3DFaceFill: An Analysis-By-Synthesis Approach to Face Completion

(Supplementary Material)

Rahul Dey Vishnu Naresh Boddeti
Michigan State University
East Lansing, MI
{deyrahul, vishnu}@msu.edu

1. Overview

We organize the supplementary as follows. In section 2, we give further experimental results to support our claims regarding the effectiveness of the proposed method. This includes additional comparisons with baselines that do not have publicly available source codes or pre-trained models, *viz.*, DSA [**?**] and PConv [**?**] (2.1). We also compare our method against UVGAN [**?**] in 2.2 by reformulating it for face completion, as well as comparing the proposed Sym-UNet against UVGAN [**?**] on the task of texture completion. We present additional qualitative evaluation in 2.3, in terms of face completion under diverse conditions (2.3A), robustness to pose and illumination variants (2.3B) and generalization performance on in-the-wild images (2.3C). In 2.4, we discuss the unique capability of 3DFaceFill in terms of completing as well as synthesizing new views of partial faces. Finally, we report additional quantitative comparison in terms of SSIM in 2.5.

In section 3, we perform further ablative analysis of the different components of 3DFaceFill. We study the effect of the proposed iterative refinement procedure in 3.1. We also analyze the effect of symmetry and visualize the gating mechanism used in Sym-UNet to control the propagation of features, as well as the uncertainty map that controls the magnitude of various training losses in 3.2. Finally, we give further implementation details including the network architectures, loss functions, training and computational complexity of our method in section 4.

2. Experimental Results

2.1. Comparison against PConv [**?**] and DSA [**?**]

PConv [**?**] and DSA [**?**] have not released publicly available source codes or pre-trained models. Hence, to compare against them, we obtained face completions for a small set of 14 partial images through correspondence with the respective authors[1]. We show qualitative results in Fig. 1. One can observe that while PConv [**?**] and DSA [**?**] tend to deform the facial components under certain conditions leading to geometric and photometric artifacts, 3DFaceFill is free of such artifacts and generates more realistic completions. In addition, we provide quantitative metrics on this small set in Table 1, where 3DFaceFill reports better PSNR, SSIM and LPIPS [**?**] metrics over both the baselines.

	PSNR (↑)	**SSIM** (↑)	**LPIPS** [**?**] (↓)
DSA [**?**]	28.6205	0.9375	0.0436
PConv [**?**]	29.3067	0.9479	0.0379
3DFaceFill	**31.8823**	**0.9615**	**0.0335**

Table 1: **Quantitative comparison** of the proposed 3DFaceFill *vs.* PConv [**?**] and DSA [**?**] on a small set of completed images obtained from the authors.

2.2. Comparison against UVGAN [**?**]

The proposed face completion method, 3DFaceFill, has three parts, (i) disentangling 2D image into factors such as 3D pose, 3D shape, albedo and illumination (*IL*), (ii) enforcing symmetry in UV albedo (*SYM*), and (iii) iterative refinement of face completion through progressively more accurate 3D pose and shape estimation (*IR*). UVGAN [7] on the other hand, (i) performs completion of the missing texture in the UV-representation due to self-occlusion instead of completing a partial face image itself, (ii) unlike 3DFaceFill, does not disentangle texture further into albedo and illumination, (iii) does not impose symmetry prior on the UV texture, and (iv) uses 3DMM on a fully visible face image rather than a partial image to obtain texture. Since no source code or pretrained model of UVGAN is available, we evaluate these differences in two ways: (A) by reformulating

[1]The images provided by PConv's authors were obtained from a model trained on 512x512 sized images, *vs.* 256x256 for the other baselines including 3DFaceFill.

Figure 1: **Qualitative evaluation** of 3DFaceFill *vs.* PConv [**?**] and DSA [**?**] on a subset of images received from the respective authors. The text on the left mention the specific deformities in the baselines (blurriness, artifacts, asymmetry and other geometric deformations), that is not present in the completions by 3DFaceFill.

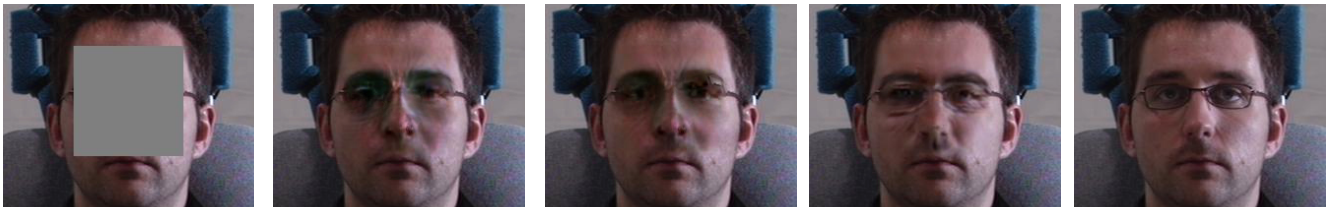
UVGAN for face completion, and (B) comparing UVGAN with our Sym-UNet model on their publicly released texture dataset. We now present the two evaluations.

A: Comparison with UVGAN [?] Reformulated for Face Completion

To simulate UVGAN [**?**] for face completion, we remove the illumination disentanglement (*IL*), symmetry loss (*SYM*) and iterative refinement (*IR*) from 3DFaceFill (refer to Fig. 2). We call the variant with *SYM* as UVGAN-*Sym*, and the variant with both *IL* and *SYM* as 3DFaceFill-*NoIR*. Adding *IR* makes for our full model 3DFaceFill. We compare the above-mentioned variants for face completion on the CelebA [**?**] dataset and report the quantitative and qualitative results in Fig. 2. One can observe that 3DFaceFill *significantly* outperforms UVGAN as well as the other variants both quantitatively as well as qualitatively. Further, we can see that introducing the symmetry loss (*SYM*) in UVGAN-*Sym* hurts performance since, unlike UV-albedo, UV-texture is not inherently symmetric in faces because of the entangled illumination. Completion on the disentangled albedo (*IL*) instead improves performance in 3DFaceFill-*NoIR*. Lastly, iterative refinement (*IR*) further improves completion on top of *IL* and *SYM*. This demonstrates the effectiveness of the novelties that 3DFaceFill introduces over UVGAN [**?**].

B: Sym-UNet *vs.* UVGAN on Texture Completion

Method	IL	SYM	IR	PSNR (↑)	LPIPS (↓)
UVGAN	✗	✗	✗	28.719	0.0383
UVGAN-*Sym*	✗	✓	✗	28.621	0.0392
3DFaceFill-*NoIR*	✓	✓	✗	29.959	0.0334
3DFaceFill	✓	✓	✓	**30.492**	**0.0326**



Input UVGAN UVGAN-Sym 3DFaceFill Ground truth

Figure 2: Comparing UVGAN [**?**] reformulated for face completion *vs.* 3DFaceFill.

In this evaluation, we trained our Sym-UNet model on the UVDB-MPIE texture dataset released by the authors of UVGAN [**?**]. We split the dataset into a 80:20 train-test split and resized the texture maps to 192×256 for training. Similar to UVGAN, we do not include the symmetry loss because of the presence of illumination variations and the availability of synthetically completed texture maps, which reduces the utility of symmetry-loss. The rest of the Sym-UNet is retained as such. On the test set, we report a PSNR of 30.1 (*vs.* UVGAN's 25.8) and SSIM of 0.937 (*vs.* UVGAN's 0.886). Further, we show qualitative results in Fig. 3, where we see that our completed textures resemble the ground truth closely (we do not have the corresponding completions by UVGAN). Thus, our proposed Sym-UNet network is comparatively better suited for UV-completion than the network used in UVGAN [**?**].

2.3. Further Qualitative Evaluation

A: Diverse Conditions

We present further qualitative comparison of face completion by the proposed 3DFaceFill *vs.* DeepFillv2 [**?**] and PIC [**?**] under diverse conditions. Fig. 4 show the qualitative comparison on faces with dark complexion, challenging poses and illumination contrast. Fig. 5 shows examples where baselines tend to introduce asymmetry in eye-gaze or deform various face components, such as, nose, mouth, *etc*. In all these cases, 3DFaceFill generates more realistic completions while preserving the visible illumination contrast and bilateral symmetry, because of the disentangled completion of albedo and explicit enforcement of 3D shape, pose, illumination and symmetric priors.

B: Robustness Across Pose and Illumination Variations

We present further cross-dataset evaluation on the pose and illumination varying MultiPIE dataset [**?**] by splitting the dataset into two subsets: (1) a pose varying subset with constant frontal illumination and expression, referred to as MultiPIE:Pose and (2) an illumination varying subset with constant frontal pose and expression, referred to as MultiPIE:Illu. Table 2 reports the PSNR, SSIM and LPIPS [**?**] metrics for all the methods on these two splits. It can be seen that 3DFaceFill significantly outperforms the baselines in both the splits. Further, we show more example completions by 3DFaceFill *vs.* the baselines DeepFillv2 [**?**] and PIC [**?**] in Fig. 6 (for Pose) and Fig. 7 (for Illumination), respectively. From Fig. 6, one can observe that the baselines tend to generate fuzzy and deformed faces for extreme poses while 3DFaceFill generates sharper and geometry-preserving completions. And, in the illumination-varying case, DeepFillv2 [**?**] tends to generate artifacts and PIC [**?**] tends to generate asymmetric completions for extreme illumination, whereas the completions by 3DFaceFill are free of such artifacts and preserve illumination contrast and symmetry.

C: Generalization Performance on In-the-Wild Images downloaded from the Internet

To compare the generalization performance of different methods, we evaluate face completion on a small dataset of $\sim$ 50 in-the-wild face images downloaded from the internet[2] (referred to as Internet). We report the quantitative metrics in Table 2, where one can see significant margins between 3DFaceFill and the closest baselines across all the three metrics, demonstrating the better generalization performance of our proposed method. Fig. 8 shows qualitative comparison on a small sample where 3DFaceFill generates more realistic completions, thanks to the explicit imposition of 3D face priors. This shows that the principles behind 3DFaceFill can improve the generalization performance of

[2]Source: `https://unsplash.com/s/photos/face`

(a) Input (b) 3DFaceFill (c) Groundtruth

Figure 3: Qualitative evaluation of texture completion by the proposed Sym-UNet on the UVDB-MPIE dataset [**?**].

Dataset	**Metric**	GFC [**?**]	SymmFC [**?**]	DeepFillv2 [**?**]	PIC [**?**]	3DFaceFill
MultiPIE:Pose	**PSNR** (↑)	24.7557	24.7177	26.3385	26.4301	**27.8226**
	SSIM (↑)	0.9187	0.9289	0.9383	0.9451	**0.9482**
	LPIPS (↓)	0.0822	0.0692	0.0527	0.0471	**0.0409**
MultiPIE:Illu	**PSNR** (↑)	23.5749	24.4813	26.4981	26.2938	**27.8865**
	SSIM (↑)	0.8676	0.8618	00.8718	0.8825	**0.8935**
	LPIPS (↓)	0.1232	0.0747	0.0640	0.0540	**0.0484**
Internet	**PSNR** (↑)	24.1775	24.2829	26.4957	25.6326	**28.8463**
	SSIM (↑)	0.9042	0.9168	0.9293	0.9317	**0.9526**
	LPIPS (↓)	0.0913	0.0625	0.0493	0.0466	**0.0390**

Table 2: Further quantitative evaluation of 3DFaceFill *vs.* the baselines on the pose-varying (MultiPIE:Pose) and illumination varying (MultiPIE:Illu) subsets of the MultiPIE dataset [**?**] and in-the-wild images downloaded from the Internet.

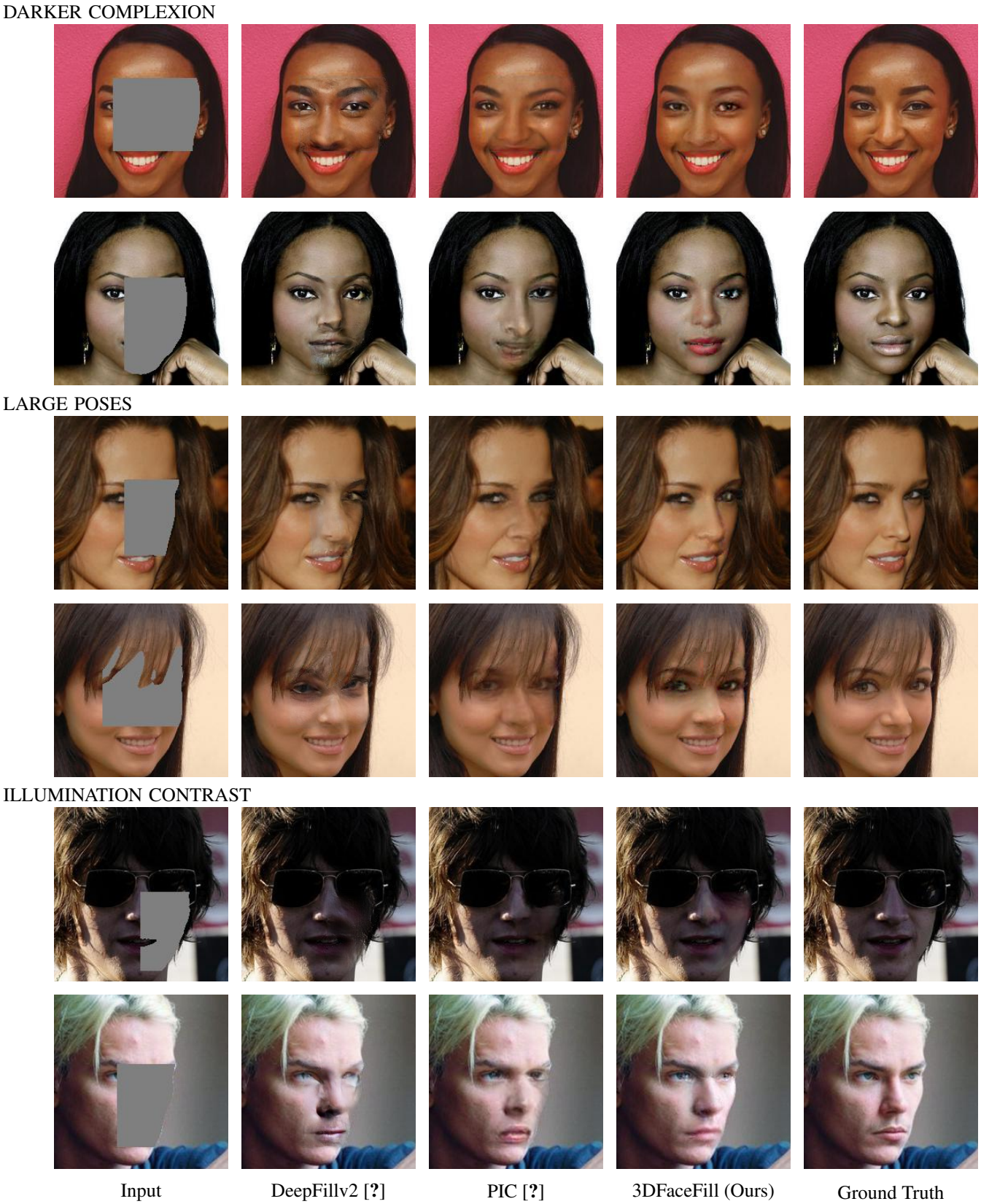
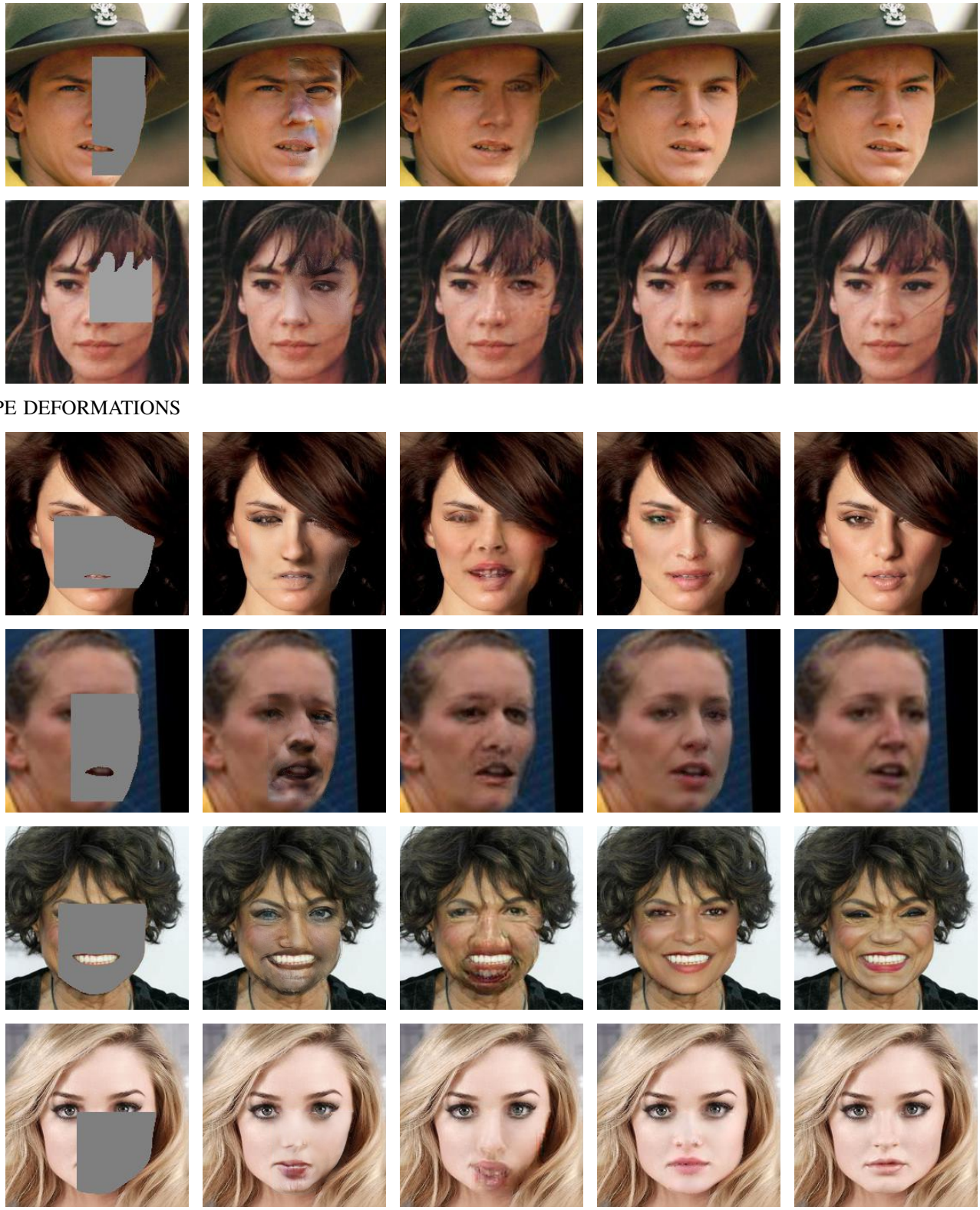


Figure 4: Qualitative evaluation under diverse conditions (complexion, pose, illumination).

ASYMMETRY IN EYE-GAZE

SHAPE DEFORMATIONS



Input | DeepFillv2 [?] | PIC [?] | 3DFaceFill (Ours) | Ground Truth

Figure 5: Qualitative evaluation under diverse conditions (eye-gaze, shape).

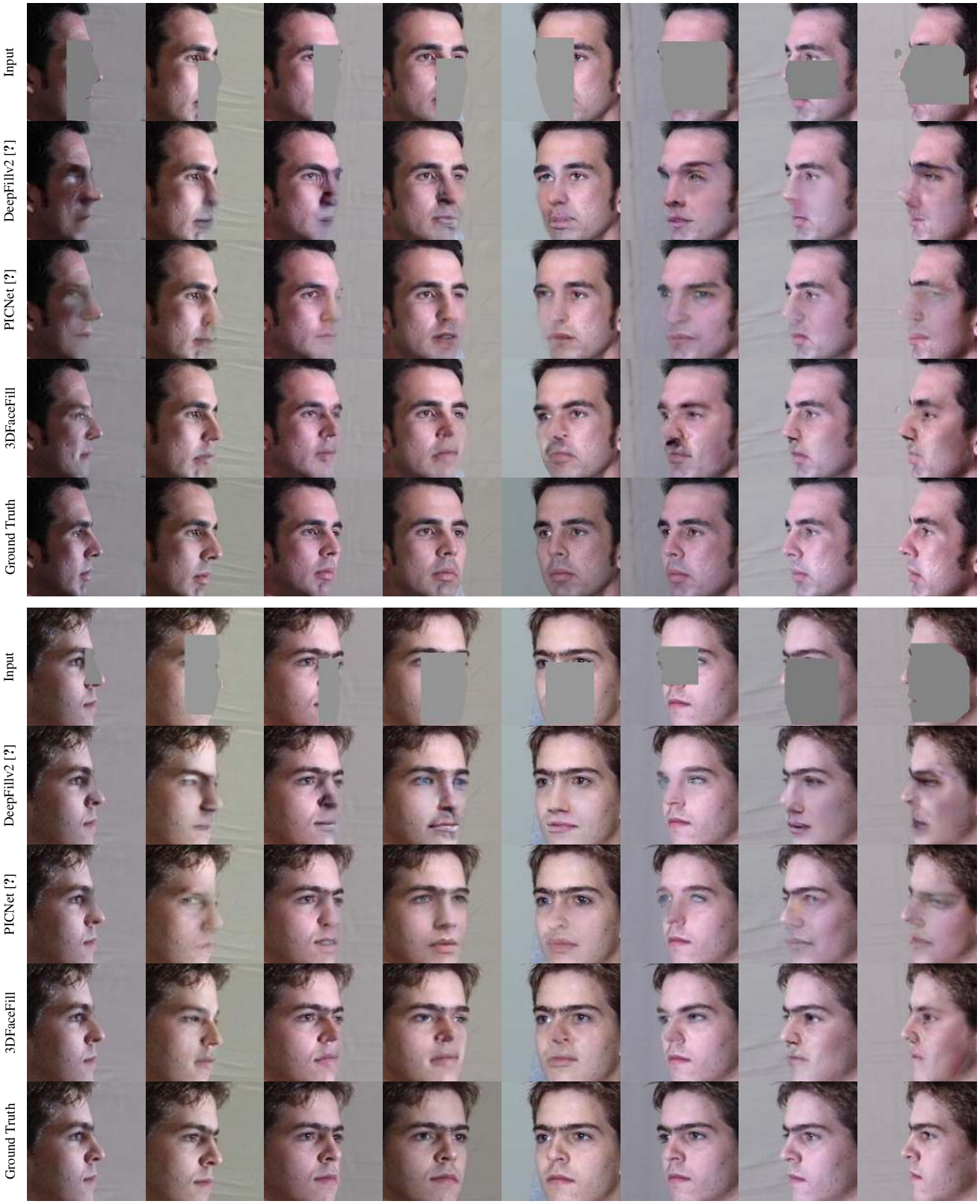


Figure 6: Qualitative evaluation of 3DFaceFill *vs.* baselines DeepFillv2 [**?**] and PIC [**?**] on the pose-varying MultiPIE:Pose split [**?**]. While the baselines tend to generate blurred and deformed faces in extreme poses, 3DFaceFill is pose-robust and generates more accurate completions across a range of pose.

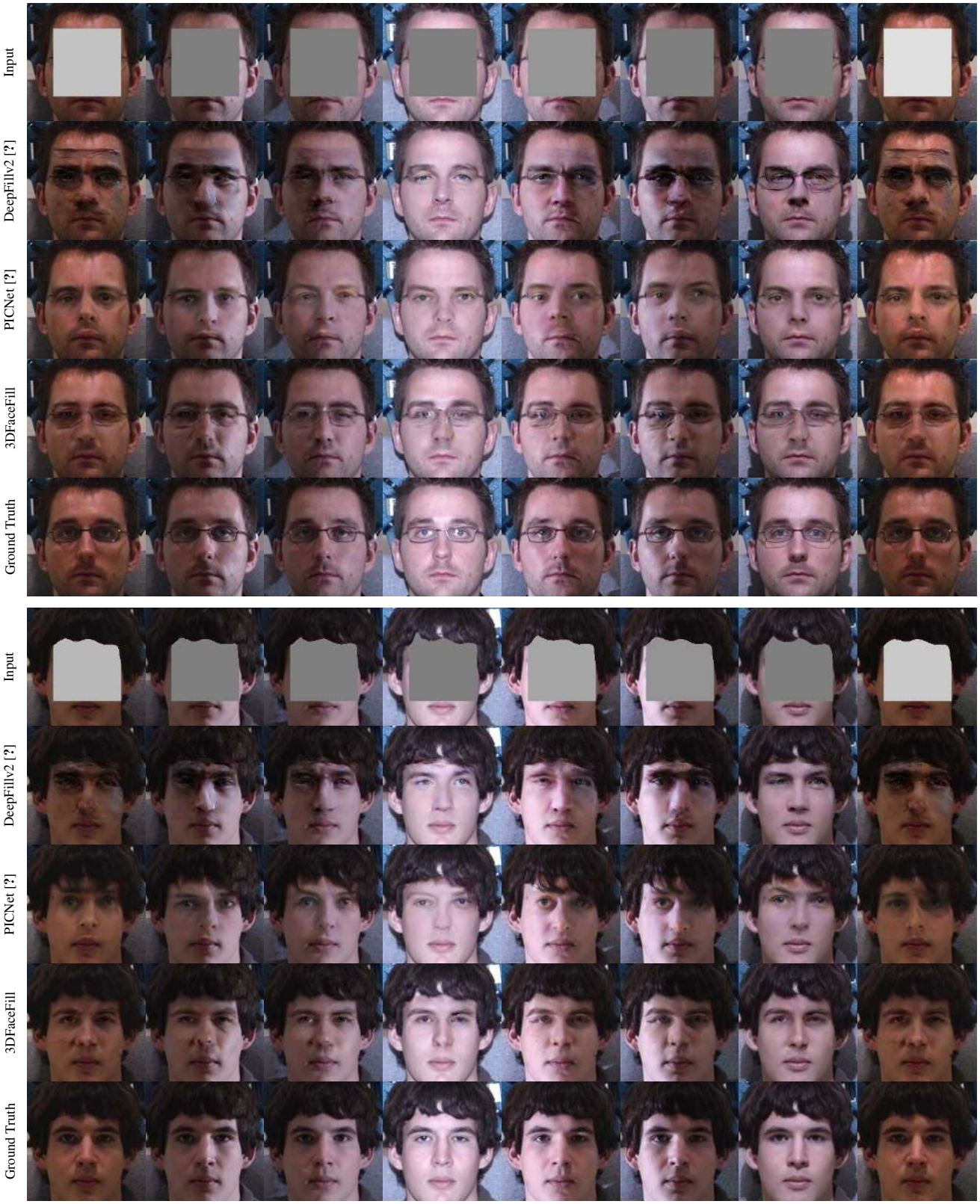


Figure 7: Qualitative evaluation of 3DFaceFill *vs.*the baselines DeepFillv2 [**?**] and PIC [**?**] on the illumination varying MultiPIE:Illu split [**?**]. While the baselines tend to generate artifacts in extreme illuminations, 3DFaceFill generates completions that look geometrically accurate and preserve the illumination contrast (notice (i) the illumination contrast in cols. 2,3,5,6 (b), and (ii) assymetric eye-brows in cols. 1,2,3,5,6 (b) by the baselines.)

Figure 8: Qualitative evaluation (of generalization performance) on the Internet downloaded images.

image completion approaches on structured objects such as faces.

2.4. 3D View Synthesis of Masked Faces

3DFaceFill has a unique advantage over other face completion approaches, in that unlike existing methods, our method can not only complete partial faces, but also render new views of the completed face from different view-points. In Fig. 9, we show this through examples of face views rendered from five different viewpoints by completing the missing albedo and self-occluded regions in the masked faces.

2.5. Evaluation in terms of SSIM

In addition to the PSNR/LPIPS *vs.* mask ratio analysis reported in the main paper, we also report similar comparison in terms of SSIM *vs.* mask ratio on the CelebA [?], CelebA-HQ [?] and MultiPIE [?] datasets in Fig. 10a, Fig. 10b and Fig. 10c, respectively. 3DFaceFill consistently out-performs the baselines in terms of SSIM too for all the mask ratios. Moreover, it can be observed that the comparative gain by 3DFaceFill *vs.* the closest baseline increases as the mask ratio increases, which we attribute to the advantage of using explicit 3D priors for completion.

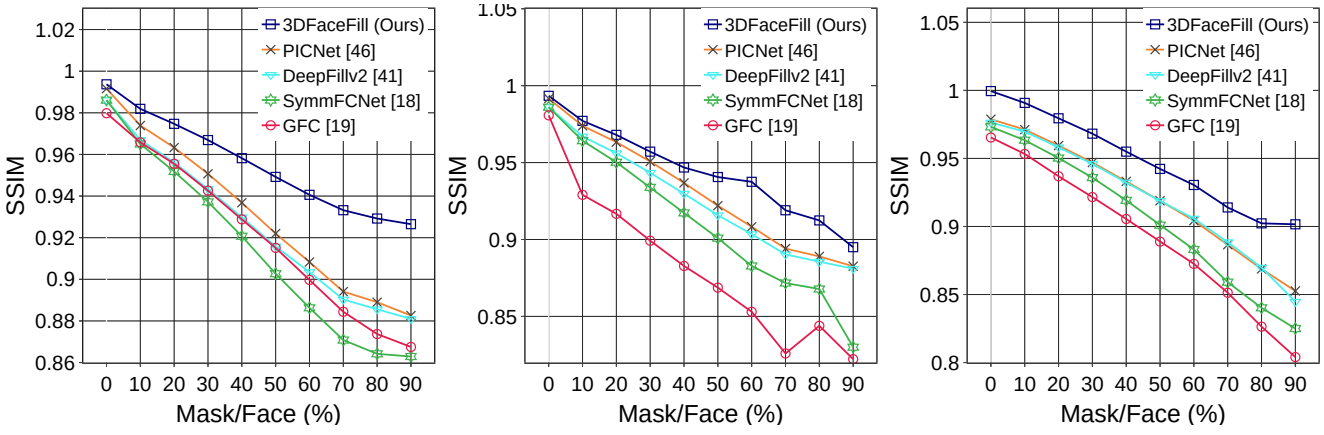
3. Analysis

3.1. Iterative Refinement

As explained in Main:Sec 3.2, we adopt an iterative refinement procedure whereby 3D factorization aids in completion and vice versa. We show heatmap-visualization of the difference between the pre-blended completions at itera-

(a) Input (b) Completed and synthesized face views (c) Ground Truth

Figure 9: **3D Face View Synthesis.** 3DFaceFill has the unique ability to not just complete masked faces realistically, but also synthesize new views from them.



(a) CelebA dataset [**?**] (b) CelebA-HQ dataset [**?**] (c) MultiPIE dataset [**?**]

Figure 10: Quantitative evaluation of 3DFaceFill *vs.* the baselines in terms of SSIM. 3DFaceFill consistently and significantly outperforms the baselines across all the Mask-to-Face area ratios and across all the datasets *viz.* CelebA [**?**], CelebA-HQ [**?**] and MultiPIE [**?**].

Figure 11: **Effect of Iterative Finetuning**. We show raw completions (without blending) at iterations 1 and 2 along with the difference heatmaps. Note the improvements in *Iter2* over *Iter1* and the corresponding heatmap activations around eyes, eye-brows and other edges on the face.

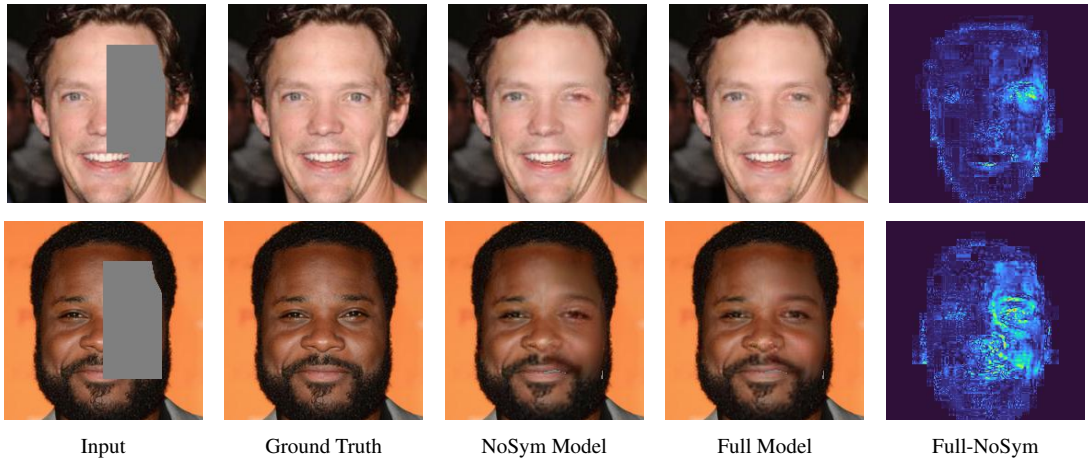


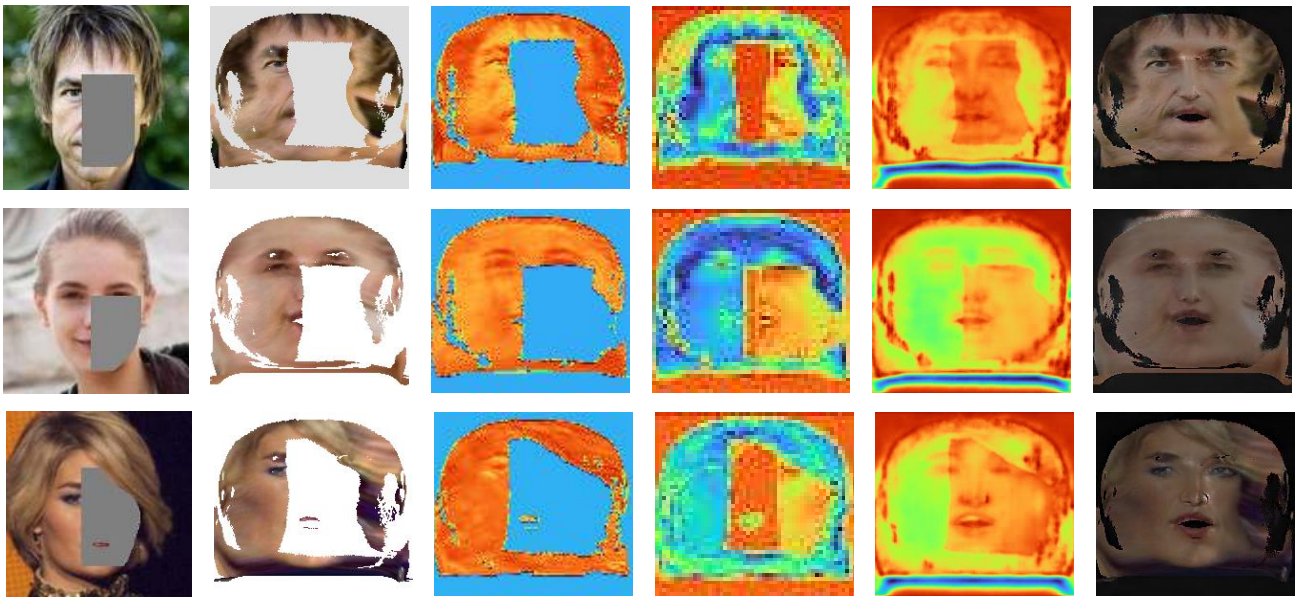
Figure 12: **Effect of using Symmetry**. The full model includes Sym-UNet and symmetry loss (during training) and can copy symmetric features when available. The absolute difference heatmaps (Full-NoSym) shows that most difference is coming from components such as eyes, eye-brows, *etc*.

tions 1 and 2 in Fig. 11. A relative comparison of the distribution of Iter1-GT and Iter2-GT shows that Iter2 is closer to the GT (ground truth) image, which indicates improved 3D pose estimation in the second iteration. The visualization of Iter2-Iter1 shows that these differences are manifesting at the detailed face components such as eyes, nose, *etc*., as well as the masked regions.

3.2. Effect of Symmetry

We further analyze the advantages of enforcing symmetry-consistency. We show completions by the Full model (Sym-UNet + symmetry loss) and the NoSym model (UNet) in Fig. 12. The completions by the NoSym variant look slightly asymmetric because of blurry and symmetry-agnostic completion around the eyes and other masked regions whereas our Full model leverages symmetry to generate sharper and symmetry-consistent completions. This can be further visualized in the Full-NoSym difference heatmaps shown in the figure that show that the difference is mostly concentrated around the eyes and eye-brows and spreads further through the masked regions.

Symmetry Gating Activation: We further visualize the intermediate gating maps used in our model that control the flow of information in the network (ref Fig. 13). We visualize two (out of 64) gating activations (1st - Gate1 and 33rd - Gate2) from the second layer of our Sym-UNet network. As can be seen in Fig. 13, while Gate1 activates for the visible regions in the input albedo, Gate2 activates for the masked regions to propagate useful features from the horizontally flipped albedo map to the symmetric side. This enables Sym-UNet to leverage and maintain facial symmetry for inpainting. We also visualize the estimated uncertainty map (σ) in Fig. 13 that is learned by the inpainter $\mathcal{G}$ in an unsupervised way. Note that the uncertainty is usually higher



(a) Input (b) Input Albedo (c) Gate 1 (d) Gate 2 (e) Uncertainty σ (f) Output Albedo

Figure 13: **Visualizing the Gating Activations and the Uncertainty-Maps**. Observe that, while *Gate 1* activates for the visible regions, *Gate 2* activates for the masked regions to propagate useful features from the visible symmetric parts to their masked counterparts. The uncertainty map captures the model's uncertainty around the masked regions and the facial components such as the eyes, thus incurring higher losses for these regions. (*Note:* higher values are represented by warmer (redish) colors in the gating and uncertainty heatmaps).

around important facial components like the eyes and the masked regions, which increases the loss incurred in these regions.

4. Implementation Details

In this section, we provide further implementation details regarding the proposed approach. In sub-section 4.1, we give detailed network architectures for the modules used in 3DFaceFill. In sub-section 4.2, we provide details of the loss functions used to train the 3D factorization module. Lastly, we give full training details of the different components in sub-section 4.3.

4.1. Network Architectures

We report the detailed network architectures for the 3DMM Encoder $\mathcal{E}$, the Albedo Decoder $\mathcal{D}_A$, the Sym-UNet module, the PyramidGAN discriminator and the Face Segmenter $\mathcal{S}$ in Tables 3 to 7. Our network architectures for the 3DMM modules are based on the architectures used in [**?**] for the corresponding modules. Insipired by Miyato *et al.* [**?**], we use spectral normalization in all our convolution layers. The abbreviated operators used are defined as follows:

- Conv$(c_{in}, c_{out}, k, s, p)$: 2D convolution with c_{in} input channels, c_{out} output channels, kernel size k, stride s and padding p.
- Deconv$(c_{in}, c_{out}, k, s, p)$: 2D transposed convolution (deconvolution) with c_{in} input channels, c_{out} output channels, kernel size k, stride s and padding p.
- GN(n): Group normalization [**?**] with n groups
- ELU: Exponential linear unit [**?**] activation, LReLU(α): Leaky ReLU [**?**] with a negative slope of α
- ResUnit$(c_{in}, c_{out}, k, s, p)$: Residual unit [**?**] with c_{in} input channels, c_{out} output channels, kernel size k, stride s, padding p with group normalization [**?**] and ELU activation [**?**]
- SigGNConv$(c_{in}, c_{out}, k, s, p)$: 2D convolution with c_{in} input channels, c_{out} output channels, kernel size k, stride s and padding p followed by group normalization [**?**] and sigmoid activation
- SigGNDeconv$(c_{in}, c_{out}, k, s, p)$: 2D transposed convolution with c_{in} input channels, c_{out} output channels, kernel size k, stride s and padding p followed by group normalization [**?**] and sigmoid activation

Table 3: Network architecture of the 3DMM Encoder $\mathcal{E}$. The $Pose_1$ corresponds to the scale, $Pose_{2:4}$ correspond to the yaw, roll and pitch angles normalized by $\pi/2$ and $Pose_{5:6}$ correspond to the X and Y translations normalized by the input image size.

3DMM Encoder	Output size
Image → SpectralConv(3, 32, 7, 2, 3) + GN(8) + ELU	112x112
SpectralConv(32, 64, 3, 1, 1) + GN(16) + ELU	112x112
SpectralConv(64, 64, 3, 2, 1) + GN(16) + ELU	56x56
SpectralConv(64, 96, 3, 1, 1) + GN(24) + ELU	56x56
SpectralConv(96, 128, 3, 1, 1) + GN(32) + ELU	56x56
SpectralConv(128, 128, 3, 2, 1) + GN(32) + ELU	28x28
SpectralConv(128, 196, 3, 1, 1) + GN(48) + ELU	28x28
SpectralConv(196, 256, 3, 1, 1) + GN(64) + ELU	28x28
SpectralConv(256, 256, 3, 2, 1) + GN(64) + ELU	14x14
SpectralConv(256, 256, 3, 1, 1) + GN(64) + ELU	14x14
SpectralConv(256, 256, 3, 1, 1) + GN(64) + ELU	14x14
SpectralConv(256, 512, 3, 2, 1) + GN(128) + ELU	7x7
SpectralConv(512, 512, 3, 1, 1) + GN(128) + ELU → *feats*	7x7
feats → SpectralConv(512, 160, 3, 1, 1) + GN(40) + ELU	7x7
AvgPool(7,7)	1x1
Linear(160, 6) + Tanh → *Pose*	
feats → SpectralConv(512, 160, 3, 1, 1) + GN(40) + ELU	7x7
AvgPool(7,7)	1x1
Linear(160, 27) → *Illumination*	
feats → SpectralConv(512, 512, 3, 1, 1) + GN(128) + ELU	7x7
SpectralConv(512, 512, 3, 1, 1) + GN(128) + ELU	7x7
AvgPool(7,7)	1x1
Linear(512, 199+29) → *199 Shape + 29 Expression coefficients*	
feats → SpectralConv(512, 512, 3, 1, 1) + GN(128) + ELU	7x7
AvgPool(7,7) → *Albedo features*	1x1
Model Complexity	17.4M

- SpectralConv$(c_{in}, c_{out}, k, s, p)$: 2D convolution with c_{in} input channels, c_{out} output channels, kernel size k, stride s, padding p and spectral normalization [**?**]
- Upsample(s_h, s_c): Upsamples height by s_h and width by s_w using nearest neighbour interpolation.

4.2. 3DMM Module Losses

The 3DMM module is trained using a combination of supervised, reconstruction and regularization losses:

$$\mathcal{L}_{3DMM} = \lambda_{sup}\mathcal{L}_{sup} + \lambda_{rec}\mathcal{L}_{rec} + \lambda_{reg}\mathcal{L}_{reg}, \quad (1)$$

where, $\mathcal{L}_{sup} = \lambda_S\mathcal{L}(\mathbf{S}, \tilde{\mathbf{S}}) + \lambda_p\mathcal{L}(\mathbf{p}, \tilde{\mathbf{p}}) + \lambda_T\mathcal{L}(\mathbf{T}^{uv}, \tilde{\mathbf{T}}^{uv}) + \lambda_{lmark}\mathcal{L}_{lmark}$ use the groundtruth shape, pose, texture and 2D landmarks when available, $\mathcal{L}_{rec}$ enforces similarity between the rendered and groundtruth images and $\mathcal{L}_{reg} = \lambda_{3dsym}\mathcal{L}_{3dsym} + \lambda_{const}\mathcal{L}_{const}$ are regularization losses to enforce bilateral symmetry of albedo and effective separation of shade and albedo. All loss coefficients λ's are set to have equal weightage for all the loss terms. We now define these losses:

Table 4: Network architecture of the Albedo Decoder $\mathcal{D}_A$ that decodes the 512 dimensional Albedo features from the 3DMM Encoder $\mathcal{E}$ into $3 \times 192 \times 256$ dimensional Albedo representation in the UV space.

Albedo Decoder	Output size
Albedo features → Upsample(3,4)	3x4
SpectralConv(512, 512, 3, 1, 1) + GN(128) + ELU	3x4
SpectralConv(512, 256, 3, 1, 1) + GN(64) + ELU	3x4
Upsample(2,2)	6x8
SpectralConv(256, 256, 3, 1, 1) + GN(64) + ELU	6x8
SpectralConv(256, 128, 3, 1, 1) + GN(32) + ELU	6x8
SpectralConv(128, 128, 3, 1, 1) + GN(32) + ELU	6x8
Upsample(2,2)	12x16
SpectralConv(128, 160, 3, 1, 1) + GN(40) + ELU	12x16
SpectralConv(160, 96, 3, 1, 1) + GN(32) + ELU	12x16
SpectralConv(96, 128, 3, 1, 1) + GN(32) + ELU	12x16
Upsample(2,2)	24x32
SpectralConv(128, 128, 3, 1, 1) + GN(32) + ELU	24x32
SpectralConv(128, 64, 3, 1, 1) + GN(16) + ELU	24x32
SpectralConv(64, 96, 3, 1, 1) + GN(24) + ELU	24x32
Upsample(2,2)	48x64
SpectralConv(96, 96, 3, 1, 1) + GN(32) + ELU	48x64
SpectralConv(96, 64, 3, 1, 1) + GN(16) + ELU	48x64
SpectralConv(64, 64, 3, 1, 1) + GN(16) + ELU	48x64
Upsample(2,2)	96x128
SpectralConv(64, 64, 3, 1, 1) + GN(16) + ELU	96x128
SpectralConv(64, 32, 3, 1, 1) + GN(8) + ELU	96x128
SpectralConv(32, 32, 3, 1, 1) + GN(8) + ELU	96x128
Upsample(2,2)	192x256
SpectralConv(32, 32, 3, 1, 1) + GN(8) + ELU	192x256
SpectralConv(32, 16, 3, 1, 1) + GN(4) + ELU	192x256
SpectralConv(16, 16, 3, 1, 1) + GN(4) + ELU	192x256
Conv(16, 3, 1, 1, 0) + Tanh → *Albedo*	
Model Complexity	5.54M

- Shape loss is defined as:

$$\mathcal{L}(\mathbf{S}, \tilde{\mathbf{S}}) = \mathbb{E}\left[||\mathbf{f}_S - \tilde{\mathbf{f}}_S||_2^2\right],$$

where $\mathbf{f}_S$ and $\tilde{\mathbf{f}}_S$ are the output and groundtruth shape and expression coefficients, respectively.

- Pose loss is defined as a combination of scale, translation and rotation losses:

$$\mathcal{L}(\mathbf{p}, \tilde{\mathbf{p}}) = \lambda_s\mathbb{E}\left[(s - \tilde{s})^2\right] + \lambda_t\mathbb{E}\left[||\mathbf{t}_{x,y} - \tilde{\mathbf{t}}_{x,y}||_2^2\right] + \lambda_r\mathcal{L}_R,$$

where s represents scale, $\mathbf{t}_{x,y}$ represents the X and Y translations, and $\mathcal{L}_R = \mathbb{E}\left[||quat(\mathbf{R}) - quat(\tilde{\mathbf{R}})||_2^2\right]$ is the rotation loss with $\mathbf{R}$ representing the rotation along the X, Y and Z axes and $quat(.)$ gives its quaternion representation.

- Texture loss is defined as:

$$\mathcal{L}(\mathbf{T}^{uv}, \tilde{\mathbf{T}}^{uv}) = \mathbb{E}\left[||\mathbf{T}^{uv} - \tilde{\mathbf{T}}^{uv}||_2^2\right],$$

where $\mathbf{T}^{uv}$ is the texture represented in UV space.

Table 5: Network architecture of the Albedo Inpainter $\mathcal{G}$ (Sym-UNet). The input to the network is the concatenation of the masked Albedo $\mathbf{A}_m^{uv}$ and the mask $\mathbf{M}^{uv}$ in the UV space $X = (\mathbf{A}_m^{uv}, \mathbf{M}^{uv})$. Outputs are the completed Albedo $\hat{\mathbf{A}}^{uv}$ and the uncertainty map σ^{uv}.

Input	Layer	Output
X	ResUnit(4, 32, 3, 2, 1)	$f1$
X	SigGNConv(4, 32, 3, 2, 1)	$g1$
$hflip(X)$	ResUnit(4, 32, 3, 2, 1)	$f1'$
$hflip(X)$	SigGNConv(4, 32, 3, 2, 1)	$g1'$
$(f1 \odot g1, f1' \odot g1')$	ResUnit(64, 64, 3, 2, 1)	$f2$
$(f1 \odot g1, f1' \odot g1')$	SigGNConv(64, 64, 3, 2, 1)	$g2$
$f2 \odot g2$	ResUnit(64, 128, 3, 2, 1)	$f3$
$f2 \odot g2$	SigGNConv(64, 128, 3, 2, 1)	$g3$
$f3 \odot g3$	ResUnit(128, 256, 3, 2, 1)	$f4$
$f3 \odot g3$	SigGNConv(128, 256, 3, 2, 1)	$g4$
$f4 \odot g4$	ResUnit(256, 512, 3, 2, 1)	$f5$
$f4 \odot g4$	SigGNConv(256, 512, 3, 2, 1)	$g5$
$f5 \odot g5$	ResUnit(512, 256, 3, 1, 1)	$f5^1$
$f5 \odot g5$	SigGNConv(512, 256, 3, 1, 1)	$g5^1$
$f5^1 \odot g5^1$	Upsample(2,2)	$x4$
$(x4, f4 \odot g4)$	ResUnit(512, 128, 3, 1, 1)	$f4^1$
$f5^1 \odot g5^1$	SigGNDeconv(256, 128, 4, 2, 1)	$g4^1$
$f4^1 \odot g4^1$	Upsample(2,2)	$x3$
$(x3, f3 \odot g3)$	ResUnit(256, 64, 3, 1, 1)	$f3^1$
$f4^1 \odot g4^1$	SigGNDeconv(128, 64, 4, 2, 1)	$g3^1$
$f3^1 \odot g3^1$	Upsample(2,2)	$x2$
$(x2, f2 \odot g2)$	ResUnit(128, 64, 3, 1, 1)	$f2^1$
$f3^1 \odot g3^1$	SigGNDeconv(128, 64, 4, 2, 1)	$g2^1$
$f2^1 \odot g2^1$	Upsample(2,2)	$x1$
$(x1, f1 \odot g1)$	ResUnit(128, 64, 3, 1, 1)	$f1^1$
$f2^1 \odot g2^1$	SigGNDeconv(128, 64, 4, 2, 1)	$g1^1$
$f1^1 \odot g1^1$	Upsample(2,2)	$x0$
$x0$	ResUnit(64, 32, 3, 1, 1)	$f0^1$
$f1^1 \odot g1^1$	SigGNDeconv(64, 32, 4, 2, 1)	$g0^1$
$f0^1 \odot g0^1$	Conv(32, 4, 1, 1, 0)	$(\hat{\mathbf{A}}^{uv}, \sigma^{uv})$
Model Complexity		11.7M

Table 6: Network architecture of the PyramidGAN discriminator $\mathcal{D}$.

Input	Layer	Output
$\mathbf{I}_{gt}/\tilde{\mathbf{I}}$	SpectralConv(3, 32, 4, 2, 1) + GN(8) + LReLU(.2)	$x0$
$x0$	SpectralConv(32, 64, 4, 2, 1) + GN(16) + LReLU(.2)	$x1$
$x1$	SpectralConv(64, 1, 1, 1, 0)	$out1$
$x1$	SpectralConv(64, 128, 4, 2, 1) + GN(32) + LReLU(.2)	$x2$
$x2$	SpectralConv(128, 1, 1, 1, 0)	$out2$
$x2$	SpectralConv(128, 256, 4, 2, 1) + GN(64) + LReLU(.2)	$x3$
$x3$	SpectralConv(256, 1, 1, 1, 0)	$out3$
$x3$	SpectralConv(256, 512, 4, 2, 1) + GN(128) + LReLU(.2)	$x4$
$x4$	SpectralConv(512, 1, 1, 1, 0)	$out4$
Model Complexity		2.79M

- Landmark loss is defined as:

$$\mathcal{L}_{lmark} = \left\| \mathbf{M}(\mathbf{p}) * \begin{bmatrix} \mathbf{S}(:, \mathbf{d}) \\ \mathbf{1} \end{bmatrix} - \mathbf{U} \right\|_2^2,$$

where $\mathbf{M}$ is the camera projection matrix obtained from the pose $\mathbf{p}$, $\mathbf{d}$ selects 68 indices corresponding to sparse 2D landmarks on the 3D face mesh $\mathbf{S}$ and $\mathbf{U} \in \mathbb{R}^{2\times 68}$ are the groundtruth locations of 2D facial landmarks.

Table 7: Network architecture of the face segmenter $\mathcal{S}$. (x, y) represents the concatenation of tensors x and y along the channel dimension. The output of the network consist of a face mask $\mathbf{M}_f$, an occlusion mask $\mathbf{M}_o$ and a background mask $\mathbf{M}_b$.

Input	Layer	Output
Image	ResUnit(3, 32, 3, 1, 1)	$x1$
$x1$	ResUnit(32, 64, 3, 2, 1)	$x2$
$x2$	ResUnit(64, 128, 3, 2, 1)	$x3$
$x3$	ResUnit(128, 256, 3, 2, 1)	$x4$
$x4$	ResUnit(256, 256, 3, 2, 1)	$x5$
$x5$	ResUnit(256, 256, 3, 2, 1)	$x6$
$x6$	Upsample(2,2)	$x5^1$
$(x5^1, x5)$	ResUnit(512, 256, 3, 1, 1)	$x5^2$
$x5^2$	Upsample(2,2)	$x4^1$
$(x4^1, x4)$	ResUnit(512, 128, 3, 1, 1)	$x4^2$
$x4^2$	Upsample(2,2)	$x3^1$
$(x3^1, x3)$	ResUnit(256, 64, 3, 1, 1)	$x3^2$
$x3^2$	Upsample(2,2)	$x2^1$
$(x2^1, x2)$	ResUnit(128, 32, 3, 1, 1)	$x2^2$
$x2^2$	Upsample(2,2)	$x1^1$
$(x1^1, x1)$	ResUnit(64, 32, 3, 1, 1)	$x1^2$
$x1^2$	Conv(32, 3, 1, 1, 0) + Softmax2d	$(\mathbf{M}_f, \mathbf{M}_o, \mathbf{M}_b)$
Model Complexity		7.18M

- Reconstruction is defined as:

$$\mathcal{L}_{rec} = \mathbb{E}\left[||\mathbf{I} - \hat{\mathbf{I}}||_1 \right],$$

where $\mathbf{I}$ and $\tilde{\mathbf{I}}$ are the rendered and ground-truth images, respectively.

- Albedo symmetry loss is defined as:

$$L_{3dsym}(\mathbf{A}) = \|\mathbf{A}^{uv} - hflip(\mathbf{A}^{uv})\|_1,$$

where $\mathbf{A}^{uv}$ is the UV representation of albedo and hflip() is the horizontal image flipping operation.

- Albedo constancy loss is defined as:

$$L_{const}(\mathbf{A}) = \sum_{\mathbf{v}_j^{uv} \in \mathcal{N}_i} \omega(\mathbf{v}_i^{uv}, \mathbf{v}_j^{uv}) \|\mathbf{A}^{uv}(\mathbf{v}_i^{uv}) - \mathbf{A}^{uv}(\mathbf{v}_j^{uv})\|_2^p,$$

where $\mathcal{N}_i$ denotes the 4-neighborhood around $\mathbf{v}_i^{uv}$ and the weight $\omega(\mathbf{v}_i^{uv}, \mathbf{v}_j^{uv}) = exp(-\alpha \|c(\mathbf{v}_i^{uv}) - c(\mathbf{v}_j^{uv})\|)$ enforce that pixels with similar chromaticity should have similar albedo.

4.3. Training Details

3DMM Module: We train the 3DMM module in two stages. First, we train it using the 300W-3D dataset [**?**],

which has ground-truth shape, pose, texture and landmark annotations, for 100k iterations in a supervised way. Then, we further train it on the CelebA dataset [**?**] with 1/10th of the original learning rate for further 30k iterations in an unsupervised way, whereby we use only the reconstruction loss, 2D landmark loss and the regularization losses. During this stage, we use landmark detections from HRNet [**?**] as groundtruth for the landmark loss. To make the 3DMM encoder robust to partial face images, we introduce artificial occlusions in the training images using random rectangular masks of varying sizes and locations. In addition, we also use random horizontal flipping as a data augmentation. During inference, occlusions are removed from the input image using the occlusion mask and passed through the 3DMM encoder to obtain occlusion-robust factorization.

Albedo Inpainting Module: The albedo inpainting module is trained on the CelebA dataset [**?**] for 30k iterations. To obtain the UV representations of the partial albedo and the mask, we re-project the 3D mesh obtained from the pretrained 3DMM module on the partial image and mask, respectively as shown in Main:Fig.3.b. On the GAN loss (Main:Eqn.2), we update the inpainter $\mathcal{G}$ and the discriminator $\mathcal{D}$ alternatively using a ratio of 1:1. On all the other completion losses, we update the inpainter $\mathcal{G}$ continuously. Other than the random face masks, we use random horizontal flipping as the only data augmentation to train the albedo inpainter.

Face Segmentation Module: Since our method inpaints only the facial region in the UV domain, we restrict the image masks to lie on the face region too. For this, we train a UNet [**?**] based face segmentation model that separates the face region from the background, hair and inner mouth. The face segmenter predicts segmentation masks for (a) the face, (b) hair and other occlusions and (c) the background. We train the face segmentation module on the CelebAMask-HQ dataset [**?**] for a total of 50k iterations using the ground-truth annotations provided by the dataset. We use Focal loss [**?**] to train this module.

For all the modules, except the discriminator $\mathcal{D}$, we use the Adam optimizer with an initial learning rate of 10^{-4} and a step-decay of 0.98 per epoch, while for the PyramidGAN discriminator, we use an initial learning rate of 3×10^{-4}. The input images are first aligned to 256×256 using the method suggested in [**?**], which is the alignment used in the CelebA-HQ dataset. For training, we randomly crop the images to a size of 224×224 while during inference we use central crop. The full training takes 2 days on an Intel Xeon E5-2650 machine with two NVIDIA RTX 2080 GPUs, while inference takes 0.1 sec per image on a single GPU.