Supplementary Material for
LiP-Flow: Learning Inference-time Priors for Codec Avatars via Normalizing Flows in Latent Space

Emre Aksan[§1], Shugao Ma[3], Akin Caliskan[§2], Stanislav Pidhorskyi[3], Alexander Richard[3], Shih-En Wei[3], Jason Saragih[3], and Otmar Hilliges[1]

[1] ETH Zürich, Department of Computer Science
[2] CVSSP, University of Surrey
[3] Meta Reality Labs Research

Fig. 8. Random latent samples decoded for the frontal view. (Left) DAM results sampled from $\mathcal{N}(\mathbf{0}, \mathbf{I})$. (Right) Our DAM-Flow variant with an unconditional normalizing flow between the DAM-encoder and the standard Normal prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$ (Sec. A). Before passing to the decoder, latent samples are first transformed to the DAM-encoder's (Q) space via $\mathbf{z} = F^{-1}(\mathbf{z}^{(p)})$. Our flow-based formulation yields a much more expressive latent space, generating higher-quality samples with diverse expressions compared to the KL-based latent space.

This material includes this document and a video. We provide additional results, qualitative evaluations, implementation details of the models and experimental details. Finally, we provide our broader impact statement, in Sec. I.

A DAM with Unconditional Latent Flow

To evaluate the effectiveness of our flow-based prior formulation, we run an ablation by ignoring the LiP-encoders. To this end, we tie a standard Normal prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$ to the DAM-encoder (Q) via an unconditional normalizing flow and compare this to the DAM trained via KL-D (see Fig. 9). We follow the same formulation with LiP-Flow except that we use a standard Normal prior instead of a conditional learned prior and an unconditional normalizing flow model $\tilde{F}$.

[§] This work was performed during an internship at Meta Reality Labs Research.

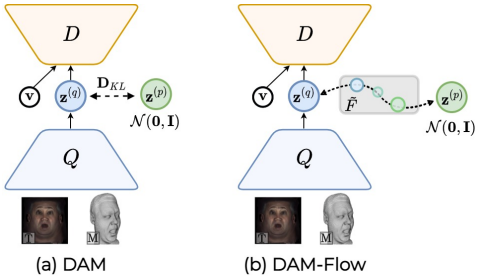

Fig. 9. DAM with unconditional latent flow. We replace KL-divergence with our flow-based latent formulation where the prior distribution is a standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$ in contrast to the learned prior in our main model LiP-Flow .

	DAM-encoder Decoding			Frontal Fitting		
	PSNR ↑	SSIM ↑	Geom ↓	PSNR ↑	SSIM ↑	Geom ↓
DAM	34.95	0.879	0.013	34.67	0.879	0.180
DAM-Flow	35.02	0.881	**0.012**	35.49	0.885	0.067
LiP-Flow-HMC	35.20	**0.882**	**0.012**	**35.91**	**0.889**	**0.017**
LiP-Flow-KPT	**35.25**	**0.882**	**0.012**	35.90	**0.889**	**0.017**

Table 6. DAM-Flow, DAM with unconditional latent flow. Introducing our flow-based latent space formulation into the base model with a standard Gaussian prior. We report reconstruction performance of the DAM-encoder and the inference-time optimization performance with frontal-view targets on one subject.

$$\bar{\mathbf{z}}^{(p)} = \tilde{F}(\mathbf{z}^{(q)}), \quad \tilde{F} = \tilde{f}_K \circ \cdots \circ \tilde{f}_2 \circ \tilde{f}_1, \tag{1}$$

$$\log p_P(\mathbf{z}^{(q)}) = \log p_P(\bar{\mathbf{z}}^{(p)}) + \log \left(\det \left| \frac{\partial \tilde{F}(\mathbf{z}^{(q)})}{\partial \mathbf{z}^{(q)}} \right| \right), \tag{2}$$

where $\log p_P(\bar{\mathbf{z}}^{(p)}) = \log \mathcal{N}(\bar{\mathbf{z}}^{(p)}; \mathbf{0}, \mathbf{I})$ and $\mathbf{z}^{(q)}$ is a latent sample from the DAM-encoder (Q). Our DAM-Flow is conceptually similar to the LSGM [9] in terms of learning an invertible mapping between the encoder space and a standard Normal $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Different from [9], we use a normalizing flow network instead of a continuous diffusion process to learn the mapping, and our encoder (Q) still parameterizes a Gaussian. Moreover, we ignore the negative encoder entropy term in the KL decomposition and the KL-D latent regularization term $\mathcal{L}_L$ (Eq. 4) in the training objective becomes $-\log p_P(\mathbf{z}^{(q)})$. We find this formulation to be more expressive than the base model even with the standard prior. We show that it improves the base model's reconstruction performance both for training and optimization (see Tab. 6). The DAM-Flow's reconstruction performance is on par with the DAM. However, it achieves significantly better reconstructions in the inference-time optimization task when we fit latent codes to single frontal-view images only. We also see that our learned conditional prior is more favorable compared to the standard Gaussian prior.

In Fig. 8, we present samples generated by the base network DAM and our variant DAM-Flow. We decode random samples from the standard Gaussian prior $\mathcal{N}(\mathbf{0}, \mathbf{I})$ for the frontal view. In our variant DAM-Flow, we first transform the latent samples to the Q space via our latent flow function F such that $\mathbf{z} = F^{-1}(\mathbf{z}^{(p)})$. Our modification to the base DAM network can be considered

Models	PSNR ↑	SSIM ↑	Geom ↓	PSNR ↑	SSIM ↑	Geom ↓
	HMC-Encoder Decoding			**DAM-Encoder Decoding**		
(a) DAM	n/a	n/a	n/a	36.05	0.893	0.015
(b) DAM-HMC Enc.	33.93	0.872	0.186	33.93	0.872	0.186
(c) DAM-HMC Reg.	34.16	0.872	0.248	n/a	n/a	n/a
(d) LiP-KL-HMC	34.39	0.876	0.146	35.79	0.890	0.018
(e) LiP-Flow-HMC	34.04	0.871	0.161	**36.21**	**0.895**	**0.014**
	HMC Fitting w/o $\mathcal{L}_L$			**HMC Fitting**		
(a) DAM	31.00	0.853	0.742	31.65	0.858	0.676
(b) DAM-HMC Enc.	33.68	0.873	0.189	34.14	0.877	0.133
(c) DAM-HMC Reg.	34.28	0.881	0.158	34.58	0.882	0.159
(d) LiP-KL-HMC	33.73	0.876	0.121	34.36	0.881	0.090
(e) LiP-Flow-HMC	33.57	0.876	0.142	**34.98**	**0.885**	**0.087**
	Frontal Fitting w/o $\mathcal{L}_L$			**Frontal Fitting**		
(a) DAM	35.32	0.888	0.312	35.16	0.888	0.212
(b) DAM-HMC Enc.	35.21	0.882	0.047	35.36	0.883	0.047
(c) DAM-HMC Reg.	36.20	0.894	0.049	36.28	0.895	0.057
(d) LiP-KL-HMC	35.96	0.891	0.045	36.08	0.892	0.039
(e) LiP-Flow-HMC	36.10	0.893	0.067	**36.50**	**0.898**	**0.022**
	Masked Frontal Fitting w/o $\mathcal{L}_L$			**Masked Frontal Fitting**		
(a) DAM	32.30	0.867	0.770	32.38	0.867	0.604
(b) DAM-HMC Enc.	34.37	0.876	0.108	34.59	0.878	0.095
(c) DAM-HMC Reg.	34.89	0.883	0.152	35.17	0.886	0.149
(d) LiP-KL-HMC	34.93	0.883	0.094	35.17	0.885	0.071
(e) LiP-Flow-HMC	35.08	0.886	0.088	**35.44**	**0.889**	**0.060**

Table 7. Evaluations in HMC setting. Reporting average performance over 4 subjects. (Left) We provide the forward-pass performance of the HMC-encoder as well as the inference-time optimization results without using the latent likelihood $\mathcal{L}_L$ (Eq. 10). (Right) Results for the DAM-encoder's forward-pass and inference-time optimization with the latent likelihood. In the fitting tasks, we initialize the latent code with the same latent sample we use for evaluating the HMC-encoder. Hence, the "HMC-encoder Decoding" results denote the performance before running fitting.

minimal. Yet our approach yields more diverse and higher-quality samples. This ablation shows that our flow-based latent formulation is an effective approach, and our contribution is not only from learning a conditional prior, both of which constitute a powerful means to solve our problem.

B Inference-time Optimization Ablations

In Tables 7 and 8, we provide ablations for inference-time optimization and the latent likelihood term $\mathcal{L}_L$ (Eq. 10). We evaluate all the models used in the main paper in both the HMC (Tab. 7) and the 2D keypoint (Tab. 8) settings. The "HMC-encoder Decoding" column in Tab. 7 reports the HMC-encoder's forward pass performance. More specifically, given a set of input HMC images, we estimate a prior distribution via the HMC-encoder, and then decode its mean for the frontal-view direction. Note that we use the same latent code to initialize the fitting tasks. Hence, this evaluation provides the performance before opti-

Models	PSNR ↑	SSIM ↑	Geom ↓	PSNR ↑	SSIM ↑	Geom ↓
	KPT-Encoder Decoding			**DAM-Encoder Decoding**		
(a) DAM	n/a	n/a	n/a	36.05	0.893	0.015
(b) DAM-KPT Enc.	33.50	0.866	0.142	33.50	0.866	0.142
(c) DAM-KPT Reg.	33.67	0.864	0.151	n/a	n/a	n/a
(d) LiP-KL-KPT	34.21	0.872	0.128	35.71	0.888	0.021
(e) LiP-Flow-KPT	33.72	0.863	0.150	**36.22**	**0.895**	**0.014**
	KPT Fitting w/o $\mathcal{L}_L$			**KPT Fitting**		
(a) DAM	31.62	0.865	0.431	31.62	0.865	0.431
(b) DAM-KPT Enc.	33.91	0.873	0.126	33.96	0.873	0.119
(c) DAM-KPT Reg.	34.71	0.884	0.061	34.88	0.886	0.059
(d) LiP-KL-KPT	34.74	0.883	0.099	35.08	0.886	0.089
(e) LiP-Flow-KPT	34.79	0.884	0.063	**35.55**	**0.891**	**0.053**
	Frontal Fitting w/o $\mathcal{L}_L$			**Frontal Fitting**		
(a) DAM	35.32	0.888	0.312	35.16	0.888	0.212
(b) DAM-KPT Enc.	34.74	0.874	0.060	34.75	0.874	0.059
(c) DAM-KPT Reg.	36.27	0.895	0.032	36.31	0.895	0.030
(d) LiP-KL-KPT	35.89	0.890	0.044	35.94	0.890	0.041
(e) LiP-Flow-KPT	36.46	0.897	0.029	**36.53**	**0.898**	**0.025**
	Masked Frontal Fitting w/o $\mathcal{L}_L$			**Masked Frontal Fitting**		
(a) DAM	32.30	0.867	0.770	32.38	0.867	0.604
(b) DAM-KPT Enc.	33.94	0.869	0.106	34.16	0.870	0.085
(c) DAM-KPT Reg.	35.01	0.884	0.060	35.13	0.884	0.055
(d) LiP-KL-KPT	34.72	0.880	0.080	35.14	0.884	0.060
(e) LiP-Flow-KPT	35.09	0.885	0.071	**35.45**	**0.887**	**0.054**

Table 8. Evaluations in 2D key point setting. Reporting average performance over 4 subjects. (Left) We provide the forward-pass performance of the KPT-encoder as well as the inference-time optimization results without using the latent likelihood $\mathcal{L}_L$ (Eq. 10). (Right) Results for the DAM-encoder's forward-pass and inference-time optimization with the latent likelihood. In the fitting tasks, we initialize the latent code with the same latent sample we use for evaluating the KPT-encoder. Hence, the "KPT-encoder Decoding" results denote the performance before running fitting.

mization. The same evaluation steps also apply to the keypoint setting (Tab. 8) where we evaluate the models with the KPT-encoder expecting 2D keypoints.

When we decode the LiP-encoder samples without optimization, we observe that the HMC- and the KPT-encoder trained via our flow-based latent formulation performs poorly whereas the KL-based training achieves the best performance. We think that this is due to the similarity assumption imposed by the KL-divergence loss. It can be explained as a trade-off between the forward-pass accuracy and the iterative fitting performance. While our flow-based formulation yields a prior performing well in the iterative fitting, the KL-based formulation works better in the forward-pass case. However, our proposed LiP-Flow achieves a significant improvement when the latent code is optimized (see "HMC-encoder Decoding" vs. "HMC Fitting" in Tab. 7 and "KPT-encoder Decoding" vs. "KPT Fitting" in Tab. 8).

In both settings, we also see that using the latent likelihood in the optimization objective is effective for all the models. While its contribution is limited in

Fitting latent code and headset pose to HMC observations			
Models	**PSNR ↑**	**SSIM ↑**	**Geom ↓**
(a) DAM	28.54	0.828	1.131
(b) DAM-HMC Enc.	31.52	0.855	0.323
(c) DAM-HMC Reg.	32.93	0.860	0.237
(d) LiP-KL-HMC	33.66	0.869	0.144
(e) LiP-Flow-HMC	**34.09**	**0.872**	**0.141**

Table 9. Fitting both *latent code* and *headset pose*. It is a 6 DOF global transformation from the avatar's reference frame to the headset. In Tab. 2, the headset pose is assumed to be given.

the baseline models, the learned priors (*i.e.*, Lip-KL and Lip-Flow) benefit the most when the fitting targets carry less information.

We also provide iterative fitting results for the DAM baselines, namely the "DAM-HMC Encoder" and "DAM-HMC Regressor" in Tab. 7 and the "DAM-KPT Encoder" and "DAM-KPT Regressor" in Tab. 8. In both settings, both baselines show improvements via iterative fitting. This suggests our proposed inference-time optimization approach is necessary to optimize the performance for all the models.

To make a direct comparison between the clean and noisy targets, we limit the amount of information in the frontal-view targets by applying masks (Fig. 10). We use the same set of target images and mask out the face except the mouth and eye regions. Similar to the "Frontal-view Fitting" task (Tab. 4), the models have access to the HMC images or the 2D keypoints as the driving signal depending on the setting and the masked frontal views targets. Although the DAM improves its reconstruction performance when the latent codes are optimized for the frontal-view targets ("Frontal Fitting" in Tables 7 and 8), it degrades significantly on the masked targets. This is inline with the HMC- and keypoint-fitting results.

In Tab. 9, we introduce headset pose as a source of noise to the HMC fitting task (Tab. 2) and optimize both the latent codes and the headset pose parameters simultaneously. The headset pose determines the alignment of the renderings (*i.e.*predicted HMCs) to the observations.

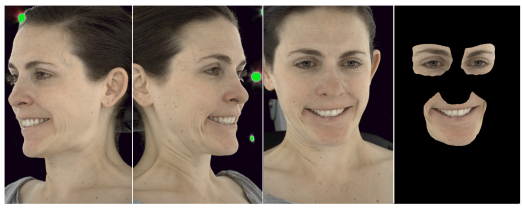

Fig. 10. Ground-truth images in left, right, and frontal views. We apply a mask on the frontal-view images to limit the amount of information in the fitting task.

Finally, in our supplementary video, we provide optimization results with noisy observations where we apply random perturbations to the headset camera parameters. We show that our learned prior is more robust to noise, achieving notably less jitter and more temporally coherent fitting results although we do not apply any temporal regularizations.

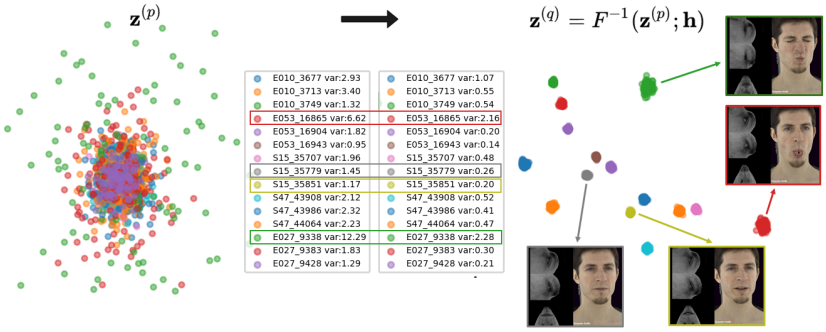

Fig. 11. Latent space visualization in PCA (Left) Samples in the P space. (Right) Same samples in the Q space after applying our latent transformation via flow F. For a given HMC sample (color-coded), the HMC-encoder P predicts a prior distribution. We visualize 100 latent codes sampled from the respective prior distribution, in total 1500 latent samples for 15 HMC inputs. The legend provides the variance of 100 latent samples, namely the variance of the predicted prior distribution $\mathcal{N}(\boldsymbol{\mu}^{(p)}, \boldsymbol{\sigma}^{(p)})$ per HMC sample. Samples with neutral expressions result in lower variance whereas rare and peak expressions cause higher variance in the latent space.

C Latent Space

Fig. 11 provides an extended version of the latent space visualization presented in the main submission. We report the variance of the latent samples for every HMC input sample. We also visualize corresponding HMC- and frontal-view images for the samples with high and low variance. We see that our flow-based latent formulation learns to assign larger variance to the inputs with rare and peak facial expressions, quantifying the difficulty of the corresponding sample. While the variances are much larger in the prior (P) space, they get smaller in the Q space after our latent transformation F^{-1}. We provide more results in our supplementary video.

D Evaluating on Real HMC Images

We evaluate our model that is trained with synthetic HMC images on real HMC images without applying any domain adaptation. We provide forward-pass results in Fig. 12. In the case of real HMC images, inference-time optimization via differentiable rendering is not feasible due to the large discrepancies in lighting, background, and cameras. Our model still achieves promising results with a forward-pass only.

E Qualitative Results

In Figures 14 and 13, we present qualitative results for both the HMC-view and 2D keypoint observations. For all the models, we optimize the latent codes via iterative fitting and present the frontal-view decodings. We provide animated flip-comparisons in our supplementary video.

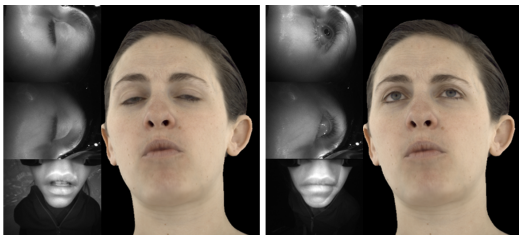

Fig. 12. Passing real HMC images to our model that is trained on synthetic HMCs. Here we do not apply inference-time optimization and present forward pass results (*i.e.*initial step only).

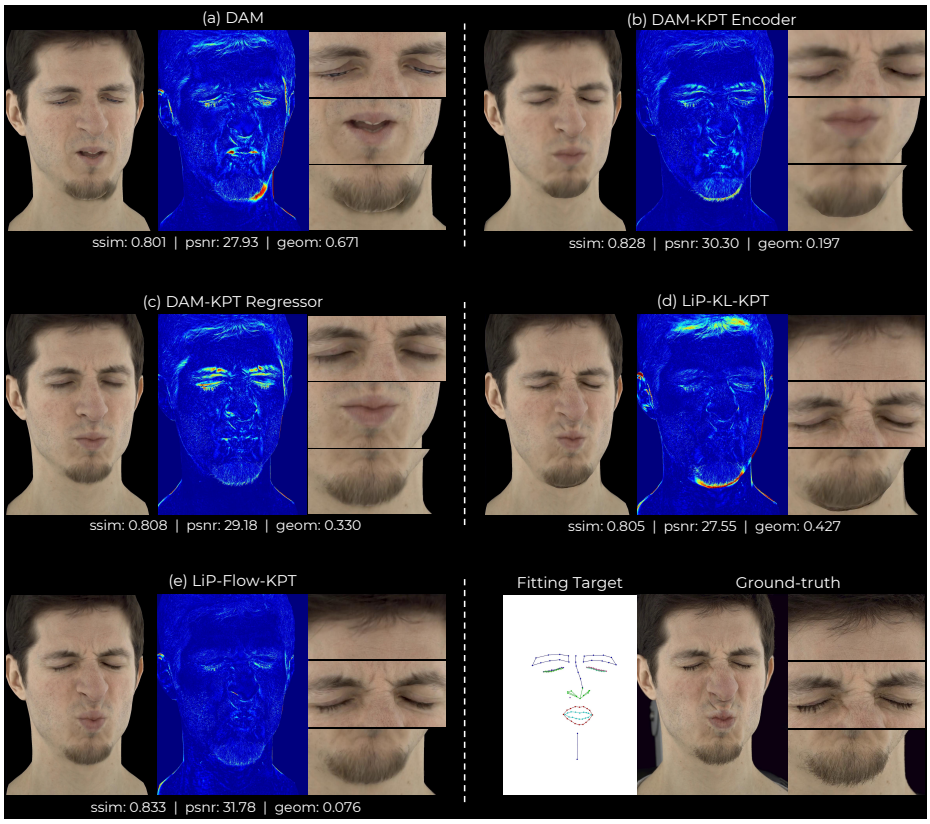

Fig. 13. Reconstruction from 2D keypoints. For all the models, we run inference-time optimization with sparse facial landmarks as the fitting target (cf. Sec. 4.2). Note that full face is reconstructed from only the 2D keypoints. We visualize (left) frontal-view renderings of the optimized latent codes, (center) the difference between rendered and the ground-truth images and (right) close looks for three regions. Compared to the baselines, our model LiP-Flow (e) achieves lower geometry error and reconstructs better textures, particularly with no errors around the chin and beard.

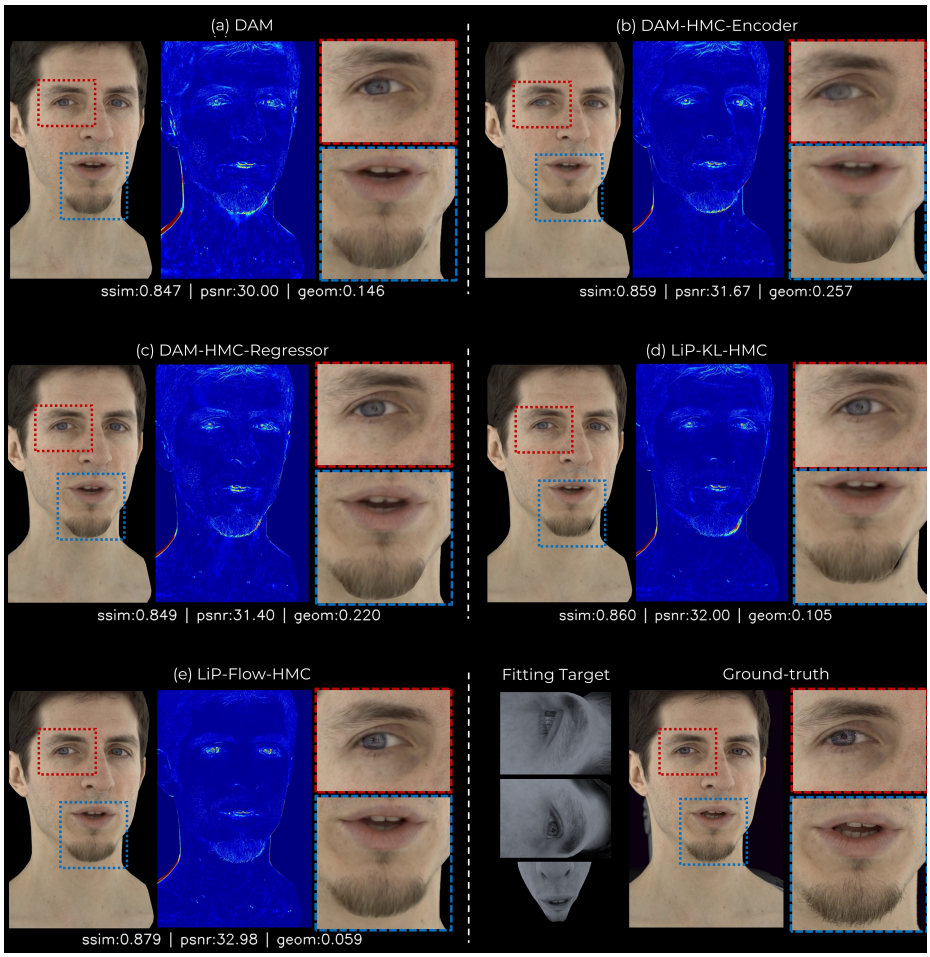

Fig. 14. Reconstruction from HMC views. For all the models, we run inference-time optimization with partial HMC views as the fitting target (cf. Sec. 4.2). Note that full face is reconstructed from only the HCM images. We visualize (left) frontal-view renderings of the optimized latent codes, (center) the difference between rendered and the ground-truth images and (right) close looks for two regions. Our model LiP-Flow (e) achieves lower geometry error and reconstructs a sharper texture compared to the baselines.

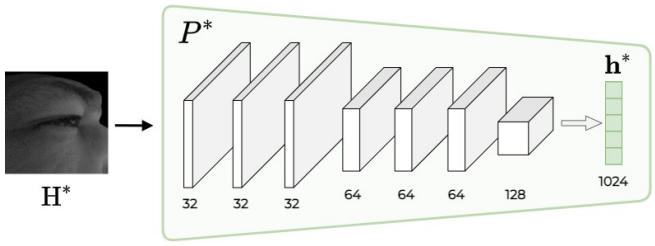

Fig. 15. HMC-encoder block for an input HMC image. Number of output channels is provided in the figure.

F Architecture Details

In our work, we choose the Deep Appearance Model (DAM) of Lombardi *et al.*[6] as the base model where the DAM-encoder (Q) and the decoder (D) networks use the same architecture and hyper-parameters. We provide details for the components introduced by us, namely the HMC-encoder (P) and the normalizing flow model in the latent space (F). We use PyTorch [8] for training our models.

In terms of computational complexity, our latent flow requires 0.011 GFLOPs to transform a latent sample. It is 1.736 and 0.004 for the HMC-Encoder and the KPT-Encoder, respectively. For reference, the DAM-Encoder and the decoder need 2.54 and 2.0 GFLOPs. The computational requirement for our flow network and the KPT-Encoder are lower due to the lower dimensional inputs. We use the `fvcore`[4] library for computing FLOPs.

F.1 HMC-encoder

Our HMC-encoder consists of 3 separate convolutional blocks for each of the HMC image $\mathbf{H}^* \in \mathbb{R}^{480\times640}$. Fig. 15 illustrates a convolutional block. We use convolutions with kernel size 4 and stride 2, followed by leaky relu activation functions [7] with a negative slope of 0.2. The final output is reshaped into a 1024-dimensional hidden vector $\mathbf{h}^*$.

After getting initial representations $\mathbf{h}^*$ for 3 HMC views, we apply a 3-channel attention operation to calculate the HMC representation vector $\mathbf{h} \in \mathbb{R}^{1024}$ by following Eq. 5. We experimented with different alternatives such as concatenation of all the representations $\mathbf{h}^*$ and experimentally verified that the attention mechanism yields the best performance. The attention weights $\mathbf{W}^A \in \mathbb{R}^{1024\times3}$ are initialized with samples from $\mathcal{N}(0, 1)$. We finally estimate the mean $\boldsymbol{\mu}^{(p)} \in \mathbb{R}^{256}$ and the standard deviation $\boldsymbol{\sigma}^{(p)} \in \mathbb{R}^{256}$ by using linear layers without any activation.

F.2 KPT-encoder

Our KPT-encoder takes 274×2 dimensional keypoint inputs, which is mapped to an initial hidden representation of size 512 by a dense layer. We then stack

[4] https://github.com/facebookresearch/fvcore/blob/main/docs/flop_count.md

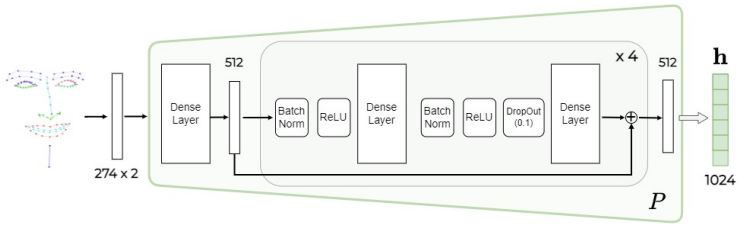

Fig. 16. KPT-encoder. Input keypoints are of shape 274×2. We stack 4 residual blocks with inputs and outputs of shape 512. The final representation $\mathbf{h}$ is a 1024 dimensional vector.

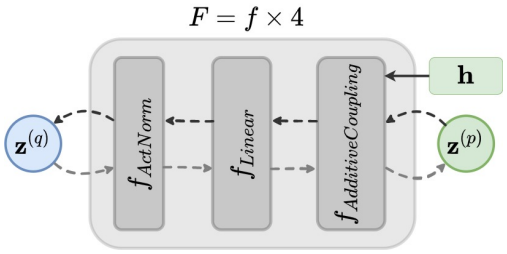

Fig. 17. Overview of our normalizing flow network F. We stack 4 flow steps following [4]. Each step consists of activation normalization, linear and additive coupling layers. The condition input $\mathbf{h}$ is passed to the additive coupling layer at every level. $\mathbf{h}$ corresponds to the HMC-view representations (cf. Eq. 5.)

4 residual blocs consisting of batch normalization, ReLU, dense layers and a dropout (Fig. 16). Finally, the 512 dimensional hidden representation is transformed to the keypoint representation vector $\mathbf{h} \in \mathbb{R}^{1024}$ by a dense layer.

F.3 Latent Flow

We use the normalizing flow architecture proposed in [5]. Our flow network adapts building blocks of Glow [4] to 1D input where the invertible 1×1 convolution layer is replaced by a linear layer. We implement the normalizing flow network in our LiP-Flow by using the `nflows` library [2]. More specifically, we use `ActNorm`, `LULinear` and `AdditiveCouplingTransform` classes to implement the $f_{ActNorm}$, f_{Linear} and $f_{AdditiveCoupling}$ layers in Fig. 17. Hence, in Tab. 10, we provide the hyper-parameters in the same interface with the `nflows` library. Our choice of the internal network `NN` for the coupling layer is a residual block conditioned on $\mathbf{h}$ (*i.e.*, context) as in [5].

In our experiments, we find that the volume preserving additive coupling layer [1] is essential. Our model LiP-Flow often does not converge with the affine coupling layer.

G Dataset

We follow the same data preprocessing steps as in [6]. Each subject's data consists of ∼10,000 training frames where we have 40 camera views for each frame. We evaluate models on a held-out set of ∼500 frames.

Hyper-parameter	Value
features	256
hidden_features	1024
num_layers	4
num_blocks_per_layer	2
dropout_probability	0
activation	ReLU
batch_norm_within_layers	True

Table 10. Hyper-parameters of our normalizing flow network.

Our synthetic HMC dataset simulates the application conditions by applying augmentations including different lighting, background and headset orientation. The synthetic HMC images are generated by re-projecting the multi-view camera images into virtual head-mounted camera views. More specifically, we sample headset camera parameters and render the corresponding HMC view by using the tracked mesh and the texture available in the multi-view dataset. We note that the synthetic and real HMC images are not photometrically alike due to the differences in the lighting conditions and camera properties.

H Experiment Details

H.1 Training

In our experiments, we use the same hyper-parameters for all the models as in [6]. The batch size and the learning rate are 16 and $5e^{-4}$, respectively. We allow models to train for $200,000$ steps. Different from [6], we implement early stopping as we observe overfitting issues for all the models. Accordingly, if the training loss on the held-out set does not improve for $20,000$ steps, training is terminated. We use the following training objective (see Eq. 2):

$$\mathcal{L} = \sum_{v} \lambda_I \mathcal{L}_I + \lambda_M \mathcal{L}_M + \lambda_L \mathcal{L}_L, \tag{3}$$

In the training objective $\lambda_I = 10$, $\lambda_G = 1$. In the keypoint setting where we use the KPT-encoder conditioned on 2D keypoints, we observe very high amount of variance in the P space. To alleviate this, we introduce an additional loss term to the training objective, penalizing the entropy of the predicted Gaussian $\mathcal{N}(\boldsymbol{\mu}^{(p)}, \boldsymbol{\sigma}^{(p)})$. Its weight is the same as the latent log-likelihood term λ_L. Note that our LiP-Flow-KPT achieve the best results even in the absence of this prior entropy regularization term.

Since the latent regularization term $\mathcal{L}_L$ takes different forms in different models, its corresponding weight λ_L is the most important hyper-parameter for the training. We run a mini hyper-parameter tuning on subject 1 and use the same set of values on the remaining subjects.

DAM We considered KL-divergence (KL-D) weights of 0.01 and 1. We also followed an annealing strategy by starting it from 0.01 and increasing it until 1. In our experiments, the reported model achieved the best performance with the KL-D weight of 0.01 as in [6].
DAM-HMC-encoder Unlike DAM, this model performed best when trained with the KL-D weight of 1.
DAM-HMC-Regressor In this setting, we use a pre-trained DAM-encoder to get the training labels and train the HMC-encoder with the log-likelihood objective. As it is the only training objective, we set the weight to 1.
LiP-KL We followed the same hyper-parameter setup with the DAM. LiP-KL performs better when it is trained with KL-D weight 1.
LiP-Flow Since we replace the KL-D term with a latent likelihood, we use a different set of weights. We consider weights $0.005, 0.0001, 0.0005$. Our model achieved the best performance when $\lambda_L = 0.0005$.

H.2 Inference-time Optimization (Sec. 4.2

We use the HMC-encoder to estimate a prior distribution which is then used for initialization and likelihood evaluation of the latent code. The HMC-encoder is available for all the models except the DAM (see Fig. 4). The prior distribution is set to Normal $\mathcal{N}(\mathbf{0}, \mathbf{I})$ for the DAM baseline.

In our evaluations, we assume that the camera parameters and the view-vector are available. The latent code is decoded into 3D geometry and view-dependent texture which are then rendered to either frontal view or HMC views. Note that we decode the latent code 3 times for each HMC-view if the target observations are the HMC images. We sample HMC camera parameters and the corresponding view-vector from an HMC dataset.

The optimization objective (Eq. 10) consists of image reconstruction and latent likelihood terms. For the masked frontal-view and partial HMC-view targets, the image loss is calculated on the visible regions only. After setting the image loss weight to 1, a grid search is performed on the first evaluation batch for the latent likelihood weight λ_L and the learning rate. We then report the performance with the best performing hyper-parameters. The learning rate is determined from $\{0.01, 0.1, 0.5\}$. We use different sets for the latent likelihood weight λ_L for the frontal-view and HMC-view targets, which are $\{0.01, 0.001, 0.0001\}$ and $\{0.1, 0.01, 0.001\}$, respectively. Since the fitting task is more challenging with the HMC-view targets, we observe that the models rely on the prior more and hence we run the hyper-parameter tuning with larger weights.

We use the ADAM algorithm [3] for optimizing the latent code. The pre-trained decoder is frozen during the optimization. We run the optimization up to 200 steps and stop if there is no improvement in terms of the fitting objective. The average number of steps was < 200 for all models.

I Broader Impact Statement

Since we aim to build photorealistic personalized avatars, our work constitutes a potential risk for negative use cases such as fake content generation and identity theft. Although the current state of the art including our work has not yet achieved photorealism, future research may achieve metric quality and produce synthetic data that are indistinguishable from the real ones. While synthesis of fake content is a problem for all generative models, our work as well as the prior works on 3D avatars present another potential misuse. By driving an established personalized avatar model, third parties may fake the identity. This can be prevented by introducing a verification layer to the mobile telepresence pipeline via retina-based bio-metric authentication systems.

References

1. Dinh, L., Krueger, D., Bengio, Y.: Nice: Non-linear independent components estimation. arXiv preprint arXiv:1410.8516 (2014) 10
2. Durkan, C., Bekasov, A., Murray, I., Papamakarios, G.: nflows: normalizing flows in PyTorch (Nov 2020). https://doi.org/10.5281/zenodo.4296287, https://doi.org/10.5281/zenodo.4296287 10
3. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 12
4. Kingma, D.P., Dhariwal, P.: Glow: Generative flow with invertible 1x1 convolutions. arXiv preprint arXiv:1807.03039 (2018) 10
5. Kolotouros, N., Pavlakos, G., Jayaraman, D., Daniilidis, K.: Probabilistic modeling for human mesh recovery. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 11605–11614 (2021) 10
6. Lombardi, S., Saragih, J., Simon, T., Sheikh, Y.: Deep appearance models for face rendering. ACM Transactions on Graphics (TOG) **37**(4), 1–13 (2018) 9, 10, 11, 12
7. Maas, A.L., Hannun, A.Y., Ng, A.Y., et al.: Rectifier nonlinearities improve neural network acoustic models. In: Proc. icml. vol. 30, p. 3. Citeseer (2013) 9
8. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019), http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf 9
9. Vahdat, A., Kreis, K., Kautz, J.: Score-based generative modeling in latent space. Advances in Neural Information Processing Systems **34** (2021) 2