Pixel Codec Avatars: Supplemental Document

Shugao Ma Tomas Simon Jason Saragih Dawei Wang
Yuecheng Li Fernando De La Torre Yaser Sheikh
Facebook Reality Labs Research
{shugao, tsimon, jsaragih, dawei.wang, yuecheng.li, ftorre, yasers}@fb.com

A. More Qualitative Results

Please see the supplemental video for results. It consists of two parts. The *first part* shows the decoding result of our method with PiCA (Full) at **novel viewing directions and distances that are unseen during training**, for a sentence-reading *test* sequence. In the video, one can see that the rendered face is highly photorealistic, even though both the viewing direction and distance, as well as the facial expressions are all unseen during training. Note that while there are some visible artifacts in the eye region, achieving photorealistic looking eye appearance is a very challenging task [2], and having explicit eye modeling is future work. The *second part* shows some *test* expressions at novel test views. We also flip to the rendering results by the baseline method for comparison at certain frames. In particular one can see the teeth, tongue and month shape are all better reconstructed by PiCA. Note the video is compressed due to submission file size limit.

B. Architecture Details

B.1. Encoder and Decoder Architectures

Encoder The encoder consists of three major components: the *tex-head*, *geom-head* and the *tex-geom-encoder*. The *tex-head* has two blocks of *conv+leakyrelu*, where the *conv* for both layers have kernel size 4, stride 2. The first one has 512 output channels, and the second has 256 channels. The *geom-head* has one block of *conv+leakyrelu* where the *conv* has kernel size 1, stride 1 and output channel number 256. The output of *tex-head* and *geom-head* are both 256x256x256, and are concatenated and passed to *tex-geom-encoder*, which has 5 blocks of *conv+leakyrelu*. The kernel size and stride are all 4 and 2 for all *conv*, while the output channel numbers are 128, 64, 32, 16 and 8 respectively. The output of *tex-geom-encoder* is further passed to two separate 1x1 *conv* layers to produce mean and variance. *leakyrelu* always having leaky threshold set to 0.2.

Per-Object Decoder This decoder decodes the local expression code and the dense mesh from the latent code which is of dimension 8x8x4. It consists the *geometry decoder*, containing 5 blocks of the building block showing in Fig. 3b in the main text, with output channel numbers 32, 16, 16, 8, 3 respectively. The output size is 256x256x3, from which the dense mesh can be retrieved using the uv coordinates of the mesh vertices. The *expression decoder* takes the concatenated latent code and view direction as input, which is of size 8x8x7, and it contains 5 building blocks as showing in Fig. 3b as well, with output channel numbers 32, 16, 16, 8, 4 respectively. Note in both cases the first *conv* in the block in Fig. 3b has a per-channel per-spatial location bias parameter, following [1].

Pixel Decoder The entries in the 2D and 1D encoding maps in the pixel decoder are initialized to have uniform distribution in the range [-1, 1]. The 3D coordinate input (x,y,z) are first converted to a 4-dimensional vector via a two layer SIREN with output channel numbers 4 and 4 respectively, and then it is concatenated with the encoded uv (8-dimension) and the local expression code to form a 16 dimensional input to the final SIREN. The final SIREN has 4 layers with output channel numbers 8, 8, 8, 3 respectively to compute the RGB color at a pixel.

B.2. Geometric Smoothness

To recap, $\mathbf{G} \in \mathbb{R}^{w \times w \times 3}$, with w=256, is a decoded position map describing the geometry, and $S(\cdot) : \mathbb{R}^{w \times w \times 3} \rightarrow \mathbb{R}^{N_V \times 3}$ is a function that bilinearly interpolates the position map at the vertex uv locations to produce face-centric xyz locations for the set of N_V mesh vertices, where N_V is the number of vertices in a fixed mesh topology. Our geometric smoothness regularization term $\mathcal{L}_S$ combines two common gradient-based smoothness energies,

$$\mathcal{L}_S = \lambda_g \left[||\mathcal{D}_x(\mathbf{G})||_2 + ||\mathcal{D}_y(\mathbf{G})||_2 \right] \quad (1)$$
$$+ \lambda_l ||\mathbf{W}_L \mathbf{L}(S(\mathbf{G}) - \mathbf{V}_\mu)||_2, \quad (2)$$

where we identify:

Gradient Smoothness. Linear operators $\mathcal{D}_*$ compute the x and y derivatives of the position map using finite differences. These terms prevent large changes across neighboring texels in the position map itself.

Mesh Laplacian. The linear operator $\mathbf{L} \in \mathbb{R}^{N_V \times N_V}$ represents the mesh Laplacian discretized using cotangent weights [3] computed on the coarse neutral input mesh. Here, $\mathbf{V}_\mu \in \mathbb{R}^{N_V \times 3}$ is a mean face mesh used as a regularization target. The diagonal matrix $\mathbf{W}_L \in \mathbb{R}^{N_V \times N_V}$ weights the regularization on hair and mouth vertices at 1.25 and the remaining vertices at 0.25. This regularization prevents the differential mesh coordinates (as computed by the mesh Laplacian) from deviating excessively from the regularization target.

The regularization target $\mathbf{V}_\mu$ is initialized with the coarse neutral mesh geometry. However, because the coarse geometry lacks detail in the mouth, hair, and eye regions, using it as a regularization target tends to oversmooth these areas. Therefore, we update the target on the fly during training using exponential smoothing, obtaining a slowly-changing, moving average estimate of the mean face geometry at dense resolutions. At every SGD iteration, we update $\mathbf{V}_\mu$ as follows:

$$\mathbf{V}_\mu \leftarrow (1 - \lambda_\mu)\mathbf{V}_\mu + \lambda_\mu \frac{1}{B} \sum_{b=1}^{B} S(\mathbf{G}_b), \tag{3}$$

where $\lambda_\mu = 1e^{-4}$ and $b \in \{1 \ldots B\}$ iterates over samples in the SGD batch. No SGD gradients are propagated by the update in Eq. (3).

In our experiments, we set $\lambda_l = 0.1$ and $\lambda_g = 1$.

B.3. Differentiable Rasterizer

We use a differentiable rasterizer for computing the screen space inputs given dense mesh and local expression code map, as illustrated in Fig.2 in the main text. Note that the geometry information affects the final decoded image via two gradient paths: one is in the rasterization, and the other is as input to the pixel decoder. We empirically found that allowing gradient from the image loss to pass to the geometry decoder from both paths leads to unstable training and geometry artifacts, so we disable the second gradient path mentioned above to achieve stable training. Intuitively, this is to enforce that the geometry decoder should focus on producing correct facial shape, instead of *coordinating* with the pixel decoder to produce correct color values.

References

[1] Stephen Lombardi, Jason Saragih, Tomas Simon, and Yaser Sheikh. Deep appearance models for face rendering. *ACM Trans. Graph.*, 37(4), 2018.

[2] Gabriel Schwartz, Shih-En Wei, Te-Li Wang, Stephen Lombardi, Tomas Simon, Jason Saragih, and Yaser Sheikh. The eyes have it: An integrated eye and face model for photorealistic facial animation. *ACM Trans. Graph.*, 39(4), 2020.

[3] Olga Sorkine, Daniel Cohen-Or, Yaron Lipman, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Proceedings of the EUROGRAPHICS/ACM SIGGRAPH Symposium on Geometry Processing*, pages 179–188. ACM Press, 2004.