Content-Consistent Generation of Realistic Eyes with Style (Supplementary Material)

Marcel Bühler [1], Seonwook Park[1], Shalini De Mello[2], Xucong Zhang[1], Otmar Hilliges[1]
[1]ETH Zürich, [2]NVIDIA
{buehlmar, spark, zhangxuc, otmarh}@ethz.ch; shalinig@nvidia.com

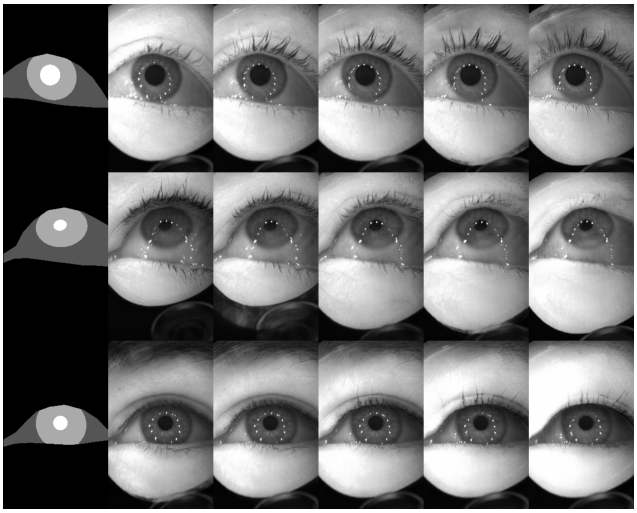

Figure 1: Interpolation results with segmentation mask. We interpolate between two styles given a semantic segmentation mask (content). The content is consistent across all generated images.

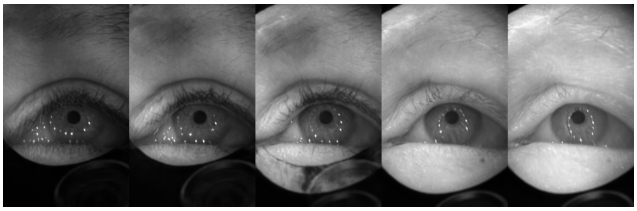

Figure 2: Mode crossing in latent space. When we interpolate between style images from different modes, we observe a "mode crossing" (middle image). In this case we go from an image with a large black background (left) to an image with only a small background (right).

A. Visualizations for Seg2Eye

Figure 1 shows additional interpolation results. Note how eye shape (content) is consistent across all generated images. Figure 2 demonstrates an example for "mode crossing" in latent space. Mode crossing happens when interpolating between images from different modes (e.g. from an image with a large background to an image with a very small background, which we believe indicates whether an image was taken from the left-eye camera or the right-eye camera). In Figure 3, we can see a limitation of our approach. Due to an under-representation of subjects wearing glasses in the training set (only 3 of 29 subjects wear glasses), Seg2Eye fails to pick up the signal and does not produce eyes with glasses.

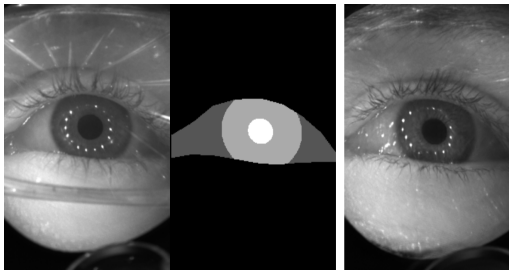

Figure 3: Limitation. Given the style and content inputs (first two columns), Seg2Eye produces an output eye without glasses (last column). Among all 29 subjects in the training set, only 3 of them wear glasses, making the learning of this aspect challenging.

B. Implementation Details

B.1. Segmentation Network

We build on top of an existing PyTorch implementation[1] of the DeepLab v3+ semantic segmentation network [1] with Resnet-101 backbone. We use the learning rate 0.005 and the SGD optimizer with weight decay 0.0005 and batch size 8, and optimize a cross-entropy loss.

B.2. Refiner Network

We modify the DeepLab v3+ architecture from Section B.1 to take the concatenation of the target segmentation map M_T, a similar reference image I from the same mode and its pseudo-label M_I as input. The loss function is an L2

[1] https://github.com/jfzhang95/pytorch-deeplab-xception github.com/jfzhang95/pytorch-deeplab-xception

loss on the network output vs. the ground truth image, in the same manner as how the competition scoring is calculated. As an optimizer, we use Stochastic Gradient Descent (SGD) with learning rate 0.01 exponentially decaying over 5 epochs, weight decay 0.001 and batch size 8.

B.3. Seg2Eye

Encoder We use the same encoder architecture as [3] with 6 convolutional layers.

Generator Our generator architecture is based the implementation by [3][2], but we replace the SPADE blocks with SPADE+Style ResBlocks as described in the main paper. We use use 7 SPADE+Style ResBlocks.

Hyperparameters We use Adam [2] with an initial learning rate of 0.0002 ($\beta_1 = 0.5, \beta_2 = 0.999$) to train our network for 21 epochs with a linearly decaying learning rate for the last 7 epochs. The dimensionality of the latent style code is 16. The generator starts from a downsampled segmentation mask with dimensions 8×10 and produces an output image of dimensions 256×320, which is upsampled to full resolution (400×640) via bilinear interpolation.

During training, we sample from the 100 most similar images (in terms of predicted segmentation mask) from the unlabeled dataset (given our ranking described in the main paper). Due to GPU memory constraints, we train with batch size 1 and 4 style images per sample. In previous experiments, we used more style images, which helped with style consistency.

C. Additional Experiments with Similar Results

We conduct several additional experiment with different loss and hyper-parameters yielding similar results. For example, we substitute the L2 pixel-consistency loss with L1. As an additional experiment, we replace the SPADE+Style Block with a sequential version, where we first apply AdaIN and then SPADE. For style aggregation, we implement both taking the maximum and the mean. All these variations yield similar results to the approach described in this paper.

[2] https://github.com/NVlabs/SPADE

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 40(4):834–848, 2017. 1

[2] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. 12 2014. 2

[3] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019. 2