Errata Sheet to "Physics-based Vision Meets Deep Learning"

yy1571

March 2021

The section 2.1.1 is very high level. What is in OpenDR? What can it do? How is minimisation of an appearance error different from training?

Correction: The paragraph of section 2.1.1 should be rewritten as shown below:

In energy minimisation approaches to inverse graphics problems, there is a need for differentiable renderers that model the physical process of image intensity formation. This allows minimisation of an appearance error using gradient descent or other first-order optimisation algorithms. Loper *et al.* [13] developed a differentiable renderer package called OpenDR as a framework. OpenDR could reproduce the observed images with latent variables like appearance, geometry and camera, then the error between the reproduced image intensity map and the observation can be minimised. This differentiable renderer allows us to perform image rendering with respect to per-vertex brightness, vertex-based geometry and pinhole-plus-distortion camera, and compute partial derivatives of the rendered image with respect to these image formation parameters. Hence we can estimate these image formation parameters from an observed image through an analysis-by-synthesis optimisation, which is achieved by minimising the error between the observed image and the reproduced image rendered from the estimated model parameters. Minimising the error of the rendered image from this differentiable renderer facilitates the model parameter estimation from an input image, or it could be integrated into an inverse graphics neural network to train the network parameters. Zienkiewicz *et al.* [21] applied a similar idea for estimating height maps in a real-time robotic system. The rendering objective isheight map rather than RGB intensities in their case.

Details of the cited papers in section 2.1.2 are not provided. Cited works from Jaderberg *et al.* [9] and Hinton *et al.* [8] are not related to the topic of this section because their networks do not have a render layer.

Correction: The contents related to [9] and [8] are removed from this section, and the context is reordered to better explain the cited papers. The new section 2.1.2 should be:

Some work has investigated the idea of a trainable renderer. Dosovitskiy *et al.* [3] proposed a neural network for image reconstruction from extracted feature maps resulting from different hand-crafted or learned descriptors like HOG, SIFT or trained deterministic CNN. They trained an up-convolutional network against the loss function comparing original inputs and reconstruction. Similarly, using differentiable rendering layers inside a network to perform image reconstruction were implemented by [4, 5, 6, 7, 8]. Gregor *et al.* [6] employed an recurrent architecture in their DRAW net that learns to generate images in an iterative manner. They applied attention modules to make their network only focus on part of the image canvas at each step, and hence accumulating and gathering the generated pieces in the final results. In each step, the encoder generates latent codes conditioned on the input image and previous outputs from the decoder, which is combined with the attention model to achieve impressive performance. Gatys *et al.* [4] tackled the problem of auto-generating images with respect to the specified artistic style. The fundamental insight supporting the method is that the representations of content and style are independently captured by hierarchically different sub-networks inside CNNs. Thus the target image could be generated by minimising the content matching lossand style matching loss derived from a pre-trained VGG net [19]. Dosovitskiy *et al.* [3] developed a model to render images given scenes description parameters, like camera position and appearance style. But the effectiveness of the generator is only validated by synthetic laboratory images. Zhmoginov and Sandler [7] proposed a face image rendering network being able to construct face image from a pair of guiding image and identity embedding vector. Similar to Gatys *et al.* [4], Zhmoginov and Sandler [20] formulated network training losses by measuring the feature distance computed by the pre-trained FaceNet, which is proposed by Schroff *et al.* [18]. Nalbach *et al.* [15] explored the graphical shading problem by using rendering layers. Specifically, their Deep Shadingnet explicitly takes as input the meaningful scenery attributes defined in common shaders and use well-trained neural network to act as a self-taught shader as opposite to manually designed shader.The rendering networks introduced by these works are generative models simulating the probabilistic process of RGB image synthesis. In contrast, our BRDF estimation network models a discriminative process on the statistical BRDF parameter estimation problem and uses a fixed rendering architecture to formulate the loss function.

Goodfellow *et al.* [5] proposed the idea of Generative Adversarial Nets (GANs), which trains a generative model together with a discriminator. The discriminator is trained to classify real and generated images, and the generator is trained to fool the discriminator. This adversarial training mechanism encourages the generative model to generate realistic images capturing the hidden features in the input images. Radford *et al.* [17] applied adversarial training with convolutional neural networks in their DCGANs. By employing CNN-based generative and discriminative models, their generative model could be trained in an unsu-

pervised fashion to generate realistic images of different domains. The trained generator in GANs, however, can only take input of random noise vectors but cannot perform rendering based on semantic inputs.

The definition of SVBRDF is not given in the first paragraph on page 17: Instead of only estimating uniform BRDF, their networks tackles a more challenging problem, which decomposes a single input image to SVBRDF, normal and illumination.

Correction: Instead of only estimating uniform BRDF, their network tackles a more challenging problem, which decomposes a single input image to spatially-varying Bidirectional Reflectance Distribution functions (SVBRDF) [16], normal and illumination.

In Section 3.1 on page 26, the lighting parameters are not input and not fixed. The lighting fits to the data and hence surely it is a trainable parameter.

Correction: There is a potential confusion here over the statement that the lighting parameters are not trainable parameters. Since they are neither input nor fixed, it may appear that they must be trainable parameters but this is not the case. A trainable parameter means a parameter whose value affects the output of the network and whose value is found by minimising training loss averaged over the training dataset. To clarify the lighting parameter is not like a trainable parameter, the additional explanations should be included after the sentence, "Note that this is an untrainable layer, because the Lambertian layer contains no trainable parameters and is not updated to reduce the loss during the training process." on the line 9 of page 26:

As for inferred lighting parameter from the Lambertian layer, they are neither input nor fixed, it may appear that they must be trainable parameters but this is not the case. A trainable parameter means a parameter whose value affects the output of the network and whose value is found by minimising training loss averaged over the training dataset. The lighting parameters act like latent variables. They are themselves the solution to an optimisation problem whose inputs are the outputs of the network. The network does not output lighting parameters. It outputs other quantities from which the lighting parameters can be derived by solving an optimisation problem. During training, this is done in-network such that losses that depend on lighting can be computed and the loss backpropagated through the solution for lighting and back into the network. This all becomes explicit in Chapter 4 when such an architecture is used in InverseRenderNet.

In Equation 3.2 on page 28, $\mathbf{G}_x$ and $\mathbf{D}_x$ should be $\mathbf{G}_y$ and $\mathbf{D}_y$.

Correction:

$$\mathbf{G}_y = \mathbf{D}_y * \mathbf{Z}. \tag{1}$$

The sentence under Equation 3.5 on page 29 is unclear for describing the numbers of u and $m \times n$.

Correction: A clearer description should included after the sentence "This function is applied to the gradient estimate at each pixel, yielding u surface normal vectors, where u is length of flatten vector from $m \times n$ map.":

For clarity, $u = mn$ where u is the number of pixels, m the height and n the width of the image.

The notions used in Equation 3.5 and 3.6 on page 29 are inconsistent with the other equations, and n should be bold as it is a vector.

Correction: In Equation 3.5, replace $\bar{\mathbf{n}}$ with $\mathbf{n}$ and in following sentence. Equation 3.6 and sentence above, replace n with $\bar{\mathbf{n}}$ and in Equations 3.6, 3.7, 3.8, 3.9, 3.11 replace $n(\bar{\mathbf{n}})$ with $\bar{\mathbf{n}}(\mathbf{n}(\mathbf{g}))$.

Wrong sentence on the 4th line in the paragraph under Equation 3.25 on page 37: Finally, cameras apply a nonlinear gamma transformation to the rendered images out of this renderer.

Correction: Finally, we apply a nonlinear gamma transformation to the rendered images out of this renderer so as to model the gamma correction process.

The last two sentences of the paragraph under Equation 3.32 on page 39 are not clear and accurate enough: The training will eventually converge as long as we start the training from a reasonable initialisation, which is realised by our phased training scheme. We will discuss this in details in next chapter (see Section 4.5.2).

Correction: To make this training eventually converge, we combine supervisions derived from MVS scene reconstructions and a phased training scheme, which are described in detail in the next chapter. We empirically found alternating between this simple lighting optimisation and the gradient descent optimisation for training an inverse rendering neural network allows us to achieve the training convergence.

$\ell_{\mathbf{reg}}$ is not defined on page 42.

Correction: A new equation and the associated description should be included between the sentence "... regularisation term ℓ_{reg} is another part of our loss function:" and "The coefficient of regularisation is ..."

$$\ell_{\text{reg}} = \sum_{s=1}^{S} \|w_s\|_{\text{fro}}^2, \tag{2}$$

where w_s is the sth learnt feature kernel inside the neural network, and S is the total number of kernels. This regularisation loss penalises the overall magnitude of the values inside the learnt convolutional kernels, which measures the com-

plexity of the learnt neural network model. Minimising this loss could restrict the complexity of the network, hence effectively addressing the overfitting issue.

The sub-section "Convolutional Neural Network" on page 43 does not clearly describe the network architecture, and Figure 3.7 is not clear for demonstrating the network.

Correction: The full context in this section and Figure 3.7 should updated:

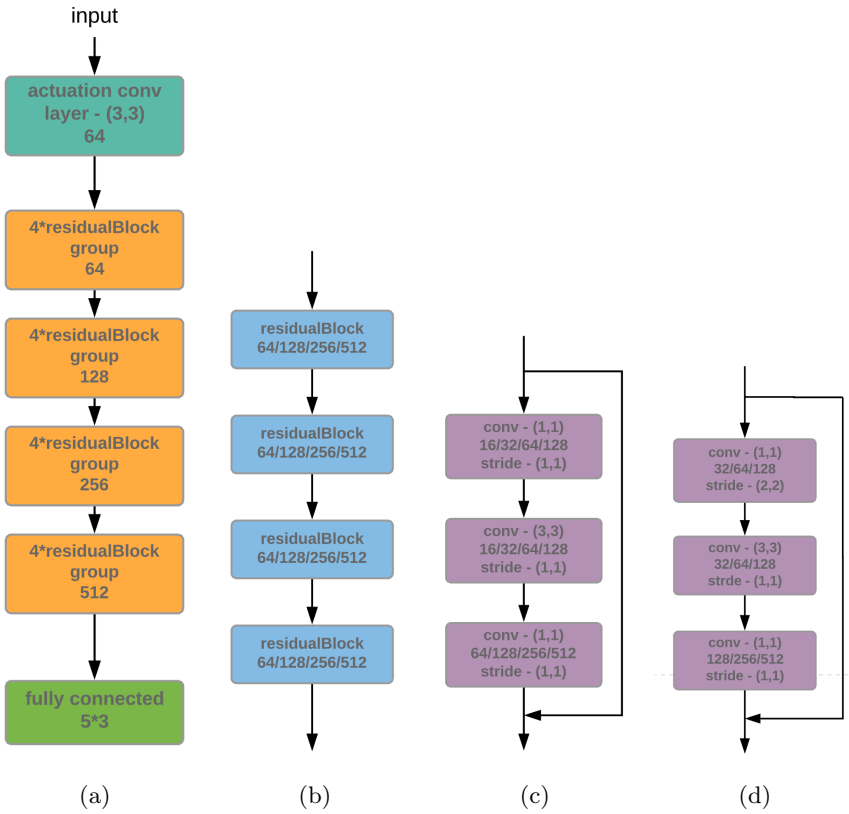

Figure 1: Network architecture. (a) Overall architecture; (b) Residual block group, shown as orange boxes placed in the middle of (a); (c) Convolutional layers inside residual block; (d) Convolutional layers inside the first residual block of each group, where the spatial resolution is reduced by the first strided convolutional layer, and the channel of output feature map is increased by the third convolutional layer.

The architecture of our ConvNet is realised by a stack of residual blocks consisting of a simplified encoder-decoder architecture and a residual shortcut introduced by He *et al.* [7]. Since our problem is to estimate BRDF parameters from only one single image, a deep network with powerful learning ability is

required. We construct our network with 50 layers following a ResNet layout. Also, following the downscale rule proposed by Simonyan *et al.* [19] in their VGG net, our network performs pooling operations followed by dimension increasing on the feature map. The visualisation of our network is shown in Figure 1. Our neural network is constituted by one actuation convolutional layer, 4 groups of residual blocks and one final fully-connected layer. Each group contains 4 residual blocks, and each residual block consists of 3 convolutional layers and one skip connection. So there are totally 48 convolutional layers within the 4 residual blocks groups. The actuation convolutional layer in our network is placed next to the input layer, converting the RGB input images to the initial feature maps with 64 channels. The following residual blocks are grouped according to the number of channels of the feature maps, which are 64, 128, 256 and 512. The 4 residual block groups are shown as orange boxes in 1a, and the internal architecture of residual blocks is shown in 1b. Note that, for each residual block and group box, the numbers written at the bottom indicate the channel dimension of the feature maps. Except for the first residual block group, the pooling and feature channel increment operations are acted by the first residual block in each group. For simplicity the pooling is done by using stride of size $(2, 2)$, whose efficiency has been proven by He *et al.* [7]. The visualisation of the residual block performing this pooling and feature channel expansion is shown in Figure 1d. Within this residual block, the first layer performs strided convolutional to reduce the spatial resolution, and the last layer increases the number of channels of the convolutional kernel to expand the dimension of the output feature map. Note that the first residual block group does not perform the operations of pooling and feature channel expansion, because the actuation convolutional layer has already transferred the 3D RGB image to the 64D feature map. The architecture of the other residual blocks can be found in Figure 1c. Inside each convolutional layer box visualised in Figure 1c and 1d, the numbers on first line stand for the size of the convolutional kernel, the numbers of the second line indicate the dimension of the output feature map, and the numbers on the last line are the step size of the stride. The fully-connected layer at the end of the net maps output from convolutional layer to 15-D vectors, which represents three 5 dimensional BRDF statistical model parameters for each colour channel respectively.

Equation 4.3 on page 54 is not an explanation for the case of $n_z = 0$.

Correction: Change the sentence prior to the Equation 4.3 from "... pixels, $n_z \geq 0$, we can compute ..." to:

Since for all visible pixels, $n_z > 0$, we can compute the surface normal direction from the estimated quantities as.

In the first sentence of the first paragraph under Section 4.4 on page 55, "L2 data term" should be referred to Equation 4.5.

Correction: Change the first sentence in the paragraph under Section 4.4 on

page 55 from "As shown in Figure 4.2, we use a L2 data term (the error between predicted and observed appearance) for self-supervision." to:

As shown in Figure 4.2, we use a L2 data term (the error between predicted and observed appearance) for self-supervision, which is described by Equation 4.5 in Section 4.4.1.

The original data of the "79 HDR spherical panoramic images", which is described on the 4th line of the paragraph under Figure 4.6 on page 58, is no longer available on the referred website.

Correction: We packed and uploaded the source data online. In order to refer to the link, an additional sentence should be inserted between the sentences of "... 79 HDR spherical panoramic images taken outdoors [10, 1]." and "As shown in Figure 4.4 ...":

The URL for the original data is no longer available. We now provide the source HDR images at:
`https://docs.google.com/uc?export=download&id=1yt4c6DriDPN26HRCVW4D1t8z-M6J43Nk`.

Why $\arccos(\mathbf{n_{guide}} \cdot \mathbf{n_{est}})$ **is used rather than** $\mathbf{n_{guide}} \cdot \mathbf{n_{est}}$ **in Equation 4.9 on page 61?**

There is no correction to make here. $\arccos(\mathbf{n}_{\text{guide}} \cdot \mathbf{n}_{\text{est}})$ is the angle between $\mathbf{n}_{\text{guide}}$ and $\mathbf{n}_{\text{est}}$ and we seek to minimise this. We could alternately maximise the dot product $\mathbf{n}_{\text{guide}} \cdot \mathbf{n}_{\text{est}}$ but the angular error is more natural and there is no visible difference between the two in practice.

"... correlation coefficient of intensity histograms close to 1." in the first sentence under Figure 4.9 on page 65, is a vague description, and the exact range of the correlation coefficient should be given.

Correction: Change the sentence from "... defined as having correlation coefficient of intensity histogram close to 1" to:

... defined as having correlation coefficient of intensity histograms greater than 0.95.

"$\gamma = 2.2$**" is directly used for nonlinear gamma transformation on the 7th line of the paragraph under Section 4.7.2 on page 70, but there is not reference to explain why this gamma transformation factor has been used.**

2.2 is the standard decoding gamma for the sRGB colour space. It is universally used, including in other computer vision works that account for nonlinear gamma, e.g. [11, 12, 14].

In Section 4.8.5, the ground truth is projected to the order 2 SH approximation of the light map. Explain the fairness of this choice

and try to provide the errors of reproducing the light map.

Correction: First, the original sentence on the 6th line of the paragraph under Table 4.7, "Since our network can only infer the lighting represented by spherical harmonics, we project the ground truth environment map onto order 2 spherical harmonics.", should be updated as:

Since our network and the employed comparison methods (SIRFS [2] and InverseRenderNet) can only infer the lighting represented by spherical harmonics, we project the ground truth environment map onto order 2 spherical harmonics.

Second, Table 4.7 could be extended with errors of reproducing lighting maps, such that the new Table should be:

Methods	Global scale		Per-colour scale	
	Reconstruction	SH	Reconstruction	SH
SIRFS [2]	0.026	0.100	0.023	0.089
InverseRenderNet	0.024	0.050	0.021	0.041
InverseRenderNet++	**0.023**	**0.038**	**0.019**	**0.033**

Table 1: Quantitative results for illumination estimation. We show global-scale and per-colour-scale MSE errors for lighting map reconstruction (2nd and 4th Columns) and relit sphere from order 2 spherical harmonics (3rd and 5th Columns).

This new quantitative evaluation table contains evaluations of the lighting map reconstruction rather than the 2nd order SH approximation. To explain this, the following sentence should be added after the sentence "... spherical harmonic lighting." on the 10th line of the paragraph under Table 4.7 on page 84:

In addition, we also provide the quantitative evaluations of the error between the front side of the lighting map reconstructed from the lighting predictions and of the ground truth lighting map.

The details of the scaling operation and interpretation of errors in Table 4.7 are not provided in Section 4.8.5.

Correction: The extra explanations should be added into the second last line of the paragraph under Table 4.7. Specifically, the context shown below needs to be inserted after the sentence "... scaling to each colour channel." on the second last line of the paragraph:

The global scaling factors adjust the intensity of the lighting predictions to fit to the ground truth and so remove the scale ambiguity, and the pre-colour scalings adjust the intensity of the predictions for each of the colour channels independently, hence excluding the affections from both scale and colour ambiguities. The global is computed by the ratio between the global median of prediction and ground truth, and the per-colour scaling factor is obtained by comparing the per-colour median of prediction and ground truth. The Global-scale and per-colour-scale MSE errors reported in Table 4.7 are compared with

the ground truth lighting map being scaled within the range [0, 1], and the relit map being scaled to the range [0, 1]. The InverseRenderNet++ achieves the best performance for both global-scale and per-colour-scale MSE errors.

Misspelling on the 2nd line of the 2nd paragraph under Section 5.8.2 on page 102: . . . InverseRendernet++ . . .

Correction: . . . InverseRenderNet++ . . .

References

[1] *HDRI-Skies*, 2020.

[2] Jonathan T Barron and Jitendra Malik. Shape, illumination, and reflectance from shading. *IEEE transactions on pattern analysis and machine intelligence*, 37(8):1670–1687, 2014.

[3] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546, 2015.

[4] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.

[5] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proc. NIPS*, pages 2672–2680, 2014.

[6] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623, 2015.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Geoffrey Hinton, Alex Krizhevsky, and Sida Wang. Transforming auto-encoders. *Proc. ICANN*, pages 44–51, 2011.

[9] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in Neural Information Processing Systems*, pages 2017–2025, 2015.

[10] HDR Labs. *sIBL Archive*, 2007–2012.

[11] Zhengqi Li and Noah Snavely. CGIntrinsics: Better intrinsic image decomposition through physically-based rendering. In *European Conference on Computer Vision (ECCV)*, 2018.

[12] Andrew Liu, Shiry Ginosar, Tinghui Zhou, Alexei A. Efros, and Noah Snavely. Learning to factorize and relight a city. In *ECCV*, 2020.

[13] Matthew M Loper and Michael J Black. OpenDR: An approximate differentiable renderer. In *Proc. ECCV*, pages 154–169. Springer, 2014.

[14] Shengjie Ma, Qian Shen, Qiming Hou, Zhong Ren, and Kun Zhou. Neural compositing for real-time augmented reality rendering in low-frequency lighting environments. *Science China Information Sciences*, 64(2):1–15, 2021.

[15] Oliver Nalbach, Elena Arabadzhiyska, Dushyant Mehta, Hans-Peter Seidel, and Tobias Ritschel. Deep shading: Convolutional neural networks for screen-space shading. 36(4), 2017.

[16] Fred E Nicodemus. Directional reflectance and emissivity of an opaque surface. *Applied optics*, 4(7):767–775, 1965.

[17] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *2016 International Conference on Learning Representations (ICLR)*, 11 2016.

[18] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[19] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[20] Andrey Zhmoginov and Mark Sandler. Inverting face embeddings with convolutional neural networks. *CoRR*, abs/1606.04189, 2016.

[21] Jacek Zienkiewicz, Andrew Davison, and Stefan Leutenegger. Real-time height map fusion using differentiable rendering. In *Proc. IROS*, pages 4280–4287, 2016.