This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

PX-NET: Simple and Efficient Pixel-Wise Training of Photometric Stereo Networks

Fotios Logothetis[1] Ignas Budvytis[2] Roberto Mecca[1] Roberto Cipolla[1,2]

[1]Cambridge Research Laboratory,Toshiba Europe Ltd.
Cambridge, UK
flogothetis,rmecca@crl.toshiba.co.uk

[2]University of Cambridge
Cambridge, UK
ib255,rc10001@cam.ac.uk

Abstract

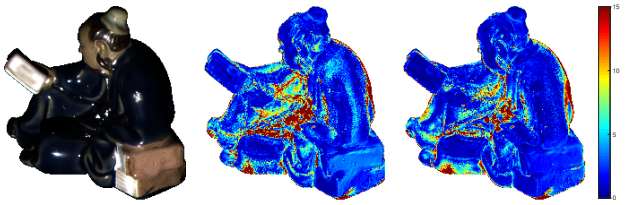
Retrieving accurate 3D reconstructions of objects from the way they reflect light is a very challenging task in computer vision. Despite more than four decades since the definition of the Photometric Stereo problem, most of the literature has had limited success when global illumination effects such as cast shadows, self-reflections and ambient light come into play, especially for specular surfaces. Recent approaches have leveraged the capabilities of deep learning in conjunction with computer graphics in order to cope with the need of a vast number of training data to invert the image irradiance equation and retrieve the geometry of the object. However, rendering global illumination effects is a slow process which can limit the amount of training data that can be generated.

In this work we propose a novel pixel-wise training procedure for normal prediction by replacing the training data (observation maps) of globally rendered images with independent per-pixel generated data. We show that global physical effects can be approximated on the observation map domain and this simplifies and speeds up the data creation procedure. Our network, PX-NET, achieves state-of-the-art performance compared to other pixelwise methods on synthetic datasets, as well as the DiLiGenT real dataset on both dense and sparse light settings.

1. Introduction

Photometric Stereo (PS) is a classical problem in computer vision since the early '80s [42]. PS assumes multiple images from the same viewpoint along with varied illumination and calculates local geometrical features (e.g. normal or depth) at each pixel by exploiting the relation between surface orientation and intensity of reflected light. This is essentially an inverse rendering problem requiring at least three input images in order to have a unique solution.

Most of the difficulty in retrieving the 3D shape from the light reflected off the object is due to the type of reflection and its non-linear dependence on material properties. This is mathematically expressed though the surface bidirectional reflectance distribution function (BRDF) which is determined by the material of the object. Over the last forty years a very wide spectrum of BRDF equations have been proposed to model the light reflection phenomena. Starting from the basic linear light response for diffuse reflection [21, 13], more specular behaviour of reflected light have been proposed [31, 3, 8, 20, 38, 40]. Comparison among numerous BRDFs can be found in [41, 11, 30, 29]. Finally, the recently proposed Disney BRDF [4, 12] was invented to unify most physical reflection effects including gloss reflection, subsurface scattering and metallic/specular roughness into a unified formulation.



'Reading' [16] MAE=12.6^o Proposed MAE=9.78^o

Figure 1. Comparison of the proposed approach versus [16] on 'Reading' of the DiLiGenT real benchmark [35]. The evaluation metric is the mean angular error (MAE) of the computed normal map compared with the ground truth.

The above mentioned advancements in computer graphics have enabled convolutional neural network (CNN)-based approaches to be useful for solving PS by rendering large number of images of various surfaces under numerous light and material configurations. They often parametrise the PS problem as normal regression from light intensity observations (i.e. observational map [16]), effectively performing an inversion of the irradiance equation. CNN-based approaches have been shown to outperform classical optimisation based methods [17, 32] mainly due to the ability of CNNs to learn how to deal with a great variety of real-

istic reflectances which lead classical optimisation methods into intractable computations and thus simplifications (e.g. assuming Lambertian reflection). In addition, CNNs can gain robustness to deviations from the irradiance equation such as global illumination effects (cast shadows, self reflections) if the training data includes them [16]. This can be achieved using 3D creation suite (like Blender [2]) which can render data containing that level of realism. However, exhaustively sampling global illumination effects (which are a function of the overall surface geometry) requires a huge number of meshes to be rendered. Furthermore, the rendering requirements grow exponentially if the rendered data are to be covering all the materials/lights configurations as well. The rendering computational cost can be reduced by rendering multi-material objects ([16]) to the detriment of realistic ray-traced self reflections. Finally, it is noted that rendering full objects is computationally expensive, therefore relatively slow and somehow inefficient, as there is a large amount of correlation among neighbouring pixels (especially for shadow/self reflection patterns).

In order to maximise the combinations of sampled materials, lights and normal directions, instead of pre-rendering training data, we train our CNN with highly efficient, independently generated pixelwise, observational maps. This allows to widen the training data variation as all of these parameters (e.g materials) can be sampled independently for every data point. Moreover, we show how global light effects can be approximated in the observational map domain, and propose a strategy that includes variations in the maps that model ambient light, cast shadows, self-reflections and reflectance mixing in discontinuity boundaries. This strategy helps to reduce the synthetic to real gap making our data applicable for challenging real data [35].

Contribution: Our CNN based approach for solving PS problem has the following main contribution: we propose a per-pixel observation map generation strategy which can replace slow-to-obtain full image rendering while still allowing the network to learn global illumination effects. Furthermore, we also propose an improvement to the CNN-PS [16] architecture termed PX-NET which benefits from the increase in the training data variation. Finally, we show that including the RGB channels in the observation map can further boost performance.

The rest of this work is divided as follows. Section 2 discusses the relevant literature. Section 3 provides details of our proposed CNN approach. Sections 4 and 5 describe the experiment setup and corresponding results.

2. Related Work

We now provide an overview of the latest PS improvements that mostly focus on deep learning approaches. For a fairly recent survey of PS techniques, refer to [1].

As Deep Learning (DL) has recently dominated the vast majority of ongoing research in computer vision, several approaches have been proposed for retrieving 3D geometry of objects using PS. The ability of DL-based approaches to approximate highly non-linear mappings is highly desired in inverting non-linear and complex irradiance equations. However, regressing very dense and accurate depth maps is not a trivial task. A preliminary study proposed by [36] used a Deep Belief Network [10] to tackle the Lambertian PS problem. Recently, some approaches have been presented addressing the PS problem with DL architectures. [43] built layers capable of modeling photometric image formation (in a unsupervised manner) that can be embedded into existent encoder-decoder architectures for establishing correspondences among light reflections and geometry. [33] proposed the first network to be trained using the real-world MERL database. They simplified the generality of the PS problem by training the network with the same light direction used for testing.

[18] used a learning procedure for separating the RGB information from multi-spectral images when solving the RGB-PS problem. This demultiplexing procedure allowed to improve the accuracy of the reconstruction when using a minimal number of light sources. [37] proposed an unsupervised method that does not require any training as they minimise the reconstruction loss between the rendered images and the input images at test time. This makes the approach slower with respect to usual DL based methods as the training computational time is partially transferred to the shape reconstruction pipeline. [7, 6] introduced PS-FCN, the first deep learning based approach not requiring identical set of lights at train and test time. Indeed, by using similar concepts, [5] proposed a DL approach a more challenging scenario for the PS problem where the light sources are unknown. In this case, a two-stage modeling is used to approximate first the uniform light directions (LC-Net) and then estimate the normals (NENet). [44] proposed SplineNet to solve the Sparse PS (e.g. PS with low number of images). They employed an interpolation network to estimate the reflectance at additional light directions. Another approach to solve Sparse PS has been presented by [39] where the lack of information due to limited amount of input images is compensated by training the network with 9x9 patches of pixels. In addition, they enforce a collocated light constrain which is derived empirically from observations in the MERL material database. [9] is able to deal with sparse configuration of lights by search and match in a set of diverse BRDFs.

As DL has been shown to be an efficient tool for solving PS in traditional setting, that is having directional light sources and orthographic viewing geometry, recent works [23, 34] exploited a CNN based approach to retrieve geometry using photometric imaging in a constrained near-field setting. Finally, [16] introduced the observation map

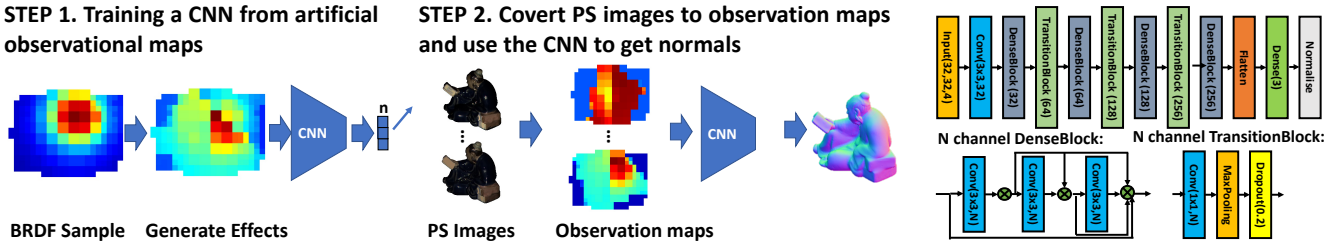


Figure 2. This figure illustrates the two key steps of our proposed approach. On the left, the network training is illustrated consisting of sampling material BRDFs and then generating synthetic observation maps (3.2) which includes modeling of global illumination and other realistic effects. In the middle, the normal estimation process is shown - for each pixel, the observation map is computed by combining the information from all PS images into a single tensor (3.1). The maps are then processed by a CNN which regresses a normal (orientation) map. The complete CNN architecture is shown on the right. ReLU activation is used after each CONV layer. $\bigotimes$ denotes concatenation along the channel axis.

parameterisation (32 by 32 gray-scale image) that merges information of multiple lights on a single tensor allowing a fixed network to be used under a varied number of light sources. The training data was obtained by rendering 15 meshes with a dense variation of material properties under a number of light directions. The purpose of this pre-rendered training dataset was to allow the network to learn the effect of global physical phenomena synthetically generated with computer graphics. Although a large amount of data was sampled, the choice of selecting specific meshes limits the possible light-normal-material configurations and constrains the patterns of the global illumination effects (cast shadows, self reflections) which are a direct function of the global surface geometry. In addition, training on purely synthetic images without data augmentation is prone to over-fitting to the synthetic distribution with potential drop of performance in real images.

In order to overcome these limitations, we propose a CNN-based approach for the PS problem with a better coverage of physical effects than [16] without relying on pre-rendered meshes. To do so, we implemented a on observation map generation procedure sampling all the relevant parameters independently for each sample.

3. Method

This section describes the mathematical formulation of the normal estimation problem and provides a detailed explanation of how pixel-wise training data is generated.

3.1. Normal estimation

Our calibrated PS approach takes as input a set of J varied illumination images. The illumination is assumed to be directional with known light directions $\mathbf{L}_j$ and the brightness ϕ_j. For each pixel p, its value at image j is denoted as $i_{j,p}$. The objective of our method is to recover the normal $\mathbf{N}_p$ at each pixel p. This is achieved by combining all *observations* of the pixel in the images with varied illumination into a single $d \times d \times 4$ map O that in turn is fed into a CNN which regresses normals. A high level diagram of this procedure is shown in Figure 2. A more detailed explanation of all the steps is given below.

Observation map. The concept of observational map has been introduced in [16] as a way to merge information of a variable number of images into a single $d \times d$ image map. The mapping procedure follows two steps: Firstly, normalised observations $\hat{i}_{j,p}$ are computed by compensating for the light sources brightness variation, converting to grayscale (adding r,g,b components) and then dividing with the maximum (of the map):

$$\hat{i}_{j,p} = \frac{i_{j,p,\mathrm{r}}/\phi_{j,\mathrm{r}} + i_{j,p,\mathrm{g}}/\phi_{j,\mathrm{g}} + i_{j,p,\mathrm{b}}/\phi_{j,\mathrm{b}}}{\max_j (i_{j,p,\mathrm{r}}/\phi_{j,\mathrm{r}} + i_{j,p,\mathrm{g}}/\phi_{j,\mathrm{g}} + i_{j,p,\mathrm{b}}/\phi_{j,\mathrm{b}})}. \quad (1)$$

This normalisation operation is designed to compensate for the albedo variation of different pixels, hence reducing the range of the data. Secondly, the normalised observations $\hat{i}_j$ (omitting dependence on p for clarity) are placed on a square grid O_n of size $d \times d$, with the location determined from the light source direction $\mathbf{L}_j = [l_j^x, l_j^y, l_j^z]$ as follows:

$$O_{\mathrm{n}}\left(\left\lfloor d\frac{l_j^x+1}{2}\right\rfloor, \left\lfloor d\frac{l_j^y+1}{2}\right\rfloor\right) = \hat{i}_j. \quad (2)$$

Note that the use of the division operation can corrupt the data in two cases. Firstly, if the maximum value is saturated, the map values are overestimated. Secondly, for very dark points, the ratio operation becomes numerically unstable and any amount of noise (or just discretisation inaccuracy) is greatly amplified. In order to overcome these limitations, we extend the observation map concept to a 3D map O which also includes the RGB channels such as:

$$O_{\mathrm{rgb}}\left(\left\lfloor d\frac{l_j^x+1}{2}\right\rfloor, \left\lfloor d\frac{l_j^y+1}{2}\right\rfloor\right) = \begin{bmatrix} i_{\mathrm{r}}/\phi_{\mathrm{r}} \\ i_{\mathrm{g}}/\phi_{\mathrm{g}} \\ i_{\mathrm{b}}/\phi_{\mathrm{b}} \end{bmatrix}_j, \quad O = [O_{\mathrm{rgb}}\,; O_{\mathrm{n}}] \quad (3)$$

where O is a concatenation on the 3rd axis so defining a $d \times d \times 4$ map. Finally, these observation maps are fed into a CNN which regresses surface normal $\mathbf{N}_p$.

Network training. As in [16], we use a CNN for regressing normals from observational maps. We use a variant of

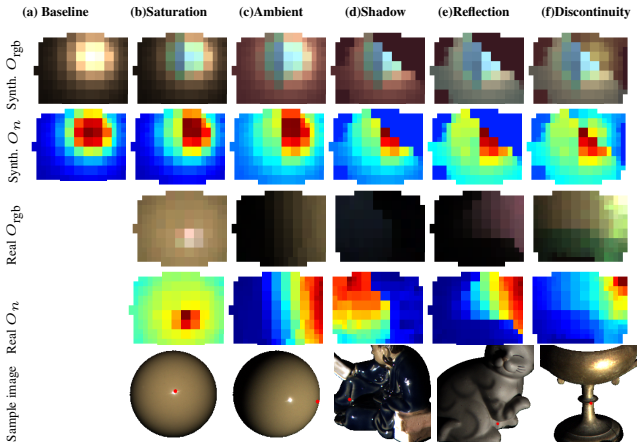


Figure 3. Demonstration of relevant effects that we model during observation map generation. RBG maps and normalised gray (components O_{rgb} and O_{n} in Equation 3) are shown. (a) is the baseline direct reflectance r_d map. (b) shows the change when variable light source brightness are considered (different pixels saturate at different levels and so the normalisation with the brightness distorts the specular highlight). (c) shows the addition of ambient light that acts as an additive offset everywhere. (d) cast shadow blocks all light expect ambient/self reflection in regions of the map. (e) Self reflection can be noticed in real data by the color change: the mostly gray cat contains red pixels at the reflection point. (f) Points at the sharp edge of the cup exhibit discontinuity (which looks like the mixing of two different maps).

DenseNet [14], with 16 colvolutional layers (followed by ReLU activation), 3 maxpooling and 3 dropout ones. The networks has around 4.9 million parameters and the complete diagram is shown in Figure 2. The main difference with [16] is that we use 4 instead of 2 dense blocks as well as more filters per layers and we removed one of the 2 fully connected layers at the top of the network with all these changes aimed at increasing the learning capacity.

3.2. Data Generation

Our CNN is trained using synthetically generated observation maps. Each map is computed independently and no global object rendering is performed, avoiding the need for expensive graphics operations like surface tessellation and rasterisation. For each data point, surface normal $\mathbf{N}$, set of lights $\{\mathbf{L}_j\}$, albedo ρ as well as material M are indepenently sampled (see Section 4 for parameter distributions). More specifically, M is either a 9 dimensional vector containing the parameters of the *Disney* BRDF [4] (excluding anisotropy) or material index from the MERL BRDF reflectance database [27]. Using these material parameters, a set of *direct* reflectance components $\{r_{d,j}\} = \text{B}(\mathbf{N}, \mathbf{L}_j, \mathbf{V}_0, \rho, M)$ can be generated with either computing the Disney [4] non-linear equation or performing table look-up for the appropriate MERL [27] material. Note that we assume orthographic setup hence the viewing direction $\mathbf{V}_0 = [0, 0, 1]$ is used for all lights L_j. However, real image pixel appearances deviate from pure BRDF refletance values. Such deviations include global illumination effects due to interaction of the incoming/reflected light with other parts of the surface as well as local effects such as ambient light and surface discontinuity. Therefore, in order to increase the realism of our synthetic data, a set of effects are approximated by adjusting the reflactances $r_{d,j}$ accordingly. The rest of the section explains these aforementioned effect in the context of observation map generation. Dependence on j is ignored for the rest of the section for brevity.

- **Cast shadows.** Cast shadows are observed in real data when a part of the surface is blocking the light, thus turning the direct reflectance to zero. This is a structured effect with very high correlation for nearby light sources. Our approximation of this effect is performed by sampling a shadow map, i.e. a binary function $S(\mathbf{L}) = 0$, if occluded $S(\mathbf{L}) = 1$, otherwise, containing regions of occluded lights (see supplementary for details of the shadow map sampling). A notable difference of our work to approximating shadows with a structured dropout (e.g. [22]) is that we consider the shadow map as a part of the data generation process. As it described below, $S(\mathbf{L})$ is combined with other effects (self-reflections/ambient) to compute a combined pixel intensity which will be nonzero, even for shaded light sources.

- **Self reflections.** Self reflections occur in specular objects as a result of parts of the surface acting as auxiliary light sources. This effect can become very complicated in reality with potentially hundreds of points contributing additional light components. To estimate a computationally efficient approximation we sample up to 5 points in directions $\mathbf{L}_R$ and compute a single light bounce from $\mathbf{L}$ to $\mathbf{L}_R$ to $\mathbf{V}_0$. We note that in the case of directional lights far away from the surface, self reflection directions are constrained to be part of the shadow map $S(\mathbf{L}_R) = 0, \forall \mathbf{L}_R$. This is true as any ray extending from P outwards, either intersects the surface at another point A or extends to infinity unintersected. In the first case, A is a potential self reflection point and in the second case light can be received from a far away light source (see Figure 4). For each of these self reflection points, surface normals $\mathbf{N}_R$ and albedo ρ_R are independently sampled but the material is assumed to be the same (see supplementary material for further justification). Then, the self reflection component for light $\mathbf{L}$ at reflection points $\{\mathbf{L}_R\}$ is:

$$r_r(\mathbf{L}, \{\mathbf{L}_R\}) = \sum_{\mathbf{L}_R,\ \mathbf{L}_R \neq \mathbf{L}} \text{B}(\mathbf{N}_R, \mathbf{L}, \mathbf{L}_R, \cdots)\text{B}(\mathbf{N}, \mathbf{L}_R, \mathbf{V}_0, \cdots). \tag{4}$$

Note that in the first BRDF term, the effective view vector is now $\mathbf{L}_R$ which is also the effective light vector in the second term (as light travels from $\mathbf{L}$ to $\mathbf{L}_R$ to $\mathbf{V}_0$). We note that this single bounce ray-tracing does not fully compensate for the case of multiple light bounces, subsurface

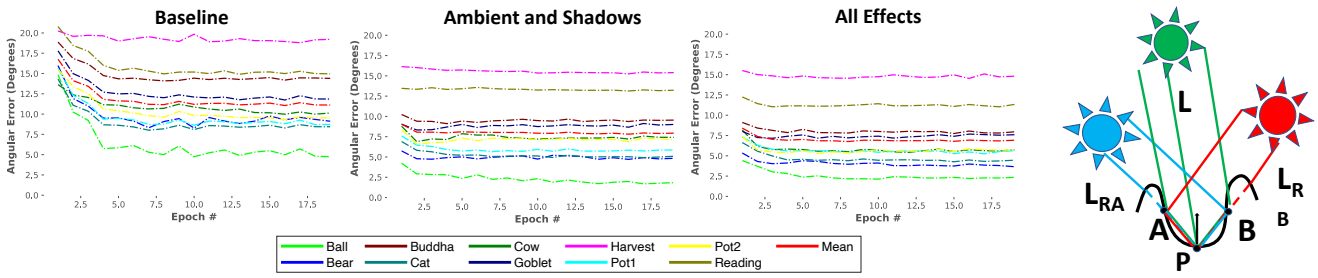


Figure 4. (from the left) MAE evolution (during training) curves illustrating the performance of the networks trained with successive effects on the DiLiGenT objects. The networks compared here are trained on maps generated with: baseline basic reflectance, ambient and global shadows only, all effects. It is observed that successive effects improve performance by sifting downwards the error curves with only notable exception being the Ball as it suffers the least from the global illumination effects. (right) Demonstration of the constraint between shadows and self reflections. In this example, red and blue light sources (assumed far away providing uniform directional illumination) are in shadow when considering the reflection of the point P. Thus, we conclude that there are the points (A,B), located in these directions (the position along the ray does not matter in a far-field setting), which generate self reflection for the rest of the light sources.

scattering or other more complicated global light transform paths. Finally, we note that self reflections alters the color of the pixel (the multiplication of the BRDFs in Equation 4 will increase the color saturation) and thus the inclusion of the RGB channels is further motivated in order to give information to the network to disambiguate the self reflection component from the main one.

- **Surface discontinuity.** It is common to assume that each pixel corresponds to the reflection of a single surface point with a specific normal (e.g. in differential approaches like [28] which assume continous surfaces). However, in practice, pixels have a finite size and thus it is likely that they record the reflectance of multiple surface points with potentially different surface normals. This effect is mostly relevant at surface discontinuity points, such as occlusion boundaries and sharp edges. As the BRDF is a nonlinear function of $\mathbf{N}$, this mixing effect needs to be accounted for as well. Our implementation approximates this effect by sampling $t \in \{1, 2, 3\}$ normals $\mathbf{N}_k$ per pixel (85% of pixels get $t = 1$ to not have this effect) and then average out the respective reflectances (both direct r_d, which may be blocked by shadow, and due to self reflection r_r) to compute an overall reflectance $r_T = \sum_k \frac{r_d(\mathbf{N}_k)S(\mathbf{L})+r_r(\mathbf{N}_k)}{t}$.

- **Ambient light.** Most real images contain some amount of ambient light mostly due to light dispersing into the atmosphere and reflecting on other objects in the environment. Even if the PS images are captured in a dark room with no reflective objects, this effect still persists even though it can be very small ($\approx 0.1\%$ of the maximum intensity). This effect is usually modeled (e.g. [26]) as a constant reflection a. We note that this constant reflection has high correlation with surface albedo and it is also diminished at very oblique angles (along with most of the reflection) so we sample such as $a \propto \rho \mathbf{N} \cdot \mathbf{V}_0$. We finally, include an additional small (up to $1e-4$) additive uniform noise n_{AU} component to account for any additional light arriving to the camera (e.g. reflecting of atmospheric dust).

Figure 5. Demonstration of the impact of incremental modeling of effects in the performance of our PX-CNN-PS on the 'Buddha' of DiLiGenT real dataset. (a) Shows the result with the baseline network. For the rest of the effects, most of the improvement is at: (b) specular highlights middle of pot, (c) top of head, (d) significantly in most concave regions, (e) middle of head (f) sleeves.

- **Camera Noise.** Real cameras are prone to noise which can reasonably approximated as Gaussian. We include two components: a multiplicative n_{MG} and an additive one n_{AG}. In practice, these components are small so we assume standard deviations of $1e-4$ (means are 1 and 0 respectively).

- **Miscellaneous.** Finally, we include a multiplicative uniform noise component n_{MU} aimed to address miscellaneous unmodeled physical effects. These include light source brightness calibration [24] uncertainty and near light attenuation (as in reality point light sources are not infinitely far away) which affect pixel brightness in a multiplicative way. We empirically observed that this was the most important noise component with an optimum value of 5%.

- **Saturation Variation.** Different real light sources have varied brightness ϕ_J. The observation map parameterisation aims to compensate for this variation through dividing by ϕ_j. However, in practice, pixel saturation makes this compensation imperfect and thus needs to be augmented for. The practical implementation involves sampling a brightness value ϕ_j, multiply the reflectance with this value, apply the rest of the augmentation and then apply discretisa-

Effects	Ball	Bear	Buddha	Cat	Cow	Goblet	Harvest	Pot1	Pot2	Reading	AVG
Baseline	5.8	9.8	14.6	8.8	10.1	12.5	18.9	9.4	9.7	15.4	11.48
+Sat	3.8	8.3	14.5	7.9	10.1	12.1	19.4	8.9	9.6	15.1	10.97
+Shadow	1.7	4.8	9.3	4.9	7.1	8.8	15.3	5.8	7.0	13.1	7.78
+Ambient	2.1	4.2	8.4	4.6	5.8	7.8	14.9	5.6	6.0	12.5	7.18
+Reflection	2.4	3.7	8.0	4.5	5.8	7.6	14.5	5.4	5.6	10.9	6.83
+Discontinuity	2.2	3.7	7.8	4.3	5.5	7.6	15.0	5.4	5.7	10.9	6.79
PX-NET O_n	1.9	3.7	7.6	4.5	5.4	7.0	13.3	5.2	5.2	10.5	6.43
PX-NET+O_{rgb}	2.0	3.6	7.6	4.4	4.7	6.9	13.1	5.1	5.1	10.3	6.28

Table 1. Ablation study of the components of the several modelled physical effects in the accuracy of our PX-CNN-PS on the real data of DiLiGenT [35]. It is observed that performance is almost monotonic for all objects. One notable exception is the Ball which has no reflections and ambient so including these effects decrease the performance for this object. Last 2 lines are obtained using the improved architecture PX-NET which further reduces the normal error (both with normalised maps as well as with the inclusion of the RGB channels).

tion and saturation i.e. $D(x) = (\text{uint}_{16}(2^{16}x))/2^{16}$. Thus for saturated pixels, the final division to create an observation map does not fully compensate the light source brightness[1]. Note that as the brightness is different for different channels, this results into specular highlights not being completely white in the brightness compensated images.

Combining all the above effects the overall generated pixel intensity i is calculated as follows:

$$i_j = D\Big((r_{T,j}+a)\phi_j n_{MU,j} n_{MG,j} + n_{AU,j} + n_{AG,j}\Big). \quad (5)$$

Finally, $\{i_j\}$ are converted into an observation map as explained in Section 3.1. Visual illustration of these effects in real image maps and our synthetically generated are shown in Figure 3. Note that the synthetic maps at Figure 3 are generated with Diligent [35] lights to be comparable to real ones - we used random lights during train time. Detailed explanation for all the relevant hyperparameters can be found in the supplementary material.

4. Experimental Setup

This section describes our experimental setup including the datasets used, training and evaluation procedure. **Datasets.** We use three synthetic and one real dataset for evaluation. The real dataset used for the experiments is DiLiGenT [35] consisting of 10 objects of varied materials and geometry. For each object, 96 images (612×512 px) are provided along with ground truth light source directions, brightness and normal maps. For the 'Bear' object, the first 20 images are corrupted and thus are removed as reported by [16]. We perform full lights as well as sparse lights evaluation, the later consisting of 10 random subsets of 10 lights. In addition, we consider two synthetic datasets rendered with Blender (using the Cycles render engine) which

[1] In Diligent, ϕ varies between 0.28 and 3.2 hence saturated values map to $0.31 - 3.57$ which is a fairly big variation.

Method	Sphere 17 (S)	Sphere 17 (M)	Turtle 17 (S)	Turtle 17 (M)	Paperbowl 17 (S)	Paperbowl 17 (M)	Sphere 305 (S)	Sphere 305 (M)	Turtle 305 (S)	Turtle 305 (M)	Paperbowl 305 (S)	Paperbowl 305 (M)	AVG
Baseline [42]	4.6	35.4	12.3	39.7	29.4	39.3	5.0	44.5	12.6	40.2	28.8	37.0	27.4
CNN-PS[16], K=1	4.9	11.9	12.9	20.2	22.2	36.8	1.6	2.0	4.2	6.5	6.8	11.1	11.8
PX-NET, K=1	1.78	4.52	7.45	13.68	22.14	36.01	0.68	6.45	3.46	10.66	7.32	22.19	11.36

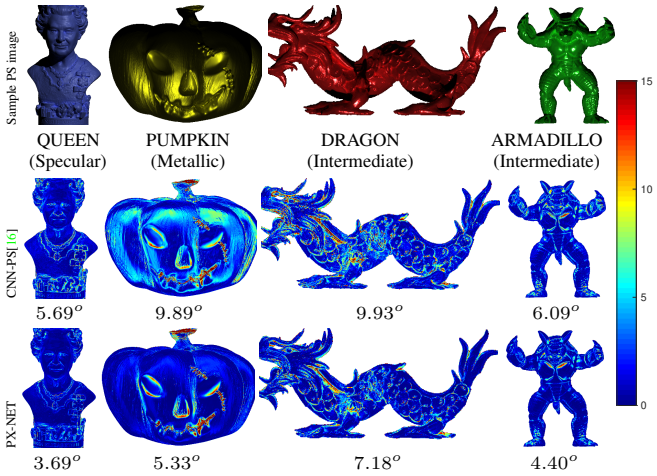


Figure 6. Comparison of our PX-NET with CNN-PS [16] on synthetic, globally rendered objects. The table at the top shows performance of Cycles-PS (from [16]) and on the bottom on our, uniform material objects. The proposed approach outperforms the competition on both datasets, especially on the uniform material objects (average MAE 7.90^o vs 5.15^o).

performs realistic computation of global illumination effects. The first dataset is Cycles-PS-Test [16] containing 3 objects. Each object is rendered in a multi-material setting (each superpixel has a different random material) from either a specular or metallic distribution. As this material distribution is unrealistic, we generated a second dataset where we rendered four single material/albedo objects namely QUEEN, PUMPKIN, ARMADILLO, DRAGON (see Figure 6). These objects are non-convex and were rendered with Blender (16 bit 512x512 px images) including realistic global illumination effects using the 96 light sources from DiLiGenT. Finally, the ability of the network to learn materials is evaluated on a synthetic dataset of spheres using the MERL materials [27]. For all 100 materials, we (pixelwise) rendered 96 spheres using the DiLiGenT lights.

Training details. Baseline experiments are performed using the exact architecture of [16]. This version will be termed PX-CNN-PS. The only difference compared to [16] for this version is the training data that was made using our data generation procedure explained in Section 3.2. The purpose of PX-CNN-PS is to show that our data generation strategy can compensate for the real and global effects and even achieve state of the art results on the dense light setting (PX-CNN-PS with K=10 is only outclassed by our PX-NET, see Table 2). Final experiments are performed with our modified version of the architecture called PX-NET which achieves significantly better results.Note that a single PX-NET was trained for all dense experiments and another one for sparse ones.

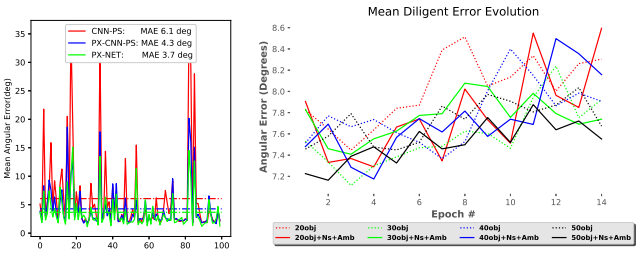


Figure 7. (left) Comparison of CNN-PS [16] with our two networks (PX-CNN-PS ,PX-NET) tested on synthetic MERL images which were rendered with the DiLiGenT lights. All results are shown for a K=1 prediction with corresponding mean errors 6.1, 4.3, 3.7 illustrated as horizontal lines. (right) Test-time accuracy (on DiLiGenT [35] dataset) evolution of CNN-PS [16] network when trained on an increased number of objects (20, 30, 40, 50) using original data generation and training protocol using (dashed lines). Solid lines indicate the performance of the corresponding networks trained with additional ambient light, camera noise and multiplicative noise effects.

Implementation. The network was implemented in Keras of Tensorflow 2.0. The data generation engine was implemented in Python and C++ with the only external dependence being OpenCV for basic vector algebra and i/o. We trained the network using the mean angular error (MAE) loss function, which is also the evaluation metric for all experiments. For the predicted normals $\mathbf{n}_p$ and ground truth ones $\mathbf{n}_t$, MAE is computed as: $|\text{atan2}(||\mathbf{n}_t \times \mathbf{n}_p||, \mathbf{n}_t \cdot \mathbf{n}_p)|$.
Hyper parameters. The training batch size was set at 2400 with 5000 batches per epoch (12M maps). We trained for 20 epochs (which is enough for convergence Figure 4) using the default settings of the Adam [19] optimiser. The train time was around 7 hours for PX-CNN-PS and 15 hours for PX-NET on a NVIDIA GeForce RTX 2080Ti. The light distribution was set to 50-1000 random lights (sampled uniformly with elevation angle ranging from 0^o to 70^o) in order to have a fair comparison with [16]. The sparse light setup uses 10 random lights up to 45^o (to match [22]). The exact hyperparameters of the data generation procedure are described in the supplementary material.
Rotation psedo-invariance: [16] notes that observation maps can be rotated in order to perform a test time augmentation (using 10 rotation which is termed as K=10). If this augmentation is not used (which is the default choice in the paper unless otherwise stated), the single network evaluation is termed K=1.

5. Experiments

In this section we present experiments showing state of the art performance in the datasets described in Section 4.
Ablation of realistic effects modeling. Our first experiment aimed at evaluating the effect of the incremental modeling to demonstrate how the network trained with per-pixel data can outperform the network trained with globally rendered training data. For that, we first trained a series of networks with the exact same architecture of CNN-PS [16], which we refer to as PX-CNN-PS, and observed the effect of incrementally applying the series of different modelled effects (the miscellaneous noise effects are always applied). The effect of the improved architecture as well as the additional RGB channels are also shown in the bottom two rows. The evaluation is performed on the real DiLiGenT dataset. This can be seen in Figure 4 and Figure 5 as well as in Table 1. We observed that the performance improves monotonically for most objects at each step, as well as the average error across the whole dataset.

Method	Ball	Bear	Buddha	Cat	Cow	Goblet	Harvest	Pot1	Pot2	Reading	AVG
Baseline [42]	4.1	8.4	14.9	8.4	25.6	18.5	30.6	8.9	14.7	19.8	15.39
SPLINE-Net[44]	1.7	4.7	9.1	5.5	9.6	9.4	24.4	5.9	7.9	12.8	9.1
ICML [37]	1.5	5.8	10.4	5.4	6.3	11.5	22.6	6.1	7.8	11.0	8.83
Exemplars [15]	1.3	5.6	8.5	4.9	8.2	7.6	15.8	5.2	6.4	12.1	7.55
PS-FCN [6]	2.7	4.8	6.2	7.7	7.2	7.5	7.8	10.9	6.7	12.4	7.4
CNN-PS[16], K=1	2.7	4.5	8.6	5	8.2	7.1	14.2	5.9	6.3	13	7.55
CNN-PS[16], K=10	2.2	4.1	7.9	4.6	8	7.3	14	5.4	6	12.6	7.21
Inv. model [39]**	1.8	4.1	6.1	4.7	6.3	7.2	13.3	6.5	6.5	10.1	6.65
PX-CNN-PS, K=1	2.2	3.7	7.8	4.3	5.5	7.6	15.0	5.4	5.7	10.9	6.79
PX-CNN-PS, K=10	2.0	3.4	7.6	4.2	5.2	6.8	14.2	4.9	5.3	10.4	6.39
PX-NET, K=1	2.0	3.6	7.6	4.4	4.7	6.9	13.1	5.1	5.1	10.3	6.28
PX-NET, K=10	2.0	3.5	7.6	4.3	4.7	6.7	13.3	4.9	5.0	9.8	6.17
10 Lights											
Baseline [42]	4.4	9.1	15.6	9.0	26.4	19.6	31.3	9.5	15.4	20.2	16.04
CNN-PS[16]	9.1	11.7	13.2	14.1	14.7	14.6	15.5	17.0	14.0	19.6	14.34
SPLINE-Net [44]	5.0	6.0	10.1	7.5	8.8	10.4	19.1	8.8	11.8	16.1	10.35
Mimify [22]	4.0	8.7	11.4	6.7	10.2	10.5	17.3	7.3	9.7	14.4	10.02
Inv. model [39]**	2.3	5.2	7.1	5.6	7.5	8.8	15.3	7.1	8.2	10.9	7.79
PX-NET, K=1	2.8 ±0.4	5.2 ±0.4	9.6 ±0.3	6.6 ±0.2	7.8 ±0.5	10.3 ±0.9	16.5 ±0.6	7.4 ±0.2	8.1 ±0.7	13.5 ±0.4	8.76 ±0.3
PX-NET, K=10	2.5 ±0.4	4.9 ±0.3	9.4 ±0.3	6.3 ±0.2	7.2 ±0.4	9.7 ±0.8	16.1 ±0.5	7.0 ±0.2	7.7 ±0.6	13.1 ±0.5	8.37 ±0.3

Table 2. Quantitative comparison of the proposed method (both simplified PX-CNN-PS and full PX-NET) on the DiLiGenT benchmark [35]. For our networks as well as for [16] results using K=10 are also presented for completeness. The bottom portion of the table presents evaluation with 10 random lights (**[39] uses 9x9 pixels patches and a specific illumination constraint so the comparison is not fully fair). We perform the experiment 10 times and report mean and standard deviation of the each error over the 10 tries (also computing K=10)

It is noted that after the inclusion of the ambient, shadow and saturation variation steps, PX-CNN-PS outperform CNN-PS (6.79^o vs 7.21^o, see Table 2) which is trained with globally rendered data. This can be explained by the following three reasons. Firstly, the synthetic training data of CNN-PS do not include some of the effect we are modeling here namely light source brightness variation, noise and ambient light. Secondly, CNN-PS was trained on a limited subset of Disney material parameters (due to being constrained by slow, global object rendering). Finally, it is likely, that CNN-PS did overfit on the specific distribution of global effect of its training data rendered using only 15 meshes.

The superiority of our training data compared the ones of CNN-PS is also confirmed on the synthetic, globally rendered, objects of Figure 6 as well as the Cycles-PS dataset. PX-NET outperforms CNN-PS in of these 4 ob-

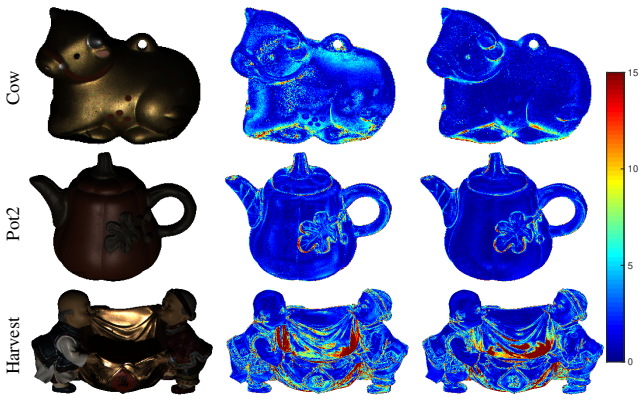


Figure 8. Some error maps from Table 2 (for k=1) comparing our PX-NET to CNN-PS [16]. It is noted that we significantly outperform the competition in convex region due to using more broad set of materials at train time (this is evident on the COW which is coated with metallic paint that is quite different that the metallic materials considered by CNN-PS). The Pot2 error map demonstrates the strength of our discontinuity augmentation on the leaf boundary. Finally, the Harvest error map shows that we can outperform in some concave regions (bellow left head); however regions with complicated self-reflection patterns (middle of the image) are a potential limitation to our approach (due to the single bounce self reflection assumed.

jects. The most significant improvement is on the PUMPKIN object (5.33^o vs 9.89^o) and it can be explained by the fact that CNN-PS does not include the discontinuity modelling which is important as the surface of this object is rough and detailed. The comparison its much closer on Cycles-PS (11.36^o vs 11.8^o) and this is expected as it is made in a similar way to the training data of Cycles-PS. The reason for the superiority of PX-NET is probably the higher learning capacity and the inclusion of the RGB channels.

Evaluation on MERL spheres. The next synthetic experiment (Figure 7) compares both of our networks PX-CNN-PS (with all effects) and PX-NET with CNN-PS on the synthetic images rendered with MERL materials [27]. The aim of this evaluation is to demonstrate that our networks can deal with various real world reflectances. It is noted that PX-CNN-PS outperforms CNN-PS (6.1^o vs 4.3^o MAE) and this is expected as we include MERL materials in the training data. PX-NET further reduces the MAE to 4.8^o, but there are still materials which are not very well recovered and thus this motivates for future research.

Increasing size of globally rendered objects for CNN-PS. In order to justify the use of our per-pixel training procedure, we examine the effect of increased amount of objects in the Cycles-PS [16] dataset to CNN-PS [16] network. We supplement original 15 objects of CNN-PS [16] with objects from [35] Thingi10K [45] dataset (see supplementary material for full list and examples of the objects). Test-time accuracy evolution of CNN-PS [16] network when trained in total on 20, 30, 40 and 50 objects is shown in Figure 7 (right), using dashed lines. Note that all networks seem to achieve their best performance in early epochs and do not benefit from longer training. Also, adding more than 30 objects does not seem to benefit the performance on the DiLiGenT [35] dataset. This is likely due to overfitting to both global shapes of the objects provided in the dataset and particular choice of mixed-material rendering by [16]. If some modelling of realistic effects (in this case, applied on globally rendered images) mentioned in Section 3 are applied such as ambient light, camera noise and multiplicative noise, the performance in initial epochs (solid lines) slightly improves to corresponding networks trained without data augmentation, however still the best performance of 7.11° error is achieved by a network trained on 30 objects with no augmentations. In contrast PX-CNN-PS is able to achieve accuracy of 6.79° by avoiding computationally inefficient global object rendering.

Comparison with the state-of-the-art. Finally, we compare our two networks with other state-of-the-art methods in the DiLiGenT [35] dataset in Table 2, in both dense and sparse light settings. For completeness, we also include the results after applying the test time rotation pseudo-invariance augmentation (K=10). Three sample error maps are shown in Figure 8 (for the K=1 network evaluation). Both of our networks significantly outperform the competition in the average error as well in almost all objects individually. The success of our method can be attributed on the ability of the network to deal with real world materials with complex reflectance (we exhibit very minimal error in convex regions where the PS problem reduced to BRDF inversion) as well as simultaneously being very robust to global illumination effects due to our modeling strategy. We note that the best performing method on the sparse setting [39] uses 9x9 pixel patches as well as a constraint on the light setup at both train and test time (instead of 10 fully random lights) so the comparison is not fully fair (we still outperform it in the dense setting non the less).

6. Conclusion

In this work we presented a novel, simple and efficient concept for generating in-line training data for solving the PS problem, using a simple pixel observation map generation procedure. We approximate global effects like shadows, self-reflections, etc. by adopting a modeling strategy based on real and synthetic data observations. We analysed the performance of our approach while progressively adjusting the training data and we quantitatively showed the actual benefits in adopting such a modeling strategy. State-of-the-art results are achieved on the real DiLiGenT [35] benchmark as well as the synthetic Cycles-PS [16] one.

Future work includes considering extension to mutliview PS setting, e.g. using SDF representation [25].

References

[1] J. Ackermann and M. Goesele. A survey of photometric stereo techniques. *Foundations and Trends in Computer Graphics and Vision*, 2015. 2

[2] Blender-Online-Community. *Blender - A 3D modelling and rendering package*. Blender Foundation, 2018. 2

[3] J. F. Blinn. Models of light reflection for computer synthesized pictures. In *SIGGRAPH*, 1977. 1

[4] B. Burley. Physically-based shading at disney. In *SIGGRAPH Course Notes*, 2012. 1, 4

[5] G. Chen, K. Han, B. Shi, Y. Matsushita, and K.-Y. K. Wong. Self-calibrating deep photometric stereo networks. In *CVPR*, pages 8739–8747, 2019. 2

[6] G Chen, K. Han, B. Shi, Y. Matsushita, and K.-Y. K. Wong. Deep photometric stereo for non-Lambertian surfaces. *PAMI*, 2020. 2, 7

[7] G. Chen, K. Han, and K.-Y. K. Wong. PS-FCN: A flexible learning framework for photometric stereo. In *ECCV*, 2018. 2

[8] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics*, 1982. 1

[9] K. Enomoto, M. Waechter, K. N. Kutulakos, and Y. Matsushita. Photometric stereo via discrete hypothesis-and-test search. In *CVPR*, pages 2311–2319, 2020. 2

[10] E. Hinton G. Deep belief networks. *Scholarpedia*, 2009. 2

[11] V. Havran, J. Filip, and K. Myszkowski. Perceptually motivated BRDF comparison using single image. *Comput. Graph. Forum*, 2016. 1

[12] S. Hill, S. McAuley, B. Burley, D. Chan, L. Fascione, M. Iwanicki, N. Hoffman, W. Jakob, D. Neubelt, A. Pesce, and M. Pettineo. Physically based shading in theory and practice. In *SIGGRAPH*, 2015. 1

[13] B. K. P. Horn. Obtaining shape from shading information. *The Psychology of Computer Vision, Winston, P. H. (Ed.)*, 1975. 1

[14] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017. 4

[15] Z. Hui and A. C Sankaranarayanan. Shape and spatially-varying reflectance estimation from virtual exemplars. *PAMI*, 2016. 7

[16] S. Ikehata. CNN-PS: CNN-based photometric stereo for general non-convex surfaces. In *ECCV*, 2018. 1, 2, 3, 4, 6, 7, 8

[17] S. Ikehata and K. Aizawa. Photometric stereo using constrained bivariate regression for general isotropic surfaces. In *CVPR*, 2014. 1

[18] Y. Ju, L. Qi, H. Zhou, J. Dong, and L. Lu. Demultiplexing colored images for multispectral photometric stereo via deep neural networks. *IEEE Access*, 2018. 2

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015. 7

[20] E. P. F. Lafortune, S.-C. Foo, K. E. Torrance, and D. P. Greenberg. Non-linear approximation of reflectance functions. In *SIGGRAPH*, 1997. 1

[21] J. H. Lambert. *Photometrie*. 1760. 1

[22] J. Li, A. Robles-Kelly, S. You, and Y. Matsushita. Learning to minify photometric stereo. In *CVPR*, pages 7568–7576, 2019. 4, 7

[23] F. Logothetis, I. Budvytis, R. Mecca, and R. Cipolla. A CNN-Based Approach for the Near-Field Photometric Stereo Problem. In *BMVC*, 2020. 2

[24] Fotios Logothetis, Roberto Mecca, and Roberto Cipolla. Semi-calibrated near field photometric stereo. In *CVPR*, 2017. 5

[25] Fotios Logothetis, Roberto Mecca, and Roberto Cipolla. A differential volumetric approach to multi-view photometric stereo. In *ICCV*, 2019. 8

[26] F. Logothetis, R. Mecca, Y. Quéau, and R. Cipolla. Near-field photometric stereo in ambient light. In *BMVC*, 2016. 5

[27] W. Matusik, H. Pfister, M. Brand, and L. McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 2003. 4, 6, 8

[28] R. Mecca, Y. Quéau, F. Logothetis, and R. Cipolla. A single lobe photometric stereo approach for heterogeneous material. *SIAM Journal on Imaging Sciences*, 9(4):1858–1888, 2016. 5

[29] A. Ngan, F. Durand, and W. Matusik. Experimental validation of analytical BRDF models. In *SIGGRAPH*, 2004. 1

[30] A. Ngan, F. Durand, and W. Matusik. Experimental analysis of BRDF models. In *EUROGRAPHICS*, 2005. 1

[31] B. T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975. 1

[32] Y. Quéau, R. Mecca, and J.-D. Durou. Unbiased photometric stereo for colored surfaces: A variational approach. In *CVPR*, 2016. 1

[33] H. Santo, M. Samejima, Y. Sugano, B. Shi, and Y. Matsushita. Deep photometric stereo network. In *ICCV Workshops*, 2017. 2

[34] H. Santo, M. Waechter, and Y. Matsushita. Deep near-light photometric stereo for spatially varying reflectances. In *ECCV*, 2020. 2

[35] B. Shi, Z. Wu, Z. Mo, D. Duan, S.-K. Yeung, and P. Tan. A benchmark dataset and evaluation for non-lambertian and uncalibrated photometric stereo. In *CVPR*, 2016. 1, 2, 6, 7, 8

[36] Y. Tang, R. Salakhutdinov, and G. E. Hinton. Deep lambertian networks. In *ICML*, 2012. 2

[37] T. Taniai and T. Maehara. Neural inverse rendering for general reflectance photometric stereo. In *ICML*, 2018. 2, 7

[38] K. E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces. *Journal of the Optical Society of America*, 1967. 1

[39] X. Wang, Z. Jian, and M. Ren. Non-lambertian photometric stereo network based on inverse reflectance model with collocated light. *IEEE Transactions on Image Processing: a Publication of the IEEE Signal Processing Society*, 2020. 2, 7, 8

[40] G. J. Ward. Measuring and modeling anisotropic reflection. *SIGGRAPH*, 1992. 1

[41] R. M. J. Watson and P. N. Raven. Comparison of measured BRDF data with parameterized reflectance models. International Society for Optics and Photonics, SPIE, 2001. 1

[42] R. J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 1980. 1, 6, 7

[43] Y. Yu and W. A. P. Smith. Pvnn: A neural network library for photometric vision. In *ICCV Workshop*, 2017. 2

[44] Q. Zheng, Y. Jia, B. Shi, X. Jiang, L.-Y. Duan, and A. C Kot. SPLINE-Net: Sparse photometric stereo through lighting interpolation and normal estimation networks. In *ICCV*, 2019. 2, 7

[45] Q. Zhou and A. Jacobson. Thingi10K: A Dataset of 10,000 3D-Printing Models. *arXiv*, 1605.04797, 2016. 8