

Supplementary Document

GVP: Generative Volumetric Primitives

Mallikarjun B R ^{1,2}, Xingang Pan ¹, Mohamed Elgharib ¹, Christian Theobalt ^{1,2},

¹ Max Planck Institute for Informatics ² Saarland University

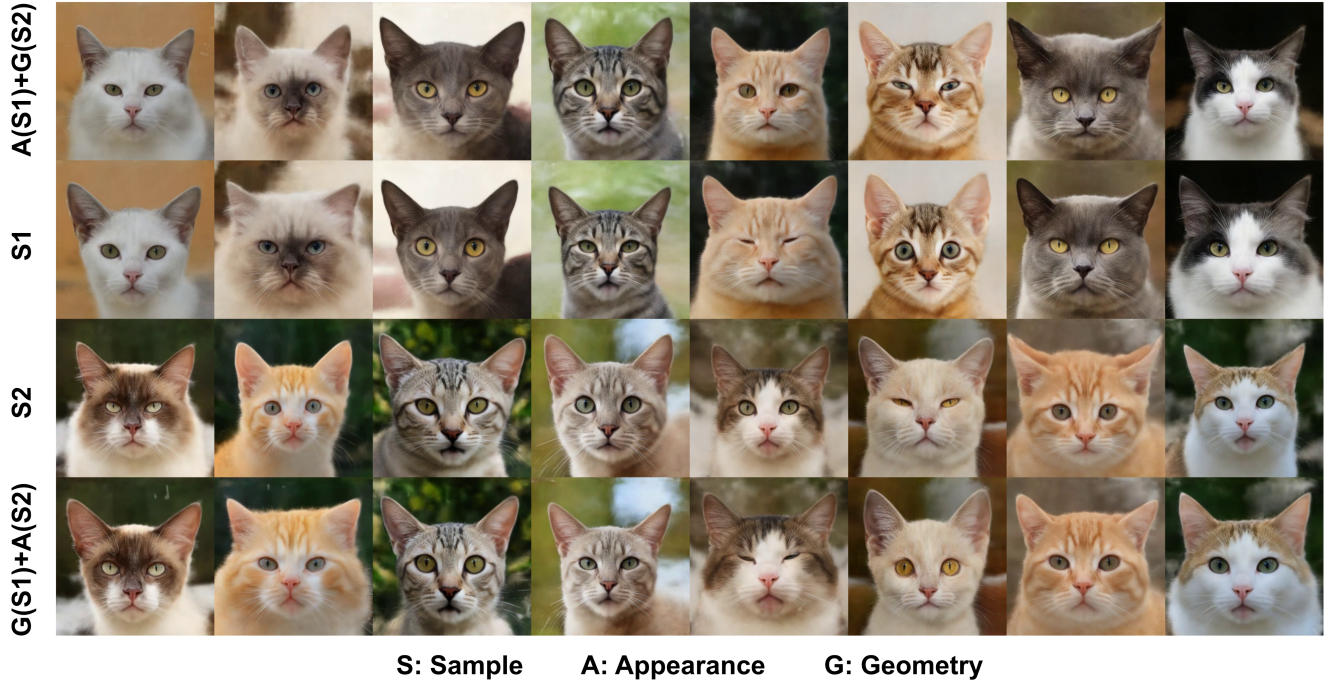


Figure 1. Here we show the quality of correspondence learned using appearance and geometry transfer between 2 samples. S1 and S2 are 2 random samples (second and third rows). We take the color and density output (denoted A) corresponding to a sample and primitives’ spatial information (denoted G) corresponding to the other to render the final result (first and fourth rows).

This document provides more details about our training details, including network architecture and more results. We encourage the readers to watch the supplemental video, where the quality of correspondences learned by our model are better appreciated in the interpolation, and appearance/shape transfer results.

1. Network Architecture

We provide the detailed network architectures of geometry, color, and density generators in Tab. 1, 2, and 3 respectively.

2. Results

In this section, we provide more results on correspondences learned by our model and more samples from the learned

model.

2.1. Correspondences

Our model consists of a fixed number of primitives. These primitives’ spatial information is also learned along with its contents. As these primitives occupy very sparse regions in 3D space, we surprisingly found that each primitive occupies semantically similar parts across different samples. This happens because it is easier for the network to adjust the spatially adjust position, orientation, and scale of the primitives than to synthesize new content. As explained in main paper, we can use this information to obtain dense correspondences between all the samples. In the following, we provide various ways to qualitatively evaluate the learned correspondences.

2.1.1 Interpolation

The learned dense correspondences between samples can be observed in latent interpolation results. We provide video results of latent interpolation, where we sample a set of latent samples and visualize the linearly interpolated samples between them. Along with rendered samples, we provide 4 other types of visualization to showcase the quality of learned correspondences.

1. **Tracked Landmarks:** We obtain set of 2D landmarks for the first sample using off-the-shelf detector [1] and track these landmarks in the rest of samples. One can observe these tracked landmarks faithfully occupy corresponding semantic parts in the rest of the samples, even with diverse expressions.
2. **Primitives:** We also visualize the primitives. One can also observe the smooth change in the primitives' spatial information and primitives consistently occupy similar semantic parts across different samples.
3. **UV Map:** We also provide dense tracking of all the pixels from the first sample (reference) across rest of the samples using reference image.
4. **Dense Correspondence Flow:** We also visualize the magnitude and direction of the tracked pixels from reference sample similar to optical flow visualization.

We strongly recommend the readers watch the supplemental video for the interpolation results.

2.1.2 Shape and Appearance Transfer

We can observe correspondence between samples by also doing appearance and shape transfer between 2 samples. In specific, let w_1 and w_2 be 2 latent vectors of 2 different samples. We take the color and density output corresponding to w_1 and primitives spatial information (i.e., the output of geometry generator) corresponding to w_2 to render the final result. We provide such a result in the video where we keep w_1 fixed and smooth interpolate w_2 between random samples. Fig 1 provides a snapshot of such a result. We strongly recommend readers watch the supplemental video.

2.2. More samples

We provide uncured samples of our model in Fig. 2 and 3. We also provide novel view renderings of several samples in the supplemental video. We strongly recommend readers watch the supplemental video.

2.3. Comparison with EG3D

We also provide video comparison to EG3D [2] by providing novel view renderings of multiple samples in training camera distribution and out-of-training-camera distribution for our and EG3D methods. As EG3D relies on 2D super-resolution, it cannot guarantee multi-view consistency. We strongly recommend readers watch the supplemental video.

Input	Layer	Activation	Output Dim.	KS, Stride
w	Linear	LeakyReLU	1024	
-	Reshape (:, 1024, 1, 1)	-	-	-
-	De-Conv	LeakyReLU	512	$4 \times 4, 2$
-	De-Conv	LeakyReLU	512	$4 \times 4, 2$
-	De-Conv	LeakyReLU	256	$4 \times 4, 2$
-	De-Conv	LeakyReLU	256	$4 \times 4, 2$
-	De-Conv	-	9	$4 \times 4, 2$
-	Reshape (:, 9, 32, 32)	-	-	-

Table 1. This table provides the network architecture of our Geometry Generator. Here KS denotes kernel size.

Input	Layer	Activation	Output Dim.	KS, Stride
w, d	Linear	LeakyReLU	4096	
-	Reshape (:, 256, 4, 4)	-	-	-
-	De-Conv	LeakyReLU	256	$4 \times 4, 2$
-	De-Conv	LeakyReLU	256	$4 \times 4, 2$
-	De-Conv	LeakyReLU	64	$4 \times 4, 2$
-	De-Conv	LeakyReLU	64	$4 \times 4, 2$
-	De-Conv	LeakyReLU	64	$4 \times 4, 2$
-	De-Conv	LeakyReLU	32	$4 \times 4, 2$
-	De-Conv	LeakyReLU	32	$4 \times 4, 2$
-	De-Conv	-	96	$4 \times 4, 2$
-	Reshape (:, 3, 32 ³)	-	-	-

Table 2. This table provides the network architecture of our Color Generator. Here KS denotes kernel size.

References

- [1] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230,000 3d facial landmarks). In *International Conference on Computer Vision*, 2017.
- [2] Eric R. Chan, Connor Z. Lin, Matthew A. Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas Guibas, Jonathan Tremblay, Sameh Khamis, Tero Karras, and Gordon Wetzstein. Efficient geometry-aware 3D generative adversarial networks. In *CVPR*, 2022.



Figure 2. Here we show uncured samples of our method trained on the FFHQ dataset. We also provide renderings of primitives. The color coding of primitives is consistent across different samples. Please note the semantically similar parts are occupied by the same colored primitives across different samples, which provides correspondence information.



Figure 3. Here we show uncured samples of our method trained on the FFHQ dataset. We also provide renderings of primitives. The color coding of primitives is consistent across different samples. Please note the semantically similar parts are occupied by the same colored primitives across different samples, which provides correspondence information.

Input	Layer	Activation	Output Dim.	KS, Stride
w	Linear	LeakyReLU	4096	
-	Reshape (:, 256, 4, 4)	-	-	-
-	De-Conv	LeakyReLU	256	$4 \times 4, 2$
-	De-Conv	LeakyReLU	256	$4 \times 4, 2$
-	De-Conv	LeakyReLU	64	$4 \times 4, 2$
-	De-Conv	LeakyReLU	64	$4 \times 4, 2$
-	De-Conv	LeakyReLU	64	$4 \times 4, 2$
-	De-Conv	LeakyReLU	32	$4 \times 4, 2$
-	De-Conv	LeakyReLU	32	$4 \times 4, 2$
-	De-Conv	-	32	$4 \times 4, 2$
-	Reshape (:, 1, 32^3)	-	-	-

Table 3. This table provides the network architecture of our Density Generator. Here KS denotes kernel size.