Deep Appearance Maps
— Supplementary Material —

Maxim Maximov
Technical University Munich

Laura Leal-Taixé
Technical University Munich

Mario Fritz
CISPA Helmholtz Center for Information Security

Tobias Ritschel
University College London

1. Introduction

In this supplemental document, we show more results, perform evaluation of adversarial loss and introduce additional applications with our method. We also included more detailed pipeline figures for each task in Fig. 3.

2. Adversarial loss

In this section, we explain how we trained our models with adversarial loss and show advantages of using it.

2.1. Training

We implemented DAM as a stack of convolution layers, each layer applies 1x1 convolutions using stride 1 without padding so it maintains the same resolution. We initialize DAM model weights using Xavier initialization, which makes convergence faster and more stable.

Since DAM model uses 1x1 convolutions, we can input any geometry (in the form of normals) to apply our learned appearance on it. We use this to add the adversarial loss to our method: instead of comparing a reconstructed model, we pass RM normals from the same view through a DAM to see how RM would look from the same position. Estimated RMs are going to be "fake" samples in the discriminator and "real" samples are pre-rendered RMs. We rendered RMs similarly as our main dataset - from varying angles, with different point-light and environment maps illuminations. During training, DAM estimate RMs from several views and purpose of adversarial loss is to make them look more "realistic" - similar as in ground truth RMs. The left part of Fig. 3 shows our DAM pipeline with adversarial loss.

2.2. Comparison

Here we study the influence of the adversarial loss on DAM. We compare using only L2 loss and combination of L2 with the adversarial loss.

We trained DAM networks on 20 image sequences with adversarial loss and without it. Then we computed DSSIM error on our training and test set for both loss combinations.

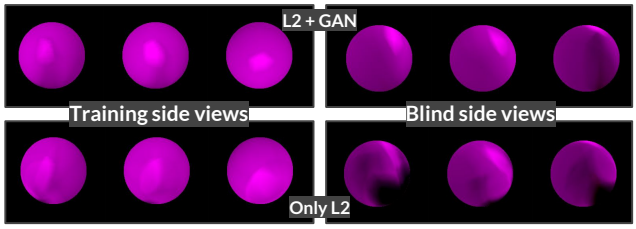

Figure 1. Results of our DAM representation trained using only L2 loss (bottom six) and L2 with adversarial (GAN) loss (top six). We show the same sphere estimated from two different sides: training side views were taken close to the training set and blind side views were taken from the opposite side.

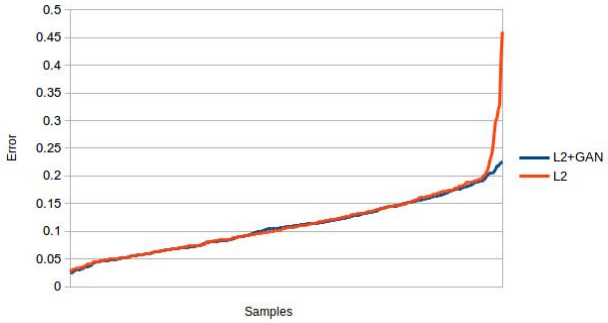

Figure 2. DSSIM error plot for two types of losses. Each curve is produced by sorting all samples in the dataset by their DSSIM (less is better).

Average DSSIM using only L2 loss is .126 and L2 with adversarial loss is .120. The adversarial loss showed only a bit better result. To investigate further we sorted and then plotted errors from all samples to Fig. 2. We can see that using only L2 loss produces outliers for few samples, while L2 with adversarial loss showed more stable results.

Additionally, we show qualitative results of using the adversarial loss in Fig. 1. We use a small weight on our adversarial loss, so samples that were closer to the training views look similar in both cases because L2 loss has more impact. Samples that are far from the training distribution

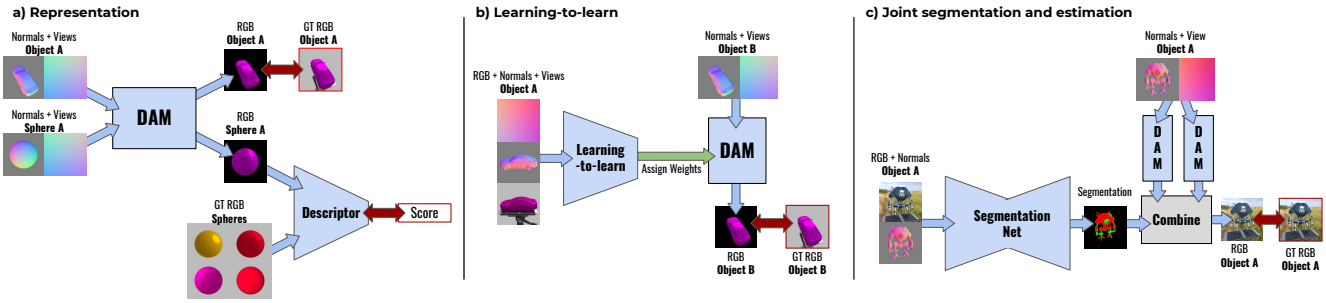

Figure 3. Pipeline figures for each task. Red arrows show losses between estimated and ground truth images. The joint task on the right part has "Combine" module that takes outputs of two DAM models and multiplies them with respective estimate segmentation channel to produce reconstruction.

(taken from the "blind" side) mostly rely on the adversarial loss. We noticed that DAM can over-train and produce missing (black) regions on a blind side - it's a reason for outliers in Fig. 2. The adversarial loss prevents such over-trained results.

Having adversarial loss is not an absolute necessity since it slows down the training. But it still helps to stabilize output for unseen combinations of input normals and camera views.

3. Example application: Denoising and Super-resolution

We have formally introduced and analyzed three tasks making use of DAMs. However, we believe DAMs not be limited to these, and would argue that representing appearance itself as a network allows for interesting applications.

An example of further applications is de-noising of Monte-Carlo rendered images. This works as follows: An input to the representation task is just a number of unstructured samples of normal, view direction and RGB pairs. Therefore, we can also insert noisy path-traced pixels, either from the full image or from a (weighted) local neighborhood. The procedure will fit a NN $\hat{L}$ to these observations that reproduces the data, yet is non-suspicious to the adversarial. This is similar to image filters that fit a linear or higher-order model to data and the re-evaluate it for smoothing, just that the model has become a neural network that is a function of normal and view direction, not pixel position. Other modern MC denoisers use deep networks, but to produce colors directly [1, 4]. Our NN $\hat{L}$ will smooth the data, but it will preserve 4D-directional dependence. Also, it is not by definition band-limited such as a linear basis like spherical harmonics: it will preserve small highlights that correctly respond to view or surface orientation changes. Consequently, it can be evaluated at arbitrary normals and view directions, including the ones it was fit to, but also for a slightly different image, or for an image in a higher resolution.

In Fig. 4, we show de-noising results: The DAM is fit to a noisy path-traced image and when re-evaluating it, the noise is disappeared, but still, the highlights and shading details are preserved. In a more refined version, a DAM would be fit to local patches. Additionally, in Fig. 5 we show qualitative comparison between DAM, build-in Blender Cycles [3] denoiser and state-of-the-art NFOR denoiser [2] on renders.

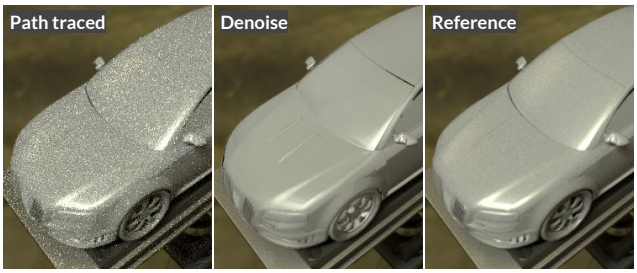

Figure 4. De-noising results. It shows a noisy path-traced input image (left), our de-noising result (middle) and a reference that was rendered with a high number of samples(right). Our approach trains a DAM on noisy images, and because the DAM is going through all combinations of normals and views, it can find an averaged result. So the output of the DAM with the same input is a de-noised version.

Another application is super-resolution. Rendering high-resolution (HR) image is a time-consuming task. Our DAM uses 1x1 convolutions, so it can take as input images with varying resolution. DAM can be trained on low-resolution (LR) images and then we can pass HR images of normals and view directions to render our HR appearance.

In Fig. 6, we show super-resolution results: we train several DAM on different LR images types (from 32x32 to 256x256) and then estimate HR image (512x512) given HR normals. We also provide LR image to demonstrate resolution that each respective DAM was using during training. Additionally, we show result of state-of-the-art single image super-resolution method [5].

HR results look comparable. DSSIM results on HR test set are very similar: .125, 0.112, 0.110, 0.122 for 32x32, 64x64, 128x128, 256x256 resolutions respectively. It shows that we can train DAM on images with lower resolution

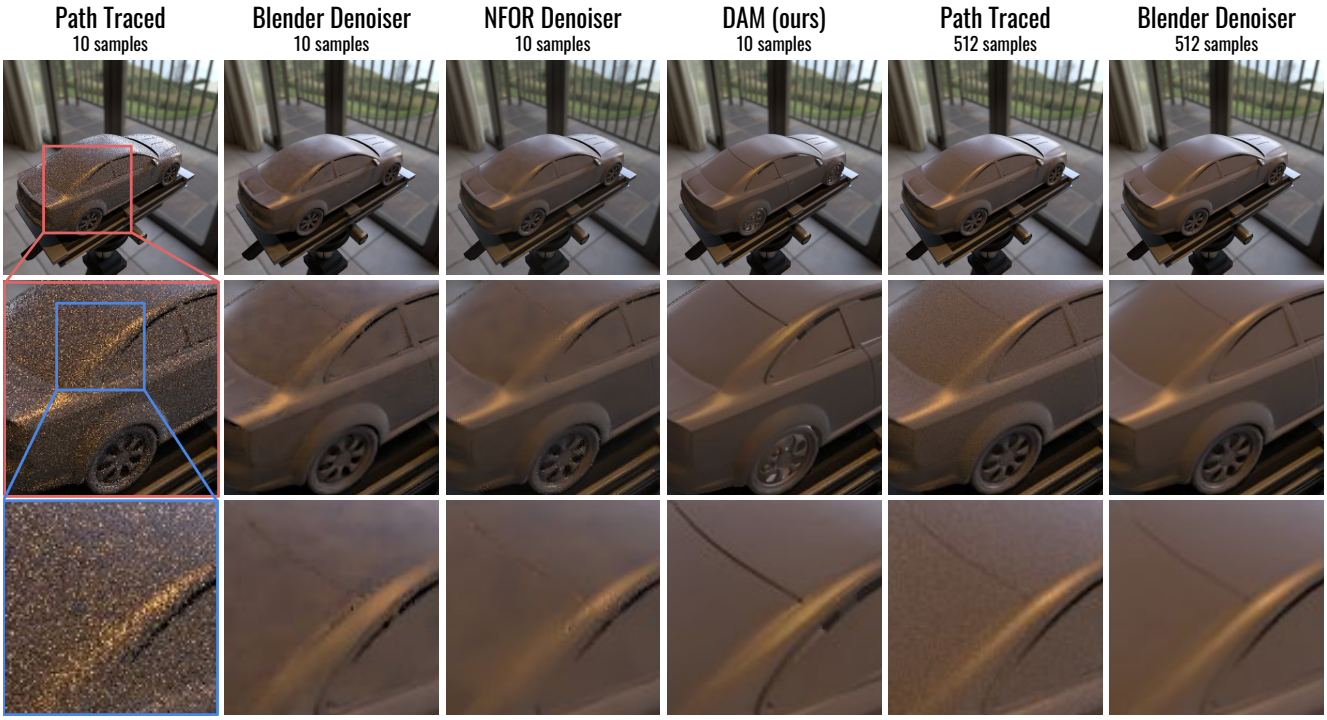

Figure 5. De-noising results. [2] It shows a noisy path-traced input image (left) rendered with only 10 samples, results of several de-noisers (Blender Cycles, NFOR, DAM), a reference that was rendered with a high number of samples (512) and the same high sample reference de-noised with Blender Cycles. Middle and bottom rows show the same results on different zoom levels.

and still get comparable results on HR images. We used this notion to train DAM on real data where the original resolution was 1024x1024 and we trained on downscaled 256x256 images to reduce computation time.

While an in-depth comparison to other de-noising and super-resolution alternatives is required to understand its full potential, it indicates many more applications of DAMs exist to explore.

4. Additional Results

We show more results on appearance transfer (see Fig.7), de-noising (see Fig. 8, Fig. 9), a single representation (see Fig. 10), failure cases (see Fig. 11), Learning2Learn (see Fig. 12)), joint multi-material segmentation and estimation (see Fig. 13, Fig. 14).

References

[1] Steve Bako, Thijs Vogels, Brian McWilliams, Mark Meyer, Jan Novák, Alex Harvill, Pradeep Sen, Tony DeRose, and Fabrice Rousselle. Kernel-predicting convolutional networks for denoising monte carlo renderings. *ACM Trans. Graph. (Proc. SIGGRAPH 2017)*, 36(4), 2017. 2

[2] Benedikt Bitterli, Fabrice Rousselle, Bochang Moon, José A. Iglesias-Guitián, David Adler, Kenny Mitchell, Wojciech Jarosz, and Jan Novák. Nonlinearly weighted first-order regression for denoising monte carlo renderings. *Comp. Graph. Forum (Proc. EGSR)*, 35(4):107–17, 2016. 2, 3

[3] Blender Foundation. *Blender - a 3D modelling and rendering package*, 2018. 2

[4] Chakravarty R Alla Chaitanya, Anton S Kaplanyan, Christoph Schied, Marco Salvi, Aaron Lefohn, Derek Nowrouzezahrai, and Timo Aila. Interactive reconstruction of monte carlo image sequences using a recurrent denoising autoencoder. *ACM Trans Graph. (Proc. SIGGRAPH)*, 36(4):98, 2017. 2

[5] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. ESRGAN: Enhanced super-resolution generative adversarial networks. In *Lecture Notes in Computer Science*, pages 63–79. Springer International Publishing, 2019. 2, 4

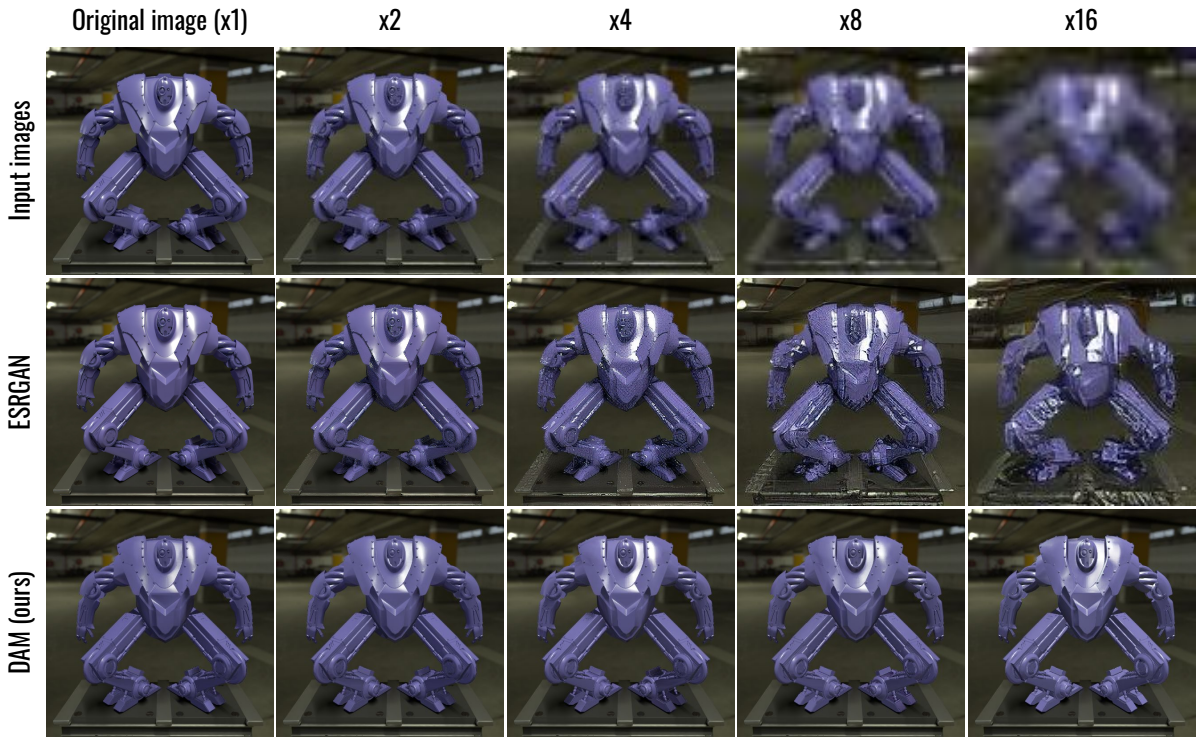

Figure 6. Super-resolution comparison. The left column shows HR ground truth image, all other columns show LR images that are x times of the original image scale. The top row shows input images. The middle row shows results from [5]. The bottom row shows results of DAM models (trained on LR images with corresponding scale) if input normals were of the same resolution as the original image.

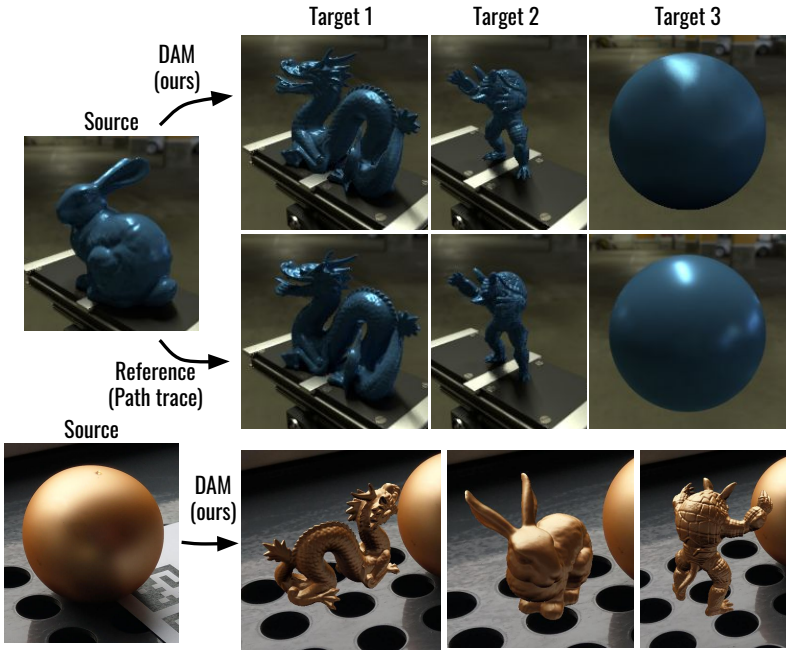

Figure 7. Appearance transfer from a source shape (left)to multiple target shapes (right) using DAM (1st and 3rd and row) and a path-traced reference rendering of the target shape with the source appearance (2nd row).

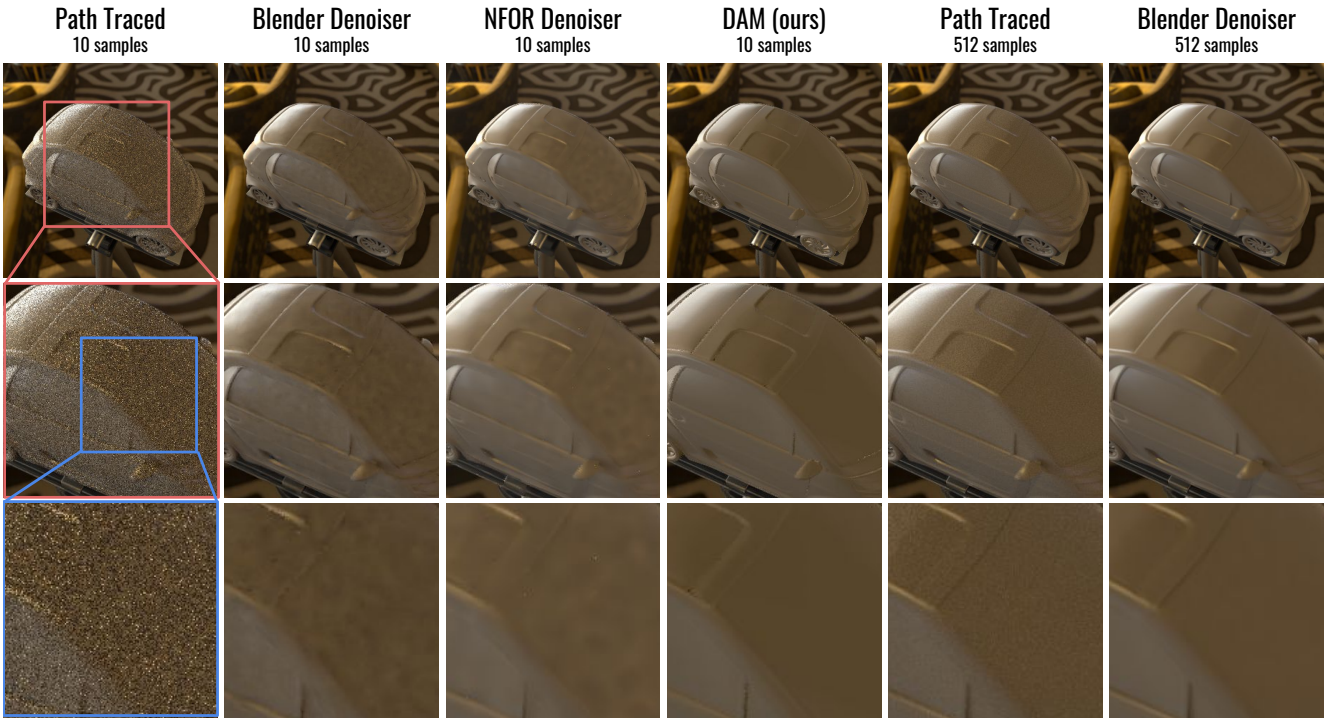

Figure 8. De-noising comparison results.

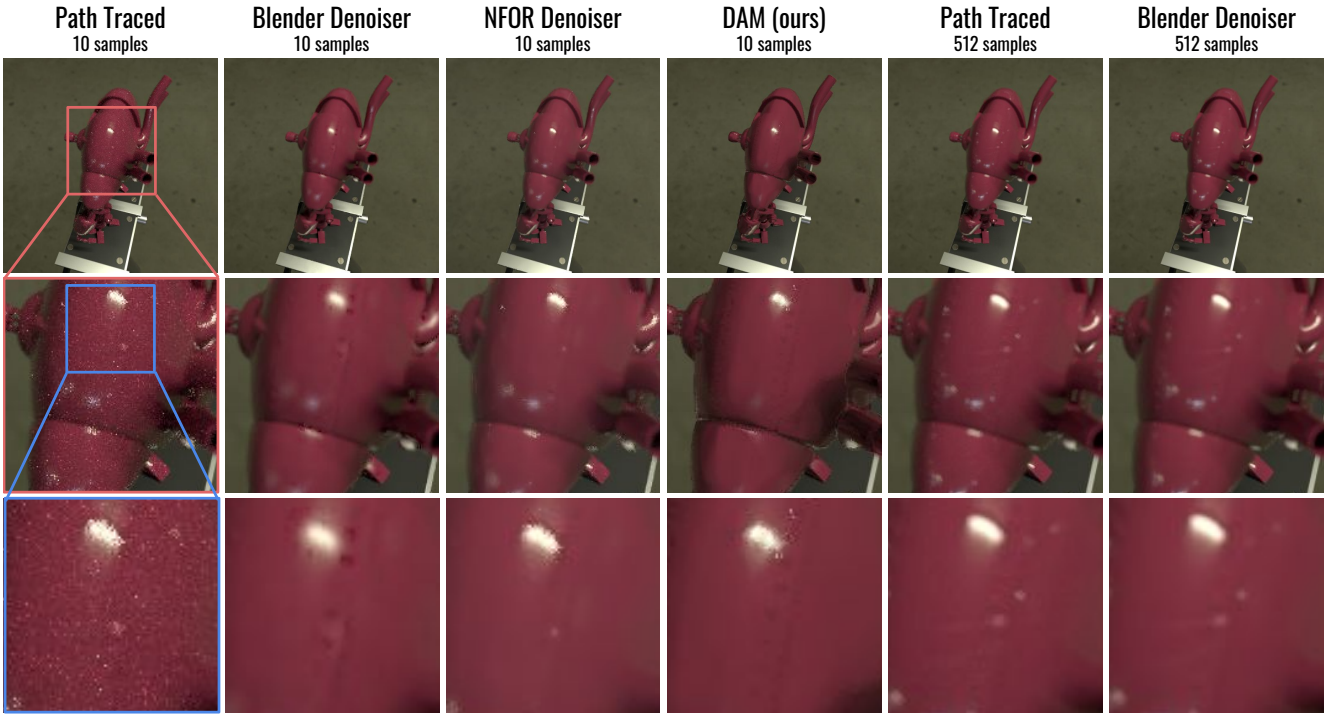

Figure 9. De-noising comparison results.

Figure 10. Results of reflectance maps **(1st and 4th column)**, our deep appearance representation **(2nd and 5th column)** and the reference **(3rd and 6th column)** for a novel view. Our approach performs quite well with diffuse materials and with a strong point light. Note that all object are rendered with self-shadowing and global illumination turned on, which leads to some unnaturalness and artifacts in some results. Both approaches produce close to ground-truth results. However, reflectance maps are taken from the ground truth and present the best case that should be able to be estimated.

Figure 11. Here we present most common failure types in our method. Our approach has difficulties in reconstructing high-frequency information on reflective materials **(1st row)**. Small highlights are also not always stored in our network **(2nd row)**. Mainly because they generate a smaller error and the network doesn't pick up on it. Similarly, the network doesn't always reconstruct highlights that have very similar hue as the diffuse albedo **(3rd row)**. Unobserved sides tend to deviate from the ground truth **(4th and 5th row)**.

Figure 12. Results of our DAM representation trained using stochastic gradient descent **(2nd column)**, our DAMs produced by our learning-to-learn (L2L) network **(3rd column)** from the input view **(1st column)** as well as a reference **(4th column)** in a novel-view task. L2L was trained from many sides on many appearances, it has the advantage of having more stable outputs for unobserved views than just a single DAM representation trained only on one specific appearance from few views. However, L2L struggles to reconstruct highlights.

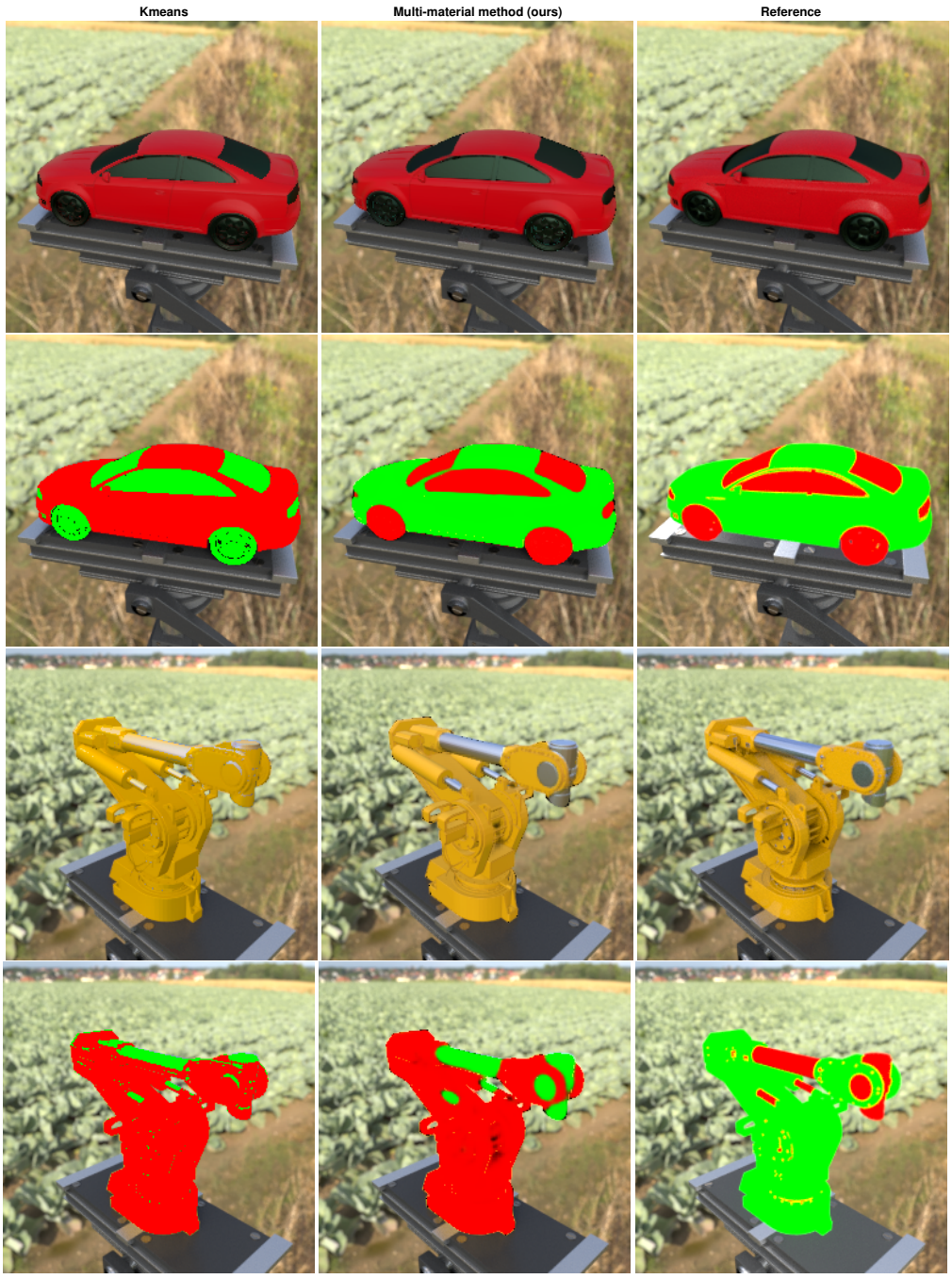

Figure 13. Results of our joint multi-material segmentation and estimation **(2nd column)**, K-means **(1st column)** and reference **(3rd column)** in a novel-view task. We can see reconstruction **(1st and 3rd row)** based on segmentation**(2st and 4rd row)**.

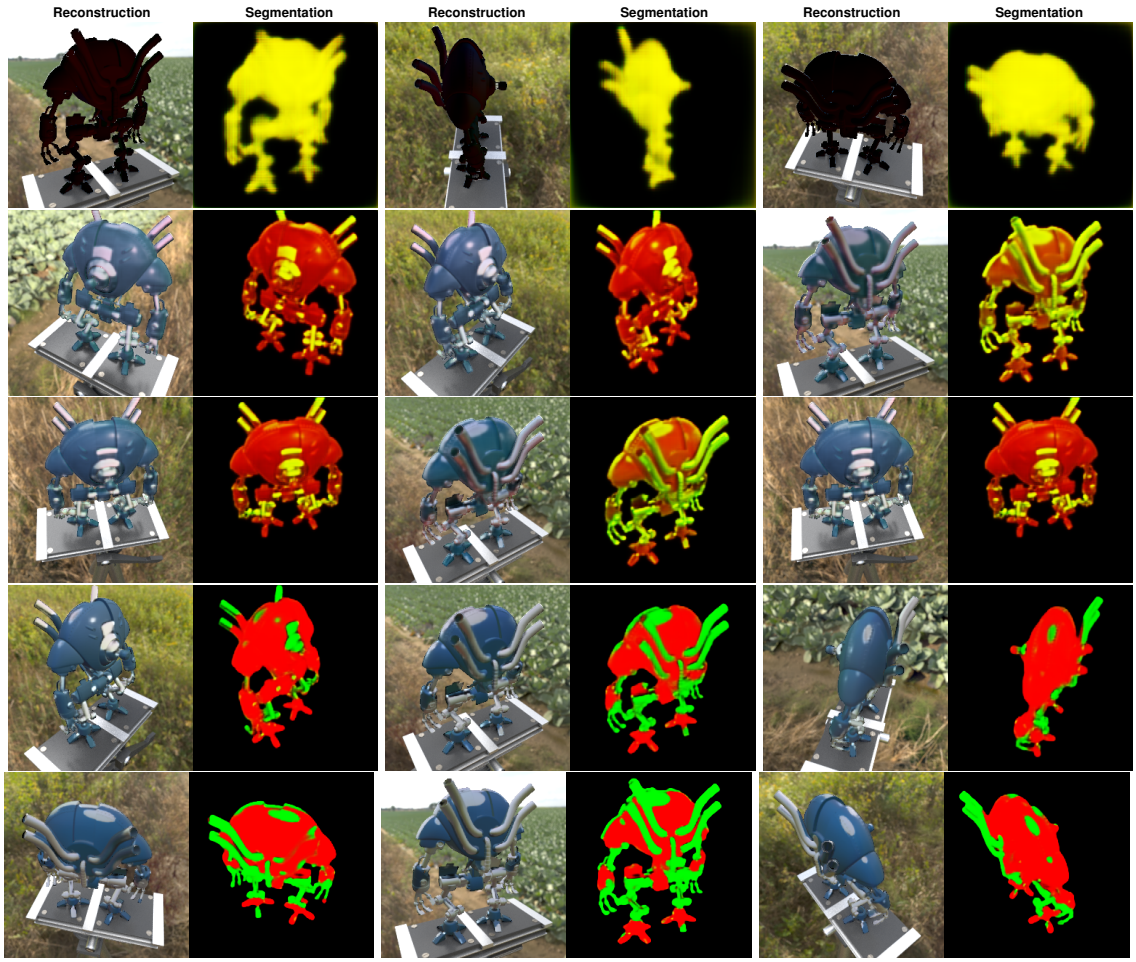

Figure 14. Progressive multi-material segmentation and estimation process. Each row shows 3 random views of the same object presented in a pair of reconstruction and respective segmentation. All pixels of both mask channels are equal to 1. From there, the model gradually make each pixel to choose certain material based on current estimated materials.