Scenimefy: Learning to Craft Anime Scene via Semi-Supervised Image-to-Image Translation

Supplementary Material

Yuxin Jiang* Liming Jiang* Shuai Yang Chen Change Loy✉
S-Lab, Nanyang Technological University
{c200203, liming002, shuai.yang, ccloy}@ntu.edu.sg

Abstract

The document provides supplementary information that is not elaborated on in our main paper due to the space constraints: implementation details (Section A), additional comparative results (Section B), loss variant study (Section C), more results of Scenimefy (Section D), and potential limitations (Section E). Code: https://github.com/Yuxinn-J/Scenimefy.

A. Implementation Details

A.1. Network Architecture

StyleGAN finetuing. For the StyleGAN2 finetuning implementation, we modified the code based on FreezeG [2] by incorporating a freezed style vector and low-resolution layers of the generator. Only the last 3 generator blocks were made trainable, each comprising 2 style blocks and a ToRGB layer. To maintain global consistency using pre-trained model priors, we utilized the VGG-19 [9] based perceptual loss [11] to extract features of $G_s(w)$ and $G_t(w)$ up to the $conv4\_4$ layer. For the pre-trained CLIP [8] model, the official version of ViT-B/32 was applied. For the patch-wise contrastive loss, We selected 16 patches of 32×32 at random locations from the images generated by $G_s(w)$ and $G_t(w)$, followed by embedding the image patches using the same CLIP encoder.

Semi-supervised image-to-image (I2I) translation. To implement the semi-supervised I2I translation, we adopted the official source code of [6] in a single-modal setting. Our generator architecture is based on CUT [7], using the ResNet-based generator [4] with 9 residual blocks for training. It contains 2 downsampling blocks, 9 residual blocks [4], and 2 upsampling blocks. A ResNet block is a conv block with skip connections, including a convolution, normalization, ReLU, convolution, normalization, and a residual connection. The downsampling and upsampling blocks contain a two-stride convolution/deconvolution, normalization, and ReLU. The full unsupervised branch is mainly built upon [6].

To incorporate the supervised branch with our proposed *StylePatchNCE* loss, we utilized the same PatchGAN discriminator [5] architecture as our conditional discriminator, classifying whether 70×70 overlapped patches are real or fake.

Regarding the architecture of the generator and layers used for the *StylePatchNCE* loss, we define the first half of the generator G and a 2-layer MLP as an encoder, which is represented as G_{enc} and F. To calculate our multi-layer patch-wise contrastive style loss, we extract features from a total of 5 layers of G_{enc}, which are RGB pixels, the first and second downsampling convolution, and the first and the fifth residual block, corresponding to layer ids $0, 4, 8, 12, 16$. These layers correspond to receptive fields of sizes 1×1, 9×9, 15×15, 35×35, and 99×99. Following CUT, for each layer's features, we sampled 256 patches from random locations and applied F to obtain 256-dimensional final features.

A.2. More Details on Data Selection and Training

We presented most of the data selection and model training details in our main paper. Below we show some additional details.

Semantic segmentation guided data selection. To filter the data, the Mask2Former [3] semantic segmentation model with Swin-B (IN21k) as the backbone and pre-trained on the ADE20K [12] dataset was utilized.

More training details. The training of our semi-supervised I2I translation model uses a single NVIDIA GeForce RTX 3090 GPU and a batch size of one. At each iteration, the generator G forwards twice, generating both $G(x)$ and $G(x^p)$. After calculating the total loss, backpropagation is performed once. The conditional discriminator D_P is trained to distinguish between the concatenated $\{(y^p, x^p)\}$

* Equal contribution.

and $\{(G(x^p), x^p)\}$. During training, the model consumes approximately 5k MiB memory. In terms of the inference speed, the rate is tested as 0.027 second per image, achieving real-time anime-style rendering.

B. Additional Comparative Results

More visual comparisons. In Section 4.2 of the main paper, we compared our methods with five state-of-the-art methods. Due to space limit, four groups of examples were presented. In Figures 3, 4 and 5, we provide additional comparative results to further demonstrate the exceptional performance of our approach. All the previous baselines generate inferior results than our method in terms of evident stylization, consistent semantic preservation, and fine-detailed anime textures. Scenimefy achieves a more faithful anime stylization with richer colors and stronger artistic expression, resulting in the best quality output.

C. Loss Variant Study

As mentioned in Section 3.3.1 of the main paper, we introduce a novel patch-wise contrastive style loss, *i.e.*, *StylePatchNCE*, instead of the strict reconstruction loss [5] used in typical supervised I2I translation frameworks. We have ablated the proposed *StylePatchNCE* loss in Section 4.3 of the main paper to show its usefulness. To further verify its effectiveness, here we directly replace the *StylePatchNCE* loss with the standard L_1 reconstruction loss as a variant for a more comprehensive comparison.

The qualitative results are presented in Figure 1. The L_1 loss variant resulted in an unnatural color and learned poor local anime textures, such as the clouds (in Figure 1(b), Row 2) and the stones (in Figure 1(b), Row 3). In contrast, our method successfully translated both the global anime style and local anime textures.

D. More Examples Generated by Scenimefy

D.1. Additional Results on Scene Stylization

We present more generated images (in Figures 6 and 7) by Scenimefy. Thanks to the fully convolutional neural network, our model can produce high-quality rectangular images despite being trained on squared images with a resolution of 256×256.

D.2. Generalizability to Other Cases

When we directly apply our model trained on natural scene landscapes [10], we find that it performs relatively well even in some other cases. The content of the test data includes city views, architecture, portraits, animals, plants, food, and other objects from the DIV2K [1] dataset. Overall, the results (see Figures 8, 9 and 10) demonstrate that our method can generate high-quality anime stylized images in

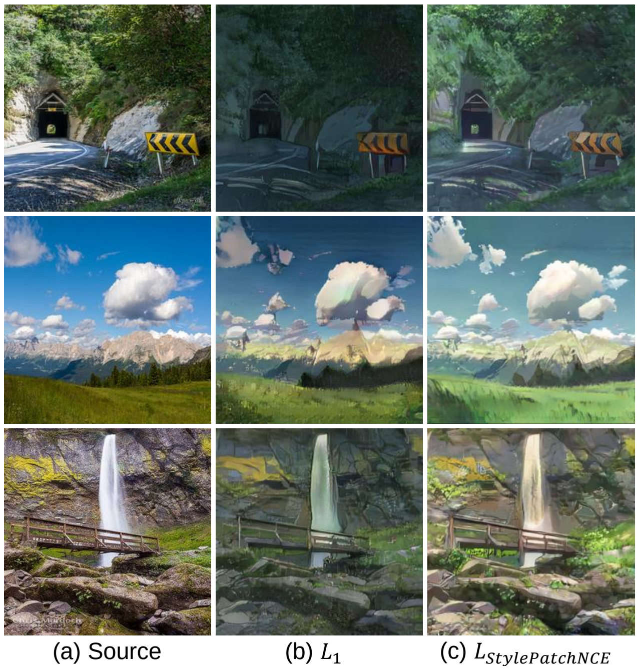

Figure 1: **Effects of the loss function variants for the supervised branch.** We compare the effects of the L_1 loss with $L_{StylePatchNCE}$ loss applied to pseudo paired data.

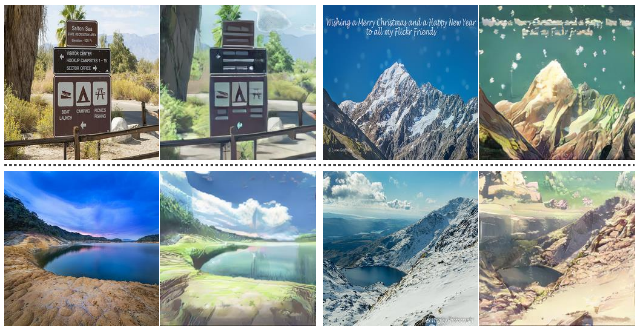
Wishing a Merry Christmas and a Happy New Year to all my Flickr Friends

Figure 2: **Certain failure cases.** Top: the relatively poor preservation of the tiny text-like details; Bottom: the incorrect semantic translation in a small number of cases.

other diverse use cases and real-world scenes, indicating its certain generalizability.

E. Limitations

Despite the promising results, our method also has certain limitations. Due to the absence of strong constraints imposed between the output image and the source image, our model fails to preserve intricate tiny details, such as text, as depicted in Figure 2 (Top). However, this limitation may be overcome by introducing a simple content loss, such as the reconstruction loss or perceptual loss. This encourages the model to preserve the content of the source image more closely, albeit at the cost of slightly compromising the anime-style effect.

Moreover, our model exhibits a small number of failure cases, where it translates semantically distinct objects incorrectly due to the scene complexity and the biases in the training data. For instance, as shown in the bottom row of Figure 2, the model translates the stone ground into the grass, as well as the ice mountain into the stone mountain. Addressing these failure cases with more semantic consistency can also be interesting future work apart from the ones we discussed in Section 5 of the main paper.

References

[1] Eirikur Agustsson and Radu Timofte. NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *CVPRW*, 2017. 2

[2] bryandlee. Freeze generator. `https://github.com/bryandlee/FreezeG`, 2020. 1

[3] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022. 1

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 1, 2

[6] Chanyong Jung, Gihyun Kwon, and Jong Chul Ye. Exploring patch-wise semantic relation for contrastive learning in image-to-image translation tasks. In *CVPR*, 2022. 1

[7] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *ECCV*, 2020. 1

[8] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint*, 2014. 1

[10] Ivan Skorokhodov, Grigorii Sotnikov, and Mohamed Elhoseiny. Aligning latent and image spaces to connect the unconnectable. In *CVPR*, 2021. 2

[11] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 1

[12] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017. 1

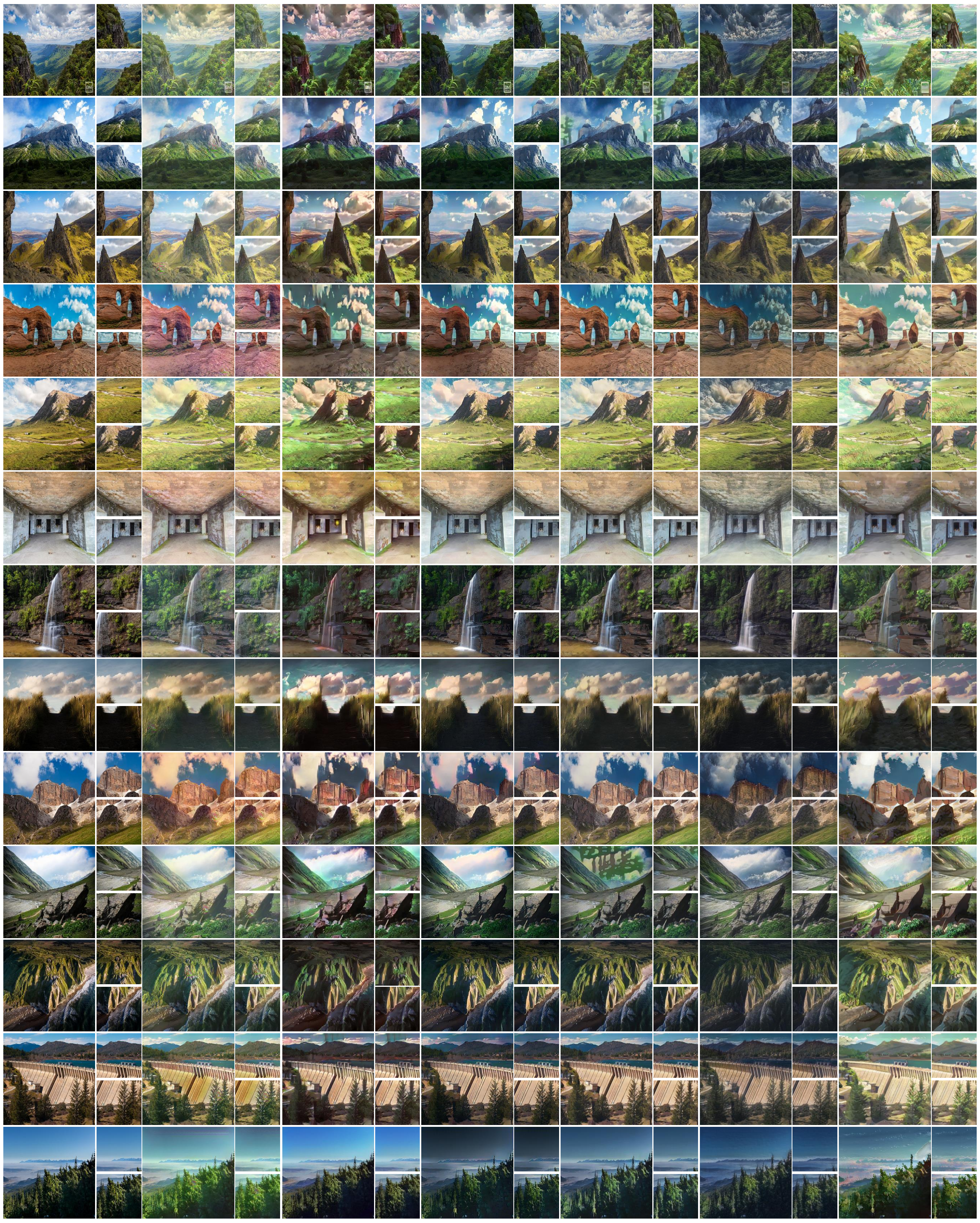

Figure 3: **Additional qualitative comparative results.** Zoom in for details.

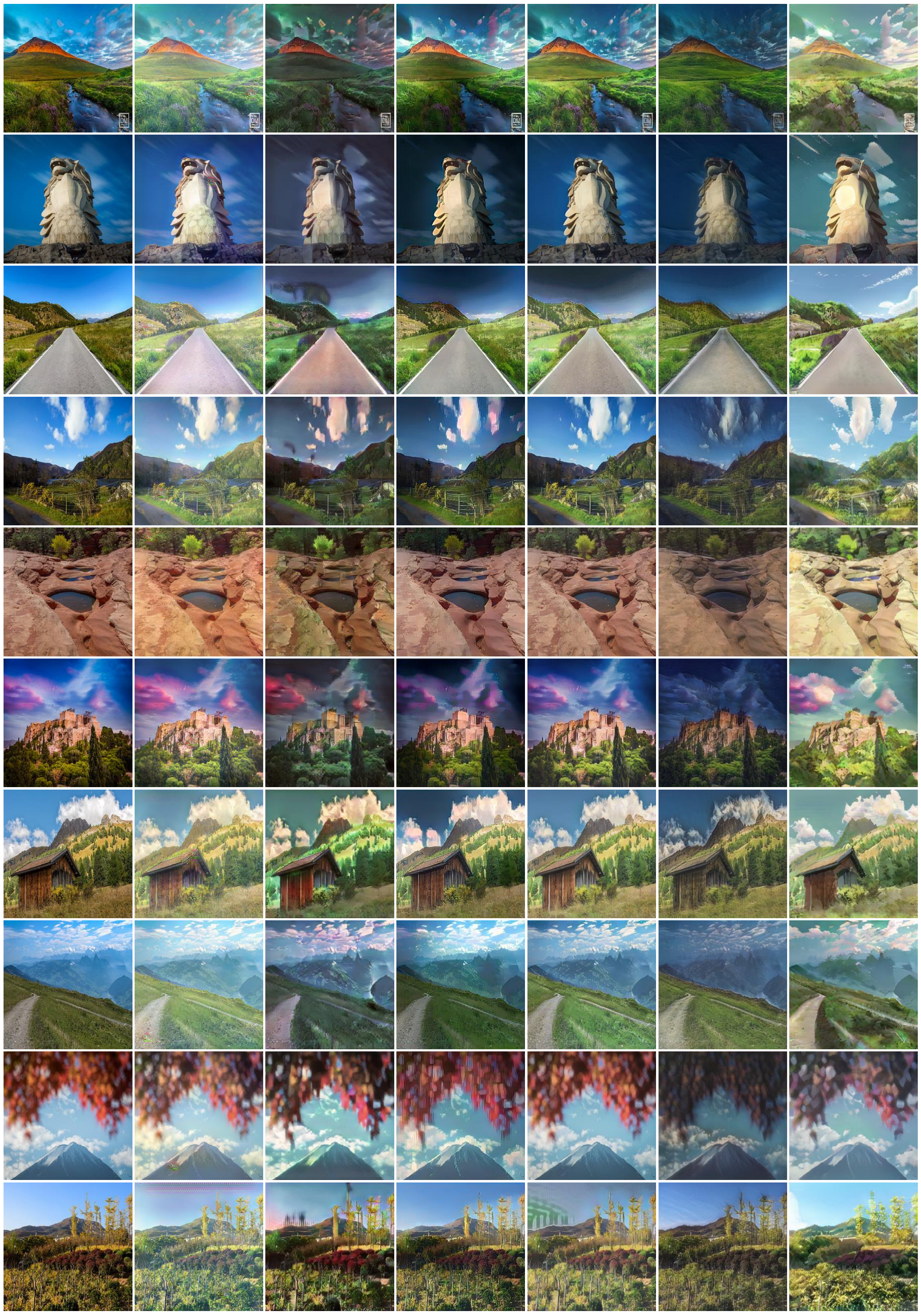

Figure 4: **Additional qualitative comparative results.** Zoom in for details.

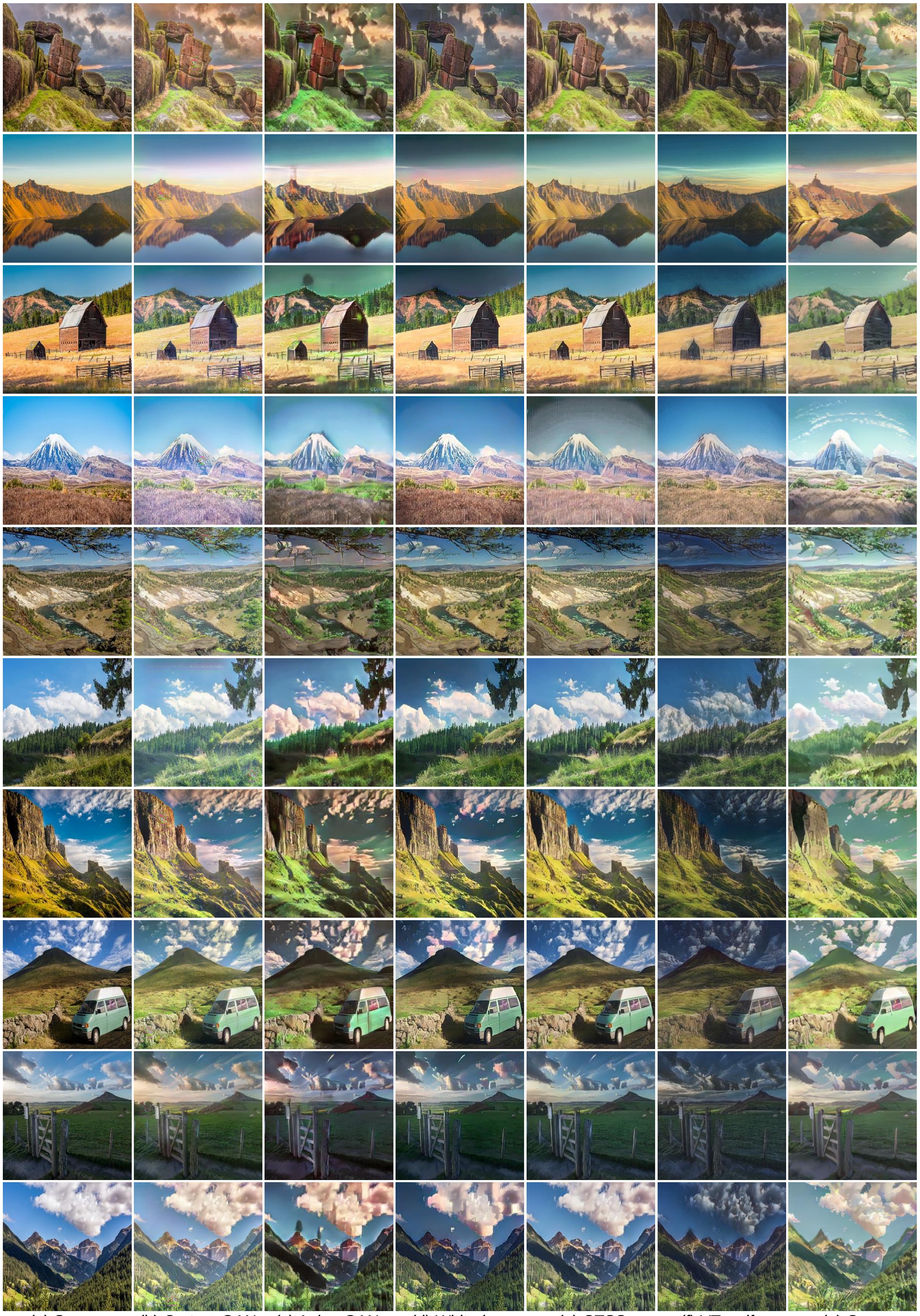

Figure 5: **Additional qualitative comparative results.** Zoom in for details.

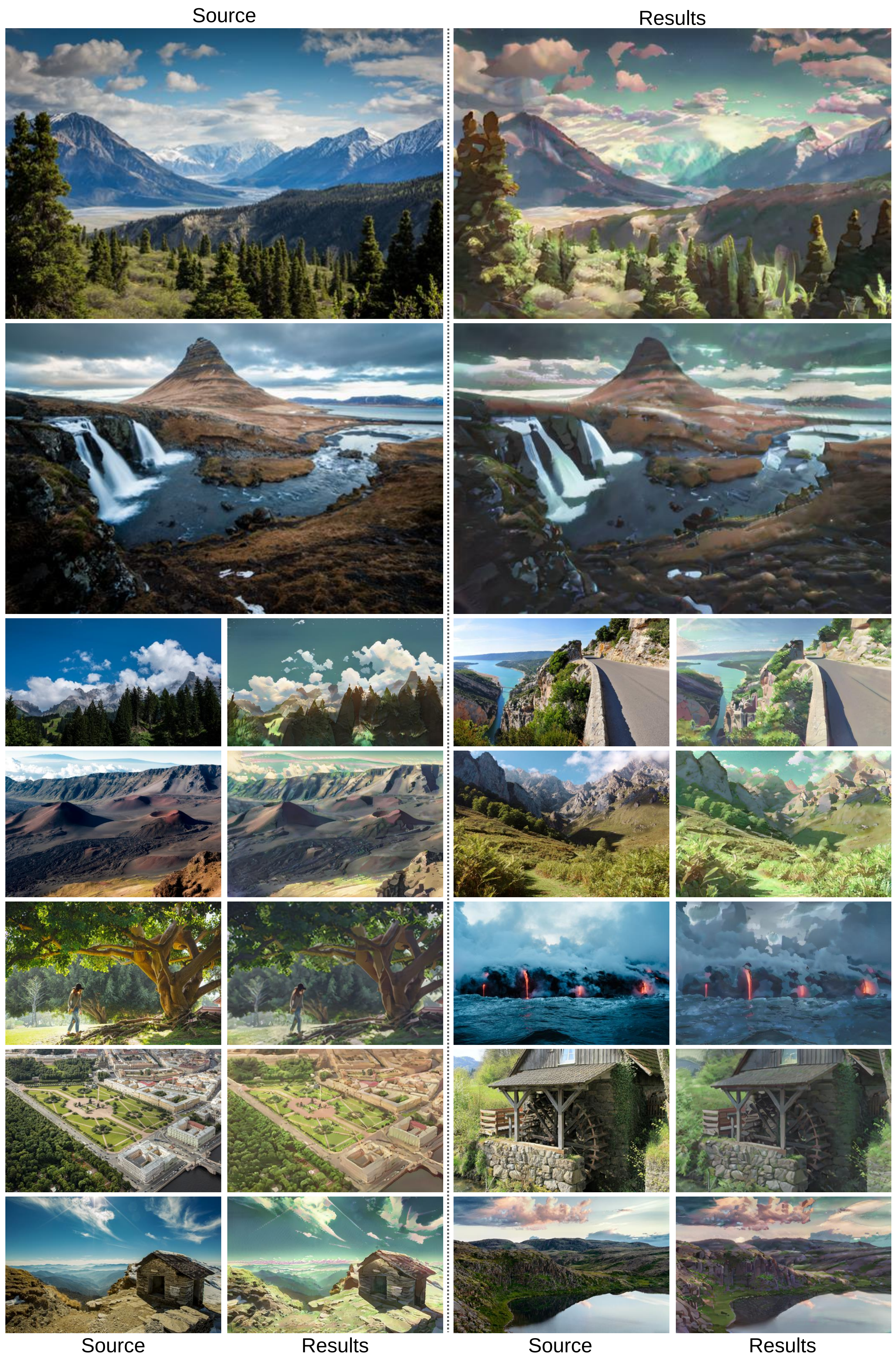

Figure 6: **Additional results of Scenimefy on natural landscapes.** Zoom in for details.

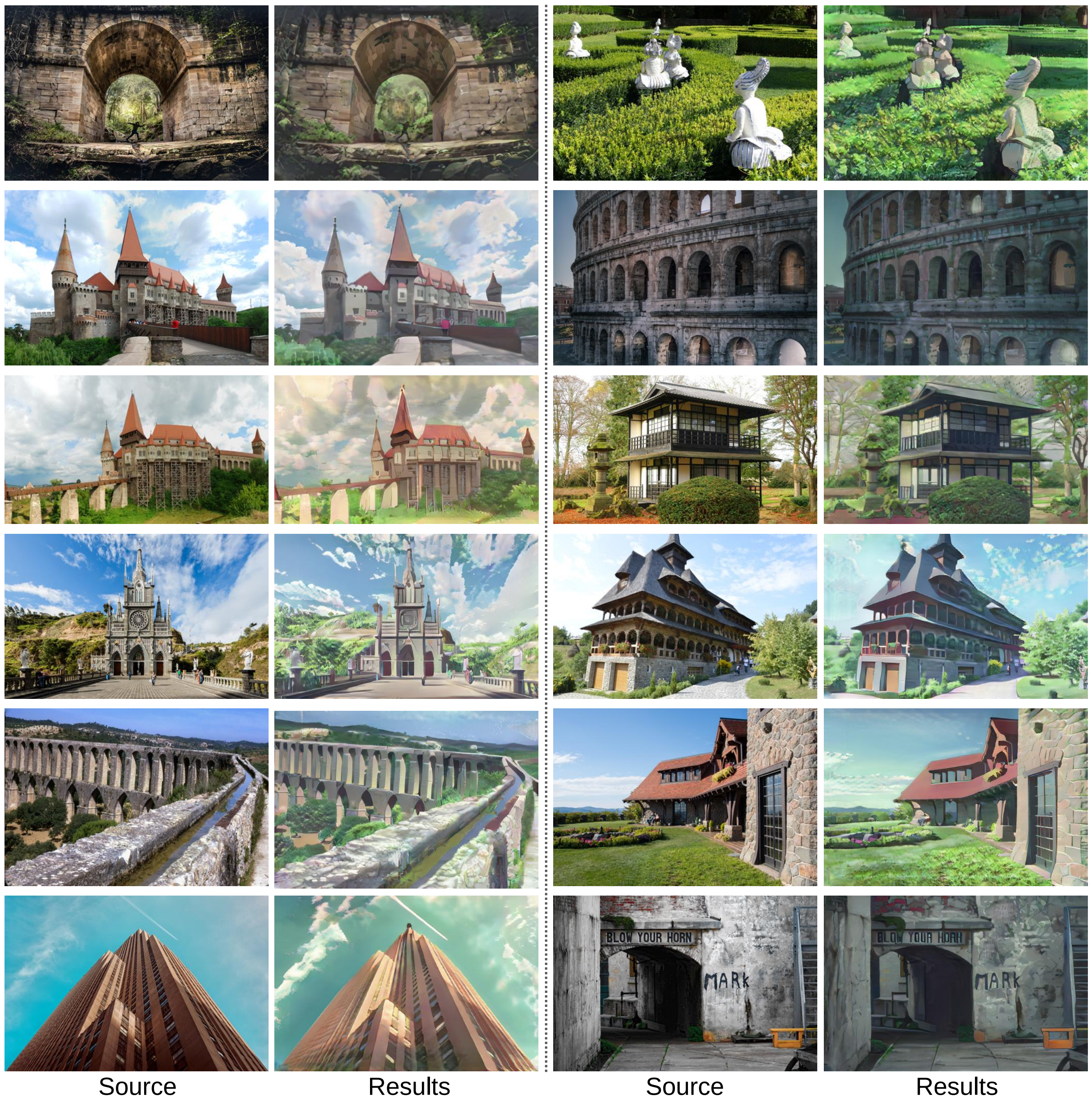

Figure 7: **Additional results of Scenimefy on architecture and buildings.** Zoom in for details.

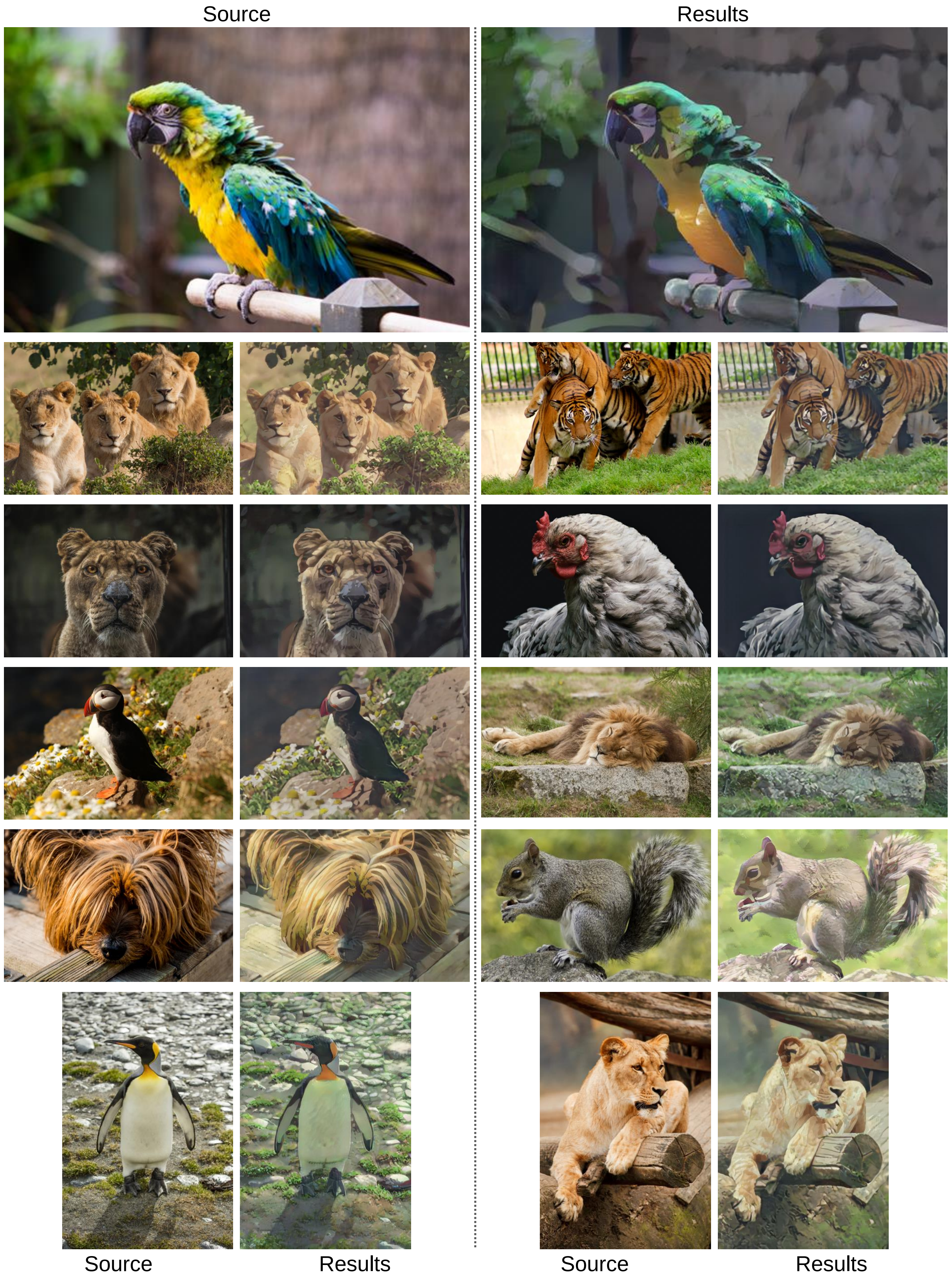

Figure 8: **Additional results of Scenimefy on animals.** Zoom in for details.

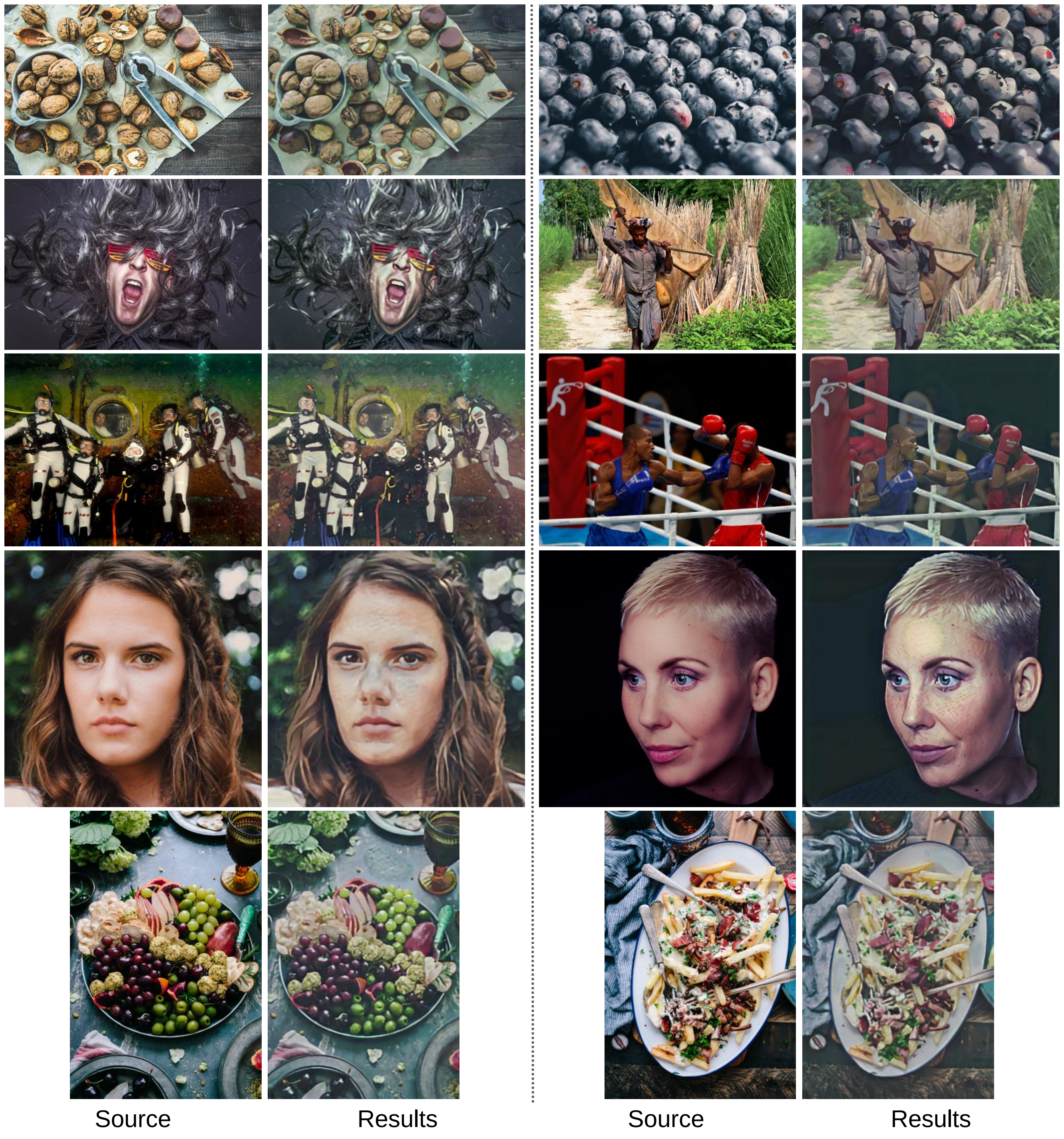

Figure 9: **Additional results of Scenimefy on food and people.** Zoom in for details.

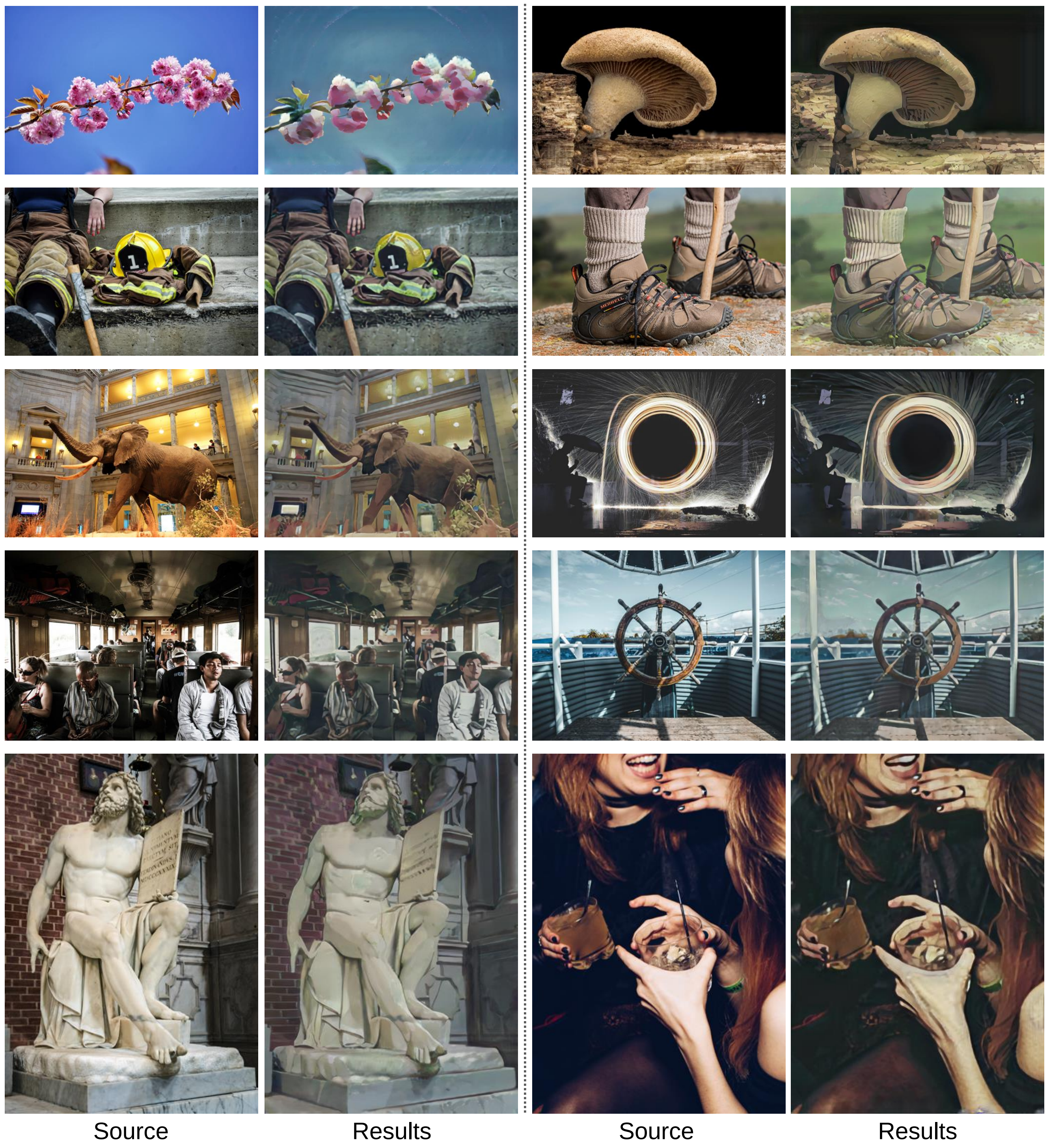

Figure 10: **Additional results of Scenimefy on other objects.** Zoom in for details.