UMat: Uncertainty-Aware Single Image High Resolution Material Capture Supplementary Material

Carlos Rodriguez-Pardo[1,2] Henar Dominguez-Elvira[1,2] David Pascual-Hernandez[1] Elena Garces[1,2]
[1]SEDDI, Spain [2]Universidad Rey Juan Carlos, Spain

1. Overview

In this supplementary material, we aim to provide extensive results, implementation details and further analysis that are not present on our main paper. We accompany this document with a video, with high resolution images to illustrate the capabilities of our material estimation method.

We divide this supplementary material in different sections, as follows:

- On Section 2, we provide exhaustive implementation details for our model design and training configuration, as well as our metrics.
- On Section 2.4, we provide details on the configurations we used to capture the materials.
- On Section 2.5, we provide details on how we performed comparisons with previous work.
- On Figs. 1 and 2, we illustrate our training dataset.
- On Figs. 3 and 4, we provide exhaustive diagrams of our generator and discriminator architectures.
- On Figs. 5 to 11, we provide additional results of our ablation study of our model architecture.
- On Figs. 12 and 13, we provide additional results of our uncertainty metric.
- On Fig. 14, we provide additional results of our active learning experiment.
- On Figs. 16 to 18, we show results of our model robustness when we apply different degradations to the input images. To ease comparisons, we use the same subset of materials in our test set. Our model is robust to many types of input degradations.
- On Fig. 19, we show results of our model on datasets of previous work, including smartphone images, material graphs and captured BTFs.
- On Tabs. 1 to 11, we provide additional comparisons with previous work for a set of materials on our test set.

2. Implementation Details

2.1. Model Design

Our model is trained using a GAN framework. In this section, we detail our design choices for the generator and discriminator architecture.

Generator For the generator, we use a U-Net [18] model, with a few modifications design to maximize its efficiency, robustness, and generalization capabilities. We specify the full model architecture and layer sizes on Figure 3. We use residual connections [2, 3, 7] in every convolutional block of the model, for better training convergence and preserving details present in the input images. We use 1×1 convolutions on the skip connections. Further, to maximally preserve the appearance and characteristics of every target map, we use a single decoder for each. This has been proposed in different applications, including intrinsic images, material capture and texture synthesis [2, 6, 17, 28]. To improve the results and enable our uncertainty metric, we append a pixel-wise MLP to each decoder in the model, with Dropout [21] regularization. Using MLPs after the decoders has been previously explored for material capture [1]. We use Group Normalization [26] (with 16 *groups* per layer) and SiLU [4] non-linearities throughout the model. Each convolutional block in the encoder is enhanced with a lightweight Linear Attention module [22], with 4 *attention heads*, each with a dimension of 32 hidden units. On the bottleneck, we use a lightweight *MobileViT* Transformer block [11], with 128 hidden dimensions for the self-attention and MLPs, 4 layers and a kernel size of 3. We use a Dropout rate of 0.2. We use transposed convolutions for upsampling. Every other implementation detail in the model (strides, bias, poolings) follow [18].

Discriminator For the discriminator [19], we also use a U-Net [18] model, with a few modifications to improve its performance as a discriminator. We specify the full model architecture and layer sizes on Figure 4. As in the generator, we use residual connections [2, 3, 7] in every convolutional block of the model, for better training convergence and preserving details present in the input images. We use Spectral Normalization [13] and SiLU [4] non-linearities throughout the model. On the bottleneck, we use a lightweight *CBAM* attention block [24]. The single-scalar estimation of the discriminator $\mathcal{D}_{enc}$ is provided by a MLP with a similar architecture to the *CBAM* module. We use transposed convolutions for upsampling. Every other implementation detail in the model (strides, bias, pooling) follow [18].

2.2. Model Training

Optimization We train the models using PyTorch [14] and TorchVision [10]. We leverage Kornia for data augmentation [15]. To accelerate the training process, we leverage mixed precision training and automatic gradient scaling [12], and train the whole model natively on GPU. Optimization is done using Adam [9]. Following [19], we use different learning rates for the generator lr $= 0.001$ and the discriminator lr $= 0.005$ and a batch size of 10.We train the models for 100 epochs, which takes 10 hours on an NVIDIA RTX 3060 GPU.

Loss Function We use the following weights for the loss function: $\lambda_{\measuredangle} = 3, \lambda_{spec} = 1, \lambda_{rough} = 1, \lambda_{adv} = 0.2, \lambda_{style} = 0.25, \lambda_{freq} = 0.2, \lambda_{cons} = 0.3$. For the style loss function, we use the *AlexNet* variant of LPIPS [27], as it provides a lightweight style loss which has shown success on texture transfer [16].

2.3. Artifact Detection

The thresholds for each material map and the kernel size for the uniformity metric have been optimized given a set of 102 manually labeled textures: $t_1(\mathrm{M}_s) = 0.01$, $t_2(\mathrm{M}_s) = 1.41$, $t_3(\mathrm{M}_s) = 1.33$; $t_1(\mathrm{M}_r) = 0.01$, $t_2(\mathrm{M}_r) = 0.99$, $t_3(\mathrm{M}_r) = 3.12$. The size of the box filter is $s_{\text{Box}} = 127.5$.

2.4. Capture Details

We construct the training and testing dataset capturing 10×10 cm samples at 1000 PPI using an *EPSON V850 Pro* on its defaults settings.

For the comparisons with previous work, we place the fabric samples on a black surface. We use a *Huawei Nova 5T* smartphone, and capture the materials at two distances: a *close-up*, capturing 4×4 cms at a distance to the sample of 8 cms, and a *full-size* image, capturing capturing 8×8 cms, at a distance to the sample of 13 cms. We use ISO=50, aperature of $\frac{f}{1.8}$, and a focal length of 26mm for every image. For the two distances, we capture the material using ambient illumination, with exposure times of $\frac{1}{10}s$ for the *full size* and of $\frac{1}{8}s$ for the *close-up*. We also capture the images using the smarphone flash lighting, with exposures of $\frac{1}{40}s$ and $\frac{1}{50}s$ for the *full size* and *close-up*, respectively.

2.5. Comparisons with Previous Work

We perform every comparison with previous work on a RTX NVIDIA 2080, using the default configuration for every method. However, for [20], we initialize the material graph with a fabric material (`fabric suit vintage`) provided in their repository, to better match our test data. For [8], we use the *fine-tuning* configuration.

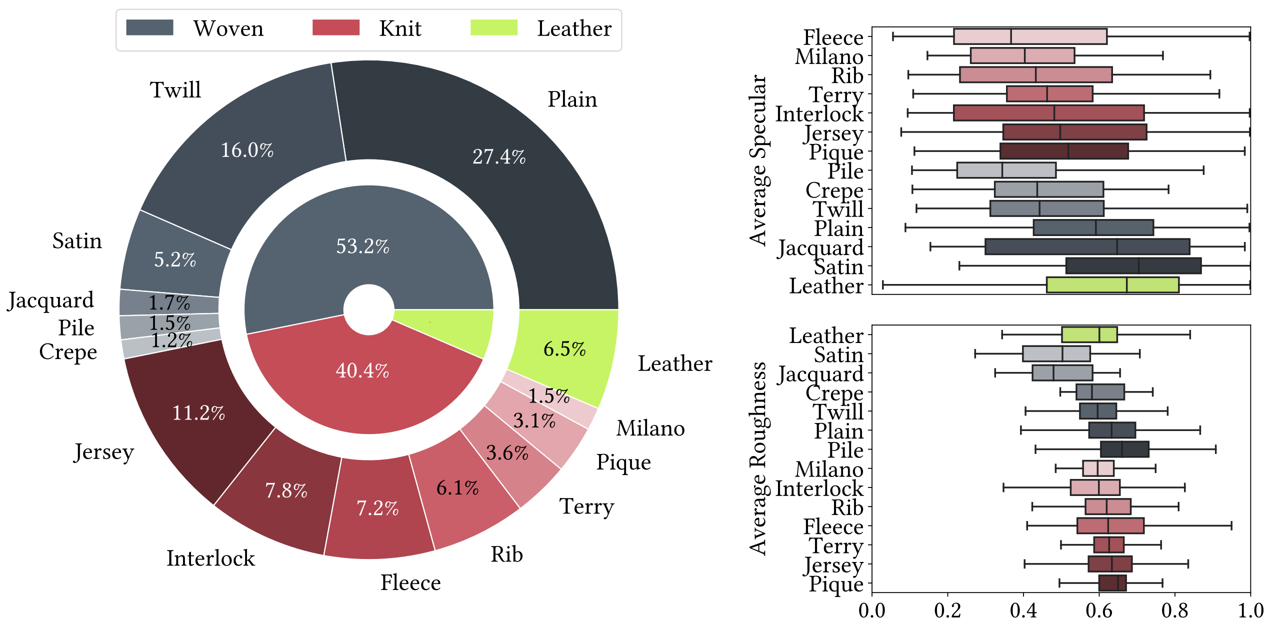

Figure 1. Visualization of our dataset. We show the percentages of materials in our training dataset, including more detailed subcategories. On their right, we show the average specular and roughness for every category. As shown, there are some structures with distinct characteristics: Satins are highly specular due to the particularities of their yarns, and Piles (eg corduroy) or Plain Weave (eg linen fabrics) are much less glossy. We exploit this relationship between microgeometry and specularity for our estimations.

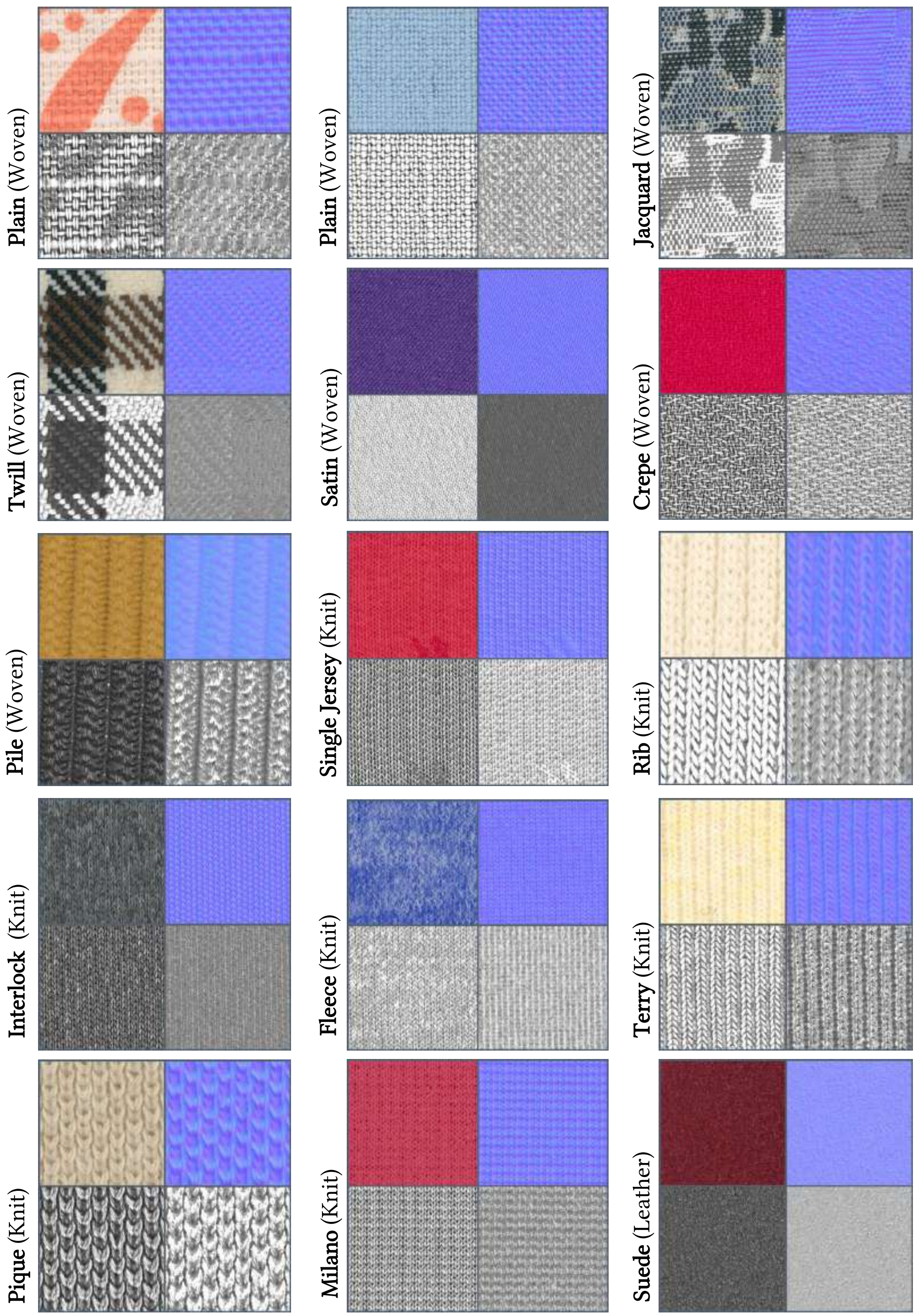

Figure 2. Visualization of some Ground Truth SVBRDF of the different families in our test set. Textiles have very complex and varied microstructures which play an important role on their appearance at different scales.

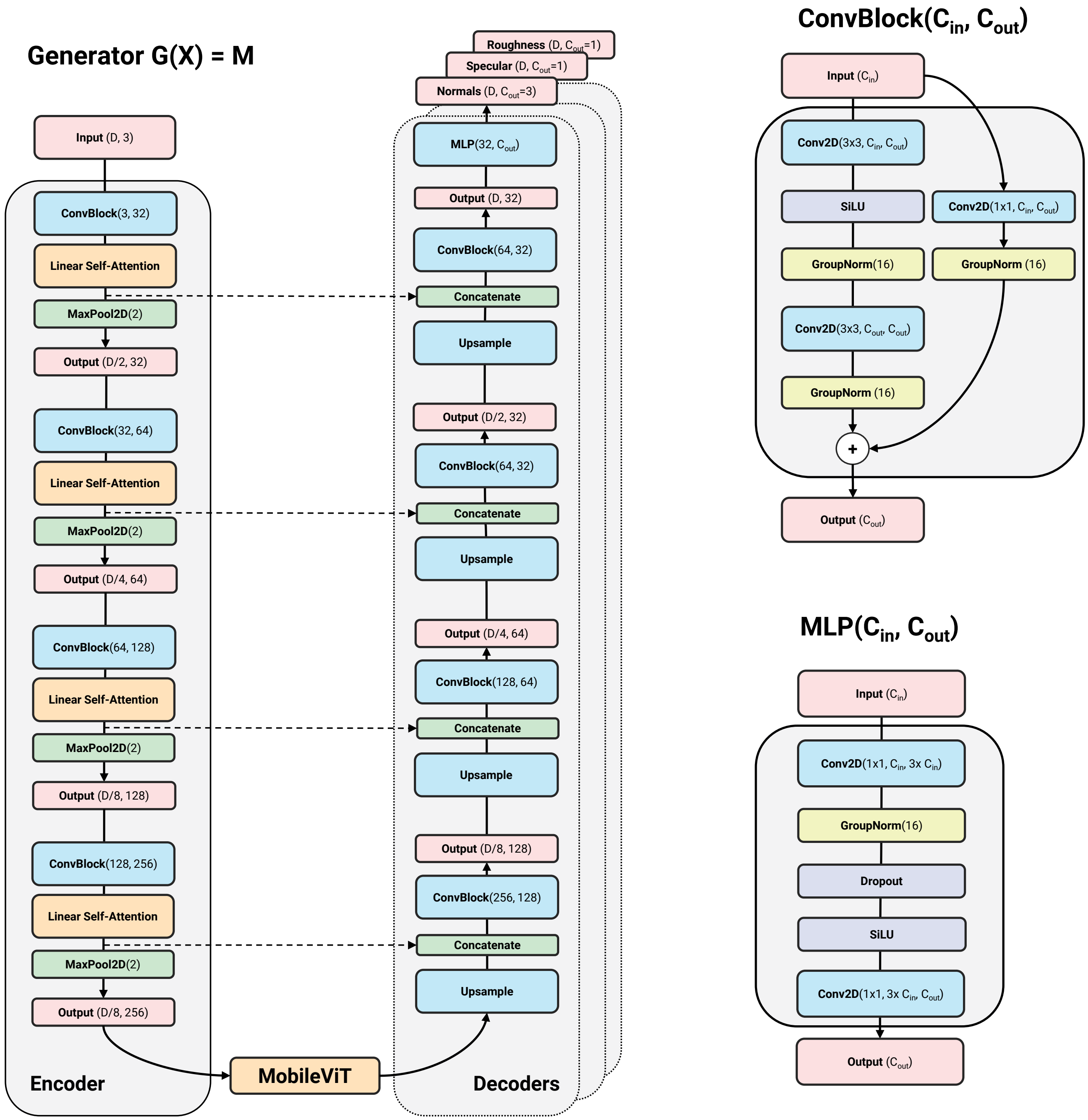

Figure 3. A full diagram of our generator, including layer sizes and output dimensions for each layer. For Self-Attention, we leverage Linear Attention [22], we use a MobileVIT transformer on the bottleneck [11], Group Normalization [26] and SiLU [4] non-linearities, one decoder per output map and residual connections in every convolutional block. In red, we show the input/output dimensions (spatial, channels) of each layer; in orange, we show attention modules; in blue, convolutional blocks and layers; in green, upsampling and concatenating operations; in yellow, normalization layers; and in purple, regularizations and non-linearities.

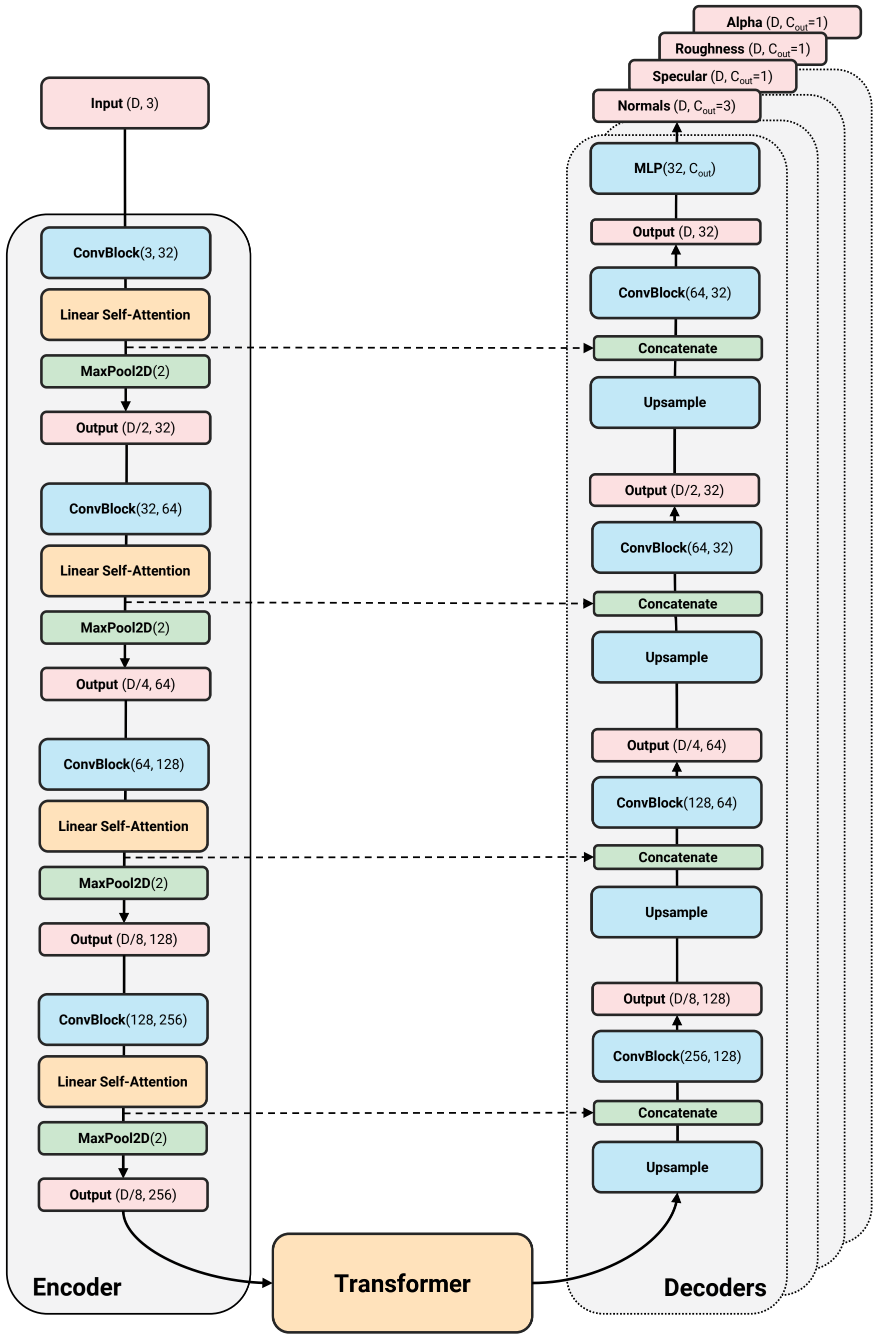

Figure 4. A full diagram of our U-Net residual discriminator, including layer sizes and output dimensions for each layer. We use a CBAM [24] module on the bottleneck and Spectral Normalization [13] throughout the network and residual connections in every convolutional block. In red, we show the input/output dimensions of each layer; in orange, we show attention modules; in blue, convolutional blocks and layers; in green, upsampling and concatenating operations; in yellow, normalization layers; and in purple, non-linearities.

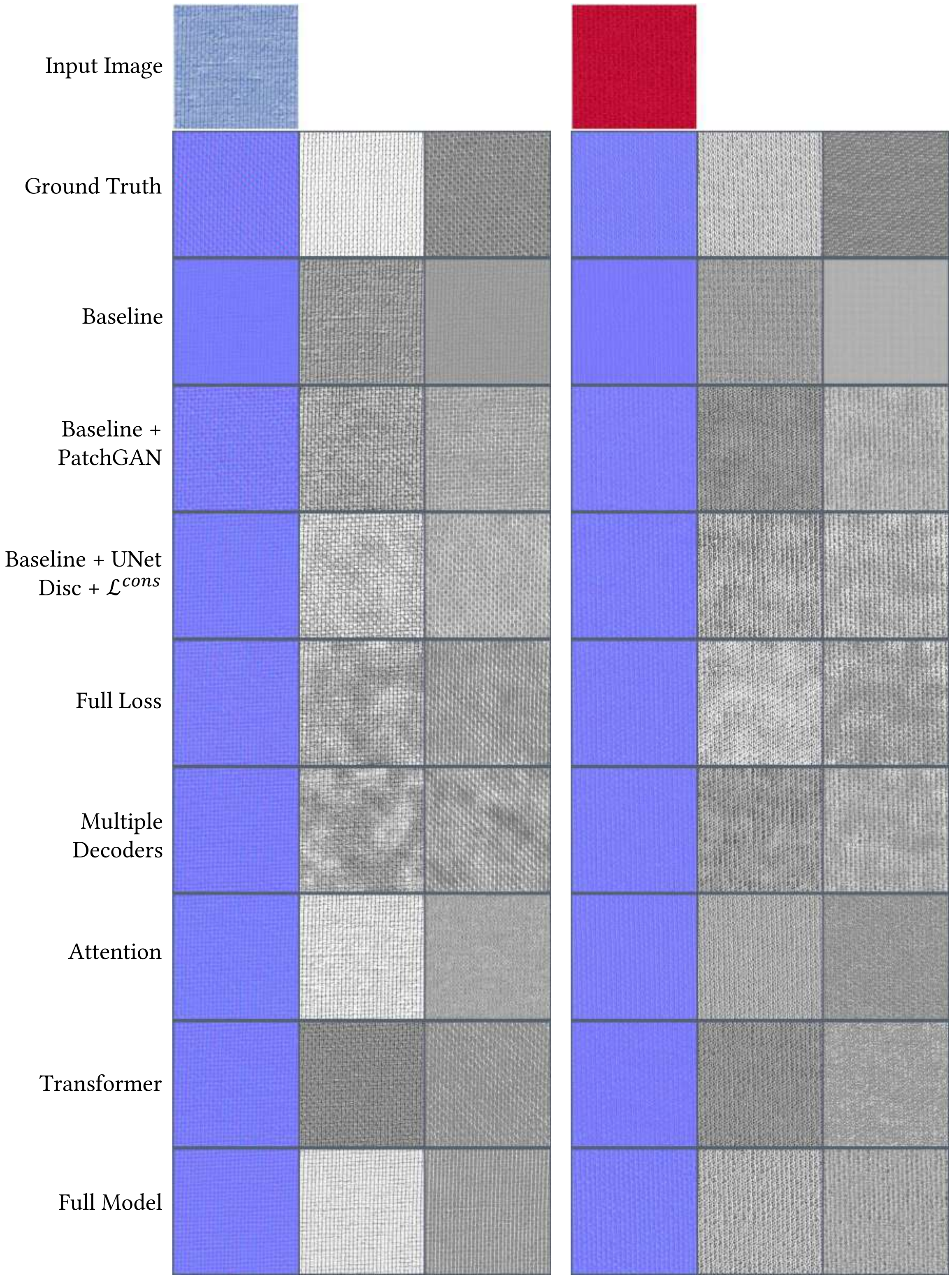

Figure 5. Further qualitative results of our ablation study. In order, normals, specular and roughness maps. The attention module removes artifacts.

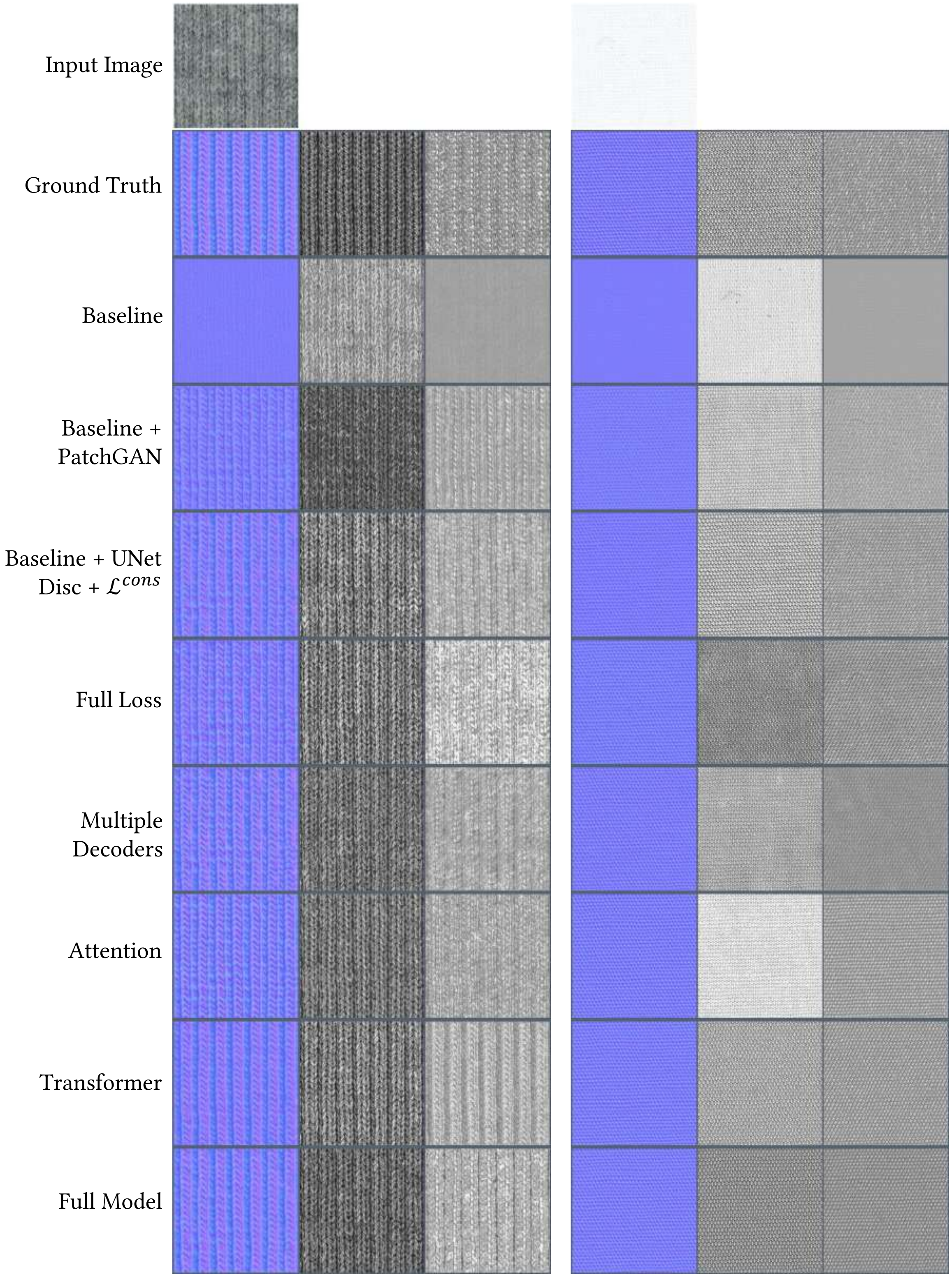

Figure 6. Further qualitative results of our ablation study. In order, normals, specular and roughness maps. The transformer module enhances the normal map for the highly-structured rib fabric on the left.

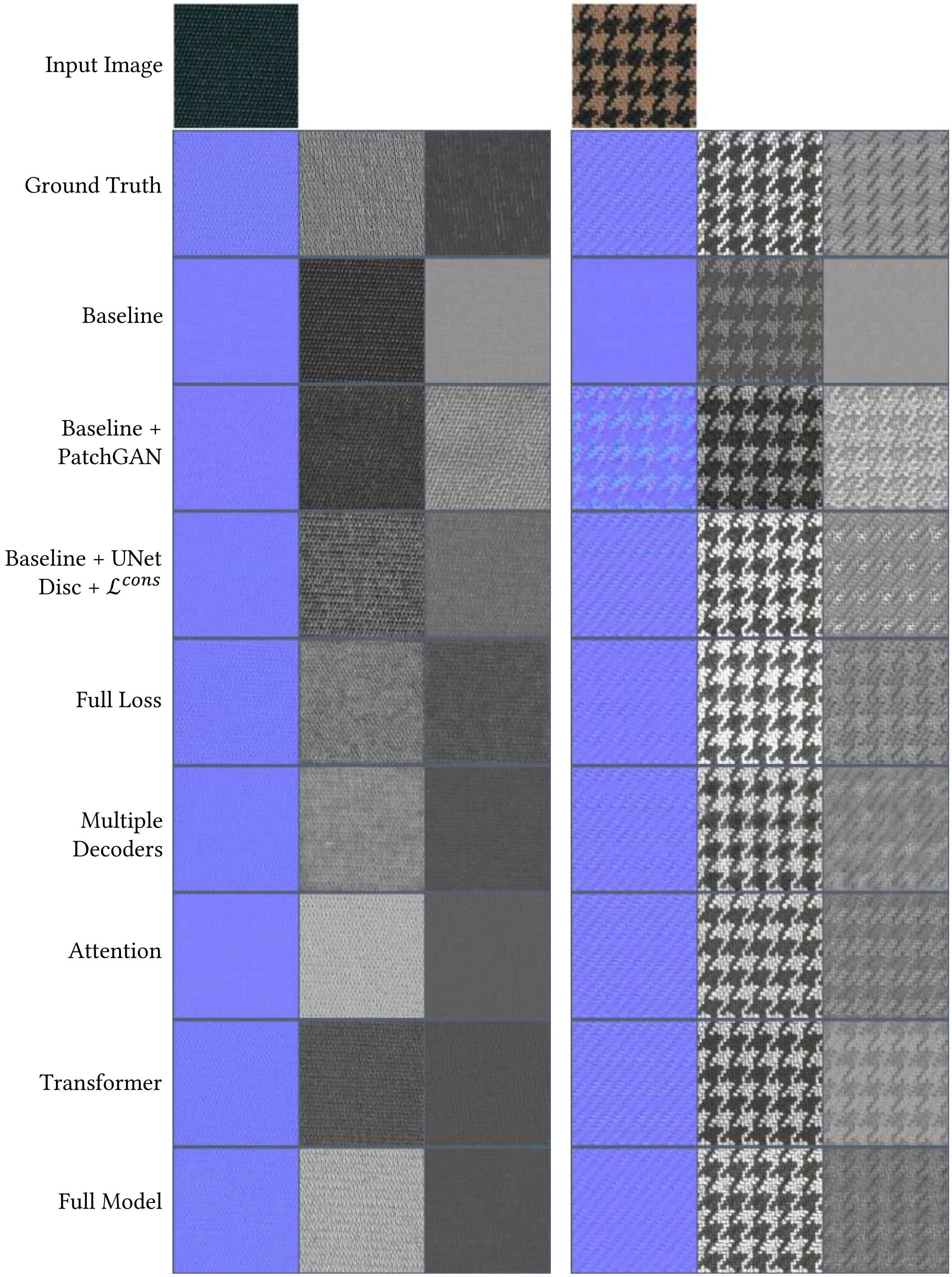

Figure 7. Further qualitative results of our ablation study. In order, normals, specular and roughness maps. The full model achieves the most accurate and sharper results.

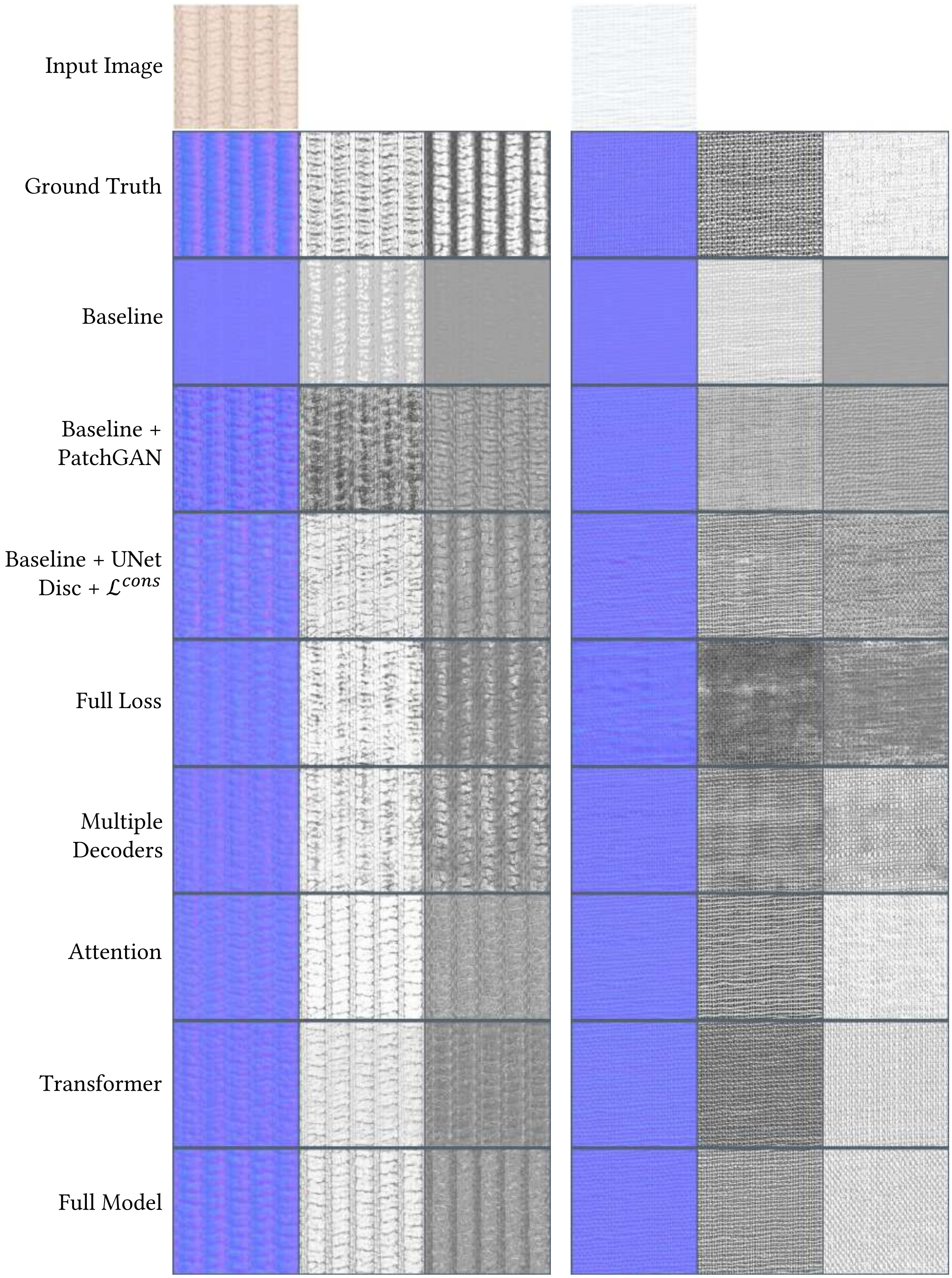

Figure 8. Further qualitative results of our ablation study. In order, normals, specular and roughness maps. The transformer module and the full model enhance the normal map for the highly-structured curdoroy fabric on the left.

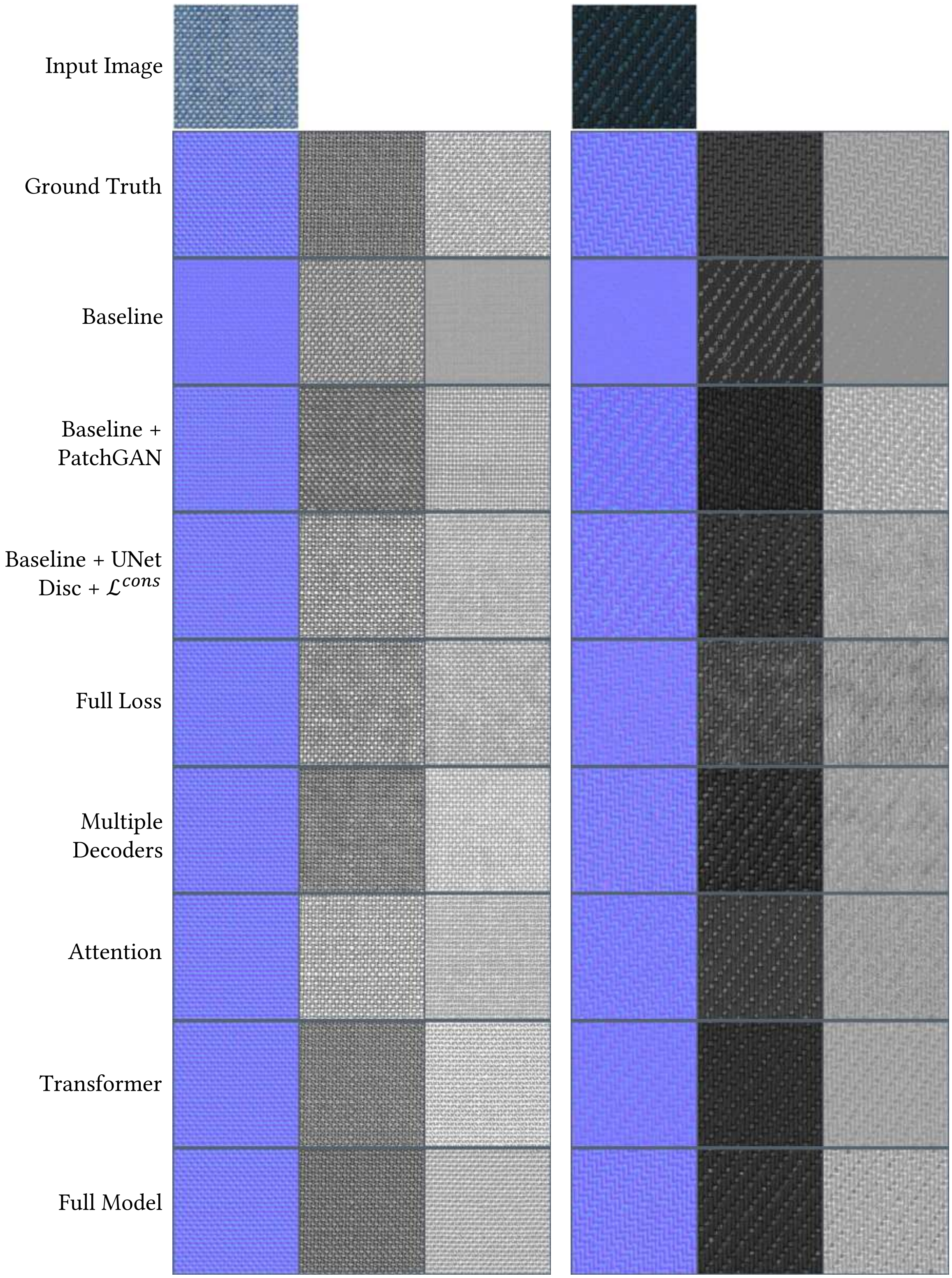

Figure 9. Further qualitative results of our ablation study.In order, normals, specular and roughness maps.

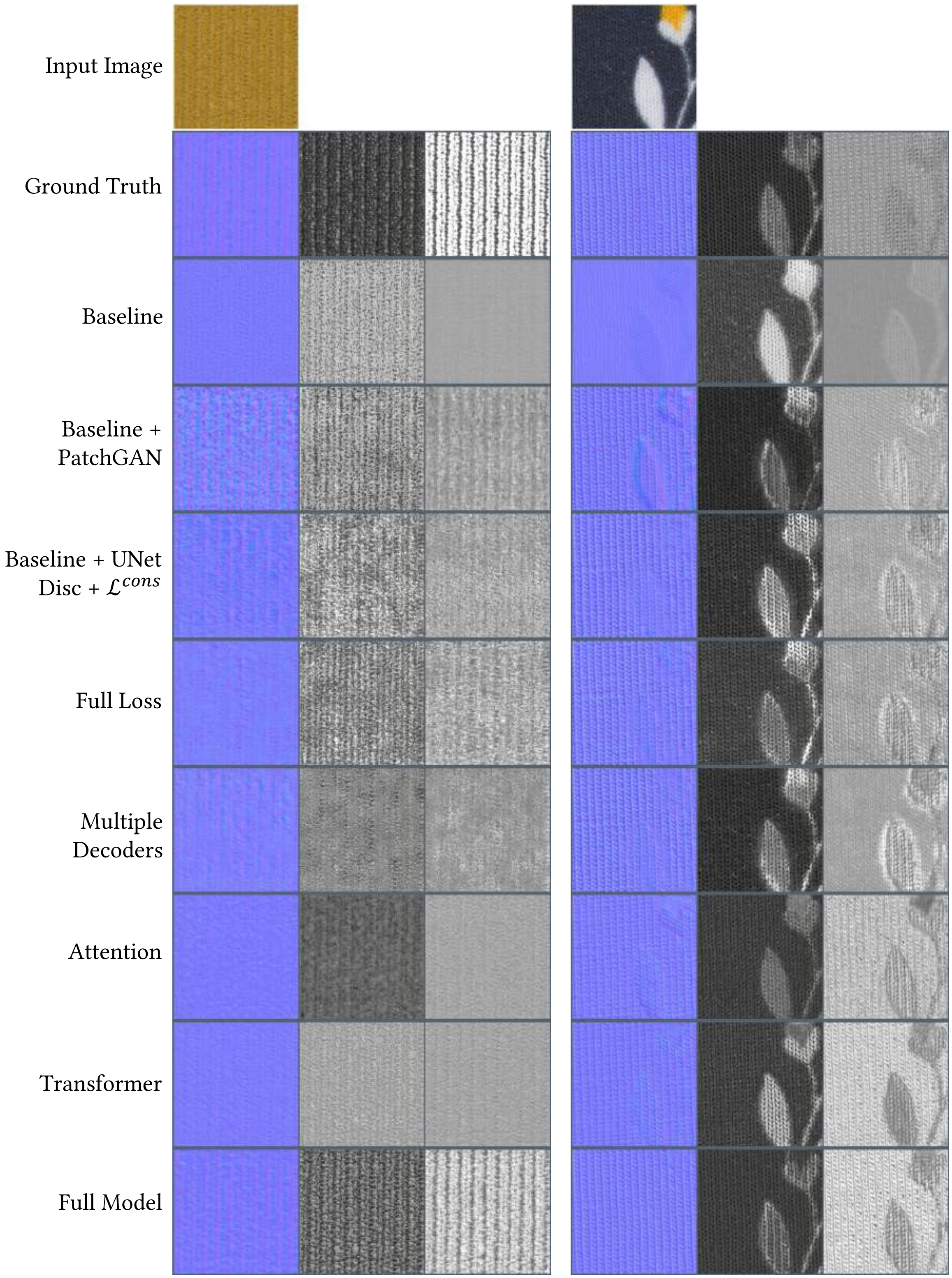

Figure 10. Further qualitative results of our ablation study. In order, normals, specular and roughness maps. The transformer module and the full model enhance the normal map for the highly-structured curdoroy fabric on the left and the printed knit fabric on the right.

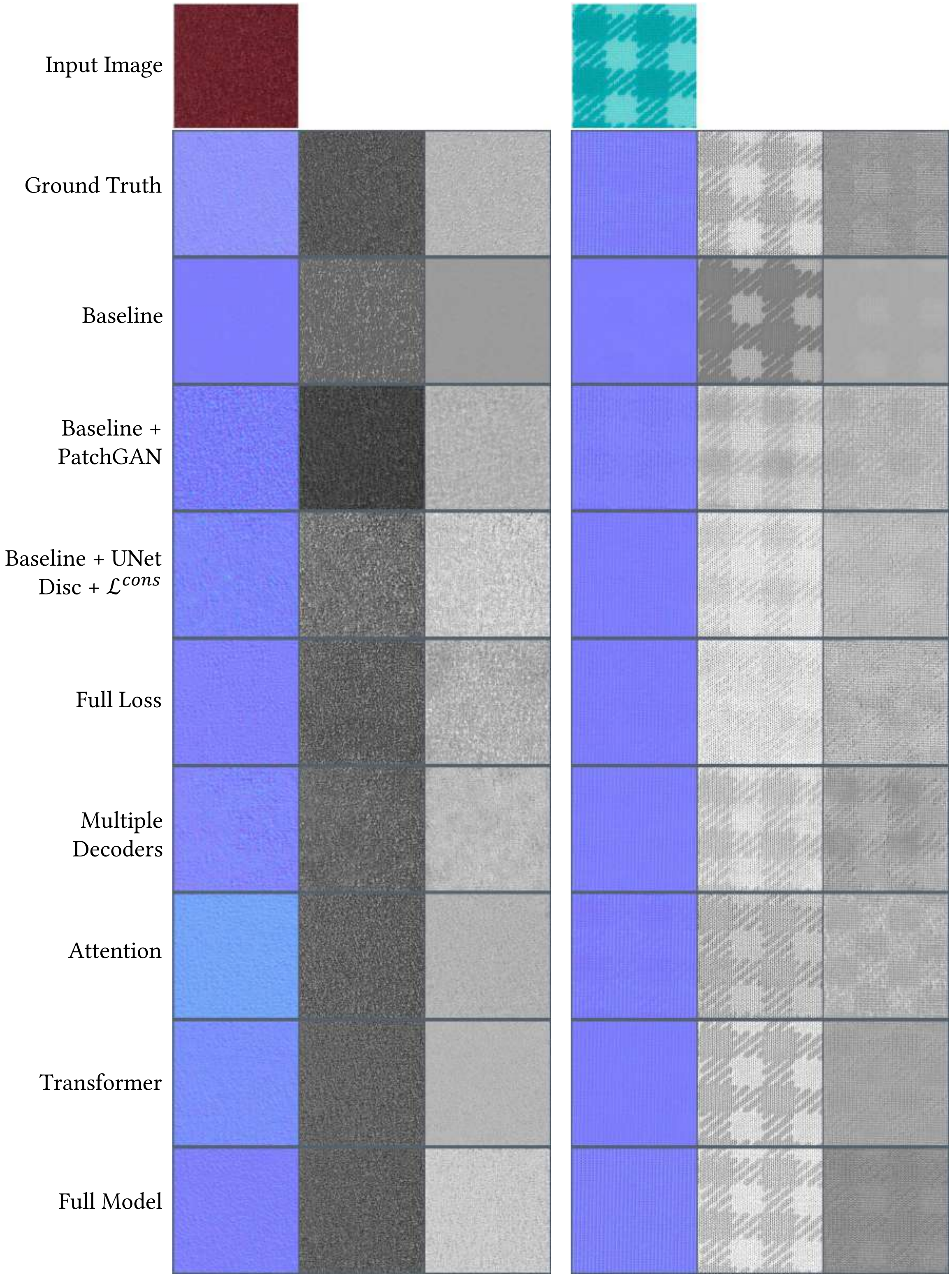

Figure 11. Further qualitative results of our ablation study, for a suede leather on the left and a printed knit on the right. In order, normals, specular and roughness maps.

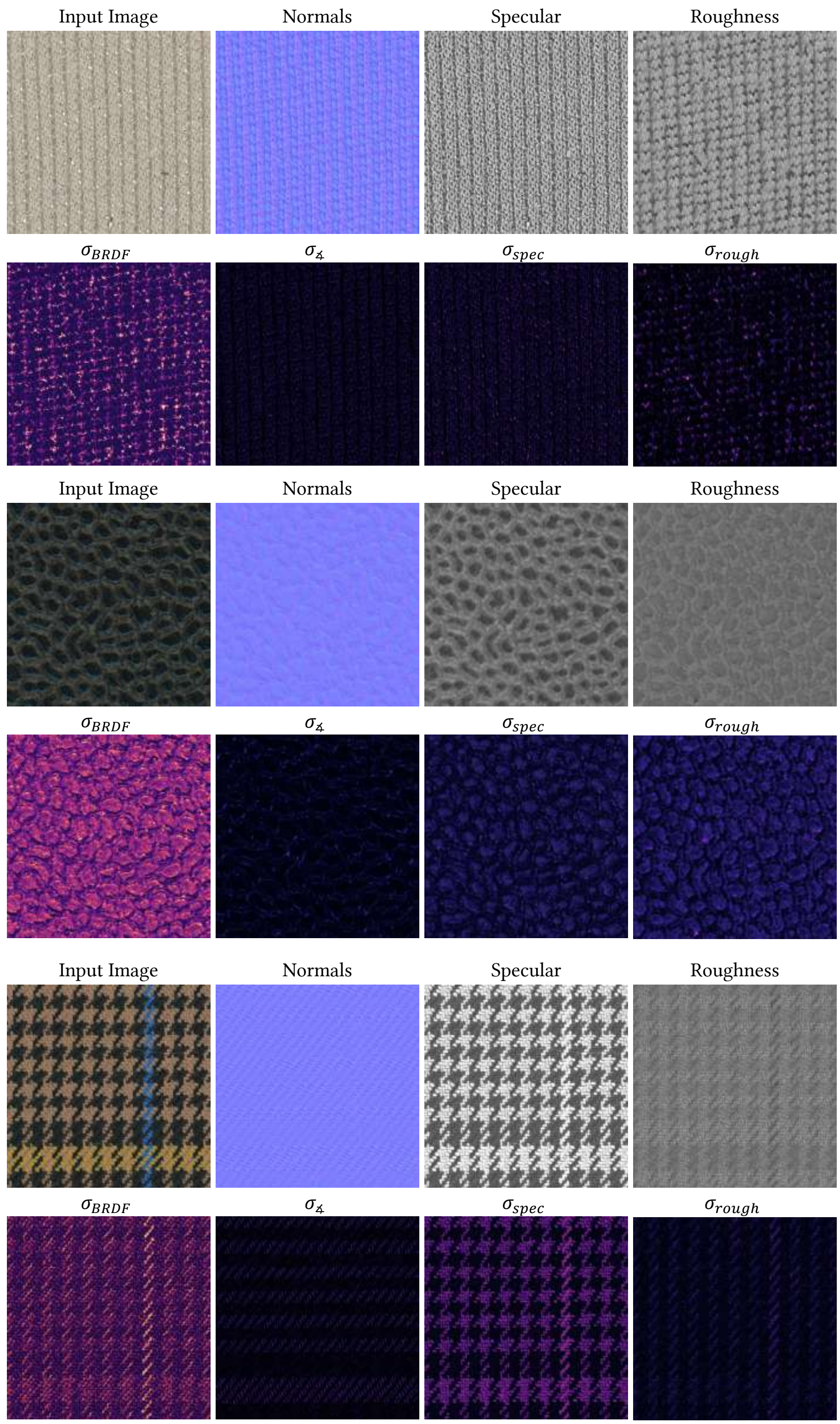

Figure 12. Additional results of our uncertainty estimation method. On the top, we show a beige *rib* fabric with gray metallic yarns. While there is a low uncertainty for the beige yarns, the metallic yarns are harder to digitize for our model (we do not support metalness in our material model) and it shows a higher uncertainty on those yarns. In the middle, we show a leather material with a very strong structural pattern. Our model shows very low confidence for this material. In the bottom, we show a tartan fabric. Interestingly, our model shows a higher uncertainty on the blue yarns, which are less common than the other yarns in the material.

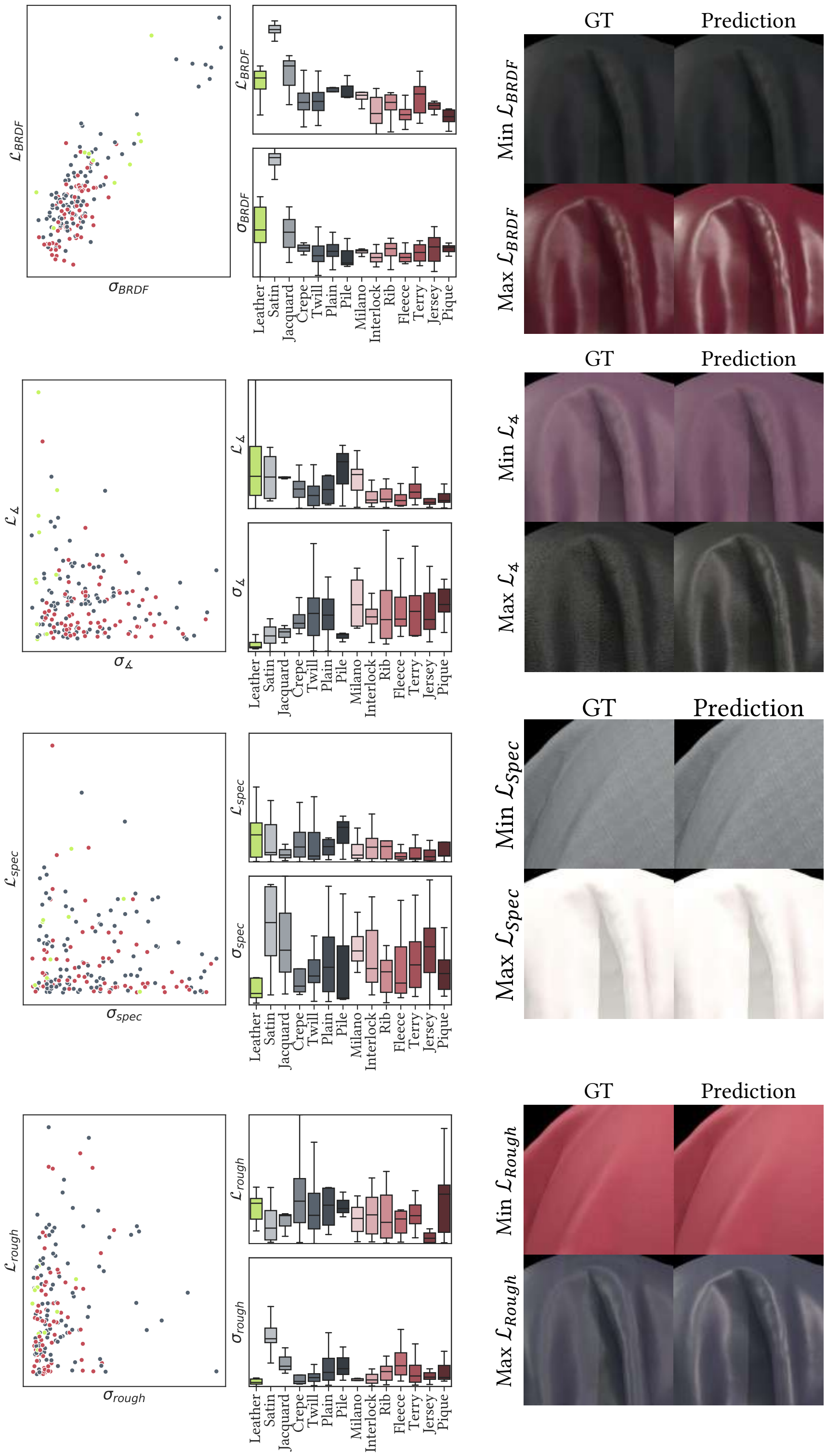

Figure 13. Additional results of our uncertainty metric. On the top row, we show a plot between our proposed render uncertainty σ_{BRDF} and the render error, which are highly correlated. We also show average error and uncertainty per family, as well as renders with the lowest and highest errors, compared to the ground truth. Below, we show the same data for normals, specular and roughness errors and uncertainties. There are no correlations between uncertainties and errors for these maps.

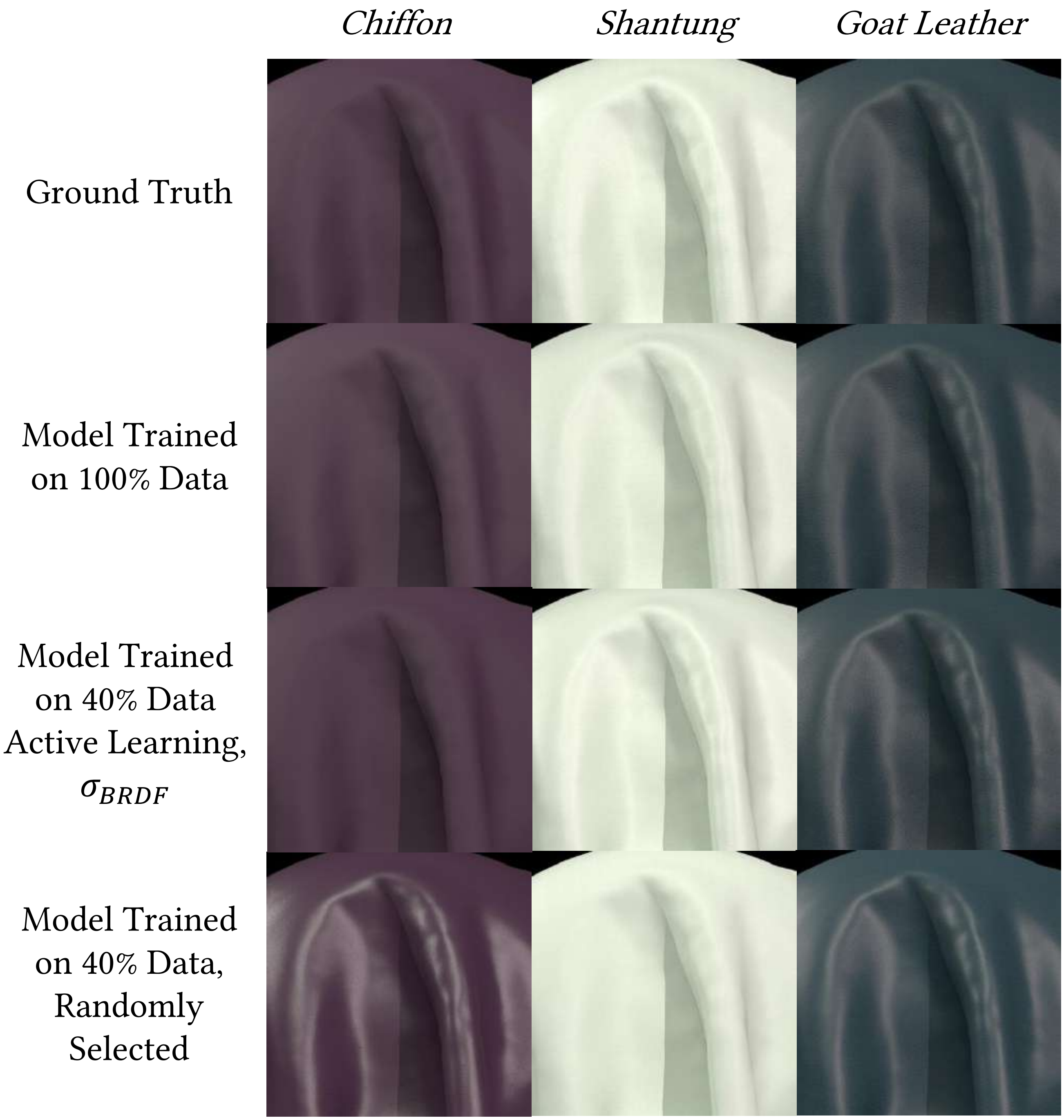

Figure 14. Additional results of our active learning experiment. From top to bottom, we show the renders of ground truth SVBRDFs, the model trained on the 100% of the training data, a model trained on 40% of the training data available, selected following an active learning approach using our uncertainty σ_{BRDF} as guidance, and a model trained on 40% data, selected randomly. From left to right, we show a very diffuse *Chiffon* fabric, a highly specular *Shantung* fabric and a *Goat Leather* material with very varied microgeometry. In every case, our model trained on 40% of the data following an active learning approach obtains results which are very similar to a model trained on 100% of the data. The model trained on randomly selected 40% data produces highly innacurate specularity and microgeometry estimations.

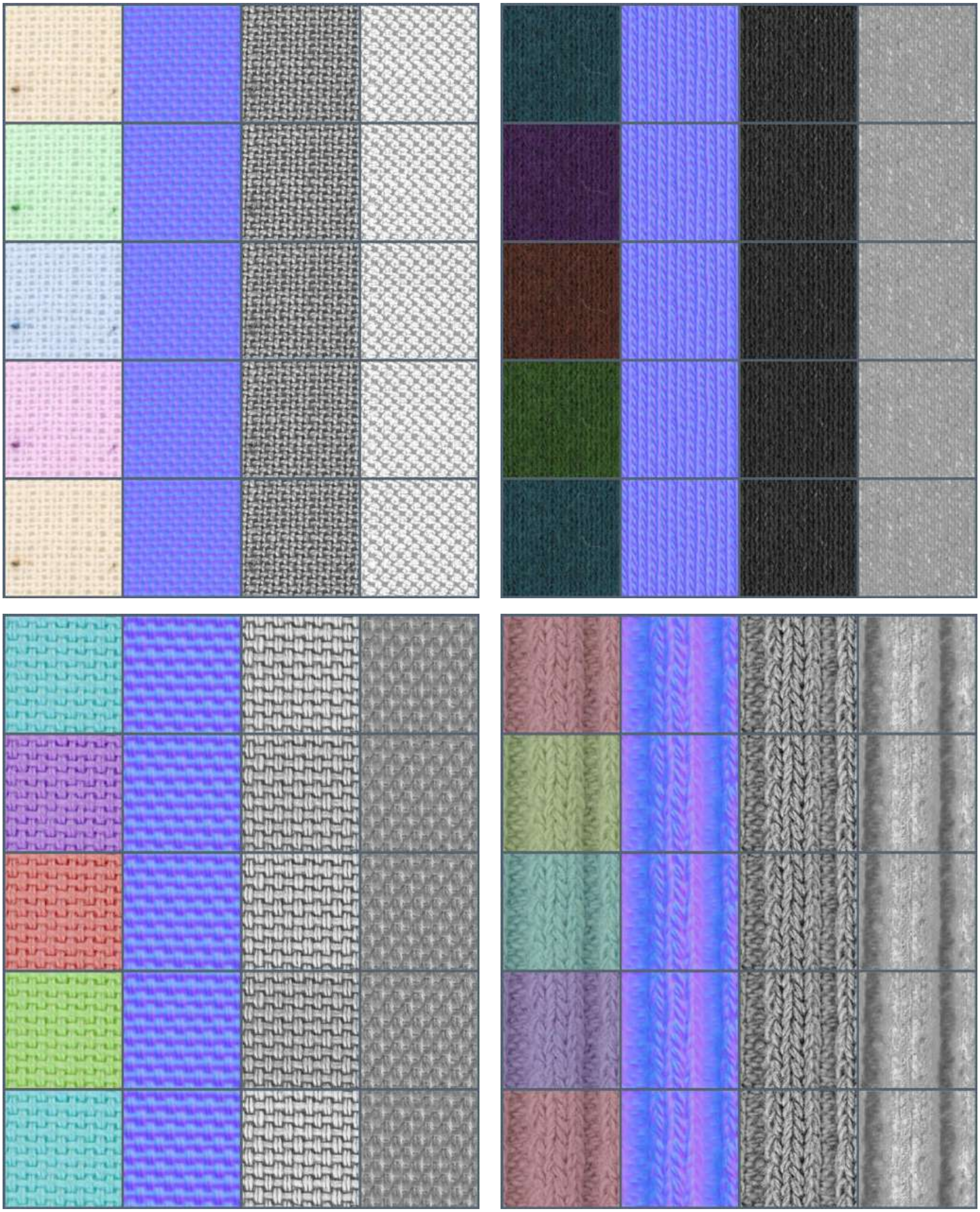

Figure 15. Robustness of our model with respect to hue changes applied to the input images. On the left, we show the input images, on their right, the three estimated maps (normals, specular, roughness).

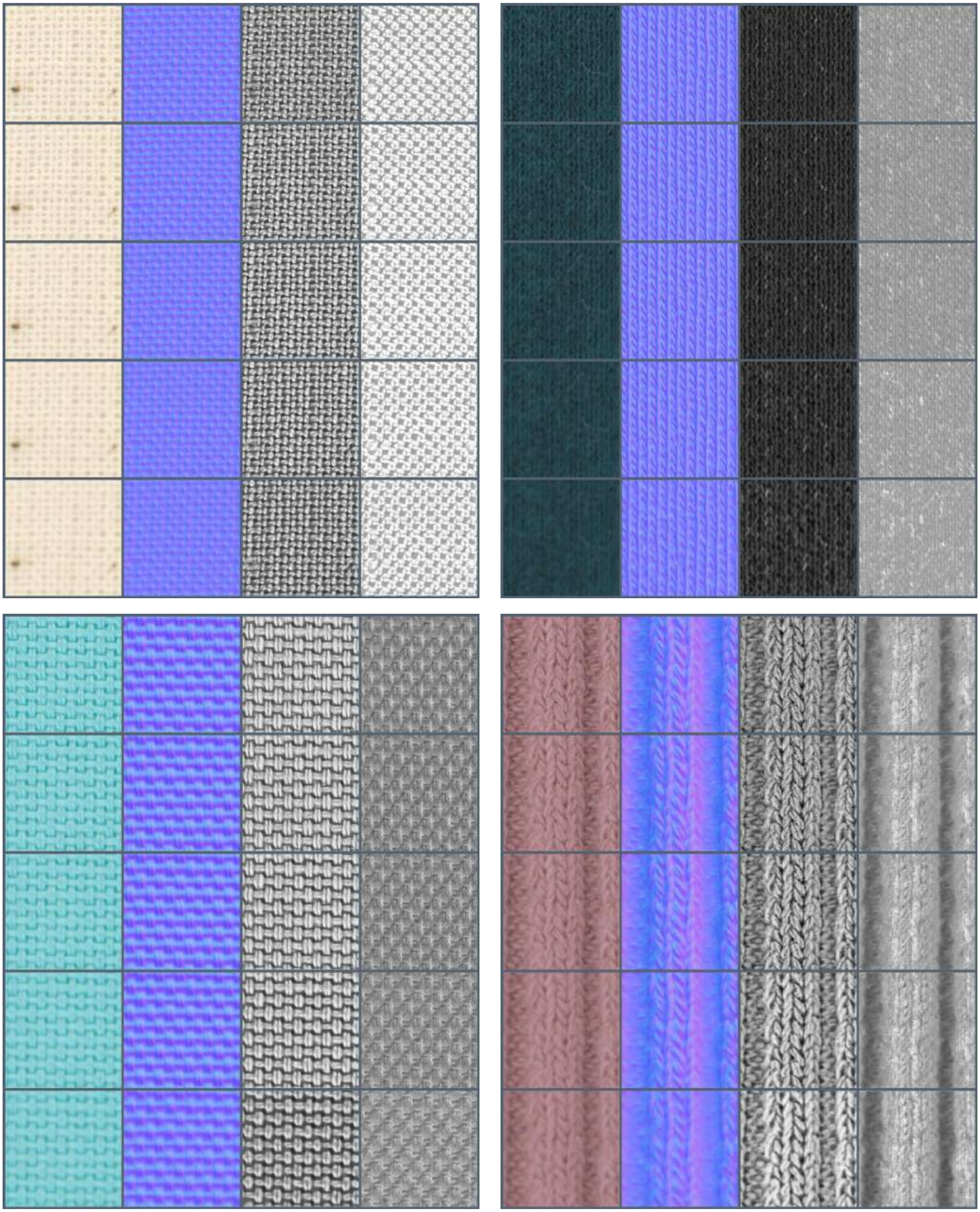

Figure 16. Robustness of our model with respect to Gaussian blur applied to the input images. On the left, we show the input images, on their right, the three estimated maps (normals, specular, roughness).

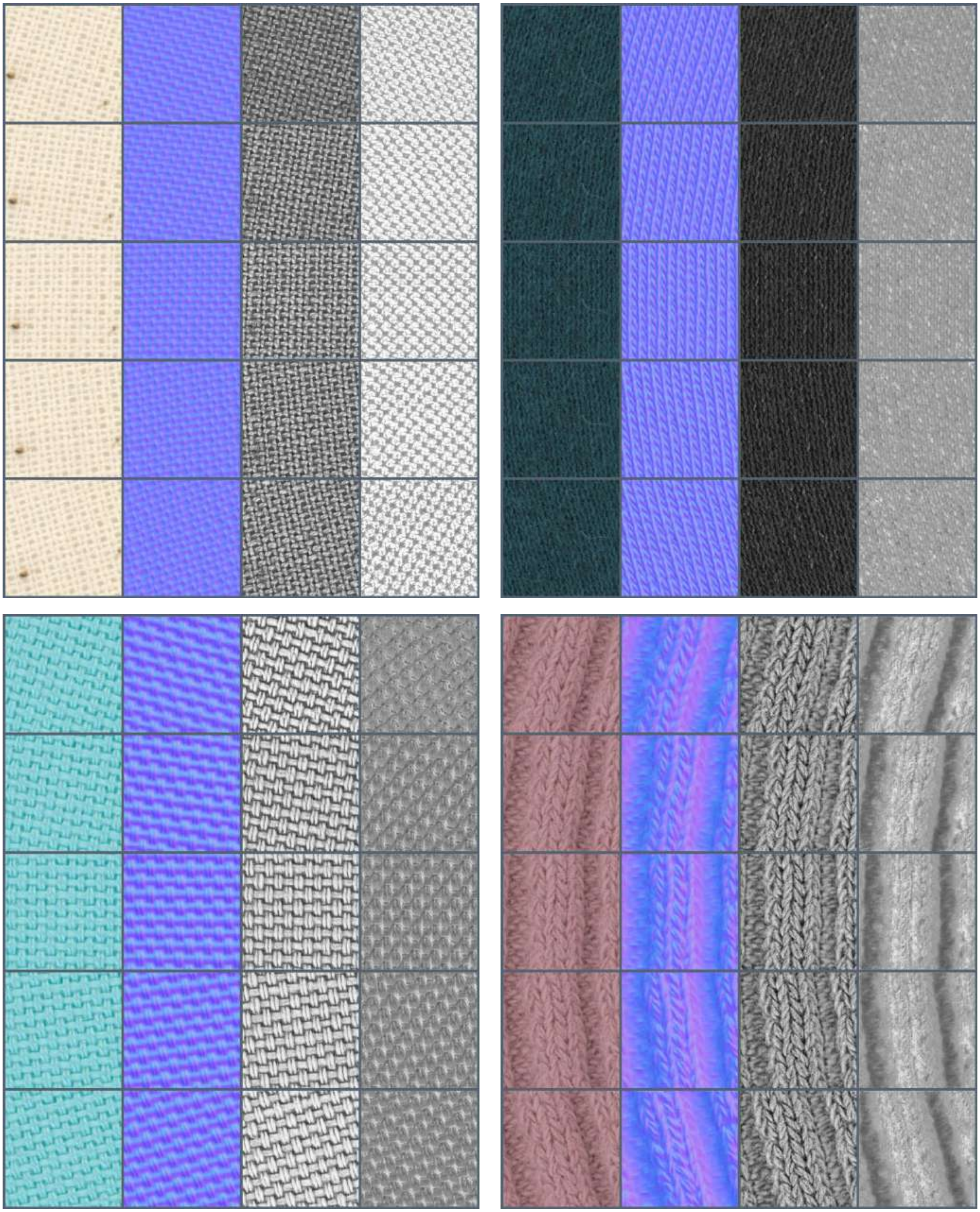

Figure 17. Robustness of our model with respect to rotations applied to the input images. On the left, we show the input images, on their right, the three estimated maps (normals, specular, roughness)

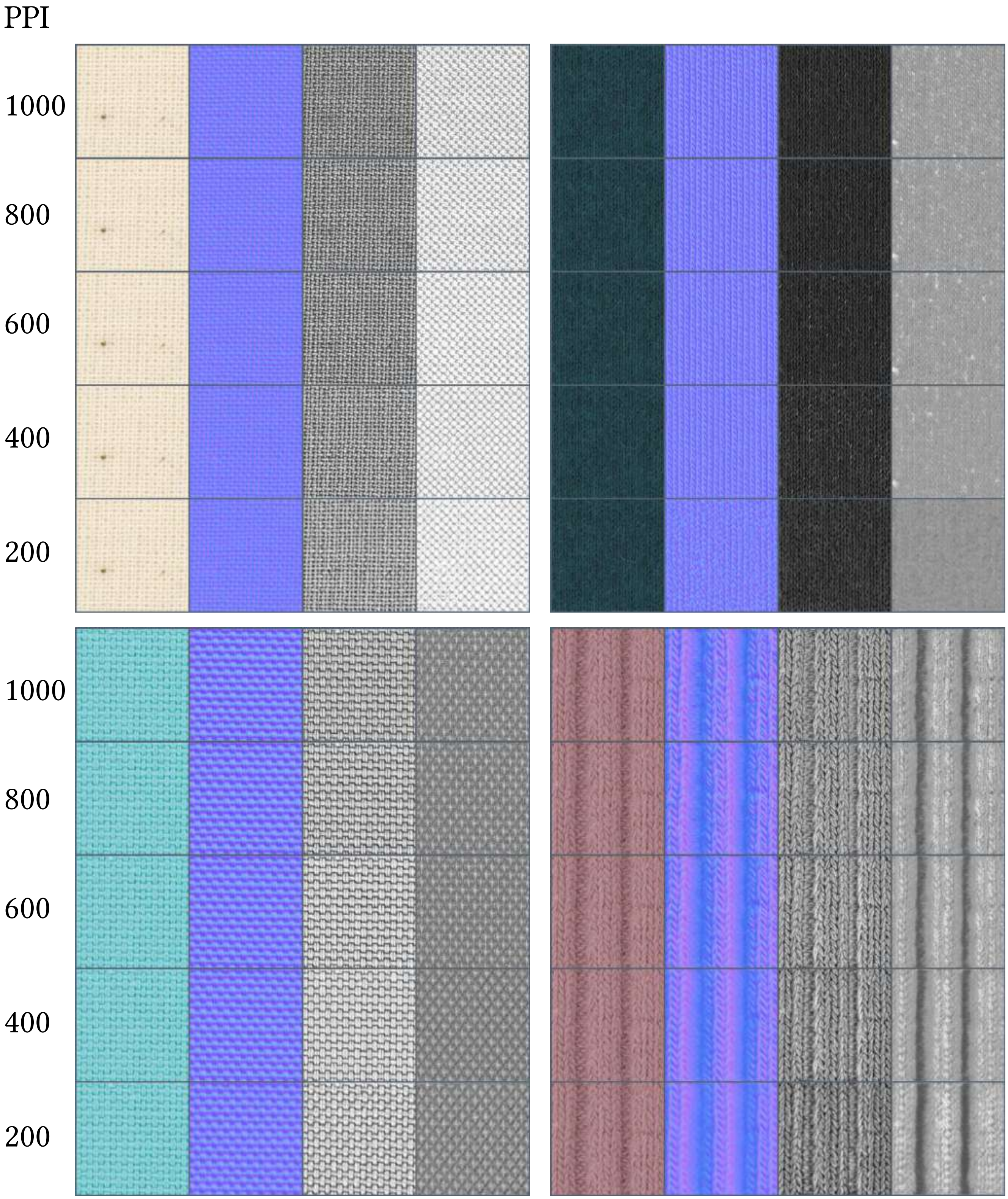

Figure 18. Robustness of our model with respect to rescales changes applied to the input images, for different PPI. On the left, we show the input images, on their right, the three estimated maps (normals, specular, roughness). For the downsampled images (< 1000 PPI), we upsample them using bilinear interpolation to PPI to make the results comparible.

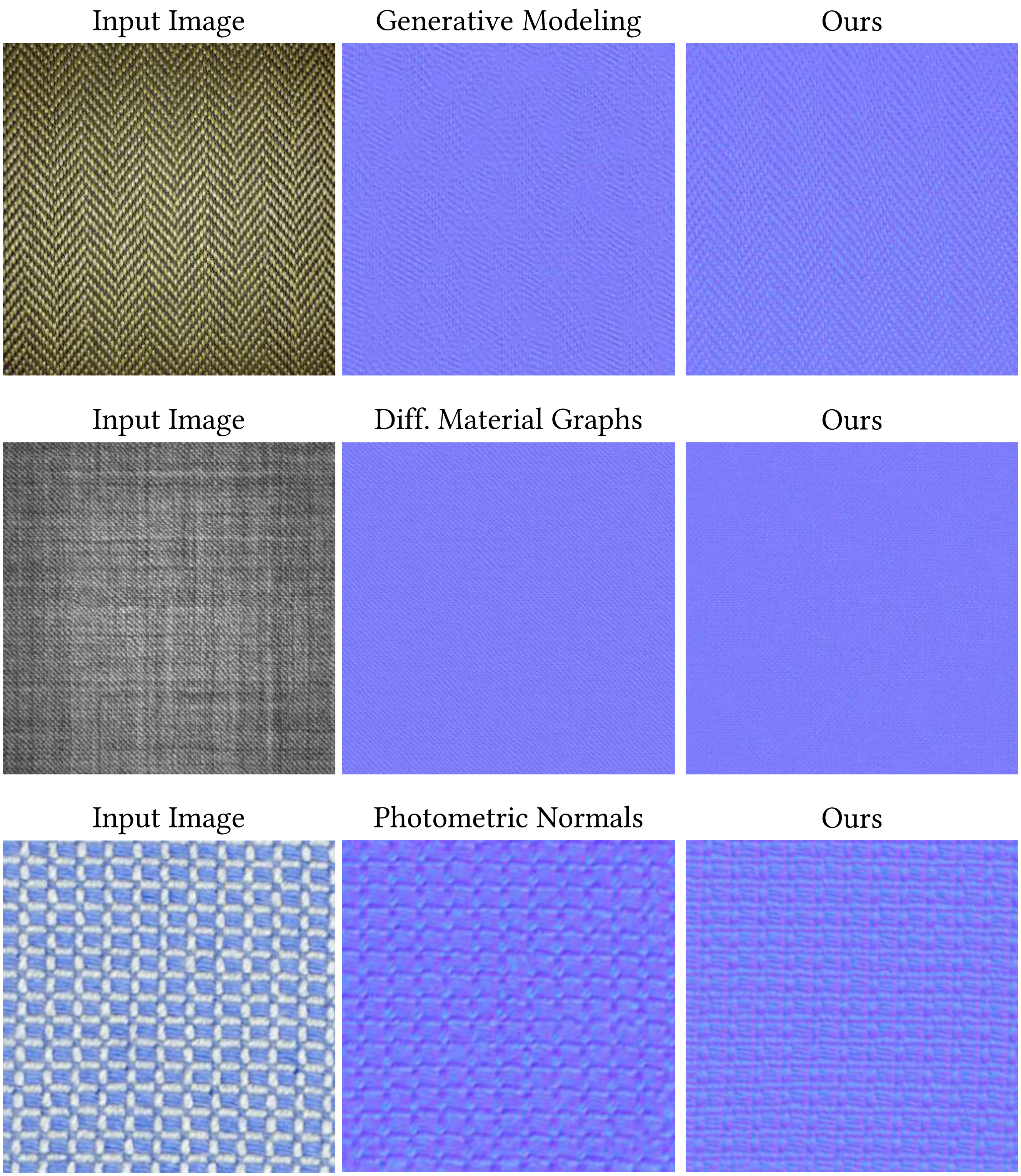

Figure 19. Results of our method on datasets of previous work. On the top, we show the results of [8] on their own test set. Our method provides sharper normals which better preserve the structure of the input images. On the middle, we show the results of our method on synthetic rendered data from a material graph from [20]. Our method provides sharp normals for this synthetic image, which lies outside the distribution of our dataset, composed exclusively of real images. On the bottom, we show an albedo and normals computed using Photometric Stereo [25] for a captured BTF [23], for which our model also provides highly detailed results.

Input	Deep Inverse Rendering [5]	Generative Modeling [8]	Diff. Material Graphs [20]	Adversarial Estimation [28]	**Our Method**

Table 1. Comparisons of our results with previous work on images captured under different conditions, for a *tartan fabric*: On the first rows, images were captured with a smartphone using the flash image. On the middle rows, using the same smartphone with ambient lighting on different scales. On the final row, a scanner image. Note that for [20] we use a fabric material for initialization and use their metallic map instead of specular, that we do not estimate albedos and that the material models are not necessarily comparible.

Input	Deep Inverse Rendering [5]	Generative Modeling [8]	Diff. Material Graphs [20]	Adversarial Estimation [28]	**Our Method**

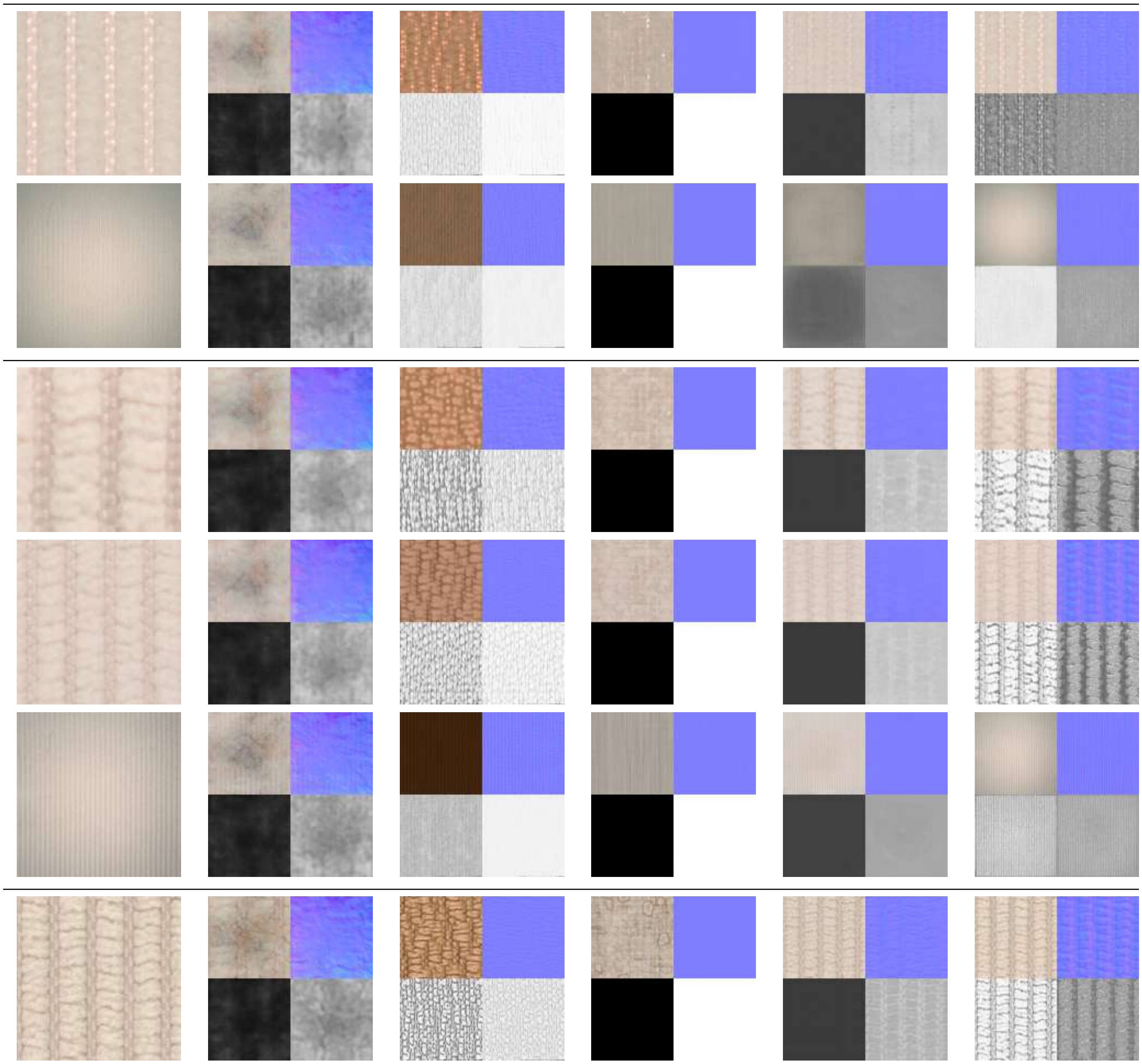

Table 2. Comparisons of our results with previous work on images captured under different conditions for a *curdoroy fabric*: On the first rows, images were captured with a smartphone using the flash image. On the middle rows, using the same smartphone with ambient lighting on different scales. On the final row, a scanner image. Note that for [20] we use a fabric material for initialization and use their metallic map instead of specular, that we do not estimate albedos and that the material models are not necessarily comparible.

Input	Deep Inverse Rendering [5]	Generative Modeling [8]	Diff. Material Graphs [20]	Adversarial Estimation [28]	**Our Method**

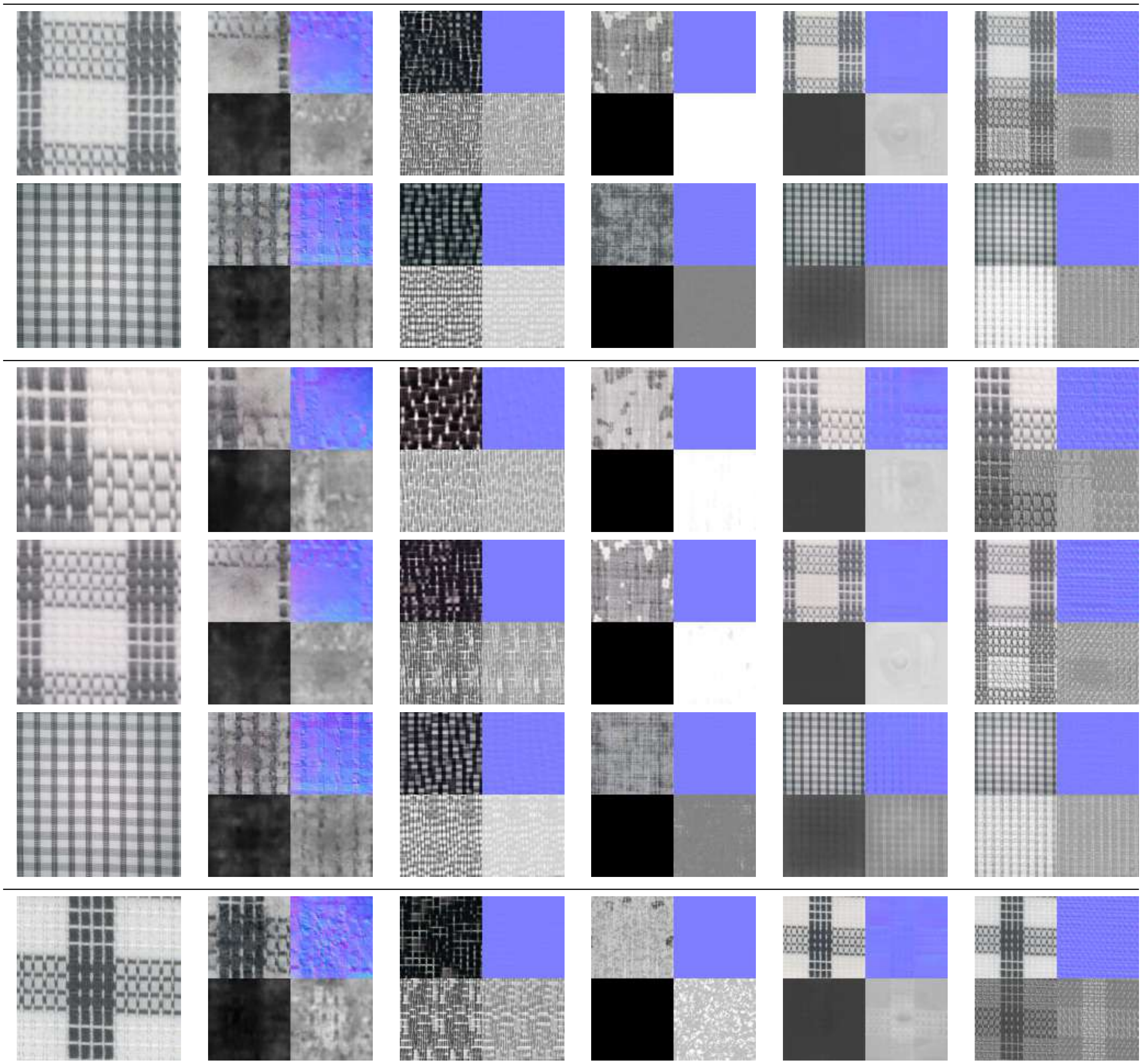

Table 3. Comparisons of our results with previous work on images captured under different conditions for a *plain weave fabric*: On the first rows, images were captured with a smartphone using the flash image. On the middle rows, using the same smartphone with ambient lighting on different scales. On the final row, a scanner image. Note that for [20] we use a fabric material for initialization and use their metallic map instead of specular, that we do not estimate albedos and that the material models are not necessarily comparible.

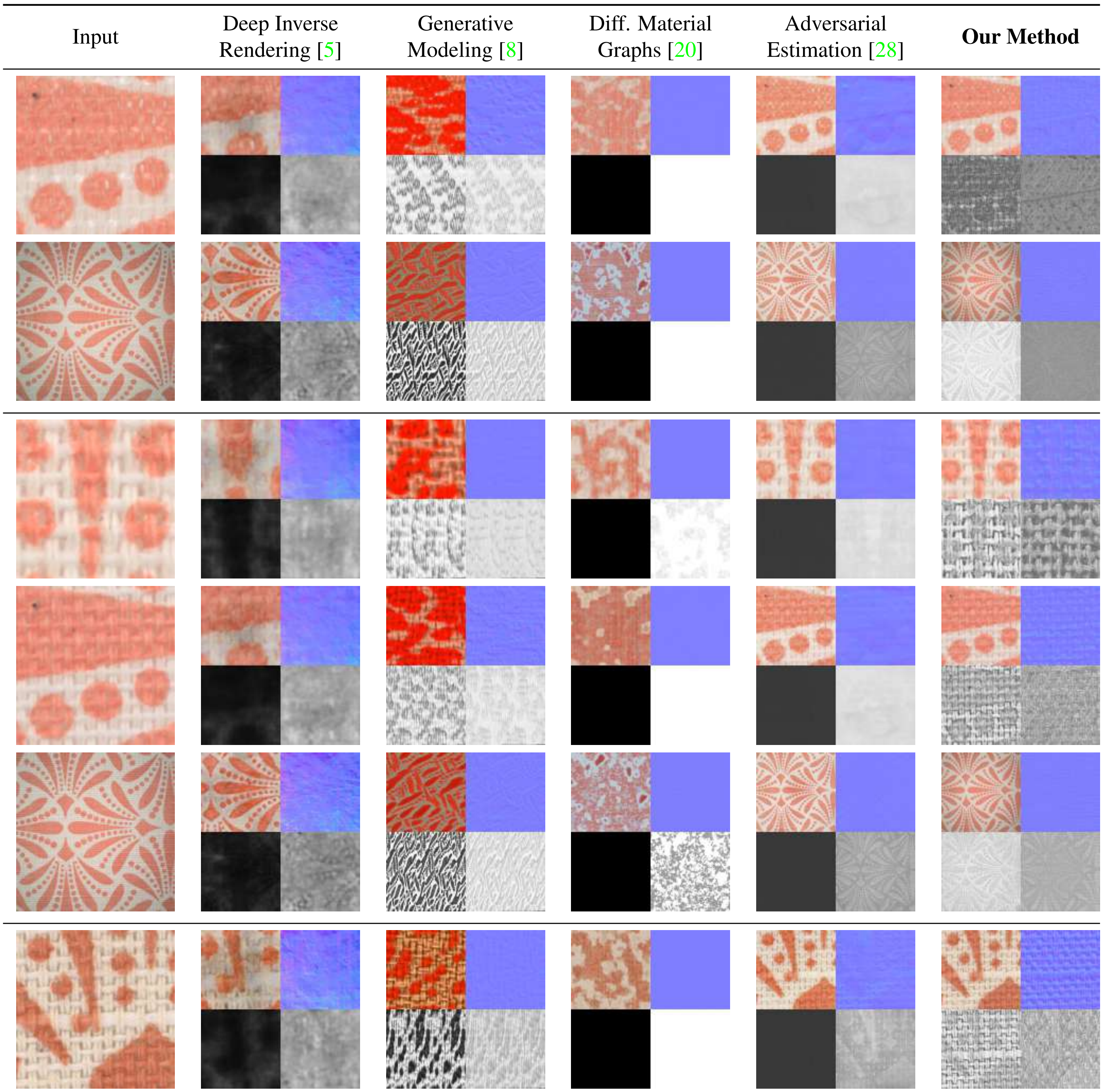

Table 4. Comparisons of our results with previous work on images captured under different conditions for a *plain weave fabric*: On the first rows, images were captured with a smartphone using the flash image. On the middle rows, using the same smartphone with ambient lighting on different scales. On the final row, a scanner image. Note that for [20] we use a fabric material for initialization and use their metallic map instead of specular, that we do not estimate albedos and that the material models are not necessarily comparible.

Input	Deep Inverse Rendering [5]	Generative Modeling [8]	Diff. Material Graphs [20]	Adversarial Estimation [28]	**Our Method**

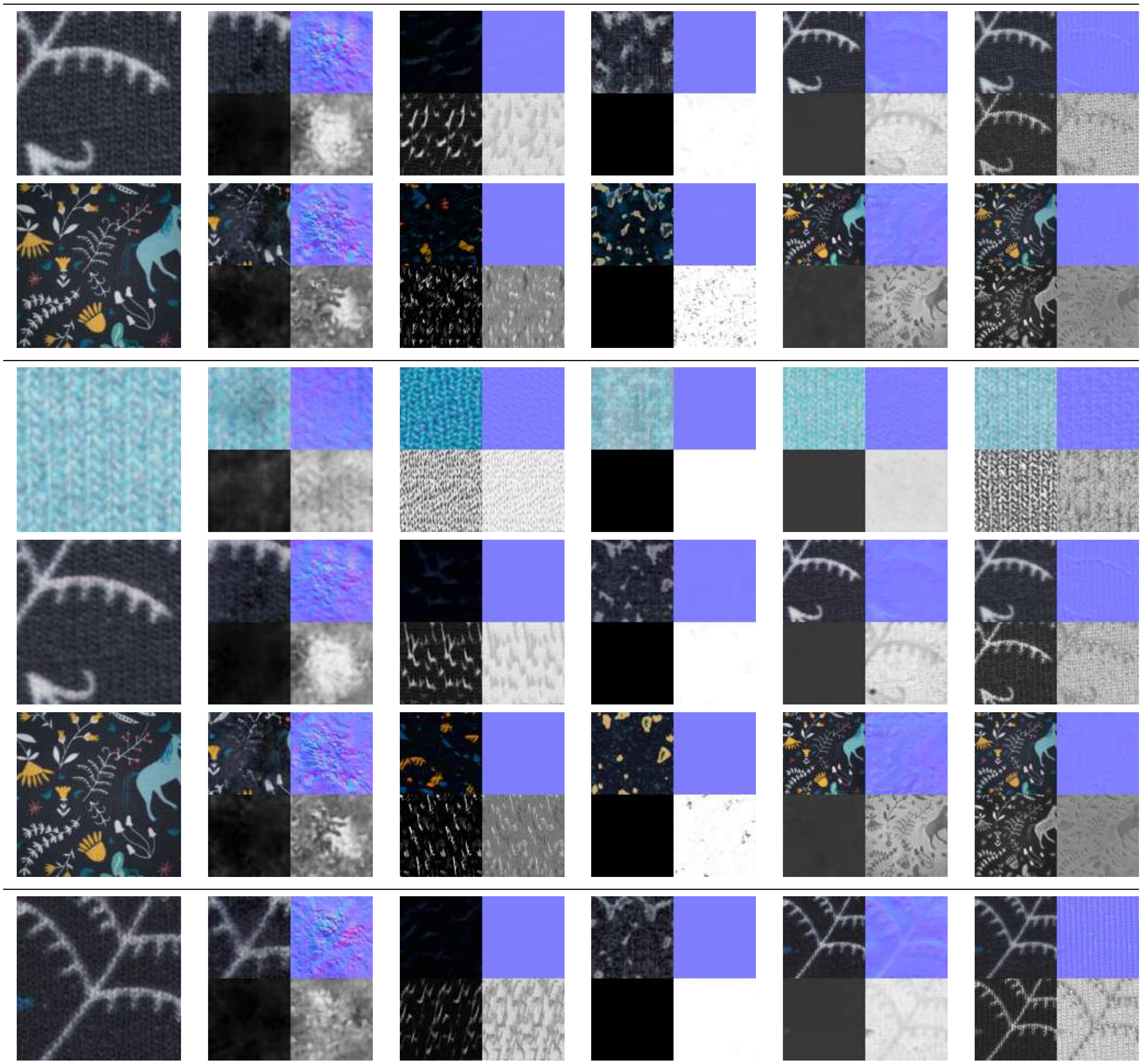

Table 5. Comparisons of our results with previous work on images captured under different conditions for a *single jersey fabric*: On the first rows, images were captured with a smartphone using the flash image. On the middle rows, using the same smartphone with ambient lighting on different scales. On the final row, a scanner image. Note that for [20] we use a fabric material for initialization and use their metallic map instead of specular, that we do not estimate albedos and that the material models are not necessarily comparible.

Input	Deep Inverse Rendering [5]	Generative Modeling [8]	Diff. Material Graphs [20]	Adversarial Estimation [28]	**Our Method**

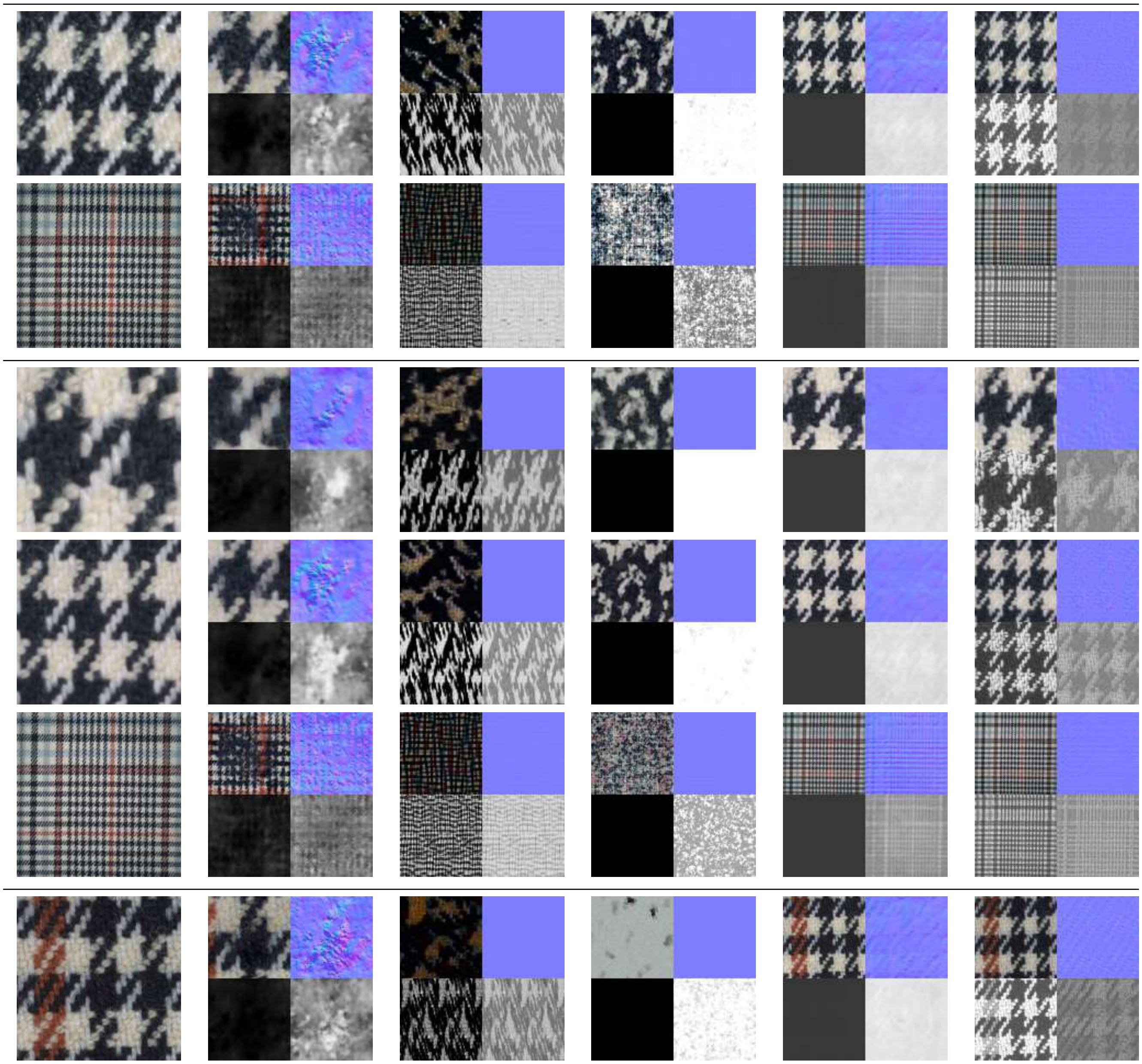

Table 6. Comparisons of our results with previous work on images captured under different conditions for a *houndstooth fabric*: On the first rows, images were captured with a smartphone using the flash image. On the middle rows, using the same smartphone with ambient lighting on different scales. On the final row, a scanner image. Note that for [20] we use a fabric material for initialization and use their metallic map instead of specular, that we do not estimate albedos and that the material models are not necessarily comparible.

Input	Deep Inverse Rendering [5]	Generative Modeling [8]	Diff. Material Graphs [20]	Adversarial Estimation [28]	**Our Method**

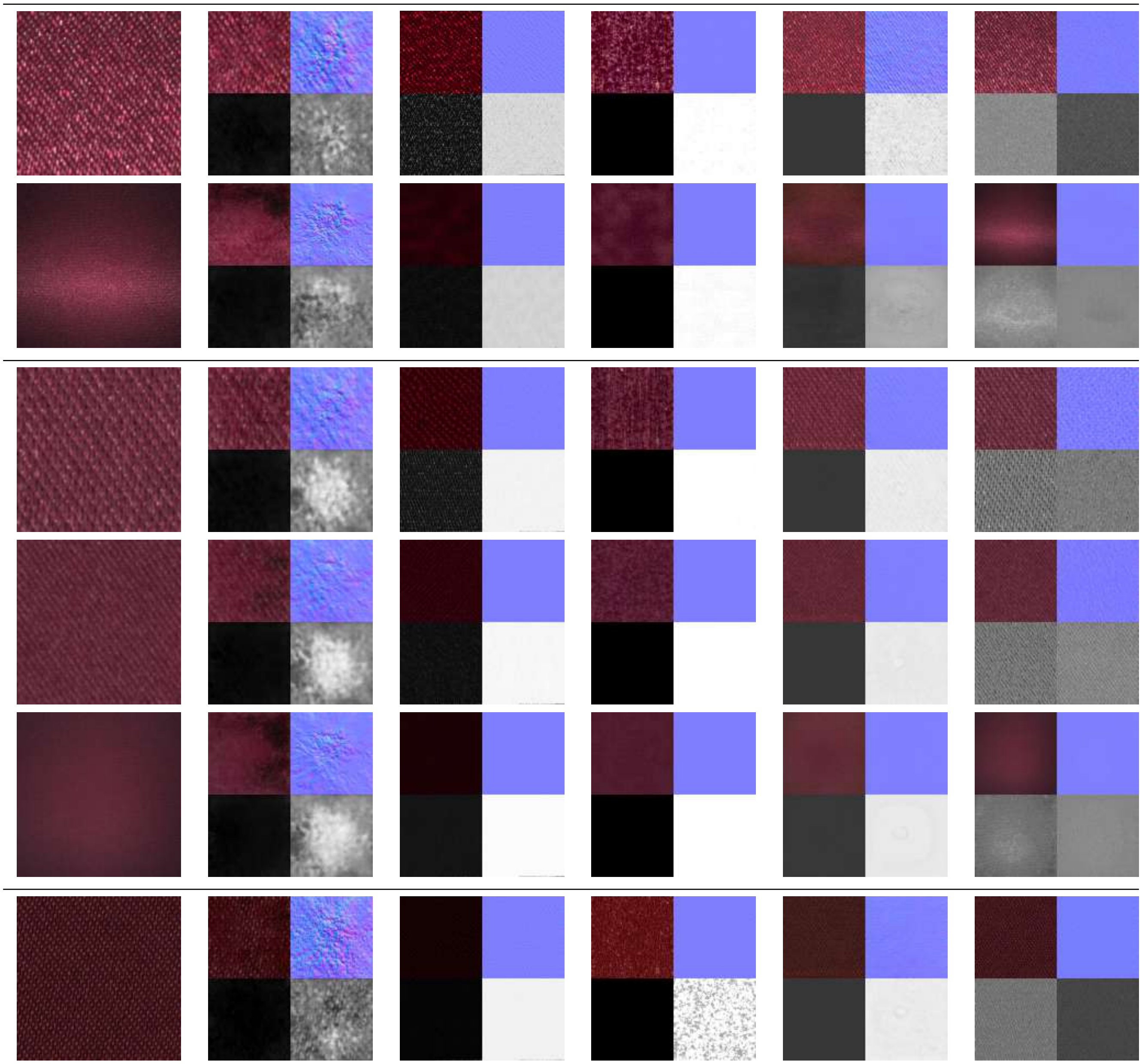

Table 7. Comparisons of our results with previous work on images captured under different conditions for a *satin fabric*: On the first rows, images were captured with a smartphone using the flash image. On the middle rows, using the same smartphone with ambient lighting on different scales. On the final row, a scanner image. Note that for [20] we use a fabric material for initialization and use their metallic map instead of specular, that we do not estimate albedos and that the material models are not necessarily comparible.

Input	Deep Inverse Rendering [5]	Generative Modeling [8]	Diff. Material Graphs [20]	Adversarial Estimation [28]	**Our Method**

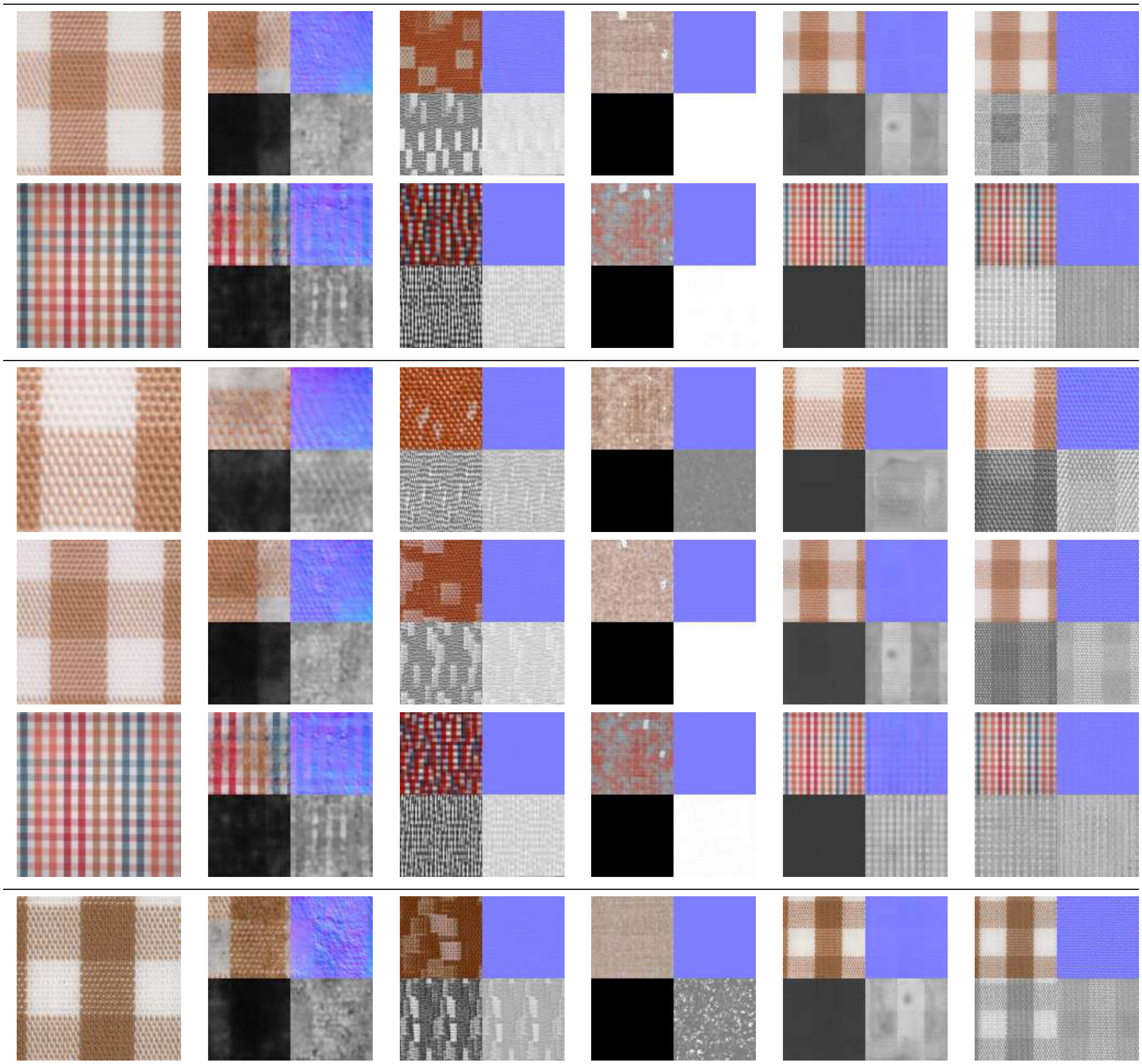

Table 8. Comparisons of our results with previous work on images captured under different conditions for a *plaid fabric*: On the first rows, images were captured with a smartphone using the flash image. On the middle rows, using the same smartphone with ambient lighting on different scales. On the final row, a scanner image. Note that for [20] we use a fabric material for initialization and use their metallic map instead of specular, that we do not estimate albedos and that the material models are not necessarily comparible.

Input	Deep Inverse Rendering [5]	Generative Modeling [8]	Diff. Material Graphs [20]	Adversarial Estimation [28]	**Our Method**

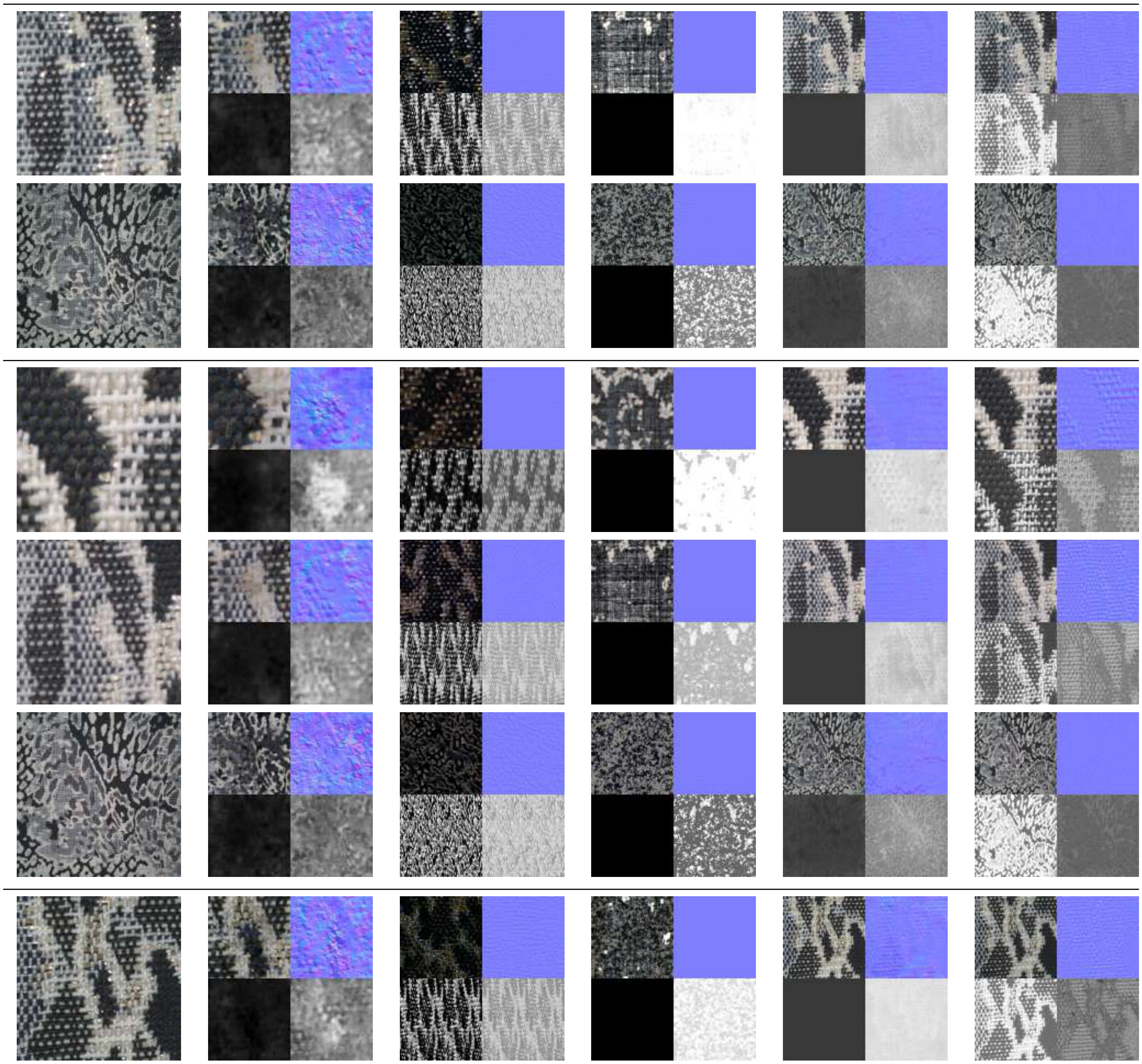

Table 9. Comparisons of our results with previous work on images captured under different conditions for a *jacquard fabric*: On the first rows, images were captured with a smartphone using the flash image. On the middle rows, using the same smartphone with ambient lighting on different scales. On the final row, a scanner image. Note that for [20] we use a fabric material for initialization and use their metallic map instead of specular, that we do not estimate albedos and that the material models are not necessarily comparible.

Input	Deep Inverse Rendering [5]	Generative Modeling [8]	Diff. Material Graphs [20]	Adversarial Estimation [28]	**Our Method**

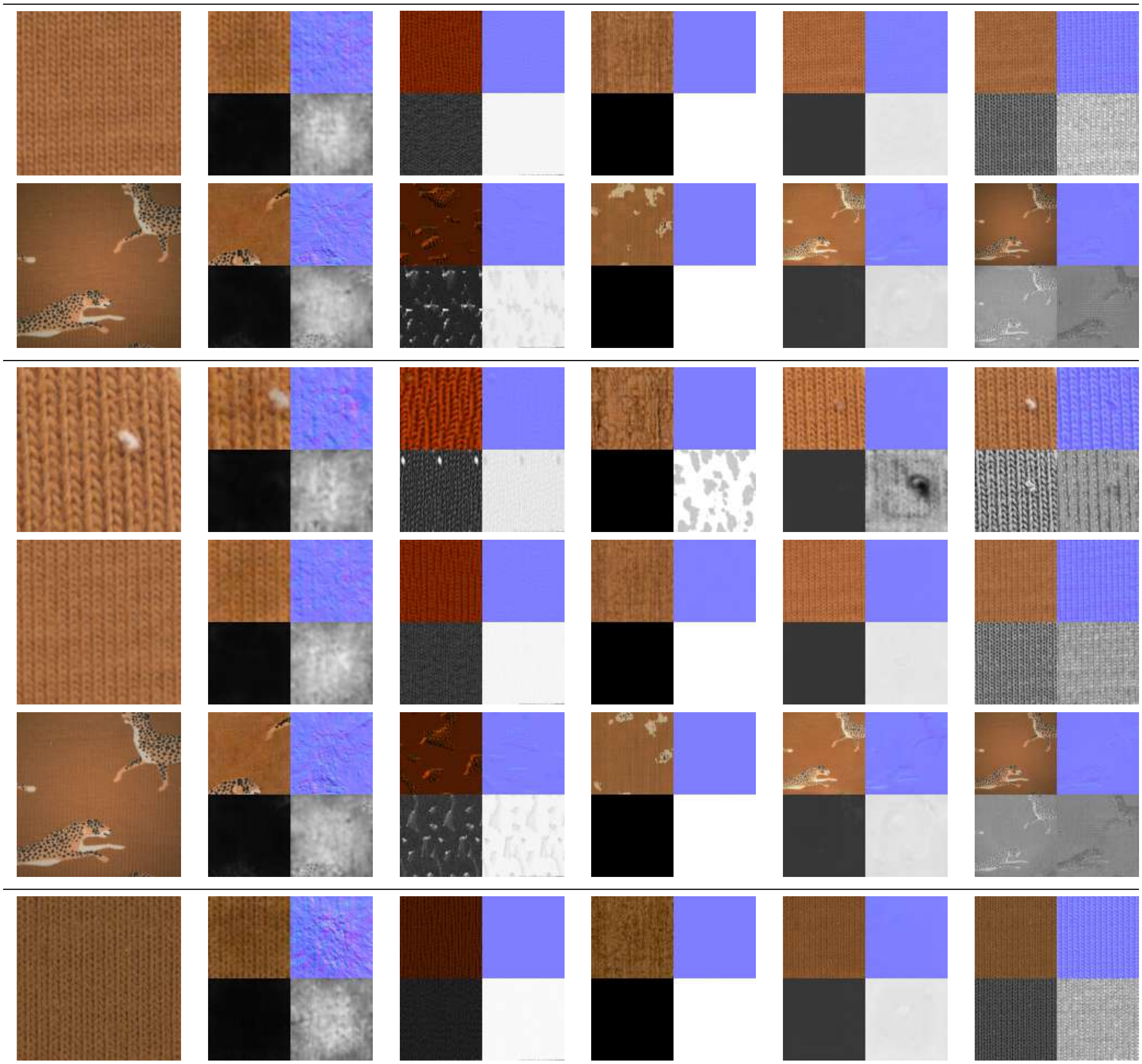

Table 10. Comparisons of our results with previous work on images captured under different conditions for a *single jersey fabric*: On the first rows, images were captured with a smartphone using the flash image. On the middle rows, using the same smartphone with ambient lighting on different scales. On the final row, a scanner image. Note that for [20] we use a fabric material for initialization and use their metallic map instead of specular, that we do not estimate albedos and that the material models are not necessarily comparible.

Input	Deep Inverse Rendering [5]	Generative Modeling [8]	Diff. Material Graphs [20]	Adversarial Estimation [28]	**Our Method**

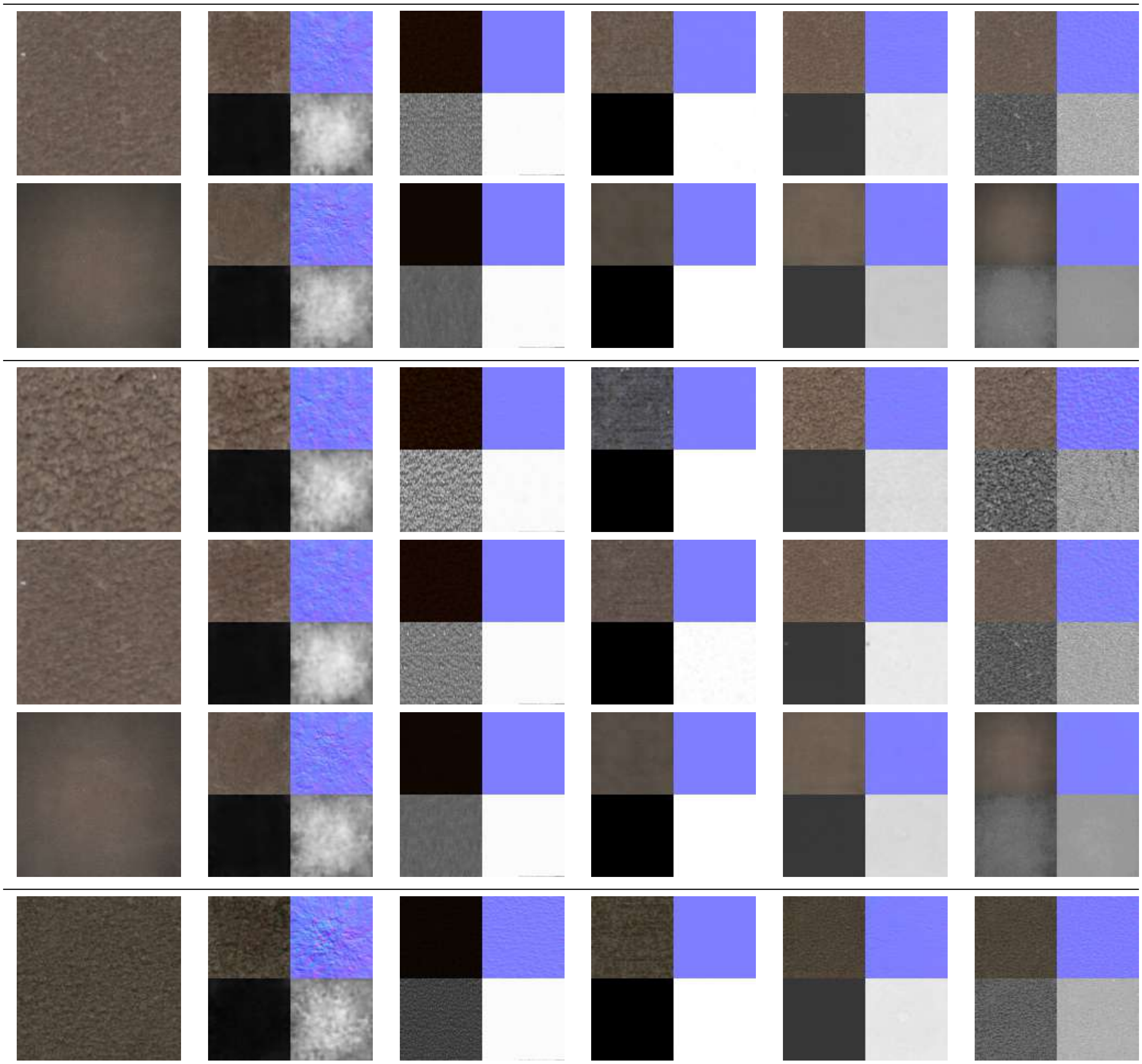

Table 11. Comparisons of our results with previous work on images captured under different conditions for a *suede leather*: On the first rows, images were captured with a smartphone using the flash image. On the middle rows, using the same smartphone with ambient lighting on different scales. On the final row, a scanner image. Note that for [20] we use a fabric material for initialization and use their metallic map instead of specular, that we do not estimate albedos and that the material models are not necessarily comparible.

References

[1] Valentin Deschaintre, Miika Aittala, Frédo Durand, George Drettakis, and Adrien Bousseau. Flexible svbrdf capture with a multi-image deep network. In *Computer Graphics Forum*, volume 38, pages 1–13. Wiley Online Library, 2019. 2

[2] Valentin Deschaintre, Yiming Lin, and Abhijeet Ghosh. Deep polarization imaging for 3d shape and svbrdf acquisition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15567–15576, 2021. 2

[3] Foivos I Diakogiannis, François Waldner, Peter Caccetta, and Chen Wu. Resunet-a: A deep learning framework for semantic segmentation of remotely sensed data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 162:94–114, 2020. 2

[4] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. 2, 5

[5] Duan Gao, Xiao Li, Yue Dong, Pieter Peers, Kun Xu, and Xin Tong. Deep inverse rendering for high-resolution svbrdf estimation from an arbitrary number of images. *ACM Transactions on Graphics (ToG)*, 38(4):134:1–134:15, July 2019. 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32

[6] Elena Garces, Carlos Rodriguez-Pardo, Dan Casas, and Jorge Lopez-Moreno. A Survey on Intrinsic Images: Delving Deep into Lambert and Beyond. *International Journal of Computer Vision*, 2022. 2

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 2

[8] Philipp Henzler, Valentin Deschaintre, Niloy J Mitra, and Tobias Ritschel. Generative modelling of brdf textures from flash images. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 40(6), 2021. 2, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32

[9] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[10] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010. 2

[11] Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021. 2, 5

[12] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017. 2

[13] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018. 2, 6

[14] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 2

[15] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3674–3683, 2020. 2

[16] Carlos Rodriguez-Pardo and Elena Garces. Neural photometry-guided visual attribute transfer. *IEEE Transactions on Visualization and Computer Graphics*, 2022. 2

[17] Carlos Rodriguez-Pardo and Elena Garces. Seamless-GAN: Self-Supervised Synthesis of Tileable Texture Maps. *IEEE Transactions on Visualization and Computer Graphics*, 2022. 2

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015. 2

[19] Edgar Schonfeld, Bernt Schiele, and Anna Khoreva. A u-net based discriminator for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8207–8216, 2020. 2

[20] Liang Shi, Beichen Li, Miloš Hašan, Kalyan Sunkavalli, Tamy Boubekeur, Radomir Mech, and Wojciech Matusik. Match: differentiable material graphs for procedural material capture. *ACM Transactions on Graphics (TOG)*, 2020. 2, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32

[21] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 2

[22] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 2, 5

[23] Michael Weinmann, Juergen Gall, and Reinhard Klein. Material classification based on training data synthesized using a btf database. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 156–171. Springer, 2014. 21

[24] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. 2, 6

[25] Robert J Woodham. Photometric method for determining surface orientation from multiple images. *Optical engineering*, 19(1):139–144, 1980. 21

[26] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. 2, 5

[27] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 2

[28] Xilong Zhou and Nima Khademi Kalantari. Adversarial single-image svbrdf estimation with hybrid training. In *Computer Graphics Forum*, volume 40, pages 315–325. Wiley Online Library, 2021. 2, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32