Fast Light-Weight Near-Field Photometric Stereo

Daniel Lichy[1] Soumyadip Sengupta[2] David W. Jacobs[1]
[1]University of Maryland, College Park [2]University of Washington
dlichy@umd.edu, soumya91@cs.washington.edu, djacobs@cs.umd.edu

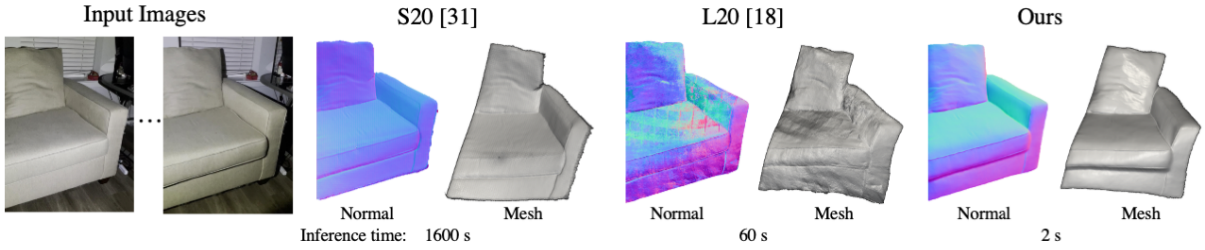

Figure 1. We present a fast light-weight approach for solving near-field Photometric Stereo (PS), which is particularly useful for capturing large objects, e.g. a sofa, in a small confined space. We capture an object with an iPhone camera and a handheld flashlight. Our method is significantly faster during inference and produces more accurate reconstructions than existing methods S20 [31] and L20 [18].

Abstract

We introduce the first end-to-end learning-based solution to near-field Photometric Stereo (PS), where the light sources are close to the object of interest. This setup is especially useful for reconstructing large immobile objects. Our method is fast, producing a mesh from 52 512×384 resolution images in about 1 second on a commodity GPU, thus potentially unlocking several AR/VR applications. Existing approaches rely on optimization coupled with a far-field PS network operating on pixels or small patches. Using optimization makes these approaches slow and memory intensive (requiring 17GB GPU and 27GB of CPU memory) while using only pixels or patches makes them highly susceptible to noise and calibration errors. To address these issues, we develop a recursive multi-resolution scheme to estimate surface normal and depth maps of the whole image at each step. The predicted depth map at each scale is then used to estimate 'per-pixel lighting' for the next scale. This design makes our approach almost 45× faster and 2° more accurate (11.3° vs. 13.3° Mean Angular Error) than the state-of-the-art near-field PS reconstruction technique, which uses iterative optimization.

1. Introduction

In this work, we introduce a fast light-weight Photometric Stereo (PS) technique for near-field illumination. Photometric Stereo aims to reconstruct object geometry from a sequence of images captured with a static camera and varying light sources. Existing near-field PS approaches are slow and extremely memory intensive. Being fast and light-weight enables users to capture images and process them on their laptop within a few seconds, allowing multiple retakes if needed. This light-weight reconstruction technique can be extremely useful for several AR/VR applications. While our method is primarily developed for calibrated lighting, in line with existing far-field approaches, we also show how our method can be extended to uncalibrated real-world captures by introducing a calibration network.

Near-field PS is often preferred over far-field or distant lighting-based PS for both practical and theoretical reasons. It is extremely useful for capturing large objects, e.g. furniture or humans, especially in a confined space like a room [3, 16, 25]. This is because far-field PS approaches assume the lighting to be distant, e.g. 10× the object dimensions is suggested by [33, 34], causing it to be unsuitable for 3D imaging in many indoor spaces. Additionally, low-intensity LED lights on handheld devices (e.g. flashlight on a phone) may not be bright enough to illuminate an object from a large distance [28]. Theoretically, in the case of uncalibrated lighting, near-field PS has no linear ambiguity in contrast to far-field PS where there is the well-known Generalized Bas-Relief ambiguity [4], as shown in [25].

We make our method fast and accurate by forgoing traditional optimization in favor of a recursive multi-scale algorithm. Our proposed method consists of two recursive networks one for predicting surface normal and another for depth maps. At each step of the recursion, we increase the input image resolution by a factor of 2. We first analytically estimate the relative lighting direction and attenuation factor for each pixel in the image (termed 'per-pixel lighting' for clarity) by upsampling the predicted depth map from the previous step. We then infer the surface normal for this scale given the input image, 'per-pixel lighting,' and esti-

mated normal map from the previous scale. Finally, the depth map is predicted conditioned on the estimated normal map and the depth map from the previous scale. The number of steps for this recursion is dictated by the input image resolution making the inference extremely fast, requiring only a few forward passes. We also improve inference speed by using a recursive deep network for estimating depth map from normals instead of solving normal integration by e.g. solving the Poisson equation [13, 30], making it more robust to outliers during training. The recursion allows the use of one network for all scales, thus heavily reducing the memory footprint. This approach is also more robust to noise and lighting calibration errors than existing per-pixel based methods [18, 31] as the recursion leads to a larger receptive field for the network.

Our method is built on the shoulders of existing near-field and far-field PS techniques by adapting the ideas that can best improve performance, inference speed, and memory requirements. Our recursive approach is inspired by [17], which uses a single network for predicting normal at each scale conditioned on the image and the estimated normal from the previous scale. It is non-trivial to adapt the recursion idea proposed in [17] from far-field distant lighting to near-field because per-pixel lighting directions are not known a priori. Our ablation study shows that a trivial extension of [17] to near-field PS that does not refine lighting directions based on depth performs significantly worse (by 3.5°) than our proposed approach. The idea of using depth map to predict 'per-pixel lighting' is inspired by [18,25,31]. However, these approaches operate on pixels or patches using iterative optimization, causing extensive memory usage, slow inference speed and making them highly susceptible to noise and lighting calibration errors.

We first evaluate our method quantitatively on the LUCES dataset [21] with calibrated lighting and show that our method is 2° more accurate in surface normal prediction (11.3° vs. 13.3° Mean Angular Error) than state-of-the-art near-field PS approach L20 [18], and another prior approach S20 [31]. In terms of computational efficiency, our method requires 4GB CPU memory and 12GB GPU memory compared to 27GB CPU and 17GB GPU of L20 [18] for 1024×786 resolution, while S20 [31] fails to scale up to this resolution. Our inference speed is 1.3 secs compared to 59.5 secs of L20 [18] and 2435 secs of S20 [31] for 52 512×384 resolution images; tested on the same hardware.

For many practical applications, such as quickly reconstructing 3D models at the home, calibrated lighting is impractical. In the absence of calibrated lighting, we also introduce an additional lighting calibration network. We first show that on the LUCES dataset with uncalibrated lighting our method is more robust than existing approaches, producing 14.11° Mean Angular Error (MAE) vs 18.85° of L20 and 16.03° of S20. Finally, we capture a few real-world objects with near-field lighting with a commodity flashlight and show that our reconstructed mesh is qualitatively more accurate than existing approaches S20 [31] and L20 [18], after using the same calibration network, see Fig. 1 and 4.

In summary our contributions are as follows:

- A state-of-the-art, fast, light-weight, near-field PS method with 45× faster inference speed and significantly lower memory requirements than existing methods.
- We build on [17], developed for far-field PS, by incorporating 'per-pixel lighting', adding recursive depth prediction from normal, and allowing the flexibility to use unstructured lighting.
- We also introduce a calibration network to facilitate uncalibrated capture in-the-wild with an iPhone camera and a handheld flashlight.

2. Prior Work

Research on Photometric Stereo (PS), introduced in [33], can be divided along a number of dimensions: diffuse vs. specular materials, calibrated vs. uncalibrated lighting, distant vs. nearby lights. In this work, we focus on near-field PS with both known and unknown lighting conditions.

Far-Field Photometric Stereo. We briefly mention some recent far-field PS works that are particularly relevant to this work. For a more comprehensive survey see [2, 9]. Our work is inspired by [17] which introduces a recursive neural net to predict surface normal at each scale given the input image at that scale and the predicted normal map from the previous scale. The authors showed that using a recursive architecture significantly improves performance by capturing global context that is often absent in per-pixel techniques [14] and patch-based techniques [6].

Near-Field Photometric Stereo. Solutions to near-field PS can be roughly divided into two broad approaches.

The first approach relies on a three step iterative refinement [3, 5, 8, 18, 24, 25, 28], starting with an initial shape, e.g. a plane, until convergence: (1) based on the current shape calculate the light directions and intensity at each point; (2) using these light estimates, predict surface normals; (3) integrate normals to update the shape. Logothetis *et al.* [18] uses a per-pixel far-field deep neural network in step (2) while the rest of these methods are purely optimization driven. In contrast, we use two deep recursive neural nets for steps (2) and (3), trained on the whole image for near-field lighting.

Direct optimization approaches rely on inverting the image formation process, often by solving a system of PDEs [22, 23, 27–29]. For a detailed discussion of these methods see [29]. In [34] the authors use a local-global mesh deformation scheme to optimize a mesh that reconstructs the images. Santo *et al.* [31] also optimizes a reconstruction loss. However, as part of their forward pass they decompose observations into reflectance and normal using a far-field deep neural network.

R	number of resolutions
$r_0, ..., r_{R-1}$	sequence of resolutions $r_0 = 64$, $r_{i+1} = 2r_i$, r_{R-1} input image resolution
I_i^j	jth image at resolution r_i
N_i	normal at resolution r_i
D_i	depth at resolution r_i
A_i^j, L_i^j	per-pixel light attenuation and direction at resolution r_i for image j
p^j, d^j, μ^j	light parameters of jth image
$Up(I)$	upsample I by a factor of 2
ones$(r \times r)$	$r \times r$ array of ones

Table 1. Summary of major notations used throughout the text.

Light Calibration. Research on uncalibrated PS either separately estimates lighting or alternately solves for light and shape simultaneously using a variational approach [10]. For the former, the lighting estimation can be physically performed by inserting additional objects [11, 16] in the scene or by using a deep network for prediction [6, 7, 15]. While the these methods have been introduced for far-field PS, we propose a calibration network for near-field PS.

Normal Integration. Normal integration techniques estimate a depth map that is consistent with a normal map. For a detailed discussion see [30]. Ho *et al.* [12] uses the similarity between normal integration and shape from shading (SfS) to develop a normal integration technique. Similarly, we also introduce a deep network for faster and stable normal integration during training based on SfS.

3. Background

In this section, we describe our image formation model for near-field Photometric Stereo (PS). Given M images of an object $(I^1, ..., I^M)$ captured under different known anisotropic point light sources from a fixed viewpoint, we estimate the surface normal and the depth map. Additionally, we assume the camera has known intrinsic parameters, and the mean distance to the object is known (WLOG assume mean distance is 1. See supplement for details). This is the same setup as [18, 31]. In Sec. 4.4, we show how to remove the restriction on known lights and mean distance.

Camera Model We use the standard pinhole camera model centered at the origin in world coordinates and looking down the z-axis. The camera is specified by a 3×3 intrinics matrix K. Any world point $X = (x, y, z)$, projects onto a pixel (u, v) by the formula:

$$(u, v, 1)^T \sim K(x, y, z)^T. \tag{1}$$

Geometry Model We only consider reconstructing the visible region of an object. Therefore the object is completely described by a normal and depth map. Concretely, $X(u, v) \in \mathbb{R}^3$ describes a point on the object appearing in pixel (u, v). Then we can define the depth map by $D(u, v) = X(u, v)_3$, where the subscript 3 refers to the 3rd i.e. z component of $X(u, v)$. We can also recover $X(u, v)$ from the depth map $D(u, v)$ following eqn. 2:

$$X(u, v) = D(u, v)K^{-1}(u, v, 1)^T \tag{2}$$

If $n(X)$ is the normal at the point X then the normal map is defined by $N(u, v) = n(X(u, v))$. Since $X(u, v)$ is a parametrization, we can also calculate the normal map as:

$$N = \frac{(\frac{\partial X}{\partial u} \times \frac{\partial X}{\partial v})}{\|(\frac{\partial X}{\partial u} \times \frac{\partial X}{\partial v})\|}. \tag{3}$$

Light Model We assume each image I^j is illuminated by an anisotropic point light source. We describe this light by a position $p^j \in \mathbb{R}^3$, a direction $d^j \in S^2$, and an angular attenuation coefficient $\mu^j \in \mathbb{R}$. We assume all lights have unit intensity. If that is not the case, we divide the image by the intensity of the light sources.

We can then describe the direction of the light arriving at a point X on the surface of the object by:

$$L^j(X) = \frac{(X - p^j)}{\|X - p^j\|}, \tag{4}$$

and the attenuation of the light at the same point by:

$$A^j(X) = \frac{(L^j \cdot d^j)^{\mu^j}}{||X - p^j||^2}. \tag{5}$$

Thus lighting at any pixel (u, v), given the depth map $D(u, v)$, can be described by a direction term $L^j(X(u, v))$ and an intensity attenuation term $A^j(X(u, v))$ (where X is expressed with depth D by eqn. 2). To keep it concise, we term these lighting factors, relative direction and attenuation, at each pixel **'per-pixel lighting'**.

Admissible lights The configuration of possible anisotropic point lights is huge, taking 3+2+1 parameters to describe. To remedy this we restrict ourselves to lights with positions in a cylinder around the camera and direction pointing roughly toward the object. We term this region the 'admissible light region'. It covers positions of lights used in most existing datasets (e.g. [21, 31]) and the uncalibrated data we capture. For the exact specification of the admissible light region please see the supplement.

Reflectance Model We model the reflectance as a general spatially varying BRDF that depends on the lighting direction ω_l, the viewing direction ω_v and the position on the surface X. Denote this as $B(\omega_l, \omega_v, X)$.

Rendering Equation Now given the depth map D, normal N, camera intrinsics K, and light parameters p^j,d^j,μ^j, we can write the rendering equation for the jth image as a function of (u, v):

$$I^j(u, v) = A^j(X)B(\omega_v, L^j(X))(N(u, v) \cdot L^j(X)) + \eta(u, v) \tag{6}$$

where η represents indirect lighting effects such as shadows and inter-reflections. Note that $\omega_v = -X/\|X\|$ because the camera is centered at the origin.

4. Our Approach

We aim to predict normal map N and depth map D, given a set of images $I^1, ..., I^M$. We propose a recursive solution to this problem. We introduce two recursive networks, one for predicting normal $G_{RN}(\cdot;\theta_{RN})$ and another for predicting depth $G_{RD}(\cdot;\theta_{RD})$ given normals. At each step of the recursion we increase the image resolution by a factor of two and use these two networks to predict the depth map and the normal map. For a robust and accurate normal estimation, we calculate the 'per-pixel lighting' (L^j and A^j) and use it as an input to the normal estimation network, which we ablate in Sec. 5.4.

Lichy *et al.* [17] introduced a similar recursive normal estimation network, RecNet, for far-field PS. They showed that the recursive network has a large receptive field and produces high-quality reconstruction by refining the predictions from the previous scale. We also find this idea to be suitable to produce fast and light-weight inference. Thus we developed our own version of recursively reconstructing the object for near-field PS which we describe in Sec. 4.1. Network architecture, training data and loss functions are described in detail in Sec. 4.2.

The key differences between our approach and [17] are:

- We create synthetic data for training that emulates near-field capture with lighting in the admissible region.
- We calculate the per-pixel lighting and use it as an extra input to the recursive normal estimation network, which improves performance by 3.5°, as shown in ablation study (Sec. 5.4).
- We introduce a recursive normal to depth integration network, which is fast and robust during training. Predicted depth map is then used for calculating the per-pixel lighting in the next scale.
- Unlike RecNet, which requires a fixed sequence of lights, our method is permutation invariant to lighting order and can use arbitrary lighting within the admissable region.

4.1. Recursive Reconstruction

We first initialize the recursion with input resolution of $r_0 = 64 \times 64$.

- We first calculate the 'per-pixel lighting' parameters $L_0^j(X)$ and A_0^j by assuming the depth map is a plane at depth 1 (see, Sec. 3 and supplement). This calculation is done following Algo. 1.
- Then we use an initial normal estimation network $G_{IN}(\cdot;\theta_{IN})$, which takes the input image and the per-pixel lighting parameters to predict the normal map N_0:

$$N_0 = G_{IN}(\{I_0^j, L_0^j, A_0^j\}_{j=1}^M); \theta_{IN}) \quad (7)$$

- Finally, we introduce another initialization network to predict a depth from the normals: $D_0 = G_{ID}(N_0;\theta_{ID})$

The recursive network progressively increases input image resolution by a factor of 2, until it reaches the input image resolution. The steps of the recursive network are in principle similar to the initialization network, except for the fact that the normal and depth estimation networks G_{IN} and G_{ID} do not use any recursion and simply predict at low resolution in a feed-forward fashion. The steps of the recursion are explained below:

- For each step i with resolution $r_i \times r_i$, we first calculate the per-pixel lighting (Algo. 1) using the depth map of the previous scale D_{i-1} upsampled by a factor of 2.
- Then normal map N_i is predicted with the recursive normal prediction network $G_{RN}(\cdot;\theta_{RN})$, given the input images and per-pixel lighting along with depth map D_{i-1} and normal map N_{i-1} of the previous scale following:

$$N_i = G_{RN}(\{I_i^j, L_i^j, A_i^j\}_{j=1}^M, N_{i-1}, D_{i-1}; \theta_{RN}) \quad (8)$$

- Finally we predict the depth map D_i from the normal map N_i using another recursive network $G_{RD}(\cdot;\theta_{RD})$, which is conditioned on the depth map of the previous scale D_{i-1}: $D_i = G_{RD}(N_i, D_{i-1};\theta_{RD})$

The forward pass of our recursive process is also summarized in Algo. 2.

Algorithm 1 Calculate the per-pixel lighting given depth D.

PPLight(K, D, μ, p, d)
$X[u,v] = D[u,v]K^{-1}(u,v,1)^T$
$L[u,v] = normalize(X[u,v] - p)$
$A[u,v] = \frac{(L[u,v]\cdot d)^\mu}{||x[u,v]-p||^2}$
return A, L

Algorithm 2 Forward pass of our approach: See Tab. 1 definition of the notation.

$L_0^j, A_0^j = \text{PPLight}(K, \text{ones}(r_0 \times r_0), \mu^j, p^j, d^j)$
$N_0 = G_{IN}(\{(I_0^j, L_0^j, A_0^j)\}_{j=1}^M; \theta_{IN})$
$D_0 = G_{ID}(N_0; \theta_{ID})$
for i = 1 to R-1 **do**
  $L_i^j, A_i^j = \text{PPLight}(K, Up(D_{i-1}), \mu^j, p^j, d^j)$
  $N_i = G_{RN}(\{(I_i^j, L_i^j, A_i^j)\}_{j=1}^M, N_{i-1}; \theta_{RN})$
  $D_i = G_{RD}(N_i, D_{i-1}; \theta_{RD})$
end for

4.2. Implementation Details

Network Architectures. Our method consists of four neural networks, two for initialization and two for recursion with similar architectures for initialization and recursion.

The normal estimation networks consist of a shared encoder that takes in each image I_i^j concatenated with its per-pixel lighting maps A_i^j and L_i^j and returns a feature F_i^j with dimension 128 at 1/4'th of the input resolution. In the recursion step, the normal from the previous step bilinearly upsampled by a factor of 2 is used as additional input. Then we perform a max pooling operation over the features F_i^js from all input images to produce a combined feature, which is passed to a decoder to produce a normal map.

The depth prediction network takes in the normal estimated by the normal prediction network (in the recursive case the encoder takes in the depth from the previous step bilinearly upsampled by a factor of 2) and produces a depth map. It also does some preprocessing to correct for a perspective camera. Specifically, it applies a transformation (e.g., see [30] or supplement) so that in the perspective case normal integration amounts to solving $\nabla u = (p, q)$ where u is the logarithm of depth and p,q are determined by the normal map and camera intrinsics. Architecturally, it is an encoder-decoder ResNet architecture similar to [17]. Details can be found in Sec. 4.3 and the supplement.

Loss Function. We train our network with three loss functions. We use depth loss L_{depth} and normal loss L_{normal} to produce accurate reconstruction. We also use a loss to ensure the normals derived from the predicted depth map are consistent with those derived from the ground truth depth map. This loss is necessary to produce smooth depth maps. We term this loss L_{nfd}, *nfd* is an abbreviation for 'normal from depth'. The losses are defined as:

$$L_{depth} = \sum_{i=0}^{R-1} ||D_i - \bar{D}_i||_1, \tag{9}$$

$$L_{normal} = \sum_{i=0}^{R-1} ||N_i - \bar{N}_i||_1, \tag{10}$$

$$L_{nfd} = \sum_{i=0}^{R-1} ||nfd(D_i) - nfd(\bar{D}_i)||_1, \tag{11}$$

where we use a bar above a letter to indicate the Ground Truth (GT) measurement. nfd is the function that takes a depth map and produces a normal map. This is implemented using eqn. 2 and 3. In eqn. 3 we approximate the derivatives with a central finite difference.

Training Details. Our network is trained completely on synthetic data. First we generate depth, normal, spatially-varying albedo, and Cook-Torrance roughness maps using 14 objects from the statue dataset [32] and freely available albedo maps from [1]. These are rendered at 512×512 resolution. At training time, for each normal, depth, albedo, and roughness, 10 lights are uniformly randomly sampled from the admissible region 3. With a 50% probability, we replace the object's material with one from the MERL dataset [20]. We then render the 10 images using eqn. 6.

For augmentation, we randomly zero patches and add random noise to each pixel to simulate the indirect lighting term η in eqn. 6. Images are also randomly cropped to simulate a diverse set of camera intrinsics. More details about data generation and augmentation can be found in the supplement. We trained our network end-to-end for 22 epochs using the Adam optimizer with learning rate 0.0001. Training took about 2 days on 4 Nvidia P6000 GPUs.

4.3. Normal Integration Network

We found that existing normal integration algorithms are too slow for use during training of a neural network. Additionally, they fail on our challenging synthetic data due to large discontinuities. Our solution is to replace a classical normal integration routine with a network, but this is a non-trivial task. Solving normal integration requires global information (details in supplement), but convolutional networks have limited receptive fields, and therefore cannot take global information into account for large enough images.

RecNet, a recursive architecture introduced in [17], creates a convolutional network with potentially infinite receptive field. We found a straight forward application of RecNet fails for normal integration. We believe this has to do with the relation between normals and depth. To understand this, we look at the opposite problem i.e. we want to train a network to predict normals from depth. To keep things simple let's consider the orthographic case in 1D, where estimating the normal is the same as estimating the derivative.

We consider an image as discrete samples of a function on domain $[0, 1]$. Let $0 = x_1, ..., x_r = 1$ be the sample points and let $h = 1/r$ be the distance between them, where r is the image resolution. Let u be the depth and $u_i = u(x_i)$. Let u' be the derivative of u and $[u']_i = u(x_i)$. Let $\{u_i\}$ indicate the sequence of all the elements u_i.

Suppose we train a fully convolutional network to predict normal $\{[u']_i\}$ from depth $\{u_i\}$ at a resolution r. It will learn something similar to a finite difference and return $\{\frac{u_{i+1}-u_{i-1}}{h}\}$. Now if we test the network on an image $\{v_i\}$ that has a higher resolution say e.g. $2r$. Then the network will predict $\{\frac{v_{i+1}-v_{i-1}}{h}\}$, but this is not the desired result. The correct result is $\{\frac{v_{i+1}-v_{i-1}}{h/2}\}$, this is because the network does not know the resolution has changed. In this case, there is a simple solution: predict $\{u_i\}$ from the resolution independent $\{[u']_i \cdot h\}$ instead of $\{[u']_i\}$

This suggests that when we solve the inverse problem we should try to learn a function G that takes $\{[u']_i \cdot h\}$ and predicts $\{u_i\}$: $\{u_i\} = G(\{[u']_i \cdot h\})$. This is impossible for a fully convolutional network because it requires global information. However, if we already know a low-resolution estimate of $\{u_i\}$, termed $\{w_j\}$, we then learn a function:

$$\{u_i\} = G(\{[u']_i \cdot h\}, \{w_j\}), \tag{12}$$

i.e. we predict depth from normal and a low resolution estimate of depth. We argue that this is possible for a fully convolutional network. By applying eqn. 12 recursively, we can gradually reconstruct a full resolution depth map. This is the essential idea of our depth prediction network. For more on this argument and the depth prediction network please see the supplement.

4.4. Lighting calibration

Setting up calibrated lights in-the-wild is very challenging. Recent works have shown that in the far-field case lighting calibration can be accomplished with a neural network [6, 15]. We are not aware of any learning based approach to near-field lighting calibration.

Since, in the near-field case, there is much more freedom of possible light configurations, we make some additional

simplifying assumptions on the light: (1) the light intensity is the same in all images. (2) the light is well modeled by an isotropic point source i.e $\mu = 0$ and d is irrelevant. (3) The light is within the admissible region. We found that these assumptions are good enough to estimate light from a handheld flashlight used for capture in-the-wild.

We use essentially the same architecture as [6] to estimate light positions. This network uses a shared feature extractor to extract a feature F^j from each image I^j. It then creates a context $c = \max_j F^j$. Finally, a second network is applied to feature F_j and the context c to produce a light position estimate p^j for image I^j. To deal with a perspective camera, all input images are cropped or zero padded to have the same intrinsics.

5. Experimental Evaluation

We evaluate our method quantitatively on the LUCES dataset [21] in Sec. 5.1, and qualitatively on a dataset we captured with a handheld flashlight and iPhone in Sec. 5.3. We mainly compare our results to two state-of-the-art near-field Photometric Stereo (PS) algorithms S20 [31] and L20 [18]. In the case of uncalibrated capture, we use our calibration network described in Sec. 4.4 for S20 and L20, which are only developed for calibrated lighting conditions.

5.1. Quantitative Evaluation on LUCES [21]

The LUCES dataset consists of 14 objects, each captured in HDR under 52 calibrated near-field lighting conditions. We evaluate using the mean angular error (MAE) for normal and mean depth error (MZE) metrics.

Calibrated. In Tab. 2, we present MAE and MZE obtained by our method and compare it with existing works as reported in LUCES [21]. The table includes results from two pure optimization near-field methods L17 [19] and Q17 [29], two hybrid near-field methods using deep learning and optimization S20 [31] and L20 [18], and the far-field deep method I18 [14]. All methods are evaluated at the 2048×1536 resolution, except S20 which was evaluated at 512×384 due to its GPU memory requirements [21].

Tab. 2 shows our method outperforms all existing methods in terms of MAE, especially the state-of-the-art method L20 (MAE 13.33° vs. 11.32°). Using our integration network we are the second best in MZE, L20 outperforming us by 1.26mm. We found that our normal integration network can develop jumps at discontinuities (see supplementary for details), which increase our MZE. We tried resolving this issue by integrating our normal map predictions as a post processing step. We used the optimization approach of [26], also used by L20 for this step. After this post processing, our MZE dropped to 2.93mm, and our method becomes the top performer. We label our method with this post processing step as MZE+int in Tab. 2. It is tempting to try replacing our normal integration network with a traditional integration algorithm during training, but we find these algorithms

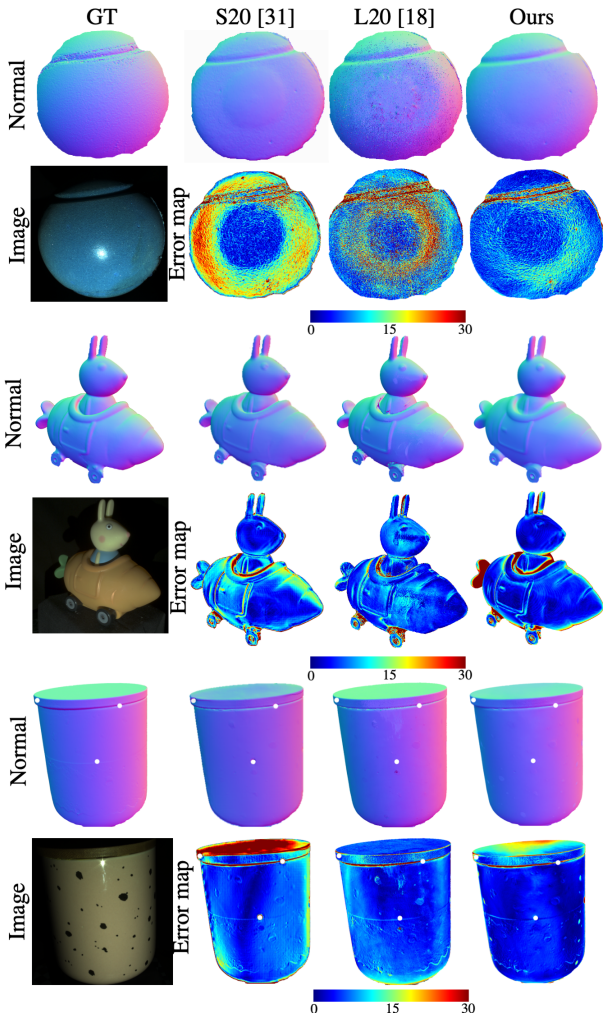

Figure 2. We compare the predicted normal map and an error map w.r.t. GT of our approach with that of S20 [31] and L20 [18] on sample objects from the LUCES [21] with calibrated lighting.

struggle with our synthetic training data due to discontinuities in the data. Furthermore, these methods are very slow for use during network training.

Tab. 2 also reports the MAE achieved by differentiating the ground truth depth map with a finite difference (Diff-MAE) and the depth error obtained by integrating the ground truth normal with [26] (Int-MZE). These errors are due to the discrete nature of images and discontinuities in the object, see [21] for detailed discussion.

We show normal prediction results and error maps from each of these methods in Fig. 2. Depth error visualizations are included in the supplement.

Uncalibrated. In Tab. 3 we compare our method to L20 and S20 where the ground truth lighting calibration is replaced with the results of our calibration network. Since our calibration network only handles the case of equal intensity lights we scale each image by their ground truth intensity. Additionally lights are assumed to be isotropic point

Method	Error	Bell	Ball	Buddha	Bunny	Die	Hippo	House	Cup	Owl	Jar	Queen	Squirrel	Bowl	Tool	Average
L17- [19]	MAE	28.25	9.77	11.5	20.15	11.95	15.42	29.69	30.76	13.77	10.56	13.05	15.93	12.5	15.1	17.03
	MZE	4.45	0.81	4.67	7.51	4.58	3.19	6.99	2.67	3.64	6.56	**1.89**	1.82	4.37	3.25	4.02
Q18- [29]	MAE	25.8	12.12	14.07	13.73	13.77	18.51	30.63	37.63	14.74	15.66	13.16	14.06	11.19	16.12	17.94
	MZE	12.03	2.5	9.28	7.06	5.91	6.8	8.02	4.83	5.83	16.87	6.92	2.55	6.48	6.69	7.27
S20- [31]	MAE	9.5	25.42	19.17	12.5	5.23	23.12	**28.02**	**14.22**	13.08	9.27	16.62	14.07	12.44	17.42	15.72
	MZE	1.9	5.5	5.53	6.02	2.76	7.04	**6.15**	**1.62**	3.75	6.09	3.91	2.81	5.22	4.68	4.5
L20- [18]	MAE	14.74	12.43	**10.73**	**8.15**	6.55	**7.75**	30.03	23.35	12.39	8.6	**10.96**	15.12	8.78	17.05	13.33
	MZE	**1.53**	**0.67**	**3.27**	2.49	4.44	**1.82**	9.14	2.04	3.44	**3.86**	1.94	**1.01**	**2.80**	5.90	3.17
I18- [14]	MAE	23.55	44.29	35.29	36	41.52	44.9	49.05	35.78	40.27	40.66	32.89	41.09	28.04	31.71	37.5
	MZE	5.93	6.59	10.92	6.88	7.83	7.59	8.98	3.17	8.67	15.54	8.08	5.8	6.69	12.45	8.22
Ours	MAE	**6.20**	**8.55**	12.69	8.63	**5.16**	8.01	29.00	17.28	**12.32**	**5.32**	12.90	**13.00**	**7.07**	**12.33**	**11.32**
	MZE	2.28	1.83	16.60	2.73	2.76	3.52	7.39	2.00	3.08	6.58	3.09	3.94	3.78	2.48	4.43
	MZE int	1.71	1.26	3.93	**2.16**	**2.16**	3.44	7.07	1.82	**2.85**	3.99	2.90	1.88	3.61	**2.27**	**2.93**
GT	Diff-MAE	2.5	2.69	2.69	2.93	2.49	3.2	9.19	2.85	4.3	1.79	4.22	3.26	2.27	2.34	3.34
	Int [26]-MZE	0.08	0.22	3.28	2.30	0.56	1.28	7.43	0.02	3.51	0.12	3.25	1.12	0.12	0.13	1.67

Table 2. Evaluation on LUCES dataset with calibrated lighting. Mean angular error (MAE in degrees) and mean depth error (MZE in mm).

Method	Error	Bell	Ball	Buddha	Bunny	Die	Hippo	House	Cup	Owl	Jar	Queen	Squirrel	Bowl	Tool	Average
S20- [31]	MAE	13.43	13.68	21.85	**11.41**	**5.86**	11.33	36.28	**17.63**	17.67	11.60	**15.96**	17.24	13.92	**16.54**	16.03
	MZE	2.98	2.58	8.48	4.91	2.52	**2.85**	10.31	**1.95**	6.44	6.18	4.85	3.18	4.39	**2.38**	4.57
L20- [18]	MAE	13.97	15.50	18.92	14.71	16.14	16.20	32.06	23.80	17.81	17.65	20.79	21.45	11.83	23.00	18.85
	MZE	4.21	3.21	10.39	4.88	5.58	4.29	12.04	2.18	5.59	10.14	8.27	3.26	3.32	6.12	5.96
Ours	MAE	**7.17**	**6.59**	**14.50**	11.75	8.63	**10.64**	**31.00**	18.98	**15.92**	**9.14**	18.39	**15.97**	**10.17**	18.61	**14.11**
	MZE	**1.80**	1.40	10.27	3.84	2.77	3.59	10.04	2.64	**4.13**	7.35	**4.59**	3.51	**3.15**	6.93	4.71
	MZE int	2.32	**0.76**	**5.52**	**2.79**	**2.41**	3.32	**8.68**	2.07	4.77	**4.97**	5.15	**2.60**	3.59	6.08	**3.93**

Table 3. Evaluation on LUCES with uncalibrated lighting. Mean angular error (MAE in degrees) and mean depth error (MZE in mm).

sources i.e. we set $\mu^j = 0$ in each method. Our method and L20 were evaluated at 1024×768 resolution. S20 was evaluated at 512×384 resolution due to GPU memory limitations. Results of S20 were then bilinearly upsampled to 1024×768 for error evaluation.

We again observe that our method is the best in MAE (14.11° vs. 16.03° for S20). Using our normal integration network we are slightly surpassed by S20 in MZE (4.57mm vs. 4.71mm). However, after using post-processing normal integration we improve our MZE to 3.93mm. Results and error maps are shown in Fig. 3.

5.2. Computational Resources

method	res.	time(s)	cpu (GB)	gpu (GB)
S20- [31]	512	2435.0	5	20
L20- [18]	512	59.5	8	5
Ours	512	1.3 (2.0)	4	9
L20- [18]	1024	200.0	27	17
Ours	1024	4.0 (6.9)	4	12

Table 4. Comparison of computational resources. Our method produces significantly faster inference while consuming less CPU and GPU memory than S20 and L20. The quantities in brackets for our method indicate post-processing normal integration. S20 cannot operate on 1024 resolution (res) due to memory limitations.

We compare the memory usage and inference speed of our method to that of S20 and L20 in Tab. 4. All methods were tested on the same machine with a 24GB Nvidia P6000 GPU and 128GB of main memory. We compare our method and L20 at two resolutions 512×384 and 1024×768. S20 is only compared at 512×384 due to its GPU memory requirements.

Inference Speed We calculate inference speed without the time required for data reading and writing, which can vary depending on the cluster load. Our method is 45× faster than the closest competitor L20 at both 512 and 1024 resolution. Adding the normal integration post-processing step to our method increases the runtime by about 50%, still leaving our method 30× faster than L20. S20 is over 1000× slower than our method.

CPU memory The amount of CPU memory used by S20 and our method are essentially fixed for a given number of input images. In contrast, L20 uses CPU memory approximately proportional to image resolution.

GPU memory At 512 resolution S20 requires 20GB of GPU memory to process LUCES, see Tab. 4. This was the highest resolution we were able to run on a 24GB GPU and is consistent with that reported in [21]. GPU usage for L20 and our method are more moderate, however there are some subtleties that must be taken into account to fairly compare GPU usage. See supplement for details.

	MAE	MZE
with per-pixel lighting estimation	**11.32**	**4.43**
w/o per-pixel lighting estimation	14.88	4.89

Table 5. We show that using per-pixel lighting as input to the recursive normal prediction network improves reconstruction accuracy.

5.3. Qualitative Comparison on Captured Data

We captured a dataset of medium to large size objects using an iPhone12 mounted on a tripod and a handheld flashlight. We capture a short video (5-10 second) of each object while moving the flashlight around the admissible region. Every fifth frame from the video was then used as input images. Images were inverse tonemapped by raising them to the power 2.2. Lighting positions were estimated with our calibration network.

We compare our method with that of S20 [31] and

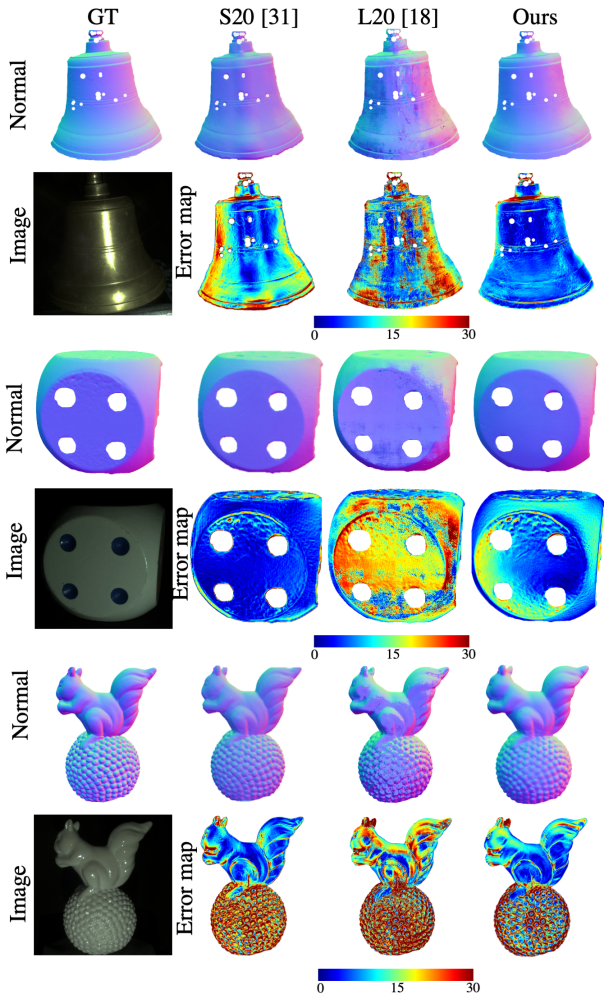

Figure 3. We compare the predicted normal map and an error map w.r.t. GT of our approach with that of S20 [31] and L20 [18] on sample objects from the LUCES [21] with uncalibrated lighting.

L20 [18], presented in Fig. 4 and 1, with more in supplementary. L20's performance is strongly affected by noise due to its per-pixel normal prediction network. S20's predicted normal maps show strong checkerboard patterns.

5.4. Ablation Study

Importance of Per-Pixel Lighting We train a variant of our network that does not require per-pixel lighting estimation as input to the normal estimation network. Concretely, rather than using the per-pixel lighting A_i^j and L_i^j as input to the network G_{IN} and G_{RN} we only input the lights' parameters p^j, d^j, μ^j at each pixel. Note that, these light parameters are associated with global lighting conditions and do not reflect per-pixel lighting effects unlike A_i^j and L_i^j. We observe that explicitly estimating and using per-pixel lighting as input improves the performance from 14.88° to 11.32° MAE. This also shows that a more direct adaptation of the recursion idea presented by Lichy *et al.* in [17] from far-field to near-field PS, is less effective without using per-pixel lighting estimation. We discuss this result in more depth in the supplement.

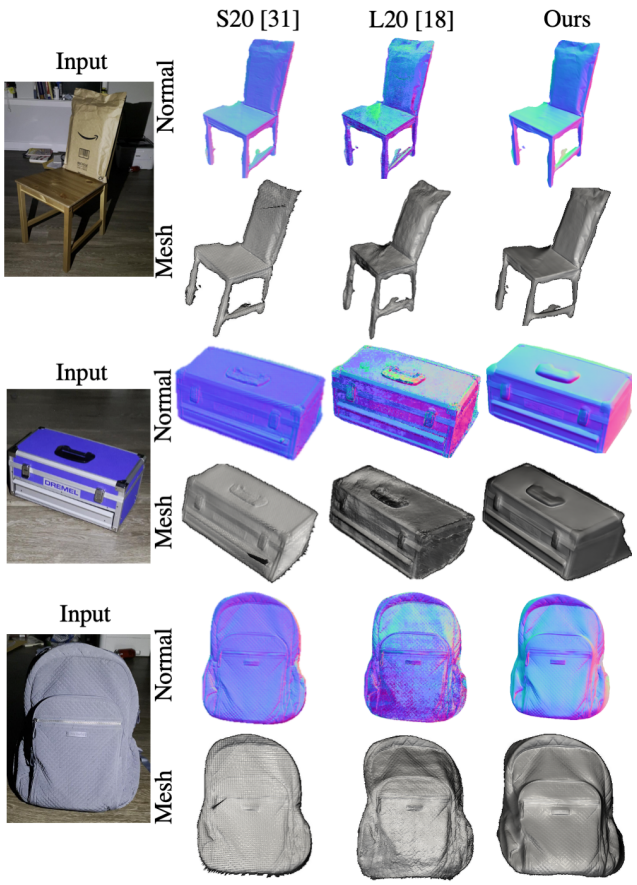

Figure 4. Qualitative evaluation on images captured with a hand-held flashlight and iPhone 12 camera mounted on a tripod. Our method outperforms L20 and S20.

6. Conclusion

In this work, we introduce a fast light-weight solution to near-field Photometric Stereo. Existing approaches rely on optimization with point or patch based inferences, which are more susceptible to noise, lack global context, and are memory intensive and slow. The key innovation of this work lies in creating a system that can capture global context to produce accurate predictions while being light-weight and fast. We adapt [17] from far-field PS to near-field by introducing per-pixel lighting estimation, a recursive normal integration network, and extend it to handle arbitrary lighting. We show in an ablation study that a straightforward adaption of [17] produces worse results than our approach. We also introduce a calibration network and show how our approach can be used for capturing 3D geometry of large and midsize real-world objects, like furniture, backpacks, etc. Our method significantly outperforms state-of-the-art methods on both the LUCES dataset and real-world captures while being fast and light-weight.

Acknowledgment This research is supported by the NSF under grant no. IIS-1910132.

References

[1] 3d textures. https://3dtextures.me/. Accessed: 2020. 5

[2] Jens Ackermann and Michael Goesele. A survey of photometric stereo techniques. *Foundations and Trends® in Computer Graphics and Vision*, 9(3-4):149–254, 2015. 2

[3] Jahanzeb Ahmad, Jiuai Sun, Lyndon Smith, and Melvyn Smith. An improved photometric stereo through distance estimation and light vector optimization from diffused maxima region. *Pattern Recognition Letters*, 50:15–22, 2014. 1, 2

[4] P.N. Belhumeur, D.J. Kriegman, and A.L. Yuille. The bas-relief ambiguity. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1060–1066, 1997. 1

[5] Alexandre Bony, Benjamin Bringier, and Majdi Khoudeir. Tridimensional reconstruction by photometric stereo with near spot light sources. *21st European Signal Processing Conference (EUSIPCO 2013)*, pages 1–5, 2013. 2

[6] Guanying Chen, Kai Han, Boxin Shi, Yasuyuki Matsushita, and Kwan-Yee K. Wong. Sdps-net: Self-calibrating deep photometric stereo networks. In *CVPR*, 2019. 2, 3, 5, 6

[7] Guanying Chen, Michael Waechter, Boxin Shi, Kwan-Yee K Wong, and Yasuyuki Matsushita. What is learned in deep uncalibrated photometric stereo? In *European Conference on Computer Vision*, 2020. 3

[8] Toby Collins and Adrien Bartoli. 3d reconstruction in laparoscopy with close-range photometric stereo. *Medical image computing and computer-assisted intervention : MICCAI ... International Conference on Medical Image Computing and Computer-Assisted Intervention*, 15 Pt 2:634–42, 2012. 2

[9] Jean-Denis Durou, Maurizio Falcone, Yvain Quéau, and Silvia Tozza. Advances in photometric 3d-reconstruction, 2020. 2

[10] Bjoern Haefner, Zhenzhang Ye, Maolin Gao, Tao Wu, Yvain Quéau, and Daniel Cremers. Variational uncalibrated photometric stereo under general lighting. 11 2019. 3

[11] Aaron Hertzmann and Steven M. Seitz. Example-based photometric stereo: Shape reconstruction with general, varying brdfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1254–1264, aug 2005. 3

[12] Jeffrey Ho, Jongwoo Lim, Ming Hsuan Yang, and David Kriegman. Integrating surface normal vectors using fast marching method. In *Computer Vision - ECCV 2006, 9th European Conference on Computer Vision, Proceedings*, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), pages 239–250, 2006. 9th European Conference on Computer Vision, ECCV 2006 ; Conference date: 07-05-2006 Through 13-05-2006. 3

[13] Berthold K.P Horn and Michael J Brooks. The variational approach to shape from shading. *Computer Vision, Graphics, and Image Processing*, 33(2):174–208, 1986. 2

[14] Satoshi Ikehata. Cnn-ps: Cnn-based photometric stereo for general non-convex surfaces. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 2, 6, 7

[15] Berk Kaya, Suryansh Kumar, Carlos Oliveira, Vittorio Ferrari, and Luc Van Gool. Uncalibrated neural inverse rendering for photometric stereo of general surfaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3804–3814, June 2021. 3, 5

[16] Jingtang Liao, Bert Buchholz, Jean-Marc Thiery, Pablo Bauszat, and Elmar Eisemann. Indoor scene reconstruction using near-light photometric stereo. *IEEE Transactions on Image Processing*, 26:1089–1101, 2017. 1, 3

[17] Daniel Lichy, Jiaye Wu, Soumyadip Sengupta, and David W. Jacobs. Shape and material capture at home. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6123–6133, June 2021. 2, 4, 5, 8

[18] Fotios Logothetis, Ignas Budvytis, Roberto Mecca, and Roberto Cipolla. A cnn based approach for the near-field photometric stereo problem. *ArXiv*, abs/2009.05792, 2020. 1, 2, 3, 6, 7, 8

[19] Fotios Logothetis, Roberto Mecca, and Roberto Cipolla. Semi-calibrated near field photometric stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 6, 7

[20] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Transactions on Graphics*, 22(3):759–769, July 2003. 5

[21] Roberto Mecca, Fotios Logothetis, Ignas Budvytis, and Roberto Cipolla. Luces: A dataset for near-field point light source photometric stereo. *ArXiv*, abs/2104.13135, 2021. 2, 3, 6, 7, 8

[22] Roberto Mecca, Yvain Quéau, Fotios Logothetis, and Roberto Cipolla. A single-lobe photometric stereo approach for heterogeneous material. *SIAM J. Imaging Sci.*, 9:1858–1888, 2016. 2

[23] Roberto Mecca, Aaron Wetzler, A. Bruckstein, and R. Kimmel. Near field photometric stereo with point light sources. *SIAM J. Imaging Sci.*, 7:2732–2770, 2014. 2

[24] Ying Nie, Zhan Song, Ming Ji, and Lei Zhu. A novel calibration method for the photometric stereo system with non-isotropic led lamps. *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 289–294, 2016. 2

[25] Thoma Papadhimitri and Paolo Favaro. Uncalibrated near-light photometric stereo. *BMVC 2014 - Proceedings of the British Machine Vision Conference 2014*, 01 2014. 1, 2

[26] Yvain Quéau and Jean-Denis Durou. Edge-preserving integration of a normal field: Weighted least-squares, tv and l^1 approaches. In *SSVM*, 2015. 6, 7

[27] Yvain Quéau, Roberto Mecca, and Jean-Denis Durou. Unbiased photometric stereo for colored surfaces: A variational approach. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4359–4368, 2016. 2

[28] Yvain Quéau, Tao Wu, and Daniel Cremers. Semi-calibrated near-light photometric stereo. In *SSVM*, 2017. 1, 2

[29] Yvain Quéau, Bastien Durix, Tao Wu, Daniel Cremers, Francois Lauze, and Jean-Denis Durou. Led-based photometric

stereo: Modeling, calibration and numerical solution. *Journal of Mathematical Imaging and Vision*, 60, 03 2018. 2, 6, 7

[30] Yvain Quéau, Jean-Denis Durou, and Jean-François Aujol. Normal integration: A survey. *Journal of Mathematical Imaging and Vision*, 60, 05 2018. 2, 3, 5

[31] Hiroaki Santo, Michael Waechter, and Yasuyuki Matsushita. Deep near-light photometric stereo for spatially varying reflectances. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 3, 6, 7, 8

[32] Olivia Wiles and Andrew Zisserman. Silnet : Single- and multi-view reconstruction by learning from silhouettes. In *British Machine Vision Conference 2017, BMVC 2017, London, UK, September 4-7, 2017*. BMVA Press, 2017. 5

[33] Robert J. Woodham. Photometric Method For Determining Surface Orientation From Multiple Images. *Optical Engineering*, 19(1):139 – 144, 1980. 1, 2

[34] Wuyuan Xie, Chengkai Dai, and Charlie C. L. Wang. Photometric stereo with near point lighting: A solution by mesh deformation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4585–4593, 2015. 1, 2