Image Completion with Heterogeneously Filtered Spectral Hints – Supplementary Material –

Xingqian Xu[1,3], Shant Navasardyan[3], Vahram Tadevosyan[3], Andranik Sargsyan[3], Yadong Mu[2], and Humphrey Shi[1,3]

[1]SHI Lab @ UIUC & UO, [2]Peking University, [3]Picsart AI Research (PAIR)

1. Architecture

In the main article, we have introduced two model settings, small and standard, and the sole difference between our small and standard settings is to shrink the base channel number by half. The base channel number is a hyperparameter that predominantly affects memory usage of the model during training and inference. In each encoder and synthesis block, the channel number N of its convolution layers is computed by an equation connecting the base channel number N_{base}, the spatial resolution N_{res} and a predefined max channel N_{max} (see Eq. 1). For our small model, N_{base} is set to 16,384. For our standard model, N_{base} is set to 32,768. All models use N_{max} equals to 512.

$$N = \min\left(\frac{N_{base}}{N_{res}}, N_{max}\right) \tag{1}$$

Our experiments showed that the standard model (*i.e.* $N_{base} = 32768$) might make the GAN training unstable using batch size 32 on 4 GPUs. Researchers were suggested to use 8 GPUs to make the batch size per GPU no larger than 4 when training the standard model. Nevertheless, our small model was more friendly towards 4 GPU training, helping run the experiment with less computational resources. The GPU memory usage on our small and standard model on resolution 256 and 512 experiments were approximately 10G and 18G per GPU. Remember that our small model could fit batch size 8 per GPU, two times larger than the standard model. On the other hand, the model size in terms of parameters does not significantly differ between our small and regular settings. For resolution 512, the small model contains 68.2 million parameters while the standard model contains 79.8 million parameters (*i.e.* 17% more parameters). The model's sizes are 68.0 million and 79.2 million parameters for resolution 256, respectively. In summary, we list out the detailed architecture in Table 1 for better illustrations.

2. Evaluation Details

SH-GAN has been fully implemented in PyTorch. Simultaneously, we replicated CoModGAN [7] using PyTorch apart from its original TensorFlow implementation. When we evaluated SH-GAN and CoModGAN, we followed the common five-run rules, and we took the mean values for all scores (*i.e.* FID, LPIPS, PSNR, and SSIM). The variations of the results came from three places: a) the latent codes were randomly sampled from the normal distribution $\mathcal{N}(0, 1)$; b) random noises with learned magnitude were injected into each synthesis block; c) masks were generated randomly. We noticed a typical ± 0.3 on FID score around the mean, which aligned with our expectation.

For other models, we used our mask generation rules to create several sets of fixed masks and then evaluated these models using the official demo code provided in their Github. LaMa [3] and Onion-Conv [2] mentioned in their work that they applied the same model for all resolutions; thus, we followed their evaluation scheme. For DeepFillV2 [4] and CR-Fill [6], we downloaded separate models for resolutions 256 and 512; thus, we evaluated using these models correspondingly. For MADF [9] and AOT-GAN [5], they only provided resolution 512 models. Therefore, in our resolution 256 evaluations, we upsampled both images and masks into 512×512, executed the model, and then downsampled the output images back to 256×256. CR-Fill [6] and Onion-Conv [2] didn't train on face datasets, so we skipped those experiments.

3. Masks

As mentioned in section 3.4 of the main article, we followed the same algorithm as DeepFillv2 [4] and CoModGAN [7] to generate free-form masks for training and evaluation. Besides, we adopted the LaMa-style [3] narrow and wide masks to extend our evaluation beyond free-form masks scenarios. We use the official code downloaded from LaMa's Github to generate both narrow and wide masks. We list visualization for all three types of masks in figure 1.

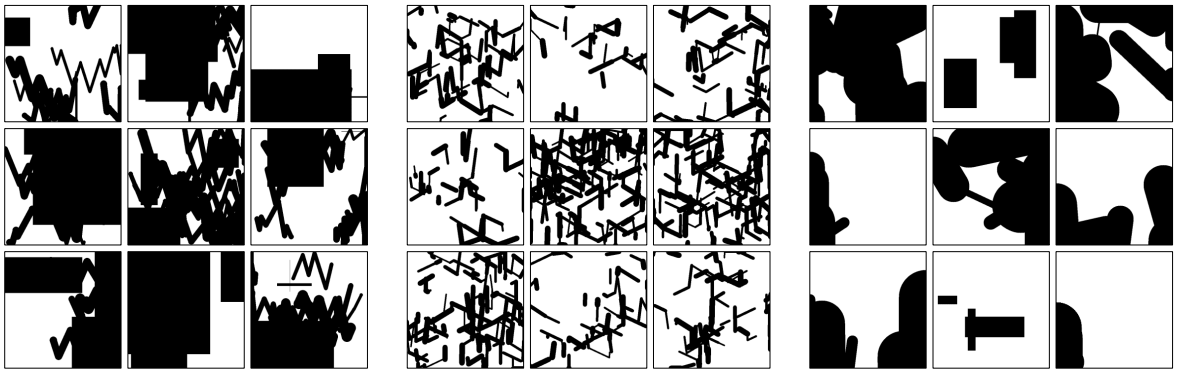
	block resolution	model 256 small	model 256 standard	model 512 small	model 512 standard
encoder	512×512			conv1×1 ($4 \rightarrow 32$) conv3×3 ($32 \rightarrow 32$) conv3×3 ($32 \rightarrow 64$, ds)	conv1×1 ($4 \rightarrow 64$) conv3×3 ($64 \rightarrow 64$) conv3×3 ($64 \rightarrow 128$, ds)
	256×256	conv1×1 ($4 \rightarrow 64$) conv3×3 ($64 \rightarrow 64$) conv3×3 ($64 \rightarrow 128$, ds)	conv1×1 ($4 \rightarrow 128$) conv3×3 ($128 \rightarrow 128$) conv3×3 ($128 \rightarrow 256$, ds)	conv3×3 ($64 \rightarrow 64$) conv3×3 ($64 \rightarrow 128$, ds)	conv3×3 ($128 \rightarrow 128$) conv3×3 ($128 \rightarrow 256$, ds)
	128×128	conv3×3 ($128 \rightarrow 128$) conv3×3 ($128 \rightarrow 256$, ds)	conv3×3 ($256 \rightarrow 256$) conv3×3 ($256 \rightarrow 512$, ds)	conv3×3 ($128 \rightarrow 128$) conv3×3 ($128 \rightarrow 256$, ds)	conv3×3 ($256 \rightarrow 256$) conv3×3 ($256 \rightarrow 512$, ds)
	64×64	conv3×3 ($256 \rightarrow 256$) conv3×3 ($256 \rightarrow 512$, ds)	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)	conv3×3 ($256 \rightarrow 256$) conv3×3 ($256 \rightarrow 512$, ds)	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)
	32×32	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)
	16×16	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)
	8×8	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)	conv3×3 ($512 \rightarrow 512$) conv3×3 ($512 \rightarrow 512$, ds)
	4×4	conv3×3 ($512 \rightarrow 512$) fc ($512 \times 4 \times 4 \rightarrow 1024$) dropout	conv3×3 ($512 \rightarrow 512$) fc ($512 \times 4 \times 4 \rightarrow 1024$) dropout	conv3×3 ($512 \rightarrow 512$) fc ($512 \times 4 \times 4 \rightarrow 1024$) dropout	conv3×3 ($512 \rightarrow 512$) fc ($512 \times 4 \times 4 \rightarrow 1024$) dropout
synthesis	4×4	fc ($1024 \rightarrow 512 \times 4 \times 4$) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)	fc ($1024 \rightarrow 512 \times 4 \times 4$) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)	fc ($1024 \rightarrow 512 \times 4 \times 4$) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)	fc ($1024 \rightarrow 512 \times 4 \times 4$) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)
	8×8	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)
	16×16	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)
	32×32	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)
	64×64	conv3×3 ($512 \rightarrow 256$, us) conv3×3 ($256 \rightarrow 256$) torgb ($256 \rightarrow 3$)	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)	conv3×3 ($512 \rightarrow 256$, us) conv3×3 ($256 \rightarrow 256$) torgb ($256 \rightarrow 3$)	conv3×3 ($512 \rightarrow 512$, us) conv3×3 ($512 \rightarrow 512$) torgb ($512 \rightarrow 3$)
	128×128	conv3×3 ($256 \rightarrow 128$, us) conv3×3 ($128 \rightarrow 128$) torgb ($128 \rightarrow 3$)	conv3×3 ($512 \rightarrow 256$, us) conv3×3 ($256 \rightarrow 256$) torgb ($256 \rightarrow 3$)	conv3×3 ($256 \rightarrow 128$, us) conv3×3 ($128 \rightarrow 128$) torgb ($128 \rightarrow 3$)	conv3×3 ($512 \rightarrow 256$, us) conv3×3 ($256 \rightarrow 256$) torgb ($256 \rightarrow 3$)
	256×256	conv3×3 ($128 \rightarrow 64$, us) conv3×3 ($64 \rightarrow 64$) torgb ($64 \rightarrow 3$)	conv3×3 ($256 \rightarrow 128$, us) conv3×3 ($128 \rightarrow 128$) torgb ($128 \rightarrow 3$)	conv3×3 ($128 \rightarrow 64$, us) conv3×3 ($64 \rightarrow 64$) torgb ($64 \rightarrow 3$)	conv3×3 ($256 \rightarrow 128$, us) conv3×3 ($128 \rightarrow 128$) torgb ($128 \rightarrow 3$)
	512×512			conv3×3 ($64 \rightarrow 32$, us) conv3×3 ($32 \rightarrow 32$) torgb ($32 \rightarrow 3$)	conv3×3 ($128 \rightarrow 64$, us) conv3×3 ($64 \rightarrow 64$) torgb ($64 \rightarrow 3$)

Table 1: The detail architecture of the encoder and synthesis network in our small and standard SH-GAN. All convolution layers are followed by the leaky ReLU activation with $\alpha = 0.2$. Annotation *ds* and *us* means downsample and upsample. The $torgb$ layer is a conv1×1 layer that converts image features into RGB images, which will be aggregated in parallel with the main architecture. For simplicity, we don't list out the mapping network, which is a sequential network with eight 512 to 512 fully connected layers plus ReLU. We don't list our Spectral Hint Unit (SHU) as well. SHU details can be found in the main article.

As shown, narrow masks yield more and thinner strokes, while wide masks yield fewer and broader strokes. Both narrow and wide masks yield easier inpainting cases than the regular free-form masks we used in the main article. Moreover, we performed both quantitative and qualitative evaluations on these masks. Please see the following sections for more details.

4. Extra Results

As mentioned in the last section, we extensively tested the robustness of the good performance of SH-GAN beyond free-form mask scenarios, using LaMa-style [3] narrow and wide masks. All quantitative results are listed in Table 2 and 3. Like in the main article, we tested all approaches with resolutions 256 and 512, using dataset FFHQ [1] and



(a) Free-form mask (b) Narrow mask (c) Wide mask

Figure 1: The three types of masks we use in our experiments. The 1-value (white) represents known pixels and the 0-value (black) represents the unknown pixels.

	FFHQ 256				**Places2 256**			
	narrow		**wide**		**narrow**		**wide**	
Method	FID(↓)	LPIPS(↓)	FID(↓)	LPIPS(↓)	FID(↓)	LPIPS(↓)	FID(↓)	LPIPS(↓)
CoModGan (small)	2.0327	0.0430	2.5680	0.1235	1.7972	0.0880	3.4491	0.2022
CoModGan (official)	1.7082	**0.0411**	2.4859	0.1224	1.5214	0.0857	3.3955	0.2016
LaMa	4.6266	0.0418	8.5166	**0.1169**	11.4755	0.0746	12.6746	**0.1792**
DeepFillV2	8.5031	0.0673	14.7631	0.1531	15.0309	0.0980	20.6116	0.2160
CR-Fill	-	-	-	-	12.5929	0.0935	18.1343	0.2036
Onion-Conv	-	-	-	-	15.0021	0.1107	18.2985	0.2239
MADF	1.5619	0.0312	7.3928	0.1257	10.4441	**0.0730**	20.0603	0.2023
AOT-GAN	2.1126	0.0355	15.6649	0.1576	9.9570	0.0843	26.5505	0.2353
(ours - small)	2.0790	0.0437	2.5108	0.1222	1.5607	0.0867	3.0663	0.2000
(ours - regular)	**1.6847**	0.0414	**2.3336**	0.1214	**1.3084**	0.0832	**2.7853**	0.1982

Table 2: The performance on resolution 256 with LaMa-style [3] narrow and wide masks.

	FFHQ 512				**Places2 512**			
	narrow		**wide**		**narrow**		**wide**	
Method	FID(↓)	LPIPS(↓)	FID(↓)	LPIPS(↓)	FID(↓)	LPIPS(↓)	FID(↓)	LPIPS(↓)
CoModGan (small)	1.4628	0.0585	2.8030	0.1936	1.2474	0.0998	4.8398	0.2574
CoModGan (official)	1.2668	0.0546	2.7336	0.1925	1.0363	0.0940	4.5978	0.2566
LaMa	2.3060	0.0608	11.6739	0.2162	1.2551	0.0884	7.6624	**0.2425**
DeepFillV2	9.0039	0.1156	19.9922	0.2361	2.0743	0.1003	18.3419	0.2871
CR-Fill	-	-	-	-	1.7619	0.0916	14.8824	0.2675
Onion-Conv	-	-	-	-	3.1910	0.1149	14.0048	0.2999
MADF	1.4295	0.0544	11.2990	0.2041	1.0556	**0.0751**	18.4953	0.2532
AOT-GAN	1.7366	**0.0516**	23.6286	0.2458	1.3204	0.0808	28.4871	0.2950
(ours - small)	1.4070	0.0592	2.7658	0.1934	1.1583	0.0993	4.4051	0.2570
(ours - regular)	**1.2053**	0.0550	**2.6009**	**0.1901**	**0.9307**	0.0916	**3.9755**	0.2532

Table 3: The performance on resolution 512 with LaMa-style [3] narrow and wide masks.

Places2 [8]. As shown in the tables, our SH-GAN still outperforms other methods in terms of FID in all experiments, demonstrating that SH-GAN is effective under a wide variety of inpainting cases.

5. Visualization

In addition to the demo we showed in the main article, we generated more images by our SH-GAN and other approaches and qualitative compared them side by side in Figure 3. Other than that, we also generate images using LaMa-style [3] narrow and wide mask and listed them accordingly in Figure 4 and 5.

6. Controllable Editing

Controllable editing is one downstream application with excellent potential for SH-GAN in production. The definition of the task is to fill the missing region of an image I with guidance from a reference image I_R. During inference, we pass both I, I_R, and the mask into the encoder. We then modulate our synthesis network using the global vector from I_R, and connect other intermediate features from I. Remind that this application is also feasible for Co-ModGAN [7] but not for LaMa [3] and other approaches. The results are shown in Figure 2, in which we successfully transit parts of the reference face into the designated regions.

Figure 2: Controllable editing on different source and reference images from FFHQ.

References

[1] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[2] Shant Navasardyan and Marianna Ohanyan. Image inpainting with onion convolutions. In *Proceedings of the Asian Conference on Computer Vision*, 2020.

[3] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2149–2159, 2022.

[4] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Free-form image inpainting with gated convolution. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4471–4480, 2019.

[5] Yanhong Zeng, Jianlong Fu, Hongyang Chao, and Baining Guo. Aggregated contextual transformations for high-resolution image inpainting. *arXiv preprint arXiv:2104.01431*, 2021.

[6] Yu Zeng, Zhe Lin, Huchuan Lu, and Vishal M Patel. Cr-fill: Generative image inpainting with auxiliary contextual reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14164–14173, 2021.

[7] Shengyu Zhao, Jonathan Cui, Yilun Sheng, Yue Dong, Xiao Liang, Eric I Chang, and Yan Xu. Large scale image completion via co-modulated generative adversarial networks. *arXiv preprint arXiv:2103.10428*, 2021.

[8] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[9] Manyu Zhu, Dongliang He, Xin Li, Chao Li, Fu Li, Xiao Liu, Errui Ding, and Zhaoxiang Zhang. Image inpainting by end-to-end cascaded refinement with mask awareness. *IEEE Transactions on Image Processing*, 30:4855–4866, 2021.

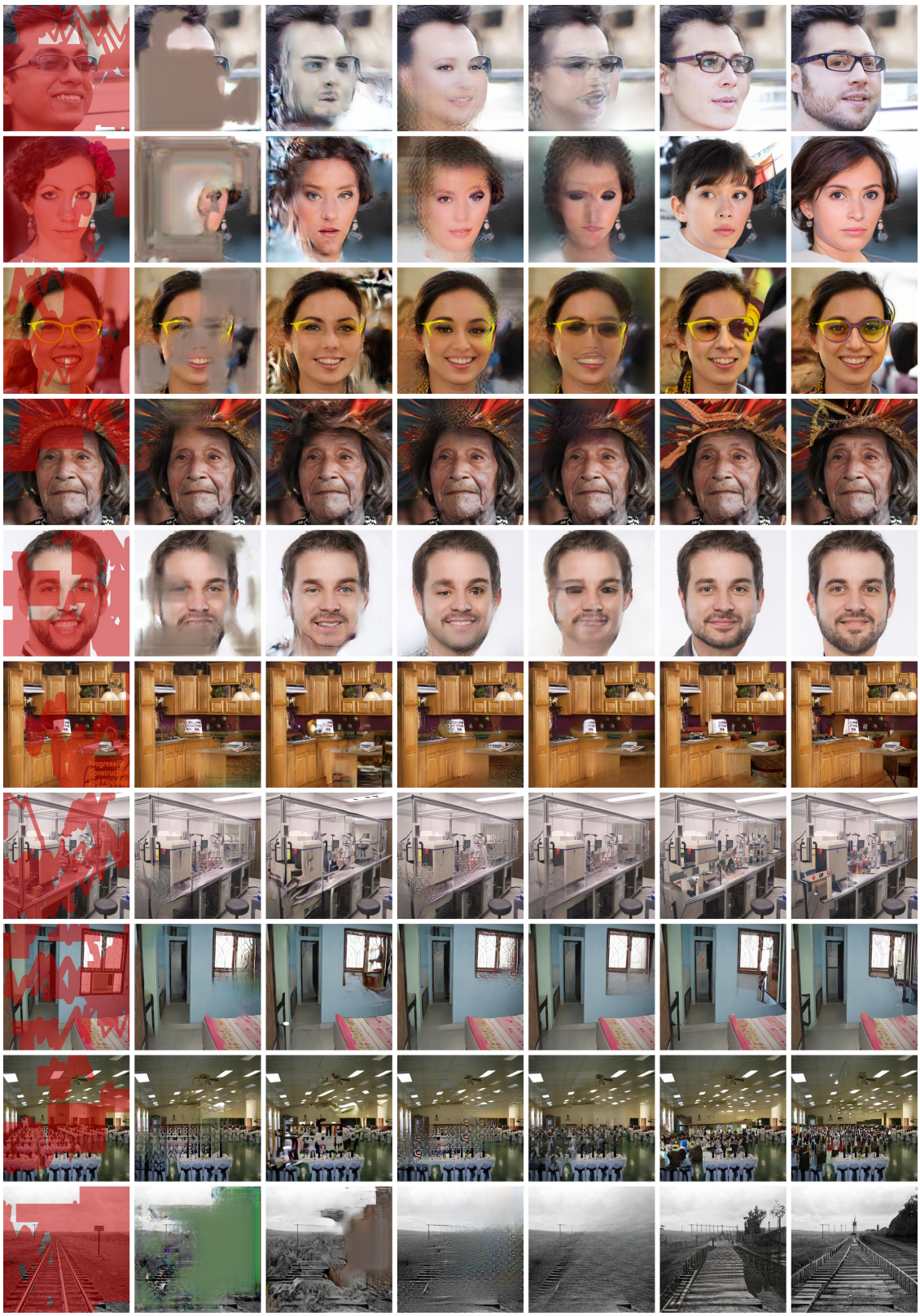


Figure 3: More qualitative examples between prior approaches and SH-GAN using free-form masks. Please zoom in for a better view.

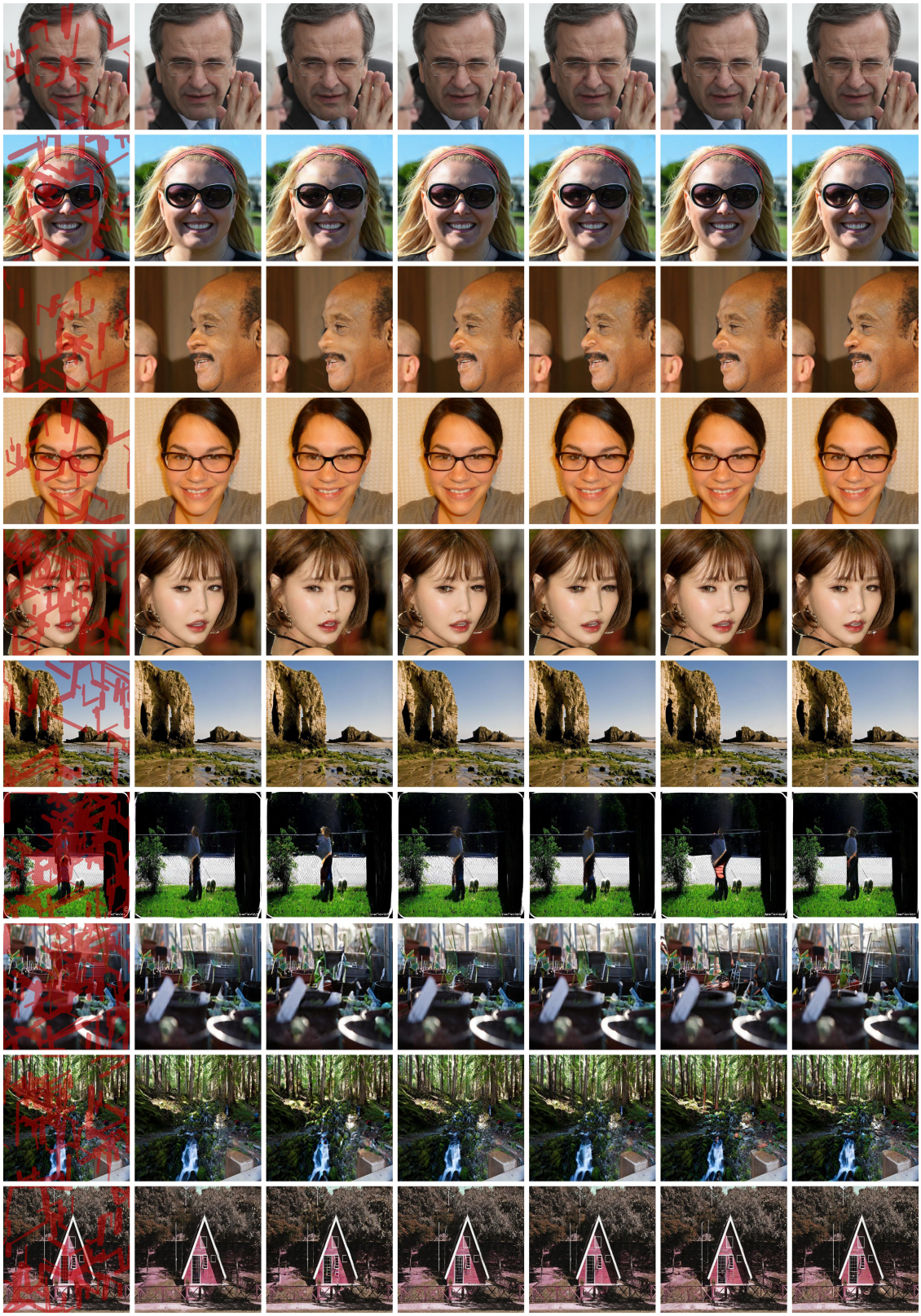


Figure 4: Qualitative examples between prior approaches and SH-GAN using narrow masks. Please zoom in for a better view.

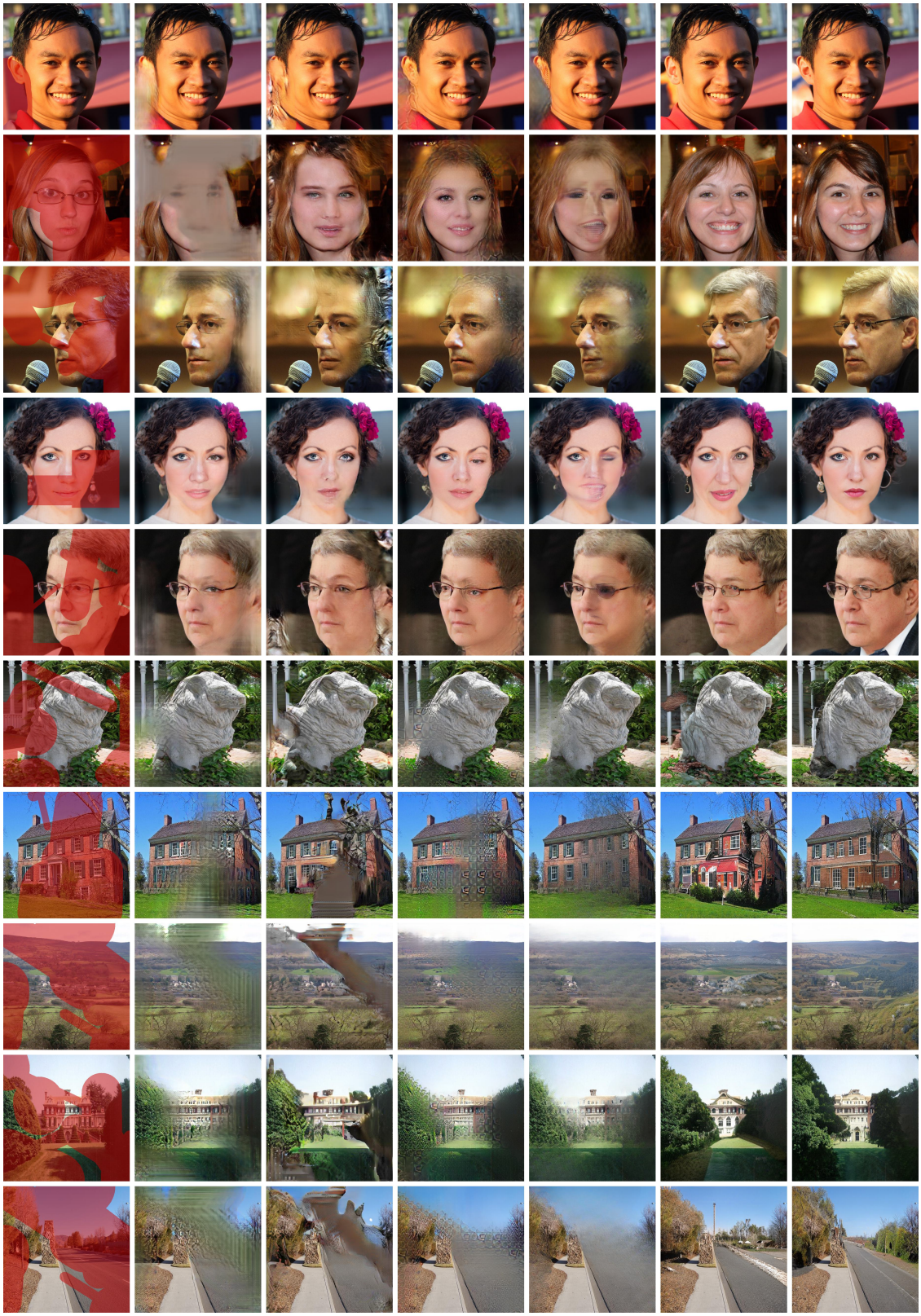


Figure 5: Qualitative examples between prior approaches and SH-GAN using wide masks. Please zoom in for a better view.