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

Fotios Logothetis\textsuperscript{1} Ignas Budvytis\textsuperscript{2} Roberto Mecca\textsuperscript{1} Roberto Cipolla\textsuperscript{1,2}

\textsuperscript{1}Cambridge Research Laboratory, Toshiba Europe Ltd. Cambridge, UK \texttt{flogothetis, rmecca@crl.toshiba.co.uk}
\textsuperscript{2}University of Cambridge Cambridge, UK \texttt{ib255, rc10001@cam.ac.uk}

\section*{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.

\section{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 through 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.

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 an 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. \( \otimes \) denotes concatenation along the channel axis.

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. Method

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 \( \phi_j \). 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):

\[
\hat{i}_{j,p} = \frac{i_{j,p,r}/\phi_j + i_{j,p,g}/\phi_j + i_{j,p,b}/\phi_j}{\max_j(i_{j,p,r}/\phi_j + i_{j,p,g}/\phi_j + i_{j,p,b}/\phi_j)}.
\]

(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 \( O_{\mathbf{n}} \) of size \( d \times d \), with the location determined from the light source direction \( \mathbf{L}_j = [l_{j,r}, l_{j,g}, l_{j,b}] \) as follows:

\[
O_{\mathbf{n}}\left(\left[d\frac{r_j + 1}{2}\right], \left[d\frac{g_j + 1}{2}\right]\right) = \hat{i}_j.
\]

(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:

\[
O_{\mathbf{rgb}}\left(\left[d\frac{r_j + 1}{2}\right], \left[d\frac{g_j + 1}{2}\right]\right) = \begin{bmatrix} i_{r,j}/\phi_j \\ i_{g,j}/\phi_j \\ i_{b,j}/\phi_j \end{bmatrix}, \quad O = [O_{\mathbf{rgb}} ; O_{\mathbf{n}}]
\]

(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
look-up for the appropriate MERL [27] material. Note that the Disney [4] non-linear equation or performing table parameters, a set of \( \{ r_{d,j} \} \) 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 \( V_0 = [0, 0, 1] \) is used for all lights \( L_j \). However, real image pixel appearances deviate from pure BRDF reflectance 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 reflectances \( r_{d,j} \) accordingly. The rest of the section explains these aforementioned effects 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 \( S(L) = 0 \), if occluded \( S(L) = 1 \), 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, \( S(L) \) 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 \( L_R \) and compute a single light bounce from \( L \) to \( L_R \) to \( V_0 \). 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 \( S(L_R) = 0 \), \( \forall L_R \). 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 \( N_R \) and albedo \( \rho_R \) 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 \( L \) at reflection points \( \{ L_R \} \) is:

\[
r_r(L, \{ L_R \}) = \sum_{L_R \neq L} B(N_R, L, L_R) \lambda B(N, L_R, V_0, \cdots).
\]

Note that in the first BRDF term, the effective view vector is now \( L_R \) which is also the effective light vector in the second term (as light travels from \( L \) to \( L_R \) to \( V_0 \)). We note that this single bounce ray-tracing does not fully compensate for the case of multiple light bounces, subsurface

DenseNet [14], with 16 convolutional 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 \( N \), set of lights \( \{ L_j \} \), albedo \( \rho \) as well as material \( M \) are independently 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 \( \{ r_{d,j} \} \) 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

![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).](image)
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 continuous 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 N, this mixing effect needs to be accounted for as well. Our implementation approximates this effect by sampling $t \in \{1, 2, 3\}$ normals $N_k$ 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 $r_T = \sum_k r_d(N_k)S(L) + r_r(N_k)$.

- **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 $a \propto \rho N \cdot V_0$. We finally, include an additional small (up to $1e-4$) additive uniform noise $n_{AG}$ component to account for any additional light arriving to the camera (e.g. reflecting of atmospheric dust).

- **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 $\phi_J$. The observation map parameterisation aims to compensate for this variation through dividing by $\phi_J$. However, in practice, pixel saturation makes this compensation imperfect and thus needs to be augmented for.
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 latter consisting of 10 random subsets of 10 lights. In addition, we consider two synthetic datasets rendered with Blender (using the Cycles render engine) which per- forms 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 performs significantly better results.

---

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).

<table>
<thead>
<tr>
<th>Effects</th>
<th>Ball</th>
<th>Bear</th>
<th>Buddha</th>
<th>Cat</th>
<th>Cow</th>
<th>Goblet</th>
<th>Harvest</th>
<th>Pot1</th>
<th>Pot2</th>
<th>Reading</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>3.8</td>
<td>9.8</td>
<td>14.6</td>
<td>8.8</td>
<td>10.1</td>
<td>12.5</td>
<td>15.9</td>
<td>9.9</td>
<td>9.7</td>
<td>15.4</td>
<td>14.4</td>
</tr>
<tr>
<td>Sat</td>
<td>3.8</td>
<td>8.3</td>
<td>14.5</td>
<td>7.9</td>
<td>9.1</td>
<td>12.1</td>
<td>19.4</td>
<td>8.9</td>
<td>9.6</td>
<td>15.1</td>
<td>10.97</td>
</tr>
<tr>
<td>Shadow</td>
<td>1.7</td>
<td>4.8</td>
<td>9.3</td>
<td>4.9</td>
<td>7.1</td>
<td>8.8</td>
<td>15.3</td>
<td>5.8</td>
<td>7.0</td>
<td>13.1</td>
<td>7.78</td>
</tr>
<tr>
<td>+Reflection</td>
<td>2.1</td>
<td>4.2</td>
<td>8.4</td>
<td>4.6</td>
<td>5.8</td>
<td>7.8</td>
<td>14.9</td>
<td>5.6</td>
<td>6.0</td>
<td>12.5</td>
<td>7.18</td>
</tr>
<tr>
<td>+Ambient</td>
<td>2.4</td>
<td>3.7</td>
<td>8.9</td>
<td>4.5</td>
<td>5.8</td>
<td>7.6</td>
<td>14.5</td>
<td>5.4</td>
<td>5.6</td>
<td>10.9</td>
<td>6.83</td>
</tr>
<tr>
<td>+Discontinuity</td>
<td>2.2</td>
<td>3.7</td>
<td>7.8</td>
<td>4.3</td>
<td>5.5</td>
<td>7.6</td>
<td>15.0</td>
<td>5.4</td>
<td>5.7</td>
<td>10.9</td>
<td>6.79</td>
</tr>
<tr>
<td>PX-NET</td>
<td>2.9</td>
<td>3.3</td>
<td>7.6</td>
<td>4.3</td>
<td>5.4</td>
<td>7.8</td>
<td>13.3</td>
<td>5.2</td>
<td>5.4</td>
<td>10.3</td>
<td>6.43</td>
</tr>
<tr>
<td>PX-NET+O</td>
<td>2.0</td>
<td>3.6</td>
<td>7.6</td>
<td>4.4</td>
<td>4.7</td>
<td>6.9</td>
<td>13.1</td>
<td>5.1</td>
<td>5.1</td>
<td>10.3</td>
<td>6.28</td>
</tr>
</tbody>
</table>

---

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° vs 5.15°).

---

1In Diligent, \( \phi \) varies between 0.28 and 3.2 hence saturated values map to 0.31 – 3.57 which is a fairly big variation.

---

Table 2. Comparative study of the performance of different methods on two synthetic datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Sphere (S) (M)</th>
<th>Cube (S) (M)</th>
<th>Cylinder (S) (M)</th>
<th>Sphere (S)</th>
<th>Cube (S)</th>
<th>Cylinder (S)</th>
<th>Sphere (S)</th>
<th>Cube (S)</th>
<th>Cylinder (S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycles-PS</td>
<td>4.9</td>
<td>13.9</td>
<td>14.9</td>
<td>20.2</td>
<td>20.6</td>
<td>16.1</td>
<td>24.2</td>
<td>9.2</td>
<td>6.3</td>
</tr>
<tr>
<td>MERL Kit</td>
<td>1.7</td>
<td>7.9</td>
<td>7.6</td>
<td>15.8</td>
<td>22.4</td>
<td>8.5</td>
<td>5.6</td>
<td>3.4</td>
<td>5.6</td>
</tr>
<tr>
<td>Baseline</td>
<td>6.43</td>
<td>11.9</td>
<td>13.4</td>
<td>27.4</td>
<td>27.4</td>
<td>20.4</td>
<td>36.8</td>
<td>5.4</td>
<td>5.4</td>
</tr>
<tr>
<td>PX-NET</td>
<td>6.28</td>
<td>11.9</td>
<td>13.4</td>
<td>27.4</td>
<td>27.4</td>
<td>20.4</td>
<td>36.8</td>
<td>5.4</td>
<td>5.4</td>
</tr>
<tr>
<td>PX-NET+O</td>
<td>6.9</td>
<td>11.9</td>
<td>13.4</td>
<td>27.4</td>
<td>27.4</td>
<td>20.4</td>
<td>36.8</td>
<td>5.4</td>
<td>5.4</td>
</tr>
</tbody>
</table>

---

The first object 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).

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

\[
i_j = D(\frac{r_{T,j} + a}{\phi_j n_{MU,j} n_{MG,j} + n_{AU,j} + n_{AG,j}}).
\]  

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 not fully white in the brightness compensated images.

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

\[
i_j = D\left(\frac{r_{T,j} + a}{\phi_j n_{MU,j} n_{MG,j} + n_{AU,j} + n_{AG,j}}\right).
\]  

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 not fully white in the brightness compensated images.
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: 
\[
\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} |\text{atan2}(\text{atan2}(\langle \mathbf{n}_t \times \mathbf{n}_p \rangle, \langle \mathbf{n}_t \cdot \mathbf{n}_p \rangle))|
\]

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° to 70°) in order to have a fair comparison with [16]. The sparse light setup uses 10 random lights up to 45° (to match [22]). The exact hyperparameters of the data generation procedure are described in the supplementary material.

Rotation pseudo-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.

It is noted that after the inclusion of the ambient, shadow and saturation variation steps, PX-CNN-PS outperform CNN-PS (6.79° vs 7.21°, 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 outperform 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-
and this is expected as we include MERL materials in the training data. PX-NET further reduces the MAE to 0.40 vs 0.88, and this is expected as we include MERL materials in the training data.

The most significant improvement is on the PUMP-KIN object (5.33° vs 9.89°) and it can be explained by the fact that CNN-PS does not include the discontinuity modeling which is important as the surface of this object is rough and detailed. The comparison is much closer on Cycles-PS (11.36° vs 11.8°) 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° vs 4.3° MAE) and this is expected as we include MERL materials in the training data. PX-NET further reduces the MAE to 4.8°, 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 multi-view PS setting, e.g. using SDF representation [25].
References


12765

