Self-supervised Outdoor Scene Relighting - Supplementary Document

Ye Yu[1], Abhimitra Meka[2], Mohamed Elgharib[2], Hans-Peter Seidel[2], Christian Theobalt[2], and William A. P. Smith[1]

[1] University of York, United Kingdom
{yy1571,william.smith}@york.ac.uk
[2] Max Planck Institute for Informatics, Saarland Informatics Campus, Germany

1 Relighting of the test dataset

We show more relighting results on the test dataset in Fig. 1. Along with relit images, we also include inverse rendering results. The first column shows the input image. Column 2 to 5 show the inverse rendering results. The last four columns show two pairs of novel illuminations and relighting results. These results demonstrate that our network generates plausible shading, shadows and colour casts for a given novel target illumination.

2 Relighting videos

We use our neural rendering network to perform continuous relighting under time-lapse illumination and rotating lighting conditions.We encourage the reader to look at the supplementary videos. The timelapse HDR environment maps are taken from the Laval HDR sky database [3]. Note that the videos rendered here are the raw output of our neural renderer without any temporal smoothing operations. Our network is able to generate plausible relighting videos under smoothly varying real-world lighting conditions.

3 Relightings without rendered skys

For relighting results shown before, we compare our relightings containing generated skys with other methods whose skys are simply rendered as blacks. To help readers better judge the quality of relighting effects without the distraction from skys, we similarly blend our relighting results with black skys and show them in Fig. 2. Other than such no-sky relighting results, we also demonstrate the relighting comparison under identical generated skys in this figure, where we blend generated skys originated from our results with comparison relighting results. Images used in Fig. 2 are from the test split in MegaDepth dataset [5]. Likewise, we show comparison results on BigTime [4] with same settings on sky regions in Fig. 3.

4 Comparison between shadow generation and Lambertian shading

We show the comparison between shadow prediction from our shadow generation network with Lambertian shading reasoning simple cosine relationship

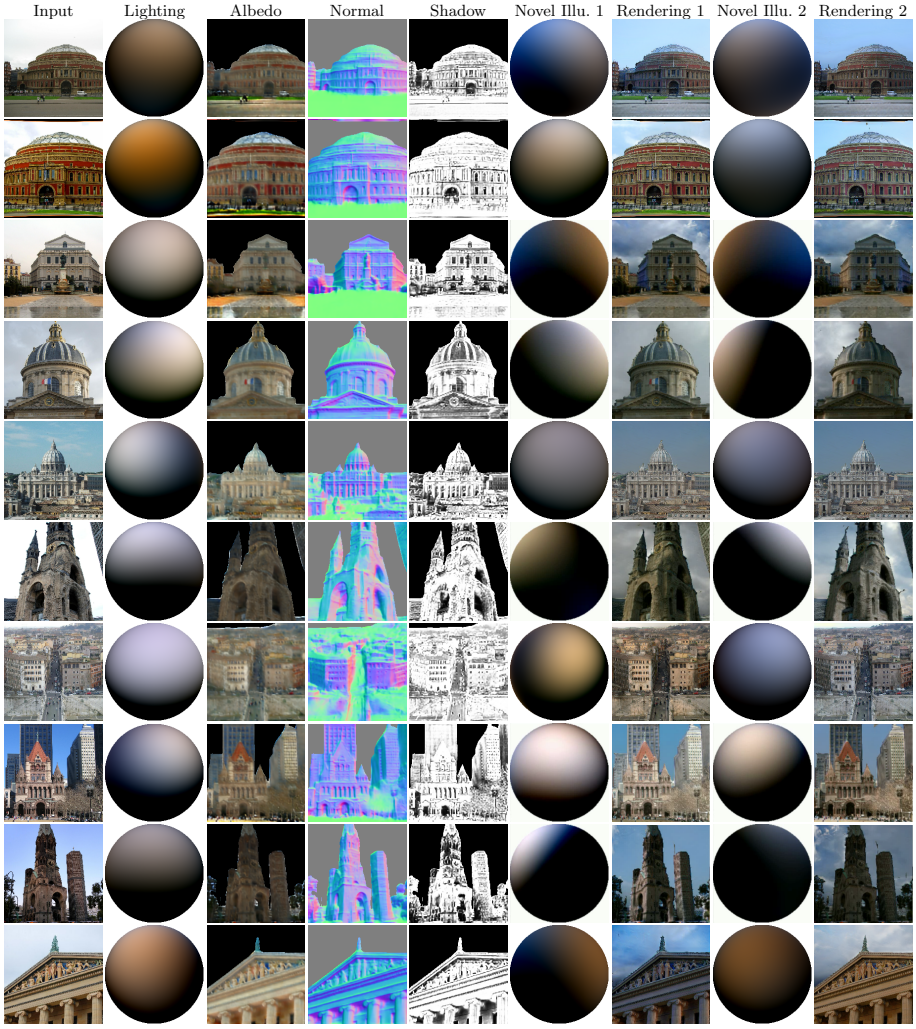

Fig. 1: Inverse rendering and relighting results from our neural rendering network pipeline. Column 1 contains input images to the pipeline. Column 2-5 show inverse rendering results. Column 6 and 7 show two novel targets illuminations. Column 8 and 9 show the relighting results from our neural renderer under the two novel target illuminations.

on illumination direction and surface orientation in Fig. 4. Unlike Lambertian shadings, our shadow generation is capable of hallucinating plausible soft shadow casting effects according to the intensity and direction of the given illumination. In ablation study section we will show that, by doing so, our relightings perform better on generating darker shadows.

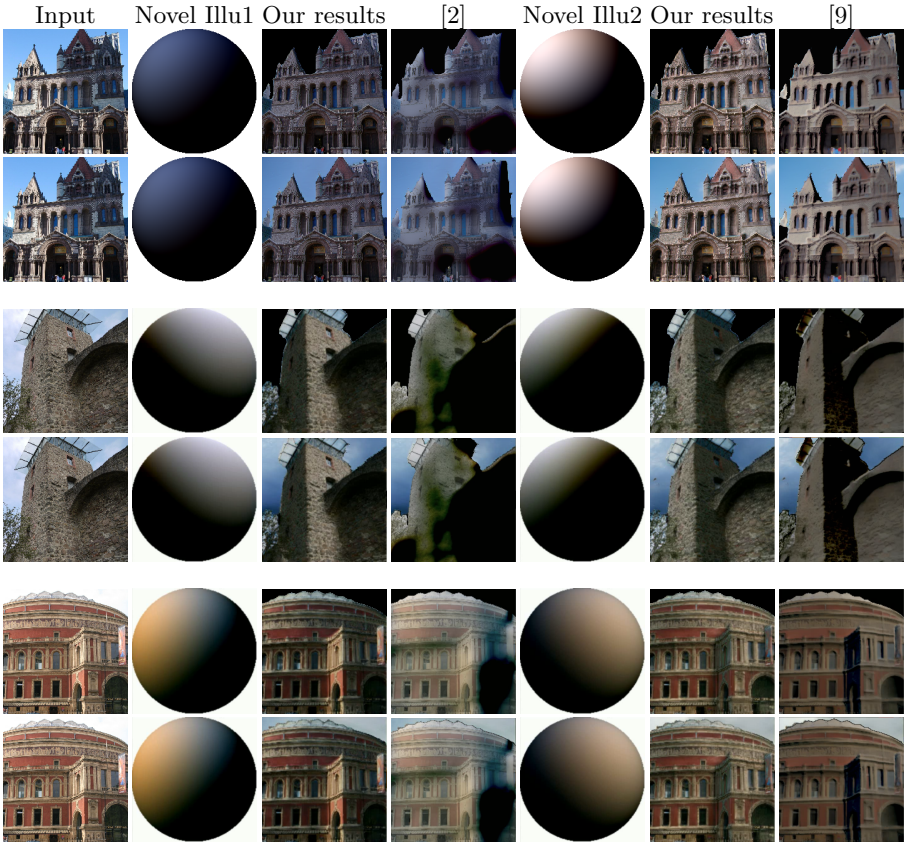

Fig. 2: Relighting results with and without generated skys. Every two consecutive rows form a group of results for an input scene, where the first row shows relightings with skys masked out and the second row is renderings with same generated skys.

Method	Benchmarking		BigTime	
	ℓ_1	SSIM	MSE	SSIM
Full	**0.079**	**0.856**	**0.021**	**0.760**
Without cross-project loss	0.082	0.836	0.023	0.744
Without cycle loss	0.097	0.853	0.026	0.729

Table 1: Quantitative evaluation of the ablated networks on the benchmarking data and the BigTime[4] time-lapse dataset

5 Ablation studies

We show an analysis on some of key design choices we make in our relighting framework. Figure 5 shows results from our neural rendering trained with and without the cross-projection loss. As evident from the figure, the cross-projection

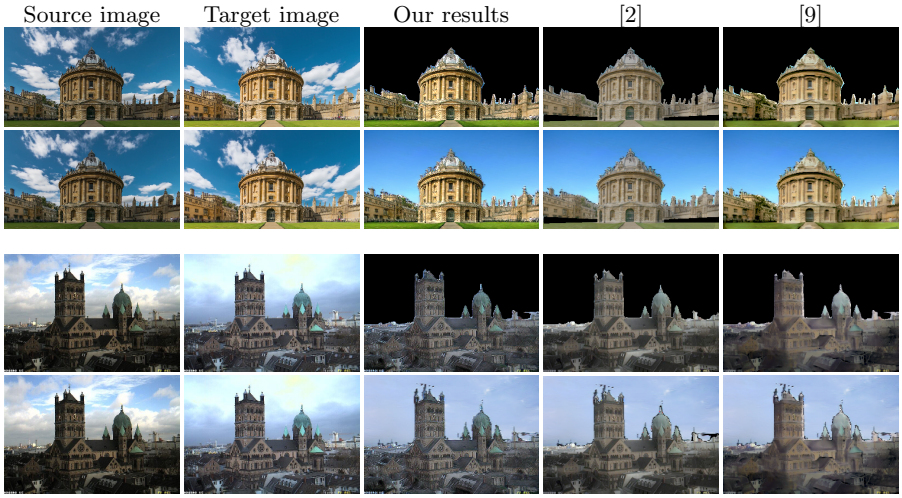

Fig. 3: Results with and without generated skys of BigTime images.

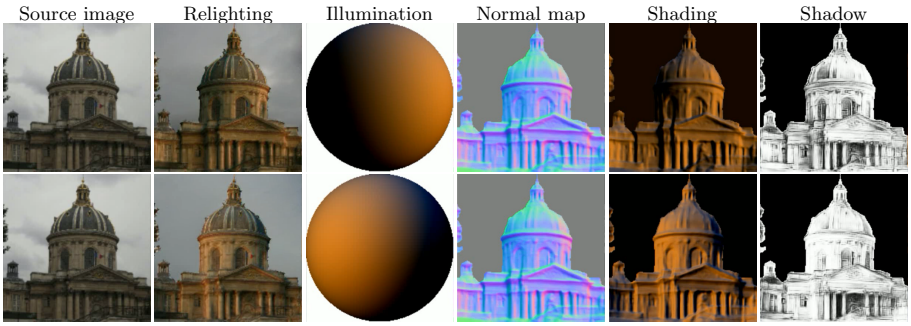

Fig. 4: Comparison between our shadow generation, shown on sixth column, and Lambertian shading, shown on fifth column, under different target illuminations (third column). Both of shadings and shadows are rendered from inputs constructed by same illumination and normal map which are third and forth columns. Initiating from these shading and shadow along with input images, demonstrated on first column and other ingredients, e.g. albedo map, residual map and etc., our relighting network produces relighting renderings that are shown on second column.

loss improves the colour expression and the photorealism of the relit images by preserving the underlying albedo more faithfully, while generating realistic shading. Figure 6 shows differences between rendering network trained by cycle consistency loss and without the cycle consistency loss. The rendering results from training with cycle loss show better shading effects, and perform better on preserving the original reflectance colours and details in the renderings. To quantitatively validate our model against ablation methods, Table 1 shows that model trained with both cross-projection loss and cycle-consistency loss outperforms others on both benchmarking data and time-lapse data.

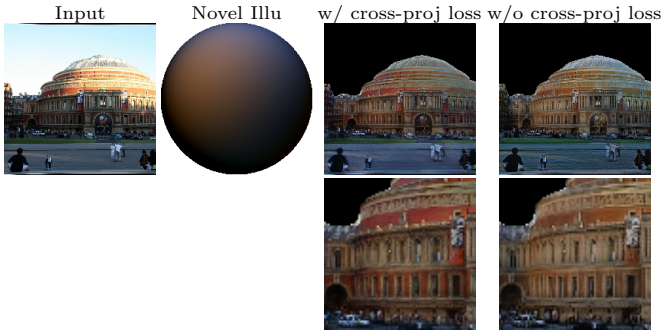

Fig. 5: Performance comparison for training with cross-projection loss and without cross-projection loss.

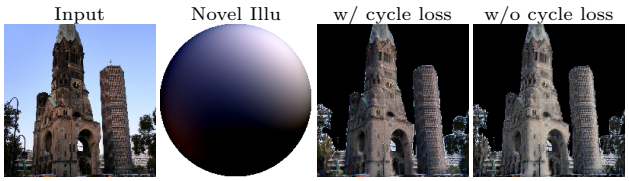

Fig. 6: Performance comparison for training with cycle consistency loss and without cycle consistency loss.

Figure 7 shows a relighting result of our method with and without the input of *shadow map* generated by our shadow network to our neural renderer. Relighting result with the shadow network is able to generate naturally darker shadows. Without the target shadow map, the shadows in the relighting result tend to be lighter and hence appear fake.

Figure 8 shows the utility of our SkyGAN network. As a baseline, we train the neural renderer to learn to generate the sky while relighting the scene, as opposed to having a dedicated SkyGAN for the task. To supervise the sky generation, we apply the same appearance loss over the sky pixels, as while training for the relighting task. Our SkyGAN results show better sky generation with realistic clouds and sky colour variations while the baseline is only able to generate a generic blue colour for the sky region.

Figure 9 shows a relighting result of our method with and without the input of *residual input map* to our neural renderer. Our method is able to reconstruct sharper details than the network trained without residual input map.

6 Dataset calibration

6.1 Multiview stereo

We apply an off-the-shelf uncalibrated structure-from-motion and multiview stereo tool [1] to all 56 images in our benchmark dataset. We initialise the focal length estimates using the known lens focal length in mm and the pixel size on the sensor. The output is a mesh (cropped to the main buildings in the scene) comprising 90k vertices and per-image camera parameters (both intrinsic,

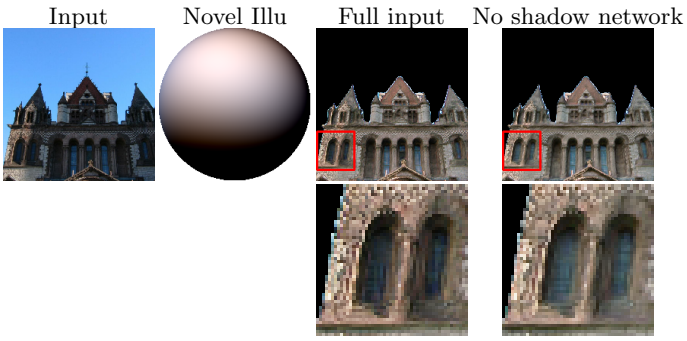

Fig. 7: Performance comparison between training without shadow map given from shadow network and our full input.

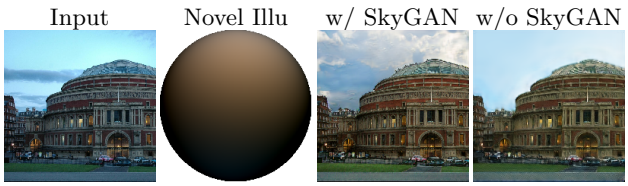

Fig. 8: Performance comparison between training with SkyGAN and training directly learn sky by appearance loss.

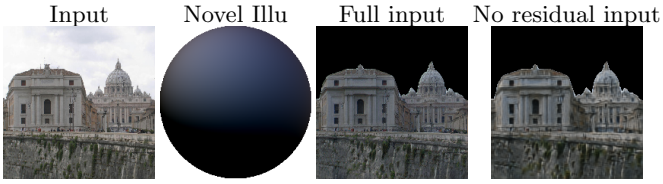

Fig. 9: Performance comparison between training without residual input map and our full input.

including nonlinear distortion parameters, and extrinsic, i.e. a rotation matrix $\mathbf{R}^{\mathrm{w2c}} \in \mathbb{R}^{3\times3}$ and translation $\mathbf{t} \in \mathbb{R}^3$ that transform world to camera coordinates).

6.2 Environment map alignment

We manually label a set of features points on the 3D mesh $\{\mathbf{v}_i\}_{i=1}^n$ with $\mathbf{v}_i \in \mathbb{R}^3$ and corresponding points on the 2D environment map $\{\mathbf{x}_i\}_{i=1}^n$ with $\mathbf{x}_i \in \mathbb{R}^2$. We transform the 2D environment map points into spherical coordinates and then unit length direction vectors $\mathbf{d}_i \in \mathbb{R}^3$ with $\|\mathbf{d}_i\| = 1$. To align the environment map to the world coordinates of the 3D mesh, we solve the following nonlinear optimisation problem:

$$\min_{\mathbf{c},\mathbf{R}} \sum_{i=1}^{n} \arccos\left(\mathbf{R}^{\mathrm{e2w}}\mathbf{d}_i \cdot \frac{\mathbf{v}_i - \mathbf{c}}{\|\mathbf{v}_i - \mathbf{c}\|}\right). \tag{1}$$

This measures the total angular error between the vector from the spherical camera centre to one of the 3D feature points and the corresponding feature on

Fig. 10: Reconstructed scene used in our benchmark. Estimated camera positions are shown as crosses and the aligned environment maps rendered on spheres.

the spherical image (the unit vector rotated to world coordinates). $\mathbf{R}^{\mathrm{e2w}} \in \mathbb{R}^{3\times3}$ is a rotation matrix that rotates environment map coordinates to world coordinates and $\mathbf{c} \in \mathbb{R}^3$ is the centre of the spherical camera in world coordinates. We optimise the rotation as a 3D axis-angle vector meaning the optimisation is 6D overall. We intialise the camera centre as the mean camera position over the multiview dataset. We solve the optimisation problem using the BFGS Quasi-Newton method. In all cases, the mean angular error upon convergence is less than 1°. We show the reconstructed model, estimated camera positions (marked as crosses) and aligned environment maps in Figure 10.

6.3 Benchmark metric

For each reillumination, we rotate the target environment map into the coordinate system of the source camera, i.e. we apply $\mathbf{R}^{\mathrm{w2c}}\mathbf{R}^{\mathrm{e2w}}$ to the environment map in spherical harmonic coefficient space. We then relight using our method or one of the comparison methods. Our evaluation metric is the L1 error between reprojected ground truth and relit image, averaged over colour channels and pixels. To remove unknown scale factors, we compute the error after applying the optimal scaling to the relit image, i.e. the scale that minimises squared error to the ground truth image.

6.4 Alignment of results from [7]

Our comparison results for [7] were provided by the authors of [7] who ran their original implementation on some of our data. As part of their pipeline they nonlinearly undistort and crop the images and did not provide parameters for this transformation. For a fair comparison, we computed the optimal distortion and uncropping parameters to register their output back to our images. To do this, we computed SURF feature matches between their relighting result and our

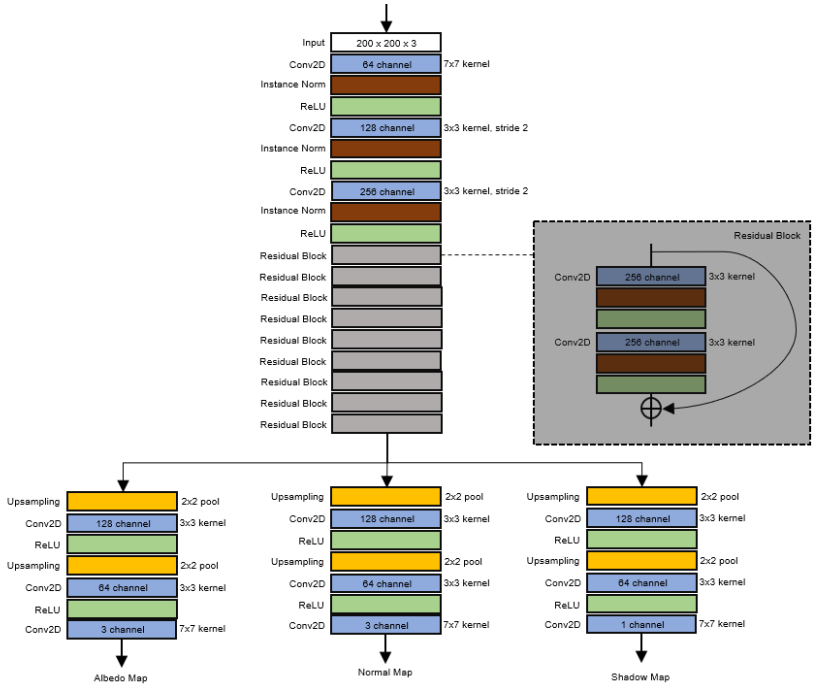

Fig. 11: The architecture of our *InverseRenderNet*, which decomposes a given input image of an ourdoor scene into its intrinsic image components - an albedo map, a normal map and a shadow map.

original input images. Outlying matches were discarded by fitting an approximate similarity transform. Then the remaining matches were used in a nonlinear least squares optimisation to find the forward distortion parameters that optimally aligned the features in their image to the corresponding features in our image.

7 Network architecture & Training

In this section, we detail the architecture of the neural networks used in our pipeline and provide some details about the training.

For our entire training, we use a fixed learning rate of 5×10^{-4}. For the various loss functions used in training the neural renderer, the relative weights are chosen such that the value of the losses are in the same order of magnitude: [albedo cycle consistency loss, shadow cycle consistency loss, normals cycle consistency loss, lighting cycle consistency loss, self reconstruction loss, cross-projection loss] $\equiv (3.0, 0.4, 1.0, 4.0, 1.0, 0.5)$.

Figure 11 shows the architecture of our inverse rendering network. We modify the *InverseRenderNet* of Yu and Smith [9] to include residual blocks and use instance normalization. We note that the instance normalization ensures that the contrast changes in the input image due to lighting variations do not appear in the estimated albedo and normal maps. In our experience, the residual blocks improve

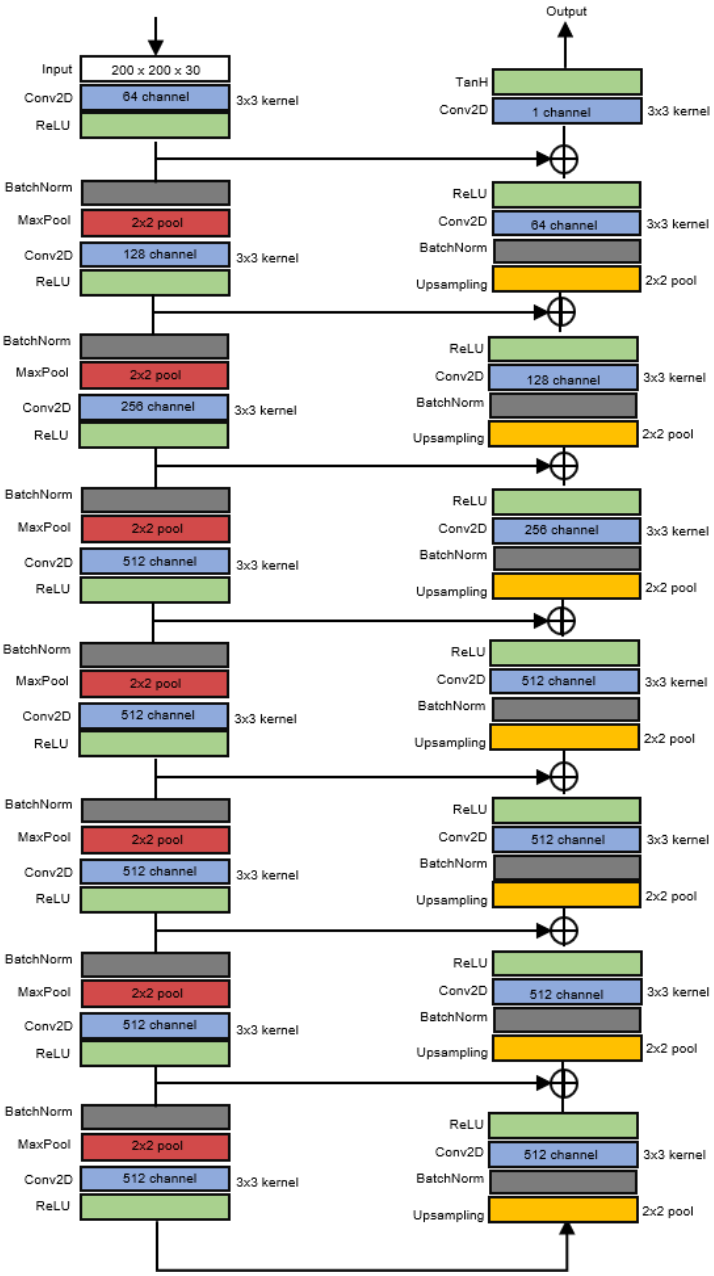

Fig. 12: The architecture of our *ShadowNet*, which learns to generate a shadow map from an input normal map and lighting condition.

the quality of the gradient back propagation and lead to better convergence of the networks, while providing higher quality results.

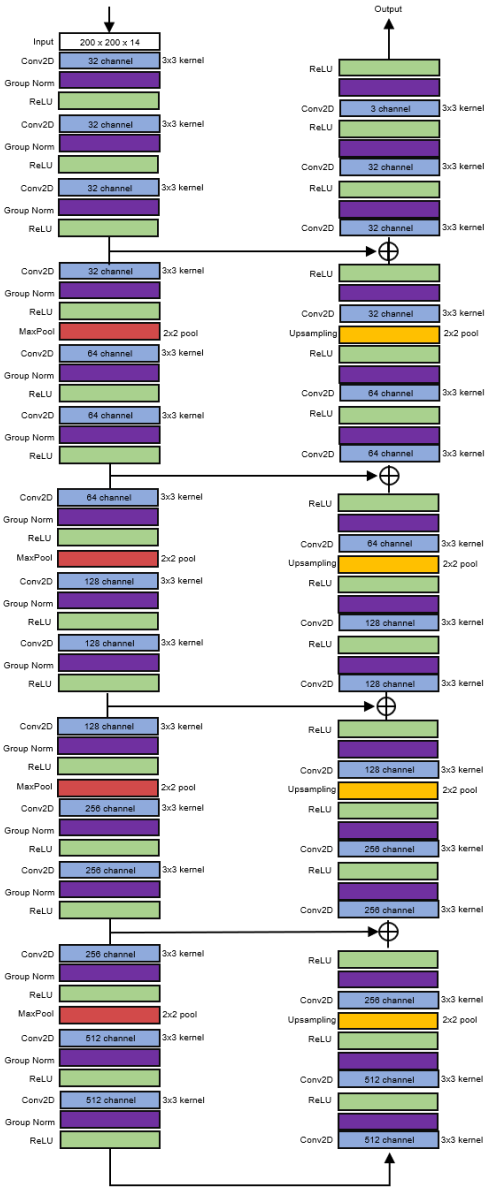

Fig. 13: The architecture of our *NeuralRenderer*, which learns to generate a photorealistic relighting result from given input appearance maps (albedo, shading, normals, shadow, detail, foreground segmentation mask).

Figure 12 shows the architecture of our shadow generation network. The network is modelled after a convolutional U-NET [8] which contains skip-connections from the encoder to the decoder network. We practically observe that the training

loss experiences large fluctuations in the initial stages, which we prevent and smooth using a batch normalization layer after every convolutional block.

Figure 13 shows the architecture of our neural rendering network. It is a deeper convolutional neural network with intermediate skip-connections, with group normalization. In order to ensure that our network does not overfit to the dataset and is able to generalize across variety of scenes, we use a deeper network with 15 convolutional block in each the encoder and the decoder part.

For our sky generation network, we use the same architecture as that of GauGAN [6], which uses spatially-adaptive normalization to ensure that semantic information in the output is preserved, as is the need of our sky generator network. We refer the reader to their paper to see the full architecture. We modify the interfaces of their network to fit our pipeline. We change the original conditional input of their semantic layout to our concatenation of the sky segmentation map and the relit output from the neural renderer. We modify the number of residual block to 8, and we change the length of the input random noise vector to 256. The rest of the architecture remains the same.

References

1. Agisoft, L., St Petersburg, R.: Agisoft metashape. Professional Edition **7** (2019)
2. Barron, J.T., Malik, J.: Shape, illumination, and reflectance from shading. TPAMI (2015)
3. Lalonde, J.F., Asselin, L.P., Becirovski, J., Hold-Geoffroy, Y., Garon, M., Gardner, M.A., Zhang, J.: The Laval HDR sky database. http://sky.hdrdb.com (2016)
4. Li, Z., Snavely, N.: Learning intrinsic image decomposition from watching the world. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9039–9048 (2018)
5. Li, Z., Snavely, N.: Megadepth: Learning single-view depth prediction from internet photos. In: Computer Vision and Pattern Recognition (CVPR) (2018)
6. Park, T., Liu, M.Y., Wang, T.C., Zhu, J.Y.: Semantic image synthesis with spatially-adaptive normalization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2019)
7. Philip, J., Gharbi, M., Zhou, T., Efros, A.A., Drettakis, G.: Multi-view relighting using a geometry-aware network. ACM Trans. Graph. **38**(4), 78:1–78:14 (Jul 2019)
8. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) Proceedings of the International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI). pp. 234–241 (2015)
9. Yu, Y., Smith, W.A.P.: InverseRenderNet: Learning single image inverse rendering. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)