PIE-Net: Photometric Invariant Edge Guided Network for Intrinsic Image Decomposition - Supplementary

1. Image Formation

In this section we briefly introduce the image formation model. We follow this with the physics-based descriptors.

1.1. Image Formation

The process of image formation can be modelled as the combination of an object's material and geometric property under a light source. Under the Lambertian assumption [5], it is:

$$I = m(\vec{n}, \vec{l}) \int_{\omega} \rho_b(\lambda)\; e(\lambda)\; f(\lambda)\; \mathrm{d}\lambda\;, \tag{1}$$

where, I is the final observed image. λ is the incoming wavelength of the light integrated over the entire visible spectrum ω. $\vec{n}$ denotes the surface normal while $\vec{l}$ denotes the incident light direction. m is a function denoting the interaction of $\vec{n}$ & $\vec{l}$. f indicates the spectral camera sensitivity. e describes the spectral power distribution of the illuminant. While ρ denotes the reflectance of the object.

Discretising the model, we obtain:

$$C_{p_1} = m(\vec{n},\; \vec{l})\; e^{C_{p_1}}(\lambda)\; \rho^{C_{p_1}}(\lambda)\;, \tag{2}$$

where C_{p_1} is colour channel C for pixel p_1 for a RGB image.

Assuming a white light source, a narrow band filter and linear sensor response of the device, Eq (2) can be further simplified as:

$$I = S \times R\;, \tag{3}$$

where the $m(\vec{n},\; \vec{l})$ component is denoted as S and $e^{C_{p_1}}(\lambda)$ & $\rho^{C_{p_1}}(\lambda)$ components as R, respectively. Consecutively, S is the shading component associated with only the geometry and illuminant of the scene. While R is the reflectance (albedo) image, corresponding to the (true) colour of the object, independent of any geometric or lighting information. Thus, the simplified image formation becomes the pixel-wise multiplication of the reflectance and the shading images.

1.2. Cross Color Ratios: Proof of Invariance

The detailed steps of deriving the CCR is shown. Given the CCR (M_{RG} for the channel RG pair for pixels p_1 and p_2:

$$M_{RG} = \frac{R_{p_1}\; G_{p_2}}{R_{p_2}\; G_{p_1}}\;, \tag{4}$$

Taking logarithm on both sides of the equation, we get:

$$log(M_{RG}) = \; log(R_{p_1}\; G_{p_2}) \; - \; log(R_{p_2}\; G_{p_1})\;, \tag{5}$$

Combining Eq (2) and Eq (5):

$$\begin{aligned}
log(M_{RG}) = &\quad log(R_{p_1}\; G_{p_2}) \; - \; log(R_{p_2}\; G_{p_1})\;, \\
log(M_{RG}) = &\quad log(R_{p_1}) \; + \; log(G_{p_2}) \\
& - \; log(R_{p_2}) \; - \; log(G_{p_1})\;, \\
log(M_{RG}) = &\quad log(m(\vec{n_{p_1}}\; \vec{l_{p_1}})) \; + \; log(e^{R_{p_1}}(\lambda)) \\
& + \; log(\rho^{R_{p_1}}(\lambda)) \; + \; log(m(\vec{n_{p_2}}\; \vec{l_{p_2}})) \\
& + \; log(e^{G_{p_2}}(\lambda)) \; + \; log(\rho^{G_{p_2}}(\lambda)) \\
& - \; log(m(\vec{n_{p_2}}\; \vec{l_{p_2}})) \; - \; log(e^{R_{p_2}}(\lambda)) \\
& - \; log(\rho^{R_{p_2}}(\lambda)) \; - \; log(m(\vec{n_{p_1}}\; \vec{l_{p_1}})) \\
& - \; log(e^{G_{p_1}}(\lambda)) \; - \; log(\rho^{G_{p_1}}(\lambda))\;,
\end{aligned} \tag{6}$$

Recall from the main paper:

$$e^{C_{p_1}} = e^{C_{p_2}}\;, \tag{7}$$

Even for a curved surfaces, this holds true, since they are very close to each other. However, for curved surfaces, the geometry might not be the same, i.e.:

$$\vec{n_{p_1}} \; \neq \; \vec{n_{p_2}}\;, \tag{8}$$

Incorporating Eq (7) in Eq (6), we have:

$$
\begin{aligned}
log(M_{RG}) = \quad & log(m(\vec{n_{p_1}}\ \vec{l_{p_1}})) \ + \ log(e^{R_{p_1}}(\lambda)) \\
& + \ log(\rho^{R_{p_1}}(\lambda)) \ + \ log(m(\vec{n_{p_2}}\ \vec{l_{p_2}})) \\
& + \ log(e^{G_{p_2}}(\lambda)) \ + \ log(\rho^{G_{p_2}}(\lambda)) \\
& - \ log(m(\vec{n_{p_2}}\ \vec{l_{p_2}})) \ - \ log(e^{R_{p_2}}(\lambda)) \\
& - \ log(\rho^{R_{p_2}}(\lambda)) \ - \ log(m(\vec{n_{p_1}}\ \vec{l_{p_1}})) \\
& - \ log(e^{G_{p_1}}(\lambda)) \ - \ log(\rho^{G_{p_1}}(\lambda)) \ , \\
log(M_{RG}) = \quad & log(m(\vec{n_{p_1}}\ \vec{l_{p_1}})) \ + \ log(e^{R_{p_2}}(\lambda)) \\
& + \ log(\rho^{R_{p_1}}(\lambda)) \ + \ log(m(\vec{n_{p_2}}\ \vec{l_{p_2}})) \\
& + \ log(e^{G_{p_2}}(\lambda)) \ + \ log(\rho^{G_{p_2}}(\lambda)) \\
& - \ log(m(\vec{n_{p_2}}\ \vec{l_{p_2}})) \ - \ log(e^{R_{p_2}}(\lambda)) \\
& - \ log(\rho^{R_{p_2}}(\lambda)) \ - \ log(m(\vec{n_{p_1}}\ \vec{l_{p_1}})) \\
& - \ log(e^{G_{p_2}}(\lambda)) \ - \ log(\rho^{G_{p_1}}(\lambda)) \ ,
\end{aligned} \tag{9}
$$

We can factor out the terms $log(e^{R_{p_2}}(\lambda))$ and $log(e^{G_{p_2}}(\lambda))$:

$$
\begin{aligned}
log(M_{RG}) = \quad & log(m(\vec{n_{p_1}}\ \vec{l_{p_1}})) \ + \ log(\rho^{R_{p_1}}(\lambda)) \\
& + \ log(m(\vec{n_{p_2}}\ \vec{l_{p_2}})) \ + \ log(\rho^{G_{p_2}}(\lambda)) \\
& - \ log(m(\vec{n_{p_2}}\ \vec{l_{p_2}})) \ - \ log(\rho^{R_{p_2}}(\lambda)) \\
& - \ log(m(\vec{n_{p_1}}\ \vec{l_{p_1}})) \ - \ log(\rho^{G_{p_1}}(\lambda)) \ ,
\end{aligned} \tag{10}
$$

from Eq (10), the terms $log(m(\vec{n_{p_1}}\quad \vec{l_{p_1}}))$ and $log(m(\vec{n_{p_2}}\quad \vec{l_{p_2}}))$ are factored out, even though for a curved surface, $\vec{n_{p_1}} \neq \vec{n_{p_2}}$. Therefore, we have:

$$
\begin{aligned}
log(M_{RG}) = \quad & log(\rho^{R_{p_1}}(\lambda)) \ + \ log(\rho^{G_{p_2}}(\lambda)) \\
& - \ log(\rho^{R_{p_2}}(\lambda)) \ - \ log(\rho^{G_{p_1}}(\lambda)) \ .
\end{aligned} \tag{11}
$$

which is the illuminant invariant CCR descriptor for the R & G channels even with curved surfaces. The other channel R & B and G & B pairs can be similarly verified.

2. Network Configuration Details

Before we provide the details of the module's technical configuration, we describe the basic building blocks. The network consists of three major structures: an encoder block, an attention layer and a decoder block.

2.1. Basic Blocks

Encoder: The basic building block of the encoder consists of a 2D convolution layer, a Batch Normalization Layer and a ReLU layer, repeated twice. These 6 layers together create a single block. Fig. 1 visualises such a block.

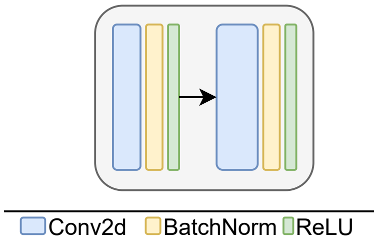

Figure 1. A basic encoder block

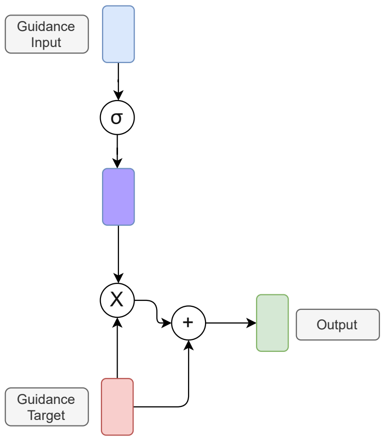

Figure 2. Basic configuration of an attention layer.

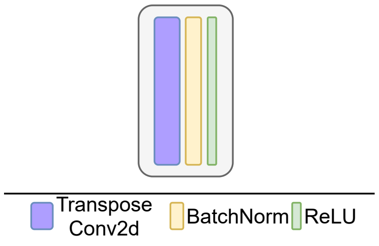

Figure 3. A basic decoder block.

Attention layer: The attention layer can be defined as:

$$
\begin{aligned}
\mathcal{F}_{int} &= \sigma(\mathcal{F}_i) \cdot \mathcal{F}_t \\
\mathcal{F}_{attn} &= \mathcal{F}_{int} + \mathcal{F}_t
\end{aligned} \tag{12}
$$

where, $\mathcal{F}_i$ is the guidance input to the layer, for example, the edge maps. $\mathcal{F}_t$ is target values that are to be guided through the attention, for example, the unrefined outputs. $\mathcal{F}_{int}$ is the intermediate output in the attention layer. And $\mathcal{F}_{attn}$ is the final attention guided output. A visual representation can be found in Fig. 2. All the operations are done element-wise.

Decoder: The basic building block of the decoder consists of a 2D Transposed Convolution layer, a Batch Normalization Layer and a ReLU layer. Fig. 3 visualises a single block.

Name	Layer	Kernel Size, Stride, Padding	Output Size
Input	conv1	3x3x64, 1, 1	256x256x64
	conv1	3x3x64, 1, 1	256x256x64
	conv2	3x3x64, 2, 1	128x128x64
	conv2	3x3x128, 1, 1	128x128x128
	conv3	3x3x128, 2, 1	64x64x128
	conv3	3x3x256, 1, 1	64x64x256
	conv4	3x3x256, 2, 1	32x32x256
	conv4	3x3x512, 1, 1	32x32x512
	conv5	3x3x512, 2, 1	16x16x512
Bottleneck	conv5	3x3x512, 1, 1	16x16x512

Table 1. Overview of our encoder configuration, used for both the image and ccr encoders.

Name	Layer	Kernel Size, Stride, Padding	Output Size
BottleNeck	deconv1	4x4x(512 * 2), 2, 1	32x32x512
	deconv2	4x4x(512 * 2 + 512 + 512), 2, 1	64x64x512
	deconv3	4x4x(512 * 2 + 256 + 256), 2, 1	128x128x256
	deconv4	4x4x(256 * 2 + 128 + 128), 2, 1	256x256x128
	conv6	3x3x(128 * 2 + 64 + 64), 1, 1	256x256x64
Output	conv6	3x3x3, 1, 1	256x256x3

Table 2. Overview of the configuration for the edge decoders. The summations represent the skip connections, while the product represents the interconnections between the decoders, i.e., reflectance edge and shading edge decoders.

Name	Layer	Kernel Size, Stride, Padding	Output Size
64x64 Output	conv1	3x3x512, 1, 1	64x64x3
128x128 Output	conv2	3x3x256, 1, 1	128x128x3

Table 3. Overview of our side output configuration, used for both the reflectance edge and shading edge decoder features.

Name	Layer	Kernel Size, Stride, Padding	Output Size
BottleNeck	deconv1	4x4x(512 * 1), 2, 1	32x32x512
	Attention	reflect edge (re) deconv1 & shading edge (se) deconv1	32x32x512
	deconv2	4x4x(512 * 2 + 512), 2, 1	64x64x512
	Attention	re deconv2 & se deconv2	64x64x512
	deconv3	4x4x(512 * 2 + 256), 2, 1	128x128x256
	Attention	re deconv3 & se deconv3	128x128x256
	deconv4	4x4x(256 * 2 + 128), 2, 1	256x256x128
	Attention	re deconv4 & se decovn4	256x256x128
	conv6	3x3x(128 * 2 + 64), 1, 1	256x256x64
	conv6	3x3x3, 1, 1	256x256x3
Output	Attention	re output & se output	256x256x3

Table 4. Overview of the configuration for the unrefined decoders. The summations represent the skip connections, while the product represents the interconnections between the decoders, i.e., unrefined reflectance and shading decoders. The attention layers take two inputs from the edge decoders.

In the following section, we list the details of each of the modules. We list the feed forward configuration in a tabular form. Each of the convolutions and transposed convolutions are always followed by a batch norm and ReLU layer. For brevity, the latter two layers are omitted from the tables.

2.1.1 Shared Image & CCR Encoder:

The configuration details for the image and CCR encoder is provided in table 1. The same configuration is used for both the image and CCR encoder.

2.1.2 Linked Edge Decoder

Table 2 lists the configuration for the edge decoder. The $*$ represents the incoming interconnection from the parallel decoder, while $+$ represents the incoming connections through skip connections. All the incoming features are concatenated depth wise before being passed onto the convolution blocks.

To add additional feature and scale space supervision to the edges, we also add a multi-scale supervision. For this we transform the intemediate features directly to an output image of size 64 & 128 and add a supervision of the corresponding scaled ground truths. Since we want to have an implicit supervision on the features themselves, we need to minimise the influence of the parameters of this transform. To enforce this, we propose to use the same "side output" convolution for both the reflectance edge and shading edges. This makes the convolution only learn a common transformation from feature space to image space. The configuration for side outputs are detailed in table 3.

2.1.3 Unrefined Decoder:

Table 4 shows the configuration for the unrefined decoder. The decoder takes the corresponding edge decoder outputs as the guidance. Skip connections from the image encoder is also provided for colour information. The input to the decoder is the bottleneck from the image encoder. The decoder outputs unrefined reflectance and unrefined shadings, which are globally consistent, but contains local artefacts and physical inconsistencies.

2.1.4 Local Refinement Module:

The configuration for our feature calibration layer is shown in table 5. Fig. 4 shows the refinement module overview. The Reflectance input is the concatenation of the unrefined reflectance (3 channels) and the reflectance edges (3 channels). Similarly, the Shading input is unrefined shading (1 channel) and shading edges (3 channels). We pass it through a $1x1$ convolution to obtain 16 channels each, allowing the network to select and expand the regions that needs most corrections. The transformation does not have any spatial dimension change.

The calibrated features are then fed into the refiner encoder to prepare for the refined decoder. The configuration is detailed in table 6.

Name	**Layer**	**Kernel Size, Stride, Padding**	**Output Size**
Reflectance Input	reflec conv1	1x1x(3 + 3), 1, 0	256x256x8
Reflectance Bottleneck	reflec conv1	1x1x16, 1, 0	256x256x16
Shading Input	shd conv1	1x1x(1 + 3), 1, 0	256x256x8
Shading Bottleneck	shd conv1	1x1x16, 1, 0	256x256x16

Table 5. Overview of the feature calibrator. It has two separate 1x1 convolutions for the reflectance and shading paths. The respective inputs are the concatenated unrefined reflectance & reflectance edge and unrefined shading & shading edges.

Name	**Layer**	**Kernel Size, Stride, Padding**	**Output Size**
Input	conv1	3x3x32, 1, 1	256x256x64
	conv1	3x3x64, 1, 1	256x256x64
	conv2	3x3x64, 2, 1	128x128x64
	conv2	3x3x64, 1, 1	128x128x128
	conv3	3x3x128, 2, 1	64x64x128
	conv3	3x3x128, 1, 1	64x64x256
	conv4	3x3x256, 2, 1	32x32x256
	conv4	3x3x256, 1, 1	32x32x512
	conv5	3x3x512, 2, 1	16x16x512
Bottleneck	conv5	3x3x512, 1, 1	16x16x512

Table 6. Overview of the refiner encoder. It takes the calibrated unrefined reflectance and shading as the inputs.

The bottleneck from the refiner encoder is then passed onto the local refinement decoder, as shown in table 7. The output of this decoder is the final IID outputs. The decoder gets skip connections from the refiner encoder. In addition, the skip connections from the first image and CCR encoders are passed through the attention layer before being added to the decoder. Both the image and CCR encoders are separately used as a guidance for the corresponding refiner encoders before being passed to the decoder. This enforces a local level correction, where the feature corresponding CCR and image features are used for the local unrefined outputs.

3. Loss Functions

To train the network we add explicit supervision to the outputs. For each type of losses, except for the edge, DSSIM and perceptual lossses, we use a combination of scale invariant MSE [4] and the standard MSE loss, as follows:

$$\mathcal{L}(I,\hat{I}) = \lambda_{smse} \times \mathcal{L}_{SMSE}(I,\hat{I}) + \lambda_{mse} \times \mathcal{L}_{MSE}(I,\hat{I}) \tag{13}$$

where, $\hat{I}$ is the ground-truth intrinsic image and I is the corresponding predicted image. We empirically set the λ_{SMSE} and λ_{MSE} to 0.95 and 0.05, respectively.

An overview of the losses can be seen in Fig. 5.

Name	**Layer**	**Kernel Size, Stride, Padding**	**Output Size**
BottleNeck	deconv1	4x4x512, 2, 1	32x32x512
	Attention	img enc conv4 & unref enc conv4	32x32x512
	Attention	ccr enc conv4 & unref enc conv4	32x32x512
	deconv2	4x4x(512 * 2 + 512 + 512 + 512), 2, 1	64x64x512
	Attention	img enc conv3 & unref enc conv3	64x64x512
	Attention	ccr enc conv4 & unref enc conv4	32x32x512
	deconv3	4x4x(512 * 2 + 256 + 256 + 256), 2, 1	128x128x256
	Attention	img enc conv2 & unref enc conv2	128x128x256
	Attention	ccr enc conv4 & unref enc conv4	32x32x512
	deconv4	4x4x(256 * 2 + 128 + 128 + 128), 2, 1	256x256x128
	Attention	img enc conv1 & unref enc conv1	256x256x128
	Attention	ccr enc conv4 & unref enc conv4	32x32x512
	conv6	3x3x(128 * 2 + 64 + 64 + 64), 1, 1	256x256x64
Output	conv6	3x3x3, 1, 1	256x256x3

Table 7. Overview of the configuration for the local refinement decoders. The summations represent the skip connections (from the unrefined encoder, the image encoder and the ccr encoder), while the product represents the interconnections between the decoders, i.e., the refined reflectance and shading decoders.

	Reflectance			Shading		
	MSE	LMSE	DSSIM	MSE	LMSE	DSSIM
w/o SSIM Loss	0.0020	0.0328	0.1766	0.0030	0.0847	0.2314
w/o Edge Loss	0.0018	0.0344	0.0846	0.0048	0.1128	0.1973
w/o Perceptual Loss	0.0019	0.0341	0.0902	0.0022	0.0500	**0.0791**
with All losses	**0.0015**	**0.0289**	**0.0688**	**0.0018**	**0.0489**	0.1005

Table 8. Ablation study on the various losses. It is shown that removing the losses degrades the performance across the metrics. The drop in performance for the DSSIM metric on the shading component is an acceptable trade-off since that loss improves the convergence of the network from 60 epochs to 15 epochs.

The DSSIM is derived from the Structural Similarity Index [6] (SSIM) as follows:

$$\mathcal{L}_{dssim}(I,\hat{I}) = \frac{1 - SSIM(I,\hat{I})}{2} \tag{14}$$

where, $\hat{I}$ is the ground-truth intrinsic image and I is the corresponding predicted image.

The reconstruction loss is as follows:

$$\mathcal{L}_{rec} = \mathcal{L}(R \odot S, RGB) \tag{15}$$

where R is the final reflectance output and S is the final shading output of the network. RGB is the input image to the network.

In order to show the influence of the various loss functions, we provide an ablation study on the losses in table 8.

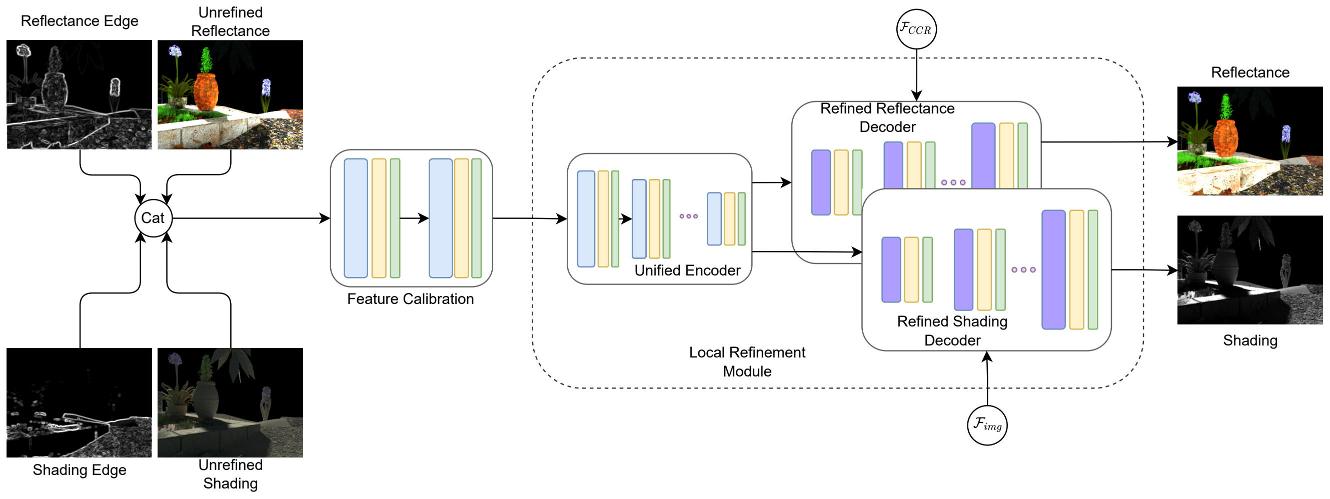

Figure 4. Overview of the local refinement module.

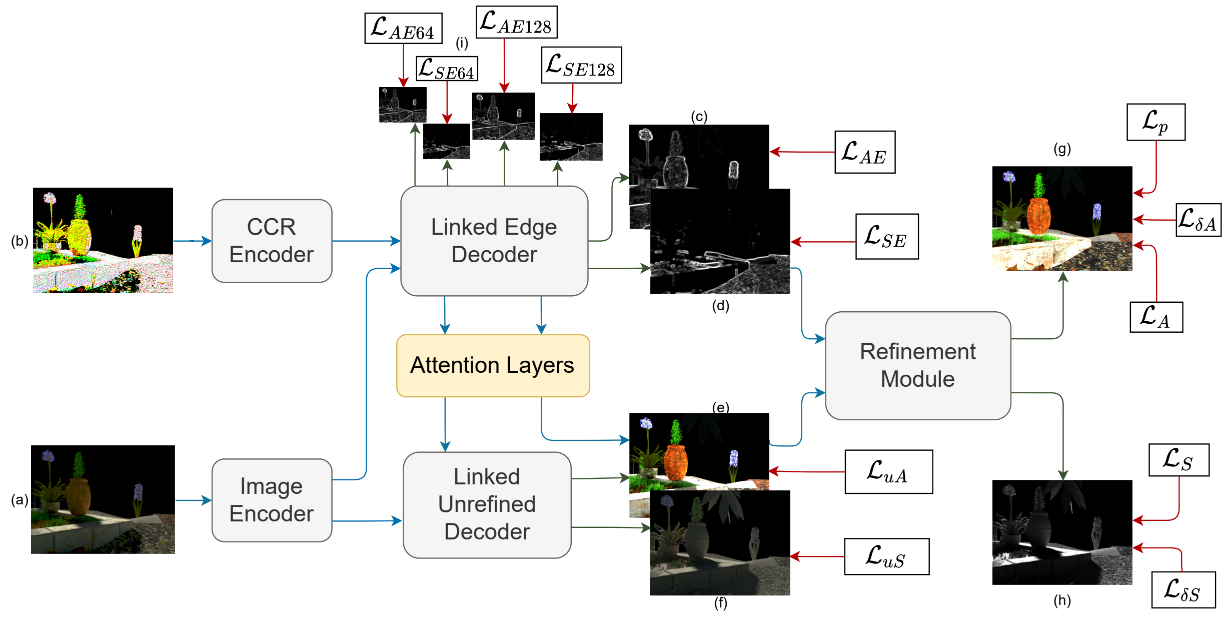

Figure 5. Overview of the supervision enforced on the network. The red arrows show the components on which the corresponding losses are applied. The blue arrows show the dataflow and the green arrows show the outputs of the network.

Here we test the performance of the network by removing the DSSIM, the scale space Edge and the Perceptual loss, respectively. We observe that removing the respective losses degrades the performance of the network across all metrics, apart from the Perceptual Loss on the shading DSSIM metric. This is expected since we add the perceptual loss for only the reflectance, since shading for our case is under a white light and hence grayscale. Perceptual loss helps us obtain a better colour as well as structural consistency. Due to the interconnected nature of the components in the network, the perceptual loss would act as a noise source to the shading. Hence, we see a small drop of performance in the structural metric for the shading only, while all the reflectance metrics show an improvement. However, even with the degradation, the performance is still better than the baselines algorithms from the literature, validating the network design choices. Additionally, removing the perceptual loss increases the convergence time from 15 epochs to about 60 epochs. Thus, we reason that this loss of performance on one metric is an acceptable trade-off.

4. Influence of the type of Attention

In this section we study the influence of the type of attention. The objective of the network is to learn a transformation of the input image into the intrinsic components. The attention layers are introduced for the network to be able to choose the incoming information. Hence, for our purpose there can be two types of attention: Channel-wise (allowing for cross channel attention, but on a more global contexts) or pixel-wise (spatial attention, allowing for a more finer-grained attention). For the channel wise attention we use the Squeeze and Excitation Network [1]. All the hyperparameters and losses are kept constant as our proposed full pipeline. A numerical comparison is shown in table 9.

	Reflectance			Shading		
	MSE	LMSE	DSSIM	MSE	LMSE	DSSIM
w/o Attention Layers	0.0019	0.0330	0.0776	0.0026	0.0704	0.1301
w Channel Attention	0.0019	0.0335	0.0758	0.0024	0.0638	0.1299
w Spatial Attention	**0.0015**	**0.0289**	**0.0688**	**0.0018**	**0.0489**	**0.1005**

Table 9. We present an ablation study on types of attention layers for our network. From the results, we can see that the spatial attention is better able to model the transformation function. This validates our hypothesis about the need for a finer-grained attention mechanism.

From the table we can see that the channel attention does not offer much improvement. In fact, the very small number of improvements are negligible, compared to without any attention layers. The reflectance component performance is almost like the configuration without any attention layers. Compare this to the spatial attention configuration, which we argue is because there is no single uniform transformation for all the pixels to arrive at the intrinsic components. Having a finer grained attention allows the network to learn a more flexible attentive transformation to arrive at a better decomposition. The numerical results, especially the local metrics (LMSE and DSSIM) that show the most significant improvements, validates this hypothesis.

5. Note on the performance on the Sintel Dataset

We observe that the network predictions for the shading is a bit darker than the ground truth. However, the outputs look structurally correct. This leads to the lower numbers on the shading MSE and LMSE metrics. Both the metrics measure the euclidean distance between the pixel values and are sensitive to outliers. DSSIM on the other hand considers spatial information and structures, in which we perform better. This is visualised in Fig. 6.

From the histogram, we can see that the distribution of the colours is similar, only varying on the scale of the value. From the standard evaluation metrics, the only metric that is sensitive to structures and spatial relationship, is DSSIM.

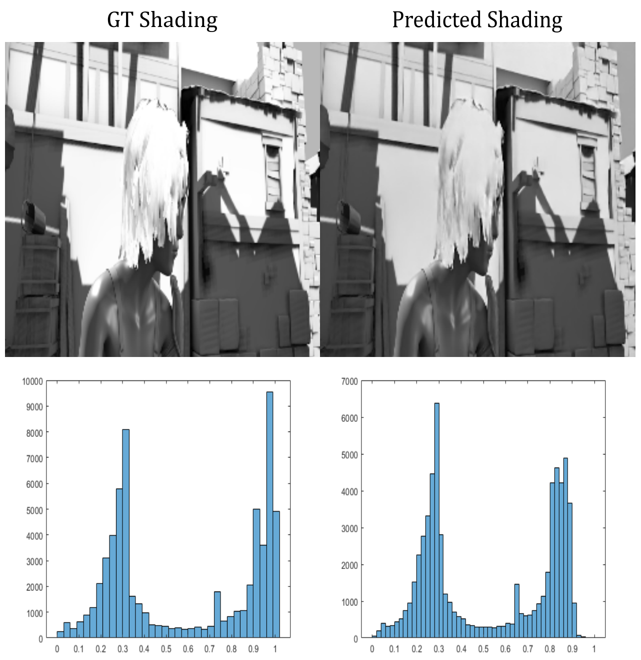

Figure 6. We show the histogram of the predicted shading and the ground truth shading image. It is shown that the predicted shading is of a different scale even though they are structurally similar.

As such, our outputs, even though colour wise and structurally correct, is not an exact match with the GT value pixel wise, resulting in the discrepancy.

Furthermore, our network, by nature is trained on scale invariant, perceptual and SSIM losses. These concentrates more on the perceived colour accuracy and structure than absolute pixel values. For the reflectance, it is not much of a problem since colours are ratios of the RGB channels. Hence, we can see that we are the best performing on the MSE metric and are the second best on the LMSE metric. For the latter metric, this can be explained by the local errors accumulating. However, in case of a grayscale shading (with our white light assumption), the shading is entirely defined as a scaling term. Hence all the scale mismatches amplify the problem, even though structurally the predictions are closer to the ground truth (as shown by the DSSIM metrics).

6. Extra Visualisations

6.1. NED

We provide visualisations for the NED test set in Figs. 7 and 8. It is shown that our network can disentangle cast shadows from the reflectance, while also preserving the proper reflectance colour smoothness and textures. The shading is similarly shown to only contain shadows and geometric details only, free from textures.

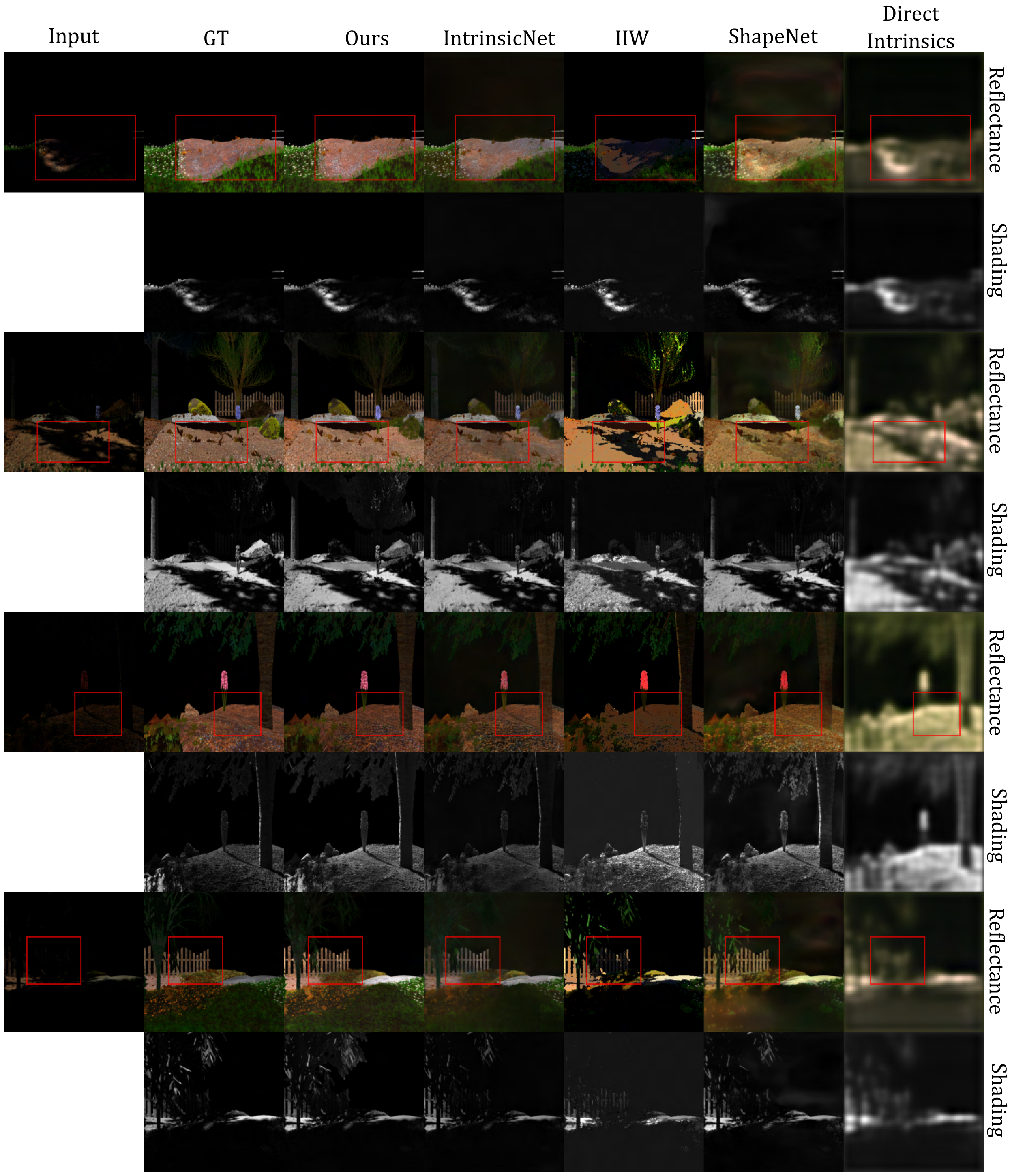

Figure 7. Visuals from the NED test set. It is shown that the network is robust enough to handle cast shadows. In rows 1, 3, 5 and 7, the heavy cast shadows on the ground are completely removed and the predictions are closer to the ground truth images. While IntrinsicNet can also remove the cast shadows, it leaves behind a discolouration in place of the shadows. The other algorithms fail at handling the cast shadows. Input images are gamma corrected for visualisation.

6.2. MPI Sintel

We visualise some of the predictions of our network from the test set. The results are visualised in the Figs. 9 and 10.

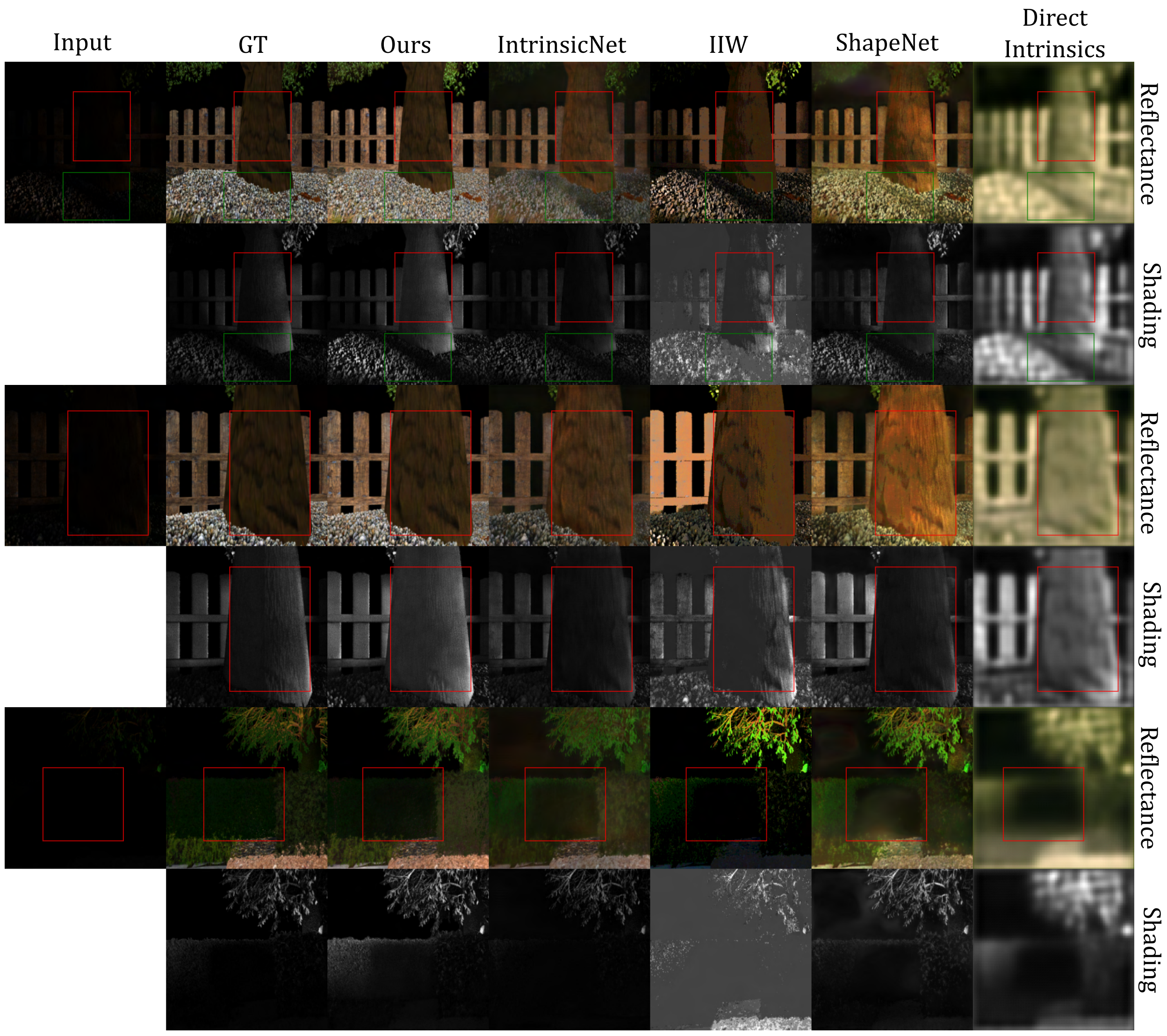

Figure 8. Additional visuals on the NED test set. It is shown that our network, in addition to handling the cast shadow problem, is also capable of preventing texture leakages. Rows 1 and 3 shows a tree trunk that has textures, which is preserved in our prediction, like the ground truth. The other baselines, however, transfer it to the shading images. Additionally, row 5 shows heavy discolouration on the bush, where there was a cast shadow, while our method can recover the reflectance with minimal discolouration. Input images are gamma corrected for visualisation.

It is shown that the predicted outputs are close to the ground truths, robustly handling the cast shadows and textures in the shading and reflectance respectively.

6.3. MIT Intrinsics

We present some visualisations of the predictions compared with various baselines on the MIT Intrinsic test set. The visuals are provided in Figs. 11 and 12. It is shown in the figure that our predictions are much closer to the ground truth compared to the other baselines. IntrinsicNet is shown to have comparatively worse performance, missing shadows (on the raccoon) and often transferring textures in the shadings. In comparison, our method is robust against both leakages, showing the effectiveness of the physics-based guidance.

6.4. IIW

We present visualisations from the IIW test set in Figs. 13 and 14. It is shown that our network can predict reflectances that are consistent with the flatness assumptions. The structural details are correctly transferred to the shadings. CGIntrinsics [3] and [2] on the other hand, misses the reflectance reflectance boundaries. They are also shown to miss finer details like tiles, while also often generating a

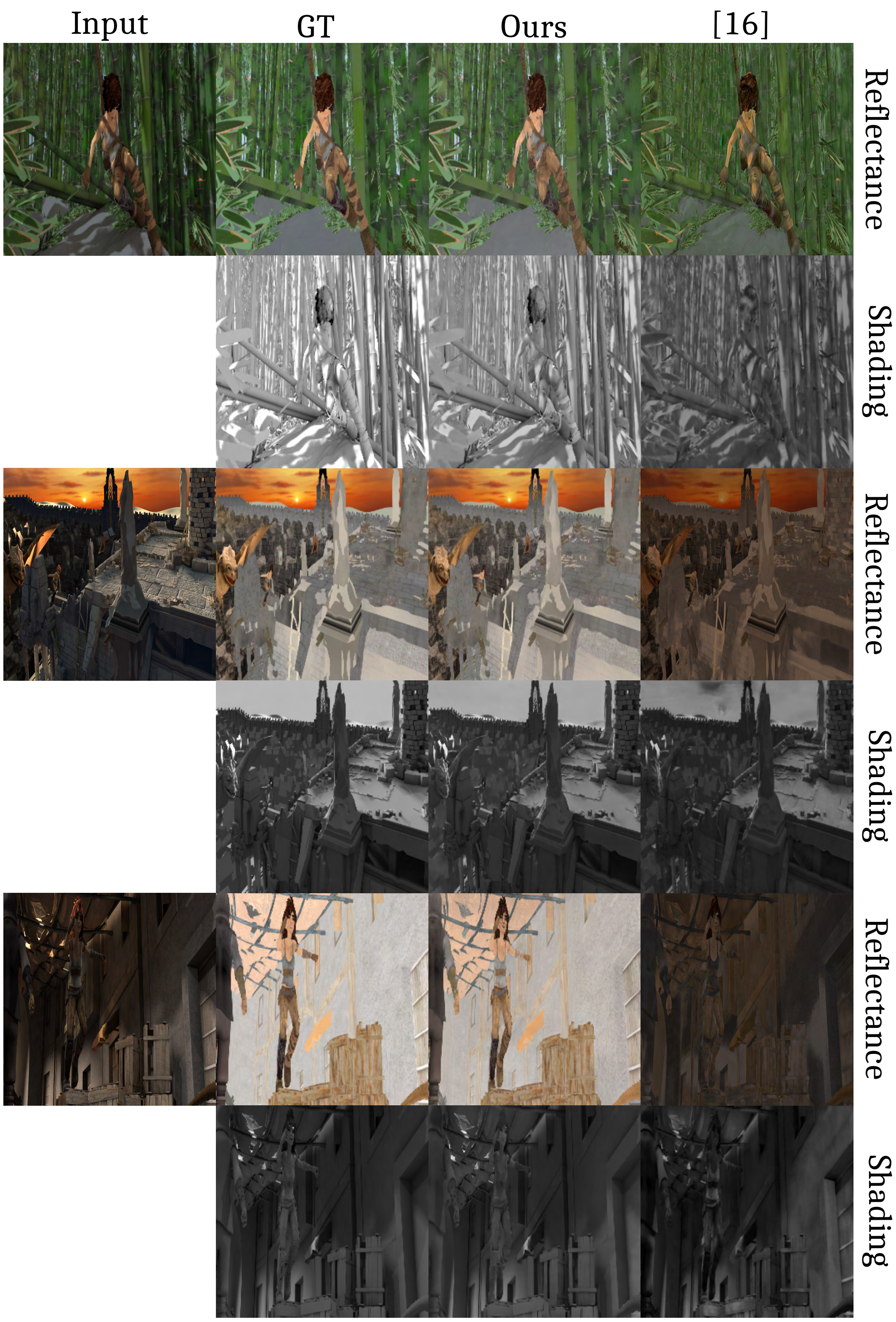

Figure 9. We show visuals on the MPI Sintel test set. It is shown that our network can remove the cast shadows, even from complex scenes like a forest (row 1). On row 5 it is shown that the reflectance is free from the cast shadows on the wall, while the shading image is free from the textures on the wooden box.

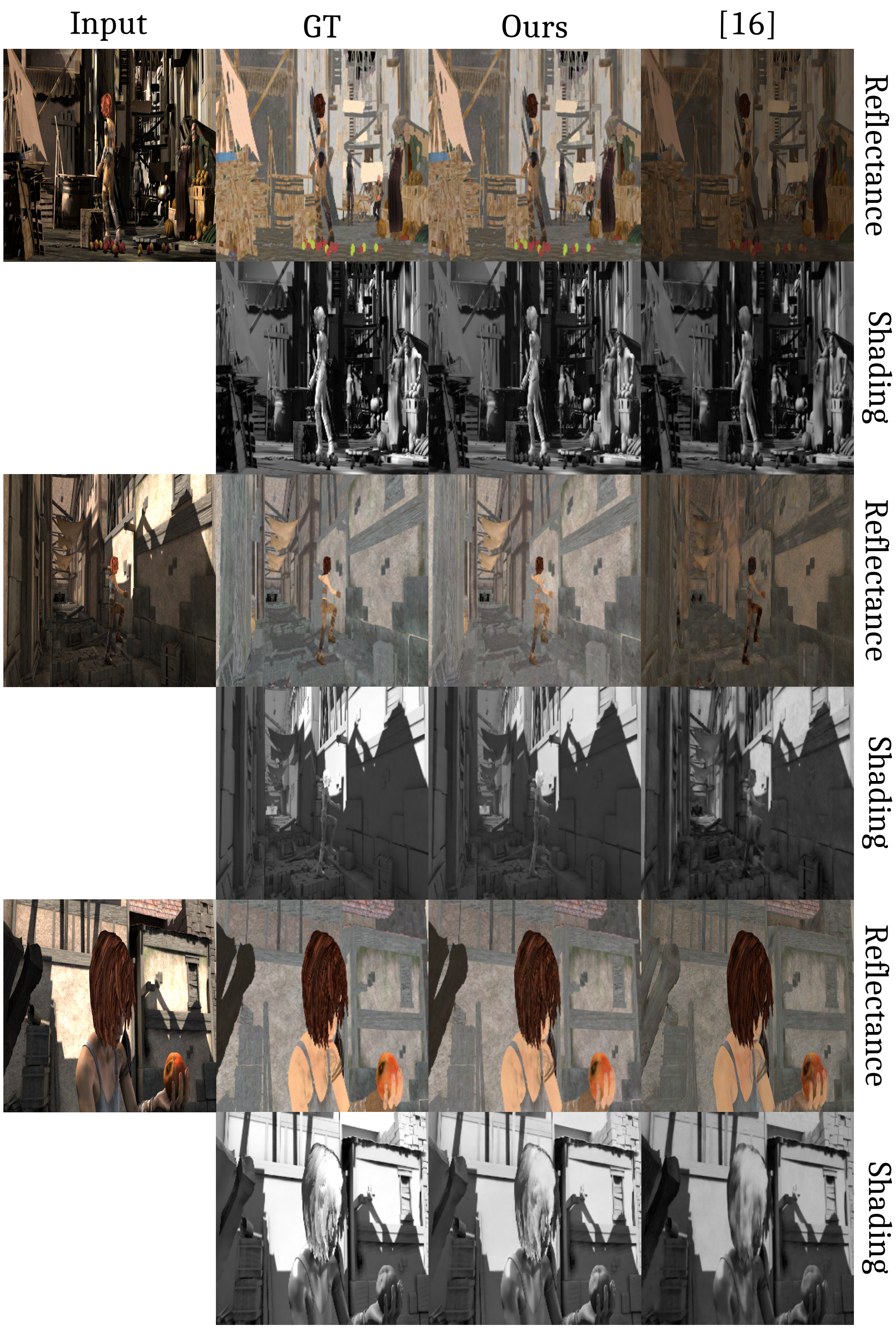

Figure 10. Additional visualisations on the MPI Sintel test set. It is shown that our network is able to handle complex scenes with complicated object interactions and cast shadows.

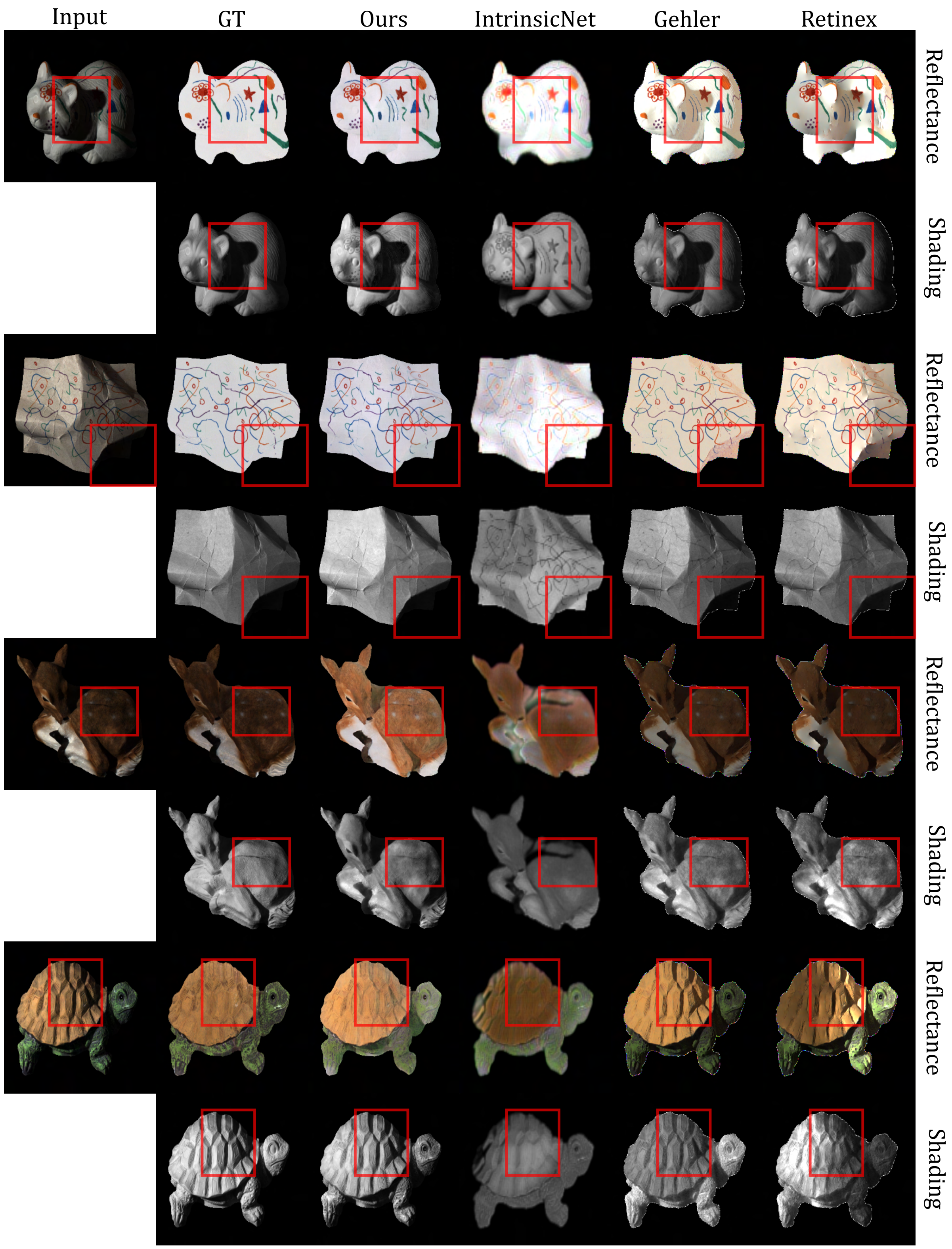

Figure 11. Visuals from the MIT Intrinsic test set. It is shown that the proposed algorithm predictions are closer to the ground truth IID components. IntrinsicNet, on the other hand, completely misses the shadow on the racoon and the paper (rows 2 & 4), while the proposed algorithm can transfer it to the shading image correctly. The deer and turtle (rows 5 & 7) show the proposed algorithm able to properly disentangle geometric patterns from reflectance, which are much flatter.

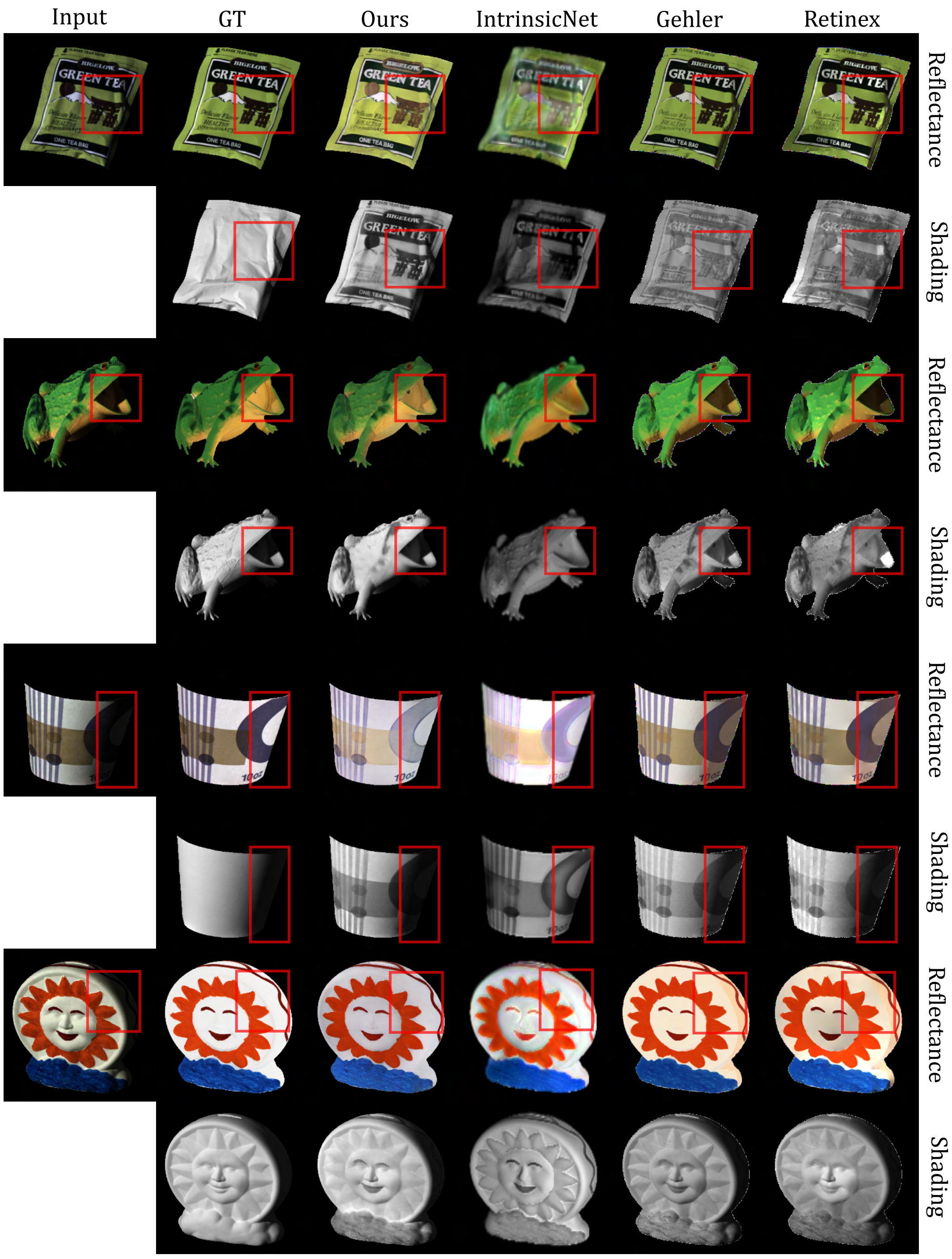

Figure 12. Additional visualisations for the MIT Intrinsic test set. The shadow from the inside of the frog's mouth (row 3) is removed. IntrinsicNet completely misses that (row 4), while other baselines predict it as part of the reflectance. For the cups and sun statue (rows 5 & 6 and 7 & 8) the edges with shadow influence (red box) is much flatter for the proposed algorithm, while the baselines have noticeable artefacts due to shadow and reflectance changes.

blurrier shading. The proposed method is able to preserve the tiles and also predict a sharper details, even though it was not trained in the same domain exclusively.

6.5. Trimbot

To test the effectiveness of our network on real world scenes, we provide outputs on the Trimbot dataset, which

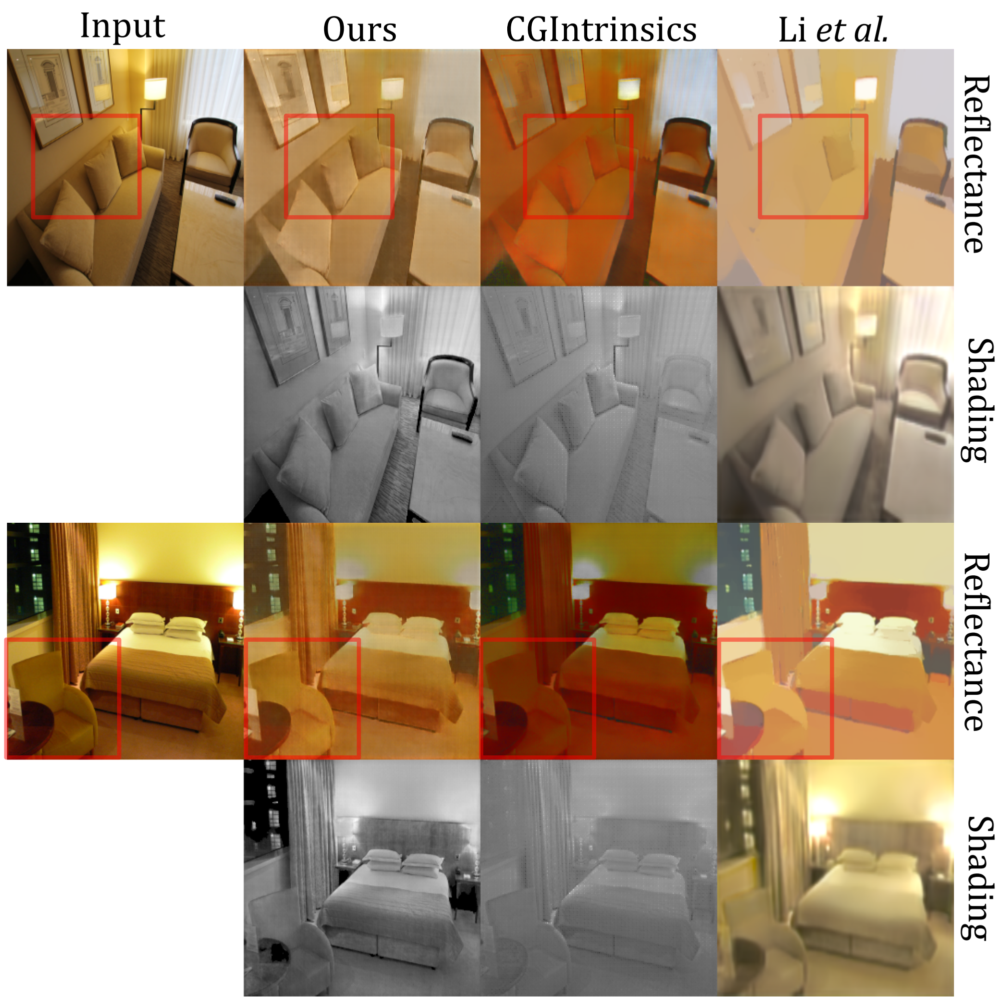

Figure 13. Visuals from the IIW test set. It is shown that our network can predict comparatively better reflectance, which is able to distinguish reflectance boundaries. The proposed network is able to separate the wall from the sofa (first row). Similarly, the competing method classifies the seat of the chair (third row) differently, even though it is part of the same reflectance. The fireplace mantle (fifth row), should be uniform, but the competing methods have uneven reflectance, while our method is able to predict comparatively flatter and uniform reflectance.

are relatively close to our synthetic data settings. The visuals are provided in the Fig. 15. It is shown in the figure that our network can distinguish not only between the shadows and reflectance, but also between the finer objects and shadings. For example, in our outputs, the ground in the shading is flat owing to the flat geometry, while the reflectance shows the reflectance boundary between the grass, thus giving it a rough texture.

6.6. Real world images

To further test our method, we take a few random internet images. We show the outputs in Fig. 16. It is shown that the network can recover the reflectance, even though it was only trained/fine-tuned on synthetic images. The shadow of the tree on the ground in the first image is separated properly into the reflectance. While in the shading image, the ground is smooth and flat, since the texture is from the reflectance side, while geometrically, the ground is flat and free from variations. Small shadows in the treetops are removed in the reflectance and only preserved in the shading image. Hence, it is shown that the network does not just learn a grayscale transformation for the shading, but also a physics guided IID.

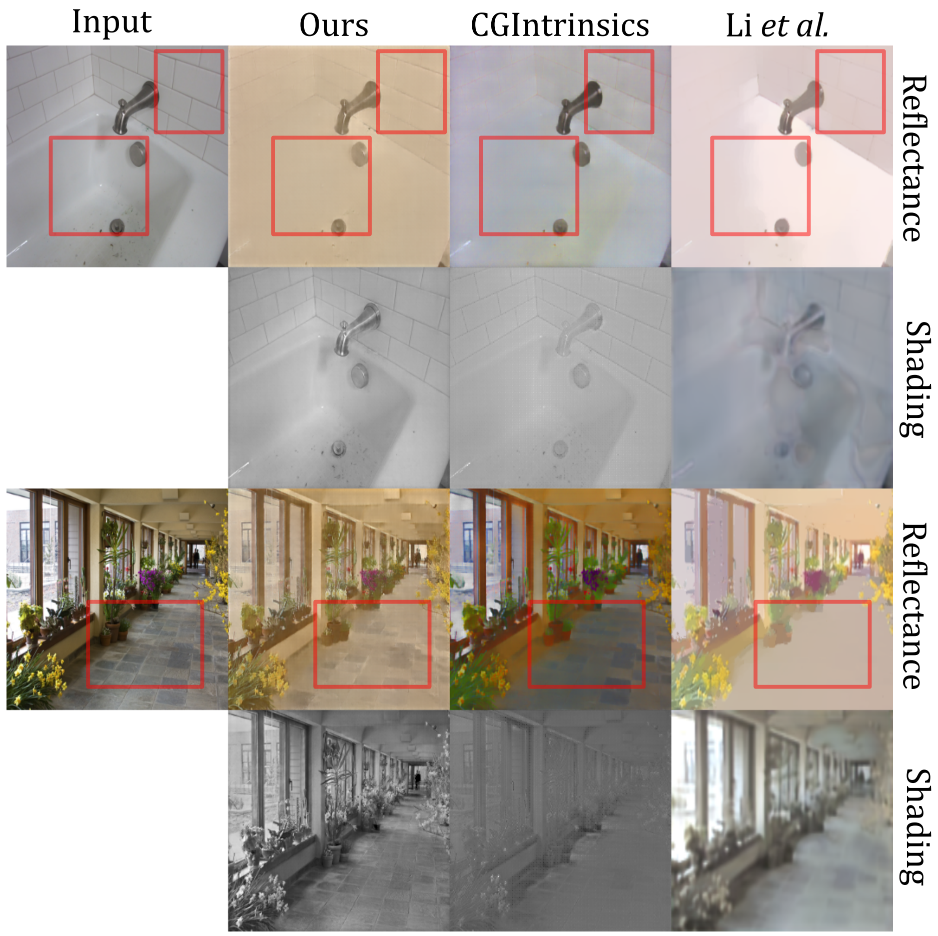

Figure 14. Additional visuals for the IIW test set. It is shown that the predicted network is able to preserve finer details like the individual tiles on the bathroom walls (first row) and floor (third row), while predicting a sharper shading image too. The competing method often predicts a flatter reflectance for all of it, with a blurrier shading image.

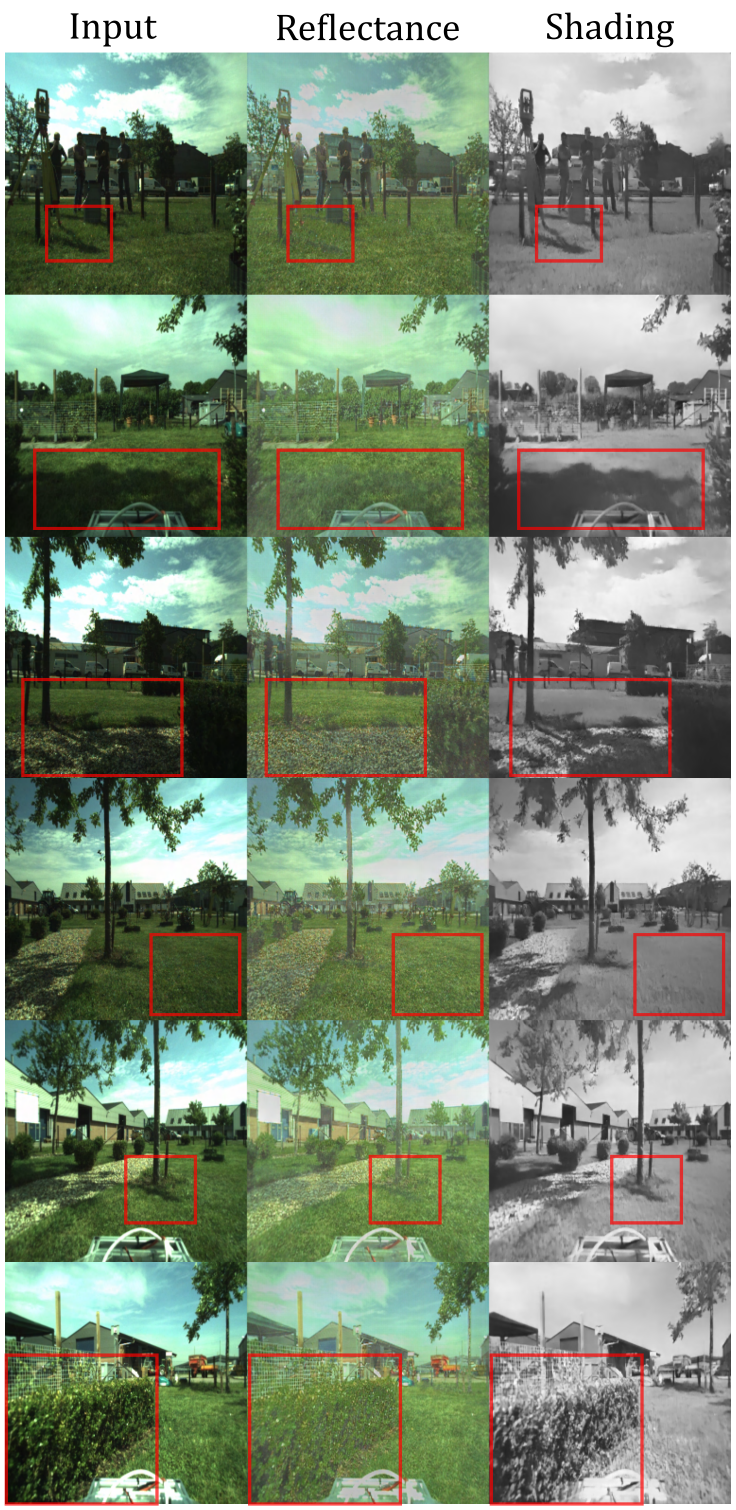

Figure 15. Visuals on the Trimbot dataset. It is shown the proposed algorithm can remove the influence of the cast shadows from the reflectance. In the shading image (row 4), it is shown that the ground is flat and free from textures, due to the flatter geometry of the ground. On the last row, it is shown that the reflectance of the bush also lacks the finer shadows, which show up only in the shading image.

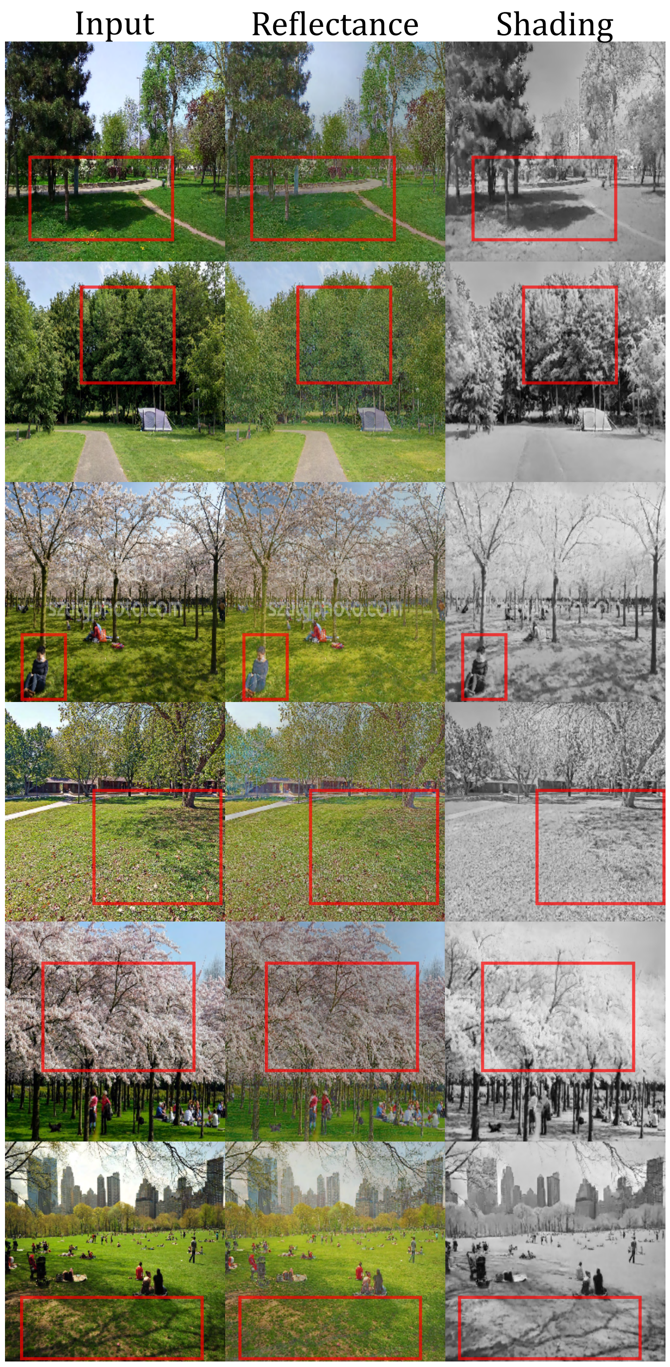

Figure 16. It is shown that the proposed algorithm can model a physics-based formation model, even though it is trained and fine-tuned on purely synthetic data. The shadows on the ground and even among the leaves are removed (rows 1 and 2), while the shading is flat and smooth on the ground, while preserving the small shadows and geometry in the treetops. On the 3rd row, the shadows are mostly removed, while structural details on the woman's shirt is completely flattened in the reflectance.

References

[1] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 6

[2] Zhengqin Li, Mohammad Shafiei, R. Ramamoorthi, Kalyan Sunkavalli, and Manmohan Chandraker. Inverse rendering for complex indoor scenes: Shape, spatially-varying lighting and svbrdf from a single image. *CVPR*, pages 2472–2481, 2020. 8

[3] Z. Li and N. Snavely. Cgintrinsics: Better intrinsic image decomposition through physically-based rendering. In *ECCV*, 2018. 8

[4] T. Narihira, M. Maire, and S. X. Yu. Direct intrinsics: Learning albedo-shading decomposition by convolutional regression. In *ICCV*, 2015. 4

[5] S. Shafer. Using color to separate reflection components. *Color Research and App.*, pages 210–218, 1985. 1

[6] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE TIP*, pages 600–612, 2004. 4