Fast Light-Weight Near-Field Photometric Stereo: Supplementary Material

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

1. Overview

This supplement includes additional details that could not be put into the main paper due to space restrictions. In Sec. 3 we define a relative coordinate system (where mean depth is 1) and show how to relate absolute coordinates to this coordinate system. In Sec. 4, we define the admissible light region in terms of this relative coordinate system. In Sec. 5, we discuss the normal integration problem and elaborate more on our depth prediction network. In Sec. 6 we give the low-level details of our network architectures. In Sec. 7, we add some additional details regarding GPU memory usage. In Sec. 8, we explain why the network without per-pixel lighting from our ablation study performs poorly. In Sec. 9, we demonstrate how certain errors can form in our depth prediction network. Finally, in Sec. 10, we present some additional experimental results.

2. Code

Code for this paper will be released on Github once the paper is accepted for publication.

3. Coordinate System and Scale

This section shows how to pick a coordinate system with a mean depth of one and how this resolves the global scale ambiguity in the uncalibrated case. Equations from the main paper Sec. 3 are included below for quick reference.

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

$$L^j(X) = \frac{(X-p^j)}{\|X-p^j\|}, \tag{2}$$

$$A^j(X) = \frac{(L^j \cdot d^j)^{\mu^j}}{||X-p^j||^2}. \tag{3}$$

$$I^j(u,v) = A^j(X)B(\omega_v, L^j(X))(N(u,v)\cdot L^j(X)) + \eta(u,v) \tag{4}$$

Light scale We assume that all images have the same light intensity. However, we train our model such that the exact value of this intensity is unimportant, i.e. if we multiply all the images by a constant factor, we get the same results.

We get this effect by dividing each image by the mean intensity of the first image I^0 in the input set i.e. $\mu_{intensity} = \text{mean}_{u,v} I^0(u,v)$ then the input to the network is the image set $I^j(u,v)/\mu_{intensity}$.

Mean depth We assume mean depth is known. We can assume the mean depth is one by changing units. In particular, if μ_{depth} is the mean depth then we can replace $D(u,v)$ with $D^j(u,v)/\mu_{depth}$ and p^j with p^j/μ_{depth}. From equations 1, 2, 3, and 4 we see that this just scales the image intensity by μ_{depth}^2, but, as stated above, our network is invariant to the image intensity scale factor.

Uncalibrated Scale Ambiguity There is a global scale factor ambiguity between the light positions and depth, which is exactly why we can assume the mean depth is one. We train our calibration network on data with mean depth one, so the network predicts lights in this relative coordinate system. This resolves the scale ambiguity.

4. Admissible Light Region

Now that we have defined our relative coordinate system with mean depth one 3, we can define the admissible light region in terms of it.

We define the admissible light region as a cylinder with its axis along the camera optical axis (a.k.a. z-axis) and radius 0.75. The extent of the cylinder is from 0.15 behind the camera plane to 0.15 in front of the camera plane. Furthermore, we specify the admissible light directions as the directions making an angle of 30° or less with the z-axis.

Note that in absolute units, the size of the admissible region depends on the distance the object is from the camera. For example, if we are capturing a big object, we would place the camera farther away, and thus in absolute units, the admissible region will be larger.

5. Normal Integration

In this section, we present the mathematical intuition that inspired our depth prediction network. We then explain the details of the network's application. This section is not particularly rigorous, but we found that the network it inspired works well in practice.

5.1. High Level Idea

Problem Statement Give functions p and q on some domain, we want to find a function on the domain satisfying the PDE

$$\nabla U = (p, q) \tag{5}$$

This is equivalent estimating depth from a normal map, see 5.3. In general, a solution U may not exist. In which case, we want to find some approximate solution.

Necessity of Global Information We can see that solving eq. 5 requires global information as follows. Observe that given any solution to eq. 5 we can obtain another solution by adding a constant to it. Now suppose we broke the domain of interest into patches and produced a solution for each patch. Because each patch solution could have a different offset, we would have to look outside the patch to find the proper offset needed to glue the patch solutions together continuously. Therefore, a standard feed-forward network, which can only look at patches in very high-resolution images due to its limited receptive field, can not generalize to high-resolution data.

Proposed Solution Suppose we divide the depth into patches as before, but we are given the mean depth of each patch. Then we could solve the equation on each patch and set the patch mean to the given mean, thus producing a solution.

Concretely, consider a rectangular domain. Divide the region into smaller rectangles call them P_1,...,P_N. Let $\mu_k = \text{mean}_{x \in P_k} U(x)$. Suppose we knew the function $V(x)$ given by

$$V(x) = \mu_k \text{ if } x \in P_k \tag{6}$$

i.e. V is constant on the patches.

Now we solve 5 individually on each patch P_k, call the solution U_k. Furthermore, choose the U_k such that they have mean value zero. Define

$$W(x) = U_k(x) \text{ if } x \in P_k \tag{7}$$

Then we can produce a solution to 5 as

$$U(x) = W(x) + V(x) \text{ if } x \in P_k \tag{8}$$

In other words

$$\nabla W = \nabla(U(x) - V(x)) = (p, q) - \nabla V(x) \tag{9}$$

can be solved by just looking at the individual patches P_k. Then $U(x)$ satisfying 5 can be recovered as $W(x) + V(x)$. Thinking of V as an upsampled version of a low-resolution approximation to U is the motivation for our depth prediction network explained next.

5.2. Depth Prediction Network

Now we give the details of the depth prediction networks forward pass. The main paper does not distinguish between the preprocessing the depth prediction network does and the convolutional network proper. Here we use G_{ID} and G_{RD} for the networks and preprocessing combined (G_{*D} to refer to both) and NET_{*D} to refer to the convolution nets proper.

Algo. 1 gives the forward pass for G_{*D} (in the initial network, G_{ID}, the input depth is just a plane at $z = 1$). Where $\mathcal{D}$ is the central finite-difference

$$\mathcal{D}[U] = (U_{(m+1)n} - U_{(m-1)n}, U_{m(n+1)} - U_{m(n-1)}) \tag{10}$$

Note that in Algo. 1 line (**) is just the discrete analog of Eq. 9, where we multiply (p, q) by the step size h_i for the reasons explained in the main paper.

Algorithm 1 Forward pass of depth prediction network

```
1: $G_{*D}(N_i, D_{i-1})$
2: $p_i, q_i =$ from $N_i$ using perspective correction
3: $U_{i-1} = ln(Upsample[D_{i-1}])$
4: $res = NET_{*D}(h_i(p_i, q_i) - \mathcal{D}[U_{i-1}])$ (**)
5: $U_i = U_{i-1} + res$
6: $D_i = exp(U_i)$
7: **return** $D_i$
```

5.3. Perspective correction

Suppose an image is taken with focal length f, depth, $D(u, v)$ and normals, $N(u, v)$. Define $U = ln(D)$ and $p = -\frac{N_1}{uN_1+vN_2+fN_3}$ and $q = -\frac{N_2}{uN_1+vN_2+fN_3}$. Then U,p, and q satisfy 5. Therefore, given normals we can make this transformation and solve for U then recover D as $D = exp(U)$. For a derivations see [4].

5.4. Alternative Fast Normal Integration Methods

We considered FFT and DCT based integration methods that are fast enough for network training. However, because these assume a periodic or rectangular domain, they perform poorly on the datasets we tested, which all have irregular domains. For more discussion on this issue please see [4] Sec. 3.3 and 3.4.

6. Network Architectures

In this section, we define the low-level architectures of all the networks used in the paper. In Tab. 1 we define the notation used to describe network layers.

6.1. Normal Prediction Network

Both the initial, G_{IN}, and recursive, G_{RN}, normal prediction networks have the same architecture. They consist of a shared feature extractor defined by

A-B	apply layer A then layer B
BN	BatchNorm
Relu(x)	Leaky Relu activation with parameter x if '(x)' is omitted x=0 (i.e. standard Relu)
conv_kn_fm_sp	convolution layer with kernel of size n with m filters and stride p. If stride is 1 we will omit _s1.
Conv_kn_fm_sp	conv_kn_fm_sp - BN - Relu
ConvL_kn_fm_sp	conv_kn_fm_sp - BN - Relu(0.1)
convt_kn_fm_sp	transposed convolution layer with kernel of size n and m filters and stride p.
Res_n	conv_k3_fn - BN - Relu - conv_k3_fn - BN - +input. This defines the residual block. +input indicates adding the input value to the output value.
Upsample	bicubic upsampling by a factor of 2
tanh	hyperbolic tangent activation

Table 1. Definition of notations for network architecture

- FE = Conv_k7_f32 - Res_32 - Res_32 - Conv_k3_f64_s2 - Res_64 - Res_64 - Conv_k3_f128_s2 - Res_128 - conv_k3_f128

and a normal regressor defined by

- NR = Res_128 - convt_k3_f64_s2 - BN - Relu - Res_64 - Res_64 - convt_k3_f32_s2 - BN - Relu - conv_k7_f3

The application of the network is given by

$$N_i = \text{NR}(\max_{j=1}^{M} \text{FE}(I_i^j, Upsample(N_{i-1}))) + Upsample(N_{i-1}) \tag{11}$$

Due to the max-pooling the network is invariant to the image ordering.

6.2. Depth Prediction Network

The architectures of the initial depth prediction network, G_{ID}, and the recursive depth prediction network, G_{RD}, are the same. They are given by

- Conv_k7_f32 - Res_32 - Res_32 - Conv_k3_f64_s2 - Res_64 - Res_64 - Conv_k3_f128_s2 - Res_128 - Res_128 - Upsample - Conv_k3_f64 - Res_64 - Res_64 - Upsample - Conv_k3_f32 - conv_k7_f1 - tanh

The full application of the normal prediction network is defined in Sec. 5.

6.3. Light Prediction Network

The general architecture of the light prediction network is taken from [1]. Like the normal prediction network, it consists of a feature extractor LFE and a regressor LR defined by

- LFE = ConvL_k3_f64_s2 - ConvL_k3_f128_s2 - ConvL_k3_f128 - ConvL_k3_f128_s2 - ConvL_k3_f128 - ConvL_k3_f256_s2 - ConvL_k3_f256
- LR = ConvL_k3_f256 - ConvL_k3_f256_s2 - ConvL_k3_f256_s2 - ConvL_k3_f256_s2

It also has three final coordinate regressors CR_i for each coordinate $x = x_1$, $y = x_2$, $z = x_3$ defined by:

- CR_i = ConvL_k1_f64 - convL_k1_f1

First the network extracts a feature F^j from each image with the LFE network $F^j = \text{LFE}(I^j)$. It then forms a context $c = \max_j F^j$. Then for each feature F^j it applies the LR network to F^j concatenated with the context c to form position features $PF^j = \text{LR}(F^j, c)$. Finally, it applies the coordinate regessor CR_i to each position feature PF^j to get the light coordinates i.e. the k coordinate of the light position in image j is:

- $p_k^j = \text{CR\_i}(PF^j)$

7. GPU Memory Usage Details

method	res.	time(s)	cpu (GB)	gpu (GB)
S20- [5]	512	2435.0	5	20
L20- [2]	512	59.5	8	5
Ours	512	1.3 (2.0)	4	9
L20- [2]	1024	200.0	27	17
Ours	1024	4.0 (6.9)	4	12

Table 2. 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.

This section includes some additional information regarding the GPU memory usage of method L20 and our method.

L20 The memory usage of L20 is dependent on the batch size (number of pixels) that the network processes at one time. For our experiments, we used a batch size of 512. GPU memory usage could potentially be reduced by decreasing the batch size.

Ours Despite our method requiring 12GB of GPU memory to run LUCES at 1024 on our cluster, with a Nvidia P6000, we were able to run LUCES at 2048 resolution on a desktop computer with an 8GB Nvidia RTX2080 GPU. This indicates that our method is even more light-weight than indicated in table 2, which is also in the main paper.

8. Ablation Details

This section explains why the network without per-pixel lighting performs worse than with per-pixel lighting.

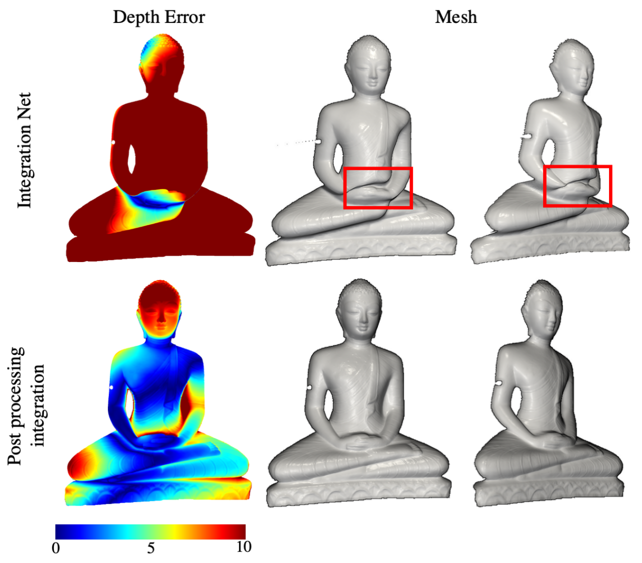

Figure 1. Depth prediction network developing jumps at discontinuities. The jump is highlighted in the red box. Traditional normal integration method [3] solves this issue (bottom row).

A convolutional neural network essentially applies the same function to each input patch. Suppose we have two patches, one on the left side of the image and one on the right side, that appear the same. In the ablated network with only global light positions as input, the patches look identical, and the network must predict the same normal. However, the lighting direction and intensity at these two patches are really different, so the patches need to be interpreted differently. The network using per-pixel lighting can distinguish between these two patches because they have different per-pixel lighting, and therefore it can produce different accurate results.

9. Limitations

Jumps at discontinuities As we mention in the main text, our depth prediction method can develop jumps at discontinuities. We show the most extreme example of this type of jump in Fig. 1.

10. Additional Results

This section we presents some additional results and figures.

8 Image Results We compared S20 [5], L20 [2], and our method on a subset of 8 randomly selected images from the LUCES dataset. They are images: 5, 7, 13, 19, 26, 39, 48, 50. We show the results in Tab. 3. S20's [5] code has a bug that caused it to fail with NaN values on two objects: Bell and House. Therefore, we report mean errors for all objects (average 14) for L20 and our methods, and the average over the 12 objects that S20 worked on (average 12).

We observe that our method only drops 1.2° MAE (from 11.32° to 12.44°) when tested on this subset. Whereas the other methods drop nearly 3° MAE.

Additional Results on Our Data In Fig. 2, we compare the results of S20 [5], L20 [2], and our method on data we captured.

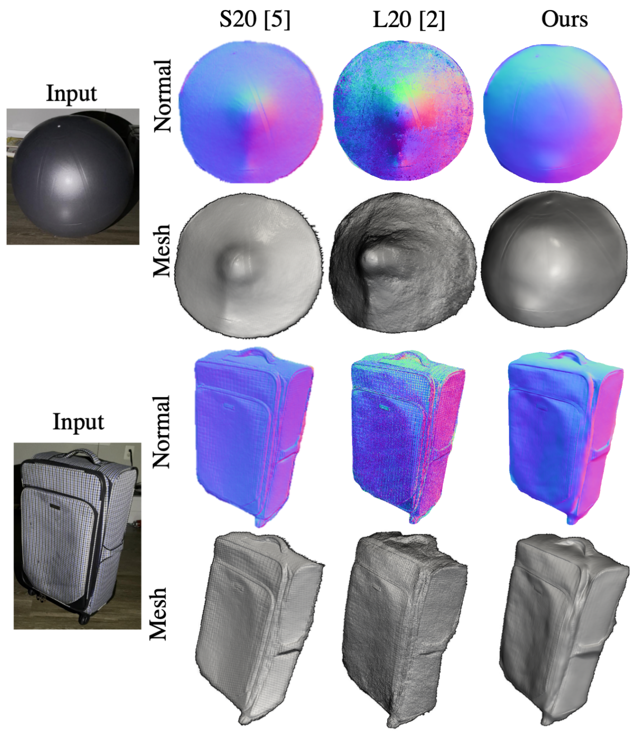

Figure 2. Additional results on data captured by us.

Additional Calibrated Results In Fig. 3 we present some additional normal maps from our tests on the LUCES dataset in the calibrated case. In Fig. 4 we show the depth error maps for each method in the calibrated case.

Additional Uncalibrated Results In Fig. 5 we present some additional normal maps from our tests on the LUCES dataset in the uncalibrated case. In Fig. 6 we show the depth error maps for each method in the uncalibrated case.

Method	Error	Bell	Ball	Buddha	Bunny	Die	Hippo	House	Cup	Owl	Jar	Queen	Squirrel	Bowl	Tool	Average 14	Average 12
S20- [5]	MAE		18.08	27.2	11.69	10.06	12.58		23.06	13.36	14.19	16.93	18.59	12.46	15.32		16.13
	MZE		2.91	6.09	3.85	3.31	**2.37**		3.62	4.52	8.83	3.30	3.13	**3.71**	3.62		4.10
L20- [2]	MAE	20.03	24.43	**12.67**	11.85	7.18	14.12	30.74	25.63	15.72	9.22	**13.12**	15.68	17.88	19.01	16.95	15.54
	MZE	3.42	6.44	**4.15**	3.20	**1.78**	3.22	8.49	3.33	5.73	4.47	4.26	2.05	8.08	9.64	4.87	4.69
Ours	MAE	**8.84**	**9.64**	13.59	**9.31**	**5.99**	**8.75**	**29.43**	**21.62**	**11.43**	**7.13**	13.38	**13.10**	**8.73**	**13.18**	**12.44**	**11.32**
	MZE	2.04	2.12	13.27	3.21	2.91	3.20	7.13	2.85	**3.51**	7.84	3.06	3.68	3.84	3.04	4.41	4.38
	MZE int	**1.68**	**1.46**	4.70	**2.22**	2.43	3.13	**6.21**	**2.23**	4.01	**4.41**	**3.01**	**1.97**	4.09	**3.07**	**3.19**	**3.06**

Table 3. Evaluation on LUCES with only 8 input images per object with calibrated lighting. Mean angular error (MAE in degrees) and mean depth error (MZE in mm). S20 failed with NaN errors on Bell and House. Average 14 is the average error with all 14 LUCES objects and Average 12 is the average error of the objects excluding Bell and House.

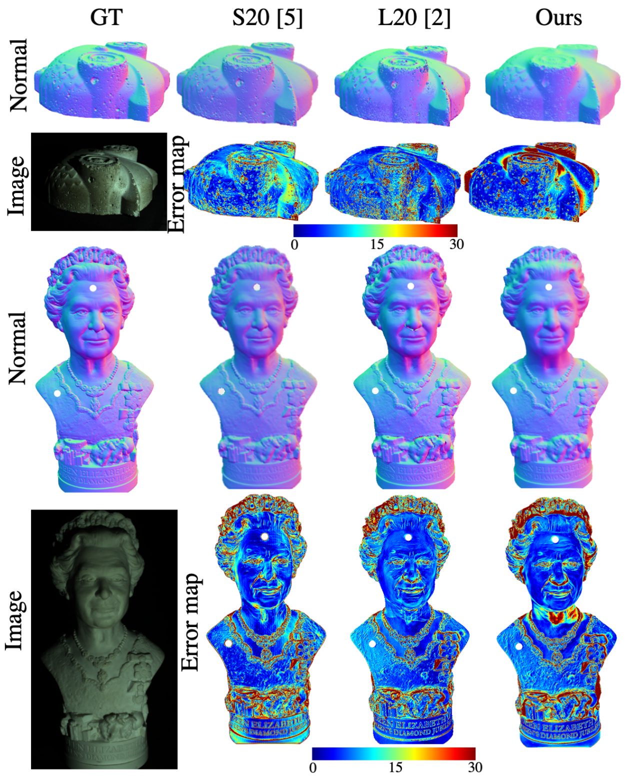

Figure 3. Additional normal predictions and error maps (in degrees) on the LUCES dataset in the calibrated case.

References

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

[2] 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. 3, 4, 5

[3] Yvain Quéau and Jean-Denis Durou. Edge-preserving integration of a normal field: Weighted least-squares, tv and l^1 approaches. In *SSVM*, 2015. 4

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

[5] Hiroaki Santo, Michael Waechter, and Yasuyuki Matsushita. Deep near-light photometric stereo for spatially varying reflectances. In *European Conference on Computer Vision (ECCV)*, 2020. 3, 4, 5

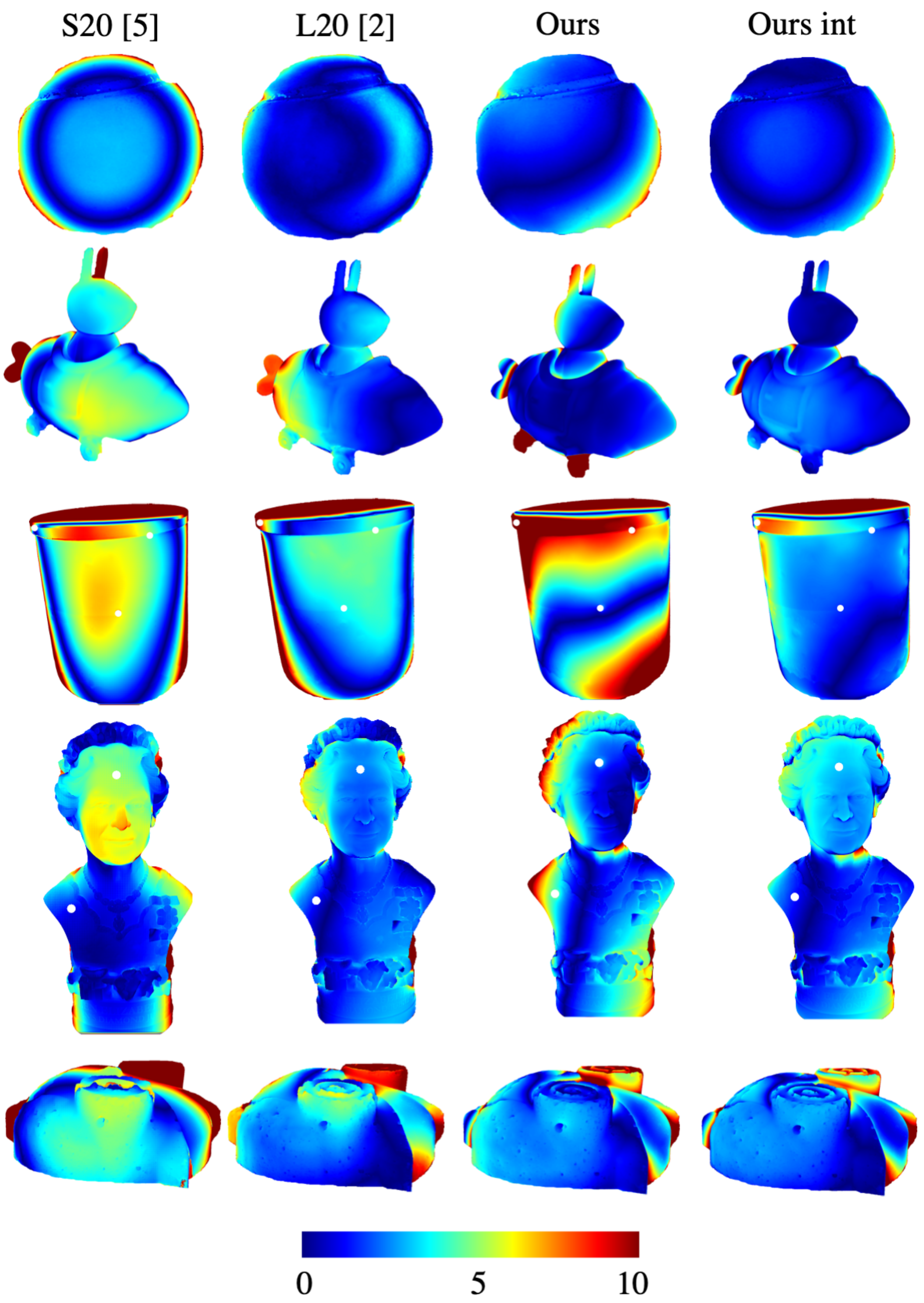

Figure 4. Depth error maps (in mm) on the LUCES dataset in the calibrated case

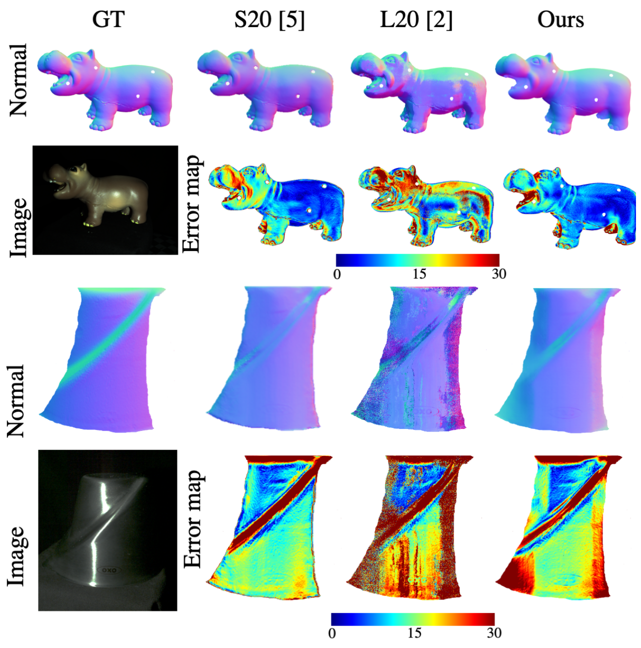

Figure 5. Additional normal predictions and error maps (in degrees) on the LUCES dataset in the uncalibrated case.

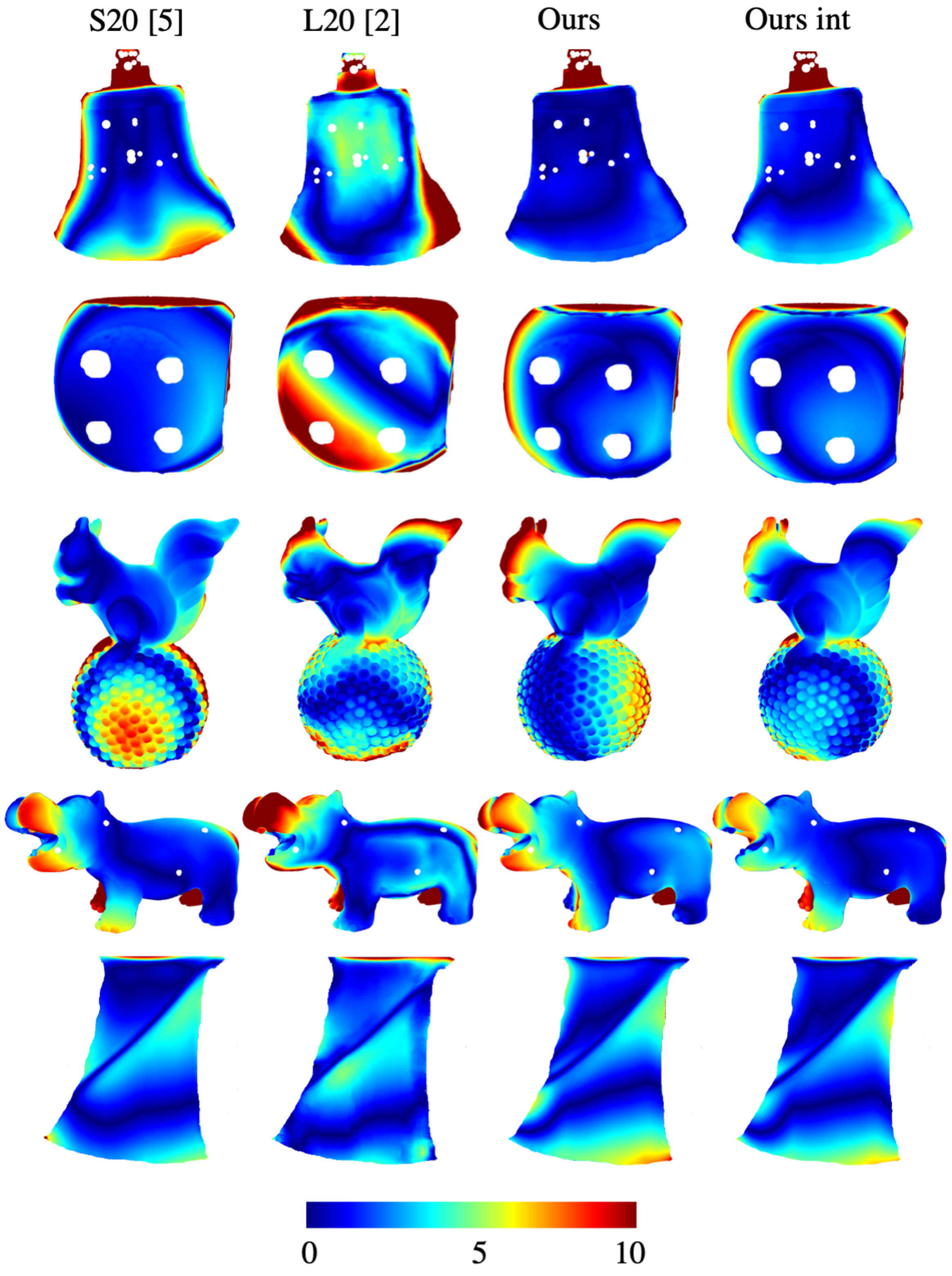

Figure 6. Depth error maps (in mm) on the LUCES dataset in the uncalibrated case