow map capturing extract information regarding the surrounding 3D geometry. Instead of greyscale pixel value, the RGB value at each shadow pixel is acquired by tracing back from the pixel to the source of the shadow which is the 3D object occluding the pixel from illuminations. With this new type of shadow map, their model is able to produce high-quality shadow estimates, leading to relighting results with realistic shadowing effects. Like our neural relighting network proposed in Chapter 5, their network is also designed for outdoor scenes. But our method is not limited to training on labelled synthetic data and performing inference based on known 3D geometry. Sengupta *et al*. [131] attempted to tackle monocular inverse rendering problem for indoor scenes, which could also be used for relighting purpose. The core idea of their work is to combine a Residual Appearance Renderer (RAR) capturing the non-Lambertian effects, and an Inverse Rendering Network (IRN) decomposing the image based on Lambertian model, to better model the input indoor image under Lambertian assumption. Then by fixing the RAR model being pre-trained on synthetic images, their IRN is finetuned on real images to achieve improved performance for real scenes. The common limitation for these works is heavily requiring synthetic data in the training, which could lead to generalisation issue.

Some relighting techniques have focused on specific types of objects such as human faces [121, 137, 142, 102, 172] and bodies [60]. Peers *et al*. [121] required two images of a training subject, one shot at the illumination of the test image, and another shot at the target illumination. The lightness

component is then estimated through a quotient image, which is the ratio between both exemplars. The quotient is transfered to the test image to relight it. Shu *et al*. [137] proposed a mass-transport formulation to generate a localized, geometry-aware relighting. For each pixel, an 8-dimensional vector is estimated containing the colour, 2D position, and 3D normal. Relighting is achieved through histogram matching between the source and example in this augmented space. Meka *et al*. [102] relighted faces in a light stage set-up using two images shot under spherical colour gradient illuminations. Results show that such simplified illumination condition is capable to estimate the full 4D reflectance field, including high frequency details and specular reflection. Sun *et al*. [142] presented a neural network for relighting faces taken by a cellphone camera. The network is trained on light-stage data of 18 subjects captured under several directional light sources. Kanamori and Endo [60] presented a human body relighting technique with good handling of occluded regions e.g. crotches, garment wrinkles, armpits and so on. A neural network infers the light transport map which encodes occlusions in form of Spherical Harmonics coefficients per pixel. The network is trained using data synthesized from scanned 3D human figures. Results show better relighting than occlusion-free formulation methods. A simpler version of the relighting problem, is of integrating virtual objects into real scenes in an illumination-consistent manner, have been solved by using proxy geometry and user interaction [61, 162, 118, 103]. But these methods do not solve the problem of general relighting of scenes. Webcam sequences have also been used for relighting [77, 143], although cast shadows often require manual layering.

2.6 Conclusion

In this chapter, we have reviewed the related work on differentiable and neural rendering, BRDF modelling, inverse rendering and its partial problems, depth and normal merging algorithm, and image relighting techniques. Differentiable renderers as a typical class of model-based decoders have been used to produce self-supervision and physical constraint on network outputs. However, such differentiable renderers have downsides in terms of the photorealism and natural shading generation, which is caused by the differentiable approximations to the underlying physics model and by the potential problem in the involved physics-based model itself. For example, Lambertian model neglects

the specular reflections, and Blinn-Phong cannot handle shadowing and masking. This leads our following research into two directions: using differentiable renderer to make network training amenable to self-supervised settings and fusing a neural decoder with a model-based decoder to achieve improved performance.

Deep learning related work has made breakthroughs in these reviewed fields over classical approaches, while it remains challenging to obtain sufficient ground truth data to supervise the network training. In particular, BRDF modelling networks heavily rely on synthetic data in training, and inverse rendering problem is often tackled by either using synthetic data or being formulated to a simplified scenario. Following the idea of self-supervised training being introduced by model-based decoder, the limitation of reliance on ground truth data can be lifted.

As for merging depth and normal by model-based decoder, it commonly assumes the accessibility to coarsely plausible depth map and finely detailed normal map, both of which can be faithfully estimated by specific deep networks. Without hybridising them, either neural networks or model-based decoders alone cannot hallucinate good quality depth refinement results. When considering the image relighting problem, hybridising model-based decoder and neural decoder again outperforms using solely one instance. Because the existing deep image relighting frameworks tend to implicitly model input image decomposition and relighting image composition processes, which neglects the physical interpretability and so is unable to control relighting by physically meaningful parameters, like the novel lighting condition. Embedding model-based decoders makes such image relighting network open to such physically explicit controls. Thus, the performance of image relighting network and deep depth refinement will benefit from taking the hybrid model of model-based and neural decoder.

Chapter 3

Model-based decoder

3.1 Introduction

Model-based decoder inside a neural network is an untrainable module/layer, which acts as a physics-based model taking as input the predictions given by the neural network and computes physically meaningful outputs. For instance, the environment illumination in an input image can be predicted by a Lambertian reflectance layer, which assumes a linear relationship of diffuse albedo, normal map, illumination being represented by spherical harmonics coefficients and a colour image. The Lambertian layer integrated in an inverse rendering network can take as inputs the normal and albedo predictions and the input colour image, and linearly compute the best fitting result in a closed form for the spherical harmonic lighting. Note that this is an untrainable layer, because the Lambertian layer contains no trainable parameters and is not updated to reduce the loss during the training process. Acting as a lighting decomposition decoder, the Lambertian model can simulate the image formation process so as to produce physically legitimate illumination prediction that is also compatible with the given input, normal and albedo.

In this chapter, we firstly discuss how to implement operations often used in photometric vision as part of a neural network. The library of such photometric vision layers is called PVNN and developed as a Theano [11] toolbox. We see this toolbox as a photometric analog of the geometric functionality provided by the Geometric Vision with Neural Networks (gvnn) toolbox [47]. In a forward pass,

combinations of our layers can act as a differentiable, physics-based renderer. In an encoder-decoder architecture as described above, PVNN can be used to train CNNs to solve physics-based vision problems in an unsupervised manner. In Section 3.2 we describe the layers that make up PVNN. In Section 3.4.1, we show how these layers can be used for differentiable forward rendering. In Section 3.4.2, we demonstrate an exemplar application by using unsupervised learning to train a CNN to predict BRDF parameters from single images.

In the second part of this chapter, we will show another differentiable renderer based on Lambertian reflectance model, which is much more simplified and flexible compared to the first differentiable renderer. The model renders images by linearly combining predictions from the encoder. Compared with PVNN, this linear renderer works more efficiently during training. At the same time, the linear model can be used as a lighting inference layer allowing lighting predictions to be directly solvable within this module, which is significantly more efficient as opposed to predicting the illuminations by training a separate neural decoder. In Section 3.3, we describe both types of decoders. In Section 3.4.3, we present experiments demonstrating the renderer and the lighting inference module, which we refer to as lighting inference decoder because this module decodes image encoding/decoding results like albedo and normal estimation into lighting estimation.

3.2 PVNN: differentiable layers for BRDF evaluation

In this section, we will define the implementations of differentiable layers for photometric vision models included in PVNN. The network layers are presented by three categories: geometric transformation layers, statistical BRDF modelling layers and reflectance evaluation layers. In this section, we provide the gradient computation for each operation defined in geometric transformations to demonstrate their differentiabilities. However, the gradient computations are not included in the rest of the thesis, as we are working on the standard deep learning frameworks, e.g. Theano [11] and TensorFlow [2], which can automate gradient computations during backpropagation.

3.2.1 Geometric transformations

Most commonly, photometric vision is concerned with single viewpoint problems and so usually operates in the depth or surface normal domains (as opposed to an object-space mesh representation). We use such a viewer-centred depth map representation in PVNN. One advantage of this formulation is that we need not model self-occlusions of the surface. Occlusion is a binary, and hence discontinuous, function. This means that it is not differentiable and therefore occlusion information cannot be exploited for learning during backpropagation.

We assume here orthographic projection, though it would be straightforward to modify our layers to account for perspective projection (taking into account the intrinsic parameters of the camera).

Surface height differentiation

This layer transforms a depth map into estimates of the surface gradient in the horizontal and vertical directions. It takes as input a depth map, $\mathbf{Z} \in \mathbb{R}^{m \times n}$, that is assumed to be smooth containing the depth values for $m \times n$ pixels:

$$\mathbf{G}_x = \mathbf{D}_x * \mathbf{Z}, \quad \frac{\partial \mathbf{G}_x^{i,j}}{\partial \mathbf{Z}_{u,v}} = \begin{cases} \mathbf{D}_x^{u-i+1,v-j+1}, & \text{if } 1 \leq u-i+1 \leq 3 \text{ and } 1 \leq v-j+1 \leq 3 \\ 0, & \text{otherwise} \end{cases}, \tag{3.1}$$

$$\mathbf{G}_x = \mathbf{D}_x * \mathbf{Z}, \quad \frac{\partial \mathbf{G}_y^{i,j}}{\partial \mathbf{Z}_{u,v}} = \begin{cases} \mathbf{D}_y^{u-i+1,v-j+1}, & \text{if } 1 \leq u-i+1 \leq 3 \text{ and } 1 \leq v-j+1 \leq 3 \\ 0, & \text{otherwise} \end{cases}, \tag{3.2}$$

where $*$ is the matrix convolution. $\mathbf{G}_x$ and $\mathbf{G}_y$ are surface gradients in horizontal and vertical directions. $\mathbf{D}_x \in \mathbb{R}^{3\times3}$ and $\mathbf{D}_y \in \mathbb{R}^{3\times3}$ evaluate the surface gradient in the horizontal and vertical directions, respectively, using forward finite differences and convolution:

$$\mathbf{D}_x = \frac{1}{12}\begin{bmatrix} -1 & 0 & 1 \\ -4 & 0 & 4 \\ -1 & 0 & 1 \end{bmatrix}, \mathbf{D}_y = \frac{1}{12}\begin{bmatrix} 1 & 4 & 1 \\ 0 & 0 & 0 \\ -1 & -4 & -1 \end{bmatrix}. \tag{3.3}$$

Gradient to normal vector

This layers applies the function $\mathbf{n} : \mathbb{R}^2 \mapsto \mathbb{R}^3$ which transforms the gradient vector into a vector whose direction is normal to the surface. Here the gradient vector $\mathbf{g}$ is the concatenated gradients in x and y directions for one pixel:

$$\mathbf{g}_{i,j} = \begin{bmatrix} \mathbf{D}_x^{i,j} & \mathbf{D}_y^{i,j} \end{bmatrix}^T. \tag{3.4}$$

The subsequent differentiable layers are defined as pixel-wise operations. For the sake of simplicity, we will omit subscript i and j from now on. The function and its first derivatives (Jacobian) are given by:

$$\bar{\mathbf{n}}(\mathbf{g}) = \begin{bmatrix} -\mathbf{g}^T & 1 \end{bmatrix}^T, \quad \mathbf{J}_\mathbf{n}(\mathbf{g}) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix}. \tag{3.5}$$

This function is applied to the gradient estimate at each pixel, yielding u surface normal vectors, where u is length of flatten vector from $m \times n$ map.

Vector normalisation

It is often convenient to work with surface normal vectors of unit length. So, this layer applies the function $\bar{\mathbf{n}} : \mathbb{R}^3 \mapsto \mathbb{R}^3$ which normalises a vector to have unit length such that $\|n\| = 1$. This function is simply:

$$n(\bar{\mathbf{n}}) = \bar{\mathbf{n}}/\|\bar{\mathbf{n}}\| = [n_1\, n_2\, n_3]^T, \quad \mathbf{J}_n(\bar{\mathbf{n}}) = \frac{\mathbf{I}}{\|\bar{\mathbf{n}}\|} - \frac{\bar{\mathbf{n}}\bar{\mathbf{n}}^T}{\|\bar{\mathbf{n}}\|^3}. \tag{3.6}$$

Unit vector to spherical coordinates

Sometimes, it is useful to transform the surface normal vector $\bar{\mathbf{n}}$ into spherical coordinates (ϕ, θ) in a viewer-centred coordinate system. The azimuth angle is computed by the function $\phi : \mathbb{R}^3 \mapsto [0, 2\pi)$:

$$\phi(n(\bar{\mathbf{n}})) = \text{atan2}(n_2, n_1), \quad \nabla\phi(n(\bar{\mathbf{n}})) = \begin{bmatrix} \frac{-n_2}{n_1^2+n_2^2} & \frac{n_1}{n_1^2+n_2^2} & 0 \end{bmatrix}^T. \tag{3.7}$$

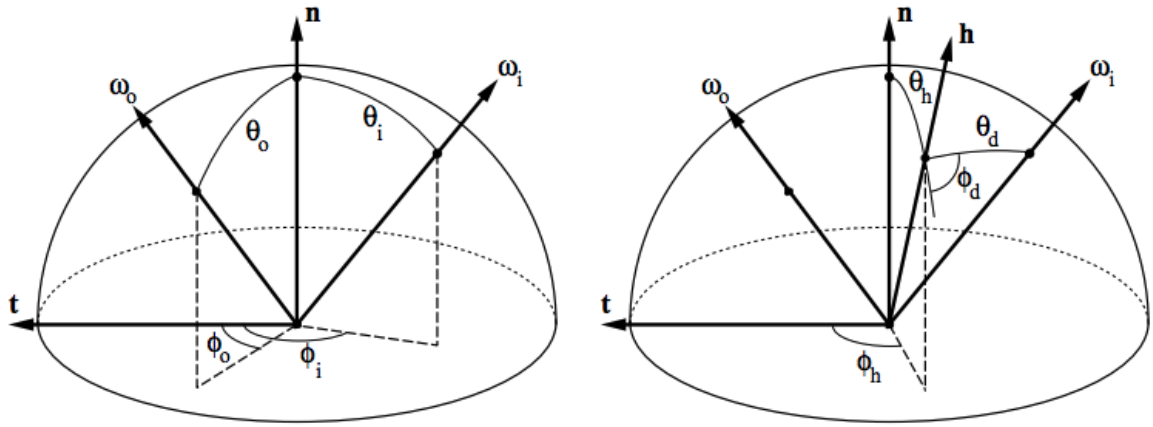


Figure 3.1: Comparison between Spherical coordinate system and Rusinkiewicz coordinate system [128]. Halfway vector $\mathbf{h}$ defined by (θ_h, ϕ_h) and difference vector $\mathbf{d}$ defined by (θ_d, ϕ_d) represents incident and viewing angles. Image taken from [128].

The zenith angle is computed by the function $\theta : \mathbb{R}^3 \mapsto [0, \pi]$:

$$\theta(n(\bar{\mathbf{n}})) = \arccos(n_3), \quad \nabla\theta(n(\bar{\mathbf{n}})) = \begin{bmatrix} 0 & 0 & \frac{-1}{\sqrt{1-n_3^2}} \end{bmatrix}^T. \tag{3.8}$$

Conversion to Rusinkiewicz coordinates

We assume that the BRDFs are isotropic and hence can be expressed as three dimensional functions. For convenience and compatibility with the statistical BRDF model used later, we parameterise BRDFs in terms of the three angles proposed by Rusinkiewicz [128]. Concretely, the Rusinkiewicz coordinates parameterise the local reflectance geometry in terms of three angles relative to the halfway vector:

$$\mathbf{h}(\mathbf{s}, \mathbf{v}) = n(\mathbf{s} + \mathbf{v}), \tag{3.9}$$

where $\mathbf{s}$ and $\mathbf{v}$ are unit vectors in the light source and viewer directions respectively. The Rusinkiewicz coordinates system is shown in Figure 3.1. In Rusinkiewicz coordinates, incident and viewing rays are defined by halfway vector $\mathbf{h}$ and difference vector $\mathbf{d}$. The angles $\theta_h(\mathbf{n}, \mathbf{s}, \mathbf{v}) \in [0, \pi/2]$ and $\phi_h(\mathbf{n}, \mathbf{s}, \mathbf{v}) \in [0, \pi]$ are the angles rotating $\mathbf{h}$ to the surface normal $\mathbf{n}$ while $\theta_d(\mathbf{n}, \mathbf{s}, \mathbf{v}) \in [0, \pi/2]$ and $\phi_d(\mathbf{n}, \mathbf{s}, \mathbf{v}) \in [0, \pi]$ are the spherical coordinates of $\mathbf{s}$ in a coordinate system in which $\mathbf{h}$ is at the north pole. Using Rusinkiewicz coordinates allow us to parameterise isotropic BRDFs by only three angles $\{\theta_h(\mathbf{n}, \mathbf{s}, \mathbf{v}), \theta_d(\mathbf{n}, \mathbf{s}, \mathbf{v}), \phi_d(\mathbf{n}, \mathbf{s}, \mathbf{v})\}$.

Hence, we define a layer that takes as input the per-pixel surface normals $\mathbf{N} \in \mathbb{R}^{u\times 3}$ for u pixels, light directions $\mathbf{S} \in \mathbb{R}^{s\times 3}$ for s light sources and per-pixel viewer directions $\mathbf{V} \in \mathbb{R}^{u\times 3}$ which are all $[0\ 0\ 1]$ for an orthographic projection. The output is a tensor $\mathcal{A} \in \mathbb{R}^{u\times s\times 3}$ containing the three reflectance angles for each pixel/light source combination, such that $\mathcal{A}_{ij1} = \theta_h(\mathbf{n}_i, \mathbf{s}_j, \mathbf{v})$, $\mathcal{A}_{ij2} = \theta_d(\mathbf{n}_i, \mathbf{s}_j, \mathbf{v})$ and $\mathcal{A}_{ij3} = \phi_d(\mathbf{n}_i, \mathbf{s}_j, \mathbf{v})$.

The Rusinkiewicz conversion layer essentially performs two rotations on coordinates system, which firstly rotates from camera coordinates to surface normal centred coordinates followed by the rotation transferring halfway vector $\mathbf{h}$ to the north pole of coordinates. The rotations from initial coordinates to target coordinates is defined by:

$$
\begin{aligned}
\text{rot}(\sigma_{\mathbf{init}}, \sigma_{\mathbf{tar}}) &= \begin{bmatrix} \cos(\phi(\sigma_{\mathbf{tar}})) & 0 & \sin(\phi(\sigma_{\mathbf{tar}})) \\ 0 & 1 & 0 \\ -\sin(\phi(\sigma_{\mathbf{tar}})) & 0 & \cos(\phi(\sigma_{\mathbf{tar}})) \end{bmatrix} \begin{bmatrix} \cos(\theta(\sigma_{\mathbf{tar}})) & -\sin(\theta(\sigma_{\mathbf{tar}})) & 0 \\ \sin(\theta(\sigma_{\mathbf{tar}})) & \cos(\theta(\sigma_{\mathbf{tar}})) & 0 \\ 0 & 0 & 1 \end{bmatrix} \sigma_{\mathbf{init}}, \\
\frac{\partial \text{rot}(\sigma_{\mathbf{init}}, \sigma_{\mathbf{tar}})}{\partial \sigma_{\mathbf{init}}} &= \begin{bmatrix} \cos(\phi(\sigma_{\mathbf{tar}})) & 0 & \sin(\phi(\sigma_{\mathbf{tar}})) \\ 0 & 1 & 0 \\ -\sin(\phi(\sigma_{\mathbf{tar}})) & 0 & \cos(\phi(\sigma_{\mathbf{tar}})) \end{bmatrix} \begin{bmatrix} \cos(\theta(\sigma_{\mathbf{tar}})) & -\sin(\theta(\sigma_{\mathbf{tar}})) & 0 \\ \sin(\theta(\sigma_{\mathbf{tar}})) & \cos(\theta(\sigma_{\mathbf{tar}})) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \\
\frac{\partial \text{rot}(\sigma_{\mathbf{init}}, \sigma_{\mathbf{tar}})}{\partial \theta(\sigma_{\mathbf{tar}})} &= \begin{bmatrix} -\cos(\phi(\sigma_{\mathbf{tar}})) * \sin(\theta(\sigma_{\mathbf{tar}})) & -\cos(\phi(\sigma_{\mathbf{tar}})) * \cos(\theta(\sigma_{\mathbf{tar}})) & 0 \\ \cos(\theta(\sigma_{\mathbf{tar}})) & -\sin(\theta(\sigma_{\mathbf{tar}})) & 0 \\ \sin(\phi(\sigma_{\mathbf{tar}})) * \sin(\theta(\sigma_{\mathbf{tar}})) & \sin(\phi(\sigma_{\mathbf{tar}})) * \cos(\theta(\sigma_{\mathbf{tar}})) & 0 \end{bmatrix} \sigma_{\mathbf{init}}, \\
\frac{\partial \text{rot}(\sigma_{\mathbf{init}}, \sigma_{\mathbf{tar}})}{\partial \phi(\sigma_{\mathbf{tar}})} &= \begin{bmatrix} -\sin(\phi(\sigma_{\mathbf{tar}})) * \cos(\theta(\sigma_{\mathbf{tar}})) & \sin(\phi(\sigma_{\mathbf{tar}})) * \sin(\theta(\sigma_{\mathbf{tar}})) & \cos(\phi(\sigma_{\mathbf{tar}})) \\ 0 & 0 & 0 \\ -\cos(\phi(\sigma_{\mathbf{tar}})) * \cos(\theta(\sigma_{\mathbf{tar}})) & \cos(\phi(\sigma_{\mathbf{tar}})) * \sin(\theta(\sigma_{\mathbf{tar}})) & -\sin(\phi(\sigma_{\mathbf{tar}})) \end{bmatrix} \sigma_{\mathbf{init}},
\end{aligned}
\tag{3.10}
$$

where $\sigma_{\mathbf{init}}$ is the vector to be rotating and $\sigma_{\mathbf{tar}}$ is the north pole of the target coordinates. Therefore, two keys vectors $\mathbf{h}$ and $\mathbf{d}$ for each pixel are computed by:

$$
\mathbf{h} = n(\text{rot}(\mathbf{s},\ \mathbf{n}) + \text{rot}(\mathbf{v},\ \mathbf{n})), \tag{3.11}
$$

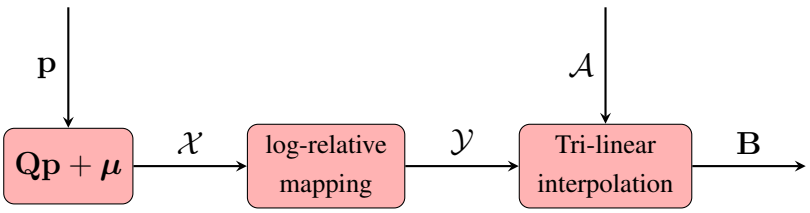


Figure 3.2: A statistical BRDF model sub-network. This network can be plugged in place of a parametric BRDF model in a reflectance evaluation network (i.e. in place of the "BRDF" node in Figure 3.3). It takes statistical BRDF parameters $\mathbf{p}$ and per-pixel Rusinkiewicz coordinates $\mathcal{A}$ as input and outputs BRDF values $\mathbf{B}$.

$$\mathbf{d} = \text{rot}(\text{rot}(\mathbf{s},\, \mathbf{n}),\, \mathbf{h}). \tag{3.12}$$

3.2.2 Statistical BRDFs

Parametric BRDF models are popular and widely used in graphics and vision. However, no single parametric model is capable of generalising to the wide range of reflectance properties observed in the real world. For this reason, statistical [117] or dictionary-based [52] models built from measured data are becoming increasingly popular.

BRDF reconstruction

As an alternative to parametric BRDFs, PVNN also supports statistical BRDF models; specifically, the model of Nielsen *et al*. [117] that is trained by running Principal Components Analysis (PCA) [160] on the MERL reflectance database [99]. This is implemented as a subnetwork (see Figure 3.2) that evaluates the statistical model to generate an empirical BRDF, inverts the log-relative mapping used in the Nielsen [117] model and finally performs differentiable interpolation into the empirical BRDF.

The statistical BRDF model layer evaluates a linear statistical model of empirical BRDFs. An empirical BRDF $\mathcal{X} \in \mathbb{R}^{90\times90\times180}$ is a representation of a discretely measured BRDF at $k = 90 \times 90 \times 180$ samples over $(\theta_h, \theta_d, \phi_d)$ space. We compute a vectorised representation $\mathbf{x} = \text{vec}(\mathcal{X}) \in \mathbb{R}^k$ as:

$$\mathbf{x} = \mathbf{Qp} + \boldsymbol{\mu}, \tag{3.13}$$

where $\mathbf{Q} \in \mathbb{R}^{k \times p}$ contains the p principal component vectors, $\boldsymbol{\mu} \in \mathbb{R}^k$ the mean and $\mathbf{p} \in \mathbb{R}^p$ is a vector of weights, which by appropriate scaling of the principal components, has elements that follow the standard normal distribution. The layer takes $\mathbf{p}$ as input and outputs $\mathcal{X}$. Note that this layer is exactly equivalent to a fully connected layer in which $\mathbf{Q}$ plays the role of the weights and $\boldsymbol{\mu}$ the biases. Hence, it would be straightforward to make this layer trainable and, in principle, to learn statistical BRDF models as part of the network training. We do not do this here though and keep the model fixed to that of Nielsen *et al*. [117].

Note that the statistical model of Nielsen *et al*. [117] contains some missing values. In particular, they excluded missing values and vectorised BRDFs from well-measured directions before computing statistical model, so their BRDF model is a vector with only 1,105,588 entries out of fully defined size of 1,458,000 ($180 \times 90 \times 90$) directions. For simplicity, we interpolated these missing values by their nearest neighbours. We precompute the relative relationship between missing values and known values, and directly perform mapping through precomputed mapping look-up table. Although a more sophisticated missing data scheme could be used, we notice no visual artefacts through the use of nearest neighbour.

Log-relative mapping

This statistical model is constructed from pre-processed data, which is mapped from raw BRDFs by taking the logarithm of the ratio between the raw BRDFs and a reference BRDF. This mapping is used to address the poor performance of PCA when it is directly applied to raw BRDFs which are distributed in a high dynamic range, and it can be computed by:

$$\rho_{\text{map}} = \ln\left(\frac{\rho \ \cos_{\text{weight}} + \epsilon}{\rho_{\text{ref}} \ \cos_{\text{weight}} + \epsilon}\right), \tag{3.14}$$

where

$$\cos_{\text{weight}} = \max\left[\cos(\mathbf{n}\cdot\mathbf{s})\cos(\mathbf{n}\cdot\mathbf{v}), \epsilon\right], \tag{3.15}$$

is a weight applied to compensate for extreme grazing-angle values, ρ_{ref} is a reference BRDF (set as the median BRDF value for each sampled angles) and ϵ is the numerical stabilisation term, set equal

to 10^{-3}.

In our statistical BRDF network, we reconstruct a discretely sampled BRDF $\mathcal{X}$ and then invert (3.14) using:

$$\mathcal{Y} = (\exp(\mathcal{X}) \odot (\mathcal{X}_{\text{ref}} \odot \cos_{\text{weight}} + \epsilon) - \epsilon) \oslash \cos_{\text{weight}}, \tag{3.16}$$

where $\odot$ is the Hadamard (element-wise) product and the division is also applied element-wise.

BRDF value sampling

Given an empirical BRDF, $\mathcal{Y}$, with the log-relative mapping inverted and per-pixel/light source Rusinkiewicz 3D coordinates, $\mathcal{A}$, we interpolate an exact BRDF value by tri-linear interpolation. The purpose of this step is to find appropriate BRDF values from a discrete BRDF lookup table when the indices $(\theta_h, \theta_d, \phi_d)$ are continuous. Differentiable tri-linear interpolation can be performed using:

$$\rho(\theta_h, \theta_d, \phi_d) = \sum_{i=1}^{90}\sum_{j=1}^{90}\sum_{k=1}^{180} \mathcal{Y}_{ijk} \times \max(0, 1-|\theta_h \frac{180}{\pi} - i|) \times \max(0, 1-|\theta_d \frac{180}{\pi} - j|) \times \max(0, 1-|\phi_d \frac{180}{\pi} - k|). \tag{3.17}$$

Here, $(\theta_h, \theta_d, \phi_d)$ are forced to fall in the correct angular range using:

$$\begin{aligned} \theta_h &= \min(\max(0, \theta_h), \frac{\pi}{2}), \\ \theta_d &= \min(\max(0, \theta_d), \frac{\pi}{2}), \\ \phi_d &= \min(\max(0, \phi_d), \pi). \end{aligned} \tag{3.18}$$

We perform such a look up for the Rusinkiewicz coordinates for each pixel/light source and output a matrix of BRDF values, $\mathbf{B}$.

3.2.3 Reflectance evaluation

A reflectance evaluation network is a sub-network that computes the radiance reflected towards the viewer, for a given BRDF, surface normal direction ($\mathbf{n}$), viewer direction ($\mathbf{v}$) and lighting environment. A BRDF is a function $\rho(\theta_h, \theta_d, \phi_d)$ that returns the ratio of reflected radiance to the inci-

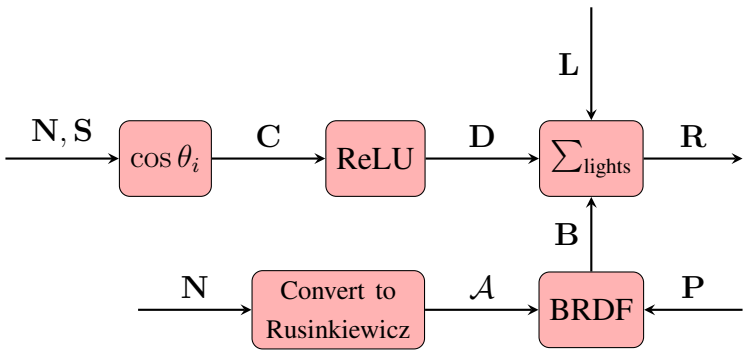


Figure 3.3: A reflectance evaluation sub-network. It takes normals $\mathbf{N}$, light source directions $\mathbf{S}$, light source colours $\mathbf{L}$ and BRDF parameters $\mathbf{P}$ as input and outputs radiance values per-pixel $\mathbf{R}$.

dent irradiance for a pair of incoming and outgoing directions. We write $\rho(\mathbf{n}, \mathbf{s}, \mathbf{v})$ as shorthand for $\rho(\theta_h(\mathbf{n}, \mathbf{s}, \mathbf{v}), \theta_d(\mathbf{n}, \mathbf{s}, \mathbf{v}), \phi_d(\mathbf{n}, \mathbf{s}, \mathbf{v}))$, where $\mathbf{s}$ is the incident radiance direction.

For a continuous lighting environment, the reflected radiance, L_o, is given by Kajiya [59]:

$$L_o = \int_{S^2} \rho(\mathbf{n}, \boldsymbol{\omega}_i, \mathbf{v}) L(\boldsymbol{\omega}_i) \max(0, \mathbf{n} \cdot \boldsymbol{\omega}_i) d\boldsymbol{\omega}_i, \tag{3.19}$$

where $L(\boldsymbol{\omega}_i)$ is the incident radiance from direction $\boldsymbol{\omega}_i$. In practice, we discretise over s point light sources and evaluate:

$$\mathbf{r} = \sum_{i=1}^{s} \rho(\mathbf{n}, \mathbf{s}_i, \mathbf{v}) \mathbf{l}_i \max(0, \mathbf{n} \cdot \mathbf{s}_i), \tag{3.20}$$

where $\mathbf{r} \in \mathbb{R}^3$ is the RGB colour radiance vector, $\mathbf{l}_i \in \mathbb{R}^3$ the colour of the ith light source and $\mathbf{s}_i \in \mathbb{R}^3$, $\|\mathbf{s}_i\| = 1$ the direction of the ith light source.

The individual layers that we use to implement a reflectance evaluation are shown in Figure 3.3. We describe each of these layers in the following sections.

Cosine weight layer

This layer computes cosine weights, $\cos(\theta_i) = \mathbf{n} \cdot \mathbf{s}$, for each pixel:

$$\mathbf{C} = \mathbf{N}\mathbf{S}^T, \tag{3.21}$$

where $\mathbf{N} \in \mathbb{R}^{u \times 3}$ contains the surface normal vectors for u pixels and $\mathbf{S} \in \mathbb{R}^{s \times 3}$ contains the lighting direction vectors for s light sources.

Clamping layer

This layer implements the max operator in (3.20) which simulates the effect of self-shadows (i.e. incident angle greater than $90°$). It takes as input the output of the cosine layer and computes:

$$\mathbf{D} = \max(0, \mathbf{C}). \tag{3.22}$$

Note that this is exactly a Rectified Linear Unit (ReLU) layer and so can be implemented using a standard ReLU layer implementation.

Colour scale and sum over light sources

This layer takes as input the clamped-cosine and BRDF values, element-wise multiplies them, uses these to scale the light source radiances and finally sums over light sources:

$$\mathbf{R}_i = \sum_{j=1}^{s} \mathbf{B}_{ij} \mathbf{D}_{ij} \mathbf{L}_j. \tag{3.23}$$

This provides the final output of the reflectance evaluation network: per-pixel reflected radiance values.

3.3 Lambertian diffuse model

A model-based decoder can act as an image renderer that takes the same inputs as a conventional renderer (which are perhaps themselves the output of a neural encoder). The model-based renderer is useful for self-supervision in image decomposition. Unlike the image rendering method introduced in Section 3.2.3, we will discuss a simplified renderer that takes more flexible inputs and consumes less time while giving good quality results on low-resolution and outdoor images. While simple and

efficient, the renderer can only work in screen space domain, which means that inputs are buffers like normal and texture.

3.3.1 Image rendering

We assume that appearance can be approximated by a local reflectance model under environment illumination. Specifically, we use a Lambertian diffuse model with order 2 spherical harmonic lighting [124]. This means that RGB intensity can be computed as

$$\mathbf{i}_{\text{lin}}(\mathbf{n}, \boldsymbol{\alpha}, \mathbf{L}) = \text{diag}(\boldsymbol{\alpha})\mathbf{L}\mathbf{b}(\mathbf{n}), \tag{3.24}$$

where $\mathbf{L} \in \mathbb{R}^{3\times 9}$ contains the spherical harmonic colour illumination coefficients, $\boldsymbol{\alpha} = [\alpha_r, \alpha_g, \alpha_b]^T$ is the colour diffuse albedo and the order 2 basis is given by:

$$\mathbf{b}(\mathbf{n}) = [1, n_x, n_y, n_z, 3n_z^2 - 1, n_x n_y, n_x n_z, n_y n_z, n_x^2 - n_y^2]^T. \tag{3.25}$$

Our appearance model means that we neglect high frequency illumination effects, cast shadows, interreflections, and it only models matte materials that diffusely reflect incident lights but fails on mirrored or dichromatic materials. However, we found that in practice this model works well for typical outdoor scenes. Finally, cameras apply a nonlinear gamma transformation to the rendered images out of this renderer. We simulate this to produce our final predicted intensities:

$$\mathbf{i}_{\text{pred}} = \mathbf{i}_{\text{lin}}^{1/\gamma}, \tag{3.26}$$

where we assume a fixed $\gamma = 2.2$.

The Lambertian diffuse model can be used to render an image given illumination coefficients, normal map and diffuse albedo map. The model can be used as an image rendering decoder in neural network when we port the model as a rendering layer inside a neural network and inputs for the model are encodings from neural encoders. The renderer itself is a gamma-transformed linear function of inputs, so it is a differentiable layer. The renderings from this layer can be compared to input images, which

forms self-supervision loss.

3.3.2 Spherical harmonic lighting inference

The Lambertian diffuse renderer model allows us to linearly relate an inverse-gamma-corrected RGB image to decomposed normal and lighting, which also enables us to infer lighting estimate by finding the best fitting of spherical harmonic lighting with given input image and normal. For this we only need to reformulate the Equation. (3.24), where the linearity relationship between quantities is still preserved. Compared with using a fully-connected layer to predict illumination vector from input image, such a model-based decoder for lighting estimation allow us to reduce the number of network parameters and condition the prediction results to follow the reflectance model.

Consider an input image comprising K pixels. We invert the nonlinear gamma and stack the linearised RGB values to form the matrix $\mathbf{I} \in \mathbb{R}^{3\times K}$. We similarly stack the estimated albedo map to form $\mathbf{A} \in \mathbb{R}^{3\times K}$, the estimated surface normals to form $\mathbf{N} \in \mathbb{R}^{3\times K}$ and define $\mathbf{B}(\mathbf{N}) \in \mathbb{R}^{9\times K}$ by applying (3.25) to each normal vector. We can now rewrite (3.24) for the whole image as:

$$\mathbf{I} = \mathbf{A} \odot \mathbf{L}\mathbf{B}(\mathbf{N}), \tag{3.27}$$

We can now solve for the spherical harmonic illumination coefficients in a least squares sense, using the whole image. This can be done using any method, so long as the computation is differentiable such that losses dependent on the estimated illumination can have their gradients backpropagated into the inverse rendering network. For example, the solution using the pseudoinverse is given by:

$$\mathbf{L} = (\mathbf{I} \oslash \mathbf{A})\mathbf{B}(\mathbf{N})^{+}, \tag{3.28}$$

where $\oslash$ denotes element-wise division and $\mathbf{B}(\mathbf{N})^{+}$ is the pseudoinverse of $\mathbf{B}(\mathbf{N})$. We compute the pseudoinverse by singular value decomposition (SVD):

$$\mathbf{B}(\mathbf{N}) = \mathbf{U}\mathbf{S}\mathbf{V}^{T}, \tag{3.29}$$

$$\mathbf{B}(\mathbf{N})^{+} = \mathbf{V}\mathbf{S}^{-1}\mathbf{U}^{T}, \tag{3.30}$$

where $\mathbf{S}^{-1}$ is the 9×9 diagonal matrix with the reciprocal of the non-zero elements in $\mathbf{S}$:

$$\mathbf{S} = diag(s_1, \ldots, s_9), \tag{3.31}$$

$$\mathbf{S}^{-1} = diag(\frac{1}{s_1}, \ldots, \frac{1}{s_9}), \tag{3.32}$$

This lighting inference module finds the lighting estimate leading to the best image reconstruction with given albedo and normal. With the lighting inference result, we can define objective function to minimise the reconstruction error with respect to albedo and normal, which is useful when training an inverse rendering network (the model decomposes input colour image into albedo and normal map). The overall pipeline is working like an optimisation approach which alternates between solving for optimal lighting and optimising the neural network parameters for predicting albedo and normal. The training will eventually converge as long as we start the training from a reasonable initialisation, which is realised by our phased training scheme. We will discuss this in details in next chapter (see Section 4.5.2).

3.4 Experiments

3.4.1 Realistic rendering

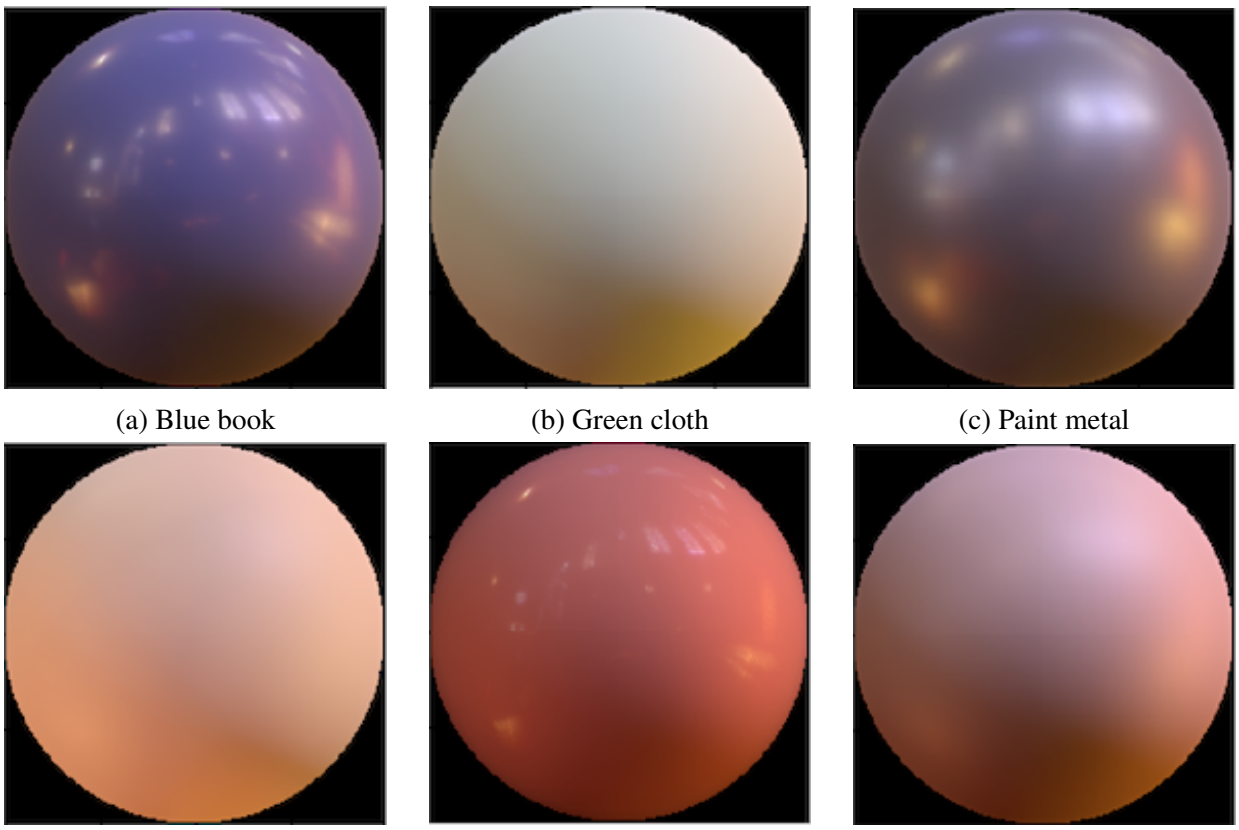
We now illustrate how the layers in PVNN can be combined to implement forward rendering. Such a network takes as input a depth map, one or more light source directions and colours and the BRDF parameters or an empirically measured BRDF and outputs a rendered image. To demonstrate the rendering capability of this network, We render spheres with illuminations represented by environment maps. Specifically, we perform geometric transformations defined above on depth map of a sphere, and the illumination is applied as a group of point lighting sources each of which is taken from a pixel in the environment map. The rendering image is computed by accumulating reflectance evaluations from all point lightings. In Figure 3.4 we show results using well reconstructed BRDF presented by

Nielsen *et al*. [117]. In Figure 3.5 we show results using a single set of statistical BRDF parameters and three different illumination environments. We note that our renderings are comparable to the output from conventional renderers as well as the statistical reconstructions shown by Nielsen *et al*. [117].

To illustrate the simplicity with which PVNN layers can be combined to achieve physically-based rendering, the following code snippet is all that is required to create images such as those in Figure 3.5:

```
# initialise renderer subnet
pvnn = pvnn_module.pvnn(shapes_var, mask_var)
# calculate surface normal map
normals = pvnn.depthToNormal(shapes_var)
# cosine weights for foreshortening
cosWeights = pvnn.cosWeight(normals, lights_var)
# filter out negatives
clampCosWeights = pvnn.clamping(cosWeights)
# reconstruct BRDF from predicted vector
brdfFn = pvnn.brdfFunction(prediction, './precomputedData')
# interpolate missing values in reconstruction
brdfFn = pvnn.brdfFnInterp(brdfFn, 'mapping.npy')
# indexing BRDF values for each pixel
brdfMatrix = pvnn.brdfValues(normals, lights_var, brdfFn)
# intensities formation
imgs = pvnn.brdfIntensity(brdfMatrix, clampCosWeights, lightColors_var, \
        mask_var)
# transpose output to have same shape with input
imgs = imgs.transpose(0,3,1,2)
```

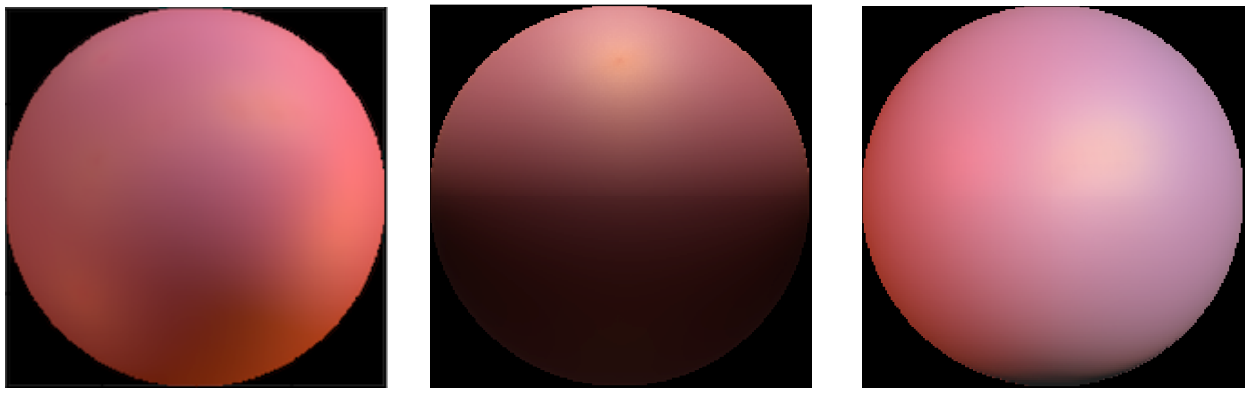
We now illustrate how the layers in PVNN can be combined to implement forward rendering. Such a network takes as input a depth map, one or more light source directions and colours and the BRDF parameters or an empirically measured BRDF and outputs a rendered image. To demonstrate the rendering capability of this network, we render spheres with environment map illuminations being decomposed as separate point light sources. In Figure 3.4 we show results using well reconstructed BRDF presented by [117]. In Figure 3.5 we show results using a single set of statistical BRDF



(a) Blue book (b) Green cloth (c) Paint metal

(d) Yellow paper (e) Glossy red paper (f) Cardboard

Figure 3.4: Images are rendered by our differentiable renderer. The inputs for the renderer are the BRDF function of the known material and environment map and depth map of the object, the output is the realistic rendered images.



(a) Grace Cathedral (b) Uffizi Gallery (c) Pisa

Figure 3.5: Three rendered examples from BRDF parameters under different HDR Environment Maps. All of our results are using same 5D parameters.

parameters and three different illumination environments. We note that our renderings are comparable to the output from conventional renderers as well as the statistical reconstructions shown by Nielsen *et al*. [117].

3.4.2 BRDF estimation

We now present an example of using PVNN in a vision application. We show how to train a CNN to directly regress BRDF parameters from a single image of the object with uniform material. Moreover, we train the network in an unsupervised fashion in the sense that BRDF parameters are not provided at training time. Instead, we use an encoder-decoder architecture in which the loss function is the L2 error between a rendering using the predicted parameters and the input image.

Architecture

An overview of the network is shown in Figure 3.6 including both training and inference subnets. The inference part inside the dashed box is a standard down-scaling ConvNet, taking input images and producing BRDF vector as output. The other parts of the network are untrainable and designed for unsupervised training. During training procedure, the weights in the ConvNet will be updated by the Error layer given output from the Renderer layer. As described in Section 3.2, the Renderer layer takes as input the depth maps, light sources and BRDF parameters and outputs realistic RGB images. The Error layer, placed at the end of our network, calculates a loss function by the L2 norm between the original input image and the rendered intensity map:

$$\ell_{\text{intensity}} = \|\mathbf{I}_{\text{obs}} - \mathbf{R}_{\text{pred}}\|_{\text{fro}}^2, \tag{3.33}$$

where $\mathbf{I}_{\text{obs}}$ is the original input image and $\mathbf{R}_{\text{pred}}$ is the rendering image that reconstructs the input.

In addition to this intensity loss, the network parameter regularisation term ℓ_{reg} is another part of our loss function. The coefficient of regularisation is 5×10^{-4}. The full training loss is:

$$\ell = \ell_{\text{intensity}} + 0.0005\ell_{\text{reg}}. \tag{3.34}$$

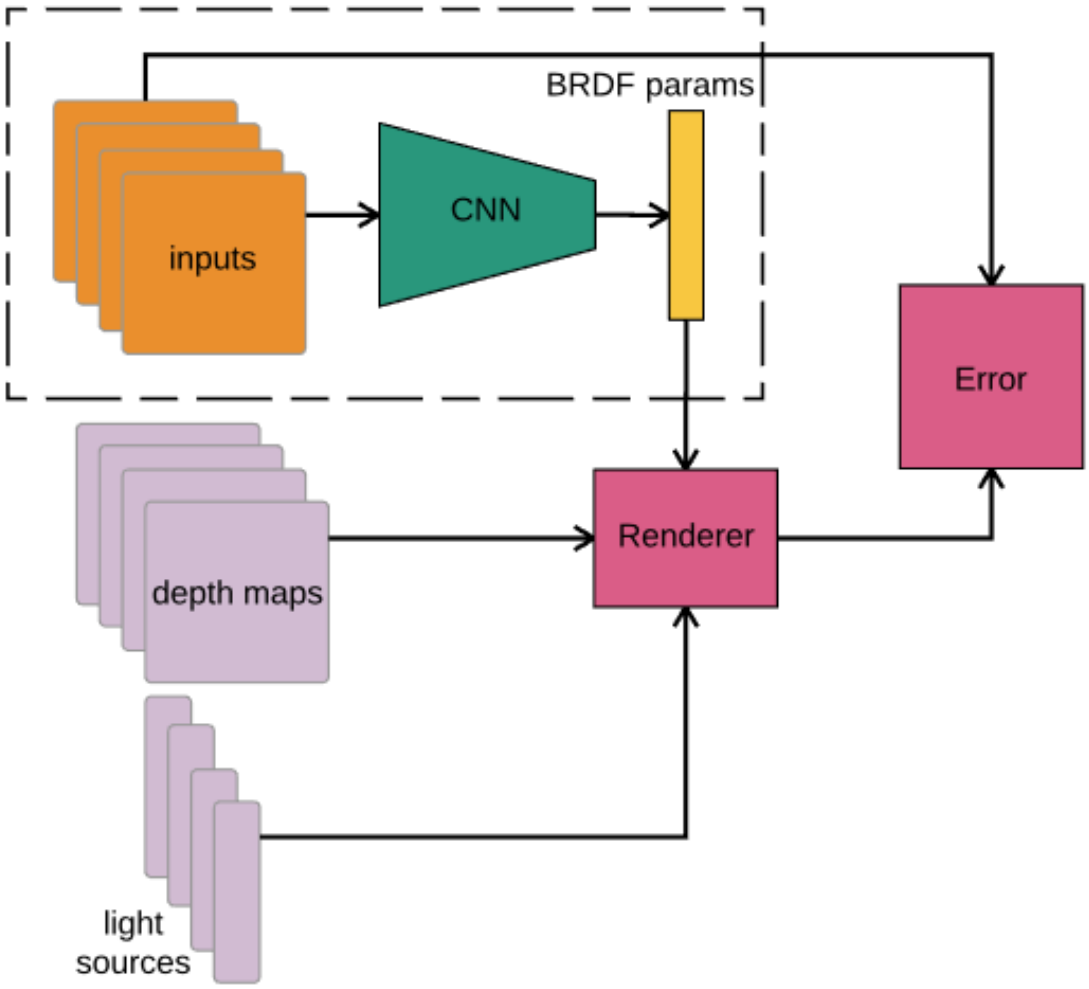


Figure 3.6: The overall architecture of our neural network. The dot-line box highlights the part for used for BRDF parameter inference during testing. The boxes in purple are indirectly supervisory annotations provided during the training to achieve unsupervised learning, and boxes in red are layers to formulate training loss.

Convolutional Neural Network

The architecture of our ConvNet is realised by a stack of residual blocks consisting of a simplified encoder-decoder architecture and a residual shortcut introduced by He *et al*. [49]. Since our problem is to estimate BRDF parameters from only one single image, a deep network with powerful learning ability is required. We construct our network with 50 layers following a ResNet layout. Also, following the downscale rule proposed by Simonyan *et al*. [140] in their VGG net, our network performs pooling operations followed by dimension increasing on the feature map. All the convolutional layers are separated by 4 groups after which spatial pooling is performed, and dimensional increments are achieved by the first layer of the next group. To concatenate 48 convolutional layers and 1 fully-

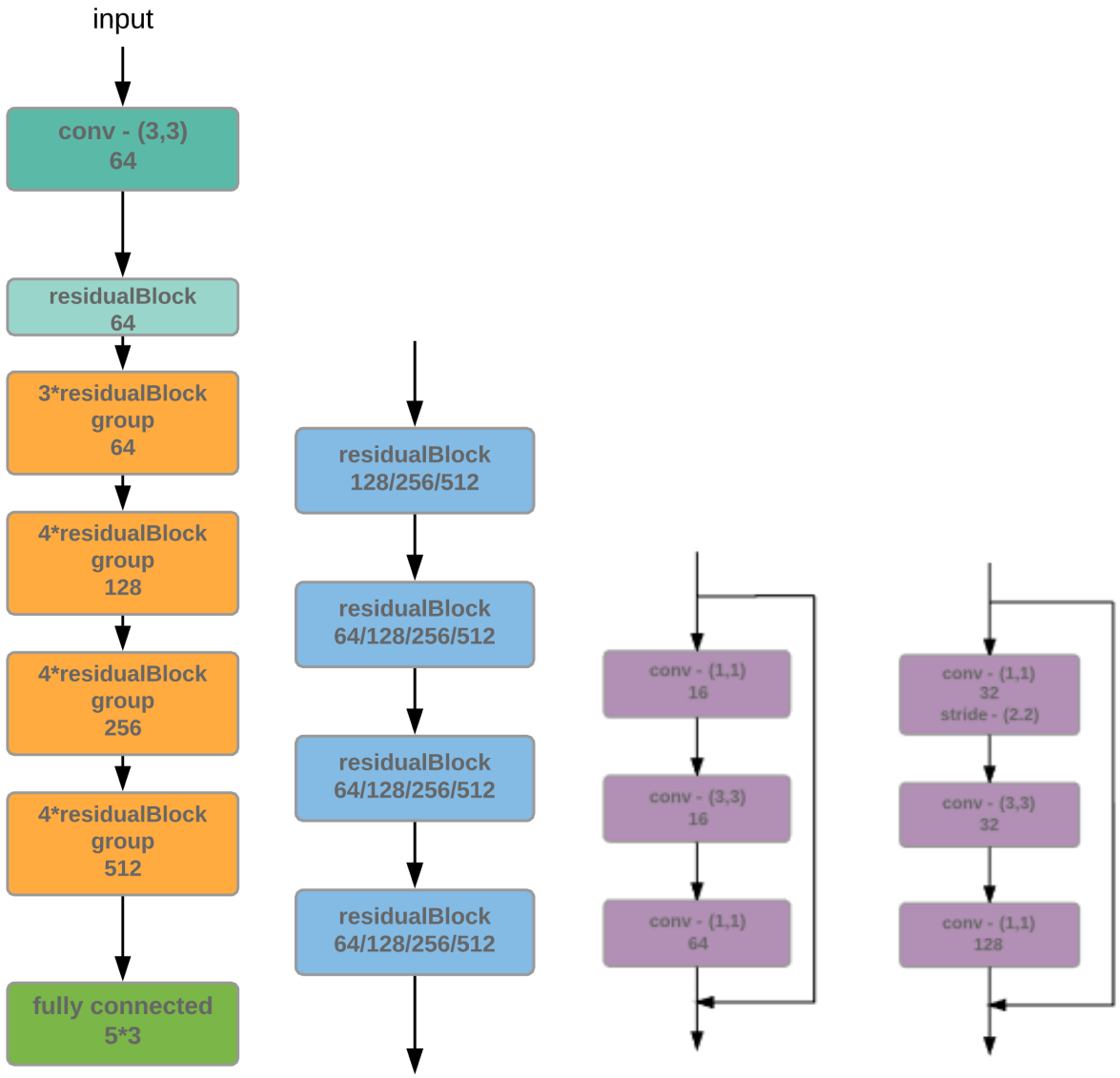


Figure 3.7: Network architecture. (a) Overall architecture; (b) Residual block stack, shown as orange boxes placed in the middle of (a); (c) Convolutional layers inside residual block; (d) Convolutional layers inside the first block of each group, which needs to increase the dimension of feature map and decrease spatial resolution at same time.

connected layer and 1 actuation convolutional layer, each convolutional group contains 1 dimension increment residual block and 3 normal blocks. The actuation layer in our network is placed next to the input layer, converting RGB input images to initial feature maps with 64 channels. The fully-connected layer at the end of the net maps output from convolutional layer to 15-D vectors, which represents three 5 dimensional BRDF statistical model parameters for each colour channel respectively. The visualisation of our network is shown in Figure 3.7. Note that the first residual block right after the actuation layer does not increase the dimension but maintains its spatial size and dimension.

Except for the first group, the pooling and dimension increment operations are proceeded by the first block in each group. For simplicity the pooling is done by using stride of size $(2, 2)$, whose efficiency has been proven by He *et al*. [49]. The architecture of the residual blocks can be found in Figure 3.7 as well. For each operational box, the numbers written at the bottom indicate the dimension of the output feature map. Figures 3.7d and 3.7c are examples of the first bottleneck block.

Inference

From the results in [117], here we use first 5 projected principal components in the statistical model, which is sufficient for good BRDF reconstructions. Although it would be straightforward to estimate more parameters, we use the same number as the original paper [117] to simplify the problem and make calibration easy. To fully reconstruct the BRDF look up table using our result vector, as part of PVNN we port their nonlinear reconstruction algorithm to Theano to output a standard `.binary` file.

Training

We employed SGD algorithm with batch size of 20 in backpropagation and run training 300 epochs. The learning schedule started by a relatively small value 0.001, which is good for network convergence as illustrated in [49]. A larger learning rate 0.01 is then substituted to speed up training. Since the learning process is unsupervised (via indirect comparison between prediction and ground truth), we found the network is slower to stabilise than supervised training which directly encourage prediction to approach ground truth. As a result, more epochs used in training are necessary. So, we keep using 0.01 as the learning rate for 200 epochs, which is longer than typical learning schemes.

For training we use 7000 synthetic images, and 3000 synthetic images are used for validation and testing respectively. Every image is rendered by using randomly drawn 15D BRDF parameters, 30 light sources with random direction and colour, and the depth map of an unit sphere. The BRDF parameters are drawn from a normal distribution with the variance for each dimension coming from the PCA model.

Dataset	**BRDF model Params.**	**Full BRDF**
Training	0.01425	1.951
Testing	0.01437	1.958

Table 3.1: RMSE on training and testing set. Two evaluation metrics are reported: the error of predicted coefficients of the BRDF PCA model, and the error of fully reconstructed BRDF from predicted parameters.

Evaluations

The experiments are deployed on synthetic data described in last section. We quantitatively evaluate the accuracy of the BRDF parameter estimates made by our inference network in two ways. First, we compute the RMSE between the predicted and actual 5D BRDF parameters (Table 3.1, first column). Second, we compute the RMSE between the BRDF reconstructed by our network and the actual BRDF (Table 3.1, second column). This second error can be compared directly to the values reported in [117], showing that we are achieving comparable accuracy without knowledge of the lighting conditions and using only a single image. We show qualitative results in Figure 3.8. The first two rows show actual input images (first row) and relightings using the estimated BRDF and ground truth lighting (second row). In the bottom two rows we show reilluminations of the ground truth (third row) and estimated (fourth row) BRDFs under environment lighting. In both cases the estimated BRDF gives very close visual appearance to ground truth and captures the key features of the reflectance properties.

3.4.3 Lambertian renderer and lighting estimation

In this section we show qualitative experiments of our Lambertian renderer and Lambertian-based lighting estimator. For clarity, we will call both of them as decoders, because they are leveraged to decode inputs like albedo and normal to specific outputs like image rendering or lighting. The two types of model-based decoders are essential components for self-supervised learning for outdoor inverse rendering network, which will be discussed specifically in next chapter, so the experimental images used in the section are only outdoor natural scenes. Although the Lambertian rendering model is known to perform badly on specularities, shadow casting and other non-Lambertian effects, we will show that the quality for renderings are good enough when only considering outdoor scenes. The

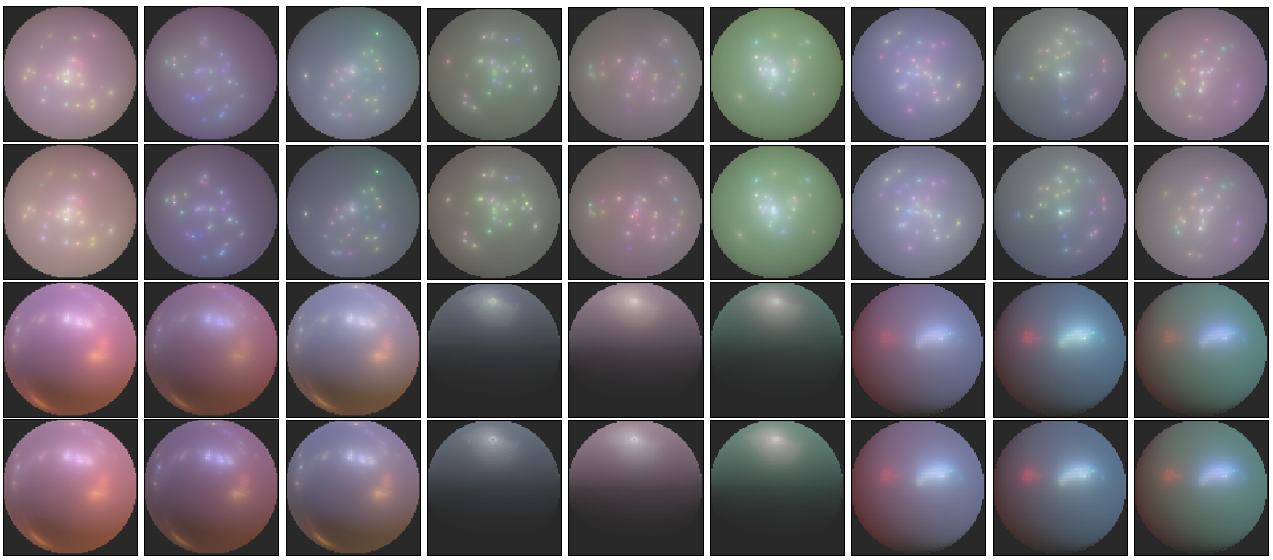


Figure 3.8: BRDF estimation results. The first row in each block is ground truth, and second row is reconstruction. Each column is the relighting results on the same material. The first two rows show the inputs for the network (row 1) and relighting using predicted BRDF (row 2). The second two rows of images are renderings under new lighting for both ground truth (row 3) and predicted (row 4) BRDF.

images used for experiments in this section are selected from MegaDepth [84], which contains a large number of outdoor images crawled from Flickr. Other than input images, both of our Lambertian-based renderer and lighting estimator require inputs from estimated intrinsic properties like normal and albedo, which again related to inverse renderings. For simplicity, we skip the details of obtaining those quantities while assuming their accessibility in evaluations.

Lambertian diffuse rendering

We present the rendering results from Lambertian renderer that takes as inputs the diffuse albedo map, normal map and spherical harmonic lighting in Figure 3.9. Note that the illuminations fed into the renderer should be a vector representing spherical harmonic parameters, but they are shown as lit spheres in Figure 3.9 for better visualisation. The second, fourth and sixth rows show renderings of the same scene under different illuminations. Although the rendering model is simple and failed at modelling complex materials and shading effects, rendering results for outdoor scenes are visually realistic. The inputs for image rendering decoder are simply albedo, normal and lighting, which means the Lambertian-based decoder enables us to train an inverse rendering neural network to decompose image into albedo, normal and lighting by comparing rendered images to the input image. As shown

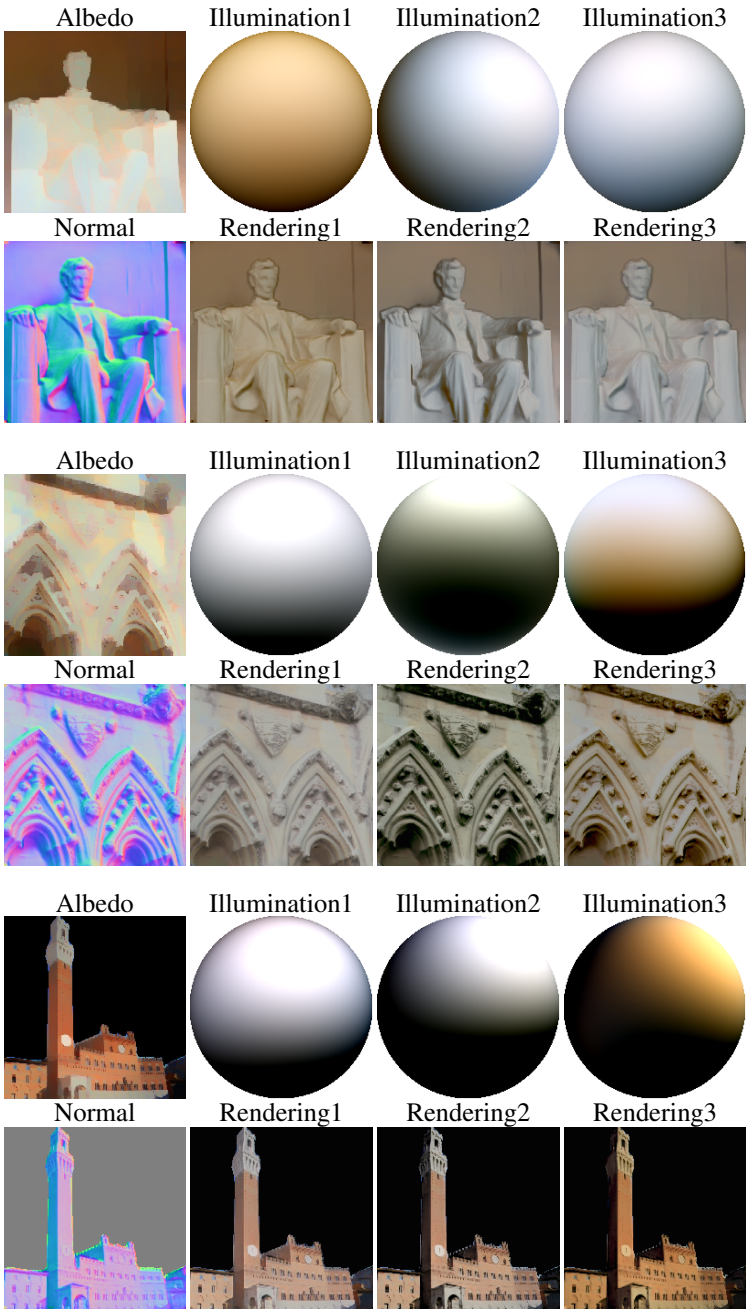


Figure 3.9: Rendering results from Lambertian renderer. Every two consecutive rows demonstrate one scene. The first column contains input albedo and normal map to the renderer. Each column after presents the illuminations and corresponding renderings.

in Figure 3.9 our rendering results demonstrate plausible and consistent shading variations and great level of realism.

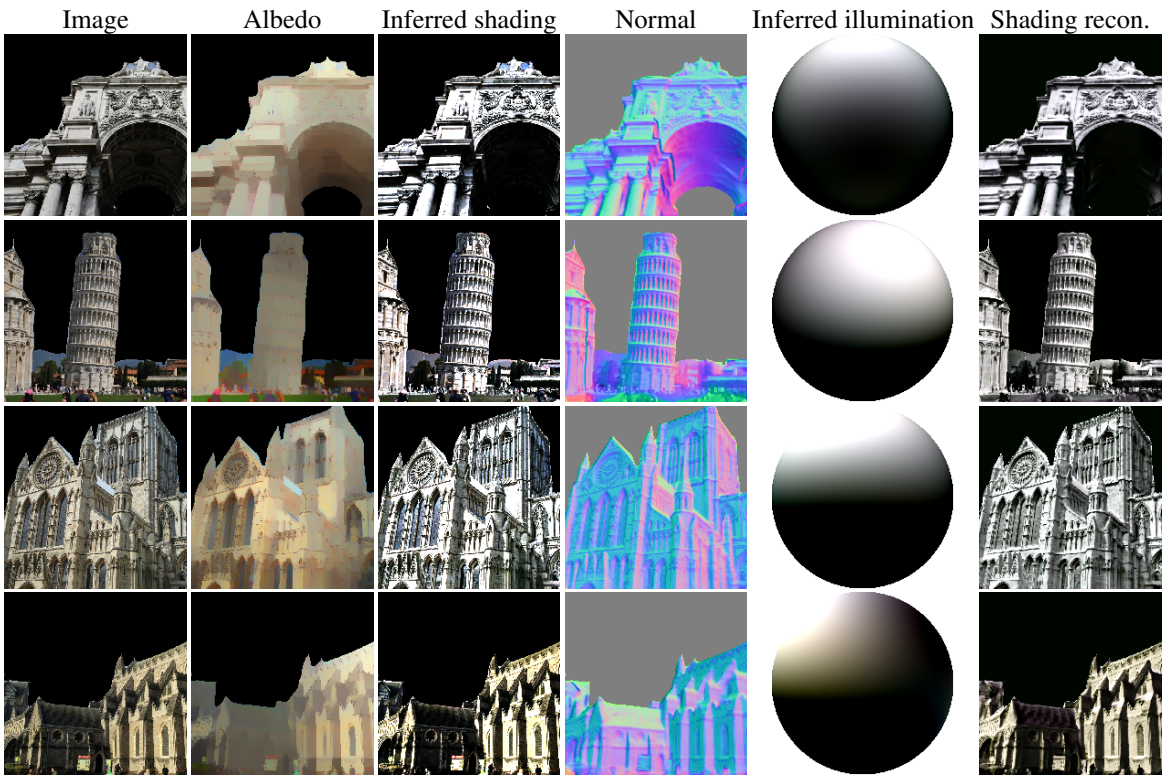


Figure 3.10: Illumination inference results. The first and second column are original images and their albedo map. The shading inferences computed by dividing images by albedos are shown in third column. Given the normal maps, illuminations can be inferred and shown on fifth column. The last column contains reconstructed shadings from inferred illuminations and normal maps.

Lighting inference

Now we show results from our illumination estimator. To evaluate the quality of our lighting inference decoder, we apply our decoder on an input image and its decomposed albedo map and normal map. The visual results are shown in Figure 3.10. In addition, the comparison between shading reconstruction and initial shading inference, column 3 and column 6 can be found in the figure, exhibiting a better view of illumination estimation fidelity. Here the inferred shading is computed by pixel-wise dividing image by albedo, and shading reconstruction is rendered by estimated lighting and normal. This consistency reflects the accuracy of our lighting inference model. From the evaluations it is evident that our simple lighting decoder can achieve plausible lighting estimates whilst performing efficiently compared to a neural decoder.

3.5 Conclusion

In this chapter we have presented a collection of novel differentiable layers and decoders that can be injected into a neural network and acts as physical photometric image formation models. We also illustrated how these layers can be used in a simple photometric vision application, though the setup was quite simple (the images are always of spheres, so the inference network essentially "knows" the shape of the object even though this is not explicit). However, there are a wide array of much more ambitious applications where the model-based decoders could be used. We will explore the application of these ideas to more general inverse rendering in the next chapter.

Regarding the model-based decoders themselves, we draw the following conclusions. First, it is computationally cheap, effective and differentiable to perform rendering in screen space, even using complex non-Lambertian reflectance models. Our models are simple enough that derivatives can be explicitly derived by hand. This is useful for sanity checking and identifying potential instabilities in gradients during backpropagation. Incorporating model-based decoder is also beneficial for pruning the network architectures as the unknowns can be inferred by other known values by our models (e.g. lighting inference model). More importantly, the model can relax the need of labels (e.g. ground truth BRDFs) that are difficult to capture, therefore making infeasible training problems more approachable and accessible. However, the rendering models described in this chapter neglect non-local effects such as shadowing and ambient occlusion. Any inverse renderer built on top of these layers will inherit these limitations. Also, compared with neural decoders consisting of numerous learned parameters, the model-based decoder is composed by only a limited number of parameters, so restricting the expressiveness of the model, which in return makes outputs worse than that of neural decoders. Although we have shown some valid differentiable models and their derivatives, it's worth notice the potential gradient flow issue commonly presented in differentiable model-based decoders. The issue is often introduced by discrete operations in forward flow, for example z-buffering in the process of rasterisation, which is a differentiable operation itself but leads to zero gradient flow during backpropagation. This issue is usually tackled by formulating the discrete operation as a probabilistic continuous operation, which in return involves extra computational efforts.

Chapter 4

Self-supervision and inverse rendering

4.1 Introduction

Deep learning has already shown good performance on components of the inverse rendering problem. This includes monocular depth estimation [26], depth and normal estimation [25] and intrinsic image decomposition [80]. However, these works use supervised learning. For tasks where ground truth does not exist, such approaches must either train on synthetic data (in which case generalisation to the real world is not guaranteed) or generate pseudo ground truth using an existing method (in which case the network is just learning to replicate the performance of the existing method). Inverse rendering of outdoor, complex scenes is itself an unsolved problem and so reliable ground truth is not available and supervised learning cannot be used. Following the idea of model-based decoder as well as self-supervised framework proposed in the last chapter, we develop a self-supervised inverse rendering network based upon model-based decoders for Lambertian image reconstruction and MVS image cross-projection. Our method defines this cross-projection as a MVS-based supervision, which uses 3D geometry and camera parameters to re-project inverse rendering results from one 2D image to 3D points followed by forward-projecting these 3D points to another image plane. The cross-projection mechanism relates network outputs from one image to that of other images, formulating the error of inverse rendering results. Note that re-projection is essentially different from our cross-projection, because our cross-projection is used to measure the error in inverse rendering while assuming geometry

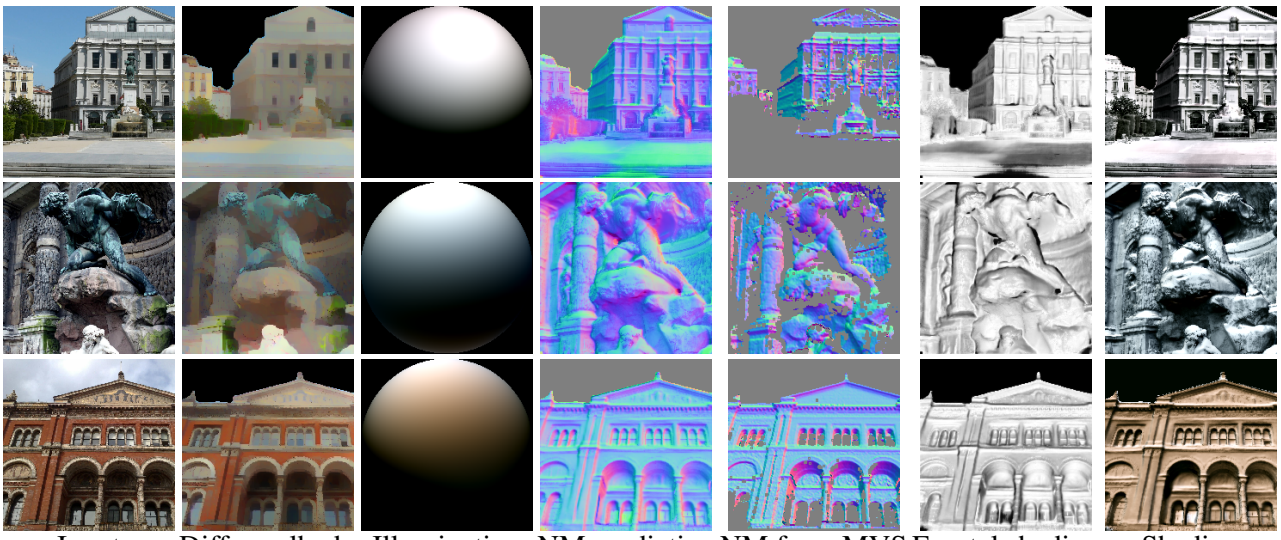


Figure 4.1: From a single image (col. 1), we propose InverseRenderNet estimating albedo and normal maps and illumination (col. 2-4); comparison multiview stereo result from several hundred images (col. 5); re-rendering of our shape with frontal/estimated lighting (col. 6-7).

is correct. But re-projection is used to measure geometry error in MVS optimisation.

In this context, we present two variants of the inverse rendering networks that help us seek insights for the following points in this chapter. First, we exploit MVS supervision for learning inverse rendering. Second, we tackle the most general version of the problem, considering arbitrary outdoor scenes and learning from real data, as opposed to restricting to a single object class [146] or using synthetic training data [170]. Third, we introduce a statistical model of spherical harmonic lighting in natural scenes that we use as a prior. Fourth, the resulting network is aiming to inverse render all of shape, reflectance, lighting and shadow in the wild, outdoors. Finally, we improve the labels obtained from MVS geometry by explicitly detecting ground plane pixels (which have unreliable MVS depth estimates) and replacing them with an estimated ground plane normal direction and use the new labels for direct normal map supervision.

The remainder of this chapter will be organised as follows: the preliminary physical model will be introduced by Section 4.2. Then the inverse rendering neural network referred as InverseRenderNet will be presented in Section 4.3, 4.4 and 4.5, followed by experimental results in Section 4.6 An improved version of InverseRenderNet will be discussed in Section 4.7, which we refer to as InverseRenderNet++. More experimental results including the comparison between InverseRenderNet and InverseRenderNet++ will be demonstrated in Section 4.8. Finally, the conclusion about the

chapter is presented in Section 4.9.

4.2 Preliminaries

In this section we define notation, introduce basic models and assumptions and describe the representation we use for shape.

We assume that a perspective camera observes a scene, such that the projection from 3D world coordinates, (u, v, w), to 2D image coordinates, (x, y), is given by:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}, \quad \mathbf{P} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.1}$$

where λ is an arbitrary scale factor, $\mathbf{R} \in SO(3)$ a rotation matrix, $\mathbf{t} \in \mathbb{R}^3$ a translation vector, f the focal length and (c_x, c_y) the principal point.

The inverse rendered shape estimate could be represented in a number of ways. For example, many previous methods estimate a viewer-centred depth map. However, local reflectance, and hence appearance, is determined by surface orientation, i.e. the local surface normal direction. So, to render a depth map for self-supervision, we would need to compute the surface normal. From a perspective depth map $w(x, y)$, the surface normal direction in camera coordinates is given by Nehab *et al*. [112]:

$$\bar{\mathbf{n}} = \begin{bmatrix} -f w_x(x, y) \\ -f w_y(x, y) \\ (x - c_x) w_x(x, y) + (y - c_y) w_y(x, y) + w(x, y) \end{bmatrix}, \tag{4.2}$$

from which the unit length normal is given by: $\mathbf{n} = \bar{\mathbf{n}} / \|\bar{\mathbf{n}}\|$. The derivatives of the depth map in the image plane, $w_x(x, y)$ and $w_y(x, y)$, can be approximated by finite differences. However, Equation (4.2) requires knowledge of the intrinsic camera parameters. This would severely restrict the applicability of our method. For this reason, we choose to estimate a surface normal map directly.

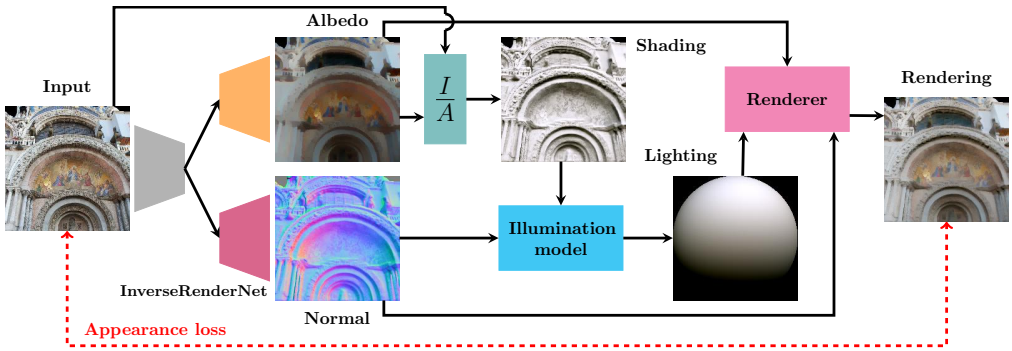


Figure 4.2: At inference time, our network regresses diffuse albedo and normal maps from a single, uncontrolled image and then computes least squares optimal spherical harmonic lighting coefficients. At training time, we introduce self-supervision via an appearance loss computed using a differentiable renderer and the estimated quantities.

Although the surface normal can be represented by a 3D vector, since $\|\mathbf{n}\|_2 = 1$ it has only two degrees of freedom. So, our network estimates two quantities per-pixel: n_x/n_z and n_y/n_z. Since for all visible pixels, $n_z \geq 0$, we can compute the surface normal direction from the estimated quantities as:

$$\mathbf{n} = \frac{[n_x/n_z, n_y/n_z, 1]^T}{\|[n_x/n_z, n_y/n_z, 1]\|}. \tag{4.3}$$

Note that the extreme surface orientations like ground plane having $n_z = 0$ can be approximated by very large predictions of n_x/n_z and n_y/n_z.

We assume that appearance can be approximated by a local reflectance model under environment illumination, modulated by a scalar shadowing/ambient occlusion term. Specifically, we use a Lambertian diffuse model with order 2 spherical harmonic lighting, which is fully explored on Section 3.3 from last chapter.

4.3 Architecture

Our inverse rendering network (see Figure 4.2) is an image-to-image network that regresses albedo and normal maps from a single image and uses these to estimate lighting. We describe these inference components in more detail here.

4.3.1 Trainable encoder-decoder

We implement a deep fully-convolutional neural network with skip connections like the hourglass architecture [115]. We use a single encoder and separate transposed convolution decoders for albedo and normal prediction. Albedo maps have 3 channel RGB output, normal maps have two channels for the surface gradient which is converted to a normal map as described above. Both convolutional subnet and deconvolutional subnet contain 15 layers and the activation functions are ReLUs. Adam Optimiser [70] is used in training, with learning rate of 0.05, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 0.9$.

4.3.2 Implicit lighting prediction

In order to estimate illumination parameters, one option would be to use a fully connected branch from the output of our decoder and train our network to predict it directly. However, fully connected layers require very large numbers of parameters and, in fact, lighting can be inferred from the input image and estimated albedo and normal maps, making its explicit prediction redundant. An additional advantage is that the architecture remains fully convolutional and so can process images of any size at inference time. Therefore, the lighting prediction in our network is implicitly estimated by the linear lighting inference model introduced in Section 3.3.

4.4 Supervision

As shown in Figure 4.2, we use a L2 data term (the error between predicted and observed appearance) for self-supervision. However, inverse rendering using only a data term is ill-posed (an infinite set of solutions can yield zero data error) and so we use additional sources of supervision, all of which are essential for good performance. We describe all sources of supervision in this section.

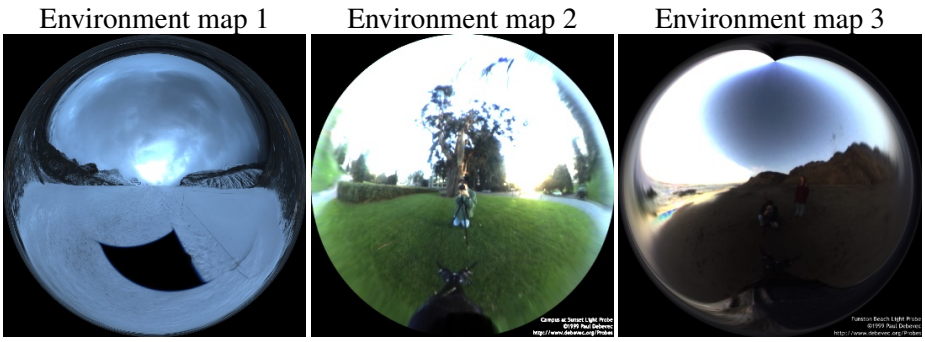


Figure 4.3: Examples of environment maps used in illumination statistical model.

4.4.1 Self-supervision via differentiable rendering

Given estimated normal and albedo maps and spherical harmonic illumination coefficients, we compute a predicted image $\mathbf{I}_{\text{pred}}$ using Equation (3.27) and (3.26):

$$\mathbf{I}_{\text{pred}}(\mathbf{A}, \mathbf{L}, \mathbf{N}) = (\mathbf{A} \odot \mathbf{LB}(\mathbf{N}))^{1/\gamma}, \tag{4.4}$$

where $\mathbf{A}$ is the albedo map, $\mathbf{L}$ is the spherical harmonic illumination coefficients, $\mathbf{N}$ represents normal map, and γ of 2.2 is used in this equation.

This local illumination model is straightforward to differentiate. Self-supervision is provided by the error between the predicted, $\mathbf{I}_{\text{pred}}$, and observed, $\mathbf{I}_{\text{obs}}$, intensities. Inspired by [168], computing this appearance loss directly in RGB space does not yield the best results since it is not the best measure of the perceptual quality of the reconstruction. For this reason, we compute this error in LAB space as this provides perceptually more convincing results:

$$\ell_{\text{appearance}} = \|\text{LAB}(\mathbf{I}_{\text{pred}}) - \text{LAB}(\mathbf{I}_{\text{obs}})\|^2_{\text{fro}}, \tag{4.5}$$

where LAB performs the colour space transformation.

4.4.2 Natural illumination model and prior

The spherical harmonic lighting model in Equation (3.24) enables efficient representation of complex lighting. However, even within this low dimensional space, not all possible illumination environments

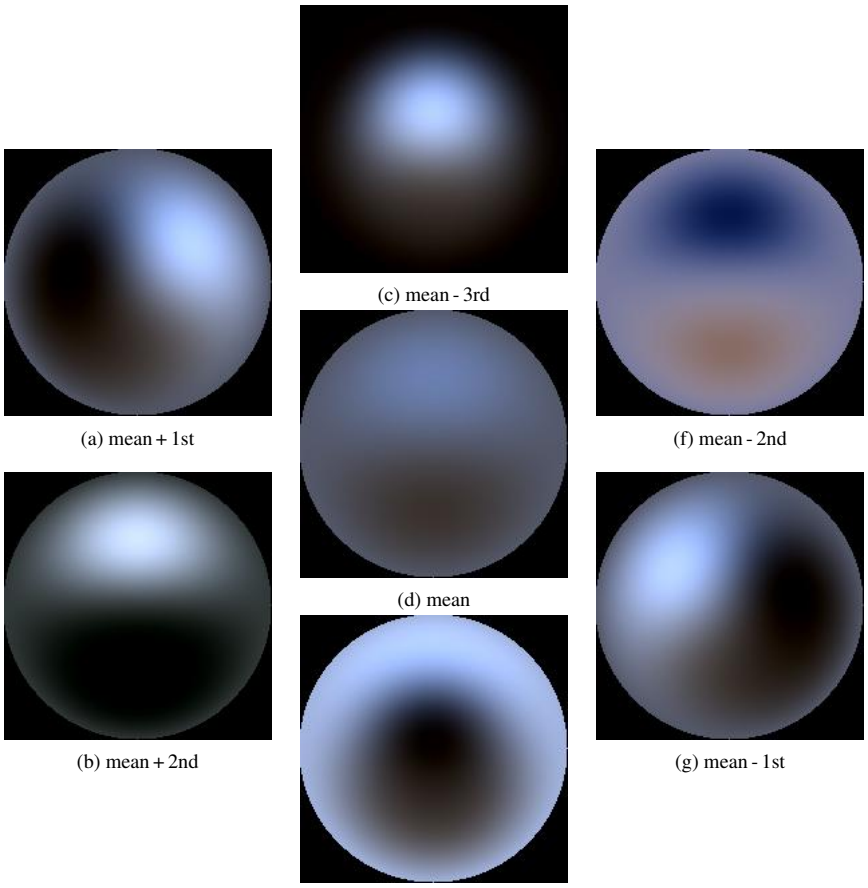


Figure 4.4: Statistical illumination model. The central image shows the mean illumination. The two diagonals and the vertical show the first 3 principal components.

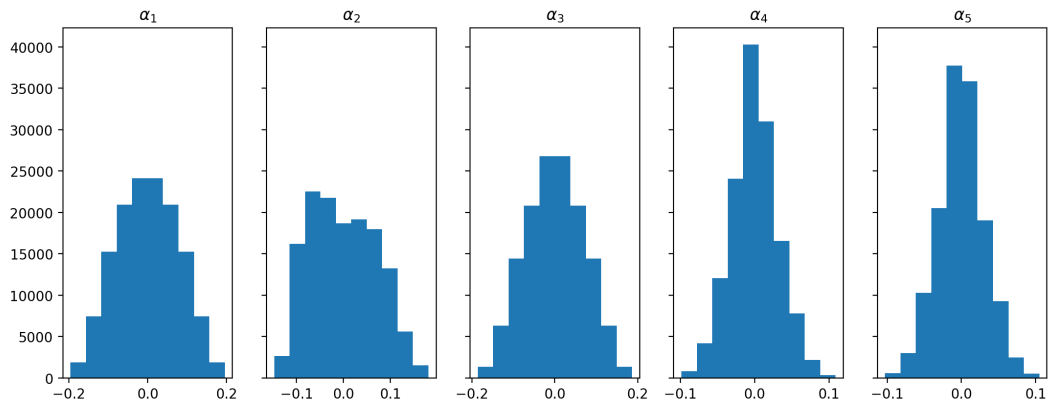


Figure 4.5: Data distribution of the first 5 coefficients.

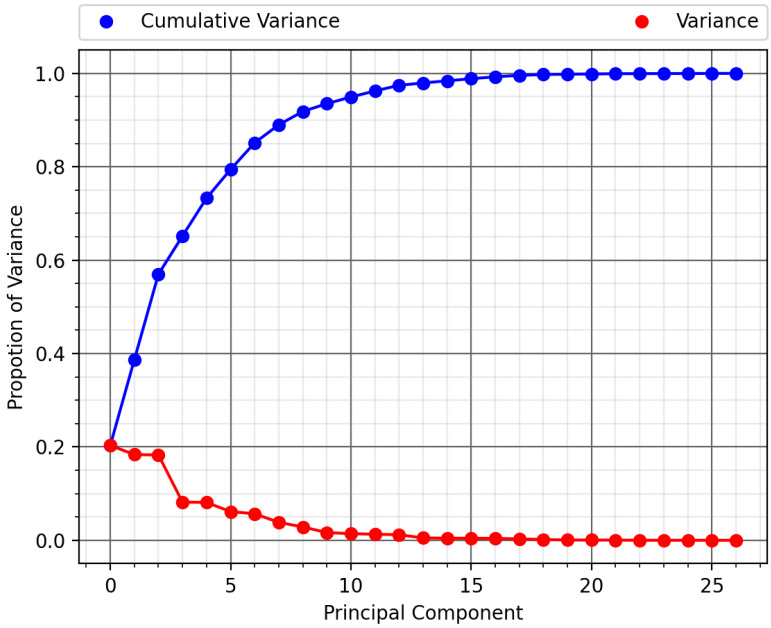


Figure 4.6: Variance and cumulative variance of the principal components of the illumination model.

are natural. The space of natural illumination possesses statistical regularities [23]. We can use this knowledge to constrain the space of possible illumination and enforce a prior on the illumination parameters. To do this, we build a statistical illumination model (see Figure 4.4) using a dataset of 79 HDR spherical panoramic images taken outdoors [74, 1]. As shown in Figure 4.4, the mean value of the model shows a blue-ish lighting coming from sky. The first three principal components introduces variations of illumination direction and colour. However, the colour only varies within the range of blue, white, yellow and black, and the variation of directions is also limited to prevent unnatural outdoor illuminations and so lightings always come from sky rather than ground. Thus, applying this PCA model on the top of spherical harmonic lighting ensures our lighting prediction is not arbitrary like green lighting emitted from ground plane. Such a lighting prior is very important for self-supervised training for inverse rendering network, because the problem itself is ill-posed and cannot be solved only with self-supervision. For examples, only self-supervision can drive network to predict an unnatural outdoor illumination and a strange albedo to perfectly reconstruct the input image.

Some examples of this set of illumination data are shown in Figure 4.3. Environment maps shown in this figure have been mapped to light probes, from which we can easily compute spherical harmonic coefficients. For each environment, we compute the spherical harmonic coefficients, $\mathbf{L}_i \in \mathbb{R}^{3\times 9}$.

Since the overall intensity scale is arbitrary, we also normalise each lighting matrix to unit norm, $\|\mathbf{L}_i\|_{\text{Fro}} = 1$, to avoid ambiguity with the albedo scale. Our illumination model in Equation (3.27) uses surface normals in a viewer-centred coordinate system. So, the dataset must be augmented to account for possible rotations of the environment relative to the viewer. Since the rotation around the vertical (v) axis is arbitrary, we rotate the lighting coefficients by angles between 0 and 2π in increments of $\pi/18$. In addition, to account for camera pitch or roll, we additionally augment with rotations about the u and w axes in the range $(-\pi/6, \pi/6)$. This gives us a dataset of 139,356 environments. We then build a statistical model by running PCA, such that any illumination can be approximated as:

$$\text{vec}(\mathbf{L}) = \mathbf{P}\text{diag}(\sigma_1, \ldots, \sigma_D)\boldsymbol{\alpha} + \text{vec}(\bar{\mathbf{L}}). \tag{4.6}$$

where $\mathbf{P} \in \mathbb{R}^{27\times D}$ contains the principal components, $\sigma_1^2, \ldots, \sigma_D^2$ are the corresponding eigenvalues, $\bar{\mathbf{L}} \in \mathbb{R}^{3\times 9}$ is the mean lighting coefficients and $\boldsymbol{\alpha} \in \mathbb{R}^D$ is the parametric representation of $\mathbf{L}$. We empirically found using $D = 18$ dimensions allows the model to express enough variations with limited noises like unnatural outdoor illumination colours. The proportion of variance explained by including the 18th principal components is about 99%. The variance and cumulative variance of extracted principal components are shown in Figure 4.6. We project the illumination data onto this statistical model and found that the parameters are Gaussian distributed: $\boldsymbol{\alpha} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\sigma})$. Figure 4.5 visualises the distribution of the first 5 principal component coefficients. When we compute lighting, we do so within the subspace of the statistical model, i.e. we substitute Equation (4.6) into Equation (3.28) and solve linearly for $\boldsymbol{\alpha}$. In addition, we introduce a prior loss on the estimated lighting vector:

$$\ell_{\text{lighting}} = \|\boldsymbol{\alpha}\|_2^2. \tag{4.7}$$

This lighting prior loss penalises the large variation of lighting inference results and encourages the predicted lightings to be close to the mean of our statistical model. As a result, the inverse rendering network can regress normals and albedos yielding lighting estimates with reasonable colour and directional distribution.

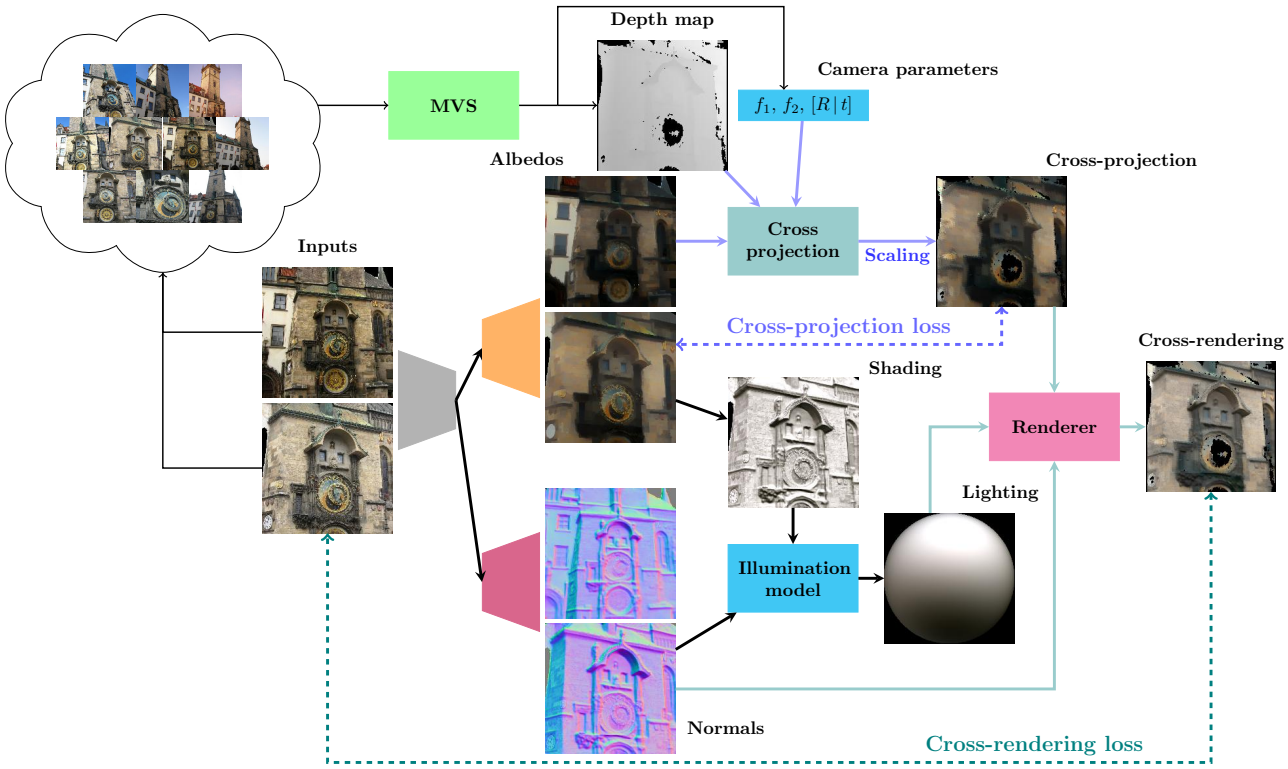


Figure 4.7: Siamese MVS supervision: albedo cross-projection consistency and cross-rendering losses (shown in one direction for simplicity). Note: shading depends on input and albedo as in Figure 4.2 but this dependency is excluded for simplicity.

4.4.3 Multiview stereo supervision

A pipeline comprising structure-from-motion followed by multiview stereo [129] (which we refer to simply as MVS) enables both cameras poses and dense 3D scene models to be estimated from large, uncontrolled image sets. Of particular importance to us, these pipelines are relatively insensitive to illumination variation between images in the dataset since they rely on matching local image features that are themselves illumination insensitive. We emphasise that MVS is run offline prior to training and that at inference time our network uses only single images of novel scenes. We use the MVS output for three sources of supervision.

Cross-projection

We use the MVS poses and depth maps to establish correspondence between views, allowing us to cross-project quantities between overlapping images. Given an estimated depth map, $w(x, y)$, in view i and camera matrices for views i and j, a pixel (x, y) can be cross-projected to location (x', y') in

view j via:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \mathbf{K}_j \begin{bmatrix} \mathbf{R}_j & \mathbf{t}_j \end{bmatrix} \begin{bmatrix} \mathbf{R}_i^T & -\mathbf{R}_i^T \mathbf{t}_i \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} w(x,y)\mathbf{K}_i^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \\ 1 \end{bmatrix}. \tag{4.8}$$

In practice, we perform the cross-projection in the reverse direction, computing non-integer pixel locations in the source view for each pixel in the target view. We can then use bilinear interpolation of the source image to compute quantities for each pixel in the target image. Since the MVS depth maps contain holes, any pixels that cross project to a missing pixel are not assigned a value. Similarly, any target pixels that project outside the image bounds of the source are not assigned a value. We denote the cross-projection of any quantity $\mathbf{x}_i$ defined on the pixels of image i onto the pixels of image j as $\text{proj}_{i \to j}(\mathbf{x}_i)$.

Direct normal map supervision

The per-view depth maps provided by MVS can be used to estimate normal maps, albeit that they are typically coarse and incomplete (see Figure 4.1, column 5). We compute guide normal maps from the depth maps and intrinsic camera parameters estimated by MVS using Equation (4.2). The guide normal maps are used for direct supervision by computing a loss that measures the angular difference between the guide, $\mathbf{n}_{\text{guide}}$, and estimated, $\mathbf{n}_{\text{est}}$, surface normals:

$$\ell_{\text{NM}} = \arccos(\mathbf{n}_{\text{guide}} \cdot \mathbf{n}_{\text{est}}). \tag{4.9}$$

Albedo consistency loss

Diffuse albedo is an intrinsic quantity. Hence, we expect that albedo estimates of the same scene point from two overlapping images should be the same, even if the illumination varies between views. Hence, we automatically select pairs of images that overlap (defined as having similar camera locations and similar centres of mass of their backprojected depth maps). We discard pairs that do not contain illumination variation (where cross-projected appearance is too similar). Then, we train our

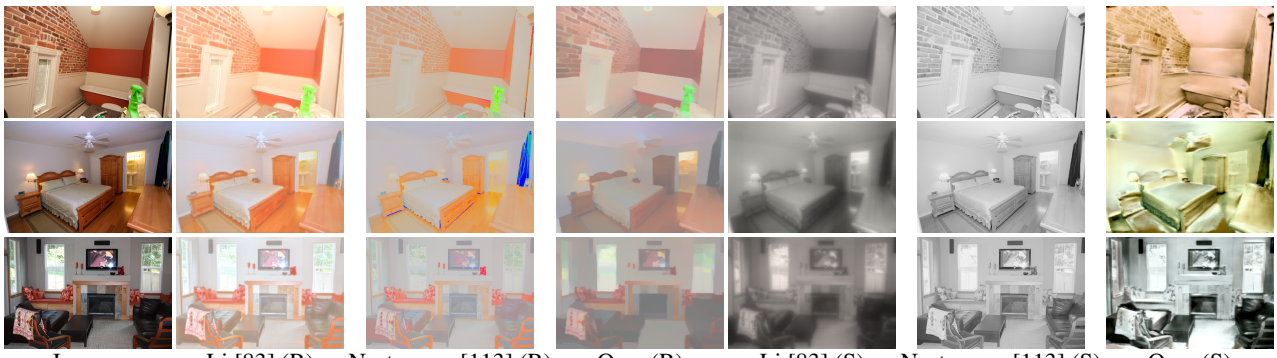


Figure 4.8: Qualitative results for IIW. Second column to forth column are reflectance predictions from [83], [113] and ours. The last three columns are corresponding shading predictions.

network in a Siamese fashion on these pairs and use the cross projection described above to compute an albedo consistency loss:

$$\ell_{\text{albedo}} = \left\| \text{LAB}(\mathbf{A}_i) - \text{LAB}\left(\text{proj}_{j \to i}(s\mathbf{A}_j)\right) \right\|_{\text{fro}}^2, \tag{4.10}$$

where $A_i, A_j \in \mathbb{R}^{3\times K}$ are the estimated albedo maps in the ith and jth images respectively. The scalar s is the value that minimises the loss and accounts for the fact that there is an overall scale ambiguity between images. This loss is masked both by the sky mask in image i and by the cross-projected pixels for which image j has a defined MVS depth value. Again, we compute albedo consistency loss in LAB space. The albedo consistency loss is visualised by the blue arrows in Figure 4.7.

Cross-rendering loss

For improved stability, we also use a mixed cross-projection/appearance loss, $\ell_{\text{cross-rend}}$. We use the cross-projected albedo above in conjunction with the estimated normals and illumination to render a new image and measure the appearance error in the same way as Equation (4.5). Note that we only cross-project albedo predictions from other views and fix normal and lighting prediction. Although normal is another invariant intrinsic quantity, which can be cross-projected like albedo, cross-projection is not applied on normal predictions. The reasons of this design choice are twofold: i) The direct normal supervision based on MVS is effective and implicitly accounting for multi-view invariant; ii) Mixing cross-projections of both albedo and normal could lead the network hard to converge and expensive to compute. This loss is visualised by the green arrows in Figure 4.7, and it could be

computed by:

$$\ell_{\text{cross-rend}} = \left\| \text{LAB}(\mathbf{I}_i) - \text{LAB}\left(\mathbf{I}_{\text{pred}}(\text{proj}_{j\to i}(s\mathbf{A}_j), \mathbf{L}_i, \mathbf{N}_i)\right)\right\|^2_{\text{fro}}, \tag{4.11}$$

where $\mathbf{I}_i$ is the ith input image, $\mathbf{L}_i$ and $\mathbf{N}_i$ are lighting and normal predictions from ith input. It follows the image formation model $\mathbf{I}_{\text{pred}}$ introduced by Equation (4.4).

4.4.4 Albedo priors

Finally, we also employ two additional prior losses on the albedo. This helps resolve ambiguities between shading and albedo. First, we introduce an albedo smoothness prior, $\ell_{\text{albedo-smooth}}$. Rather than uniformly applying smoothness penalty, we apply a pixel-wise varying weighted penalty according to chromaticities of the input image, which was originally proposed by Jeon *et al*. [56]. So, the stronger smoothness penalties are only enforced on neighbouring pixels with closer chromaticities. The loss itself is the L1 distance between adjacent pixels:

$$\ell_{\text{albedo-smooth}} = \sum_{p\in\mathcal{P}} \sum_{q\in\mathcal{N}(p)} \beta_{pq} \left\|\mathbf{A}(p) - \mathbf{A}(q)\right\|, \tag{4.12}$$

where $\mathbf{A}_p$ and $\mathbf{A}_q$ are albedo prediction at pixel p and q. $\mathcal{P}$ denotes all pixels in the image, and $\mathcal{N}(p)$ defines the neighbouring pixels for pixel p. For simplify, we define this neighbourhood as a set of two pixels that are horizontally and vertically next to the pixel p. β_{pq} indicates the chromaticity-based weight, which can be computed by:

$$\beta_{pq} = \exp(-\frac{1-(\mathbf{m}_p \cdot \mathbf{m}_q)^2}{c}), \tag{4.13}$$

in which c is the constant set as 0.0001, $\mathbf{m}_p$ and $\mathbf{m}_q$ are chromaticity values which are normalised 3D colour values of pixel p and q.

Second, during the self-supervised phase of training, we also introduce a pseudo supervision loss to prevent convergence to trivial solutions. After the pretraining process (see Section 4.5.2), our model learns plausible albedo predictions using MVS normals. To prevent subsequent training diverging too

far from this, we encourage albedo predictions to remain close to the pretrained albedo predictions:

$$\ell_{\text{albedo-pseudoSup}} = \|\text{LAB}(\mathbf{A}_{\text{pred}}) - \text{LAB}(\mathbf{A}_{\text{pretrained}})\|^2_{\text{fro}}. \tag{4.14}$$

In this pseudo supervision loss, our new albedo prediction, $\mathbf{A}_{\text{pred}}$ is encouraged to stay close to albedo predicted by pretrained model, $\mathbf{A}_{\text{pretrained}}$. Note that the pretrained model is used as an untrainable network at this phase, such that our pseudo ground truth $\mathbf{A}_{\text{pretrained}}$ can be obtained by performing albedo inference on this pretrained model.

4.5 Training

We train our network to minimise:

$$\ell = w_1\ell_{\text{appearance}} + w_2\ell_{\text{NM}} + w_3\ell_{\text{albedo}} + w_4\ell_{\text{cross-rend}} + w_5\ell_{\text{albedo-smooth}} + w_6\ell_{\text{albedo-pseudoSup}} + w_7\ell_{\text{lighting}}. \tag{4.15}$$

Empirically, we set the weights as w_1 = 0.1, w_2 = 1, w_3 = 0.05, w_4 = 0.1, w_5 = 50, w_6 = 0.1 and w_7 = 0.01.

4.5.1 Datasets

We train using the MegaDepth [84] dataset. This contains dense depth maps and camera calibration parameters estimated from crawled Flickr images. The pre-processed images have arbitrary shapes and orientations. For ease of training, we resize images such that the smaller side is of size 200 pixels, then crop multiple 200×200 square images. We choose our crops to maximise the number of pixels with defined depth values. Where possible, we crop multiple images from each image, achieving augmentation as well as standardisation. We adjust the camera parameters to account for the scale and crop. We create mini-batches in which all pairs of images within the mini-batch overlap (defined as having similar camera locations and similar centres of mass of their backprojected depth maps) and with sufficient illumination variation (we discard pairs where cross-projected appearance is too

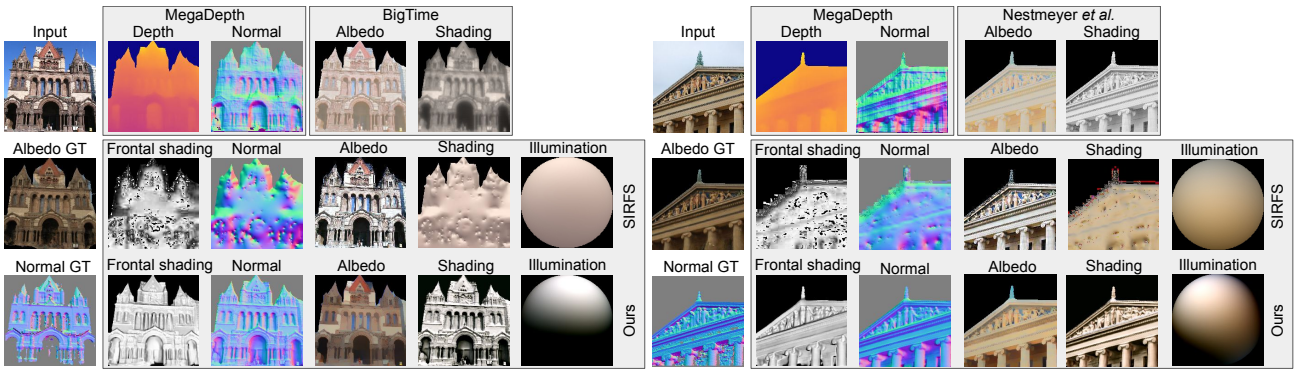


Figure 4.9: Inverse Rendering Results. We show our results with comparison to MegaDept [84], BigTime [83], Nestmeyer *et al*. [113] and SIRFS [7].

similar, defined as having correlation coefficient of intensity histograms close to 1). Finally, before inputting an image to our network, we detect and mask the sky region using PSPNet [169]. This is because the albedo map and normal map in sky area are meaningless and it severely influences illumination estimation. In total, we use 117,030 images from MegaDepth. We group these images by their scene reconstruction results, of which 193 outdoor scenes are selected for training and testing. In this 193 outdoor scenes, we select 40 scenes for testing and the remaining scenes for training. The training is stopped by either reaching 20 iterations or reaching a steady-state loss value.

4.5.2 Training strategy

We found that for convergence to a good solution it is important to include a pre-training phase. During this phase, the surface normals used for illumination estimation and for the appearance-based losses are the MVS normal maps. This means that the surface normal prediction decoder is only learning from the direct supervision loss, i.e. it is learning to replicate the MVS normals. After this initial phase, we switch to full self-supervision where the predicted appearance is computed entirely from estimated quantities. Note that this pre-training step is not using pseudo albedo supervisions.

4.6 Evaluation for InverseRenderNet

There are no existing benchmarks for inverse rendering in the wild. So, we evaluate our method on an intrinsic image benchmark and devise our own benchmark for inverse rendering. Finally, we show

Methods	Training data	WHDR
Nestmeyer [113] (CNN)	IIW	19.5
Zhou *et al*. [174]	IIW	19.9
Fan *et al*. [27]	IIW	14.5
DI [111]	Sintel+MIT	37.3
Shi *et al*. [135]	ShapeNet	59.4
Li *et al*. [83]	BigTime	20.3
Ours	MegaDepth	21.4

Table 4.1: Evaluation results on IIW benchmark using WHDR percentage (lower is better). The second column shows which dataset on which the networks were trained.

a relighting application.

4.6.1 Evaluation on IIW

The standard benchmark for intrinsic image decomposition is Intrinsic Images in the Wild [9] (IIW) which is almost exclusively indoor scenes. Since our training regime requires large multiview image datasets, we are restricted to using scene-tagged images crawled from the web, which are usually outdoors. In addition, our illumination model is learnt on outdoor, natural environments. For these reasons, we cannot perform training or fine-tuning on indoor benchmarks. Moreover, our network is not trained specifically for the task of intrinsic image estimation and our shading predictions are limited by the fact that we use an explicit local illumination model (so cannot predict cast shadows). Nevertheless, we test our network on this benchmark directly without fine-tuning. We follow the suggestion in [113] and rescale albedo predictions to the range $(0.5, 1)$ before evaluation. Quantitative results are shown in Table 4.1 and some qualitative visual comparison in Figure 4.8. Despite the limitations described above, we achieve the second best performance of the methods not trained on the IIW data.

4.6.2 Evaluation on MegaDepth

We evaluate inverse rendering using unobserved scenes from the MegaDepth dataset [84]. We evaluate normal estimation performance directly using the MVS geometry. We evaluate albedo estimation using a state-of-the-art multiview inverse rendering algorithm [68]. Given the output from their

Methods	Reflectances			Normals	
	MSE	LMSE	DSSIM	Mean	Median
Li *et al*. [84]	-	-	-	50.6	50.4
Godard *et al*. [38]	-	-	-	79.2	79.6
Nestmeyer *et al*. [113]	0.0204	0.0735	0.241	-	-
Li *et al*. [83]	0.0171	**0.0637**	0.208	-	-
SIRFS [7]	0.0383	0.222	0.270	50.6	48.5
Ours	**0.0170**	0.0718	**0.201**	**37.7**	**34.8**

Table 4.2: Quantitative inverse rendering results. Reflectance (albedo) errors are measured against multiview inverse rendering result [68] and normals against MVS results. Normal predictions are evaluated by mean angular error and median angular error.

Input Relit 1 Relit 2

Figure 4.10: Relighting results from predicted albedo and normal maps (see Figure 4.1, row 3). The novel lighting is a simple point light and shown in the upper left corner.

pipeline, we perform rasterisation to generate albedo ground truth for every input image. Note that both sources of "ground truth" here are themselves only estimations, e.g. the albedo ground truth contains ambient occlusion baked in. The colour balance of the estimated albedo is arbitrary, so we compute per-channel optimal scaling prior to computing errors. We use three metrics - MSE, LMSE and DSSIM, which are commonly used for evaluating albedo predictions. To evaluate normal predictions, we use angular errors. The correctness of illumination predictions could be inferred by the other two, so we do not perform explicit evaluations on it. The quantitative evaluations are shown in Table 4.2. For depth prediction methods, we first compute the optimal scaling onto the ground truth geometry, then differentiate to compute surface normals. These methods can only be evaluated on normal prediction. Intrinsic image methods can only be evaluated on albedo prediction. We can see that our network performs best in normal prediction and also the best in MSE and DSSIM. Qualitative example results can be seen in Figure 4.9.

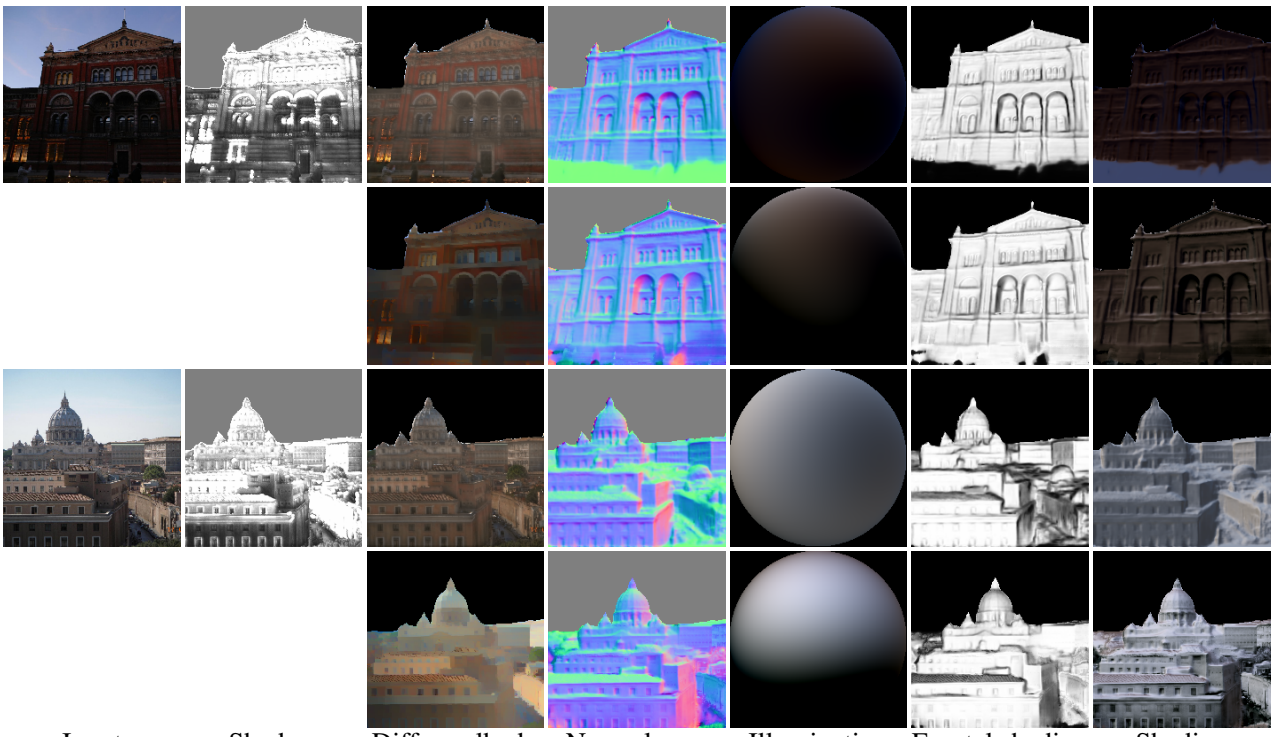


Figure 4.11: Sample output from InverserRenderNet++ (row 1,3), compared against ones from InverseRenderNet (row 2,4). From a single image (col. 1), InverseRenderNet++ can estimate shadow, albedo and normal maps and illumination (col. 2-5); re-rendering of our shape with (col. 6) frontal, white point source and with (col. 7) estimated spherical harmonic lighting.

4.6.3 Relighting

Finally, as an example application we show that our inverse rendering result is sufficiently stable for realistic relighting. A scene from Figure 4.1 is relit in Figure 4.10 with two novel illuminations. Both show realistic shading and overall colour balance.

4.7 Extensions in InverseRenderNet++

Although the InverseRenderNet can generate decent inverse rendering results, there are apparent limitations inherent to leveraged physical models, for example shadows are neglected by Lambertian model and albedo smoothness prior results in over-smoothed reconstructions. To address these issues, in this chapter we will discuss the possible improvements in applied reflection models, data used for network training and formulation for loss functions. Then we will show that InverseRenderNet++ trained with improved methods gains better performance compared with InverseRenderNet.

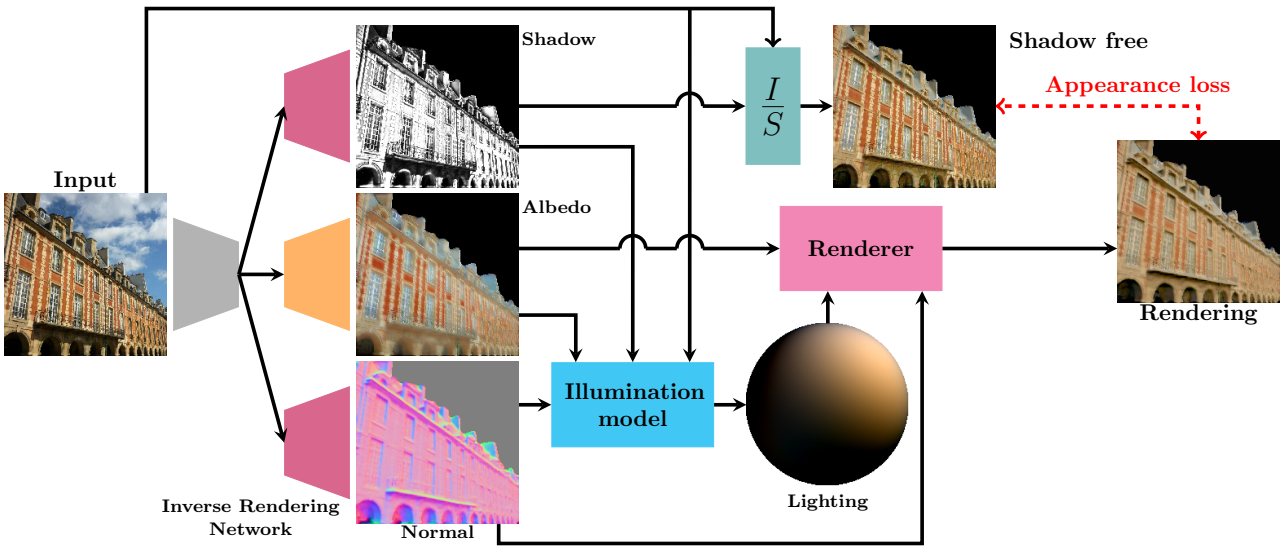


Figure 4.12: At inference time, our network regresses shadow, diffuse albedo and normal maps from a single, uncontrolled image. These are used to infer shadow free and shading only images from which we compute least squares optimal spherical harmonic lighting coefficients. At training time, we introduce self-supervision via an appearance loss computed using a differentiable renderer and the estimated quantities.

4.7.1 Reflectance model

The Lambertian model used in InverseRenderNet is known to be efficient but incapable to capture non-Lambertian effect, such as cast shadows, spatially varying illumination and specularities. For inverse rendering results of InverseRenderNet, the naive reflectance model results in those non-Lambertian effects being baked into one or both of the albedo and normal maps. Of these phenomena, the most severe are cast shadows. We introduce an additional term that acts multiplicatively on the appearance predicted by the local spherical harmonics model. Without appropriate constraint, the introduction of this additional channel could lead to trivial solutions. Hence, we constrain it in two ways. First, we restrict it to the range $[0, 1]$ so that it can only downscale appearance. Second, it is a scalar quantity acting equally on all colour channels. Together, these restrictions encourage this channel to explain cast shadows and we refer to it as a shadow map. However, note that we do not expect it to be a physically valid shadow map nor that it contains only shadows.

Under this model, RGB intensity can be computed as

$$\mathbf{i}(x,y) = \begin{bmatrix} i_r(x,y) \\ i_g(x,y) \\ i_b(x,y) \end{bmatrix} = \boldsymbol{\alpha}(x,y) \odot s(x,y)\mathbf{B}(\mathbf{n}(x,y))\mathbf{l}, \tag{4.16}$$

where $\odot$ is the Hadamard (element-wise) product, $\mathbf{l} \in \mathbb{R}^{27}$ contains the order 2, colour spherical harmonic colour illumination coefficients, $\boldsymbol{\alpha}(x,y) = [\alpha_r(x,y), \alpha_g(x,y), \alpha_b(x,y)]^T$ is the colour diffuse albedo, $s(x,y) \in [0,1]$ is the shadowing weight and the order 2 basis $\mathbf{B}(\mathbf{n}) \in \mathbb{R}^{3\times 27}$ is given by $\mathbf{B}(\mathbf{n}) = \mathbf{I}_3 \otimes \mathbf{b}(\mathbf{n})$ where $\otimes$ is the Kronecker product and $\mathbf{b}(\mathbf{n})$ is same with Equation 3.25. This appearance model neglects high frequency illumination effects and interreflections. However, we found that in practice this model works well for typical outdoor scenes.

4.7.2 Lighting inference

For changes in the lighting inference module, we first adjust the linear model with an additional shadow term. Then we reformulate the least squares solution for lighting parameters to avoid numerical instability. For example, the extremely dark or bright albedo prediction could lead to extreme shading inference result after element-wise division between image and albedo ($\mathbf{I} \oslash \mathbf{A}$), which might result in unstable lighting predictions. For an input image with K foreground pixels, we stack the K RGB values to form the vector $\mathbf{i}_{\text{obs}} \in \mathbb{R}^{3K}$. We assume that nonlinear gamma has been applied with a fixed $\gamma = 2.2$. Here we apply the inverse gamma to observation image to retain a linear relationship between inverse-gamma-corrected observation and our estimations. This linear relationship is the key to efficiently solve lighting inference problem. We invert the nonlinear gamma and equate the observed intensities with our model from Equation (3.24) extended to the whole image:

$$\begin{aligned} \mathbf{i}_{\text{obs}}^{\gamma} &= [i_r^1, \ldots, i_r^K, i_g^1, \ldots, i_g^K, i_b^1, \ldots, i_b^K]^T \\ &= \boldsymbol{\alpha} \odot (\mathbf{1}_{3\times 1} \otimes \mathbf{s}) \odot \mathbf{B}\mathbf{l} \end{aligned} \tag{4.17}$$

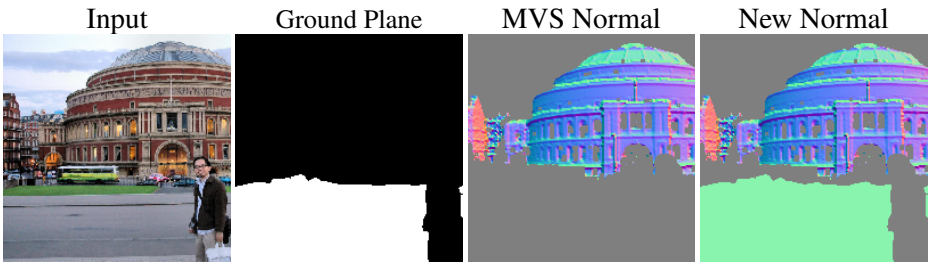


Figure 4.13: The mask for ground plane is obtained by PSPNet [169]. We then define the normal for masked ground plane by the normal of fitted camera plane.

where each row of $\mathbf{B} \in \mathbb{R}^{3K \times 27}$ contains the spherical harmonic basis for the estimated surface normal at one pixel and the foreground pixels from the estimated albedo and shadow maps are stacked to form $\boldsymbol{\alpha} \in \mathbb{R}^{3K}$ and $\mathbf{s} \in \mathbb{R}^{K}$ respectively. We can now rearrange Equation (4.17) into linear least squares form with respect to the lighting vector:

$$\mathbf{i}^{\gamma}_{\text{obs}} = [\mathbf{1} \otimes (\boldsymbol{\alpha} \odot (\mathbf{1}_{3\times 1} \otimes \mathbf{s})) \odot \mathbf{B}]\, \mathbf{l} = \mathbf{A}\mathbf{l}. \tag{4.18}$$

We can now solve for the spherical harmonic illumination coefficients in a least squares sense over all foreground pixels. This can be done using any method, so long as the computation is differentiable such that losses dependent on the estimated illumination can have their gradients backpropagated into the inverse rendering network. We solve using the pseudoinverse of $\mathbf{A}$, i.e. $\mathbf{l}^{*} = \mathbf{A}^{+}\mathbf{i}^{\gamma}_{\text{obs}}$. Note that the pseudoinverse $\mathbf{A}^{+}$ has a closed form derivative [42]. Figure 4.12 shows the inferred normal, albedo and shadow maps, and a visualisation of the least squares estimated lighting. Just like InverseRenderNet, we combine this new lighting inference model with the illumination statistical model to construct the illumination model as shown in the Figure 4.12. Instead of using Equation 3.28, the new illumination model uses the combination of Equation 4.18 and our lighting model Equation 4.6.

4.7.3 Data preprocessing

Of losses used in training InverseRenderNet, direct normal prediction loss only relies on sparsely reconstructed geometry, especially on ground plane areas. Because the MVS models always fail to reconstruct the ground plane. We find that this significantly disrupts training such that our network does not learn to predict good shape or albedo estimates in the ground plane region. For this reason,

we propose to preprocess the MVS normal maps to replace ground plane normals with an assumed ground plane normal direction. To this end, we detect ground plane pixels using PSPNet [169] and inpaint normal direction to detected pixels in the normal map. Although the true ground plane direction is unknown, a reasonable estimate can be made from the MVS reconstruction. We assume that the camera positions are located approximately a fixed height above the ground and fit a plane to their positions using principal components analysis (PCA). The normal to this plane defines the ground plane normal which we rotate into camera coordinates and use for inpainting. A sample result is shown in Figure 4.13.

Again, we perform network training primarily using MegaDepth [84] as training data. Apart from this dataset, for parts of our evaluation, we finetune using additional datasets (described in Section 4.8). These are preprocessed in the same way as MegaDepth.

4.7.4 Self-supervision

With additional input channel of shadow, we compute the self-reconstruction image by Equation (4.16). Under this new reflectance model, we found that performance was significantly improved by computing appearance error in a shadow-free space. To do so, we divide out the estimated shadow map from the linearised observed image and clamp to one (avoiding numerical instabilities caused by very small shadow values):

$$\mathbf{i}_{\text{SF}} = \min\left[1, \mathbf{i}_{\text{obs}}^{\gamma} \oslash (\mathbf{1}_{3\times 1} \otimes \mathbf{s})\right], \tag{4.19}$$

where $\oslash$ is Hadamard (element-wise) division. We compare this shadow free image to the image predicted using only the local spherical harmonic model and our estimated illumination and albedo and normal maps. In addition to the perceptual loss employed in InverseRenderNet (LAB appearance loss), we provide another perceptually meaningful loss computed by L2 loss in VGG feature space.

$$\ell_{\text{appearance}} = \varepsilon(\boldsymbol{\alpha} \odot \mathbf{Bl}, \mathbf{i}_{\text{SF}}), \tag{4.20}$$

where

$$\varepsilon(\mathbf{x}, \mathbf{y}) = w_{\text{VGG}} \left\|\text{VGG}(\mathbf{x}) - \text{VGG}(\mathbf{y})\right\|_2 + w_{\text{LAB}} \left\|\text{LAB}(\mathbf{x}) - \text{LAB}(\mathbf{y})\right\|_2, \tag{4.21}$$

where VGG computes features from the first two convolution blocks of a pre-trained VGG network [140] and LAB transforms to the LAB colour space. We include VGG loss into our perceptual loss according to the findings proposed by Johnson *et al*. [58]. Following the conclusion drawn from Johnson, VGG-based perceptual measurement could make significant contribution in image reconstruction tasks. In addition, we choose to use the first two convolution blocks out of the full model, which means only the low-level features are used to compare our reconstruction with the target. In contrast to high-level semantic features, such low-level features capture structural patterns, which typically lead the measurement of perceptual distance. We mask these losses pixel-wise using the sky mask, using appropriate downsampling of the mask within the VGG layers. We set the weights as $w_{\text{VGG}} = 2.5$ and $w_{\text{LAB}} = 0.5$ in all experiments.

4.7.5 Multiview stereo supervision

The training losses derived from multiview stereo cues are modified and applied in InverseRenderNet++, which helps improve the albedo consistency quality and solve the limitation of over-smoothed results. Whilst using the same direct normal loss, we update the formulations for albedo consistency loss and cross-rendering loss as follow.

Albedo consistency loss

Similar to InverseRenderNet, albedo consistency loss is formed by the error between aligned albedo predictions using cross-projection. Besides, we employ the same perceptual appearance resemblance measurements, where the weights w_{VGG} and w_{LAB} are set the same as in (4.20). The albedo consistency loss is visualised by the blue arrows in Figure 4.14.

Cross-rendering loss

Cross-rendering loss was proposed to remedy a deficiency in using albedo consistency alone in InverseRenderNet. However, cross-projecting albedos while computing cross-rendering loss and albedo

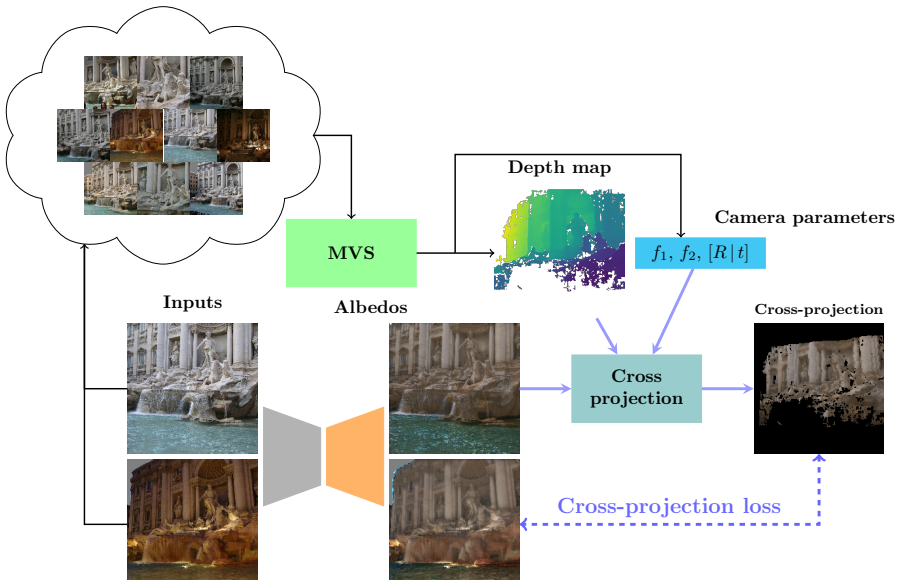


Figure 4.14: The albedo consistency loss is formulated by cross projections. Given the depth map and camera parameters from performing MVS over image collections, albedo prediction from one view can be cross projected to another view. For each image, we compute the albedo consistency loss by measuring the difference between albedo prediction and cross-projected albedo prediction from other views within each batch.

consistency loss results in blurry albedo maps. This is because the consistency constraint encourages blurred but consistent albedo maps. Here we propose a variant cross-rendering loss that avoids requiring cross projection of albedo predictions, which can help the training stabilise albedo consistency and alleviate blurry problem originated from cross-projecting albedo maps.

Instead, we cross project the images in the dataset at their original high resolution, using the high resolution depth maps. We define the cross projection of image j into image i using the original high resolution image, $\mathbf{i}_j^{\text{HR}}$, in view j as:

$$\mathbf{i}_{j\to i} = \text{downsample}(\text{proj}_{j\to i}(\mathbf{i}_j^{\text{HR}})), \tag{4.22}$$

where the downsample function downsamples to network input resolution. Note that this cross projection need only be done once as preprocessing step on the training dataset.

We then mix the albedo and normal predictions from image i with the lighting and cross projected shadow map from image j and compare it against the high resolution cross projection from image j.

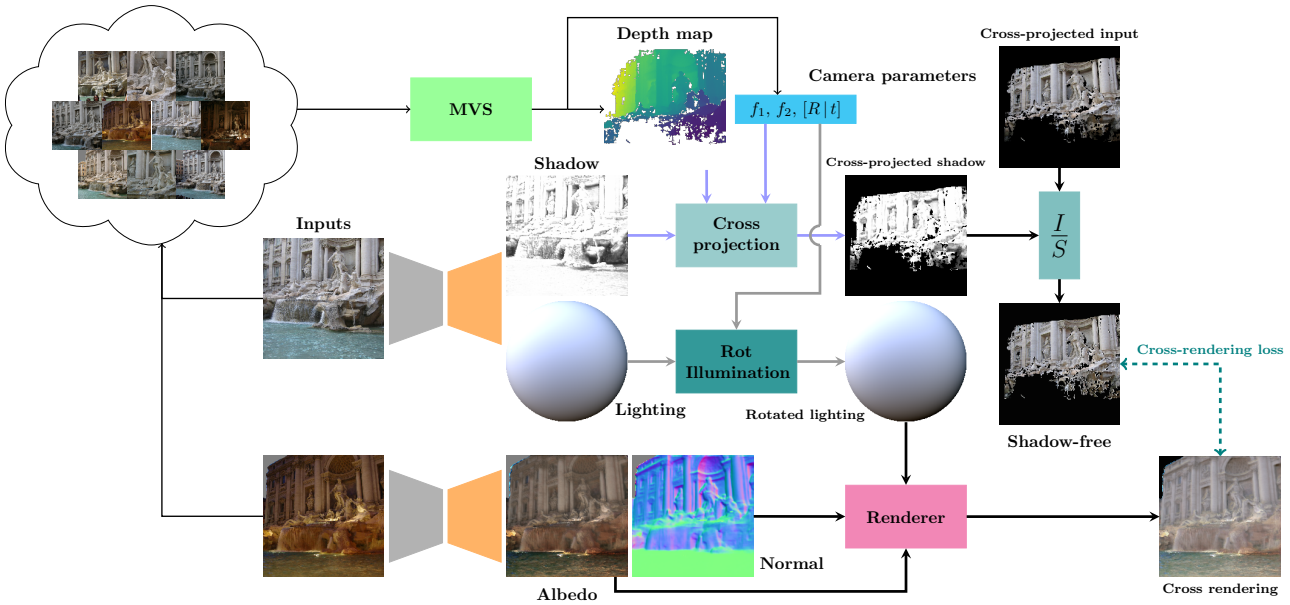


Figure 4.15: Example for cross rendering between two images. Cross rendering image is generated from relighting albedo and normal predictions from one involved image and lighting prediction from the other by Lambertian model. Before applying lighting, we rotate the lighting to align it with new view given relative camera poses. The rendering is generated without shadow, so the shadow is removed from ground truth relighting image. Both shadow and ground truth relighting image are cross projected from the view where new lighting taken from.

Again, we do so in a shadow free space, so first define the cross-projected shadow free image:

$$\mathbf{i}^{\text{SF}}_{j\to i} = \min\left[1, \mathbf{i}_{j\to i} \oslash \left(\mathbf{1}_{3\times 1} \otimes \text{proj}_{j\to i}(\mathbf{s}_j)\right)\right]. \tag{4.23}$$

We then compute the cross-rendering loss as:

$$\ell_{\text{cross-rend}} = \varepsilon(\mathbf{i}^{\text{SF}}_{j\to i}, \boldsymbol{\alpha}_i \odot \mathbf{B}_i \mathbf{R}_{j\to i} \mathbf{l}_j), \tag{4.24}$$

where $\mathbf{B}_i$ is the spherical harmonic basis computed from the normal map estimated from image i, $\mathbf{l}_j$ is the spherical harmonic lighting coefficients estimated in image j and $\mathbf{R}_{j\to i}$ rotates the coefficients from the camera coordinate system in image j to image i. The cross-rendering loss is essential for stability and alleviates issues of low resolution cross projection. We visualise the process of computing this loss in Figure 4.15.

Method	Reflectances		Normals	
	MSE	LMSE	Mean	Median
MegaDepth [84]	-	-	43.6	43.0
Godard *et al*. [38]	-	-	82.4	82.1
Nestmeyer *et al*. [113]	0.0149	0.0169	-	-
BigTime [83]	0.0116	0.0135	-	-
SIRFS [7]	**0.0070**	0.0275	50.6	48.5
InverseRenderNet	0.0112	0.0128	40.0	38.0
InverseRenderNet++	0.0093	**0.0111**	**31.2**	**30.0**

Table 4.3: Quantitative inverse rendering results. Reflectance (albedo) errors are measured against multiview inverse rendering result [68] with optimal per-channel scaling applied and normals against MVS results.

Method	Reconstruction		Consistency	
	MD [84]	BT [83]	MD [84]	BT [83]
Nestmeyer *et al*. [113]	0.1003	0.0333	0.0140	**0.0067**
BigTime [83]	0.0372	-	0.0089	-
InverseRenderNet	0.0158	0.0124	0.0168	0.0118
InverseRenderNet++	**0.0082**	**0.0082**	**0.0082**	0.0072

Table 4.4: Reconstruction error and reflectance consistency for BigTime dataset [83]. MSE is used for computing reconstruction errors, and MSE with optimal scaling is used for computing albedo consistency.

4.7.6 Training loss

We train InverseRenderNet++ to minimise:

$$\ell = w_1 \ell_{\text{appearance}} + w_2 \ell_{\text{NM}} + w_3 \ell_{\text{albedo}} + w_4 \ell_{\text{cross-rend}} + w_5 \ell_{\text{lighting}}, \tag{4.25}$$

where the weights are set as $w_1 = 0.1$, $w_2 = 1.0$, $w_3 = 0.1$, $w_4 = 0.1$ and $w_5 = 0.005$. Compared with loss function defined in InverseRenderNet (4.15), albedo priors $\ell_{\text{albedo-smooth}}$ and $\ell_{\text{albedo-pseudoSup}}$ are relaxed from loss function. The albedo priors are important for shadow bake-in disambiguations but results in albedo maps without enough sharpness. In InverseRenderNet++, we refrain from those priors to capture texture details in albedo maps whilst shadow bake-in ambiguity is resolved by incorporating shadow maps.

4.8 Evaluation for InverseRenderNet++

There are no existing benchmarks for inverse rendering of outdoor scenes, partly because there is no existing method for satisfactorily estimating scene-level geometry, reflectance and lighting for in-the-wild, outdoor scenes. For this reason, we developed our own outdoor benchmark based on the MegaDepth dataset [84]. We augment this with three further benchmarks that evaluate performance of a subset of the network outputs. First, we use the BigTime timelapse dataset [83] to evaluate albedo consistency under varying illumination. Second, we evaluate on the related task of intrinsic image decomposition using an indoor benchmark [9] which demonstrates the generalisation ability of our network. Third, we better quantify our normal prediction using the DIODE benchmark [153], which contains high quality outdoor depth and normal maps along with corresponding images. Then we explicitly evaluate our illumination prediction using our own benchmark dataset, whose details and specifications are demonstrated in the next chapter. This benchmark dataset captures multi-view, multi-illumination and HDR images and their environment maps, thus allowing us to qualitatively and quantitatively qualify our illumination estimation results. Finally, we show an ablation study across different configurations of proposed losses.

4.8.1 Evaluation on MegaDepth

We split the MegaDepth dataset [84] into training and testing data, and evaluate our performance on testing data. We evaluate the inverse rendering results through four aspects. Firstly, the normal estimation performance can be directly compared against the MVS geometry. We quantify performance using angular error in degrees. Second, like InverseRenderNet we evaluate albedo estimation against the output a state-of-the-art multiview inverse rendering algorithm [68]. Here, we use two metrics - MSE (mean squared error) and LMSE (local mean squared error). Here, LMSE is the performed by repeatedly running MSE on local crops of the image. The key difference between MSE and LMSE is that we compute and apply the optimal scale based on each crop when measuring LMSE, but an uniform scale is used over the whole image when computing MSE. The quantitative evaluations are shown in Table 4.3. As for depth prediction methods and Intrinsic image methods, we adapt the same

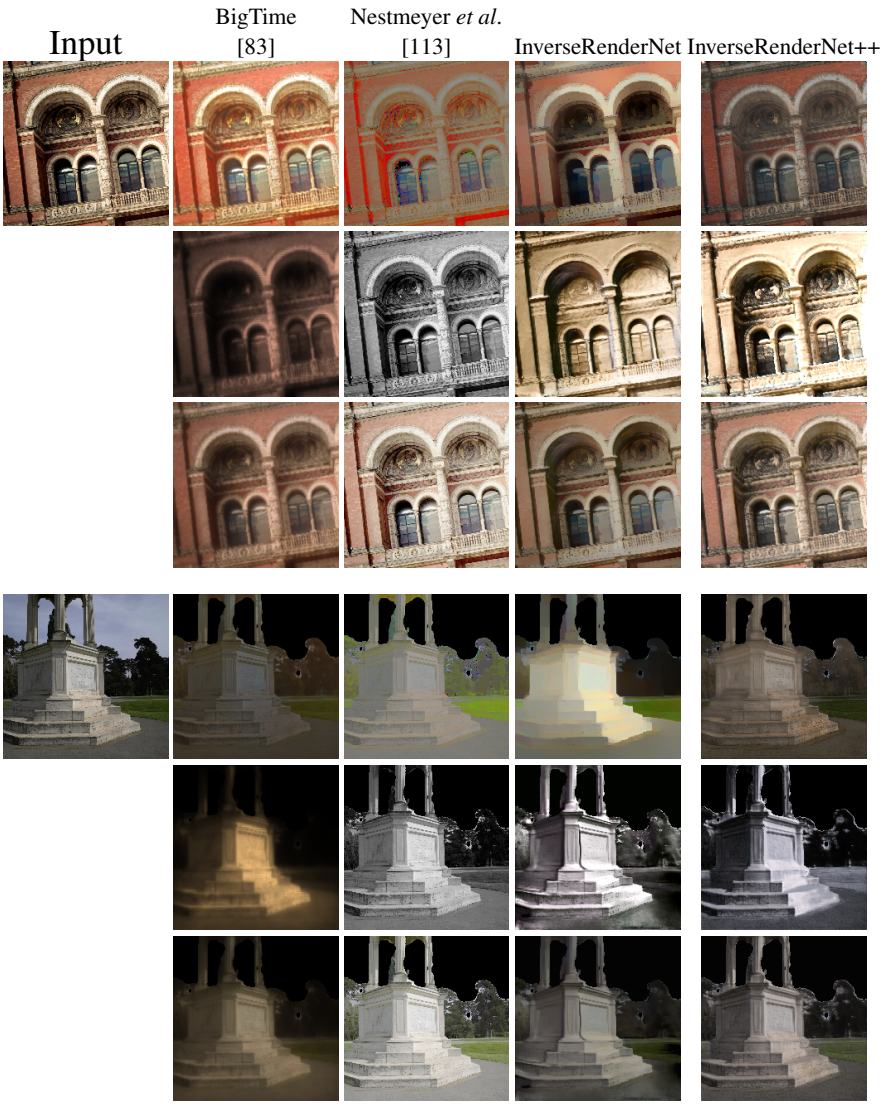


Figure 4.16: Qualitative results for reconstruction. The first and fourth rows are albedo predictions, and second and fifth rows are shading predictions from labelled methods. The reconstruction results for each method is composed by albedo and shading and is shown on third and last rows.

evaluation metrics used in InverseRenderNet. We can see that our InverseRenderNet++ performs best in both albedo and normal predictions. Qualitative results can be found in Figure 4.18. Note that, relative to the inverse rendering methods, our result is able to explain cast shadows in the shadow map, avoiding baking them into the albedo map.

Except the direct evaluations against the ground truth inverse rendering results, we propose two additional metrics which are reconstruction accuracy and albedo consistency. The introduced two metrics reflects how well our inverse rendering results explain the input image, and whether the physical invariants are consistent across the scene. They indirectly evaluate the correctness of our inverse rendering result. Quantitative results are shown in Table 4.4 Here we measure albedo consistency

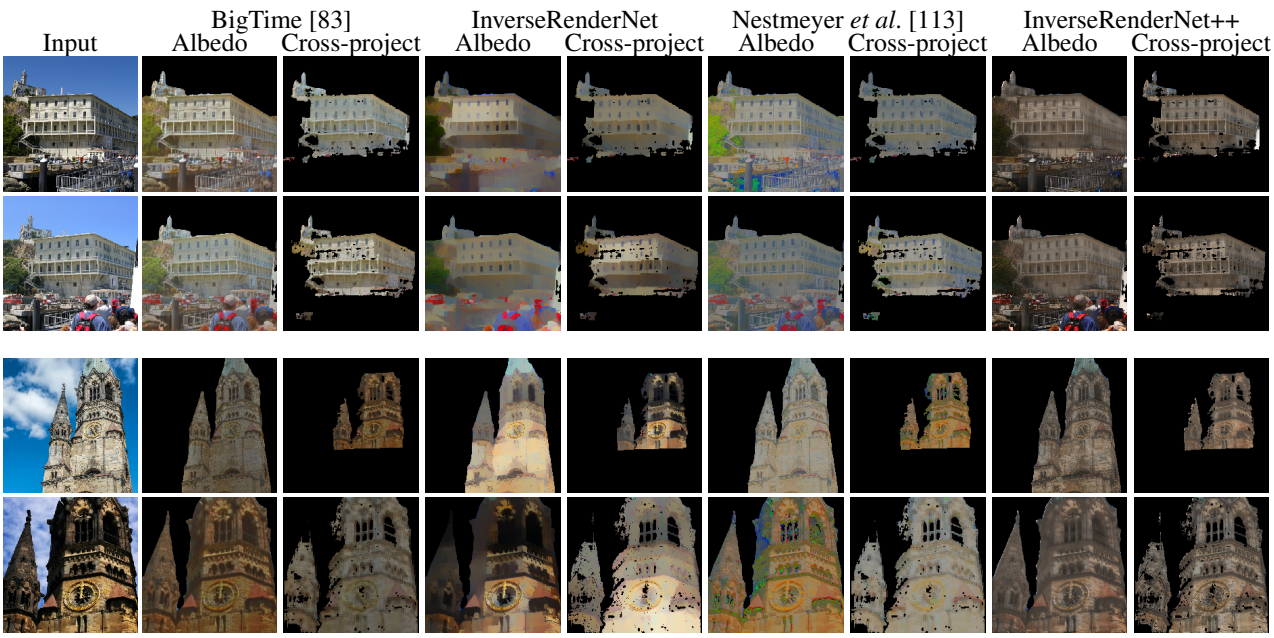


Figure 4.17: Qualitative results for albedo consistency. For each consecutive pair of rows, we show results for two overlapping images of the same scene. Albedo predictions are shown in col. 2, 4, 6 and 8 which are then cross-projected to the viewpoint of the other image in the pair in col. 3, 5, 7 and 9. Results shown for col. 2-3: BigTime [83], col. 4-5: InverseRenderNet, col. 6-7: Nestmeyer *et al.* [113], col. 8-9: InverseRenderNet++.

by cross-projecting albedo predictions to overlapping images, followed by calculating the difference between the now-aligned albedo predictions. We show qualitative results in Figure 4.16 and 4.17. Relative to InverseRenderNet, our new albedo maps preserve high frequency detail since we do not use a smoothness prior. Note also that we are able to extract albedo maps with consistent colour from images with very different illumination colour (Figure 4.17).

4.8.2 Evaluation on BigTime

Time-lapse data allows us to use the same reconstruction and consistency metrics without the need for cross-projection. We do so on the BigTime time-lapse dataset [83]. The quantitative results are shown in Table 4.4. Note that the BigTime [83] intrinsic image method is excluded from the comparison and not reported in this table, because the method itself is entirely trained based on the dataset. All other methods including our proposed work are trained without fine-tuning on this dataset. To fairly perform the evaluations, we select 15 scenes out of nearly 200 scenes which contain large portion of indoor scenes and some outdoor scenes from the dataset. Some example results are shown in Figure 4.19. In the last example in particular, note our ability to avoid baking cast shadows into the albedo map.

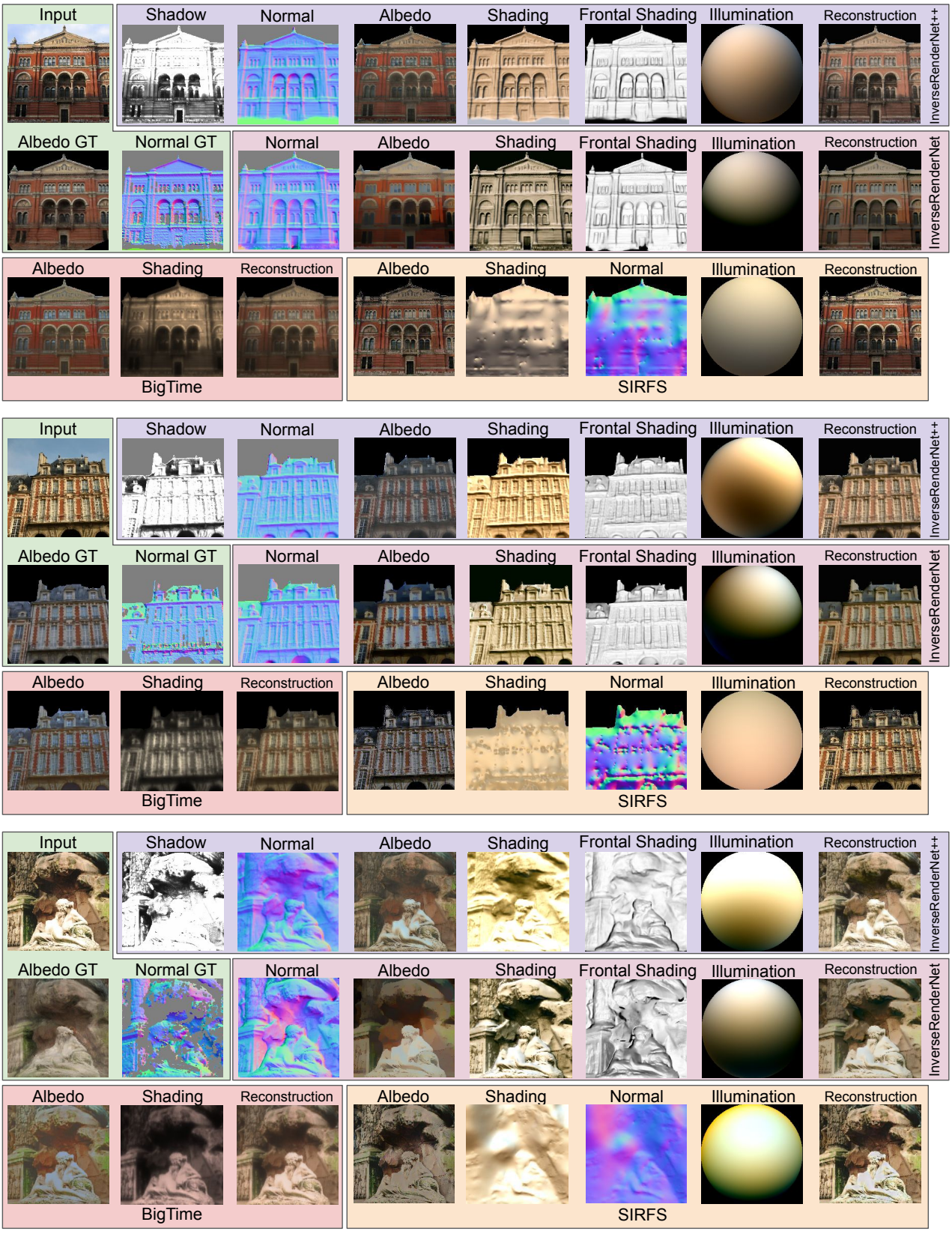


Figure 4.18: Qualitative results for our inverse rendering benchmark. We show comparison against InverseRenderNet, BigTime [83] and SIRFS [7].

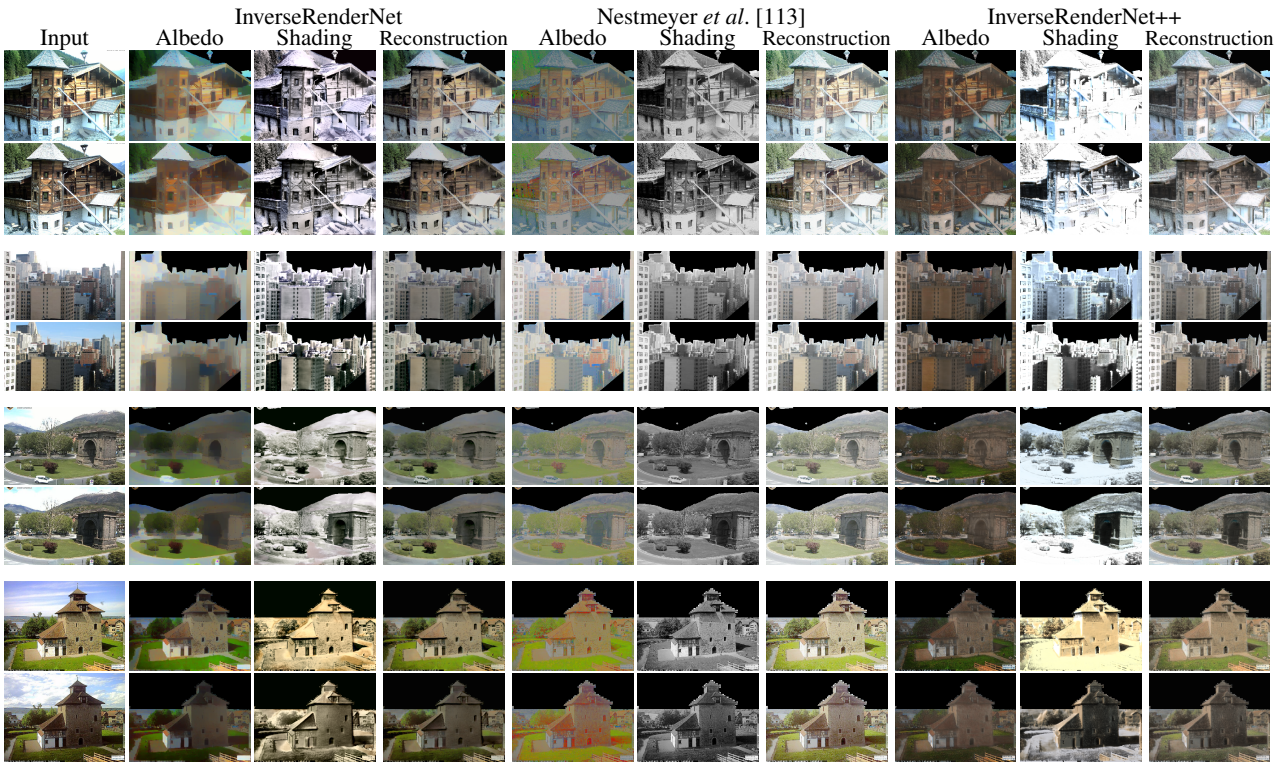


Figure 4.19: Qualitative results on BigTime data [83]. Each consecutive pair of rows shows results for two different frames from a time-lapse sequence. Col. 1: input, col. 2-4: InverseRenderNet, col. 5-7: Nestmeyer *et al*. [113], col. 8-10: ours. For each method we show albedo, shading and reconstruction.

4.8.3 Evaluation on IIW

We perform evaluations on IIW [9] for indoor scenes. As explained above, we cannot perform training or fine-tuning on this indoor benchmark. However, we perform fine-tuning using two other datasets to improve performance on the intrinsic image task on indoor scenes. Specifically, we first add the BigTime dataset [83] for which we can compute self-rendering, albedo consistency and cross-rendering losses without the need for any cross projection. Second, we sidestep the lack of surface normal labels in BigTime [83] by introducing the indoor normal supervisions from the NYU dataset [139]. To train our network with the mixture of all three sources of data, each training iteration is separated into two sub-steps. Firstly, the network is trained by MegaDepth [84] and NYU [139] to learn indoor self-reconstruction and normal prediction. Secondly, by freezing the normal prediction decoder, only shadow and albedo prediction decoders are trained by BigTime [83]. We show qualitative results in Figure 4.20 and quantitative results in Table 4.5. Note that our performance is very close to the state-of-the-art for methods not fine-tuned on the IIW training set, despite the fact that our inverse rendering solution is more constrained than intrinsic image methods (we must explain shading in terms of ge-

Methods	Training data	WHDR
Nestmeyer [113]	IIW	19.5
Li *et al*. [82]	IIW	17.5
Sengupta *et al*. [131]	IIW	16.7
Narihira *et al*. [111]	Sintel+MIT	37.3
Shi *et al*. [135]	ShapeNet	59.4
BigTime [83]	BigTime	20.3
InverseRenderNet	MegaDepth	21.4
InverseRenderNet++	MD+BT+NYU	21.1

Table 4.5: Quantitative results on IIW benchmark using WHDR percentage (lower is better). The second column shows which dataset on which the networks were trained. The upper block is trained on IIW training set.

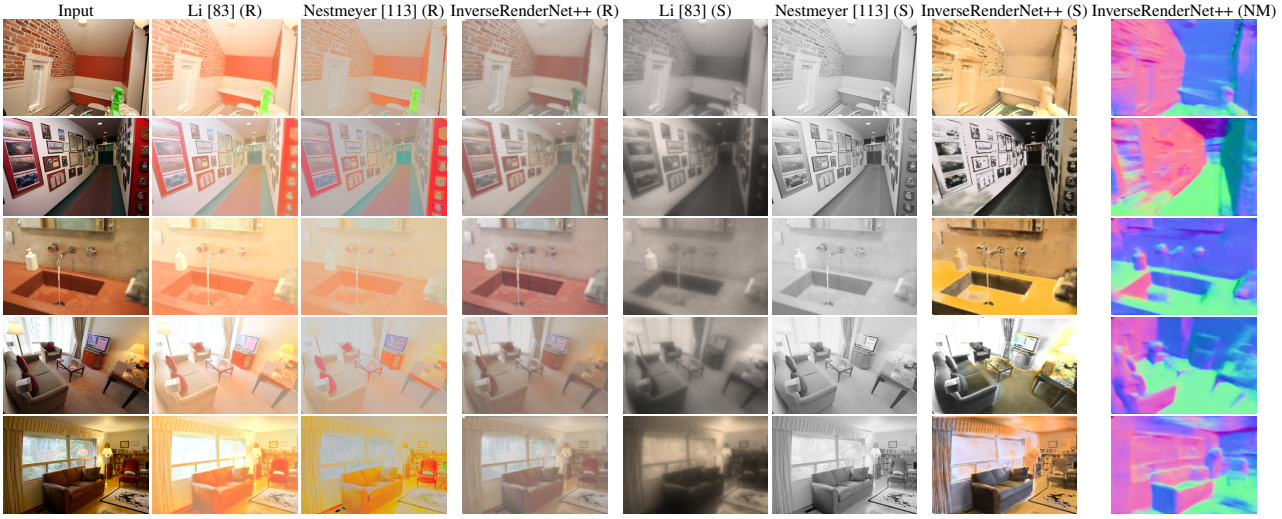


Figure 4.20: Qualitative intrinsic image results for IIW benchmark. Col. 2-4: reflectance predictions from [83], [113] and ours. Col. 5-7: corresponding shading predictions. Col. 8: surface normal prediction for our method.

ometry, lighting and shadows whereas intrinsic image methods are free to compute arbitrary shading maps).

4.8.4 Evaluation on DIODE

DIODE [153] is a very recent geometry estimation benchmark comprising images and registered depth/normal maps acquired with a Lidar scanner. The benchmark is divided into indoor and outdoor scenes and we evaluate only on the outdoor scene test set. To date, the only published result for normal map estimation is the method of Eigen *et al*. [25] against which we quantitatively compare in Table 4.6. In addition, we evaluate the performance of MegaDepth [84] on this benchmark dataset as

Methods	Mean	Median
Eigen *et al*. [25]	31.9	24.7
Li *et al*. [84]	41.7	32.9
ours	**23.9**	**15.5**

Table 4.6: Quantitative surface normal prediction errors on the DIODE dataset [153]. We show mean and median angular errors in degrees for the outdoor test set.

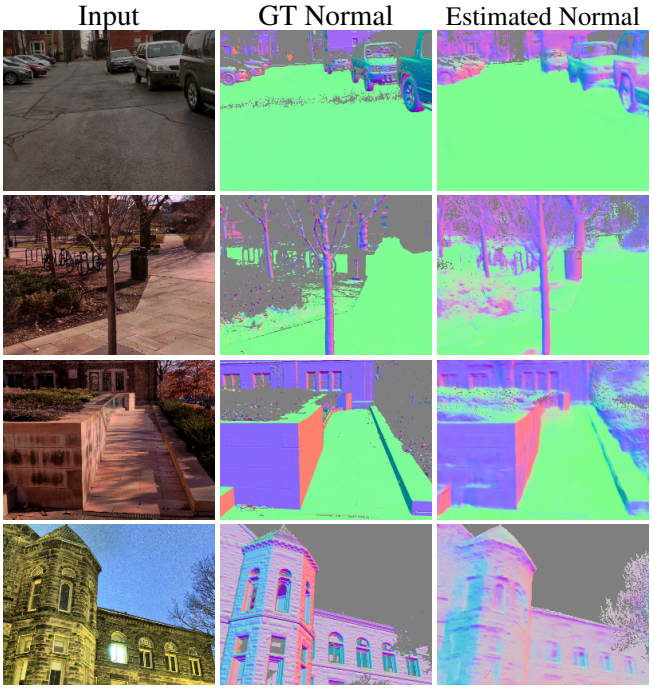


Figure 4.21: Qualitative results for surface normal prediction on DIODE dataset [153]. Left to right: input, ground truth normal map, estimated normal map. The grey pixels in the normal map are undefined values. In the second column, these grey pixels are missing data that sensors fail to capture, and in the third column, such grey pixels are segmented as sky region.

another comparison method. Because it can only predict depth map with arbitrary scale, we compute the normal estimate from depth prediction by applying camera intrinsics and the optimal scale which is obtained by comparing the median of ground truth depth and median of depth prediction. We show qualitative example results on this dataset in Figure 4.21.

4.8.5 Illumination estimation

To evaluate the accuracy of illumination estimation of our proposed methods, we show that our InverseRenderNet and InverseRenderNet++ generalise to the benchmark dataset proposed in the next chapter (Chapter 5) without fine-tuning. The benchmark dataset consists of images capturing a monument under six different illumination conditions from multiple viewpoints. Along with this multi-view

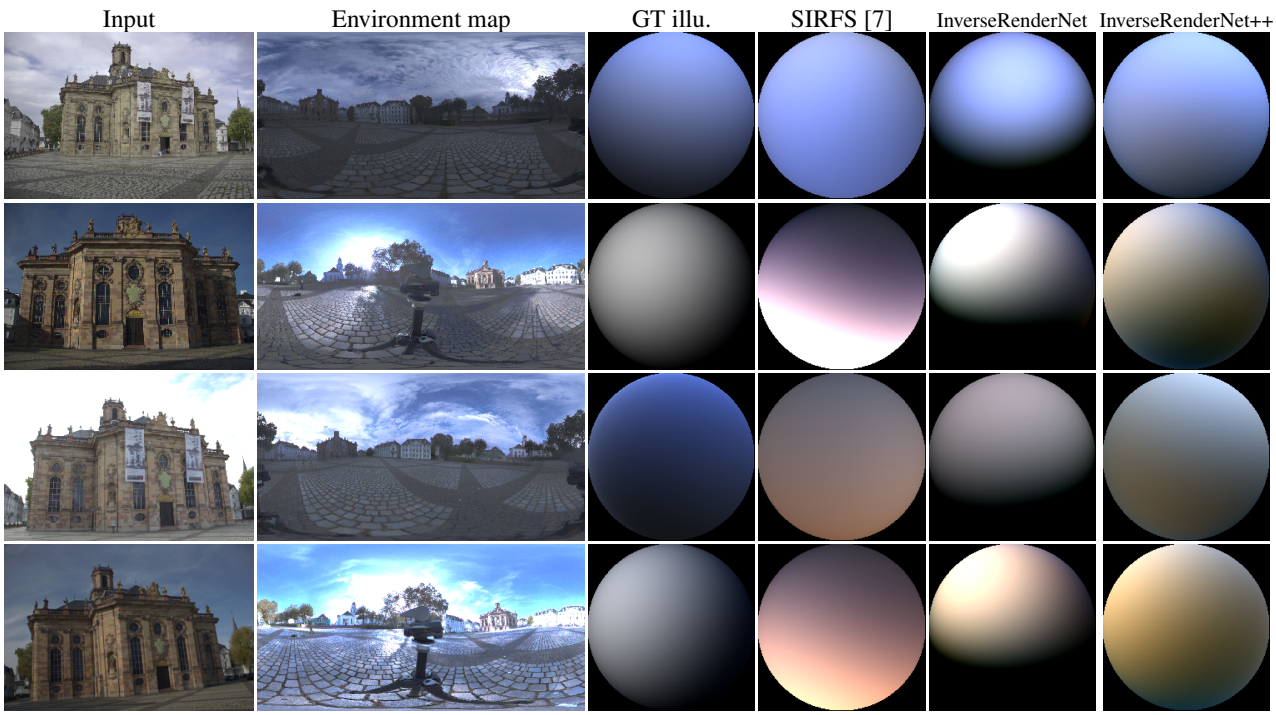


Figure 4.22: Qualitative evaluation for illumination estimation. Captured environment maps are shown in Column 2, which are projected to spherical harmonics and used for relighting hemisphere to generate ground truth illumination estimations as demonstrated in Column 3. Illumination estimates from SIRFS [7], InverseRenderNet and InverseRenderNet++ method are shown in Column 4-6. The first two rows are per-colour scaled, and last two are global scaled.

and multi-illumination image collection, the six HDR illumination environment maps and necessary alignments between image and environment map pairs are also provided. In order to evaluate our performance on lighting estimation, we feed each image into our network and compare our lighting prediction with ground truth. Since our network can only infer the lighting represented by spherical harmonics, we project the ground truth environment map onto order 2 spherical harmonics. To ensure a fair evaluation the comparison regarding backside lighting is omitted, so we compare only the front side by measuring the distance between hemispheres lit by our lighting prediction and by ground truth spherical harmonic lighting. We show quantitative evaluations in Table 4.7 and qualitative evaluations in Figure 4.22. Since there is an overall scale ambiguity between albedo and lighting, in quantitative evaluations, two scaling methods are applied. The global scaling metric is computed by applying an optimal global intensity scale prior to measuring errors, and per-colour scaling method is conducted by applying an optimal scaling to each colour channel. Accordingly, the qualitative results for these two scaling methods are shown in Figure 4.22.

Methods	Global scale	Per-colour scale
SIRFS [7]	0.100	0.089
InverseRenderNet	0.050	0.041
InverseRenderNet++	**0.038**	**0.033**

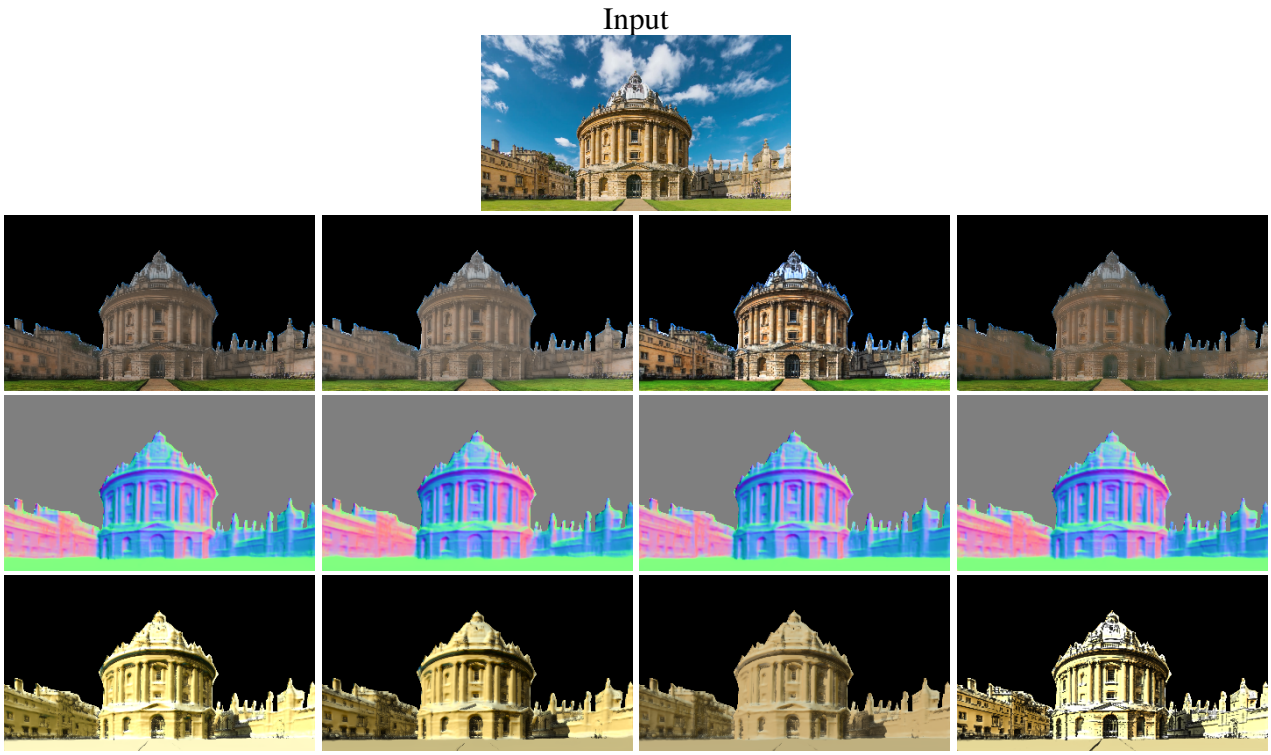
Table 4.7: Quantitative results for illumination estimation. We show global-scale and per-colour-scale MSE errors.

Methods	MD reflectance (MSE)	IIW	Illumination (global scale)
Full	**0.0093**	**21.1**	**0.038**
w/o VGG	0.0095	21.6	0.053
w/o cross-projection loss	0.0153	21.6	0.052
w/o cross-rendering loss	0.0101	21.4	0.052

Table 4.8: Quantitative comparison for ablation study. Here, we select some representative metrics used in Section 4.8.1, 4.8.3 and 4.8.5, which are MSE-based reflectance errors on MegaDepth test data (Column 1), IIW benchmark score (Column 2), and global-scaled Illumination estimation errors (Column 3).

4.8.6 Ablation study

In this section, we seek insight on some key design choices by comparing the performance of our proposed network and ablated models on evaluation datasets used above. We report the comparison between our full model and ablated model trained without VGG loss, without cross-projection loss and without cross-rendering loss. The quantitative results are summarised in Table 4.8. Our proposed model outperforms the ablated methods on all metrics across all datasets. Figure 4.23 visualises the qualitative comparison on different training setups. Here, without cross-projection loss the network cannot remove shadows from albedo prediction. Without cross-rendering loss the network can only predict over-smoothed albedo. Without VGG loss, albedo prediction looks 'bleached', since shading and shadow predictions capture too much darkness - shading predictions of this ablated variant shows less brightness contrast and darker intensity than that of our full model. While training without VGG loss results in larger errors, we found that finetuning the weights w_1, w_3 and w_4 of losses for specific tasks leads to better performance than our full model. However, it cannot providing consistently the best performance across all metrics with a uniform weights setting.



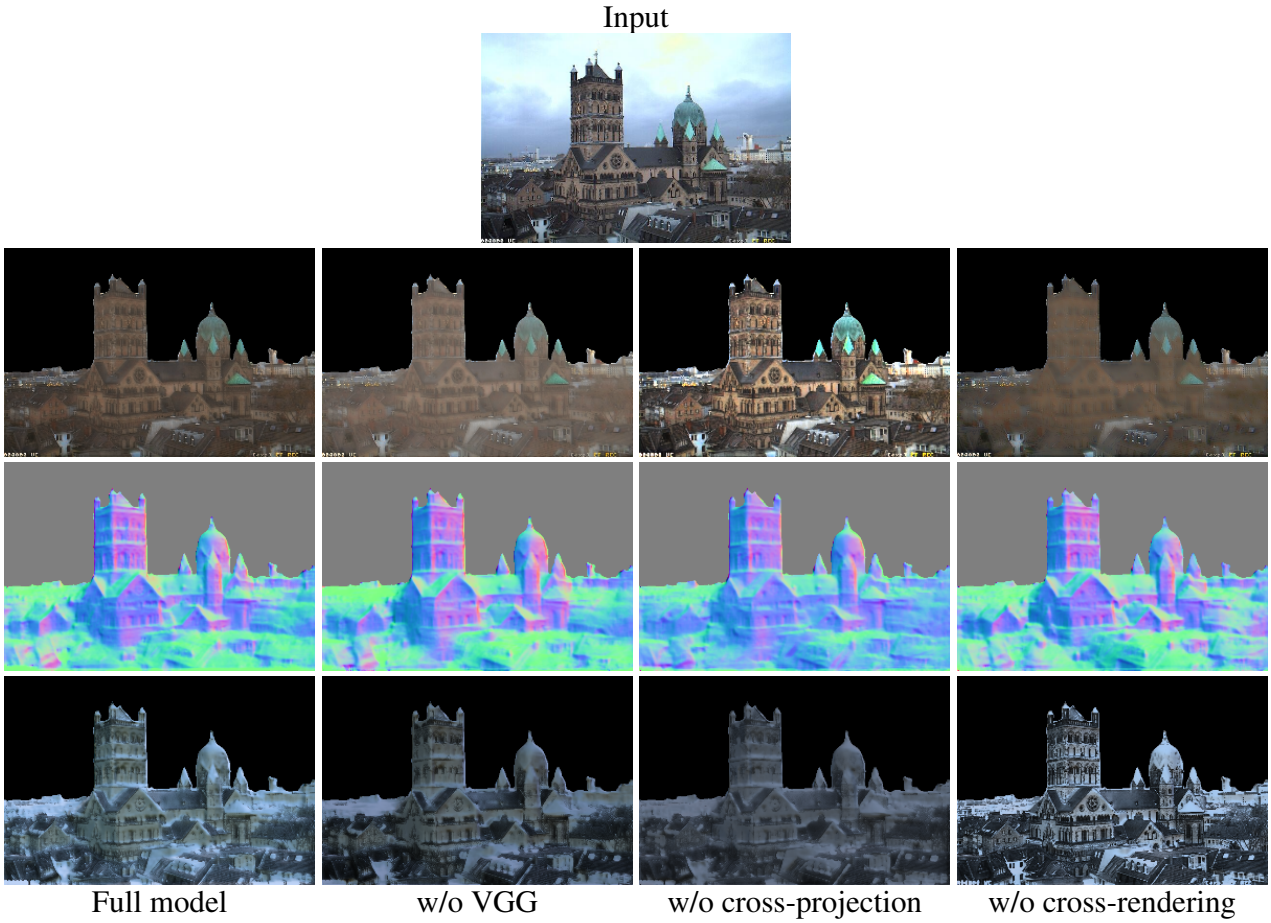


Figure 4.23: Qualitative comparison for ablation study. Row 2: albedo, Row 3: surface normals, Row 4: shading. Ablation conditions are shown column-wise.

4.9 Conclusion

In this chapter, we showed how to use a model-based decoder to learn an inverse rendering network in a self-supervised manner. The simple yet effective Lambertian model and multiview stereo model motivates us to supervise the inverse rendering network by self-reconstruction and multiview loss. Our efficient linear Lambertian model also facilitates to perform the lighting estimates such that a separate lighting estimation network is not needed in our pipeline. We have shown for the first time that the task of inverse rendering can be learnt from real world images in uncontrolled conditions. Our results show that "shape-from-shading" in the wild is possible, and our learnt InverseRenderNet/InverseRenderNet++ significantly outperforms classical methods. We are also competitive against state-of-the-art methods trained specifically for related subtasks.

Following the experiments conducted above, we can conclude that the model-based decoder is essential for achieving self-supervised training. The rendering model has the ability to regularise the solution space for estimated unknowns, hence resulting better performance than purely supervised training. While the network training can outperform other by integrating physical model, it also inherits the potential drawbacks from the model. First, the non-Lambertian effects, like specularity and inter-reflection, are baked into inverse rendering results. The misalignment raised by MVS reconstruction gives rise to oversmoothed albedo estimates. The illumination representation employed in the InverseRenderNet is spherical harmonic lighting, which can hardly capture high-frequency components. The design of this trivial lighting representation choice is assigned by the employed rendering model. In contrast to supervised training, our self-supervised training usually has a poor optimisation surface. So directly training the network from scratch is not ensured to yield decent convergence, for which staged training strategy should be applied. Finally, the inverse rendering results cannot be seamlessly used by renderers to faithfully produce high-fidelity renderings, like relighting or novel view synthesis, even if they exhibit plausible colour estimates and geometry details. In next chapter, we will report how a hybrid model between model-base and neural decoder can be exploited to solve these issues.

Chapter 5

Hybrids of model-based and neural decoders

5.1 Introduction

Over the last two chapters, we have a deeper exploration on model-based decoders and how they contribute to training neural networks. Although such decoders are capable of simplifying the computational complexity and constraining the solution space by the underlying physical rules, the performance is limited by the nature of the physical model. In contrast, neural decoders are trained to reproduce ground truth data by black-box networks representing non-explainable and complex models. To address the limitations of both kinds of decoders, we propose a hybrid decoder that sequentially fuse them as a single decoder such that the hybrid decoder is more expressive than a model-based decoder and more physically legitimate than a neural decoder. Specifically, we expand our discussions about hybrid decoders by two applications. First, we study the hybrid decoder constituted by a neural decoder followed by a model-based decoder. With this type of decoder, we propose a single image novel view synthesis network, which is shown in Figure 5.1. Second, we instead construct the hybrid decoder by concatenating a neural decoder subsequent to a model-based decoder. Deriving from this decoder, we present a single image relighting network (see Figure 5.7).

Figure 5.1: Given a single RGB image, we perform depth estimation using the MegaDepth Network [84] and inverse rendering using InverseRenderNet. The geometry is then computed by merging depth and normal estimations. Finally, we texture triangulated meshes with albedo estimates and re-render the scene with novel lighting and viewpoint.

5.2 Hybrid decoders for monocular depth estimation and novel view synthesis

Along this line of discussion, we seek to use a model-based decoder, which formulates the derivative relationship between depth and normal to refine single image depth predictions by two neural decoder networks [84] and InverseRenderNet. Note that both of InverseRenderNet and InverseRenderNet++ can serve to provide normal map in this section, we are showing results only based on InverseRenderNet, because the network is computationally lighter compared to InverseRenderNet++ due to the differences on shadow decomposition and lighting information model. Here, normal estimation picks up fine detail because it is trained to reproduce the input image and can exploit appearance/shading information at every pixel. But it does not give absolute depth so cannot do view point edits, simulate shadows etc. Depth estimates provide base of global estimate but rely on training data that misses high frequency details. Hence, merging the two outputs can potentially provide the best of both. Further, taking advantage of these two divergent strands of research we are able to synthesise novel views for the captured image. Specifically, using state of the art deep CNNs for each task, we merge the coarse depth map and high quality normal map yielding a high quality geometric model. Combined with the estimated albedo map, this provides a textured, relightable 3D model that can be used for novel view synthesis.

An overview of our approach is shown in Figure 5.2. We use a state-of-the-art neural decoder network for single image depth estimation (MegaDepth [84]) and for inverse rendering (InverseRenderNet). Our key insight is that merging the two shape estimates from these complimentary techniques, using

a model-based decoder originated from a variant of the method proposed by Nehab *et al*. [112], yields high quality geometry that can be re-lit using the albedo estimated by the inverse rendering network. The new lighting and viewing direction can be controlled by users. In the remainder of this branch of discussion, we begin by introducing necessary notation, review the depth estimation and inverse rendering networks, describe the merging process and then present our results.

5.3 Perspective geometry

We begin by introducing required notations and concepts from single view perspective geometry. We work in the coordinate system of the camera and parameterise the scene by the unknown depth function $Z(\mathbf{u})$, where $\mathbf{u} = (x, y)$ is a location in the image. The 3D coordinate at $\mathbf{u}$ is given by:

$$P(\mathbf{u}) = \begin{bmatrix} \frac{x-x_0}{f} Z(\mathbf{u}) \\ \frac{y-y_0}{f} Z(\mathbf{u}) \\ Z(\mathbf{u}) \end{bmatrix}, \tag{5.1}$$

where f is the focal length of the camera and (x_0, y_0) is the principal point.

The tangent vectors to the surface are given by:

$$\frac{\partial P(\mathbf{u})}{\partial x} = \begin{bmatrix} \frac{1}{f}\left((x-x_0)\frac{\partial Z(\mathbf{u})}{\partial x} + Z(\mathbf{u})\right) \\ \frac{1}{f}(y-y_0)\frac{\partial Z(\mathbf{u})}{\partial x} \\ \frac{\partial Z(\mathbf{u})}{\partial x} \end{bmatrix}, \tag{5.2}$$

$$\frac{\partial P(\mathbf{u})}{\partial y} = \begin{bmatrix} \frac{1}{f}(x-x_0)\frac{\partial Z(\mathbf{u})}{\partial y} \\ \frac{1}{f}\left((y-y_0)\frac{\partial Z(\mathbf{u})}{\partial y} + Z(\mathbf{u})\right) \\ \frac{\partial Z(\mathbf{u})}{\partial y} \end{bmatrix}. \tag{5.3}$$

Note that these tangent vectors are linear functions of the surface depth.

The direction of the outward pointing surface normal is defined as the cross product of the tangent

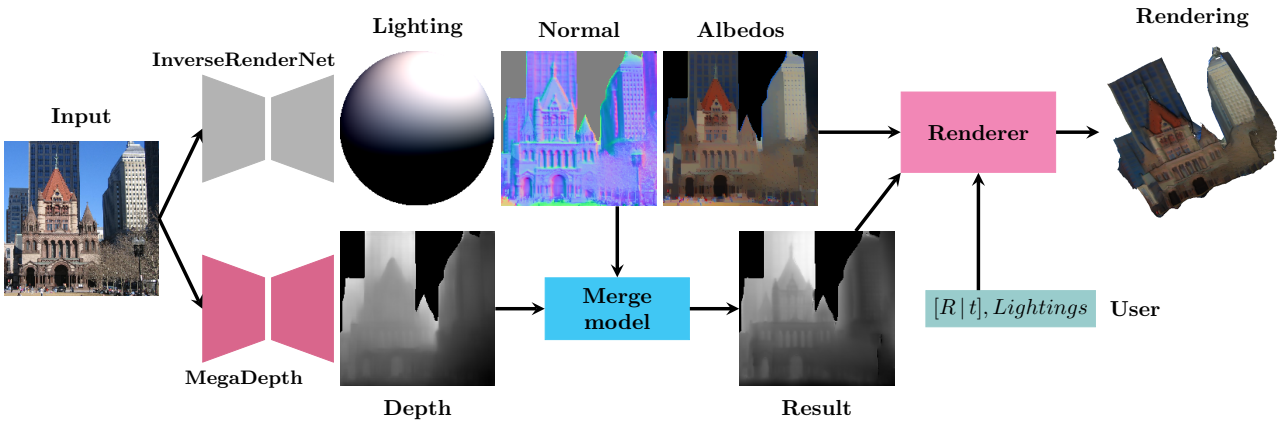


Figure 5.2: Overview of our proposed process for merging depth predictions and inverse rendering results for novel view synthesis. The input image is decomposed into normal, albedo and lighting by InverseRenderNet, and depth map by MegaDepth model [84]. Our proposed model merges depth and normal estimates to regress a new depth map, which can be combined with albedo estimate, new viewing direction and new lighting specified by user to render an image under the novel view and illumination.

Figure 5.3: Sample output from MegaDepth [84]. Dark is closer to viewer.

vectors, themselves the partial derivatives of the position function:

$$\mathbf{n}(\mathbf{u}) = \frac{\partial P(\mathbf{u})}{\partial x} \times \frac{\partial P(\mathbf{u})}{\partial y} = k \begin{bmatrix} -f\frac{\partial Z(\mathbf{u})}{\partial x} \\ -f\frac{\partial Z(\mathbf{u})}{\partial y} \\ (x - x_0)\frac{\partial Z(\mathbf{u})}{\partial x} + (y - y_0)\frac{\partial Z(\mathbf{u})}{\partial y} + Z(\mathbf{u}) \end{bmatrix}, \tag{5.4}$$

where k is an unknown scale factor. Note that the magnitude of the surface normal vector is not important, only its direction. Also note that linearly scaling the depth function does not change the direction of the surface normal vector. We denote by $\bar{\mathbf{n}}(\mathbf{u}) = \mathbf{n}(\mathbf{u})/\|\mathbf{n}(\mathbf{u})\|$, the unit length surface normal.

5.4 Single image depth estimation and inverse rendering

The goal of single image depth estimation is to compute a depth value, $Z(\mathbf{u})$, for each pixel in the image, collectively known as a depth map. Note that from Equation (5.1), this cannot be transformed into positions in world units without knowing camera calibration information. In low accuracy applications, the principal point is usually assumed to be the centre of the image and we make this assumption. The focal length however is typically unknown. However, often a good estimate can be made from image metadata and a database of sensor sizes. We take this approach allowing us to assume the focal length is known. However, estimating absolute depth from a monocular image is highly ambiguous. For this reason, depth prediction networks usually estimate depth only up to an unknown global scale $s > 0$ such that $\tilde{Z}(\mathbf{u}) = sZ(\mathbf{u})$. This does not affect the surface normals and hence the merging process described later. However, for the purposes of computing scene geometry for rendering an absolute scale must be chosen and we leave this as a parameter for the user to select.

We use the depth prediction network presented by Li and Noah [84] (known as MegaDepth). This is trained in a supervised fashion using depth maps computed from scene geometry recovered using multiview stereo applied to community photo collections. The supervised training loss function is scale invariant so that the output scale of the depth map is arbitrary as discussed above. We show sample output in Figure 5.3.

We use the inverse rendering network of InverseRenderNet, exactly as described in Chapter 4. As shown in Figure 5.4, InverseRenderNet can factor input image into normal map with fine-grained details that are hardly preserved in depth predictions of Figure 5.3, along with albedo map and illumination, allowing our framework to integrate such decomposed intrinsics, especially detailed normal map into coarse depth map.

5.5 Merging depth and normals

Our model-based decoder for merging surface normal and depth map is based on the method proposed by Nehab *et al*. [112]. However, we include the centre of projection in our derivation and provide an

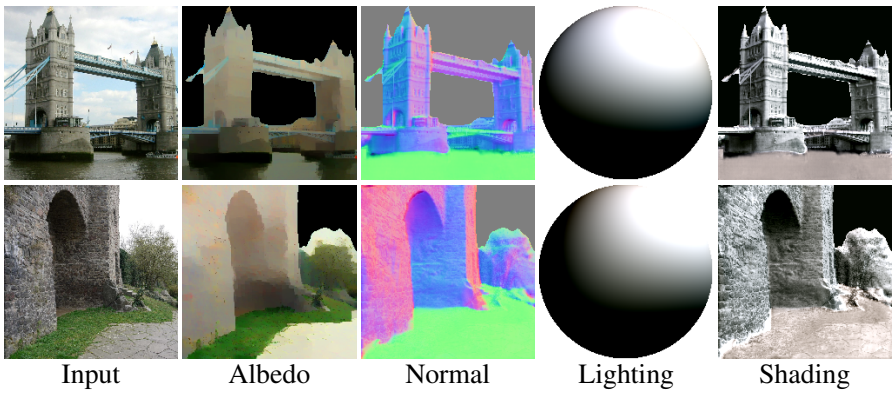


Figure 5.4: Sample outputs from InverseRenderNet.

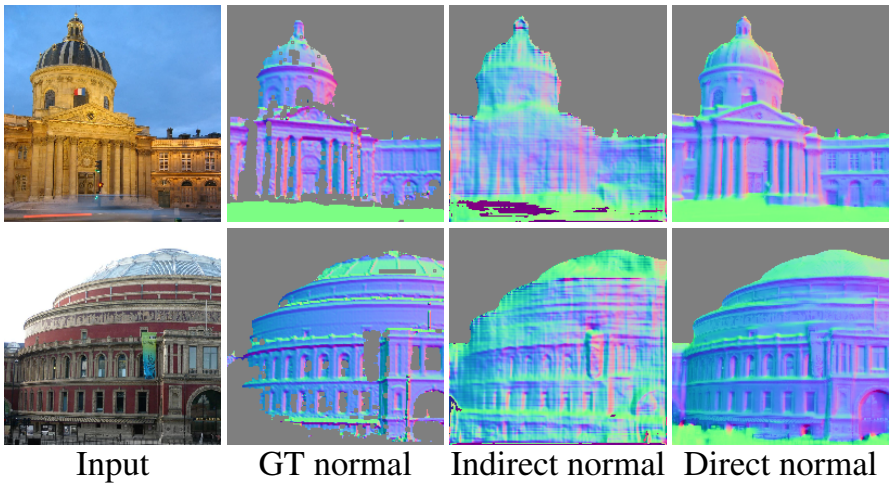


Figure 5.5: Comparison between direct normal estimates acquired by InverseRenderNet and indirect normal estimates, which is computed from depth estimate.

explicit matrix formulation, decomposing the original formulation in terms of matrices for computing tangent vectors, numerical derivatives and dot products. This makes reproducing our approach much more straightforward and we make a Matlab implementation publicly available (`https://github.com/waps101/MergePositionNormals`). Unlike the method proposed by Nehab *et al*. [112], we do not begin by removing low frequency bias from the estimated surface normals. In practice, we do not find that the surface normals delivered by InverseRenderNet are subject to low frequency bias in the same way that normals from photometric methods could be. As shown in Figure 5.5, normal map computed from depth prediction, referred to as indirect normal cannot provide more accurate low-frequency result than the direct normal estimate from InverseRenderNet. We believe this is because of the direct normal map supervision during training.

The approach introduced by Nehab *et al*. [112] is to form a linear system of equations in the merged depth that seeks to satisfy two constraints. The first seeks to preserve the gross structure of the

original estimated depth map by penalising in a least squares sense any deviations. The second seeks to encourage the surface normals of the refined depth map to align with the target normals. The key observation is that this second error function can be formulated to be linear in the surface depth. This is achieved by encouraging the tangent vectors of the refined surface to be perpendicular to the target normals, i.e., have a zero dot product.

First, we extend Equation (5.2) and Equation (5.3) to the whole image. Consider an image with shape $M \times M$ and N foreground pixels whose unknown depth values are vectorised in $\mathbf{z} \in \mathbb{R}^N$:

$$\mathbf{z} = \begin{bmatrix} Z(\mathbf{u}_{1,1}) & Z(\mathbf{u}_{2,1}) & \dots & Z(\mathbf{u}_{M,1}) & Z(\mathbf{u}_{1,2}) & Z(\mathbf{u}_{2,2}) & \dots & Z(\mathbf{u}_{M-1,M}) & Z(\mathbf{u}_{M,M}) \end{bmatrix}^T, \quad (5.5)$$

where $\mathbf{u}_{i,j}$ indicates the index of pixel at ith row and jth column.

We define matrices:

$$\mathbf{T}_x = \begin{bmatrix} \frac{-1}{f}\mathbf{X} & \frac{-1}{f}\mathbf{I} \\ \frac{-1}{f}\mathbf{Y} & \mathbf{0}_{N\times N} \\ \mathbf{I} & \mathbf{0}_{N\times N} \end{bmatrix} \begin{bmatrix} \mathbf{W}_x \\ \mathbf{I} \end{bmatrix}, \quad \mathbf{T}_y = \begin{bmatrix} \frac{-1}{f}\mathbf{X} & \mathbf{0}_{N\times N} \\ \frac{-1}{f}\mathbf{Y} & \frac{-1}{f}\mathbf{I} \\ \mathbf{I} & \mathbf{0}_{N\times N} \end{bmatrix} \begin{bmatrix} \mathbf{W}_y \\ \mathbf{I} \end{bmatrix}, \quad (5.6)$$

such that tangent vectors for the whole image concatenated into a vector can be computed by post-multiplication with the vector of depth values:

$$\mathbf{T}_x\mathbf{z} = \text{vec}\left(\begin{bmatrix} \frac{\partial P(\mathbf{u}_1)}{\partial x} & \dots & \frac{\partial P(\mathbf{u}_N)}{\partial x} \end{bmatrix}^T \right), \quad (5.7)$$

$$\mathbf{T}_y\mathbf{z} = \text{vec}\left(\begin{bmatrix} \frac{\partial P(\mathbf{u}_1)}{\partial y} & \dots & \frac{\partial P(\mathbf{u}_N)}{\partial y} \end{bmatrix}^T \right), \quad (5.8)$$

where $\mathbf{I}$ is the $N \times N$ identity matrix and $\mathbf{X} = \text{diag}(x_1 - x_0, \dots, x_N - x_0)$ and $\mathbf{Y} = \text{diag}(y_1 - y_0, \dots, y_N - y_0)$. $\mathbf{W}_x, \mathbf{W}_y \in \mathbb{R}^{N\times N}$ compute numerical approximations to the derivative of Z in the x and y directions respectively. In practice, we use forward finite differences to lower the computational cost. Hence $\mathbf{W}_x, \mathbf{W}_y$ have two non-zero values per row:

$$\mathbf{W}_x^{i,j}(\mathbf{S}) = \begin{cases} -1 & \text{if} \quad i = j \\ 1 & \text{if} \quad \mathbf{S}_j = \mathbf{S}_i + (0,1)\,, \\ 0, & \text{otherwise} \end{cases} \quad \mathbf{W}_y^{i,j}(\mathbf{S}) = \begin{cases} -1 & \text{if} \quad i = j \\ 1 & \text{if} \quad \mathbf{S}_j = \mathbf{S}_i + (1,0)\,, \\ 0, & \text{otherwise} \end{cases} \tag{5.9}$$

where $\mathbf{S}$ stands for another indexing vector with shape M that links index of depth vector $\mathbf{z}$ and indices of image pixel $\mathbf{u}$:

$$\mathbf{S} = [(1,1),\,(2,1),\,\ldots,\,(M,1),\,(1,2),\,(2,2),\,\ldots,\,(M-1,M),\,(M,M)]. \tag{5.10}$$

With this index vector $\mathbf{S}$, we can find row and column indices for $Z(\mathbf{u})$ with index in $\mathbf{z}$:

$$\mathbf{z}_i = Z(\mathbf{u}_{\mathbf{S}_i}). \tag{5.11}$$

Note that we flatten depth map as a depth vector $\mathbf{z}$ and define derivative operators $\mathbf{W}_x$ and $\mathbf{W}_y$ accordingly, so the computations define here only involve matrix multiplications without convolution.

We are now ready to write the linear system of equations:

$$\begin{bmatrix} \lambda\mathbf{I} \\ \mathbf{N}\mathbf{T}_x \\ \mathbf{N}\mathbf{T}_y \end{bmatrix} \mathbf{z} = \begin{bmatrix} \mathbf{z}_{\text{MD}} \\ \mathbf{0}_{2N\times 1} \end{bmatrix}, \tag{5.12}$$

where $\mathbf{z}_{\text{MD}} \in \mathbb{R}^N$ contains the coarse depth estimates delivered by MegaDepth and

$$\mathbf{N} = \begin{bmatrix} \text{diag}\left(n^x_{\text{IRN}}(\mathbf{u}_1), \ldots, n^x_{\text{IRN}}(\mathbf{u}_N)\right) \\ \text{diag}\left(n^y_{\text{IRN}}(\mathbf{u}_1), \ldots, n^y_{\text{IRN}}(\mathbf{u}_N)\right) \\ \text{diag}\left(n^z_{\text{IRN}}(\mathbf{u}_1), \ldots, n^z_{\text{IRN}}(\mathbf{u}_N)\right) \end{bmatrix}^T, \tag{5.13}$$

is an $N \times 3N$ matrix formed by concatenating diagonal matrices containing the x, y and z components of the target normals delivered by InverseRenderNet. Hence, each row contains one of the target

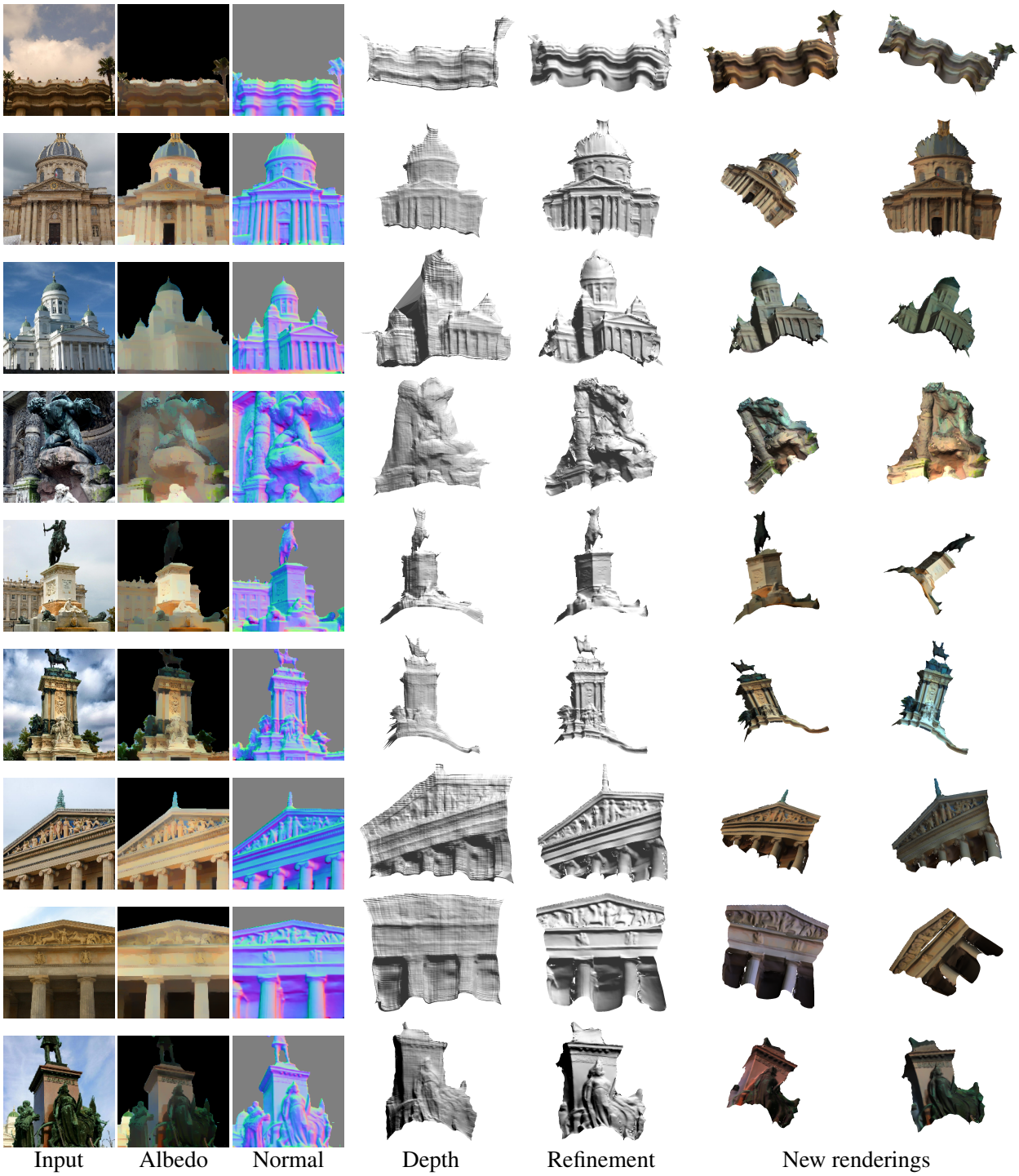


Figure 5.6: Results of applying our method to images from the MegaDepth dataset [84]. Column 1: input image. Column 2 and Column 3: albedo and surface normal maps estimated by InverseRenderNet. Column 4: rendering of the geometry provided by the depth prediction network. Column 5: refined geometry after merging with the surface normals. Column 6 and Column 7: novel views under two different lighting conditions.

surface normal vectors. The linear system of equations in Equation (5.12) is large but sparse over-determined system, and can be solved efficiently. We do so using a QR solver as implemented in Matlab's `mldivide` function. Specifically, the pseudo-inverse matrix is formed by $\mathbf{R}^{-1}\mathbf{Q}^T$ and

Method	Depth MSE	Normals Mean	Normals Median
MegaDepth [84]	**7.318**	43.6	43.0
Ours	7.961	**41.3**	**40.1**

Table 5.1: Quantitative evaluation of depth refinement results. Depth errors are measured by MSE, and normal predictions are qualified by mean and median angular errors.

multiplied to the RHS of this equation to find the least square solution of $\mathbf{z}$:

$$\begin{bmatrix} \lambda\mathbf{I} \\ \mathbf{N}\mathbf{T}_x \\ \mathbf{N}\mathbf{T}_y \end{bmatrix} = \mathbf{Q}\mathbf{R}, \quad \mathbf{z} = \mathbf{R}^{-1}\mathbf{Q}^T \begin{bmatrix} \mathbf{z}_{\text{MD}} \\ \mathbf{0}_{2N\times 1} \end{bmatrix}. \tag{5.14}$$

The parameter λ balances the influence of the two constraints. When λ is large, the refined depth will stay close to the original estimate. When it is small, the surface normals have greater influence.

5.6 Experiments on depth refinement and novel view synthesis

The overview in Figure 5.2 shows an example output from MegaDepth and InverseRenderNet. The striking feature is how the depth prediction is significantly improved when merged with the surface normal estimates. We show further results in Figure 5.6 for uncontrolled outdoor scenes. Here, we render the geometry as a mesh (by triangulating the depth map). Again, it is evident that the geometry is much improved, adding detail but also correcting gross structures. The textured models provide plausible appearance under large illumination and viewpoint changes. The quantitative comparison between our refined depth prediction and initial depth estimate from MegaDepth [84] is reported in Table 5.1. It is shown that the refined depth map from our proposed method can obtain noticeably improvements in normals prediction while retain competitive performance on depth prediction. Note that here we compute normals by applying camera intrinsics and optimal scaling on depth map, like evaluations in Chapter 4.

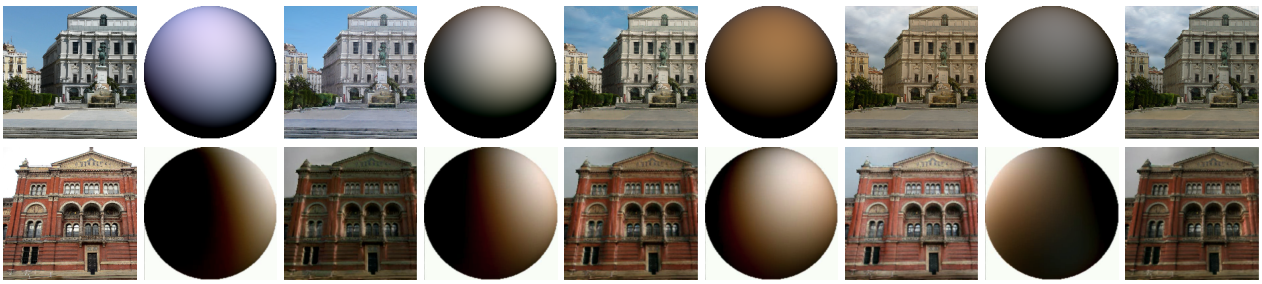


Figure 5.7: We present a novel self-supervised technique to photorealistically relight an outdoor scene from a single image to any given target illumination condition. Our method is able to generate plausible shading, shadows, colour-cast and sky region in the output image, while preserving the high-frequency details of the scene reflectance.

5.7 Self-supervised Outdoor Scene Relighting

The method described above applies a model-based decoder to outputs from neural decoders, for the purpose of merging two geometry representations. One can envisage another form of hybrid decoder that operates in the reverse manner, namely feeding outputs from a model-based decoder into a neural decoder. Such an architecture can be used to enable the trainable neural decoder to learn to remedy deficits in the model, enabling an overly simplistic model to be corrected allowing more real world phenomena to be captured. We pursue this idea for the purpose of outdoor scene relighting, with the neural decoder learning to transform basic Lambertian renderings into photorealistic images. We combine such a hybrid decoder with InverseRenderNet++, show how to train it using a self-supervised regime and generate photorealistic single image relighting results.

Recently, the advent of adversarial learning technique [43] has enabled neural networks to generate photorealistic images. 'Neural rendering' techniques based on this principle have shown promising results in various allied tasks such as novel-view synthesis [97], view-dependent effects rendering [147] and appearance modification [105].

Along with the idea of neural rendering, we propose a fully self-supervised neural rendering framework for performing photorealistic relighting of an outdoor scene from a single image with full lighting controllability (see Figure 5.7), through combining InverseRenderNet++ with a neural renderer. By incorporating InverseRenderNet++, our method takes as input a single 2D image and estimates the underlying appearance parameters such as albedo, shading, shadows, lighting and normals. Given

these physical parameters along with the target lighting condition, a model-based renderer composes the initial relighting rendering image, which will be then fed into a set of novel neural rendering decoders for refinement and enhancement to generate photorealistic relighting of the scene and the sky region. The sequential combination of model-based renderer and neural renderer enables us to produce rendering images with high standard of realism and great generalisation and stabilisation.

By training our system in a completely self-supervised manner, it generalises to unseen novel scenes and any target lighting condition of choice as provided by the user in the form of an environmental light map. Finally, in order to qualify and quantify our relighting networks we introduce a new high-resolution HDR multi-view & multi-illuminant evaluation dataset for outdoor relighting, and our extensive test results on the dataset show the efficacy of our method.

In summary, the novel contributions of the proposed relighting network are:

- The first fully-automatic single-image based relighting technique for outdoor scenes with full controllability of target lighting
- A novel self-supervised neural rendering framework that uses physical intrinsic decomposition layers of the scene to generate photorealistic relighting results without using any ground-truth data or synthetic 3D rendering
- A sky generation network that generates realistic sky image for the scene under a given target lighting environment

5.7.1 Overview

Neural inverse rendering has been recently shown to enable convincing decomposition of both indoor [131] and outdoor (InverseRenderNet/InverseRenderNet++) uncontrolled scenes into geometry (normal map), illumination and reflectance. These methods are self-supervised via a physics-based model of image formation. Such models are typically based on simple assumptions such as perfect Lambertian reflectance and ignore global illumination effects and shadowing. For this reason, re-illumination of the geometry and reflectance with novel lighting does not lead to photorealistic images. In addition, sky regions do not adhere to reflectance models, and so they are either missing from relighting results

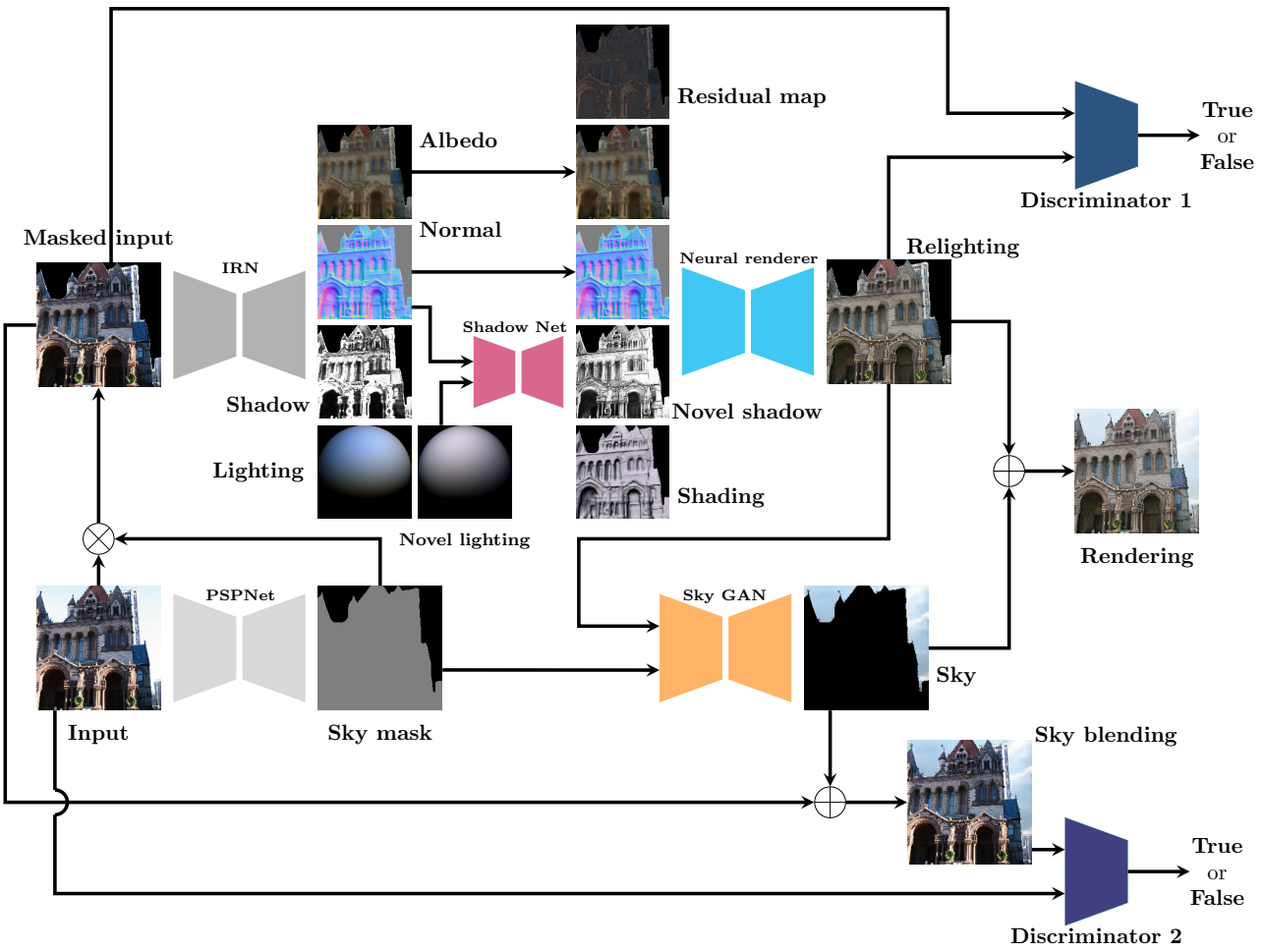


Figure 5.8: For a given *Input* image of an outdoor scene, our method first performs a physical decomposition of the scene into various components. Using a pre-trained segmentation network (**PSPNet** [169]), the scene is separated from the sky. The scene is then decomposed by the **InverseRenderNet++** into intrinsic image layers of *Albedo*, *Normal*, *Shadow* and *Lighting* . Given a target *Novel lighting* condition, **ShadowNet** uses the regressed scene normals to generate a target *Novel shadow* map for the scene. The scene albedo and normals, along with target lighting, shadow map, target shading and residual input map (see Section 5.8) are then fed to the **Neural renderer** to generate plausible *Relighting* of the scene. Given the output of the neural renderer, **SkyGAN** generates a convincing *Sky* region, and by compositing these together, a complete photorealistically relit *Rendering* is achieved. **Discriminator 1** and **Discriminator 2** are used to provide adversarial losses for training **Neural renderer** and **SkyGAN** respectively. Specifically, **Discriminator 1** takes as input skyless relighting or real image and determines if they are real or fake. **Discriminator 2** classifies the real input and the fake image (*Sky blending*) blended by real scene and fake sky.

or the original sky is pasted back, making it inconsistent with the new lighting.

Our goal in this relighting rendering network is to learn in a fully self-supervised fashion to perform photorealistic relighting of outdoor scenes from a single image. Photorealism is achieved by attaching a classical model-based renderer to a learnt neural renderer that can improve the inaccurate model-based rendering results to plausible relighting results. The neural renderer particularly

learns to synthesize global illumination effects such as plausible shadows, inter-reflections and view-dependent effects that are required for photorealism, which are much more difficult to simulate with model-based renderers. The neural renderer is trained using an adversarial loss to ensure that the generated images lie within the distribution of real images. A novel cycle consistency loss and direct supervision loss via cross projection of multi-view images is also used to ensure that the generated images exhibit the desired target lighting. We also present a sky generation network that learns to synthesize plausible skies that are consistent with the lighting within the rest of the image. An overview of our approach is shown in Figure 5.8.

The detailed illustrations of our relighting network are organised as follows. Section 5.8 presents our inverse rendering and novel neural rendering architecture, and how our hybrid decoder bridges the gap between them in the relighting process. In Section 5.9 we introduce our novel benchmarking dataset and perform detailed experimentation of our method to analyse our design choices and prove the efficacy of our methods in new unseen scenarios. Besides, the ablation study regarding each component included in our method is discussed in this section.

5.8 Inverse rendering and neural rendering

We take as our starting point the inverse rendering network of InverseRenderNet++. As described in Chapter 4, this image-to-image network decomposes an image into diffuse albedo, normal map, spherical harmonic illumination and shadow map, which yields source of input to relighting neural rendering network.

5.8.1 Shadow prediction network

There could be two ways to handle shadows in our neural renderer. The first was to leave it to the neural renderer to learn to add shadows to rendered images given only the unshadowed local shading. However, this task was too challenging for the renderer and shadows were largely missing from rendered output. The second approach was to explicitly attempt to estimate a new shadow map for

the desired lighting. To estimate such changes in shadows, we train a separate shadow prediction network. It takes as input a normal map and the spherical harmonic lighting vector and outputs a shadow map. In order to input the lighting vector while retaining the image-to-image architecture of the network, we replicate the 27D lighting vector (since $\mathbf{L} \in \mathbb{R}^{3\times 9}$) pixel-wise and concatenate it to the normal map such that the input is a 30D tensor. We train the shadow prediction network using illumination, normal and shadow maps predicted by InverseRenderNet++.

The objective function for training this network is based on L2 distance:

$$\ell_{\text{shadow}} = \|\boldsymbol{\omega}(\mathbf{n}_i, \mathbf{l}_i) - \mathbf{s}_i\|^2_{\text{fro}}, \tag{5.15}$$

where ShadowNet $\boldsymbol{\omega}(\mathbf{n}_i, \mathbf{l}_i)$ is the generated shadow map by taking as input normal $\mathbf{n}_i$ and lighting $\mathbf{l}_i$. $\mathbf{s}_i$ is the shadow map prediction from InverseRenderNet++, and it serves as supervision in this training. The normal $\mathbf{n}_i$, lighting $\mathbf{l}_i$ and shadow $\mathbf{s}_i$ predictions involved in this equation are all computed by running InverseRenderNet++ on image $\mathbf{i}_i$. So, this ShadowNet learns to generate shadow map based on how InverseRenderNet++ decomposes shadow, normal and lighting from the image.

5.8.2 Neural renderer

We now describe our neural rendering network. This can be viewed as a conditional GAN [107] in which the conditioning input is the maps required for a Lambertian rendering and the latent space is the spherical harmonic lighting parameter space. The objective of the network is to generate images indistinguishable from real ones while keeping the lighting consistent with the target lighting parameters.

The input to the neural rendering network is constructed from the outputs of InverseRenderNet++ (see Figure 5.8). The albedo and normals are taken as direct inputs from the output of InverseRendernet++, because they are scene invariants. Additional inputs of a shading map and a shadow map consistent with the target illumination are constructed. The shading channel is obtained using the Lambertian spherical harmonic lighting model under the desired lighting with the estimated normal map. The shadow map for a given novel lighting condition is predicted using a separate shadow prediction

network described in Section 5.8.1.

We concatenate the albedo prediction (3 channels), normal prediction (3 channels), shading (3 channels), shadow map (1 channel) and sky segmentation (1 channel) into an 11 dimensional tensor. In addition to this tensor, we compute another 3-channel *residual map* that contains the lost fine-scale details from original image after inverse rendering decomposition. The residual map is computed by subtracting Lambertian rendering composed by inverse rendering results from original input image. This *residual map* should contain texture details that InverseRendetNet++ fails to capture. As shown in our ablation study (Section 5.9.4), performing neural relighting simply from inverse rendering results could produce blurry relighting results, which motivates us to introduce this additional channels as input. We then stack this residual map at the end of concatenated 11 dimensional tensor and feed it to the neural rendering network.

5.8.3 Losses

We use three classes of loss function in order to train the neural renderer. First, an adversarial loss ensures the realism of the generated images. Second, direct supervision is provided in the form of self-reconstruction and cross-projection rendering losses to ensure the images are accurate predictions of the scene appearance under desired lighting conditions. Third, this direct supervision is aided by a cycle consistency loss that uses InverseRenderNet++ to measure and penalise inconsistent decompositions of original and rendered images.

Adversarial loss

For adversarial loss we use the multiscale LSGAN [95] architecture. Real images are true images with the sky masked out. Fake images are the neural renderings, again with all pixels in the sky region set to black.

The adversarial loss and discriminator loss are:

$$\ell_{\text{adv}} = \frac{1}{2} \mathop{\mathbb{E}}_{i \in A} [(\mathbf{D}_{\mathbf{G}}(\mathbf{G}_i) - 1)^2], \tag{5.16}$$

$$\ell_{\text{dis-g}} = \frac{1}{2} \mathop{\mathbb{E}}_{i \in A} [(\mathbf{D_G}(\mathbf{G}_i))^2] + \frac{1}{2} \mathop{\mathbb{E}}_{j \in B} [(\mathbf{D_G}(\mathbf{i}_j) - 1)^2], \tag{5.17}$$

where $\mathbf{D_G}$ is the discriminator, $\mathbf{i}_i$ is the real image, and $\mathbf{G}_i$ is the fake image generated by neural renderer. The discriminator being used to form ℓ_{adv} is shown as **Discriminator 1** in Figure 5.8. Note that this adversarial loss is only applied on relighting rendering results without considering the sky region. A separate GAN (**Discriminator 2** in Figure 5.8) for generating background sky image will be described in Section 5.8.4. Such a dedicated sky generation GAN is easier to train and showing improved results over a fused GAN.

Direct supervision

Our training set provides real example images under a variety of illumination conditions. We can exploit these for direct supervision. When the chosen lighting condition for relighting is the same as the original image, we expect the neural rendering to exactly match the original image. We refer to this as self-reconstruction loss. In practice, this is computed as a sum of the VGG perceptual loss [140] (difference in VGG features from the first two convolution blocks) and L2 distance in LAB colour space, which is defined by Equation (4.21).

$$\ell_{\text{self-reconstruction}} = \varepsilon(\mathbf{i}_i, \mathbf{G}(\boldsymbol{\alpha}_i, \mathbf{l}_i, \boldsymbol{\omega}(\mathbf{n}_i, \mathbf{l}_i), \mathbf{n}_i, \mathbf{r}_i)), \tag{5.18}$$

where $\mathbf{i}_i$ is the input image. The neural renderer $\mathbf{G}$ takes as input inverse rendering results from image $\mathbf{i}_i$ (albedo $\boldsymbol{\alpha}_i$, lighting $\mathbf{l}_i$ and normal $\mathbf{n}_i$), computed residual map $\mathbf{r}_i$, and the new shadow generated by ShadowNet $\boldsymbol{\omega}(\mathbf{n}_i, \mathbf{l}_i)$. Here, our neural renderer $\mathbf{G}$ is trained to perform self-reconstruction of the input image which is decomposed by InverseRenderNet++ to form input of $\mathbf{G}$.

However, self-reconstruction loss does not penalise baked-in effects. To overcome this, we use multiview supervision. A mini-batch consists of a set of overlapping images with different illumination and which can be cross projected from one view to another using the multi-view stereo (MVS) reconstructed geometry and camera parameters. We use this for additional direct supervision. Within a mini-batch, we shuffle the lighting estimates from InverseRenderNet++ so that we relight the albedo and normal predictions from one view with the lighting from another. We rotate the spherical har-

monic lighting to account for the relative pose between views. Supervision is provided by comparing the neural rendering against the cross projection of the view from which the lighting was taken, again measured in terms of VGG perceptual loss and L2 distance in LAB space. The loss can be computed by:

$$\ell'_{\text{cross-projection}} = \varepsilon(\mathbf{i}_{j\to i},\, \mathbf{G}(\boldsymbol{\alpha}_i,\, \mathbf{R}_{j\to i}\mathbf{l}_j,\, \boldsymbol{\omega}(\mathbf{n}_i,\, \mathbf{R}_{j\to i}\mathbf{l}_j),\, \mathbf{n}_i,\, \mathbf{r}_i)), \tag{5.19}$$

where $\mathbf{i}_{j\to i}$ is the cross-projected image from view j to view i, $\mathbf{R}_{j\to i}\mathbf{l}_j$ rotates lighting prediction from view j to view i. Please refer to Section 4.7.5 for the detailed definitions of this cross-projection and lighting rotation functions. Essentially, this loss trains the neural renderer to minimise the distance between the image $\mathbf{i}_j$ and the relighting result of image $\mathbf{i}_i$ under lighting in image $\mathbf{i}_j$.

However, errors in the MVS geometry and camera poses cause slight misalignments in the cross projected images. We found that applying this loss at full resolution led to a blurry output. For this reason, before computing the cross-projection loss, we downscale both the cross projected and rendered images by a factor of 4:

$$\ell_{\text{cross-projection}} = \varepsilon(\mathbf{i}'_{j\to i},\, \mathbf{G}'(\boldsymbol{\alpha}_i,\, \mathbf{R}_{j\to i}\mathbf{l}_j,\, \boldsymbol{\omega}(\mathbf{n}_i,\, \mathbf{R}_{j\to i}\mathbf{l}_j),\, \mathbf{n}_i,\, \mathbf{r}_i)), \tag{5.20}$$

where:

$$\mathbf{i}'_{j\to i} = \text{downsample}_4(\mathbf{i}_{j\to i}), \tag{5.21}$$

$$\mathbf{G}'(\boldsymbol{\alpha}_i,\, \mathbf{R}_{j\to i}\mathbf{l}_j,\, \boldsymbol{\omega}(\mathbf{n}_i,\, \mathbf{R}_{j\to i}\mathbf{l}_j),\, \mathbf{n}_i,\, \mathbf{r}_i) = \text{downsample}_4(\mathbf{G}(\boldsymbol{\alpha}_i,\, \mathbf{R}_{j\to i}\mathbf{l}_j,\, \boldsymbol{\omega}(\mathbf{n}_i,\, \mathbf{R}_{j\to i}\mathbf{l}_j),\, \mathbf{n}_i,\, \mathbf{r}_i)). \tag{5.22}$$

Cycle consistency

We found that direct supervision and adversarial loss alone are insufficient for good performance and smooth relighting under smooth illumination parameter changes. This is partly due to the fact that cross projected images are incomplete and can be quite sparse when the view change is large. Therefore, to improve stability we propose to also include a cycle consistency loss. Here, we use the InverseRenderNet++ trained as described in last chapter and measure the consistency between the input maps to the neural renderer and those obtained by decomposing the neural rendered image.

Specifically, we penalise the difference in the albedo, normal, lighting and shadow maps. Here, we denote InverseRenderNet++ as $\mathbf{V}$ and it decomposes relighting rendering from neural renderer:

$$\mathbf{V}(\mathbf{G}(\boldsymbol{\alpha}^i,\, \mathbf{l}^i,\, \boldsymbol{\omega}(\mathbf{n}^i,\, \mathbf{l}^i),\, \mathbf{n}^i,\, \mathbf{r}^i)) = \boldsymbol{\alpha}^{\mathbf{G}},\, \mathbf{n}^{\mathbf{G}},\, \mathbf{s}^{\mathbf{G}},\, \mathbf{l}^{\mathbf{G}}, \tag{5.23}$$

where $(\boldsymbol{\alpha}^i,\, \mathbf{l}^i,\, \boldsymbol{\omega}(\mathbf{n}^i,\, \mathbf{l}^i),\, \mathbf{n}^i,\, \mathbf{r}^i)$ are used to generate relighting image, which is then decomposed by InverseRenderNet++ to obtain inverse rendering results $(\boldsymbol{\alpha}^{\mathbf{G}},\, \mathbf{n}^{\mathbf{G}},\, \mathbf{s}^{\mathbf{G}},\, \mathbf{l}^{\mathbf{G}})$.

Lighting consistency is measured by the sum of VGG perceptual loss and L2 difference between the Lambertian shading maps:

$$\ell_{\text{lighting-cycle}} = \varepsilon(\mathbf{l}^i\, \mathbf{b}(\mathbf{n}^i),\, \mathbf{l}^{\mathbf{G}}\, \mathbf{b}(\mathbf{n}^{\mathbf{G}})), \tag{5.24}$$

where $\mathbf{l}^i\mathbf{b}(\mathbf{n}^i)$ computes Lambertian shading map from the inputted lighting, $\mathbf{l}^{\mathbf{G}}\, \mathbf{b}(\mathbf{n}^{\mathbf{G}})$ computes Lambertian shading map from the decomposed lighting, and the function $\mathbf{b}$ is defined by Equation (3.25). As for the VGG loss, although the VGG network is trained on natural photos rather than shading maps, we only use feature kernels in the shallow layers which are known to capture low-level structural information. So it is useful as a structural perception loss in this training.

Normal map consistency is measured by the mean angular error between original and estimated normal maps.

$$\ell_{\text{normal-cycle}} = \arccos(\mathbf{n}^i \cdot \mathbf{n}^{\mathbf{G}}). \tag{5.25}$$

For albedo consistency, we weight the error by the shading map like Meka *et al*. [104]. The idea is that albedo estimates in darkly shaded regions are unlikely to be accurate and we do not wish to overemphasise errors in these regions. Again, the albedo difference is measured in terms of VGG perceptual loss and L2 distance in LAB space:

$$\ell_{\text{albedo-cycle}} = \varepsilon(w^i\boldsymbol{\alpha}^i,\, w^{\mathbf{G}}\boldsymbol{\alpha}^{\mathbf{G}})). \tag{5.26}$$

w^i and $w^{\mathbf{G}}$ are weight maps:

$$w^i = \frac{\mathbf{l}^i_r\, \mathbf{b}(\mathbf{n}^i) + \mathbf{l}^i_g\, \mathbf{b}(\mathbf{n}^i) + \mathbf{l}^i_b\, \mathbf{b}(\mathbf{n}^i)}{3}, \quad w^{\mathbf{G}} = \frac{\mathbf{l}^{\mathbf{G}}_r\, \mathbf{b}(\mathbf{n}^{\mathbf{G}}) + \mathbf{l}^{\mathbf{G}}_g\, \mathbf{b}(\mathbf{n}^{\mathbf{G}}) + \mathbf{l}^{\mathbf{G}}_b\, \mathbf{b}(\mathbf{n}^{\mathbf{G}})}{3}. \tag{5.27}$$

These two weight maps are pixel-wise average across (r, g, b) channels of shading maps.

Shadow consistency loss is defined as L2 error between input shadow and decomposed shadow:

$$\ell_{\text{shadow-cycle}} = \|\mathbf{s}^{\mathbf{G}} - \boldsymbol{\omega}(\mathbf{n}^i, \mathbf{l}^i)\|^2_{\text{fro}}. \tag{5.28}$$

5.8.4 Sky GAN

Our physical illumination model is only able to describe non-sky regions of the image. Sky cannot be meaningfully represented in terms of geometry, reflectance and lighting. Moreover, sky appearance is partially stochastic (the precise arrangement of clouds is not informative). For this reason, we train a second network specifically to generate skies that are plausible given the rest of the image. For example, if the image contains strong cast shadows and shading, one would expect a clear sky with sunlight coming from an appropriate direction. If the image is highly diffuse with little discernible shading one would expect a cloudy sky.

For this purpose, we use the GauGAN architecture [119] with two semantic classes: sky and foreground. This network performs sky generation from random noise and conditional inputs of the sky segmentation mask and the foreground image with black sky. The output is the sky image which is blended with the foreground image using the binary sky mask. Such binary blended images are inputs to the discriminator along with the sky mask as a conditional input. Hence, the discriminator loss will help generate both more realistic skies but also skies that are plausible given the foreground appearance. Note that this sky generation network is trained by a separate discriminator which is different from the one used when training neural renderer. An visual illustration of this SkyGAN and its discriminator is shown in Figure 5.8. In this figure, the discriminator used here is referred to as **Discriminator 2**, while the discriminator supervising neural renderer is denoted as **Discriminator 1**.

To train the generator, we use the adversarial loss and the feature matching loss as in [119] but remove other appearance losses. We train using real images in which sky has been masked to black. The discriminator is trained using the same loss as the original GauGAN [119]. We find that, in practice, this network generalises well to foregrounds generated using our neural rendering network.

5.8.5 Training

Our network architecture and training were implemented in TensorFlow. The neural rendering network and shadow prediction network are modelled as U-Net architectures [127] with skip-connections in order to preserve high-frequency details in the output. The skyGAN network is modelled after ResNet architecture [49] which avoids the problem of vanishing gradient in the initial stages of training and ensures a smooth convergence of the loss value. Adam optimiser [70] is used to optimize the network parameters. A batch size of 5 is used for training. All training images were resized to a size of 200×200 pixels to keep the training tractable on single-GPU hardware. For training data, we run our training and testing on the MegaDepth dataset [84]. The dataset contains multiview stereo images, which enable us to directly train inverse rendering network and find relative rotations between image views before shuffling illumination estimates. The dataset contains a variety of outdoor scenes. We apply the same training and testing split of InverseRenderNet/InverseRenderNet++ (see Section 4.5) for training these networks.

For our entire training, we use a fixed learning rate of 5×10^{-4}. Firstly, ShadowNet is trained simply by using ℓ_{shadow}. For the various loss functions used in training the neural renderer, the relative weights are chosen such that the value of the losses are in the same order of magnitude:

$$\ell = w_1\ell_{\text{albedo-cycle}} + w_2\ell_{\text{shadow-cycle}} + w_3\ell_{\text{normal-cycle}} + w_4\ell_{\text{lighting-cycle}} + w_5\ell_{\text{self-reconstruction}} + w_6\ell_{\text{cross-projection}} + w_7\ell_{\text{adv}}, \tag{5.29}$$

where w_1 = 3.0, w_2 = 0.4, w_3 = 1.0, w_4 = 4.0, w_5 = 1.0, w_6 = 0.5, and w_7 = 0.1.

For our sky generation network, we use the same architecture as that of GauGAN [119], while changing the original conditional input of their semantic layout to concatenation of the sky segmentation map and the relit output from our neural renderer. To better learn this task with more network parameters, we also modify the number of residual blocks to 8 and change the length of the input random noise vector to 256. The rest of the architecture remains the same.

The training of the networks is performed in several stages. The inverse rendering network is trained

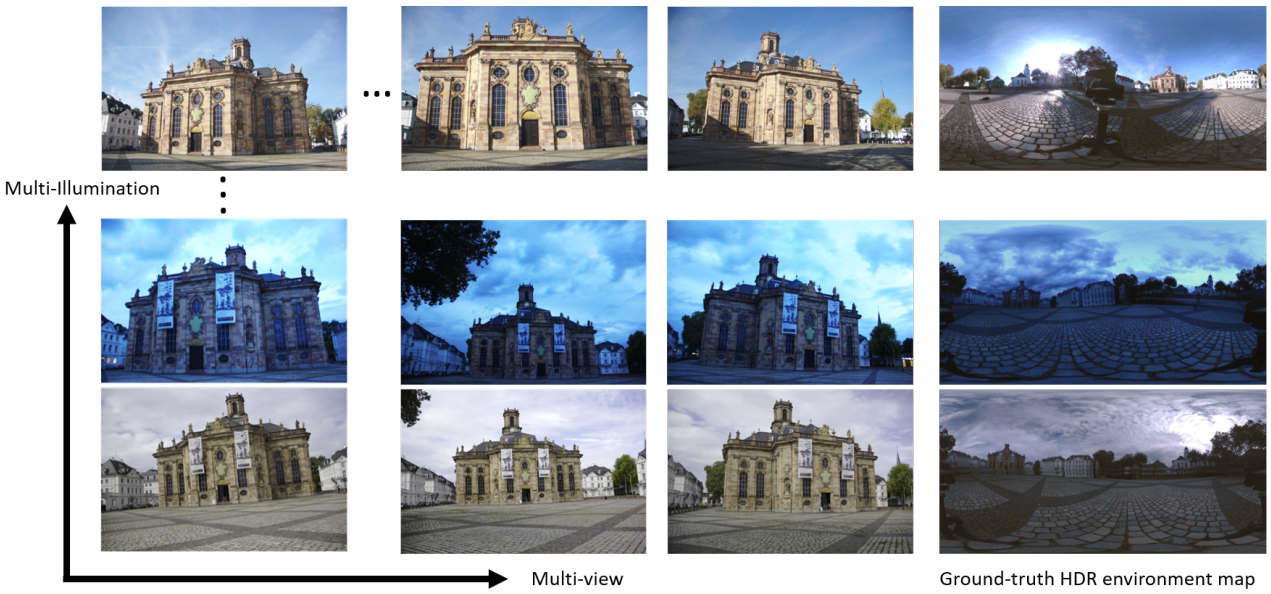


Figure 5.9: We present a new high-quality high-resolution outdoor relighting dataset. Our dataset consists of high-resolution HDR images of a single monument captured under several different lighting conditions from multiple views, along with the ground-truth HDR environment light maps.

independently as described in Section 4. The output of inverse rendering network is used to train the shadow prediction network. Given the well-trained shadow prediction network and inverse rendering network, the neural rendering network is trained. The training of the neural rendering network is done in two phases. In the first phase only a self-reconstruction loss is employed, and this stage is stopped when the loss reaches a steady-state value. In the second phase, the cycle-consistency loss and adversarial loss are added. In the experiments, we found such pre-training step ensures fast convergence and leads to renderings containing more fine details. Finally, we train our skyGAN as described in Section 5.8.4.

5.9 Experiments on outdoor relighting networks

5.9.1 Outdoor Relighting Benchmarking Dataset

We present a new high-quality benchmarking dataset for the evaluation of outdoor relighting techniques. The dataset consists of several sets of multi-view, multi-illumination high dynamic range (HDR) images of a single monument, along with ground-truth HDR environment maps for each illumination condition. We captured 6 different lighting conditions, including clear sky with bright

sunlight, cloudy overcast sky and evening light. For each lighting condition, we capture 10 images from views around the monuments and also a separate panoramic ground-truth environment light map. Each image of the monument is of resolution 5184×3456, captured with a Canon 5D Mark II DSLR camera with an 18mm focal length lens. It consists of 6 multi-exposure raw captures, which are fused in Adobe Photoshop to generate an HDR image. The lowest camera exposure time is chosen to ensure that the captured image has minimal amount of pixel saturation from bright light sources such as the sun. We use constant ISO and aperture settings in the capture. The environment light map is captured using a 360-degree camera (LG360) with 6 multi-exposure shots fused to obtain the HDR image.

While the original environment maps are captured from arbitrary viewpoints, in order to perform view consistent relighting, the environment maps need to be rotated to align them to the same viewpoint as the camera images. This is achieved by performing multi-view 3D reconstruction of the monument from all the dataset images and estimating accurate camera pose for each camera view through bundle adjustment, which is done by the multiview stereo tool [3]. The rotation between environment map and the global co-ordinate system of the monument (taken as the camera co-ordinate system of the first camera view image) is computed by performing a sparse feature match between the environment map and the 3D model and optimizing for the camera rotation between the two. This process is repeated for each of the 6 lighting conditions. In the dataset, we provide the camera pose for every image and also the rotation for each of the 6 ground-truth environment maps to the first camera view image. This provides 'aligned environment maps' for each lighting condition.

Multiview stereo

We apply an off-the-shelf uncalibrated structure-from-motion and multiview stereo tool [3] to all 56 images in our benchmark dataset. We initialise the focal length estimates using the known lens focal length in mm and the pixel size on the sensor. The output is a mesh (cropped to the main buildings in the scene) comprising 90k vertices and per-image camera parameters (both intrinsic, including nonlinear distortion parameters, and extrinsic, i.e. a rotation matrix $\mathbf{R}^{\text{w2c}} \in \mathbb{R}^{3\times3}$ and translation $\mathbf{t} \in \mathbb{R}^3$ that transform world to camera coordinates).

Figure 5.10: Reconstructed scene used in our benchmark. Estimated camera positions are shown as crosses and the aligned environment maps rendered on spheres.

Environment map alignment

We manually label a set of features points on the 3D mesh $\{\mathbf{v}_i\}_{i=1}^n$ with $\mathbf{v}_i \in \mathbb{R}^3$ and corresponding points on the 2D environment map $\{\mathbf{x}_i\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^2$. We transform the 2D environment map points into spherical coordinates and then unit length direction vectors $\mathbf{d}_i \in \mathbb{R}^3$ with $\|\mathbf{d}_i\| = 1$. To align the environment map to the world coordinates of the 3D mesh, we solve the following nonlinear optimisation problem:

$$\min_{\mathbf{c},\mathbf{R}} \sum_{i=1}^{n} \arccos\left(\mathbf{R}^{\text{e2w}}\mathbf{d}_i \cdot \frac{\mathbf{v}_i - \mathbf{c}}{\|\mathbf{v}_i - \mathbf{c}\|}\right). \tag{5.30}$$

This measures the total angular error between the vector from the spherical camera centre to one of the 3D feature points and the corresponding feature on the spherical image (the unit vector rotated to world coordinates). $\mathbf{R}^{\text{e2w}} \in \mathbb{R}^{3\times3}$ is a rotation matrix that rotates environment map coordinates to world coordinates and $\mathbf{c} \in \mathbb{R}^3$ is the centre of the spherical camera in world coordinates. We optimise the rotation as a 3D axis-angle vector and a 3D translation meaning the optimisation is 6D overall. We initialise the camera centre as the mean camera position over the multiview dataset. We solve the optimisation problem using the BFGS Quasi-Newton method. In all cases, the mean angular error upon convergence is less than 1°. We show the reconstructed model, estimated camera positions (marked as crosses) and aligned environment maps in Figure 5.10.

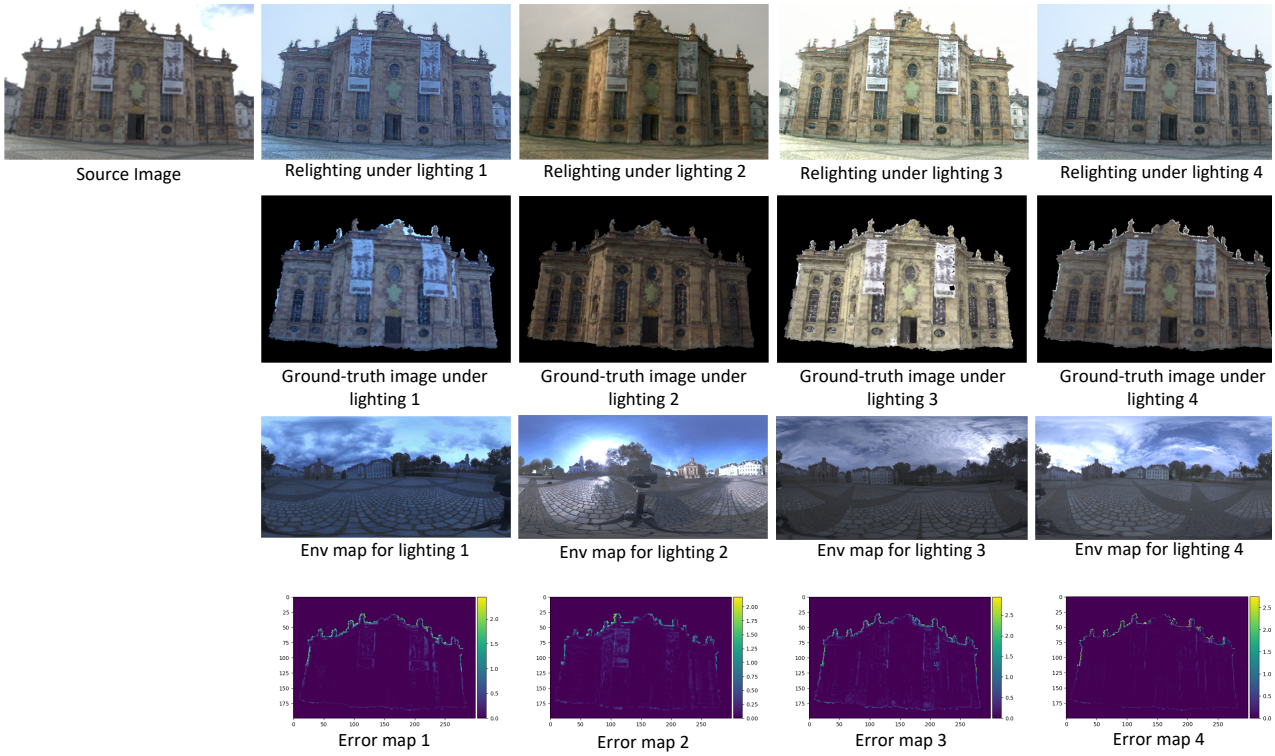


Figure 5.11: Relighting result on our new high-quality outdoor relighting dataset. Note the plausible shading effects obtained by our method on the surfaces of the monument compared to the ground-truth. The last row shows heatmaps of L2 error between relighting and ground truth.

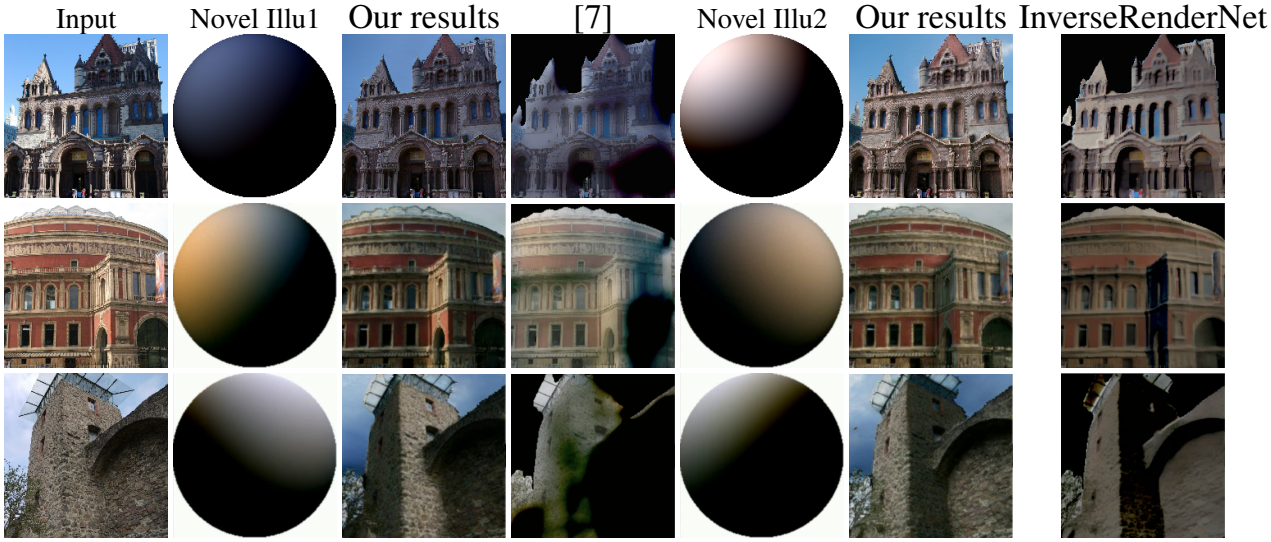


Figure 5.12: Relighting results from testing data. It shows the comparison between our methods with InverseRenderNet and SIRFS [7].

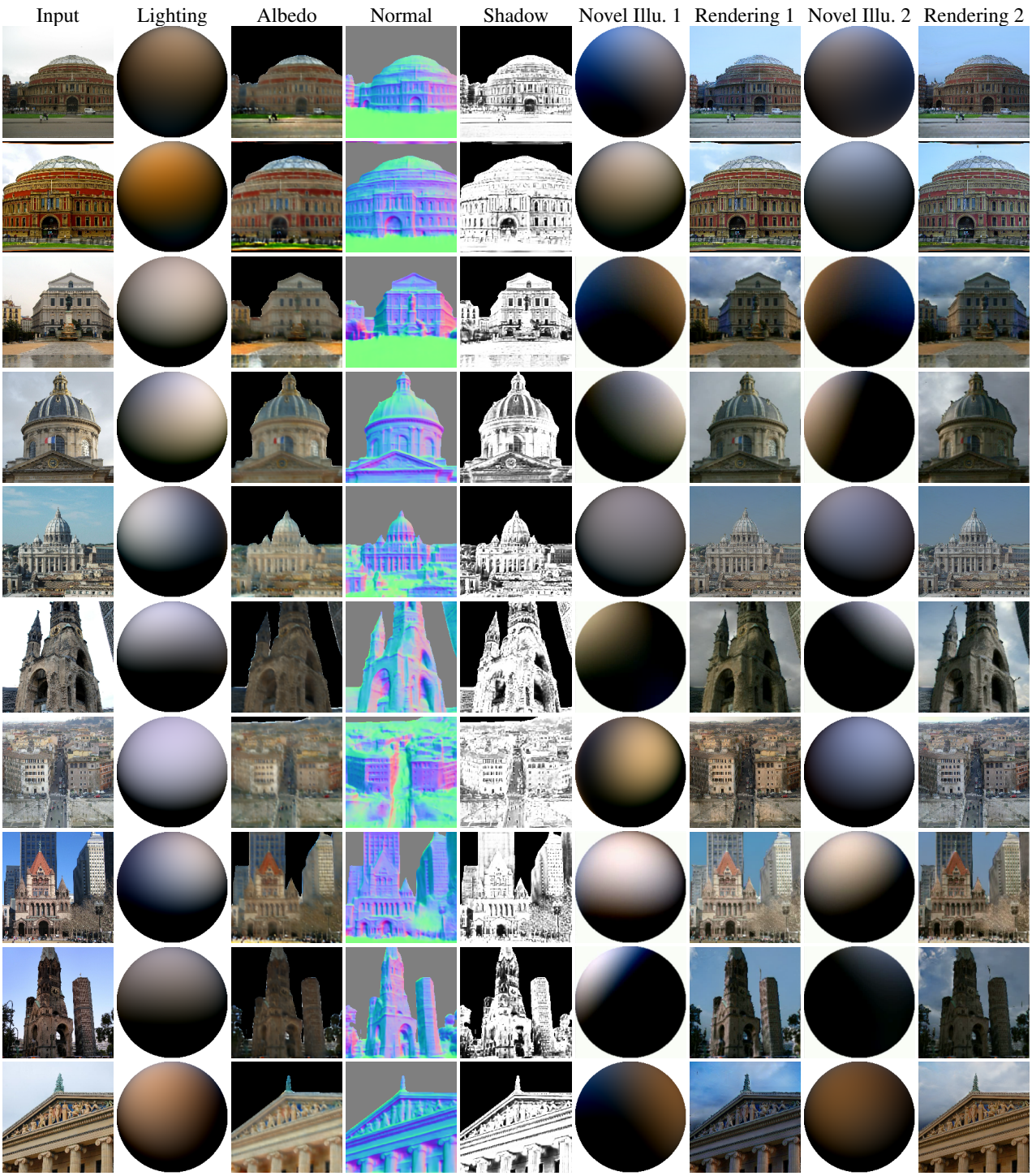


Figure 5.13: Inverse rendering and relighting results from our neural rendering network pipeline. Column 1 contains input images to the pipeline. Column 2-5 show inverse rendering results. Column 6 and Column 7 show two novel target illuminations. Column 8 and Column 9 show the relighting results from our neural renderer under the two novel target illuminations.

5.9.2 Qualitative Evaluation

On the Benchmarking Dataset

Our benchmarking dataset is used for qualitative evaluation of our method. We perform cross-relighting of the monument by taking an image for a particular lighting condition as input and per-

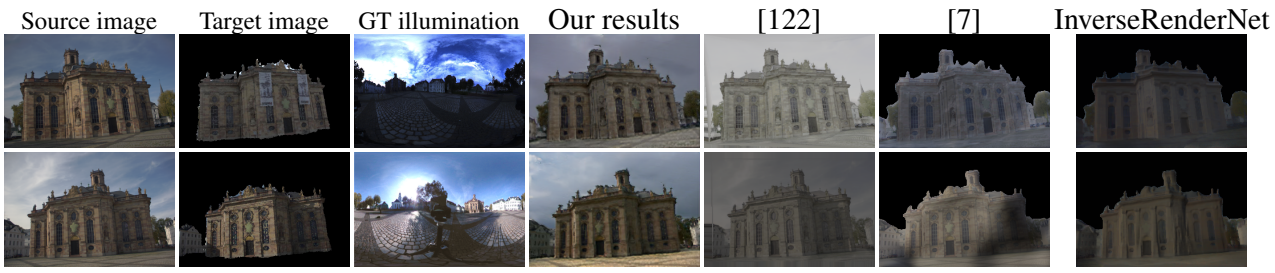


Figure 5.14: Relighting of benchmark dataset images and comparison with Philip *et al*. [122], InverseRenderNet and Barron and Malik [7].

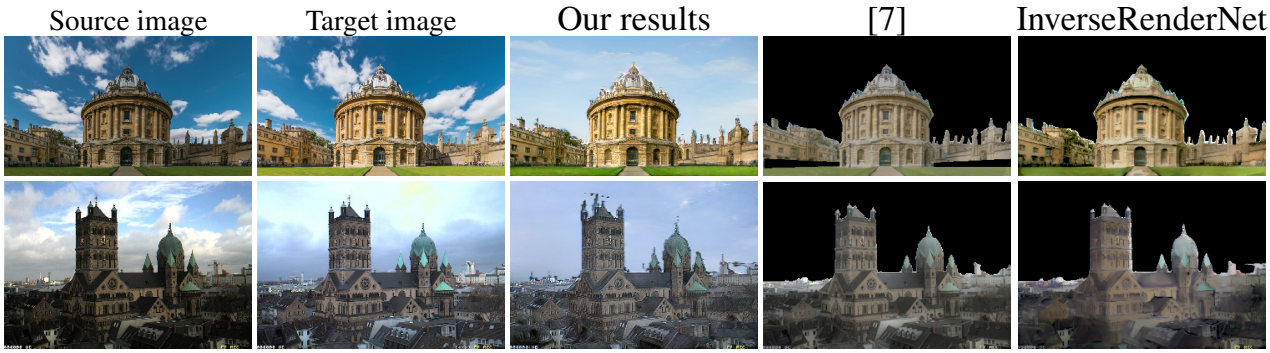


Figure 5.15: Relighting of BigTime images and comparison with InverseRenderNet and Barron and Malik [7].

forming relighting to another target light condition using as input the 2nd order spherical harmonic co-efficients of the ground-truth 'aligned' environment light map. The results for such relighting are shown in Figure. 5.11, where an image captured under a source lighting is relighted to several target lighting conditions. We also show error maps comparing relighting results and ground truth images by measuring L2 norm across different colour channels on each pixel location. The environment light map alignment is performed by rotating the target environment map into the coordinate system of the source camera, i.e. we apply $\mathbf{R}^{\text{w2c}}\mathbf{R}^{\text{e2w}}$ to the environment map in spherical harmonic coefficient space. We then relight using our method or one of the comparison methods. As can be seen, our method is able to generate relighting result that closely resembles the ground-truth images for each target lighting condition. Our method does a particularly good job of estimating plausible colour-cast and shading across various surfaces of the monument including those with intricate geometry.

On test dataset

In Figure 5.12, we show relighting results on our test dataset and comparison with other single-image relighting approaches. Our method results in realistic looking relighting results with shading and

shadows that are very consistent with the target lighting condition, while maintaining the fine underlying reflectance details. We also generate sky regions which match the general colour tone of the relit structure. InverseRenderNet generates non-photorealistic images due to its simple Lambertian reflectance model. The method of Barron and Malik [7] struggles with the darker sides of the target lighting conditions because it cannot account for global illumination.

We show more relighting results on the test dataset in Figure 5.13. Along with relit images, we also include inverse rendering results. The first column shows the input image. Column 2 to 5 show the inverse rendering results. The last four columns show two pairs of novel illuminations and relighting results. These results demonstrate that our network generates plausible shading, shadows and colour casts for a given novel target illumination.

On time-lapse dataset

We evaluate our neural rendering network on BigTime[86] dataset, which contains approximately 200 time-lapse image sequences of indoor and outdoor scenes. For each time lapse sequence, we perform cross-rendering by relighting each frame with lighting estimates from all the other frames in the sequence. To evaluate the relit results, we use multiple error metrics computed between the relit result and corresponding real image. The quantitative comparison, averaged over 15 sequences, is shown in Table 5.2, and qualitative comparison between our method and other methods is shown in Figure 5.15. The quantitative results show that our network can generalise well to time-lapse image sequences. Our method has the best performance on ℓ_1 error and the mean square error (mse) and is comparable to the method of Barron and Malik [7] on metrics measuring structural information like SSIM and DSSIM. Barron and Malik's [7] method seems to perform slightly better on these metrics because their method tends to estimate very smooth normal and shading image, which results in albedo images that although globally incorrect, still preserve local high-frequency details, leading to better structural similarity scores. This issue with their method is concealed when evaluating this dataset since the relighting is based on their estimated lighting. From Figure 5.15, it is evident that our method preserves the colour-cast and the brightness scale better and is able to generate accurate relighting effects such as consistent shading and shadows.

Method	ℓ_1	mse	SSIM	DSSIM
Proposed	**0.103**	**0.021**	0.760	0.120
InverseRenderNet	0.117	0.26	0.722	0.139
[7]	0.115	0.24	**0.770**	**0.115**

Table 5.2: Quantitative evaluation on the BigTime [86] time-lapse dataset. The error values are computed by averaging over 15 sequences.

5.9.3 Quantitative Evaluation

We also perform a quantitative evaluation on relighting results of our benchmarking dataset. Figure 5.14 shows example of the cross-relighting that we perform across all lighting conditions in the dataset. In order to get the ground-truth image for our relighting, we project all the camera images from a given target lighting condition onto the 3D geometry of the monument and average them. This is then re-projected to the camera viewpoint of the source image to obtain the ground-truth relit image. Although this leads to the loss of view-dependent effects, it still provides a plausible ground-truth image with accurate shadows and shading. Our evaluation metric is the L1 error between reprojected ground truth and relit image, averaged over colour channels and pixels, see Table 5.3. To remove unknown scale factors, we compute the error after applying the optimal scaling to the relit image, i.e. the scale that minimises squared error to the ground truth image. Our method generates plausible relighting results close the ground-truth image and produces the least error in most cases, while the other techniques struggle to preserve the high-frequency details, the colour-cast and the shading variations. For the method of Philip *et. al.* [122], we were able to obtain cross-relighting results only in specific cases since their sun-lighting model cannot be applied to cloud or evening skies. Only in one case, their method was able to outperform ours quantitatively. Please note that their method uses the full multi-view dataset for relighting whereas our method relights a single image.

5.9.4 Ablation Study

We show an analysis of several key design choices we make in our relighting framework. Figure 5.16 shows results from our neural rendering trained with and without the cross-projection loss. As evident from the figure, the cross-projection loss improves the colour expression and the photorealism of the relit images by preserving the underlying albedo more faithfully, while generating realistic

Method	Original lighting condition											
	1		2		3		4		5		6	
	ℓ_1	SSIM	ℓ_1	SSIM	ℓ_1	SSIM	ℓ_1	SSIM	ℓ_1	SSIM	ℓ_1	SSIM
Proposed	**0.077**	0.871	**0.078**	**0.850**	**0.074**	**0.876**	**0.075**	0.872	**0.076**	0.842	**0.073**	**0.839**
InverseRenderNet	0.082	0.824	0.085	0.780	0.087	0.791	0.083	0.818	0.079	0.819	0.077	0.810
[7]	0.083	**0.879**	0.097	0.826	0.091	0.852	0.080	**0.883**	0.086	0.840	0.098	0.814
[122]									0.095†	**0.871**	0.083‡	0.834

Table 5.3: Mean ℓ_1 colour error (lower is better) and SSIM index (higher is better) for relit images against cross projected ground-truth. Results are averaged across all images and all target lighting conditions. (†averaged over only target lighting condition 6 because the authors of method provided their results for only one target lighting condition.)(‡averaged over only target lighting conditions 2 & 5 for the same reason.)

Method	Benchmarking		BigTime	
	ℓ_1	SSIM	MSE	SSIM
Full	**0.079**	**0.856**	**0.021**	**0.760**
Without cross-project loss	0.082	0.836	0.023	0.744
Without cycle loss	0.097	0.853	0.026	0.729

Table 5.4: Quantitative evaluation of the ablated networks on the benchmarking data and the BigTime[86] time-lapse dataset

shading. Figure 5.17 shows a relighting result of our method with and without the input of *shadow map* generated by our shadow network to our neural renderer. Relighting result with the shadow network is able to generate naturally darker shadows. Without the target shadow map, the shadows in the relighting result tend to be lighter and hence appear fake. Figure 5.18 shows differences between rendering network trained by cycle consistency loss and without the cycle consistency loss. The rendering results from training with cycle loss show better shading effects and perform better on preserving the original reflectance colours and removing shadow from original input image. Figure 5.19 shows a relighting result of our method with and without the input of *residual input map* to our neural renderer. Our method is able to reconstruct sharper details than the network trained without residual input map. Figure 5.20 shows the utility of our SkyGAN network. As a baseline, we train the neural renderer to learn to generate the sky while relighting the scene, as opposed to having a dedicated SkyGAN for the task. To supervise the sky generation, we apply the same appearance loss over the sky pixels, as while training for the relighting task. Our SkyGAN results show better sky generation with realistic clouds and sky colour variations while the baseline is only able to generate a generic blue colour for the sky region.

To quantitatively validate our model against ablation methods, Table 5.4 shows that model trained

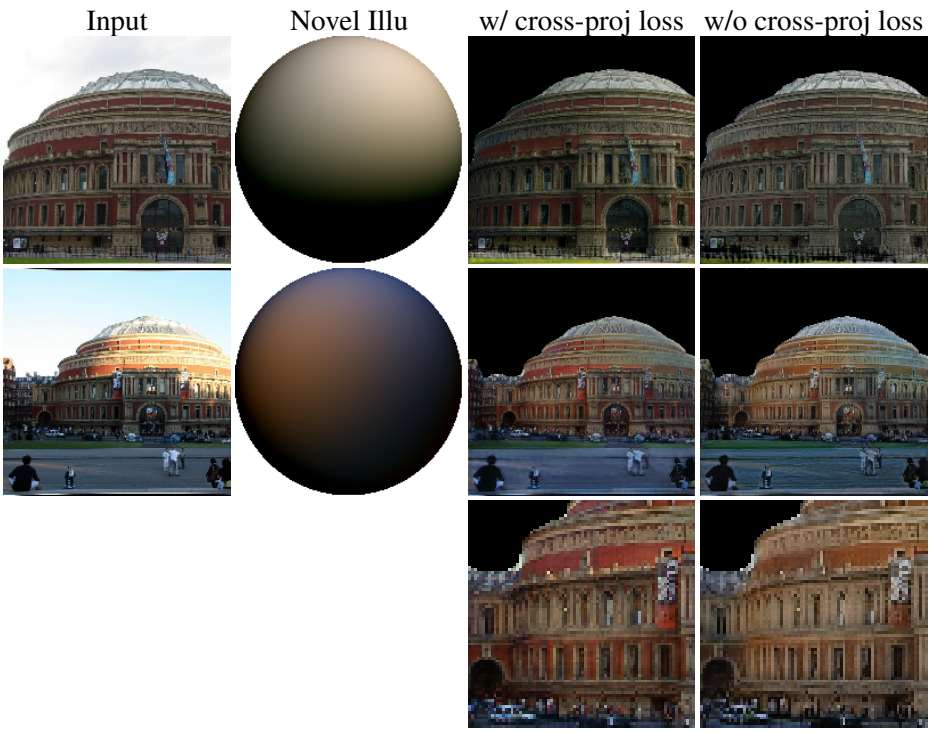


Figure 5.16: Performance between training with cross-projection loss and without cross-projection loss.

with both cross-projection loss and cycle-consistency loss outperforms others on both benchmarking data and time-lapse data.

5.10 Conclusion

In this chapter we have presented two scenarios of hybrid decoder and their advanced performance in applications. From the study of these two settings of hybrid decoder, it is shown that using the combination of model-based decoder and neural decoder can always produce superior results than using either one alone. Using a simple decoder based on classical geometry model and optimisation to merge the depth and normal maps, we obtain high quality geometry from a single image under demanding conditions. Training a neural decoder over a simple Lambertian renderer and inverse rendering networks, we relight a single image with target light condition to a novel rendering achieving a high level of photorealism.

To conclude this chapter, we have shown that the right blend of modelling and learning can dramatically improve results. A closed form normal/depth merging step significantly improves depth

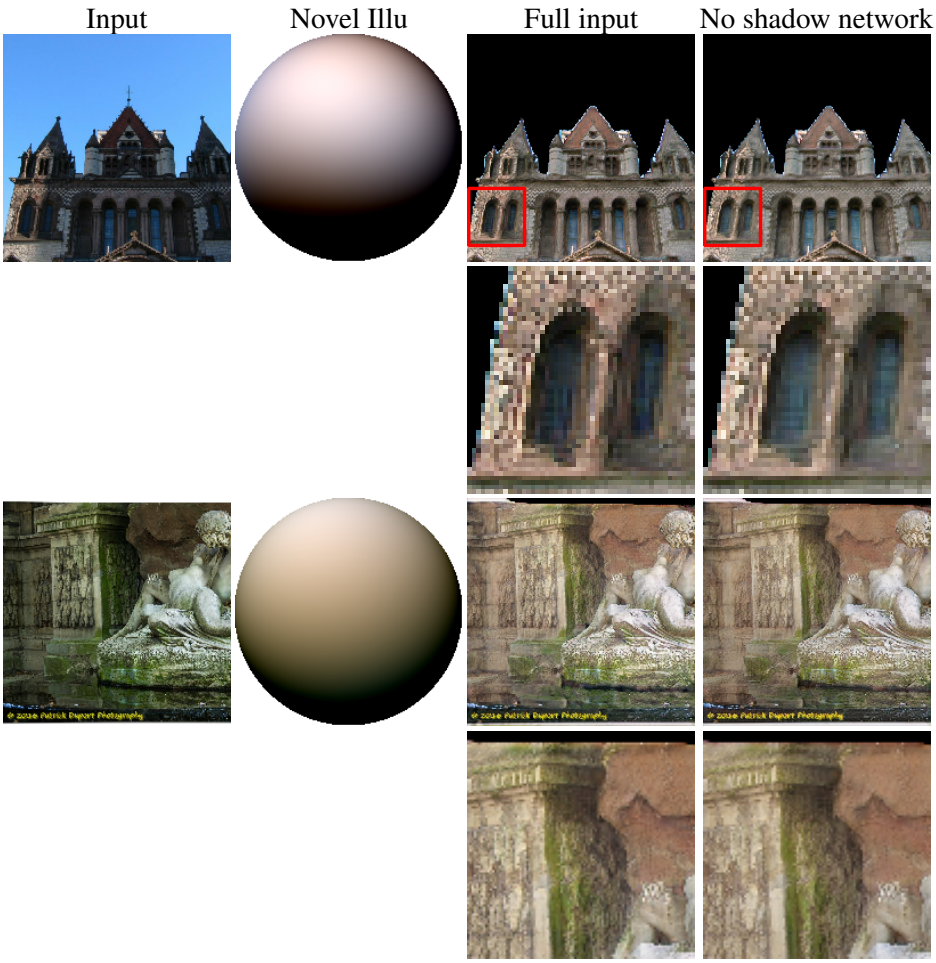


Figure 5.17: Performance comparison between training without shadow map given from shadow network and our full input.

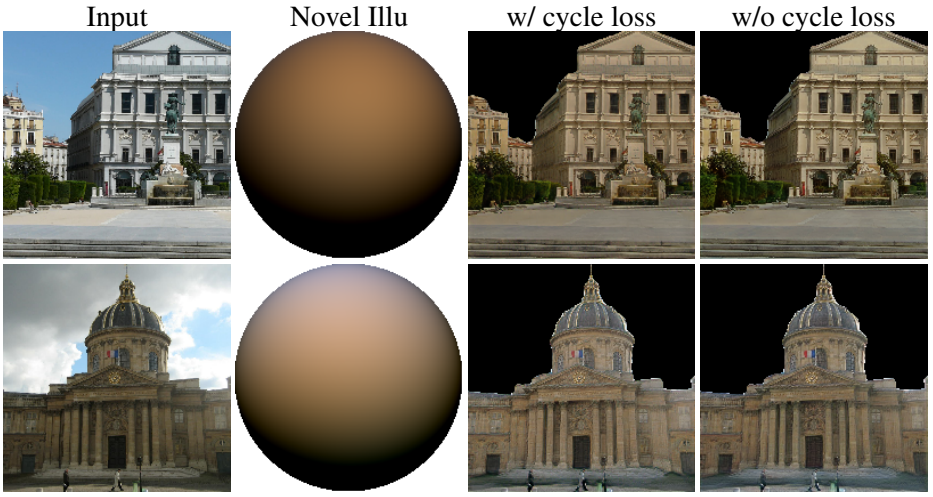


Figure 5.18: Performance comparison between training with cycle consistency loss and without cycle consistency loss.

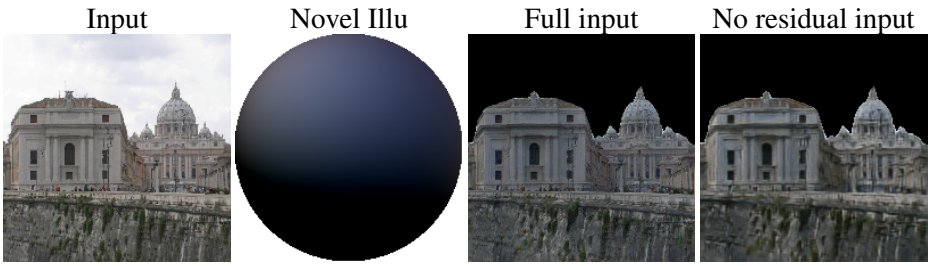


Figure 5.19: Performance comparison between training without residual input map and our full input.

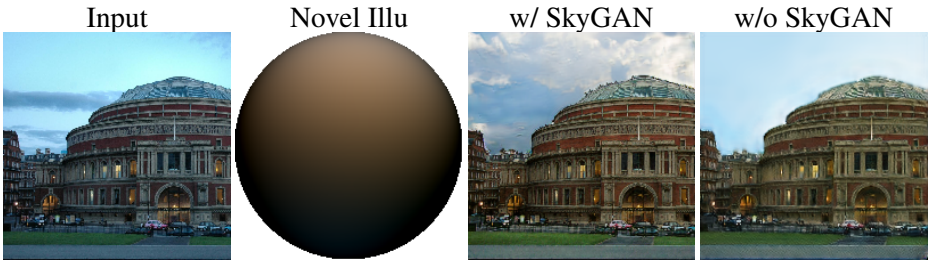


Figure 5.20: Performance comparison between training with SkyGAN and training directly learn sky by appearance loss.

maps with no additional training data and no learnable parameters. Here the depth and normal estimation seem to learn different geometric properties that are complementary and improved when combined. Following this assumption, estimating both depth and normal by shared networks that implicitly learns the conditional relationship could produce better estimates. Next, a neural renderer can do vastly better than a Lambertian fixed function renderer alone. It is shown that cross projection and perform relighting by shuffling lighting estimates is a very useful source of supervision for learning relighting. The image relighting problem can be well tackled when formulated as sub-problems, like inverse rendering, shadow estimation, relighting rendering and sky generation, and we resort to a particular neural network for learning each sub-problem. However, like InverseRenderNet the relighting neural renderer also suffers from limitations stemming from physical models involved, that is, the illumination is restricted to be low-order spherical harmonics, and inaccurate correspondence in MVS makes the learnt model cannot always produce sharp results. Besides, there exists a couple of challenging scenarios. First, shadows handling issue persists in both estimation and removal stages and re-synthesis processes, especially strong cast shadows. Second, dark images or night scenes remain a challenging problem. Finally, greatly specular surfaces like lakes or mirror walls are still an ill-posed problem for our relighting network.

Chapter 6

Conclusion

6.1 Summaries and conclusions

In this thesis, we explore the model-based decoder and its usability and scalability in deep learning methods. We propose different model-based decoders based on physical models. We study different applications including BRDF estimation, inverse rendering, fine depth prediction and image relighting with the power of model-based decoders. A differentiable rendering model facilitates a BRDF estimation network being trained by self-supervision on unlabelled data. A Lambertian model and cross-projection multivew stereo model enable us to train inverse rendering networks by self-supervision on unlabelled and unstructured data. A linear model relating depth and normal helps us emboss details onto deep depth estimates from deep normal estimates. A neural renderer on top of a Lambertian renderer can yield realistic relighting renderings that capture Lambertian reflectances as well as non-Lambertian visual effects.

By analyses on different applications, we can draw the following conclusions on model-based decoders. Integrating model-based decoders into neural networks is a promising research direction and could be the next step for the development of deep learning. Model-based decoders show great promise by combining our prior knowledge from the classical approaches and strong modelling capability of neural networks. It circumvents the heavy demands on ground truth label data, whilst some new constraints inherent to the employed models are included. For example, InverseRenderNet and

neural relighting network built on MVS model need to use multiview stereo images in training, and a fine depth prediction pipeline can only accept and produce depth map rather than other geometric forms. Another key feature of our embedded physical model in neural networks is to encapsulate physical interpretability in black-box model, by which InverseRenderNet can leverage self-rendering signal to hallucinate plausible predictions without dense and fine supervision and relighting renderer is empowered to control relighting results by explicit physical parameters. Despite showing promising usability, existing model-based decoders are only well-behaved under simple assumptions, which hinders their performance on generic and realistic scenarios, since these models have been significantly simplified to enable forward flow and backpropagation. As for more sophisticated physical models, how to approximate their behaviours by differentiable operations is still an open question.

6.2 Future Work

We discuss future work based on the presented applications. This can help us conclude the future work for the general topic of model-based decoders.

6.2.1 Deep BRDF estimate

In our BRDF estimation network, the renderer only models the interaction between the local illumination and the target surface, but there are no considerations on occlusion, global illumination and inter-reflections etc. Extending our differentiable renderer to operate on these graphics phenomenons enable us to handle more complex scenes. Also, we should extend the support for perspective projection to make our renderer usable in most common cases. A more interesting extension is to take multiple images as input to provide more observations between different incident lighting and oriented surfaces, such that the network can approach harder problems like SVBRDF estimation or doing estimation without geometry constraints. In terms of the BRDF statistical model employed, we could learn a network to capture a new model from the dataset [99]. The efficiency of the renderer can also be optimised by introducing other lighting representations rather than using point light sources.

6.2.2 Deep inverse rendering

There are many promising ways in which this work can be extended. First, our modelling assumptions could be relaxed, for example using more general reflectance models and estimating global illumination effects such as inter-reflection. Second, our network could benefit from losses used in training intrinsic image decomposition networks [83]. Third, our lighting prior could be extended to better handle indoor scenes. Fourth, our fixed reflectance and illumination model could be made partially learnable in order to be able to better explain real world appearance [131]. Finally, the network can be extended to learn inverse rendering on pedestrians in the foreground, which could improve inverse rendering results by holistically modelling the full image including both transient pedestrians and static objects.

6.2.3 Fine depth prediction and novel view synthesis

As for estimating depth maps containing details, the most obvious future work is to train the two networks simultaneously. Since the merging process involves only the solution of a linear least squares system, this could be done within the network during training. Recently, Zamir *et al*. [167] presented an insightful study of simultaneously training multi-task networks in a more generic scenario, where the results from different networks are merged by learnt models rather than physics-based models admitted by our prior knowledge. Such learnt models are able to find the hidden transition functions between different domains, for example the relationship between depth and normal can be learnt by a shallow neural network which could potentially perform better than our simplified linear model. Alternatively, one could consider estimating only depth but using inverse rendering losses such that the surface normals of the depth map better capture high frequency detail. This would require a method for estimating calibration parameters, however. Another obvious direction, relating to novel view synthesis, would be to introduce adversarial networks to enhance the quality of the synthesised views. Clearly, our results lack background and sky which a GAN may be able to synthesise realistically, while retaining semantic control over the pose and lighting of the scene.

6.2.4 Neural relighting renderer

An interesting way of extending this work would be training the whole pipeline in an end-to-end manner, which we expect to further optimise each constituting network so as to improve relighting results, or make the network implicitly aware of the 3D scene geometry hence reasoning better shadow casting effects. Alternatively, we could feed the depth map that conveys richer geometry information than normals into the relighting pipeline. With regard to illumination, our spherical harmonic lighting is sub-optimal for generating strong and sharp shadows, therefore one possible extension is to use the illumination being a union of directional lighting and spherical harmonics lighting. More ideally, our relighting pipeline should work on HDR environment maps either by directly taking it as input or encoding it into a lighting embedding. This neural rendering framework based on self-supervision from casual photography can also be extended in the future to lighting augmentation tasks such as addition or removal of existing light sources in the scene, opening up interesting applications in augmented and virtual reality domain.

Bibliography

[1] *HDRI-Skies*, 2020.

[2] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pages 265–283, 2016.

[3] LLC Agisoft. *Agisoft Metashape*. Saint Petersburg, Russia, 2019.

[4] Miika Aittala, Timo Aila, and Jaakko Lehtinen. Reflectance modeling by neural texture synthesis. *ACM Transactions on Graphics (TOG)*, 35(4):65, 2016.

[5] Oswald Aldrian and William AP Smith. Inverse rendering of faces with a 3D morphable model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5):1080–1093, 2012.

[6] N. Alldrin, T. Zickler, and D. Kriegman. Photometric stereo with non-parametric and spatially-varying reflectance. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008.

[7] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1670–1687, 2014.

[8] Anil S. Baslamisli, Hoang-An Le, and Theo Gevers. CNN based learning using reflection and retinex models for intrinsic image decomposition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[9] Sean Bell, Kavita Bala, and Noah Snavely. Intrinsic images in the wild. *ACM Trans. on Graphics (SIGGRAPH)*, 33(4), 2014.

[10] Moshe Ben-Ezra, Jiaping Wang, Bennett Wilburn, Xiaoyang Li, and Le Ma. An LED-only BRDF measurement device. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[11] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-farley, and Yoshua Bengio. Theano: A CPU and GPU math compiler in Python. In *Proc. Python in Science Conference*, 2010.

[12] Sema Berkiten, Xinyi Fan, and Szymon Rusinkiewicz. Merge2-3D: Combining multiple normal maps with 3D surfaces. *IEEE International Conference on 3D Vision (3DV)*, pages 440–447, December 2014.

[13] Sai Bi, Nima Khademi Kalantari, and Ravi Ramamoorthi. Deep Hybrid Real and Synthetic Training for Intrinsic Decomposition. In Wenzel Jakob and Toshiya Hachisuka, editors, *Eurographics Symposium on Rendering - Experimental Ideas & Implementations*. The Eurographics Association, 2018.

[14] James F Blinn. Models of light reflection for computer synthesized pictures. In *Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pages 192–198, 1977.

[15] Qifeng Chen and Vladlen Koltun. A simple model for intrinsic image decomposition with depth cues. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 241–248, 2013.

[16] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In *Advances in Neural Information Processing Systems*, pages 730–738, 2016.

[17] Forrester Cole, David Belanger, Dilip Krishnan, Aaron Sarna, Inbar Mosseri, and William T Freeman. Synthesizing normalized faces from facial identity features. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3703–3712, 2017.

[18] Kristin J Dana, Bram Van Ginneken, Shree K Nayar, and Jan J Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions On Graphics (TOG)*, 18(1):1–34, 1999.

[19] Paul Debevec. Image-based lighting. *IEEE Comput. Graph. Appl.*, 22(2):26–34, March 2002.

[20] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *Proceedings of SIGGRAPH 2000*, SIGGRAPH '00, 2000.

[21] Alexey Dosovitskiy and Thomas Brox. Inverting convolutional networks with convolutional networks. *CoRR*, abs/1506.02753, 2015.

[22] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015.

[23] Ron O Dror, Thomas K Leung, Edward H Adelson, and Alan S Willsky. Statistics of real-world illumination. In *Proc. CVPR*, 2001.

[24] Sylvain Duchêne, Clement Riant, Gaurav Chaurasia, Jorge Lopez Moreno, Pierre-Yves Laffont, Stefan Popov, Adrien Bousseau, and George Drettakis. Multiview intrinsic images of outdoors scenes with an application to relighting. *ACM Trans. Graph.*, 34(5):164:1–164:16, November 2015.

[25] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[26] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374, 2014.

[27] Qingnan Fan, Jiaolong Yang, Gang Hua, Baoquan Chen, and David Wipf. Revisiting deep intrinsic image decompositions. In *Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8944–8952, 2018.

[28] Sing Choong Foo. *A gonioreflectometer for measuring the bidirectional reflectance of material for use in illumination computation*. PhD thesis, CiteSeerX, 1997.

[29] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[30] Martin Fuchs, Volker Blanz, Hendrik P.A. Lensch, and Hans-Peter Seidel. Adaptive sampling of reflectance fields. *ACM Trans. Graph.*, 26(2), June 2007.

[31] Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1362–1376, August 2010.

[32] Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deep inverse rendering for high-resolution SVBRDF estimation from an arbitrary number of images. *ACM Trans. Graph.*, 38(4):134–1, 2019.

[33] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.

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

[35] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[36] Stamatios Georgoulis, Konstantinos Rematas, Tobias Ritschel, Mario Fritz, Luc J. Van Gool, and Tinne Tuytelaars. Delight-net: Decomposing reflectance maps into specular materials and natural illumination. *CoRR*, abs/1603.08240, 2016.

[37] Abhijeet Ghosh, Shruthi Achutha, Wolfgang Heidrich, and Matthew O'Toole. BRDF acquisition with basis illumination. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007.

[38] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.

[39] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth prediction. In *The International Conference on Computer Vision (ICCV)*, October 2019.

[40] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. Multi-view stereo for community photo collections. *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8, 2007.

[41] Dan B Goldman, Brian Curless, Aaron Hertzmann, and Steven M Seitz. Shape and spatially-varying BRDFs from photometric stereo. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(6):1060–1071, 2010.

[42] Gene H Golub and Victor Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on numerical analysis*, 10(2):413–432, 1973.

[43] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. NIPS*, pages 2672–2680, 2014.

[44] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623, 2015.

[45] Tom Haber, Christian Fuchs, Philippe Bekaer, Hans-Peter Seidel, Michael Goesele, and Hendrik PA Lensch. Relighting objects from image collections. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 627–634. IEEE, 2009.

[46] Guangyun Han, Xiaohua Xie, Jianhuang Lai, and Wei-Shi Zheng. Learning an intrinsic image decomposer using synthesized RGB-D dataset. *IEEE Signal Processing Letters*, 25(6):753–757, 2018.

[47] Ankur Handa, Michael Blösch, Viorica Patraucean, Simon Stent, John McCormac, and Andrew J. Davison. gvnn: Neural network library for geometric computer vision. *CoRR*, abs/1607.07405, 2016.

[48] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[49] 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*, pages 770–778, 2016.

[50] Geoffrey Hinton, Alex Krizhevsky, and Sida Wang. Transforming auto-encoders. *Proc. ICANN*, pages 44–51, 2011.

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

[52] Zhuo Hui and Aswin C Sankaranarayanan. A dictionary-based approach for estimating shape and spatially-varying reflectance. In *Proc. ICCP*, pages 1–9, 2015.

[53] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[54] Michael Janner, Jiajun Wu, Tejas D Kulkarni, Ilker Yildirim, and Josh Tenenbaum. Self-supervised intrinsic image decomposition. In *Advances in Neural Information Processing Systems*, pages 5936–5946, 2017.

[55] Simon Jegou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[56] Junho Jeon, Sunghyun Cho, Xin Tong, and Seungyong Lee. Intrinsic image decomposition using structure-texture separation and surface normals. In *European Conference on Computer Vision*, pages 218–233. Springer, 2014.

[57] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. SurfaceNet: An end-to-end 3D neural network for multiview stereopsis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2307–2315, 2017.

[58] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pages 694–711, 2016.

[59] James T Kajiya. The rendering equation. In *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, pages 143–150, 1986.

[60] Yoshihiro Kanamori and Yuki Endo. Relighting humans: occlusion-aware inverse rendering for full-body human images. *ACM Transactions on Graphics (Proc. of SIGGRAPH Asia)*, 37(6):270, 2018.

[61] Kevin Karsch, Varsha Hedau, David Forsyth, and Derek Hoiem. Rendering synthetic objects into legacy photographs. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, SA '11, pages 157:1–157:12, New York, NY, USA, 2011. ACM.

[62] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *arXiv preprint arXiv:2006.12057*, 2020.

[63] 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.

[64] Michael Kazhdan and Hugues Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graph.*, 32(3):29:1–29:13, July 2013.

[65] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. *CoRR, vol. abs/1703.04309*, 2017.

[66] Hyeongwoo Kim, Pablo Garrido, Ayush Tewari, Weipeng Xu, Justus Thies, Matthias Niessner, Patrick Pérez, Christian Richardt, Michael Zollhöfer, and Christian Theobalt. Deep video portraits. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[67] Hyeongwoo Kim, Michael Zollhöfer, Ayush Tewari, Justus Thies, Christian Richardt, and Christian Theobalt. Inversefacenet: Deep single-shot inverse face rendering from A single image. *CoRR*, abs/1703.10956, 2017.

[68] Kichang Kim, Akihiko Torii, and Masatoshi Okutomi. Multi-view inverse rendering under arbitrary illumination and albedo. In *European Conference on Computer Vision*, pages 750–767. Springer, 2016.

[69] Kihwan Kim, Jinwei Gu, Stephen Tyree, Pavlo Molchanov, Matthias Nießner, and Jan Kautz. A lightweight approach for on-the-fly reflectance estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 20–28, 2017.

[70] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[71] Johannes Kopf, Boris Neubert, Billy Chen, Michael Cohen, Daniel Cohen-Or, Oliver Deussen, Matt Uyttendaele, and Dani Lischinski. Deep photo: Model-based photograph enhancement and viewing. *ACM Trans. Graph.*, 27(5):116:1–116:10, December 2008.

[72] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[73] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. *Advances in neural information processing systems*, 28:2539–2547, 2015.

[74] HDR Labs. *sIBL Archive*, 2007–2012.

[75] Pierre-Yves Laffont, Adrien Bousseau, Sylvain Paris, Frédo Durand, and George Drettakis. Coherent intrinsic images from photo collections. *ACM Trans. Graph.*, 31(6):202:1–202:11, November 2012.

[76] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 239–248. IEEE, 2016.

[77] Jean-François Lalonde, Alexei A. Efros, and Srinivasa G. Narasimhan. Webcam clip art: Appearance and illuminant transfer from time-lapse sequences. In *ACM SIGGRAPH Asia 2009 Papers*, SIGGRAPH Asia '09, pages 131:1–131:10, New York, NY, USA, 2009. ACM.

[78] Jean-François Lalonde, Alexei A. Efros, and Srinivasa G. Narasimhan. Estimating the natural illumination conditions from a single outdoor image. *International Journal of Computer Vision*, 98(2):123–145, Jun 2012.

[79] Fabian Langguth. Photometric stereo for outdoor webcams. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 262–269, Washington, DC, USA, 2012. IEEE Computer Society.

[80] Louis Lettry, Kenneth Vanhoey, and Luc Van Gool. Darn: a deep adversarial residual network for intrinsic image decomposition. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1359–1367. IEEE, 2018.

[81] Xiao Li, Yue Dong, Pieter Peers, and Xin Tong. Modeling surface appearance from a single photograph using self-augmented convolutional neural networks. *ACM Transactions on Graphics (TOG)*, 36(4):45, 2017.

[82] Zhengqi Li and Noah Snavely. CGIntrinsics: Better intrinsic image decomposition through physically-based rendering. In *European Conference on Computer Vision (ECCV)*, 2018.

[83] Zhengqi Li and Noah Snavely. Learning intrinsic image decomposition from watching the world. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[84] Zhengqi Li and Noah Snavely. MegaDepth: Learning single-view depth prediction from internet photos. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

[85] 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.

[86] Zhengqin Li, Zexiang Xu, Ravi Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Learning to reconstruct shape and spatially-varying reflectance from a single image. *ACM Transactions on Graphics (TOG)*, 37(6):1–11, 2018.

[87] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5162–5170, 2015.

[88] Guilin Liu, Duygu Ceylan, Ersin Yumer, Jimei Yang, and Jyh-Ming Lien. Material editing using a physically based rendering network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2261–2269, 2017.

[89] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7708–7717, 2019.

[90] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015.

[91] Matthew M Loper and Michael J Black. OpenDR: An approximate differentiable renderer. In *Proc. ECCV*, pages 154–169. Springer, 2014.

[92] Céline Loscos, Marie-Claude Frasson, George Drettakis, Bruce Walter, Xavier Granier, and Pierre Poulin. Interactive virtual relighting and remodeling of real scenes. In Dani Lischinski and Greg Ward Larson, editors, *Rendering Techniques' 99*, pages 329–340, Vienna, 1999. Springer Vienna.

[93] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6997–7005, July 2017.

[94] Wei-Chiu Ma, Hang Chu, Bolei Zhou, Raquel Urtasun, and Antonio Torralba. Single image intrinsic decomposition without a single intrinsic image. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 201–217, 2018.

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

[96] Stephen R. Marschner, Stephen H. Westin, Eric P. F. Lafortune, Kenneth E. Torrance, and Donald P. Greenberg. Image-based BRDF measurement including human skin. In Dani Lischinski and Greg Ward Larson, editors, *Proc. Rendering Techniques*, pages 131–144, Vienna, 1999. Springer Vienna.

[97] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, Adarsh Kowdle, Christoph Rhemann, Dan B Goldman, Cem Keskin, Steve Seitz, Shahram Izadi, and Sean Fanello. Lookingood: Enhancing performance capture with real-time neural re-rendering. *ACM Trans. Graph.*, 37(6):255:1–255:14, December 2018.

[98] Vincent Masselus, Pieter Peers, Philip Dutré, and Yves D. Willems. Relighting with 4d incident light fields. *ACM Trans. Graph.*, 22(3):613–620, July 2003.

[99] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 22(3):759–769, July 2003.

[100] Wojciech Matusik, Hanspeter Pfister, Matthew Brand, and Leonard McMillan. Efficient isotropic BRDF measurement. In *Proc. Eurographics Workshop on Rendering*, EGRW '03, pages 241–247, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[101] Abhimitra Meka, Gereon Fox, Michael Zollhöfer, Christian Richardt, and Christian Theobalt. Live user-guided intrinsic video for static scenes. *IEEE Transactions on Visualization and Computer Graphics*, 23(11):2447–2454, 2017.

[102] Abhimitra Meka, Christian Häne, Rohit Pandey, Michael Zollhöfer, Sean Fanello, Graham Fyffe, Adarsh Kowdle, Xueming Yu, Jay Busch, Jason Dourgarian, Peter Denny, Sofien Bouaziz, Peter Lincoln, Matt Whalen, Geoff Harvey, Jonathan Taylor, Shahram Izadi, Andrea Tagliasacchi, Paul Debevec, Christian Theobalt, Julien Valentin, and Christoph Rhemann. Deep reflectance fields: High-quality facial reflectance field inference from color gradient illumination. *ACM Trans. Graph.*, 38(4):77:1–77:12, July 2019.

[103] Abhimitra Meka, Maxim Maximov, Michael Zollhoefer, Avishek Chatterjee, Hans-Peter Seidel, Christian Richardt, and Christian Theobalt. LIME: Live intrinsic material estimation. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[104] Abhimitra Meka, Michael Zollhöfer, Christian Richardt, and Christian Theobalt. Live intrinsic video. *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, 35(4), 2016.

[105] Moustafa Mahmoud Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Kumar Pandey, Noah Snavely, and Ricardo Martin Brualla. Neural rerendering in the wild. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

[106] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[107] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[108] Nikhil Naik, Shuang Zhao, Andreas Velten, Ramesh Raskar, and Kavita Bala. Single view reflectance capture using multiplexed scattering and time-of-flight imaging. In *Proceedings of the 2011 SIGGRAPH Asia Conference*, pages 1–10, 2011.

[109] Oliver Nalbach, Elena Arabadzhiyska, Dushyant Mehta, Hans-Peter Seidel, and Tobias Ritschel. Deep shading: Convolutional neural networks for screen-space shading. 36(4), 2017.

[110] Seonghyeon Nam, Chongyang Ma, Menglei Chai, William Brendel, Ning Xu, and Seon Joo Kim. End-to-end time-lapse video synthesis from a single outdoor image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1409–1418, 2019.

[111] Takuya Narihira, Michael Maire, and Stella X Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *Proceedings of the IEEE international conference on computer vision*, pages 2992–2992, 2015.

[112] Diego Nehab, Szymon Rusinkiewicz, James Davis, and Ravi Ramamoorthi. Efficiently combining positions and normals for precise 3D geometry. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 24(3):536–543, 2005.

[113] Thomas Nestmeyer and Peter V Gehler. Reflectance adaptive filtering improves intrinsic image estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, page 4, 2017.

[114] Thomas Nestmeyer, Jean-FranÃ§ois Lalonde, Iain Matthews, and Andreas M Lehrmann. Learning physics-guided face relighting under directional light. In *Conference on Computer Vision and Pattern Recognition*, pages 5123–5132. IEEE/CVF, June 2020.

[115] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *Proc. ECCV*, pages 483–499, 2016.

[116] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental analysis of BRDF models. *Rendering Techniques*, 2005(16th):2, 2005.

[117] Jannik Boll Nielsen, Henrik Wann Jensen, and Ravi Ramamoorthi. On optimal, minimal BRDF sampling for reflectance acquisition. *ACM Transactions on Graphics (TOG)*, 34(6):186:1–186:11, November 2015.

[118] Makoto Okabe, Gang Zeng, Yasuyuki Matsushita, Takeo Igarashi, Long Quan, and Heung-Yeung Shum. Single-view relighting with normal map painting. In *Proceedings of pacific graphics*, pages 27–34. Citeseer, 2006.

[119] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

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

[121] Pieter Peers, Naoki Tamura, Wojciech Matusik, and Paul Debevec. Post-production facial performance relighting using reflectance transfer. *ACM Trans. Graph.*, 26(3), July 2007.

[122] Julien Philip, Michaël Gharbi, Tinghui Zhou, Alexei A Efros, and George Drettakis. Multi-view relighting using a geometry-aware network. *ACM Transactions on Graphics (TOG)*, 38(4):1–14, 2019.

[123] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *2016 International Conference on Learning Representations (ICLR)*, 11 2016.

[124] 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.

[125] Konstantinos Rematas, Tobias Ritschel, Mario Fritz, Efstratios Gavves, and Tinne Tuytelaars. Deep reflectance maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4508–4516, 2016.

[126] Elad Richardson, Matan Sela, Roy Or-El, and Ron Kimmel. Learning detailed face reconstruction from a single image. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1259–1268, 2017.

[127] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015.

[128] Szymon M Rusinkiewicz. A new change of variables for efficient BRDF representation. In *Rendering techniques' 98*, pages 11–22. Springer, 1998.

[129] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

[130] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[131] Soumyadip Sengupta, Jinwei Gu, Kihwan Kim, Guilin Liu, David W Jacobs, and Jan Kautz. Neural inverse rendering of an indoor scene from a single image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8598–8607, 2019.

[132] Soumyadip Sengupta, Angjoo Kanazawa, Carlos D Castillo, and David W Jacobs. SfSNet: Learning shape, reflectance and illuminance of faces 'in the wild'. *arXiv preprint arXiv:1712.01261*, 2017.

[133] Qi Shan, Riley Adams, Brian Curless, Yasutaka Furukawa, and Steven M. Seitz. The visual turing test for scene reconstruction. In *Proceedings of the 2013 International Conference on 3D Vision*, 3DV '13, pages 25–32, Washington, DC, USA, 2013. IEEE Computer Society.

[134] Evan Shelhamer, Jonathan T Barron, and Trevor Darrell. Scene intrinsics and depth from a single image. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 37–44, 2015.

[135] Jian Shi, Yue Dong, Hao Su, and X Yu Stella. Learning non-lambertian object intrinsics across ShapeNet categories. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5844–5853. IEEE, 2017.

[136] Yichang Shih, Sylvain Paris, Frédo Durand, and William T. Freeman. Data-driven hallucination of different times of day from a single outdoor photo. *ACM Trans. Graph.*, 32(6):200:1–200:11, November 2013.

[137] Zhixin Shu, Sunil Hadap, Eli Shechtman, Kalyan Sunkavalli, Sylvain Paris, and Dimitris Samaras. Portrait lighting transfer using a mass transport approach. *ACM Trans. Graph*, 36(4), 2017.

[138] Zhixin Shu, Ersin Yumer, Sunil Hadap, Kalyan Sunkavalli, Eli Shechtman, and Dimitris Samaras. Neural face editing with intrinsic image disentangling. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 5444–5453. IEEE, 2017.

[139] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[140] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[141] Jessi Stumpfel, Andrew Jones, Andreas Wenger, Chris Tchou, Tim Hawkins, and Paul Debevec. Direct hdr capture of the sun and sky. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[142] Tiancheng Sun, Jonathan T. Barron, Yun-Ta Tsai, Zexiang Xu, Xueming Yu, Graham Fyffe, Christoph Rhemann, Jay Busch, Paul Debevec, and Ravi Ramamoorthi. Single image portrait relighting. *ACM Trans. Graph.*, 38(4):79:1–79:12, July 2019.

[143] Kalyan Sunkavalli, Wojciech Matusik, Hanspeter Pfister, and Szymon Rusinkiewicz. Factored time-lapse video. In *ACM SIGGRAPH 2007 Papers*, SIGGRAPH '07, New York, NY, USA, 2007. ACM.

[144] Yichuan Tang, Ruslan Salakhutdinov, and Geoffrey Hinton. Deep Lambertian networks. *arXiv preprint arXiv:1206.6445*, 2012.

[145] Chris Tchou, Jessi Stumpfel, Per Einarsson, Marcos Fajardo, and Paul Debevec. Unlighting the parthenon. In *ACM SIGGRAPH 2004 Sketches*, SIGGRAPH '04, New York, NY, USA, 2004. ACM.

[146] Ayush Tewari, Michael Zollhöfer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Pérez, and Christian Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *The IEEE International Conference on Computer Vision (ICCV)*, volume 2, page 5, 2017.

[147] Justus Thies, Michael Zollhöfer, and Matthias Nießner. Deferred neural rendering: Image synthesis using neural textures. *ACM Transactions on Graphics 2019 (TOG)*, 2019.

[148] Alejandro Troccoli and Peter Allen. Building illumination coherent 3D models of large-scale outdoor scenes. *International Journal of Computer Vision*, 78(2):261–280, Jul 2008.

[149] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, volume 1, page 3, 2017.

[150] Hsiao-Yu Fish Tung, Adam W Harley, William Seto, and Katerina Fragkiadaki. Adversarial inverse graphics networks: Learning 2d-to-3d lifting and image-to-image translation from unpaired supervision. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4364–4372. IEEE, 2017.

[151] Borom Tunwattanapong, Graham Fyffe, Paul Graham, Jay Busch, Xueming Yu, Abhijeet Ghosh, and Paul Debevec. Acquiring reflectance and shape from continuous spherical harmonic illumination. *ACM Transactions on graphics (TOG)*, 32(4):1–12, 2013.

[152] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on computer vision and pattern recognition (CVPR)*, volume 5, page 6, 2017.

[153] Igor Vasiljevic, Nick Kolkin, Shanyi Zhang, Ruotian Luo, Haochen Wang, Falcon Z. Dai, Andrea F. Daniele, Mohammadreza Mostajabi, Steven Basart, Matthew R. Walter, and Gregory Shakhnarovich. DIODE: A Dense Indoor and Outdoor DEpth Dataset. *CoRR*, abs/1908.00463, 2019.

[154] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. Sfm-net: Learning of structure and motion from video. *CoRR*, abs/1704.07804, 2017.

[155] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey. Learning depth from monocular videos using direct methods. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2022–2030, 2018.

[156] Peng Wang, Xiaohui Shen, Zhe Lin, Scott Cohen, Brian Price, and Alan L Yuille. Towards unified depth and semantic prediction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2800–2809, 2015.

[157] Gregory J Ward. Measuring and modeling anisotropic reflection. In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, pages 265–272, 1992.

[158] Andreas Wenger, Andrew Gardner, Chris Tchou, Jonas Unger, Tim Hawkins, and Paul Debevec. Performance relighting and reflectance transformation with time-multiplexed illumination. *ACM Trans. Graph.*, 24(3):756–764, 2005.

[159] D Rod White, Peter Saunders, Stuart J Bonsey, John van de Ven, and Hamish Edgar. Reflectometer for measuring the bidirectional reflectance of rough surfaces. *Applied optics*, 37(16):3450–3454, 1998.

[160] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[161] Jiajun Wu, Tianfan Xue, Joseph J Lim, Yuandong Tian, Joshua B Tenenbaum, Antonio Torralba, and William T Freeman. Single image 3D interpreter network. In *European Conference on Computer Vision (ECCV)*, 2016.

[162] Guanyu Xing, Xuehong Zhou, Qunsheng Peng, Yanli Liu, and Xueying Qin. Lighting simulation of augmented outdoor scene based on a legacy photograph. *Computer Graphics Forum*, 32(7):101–110, 2013.

[163] Dan Xu, Elisa Ricci, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Multi-scale continuous crfs as sequential deep networks for monocular depth estimation. In *Proceedings of CVPR*, volume 1, 2017.

[164] Zexiang Xu, Kalyan Sunkavalli, Sunil Hadap, and Ravi Ramamoorthi. Deep image-based relighting from optimal sparse samples. *ACM Trans. Graph.*, 37(4):126:1–126:13, July 2018.

[165] Yizhou Yu, Paul Debevec, Jitendra Malik, and Tim Hawkins. Inverse global illumination: recovering reflectance models of real scenes from photographs. In *Proc. SIGGRAPH*, pages 215–224, 1999.

[166] Yizhou Yu and Jitendra Malik. Recovering photometric properties of architectural scenes from photographs. In *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '98, pages 207–217, New York, NY, USA, 1998. ACM.

[167] Amir R Zamir, Alexander Sax, Nikhil Cheerla, Rohan Suri, Zhangjie Cao, Jitendra Malik, and Leonidas J Guibas. Robust learning through cross-task consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11197–11206, 2020.

[168] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. pages 649–666. Springer, 2016.

[169] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2881–2890, 2017.

[170] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. T2net: Synthetic-to-realistic translation for solving single-image depth estimation tasks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 767–783, 2018.

[171] Andrey Zhmoginov and Mark Sandler. Inverting face embeddings with convolutional neural networks. *CoRR*, abs/1606.04189, 2016.

[172] Hao Zhou, Sunil Hadap, Kalyan Sunkavalli, and David W Jacobs. Deep single-image portrait relighting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7194–7202, 2019.

[173] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, volume 2, page 7, 2017.

[174] Tinghui Zhou, Philipp Krähenbühl, and Alexei A Efros. Learning data-driven reflectance priors for intrinsic image decomposition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3469–3477, 2015.

[175] Jacek Zienkiewicz, Andrew Davison, and Stefan Leutenegger. Real-time height map fusion using differentiable rendering. In *Proc. IROS*, pages 4280–4287, 2016.

[176] Jasenko Zivanov, Pascal Paysan, and Thomas Vetter. Facial normal map capture using four lights–an effective and inexpensive method of capturing the fine scale detail of human faces using four point lights. In *Proc. GRAPP*, 2009.