PersonaLive! Expressive Portrait Image Animation for Live Streaming Supplemental Material

Zhiyuan Li[1,2,3] Chi-Man Pun[1,*] Chen Fang[2] Jue Wang[2] Xiaodong Cun[3,*]
[1] University of Macau [2] Dzine.ai [3] GVC Lab, Great Bay University
https://github.com/GVCLab/PersonaLive

A. Preliminary: Latent Diffusion Model

Latent Diffusion Models (LDMs) [4] conduct the diffusion process in a compact latent space for improving efficiency. Given an image $x \in \mathbb{R}^{H\times W\times 3}$, an encoder $\mathcal{V}_e$ first maps it to a latent representation $z = \mathcal{V}_e(x)$. After that, the latent representation is progressively corrupted by Gaussian noise ϵ as:

$$z_t = \Psi(z, \epsilon, t) = \sqrt{\bar{\alpha}_t} z + \sqrt{1-\bar{\alpha}_t}\epsilon, \tag{1}$$

where $\bar{\alpha}_t$ is a pre-defined noise schedule within a finite time horizon $t \in [0, 1000]$. A U-Net [5] denoiser ϵ_θ is trained to predict and remove the added noise from z_t. The training objective is formulated as:

$$\mathcal{L}_{LDM} = \mathop{\mathbb{E}}_{t,z,\epsilon} \|\epsilon - \epsilon_\theta(z_t, t, c)\|_2^2, \tag{2}$$

where c represents the conditioning input. In portrait animation, the input is a multi-frame latent window $\{z_t^i\}_{i=1}^M$, c contains appearance features extracted from the reference portrait and motion features derived from the driving video.

B. Experimental Details

More implementation details. Our training pipeline progresses through three stages, utilizing Stable Diffusion v1.5 (SD 1.5) as the base model. In Stage 1, we conduct image-level hybrid motion training. Specifically, we randomly sample paired reference and driving images from the training videos, enabling the model to learn appearance conditioning from the reference portrait and motion conditioning from the driving input. This stage is trained for 30K iterations with a batch size of 64. During this stage, all model parameters are updated. After this initialization, Stage 2 performs fewer-step appearance distillation following Algorithm 1. We adopt a compact sampling schedule $[0, 333, 666, 999]$, enabling the model to learn to reconstruct high-quality frames from only a few denoising steps. This stage is trained for 30K iterations with a batch size of 32, and all model parameters remain trainable. Stage 3 focuses on temporal modeling: we train the temporal attention layers following Algorithm 2. At each iteration, we slide over a 40-frame sequence and perform three model updates. Only the temporal attention layers are trainable in this stage, while all other parameters remain frozen. This stage is trained for 10K iterations with a batch size of 8. For Stage 2 and 3, λ_{lpips} and λ_{adv} are set to 2.0 and 0.05, respectively. The total training time is approximately 48 hours on 8 NVIDIA H100 GPUs (Stage 1: 13h, Stage 2: 15h, Stage 3: 20h).

Implicit 3D keypoints. As shown in Fig. 1, the implicit 3D keypoint extractor [3] produces 21 canonical keypoints (left),

Algorithm 1: Fewer-step Appearance Distillation

Input: Reference image I_R; Driving image I_D; Animation model G_θ; VAE decoder $\mathcal{V}_d$; Sampling schedule $\{t_i\}_{i=1}^N$
Output: Updated parameters θ

for *each iteration* **do**
- `/* Sample initial noisy latent and random step count n */`
- Sample $z_{\text{noise}} \sim \mathcal{N}(0, I)$
- Sample $n \sim \text{Uniform}(1, N)$
- Set $z_{t_N} \leftarrow z_{\text{noise}}$
- `/* Perform n denoising steps */`
- **for** $i = N$ **to** $N - n + 1$ **do**
  - **if** $i > N - n + 1$ **then**
    - Disable gradient computation
    - Set $\hat{z}_0 \leftarrow G_\theta(z_{t_i}; t_i, I_R, I_D)$
    - Sample $\epsilon \sim \mathcal{N}(0, I)$
    - Set $z_{t_{i-1}} \leftarrow \Psi(\hat{z}_0, \epsilon, t_{i-1})$
  - **else**
    - Enable gradient computation
    - Set $\hat{z}_0 \leftarrow G_\theta(z_{t_i}; t_i, I_R, I_D)$
- `/* Decode prediction */`
- Set $\hat{x} \leftarrow \mathcal{V}_d(\hat{z}_0)$
- `/* Update model */`
- Update θ via Distillation loss $\mathcal{L}_{distill}$

return θ

*Equal contribution

Algorithm 2: Sliding Training Strategy

Input: Reference image I_R; Driving video $\{I_D^i\}_{i=1}^S$; Animation model G_θ; VAE encoder $\mathcal{V}_e$; VAE decoder $\mathcal{V}_d$; Micro-chunk size M; Sampling schedule $\{t_i\}_{i=1}^N$

Output: Updated parameters θ

for *each iteration* **do**

  /* Construct the initial denoising window W_0 */

  Set $z^{1:M(N-1)} \leftarrow \mathcal{V}_e(I_D^{1:M(N-1)})$

  for $n = 1$ *to* $N-1$ **do**

    Sample $\epsilon \sim \mathcal{N}(0, I)$

    Set $C^n \leftarrow \Psi(z^{(n-1)M+1:nM}, \epsilon, t_n)$

  Sample $C^N \sim \mathcal{N}(0, I)$; /* Last chunk is pure noise */

  Initialize window $W_0 = \{C^1, C^2, \ldots, C^N\}$

  /* Sliding generation and training */

  for $s = 0$ *to* $\frac{S}{M} - N$ **do**

    Set $V_s \leftarrow I_D^{sMN+1:(s+1)MN}$

    if $s \bmod (N-1) \neq 0$ **then**

      Disable gradient computation

      Set $\hat{W}_s \leftarrow G_\theta(W_s, t_{1:N}, I_R, V_s)$

    else

      Enable gradient computation

      Set $\hat{W}_s \leftarrow G_\theta(W_s, t_{1:N}, I_R, V_s)$

      /* Decode sequence prediction */

      Set $\hat{x}_{seq} \leftarrow \mathcal{V}_d(\hat{W}_s)$

      /* Update model */

      Update θ via Distillation loss

      Disable gradient computation

    /* Slide window forward */

    Set $W_{s+1} \leftarrow \{\hat{C}^{s+2}, \hat{C}^{s+3}, \ldots, \hat{C}^{s+N}\}$

    Sample $\epsilon_{s+1} \sim \mathcal{N}(0, I)$

    Set $W_{s+1} \leftarrow \Psi(W_{s+1}, \epsilon_{s+1}, t_{1:N-1})$

    Sample $C^{s+N+1} \sim \mathcal{N}(0, I)$

    Set $W_{s+1} \leftarrow \{W_{s+1}, C^{s+N+1}\}$

return θ

from which we select a subset of stable landmarks (right) to encode global head pose, scale, and spatial configuration. These selected keypoints serve as an effective global motion prior in our hybrid motion control.

Motion-interpolation initialization. Given the reference image I_R and first driving frame I_D^1, we construct the first denoising window $W_0 = \{C^1, C^2, \ldots C^N\}$ using noisy reference latent $z_{ref} = \mathcal{V}_e(I_R)$:

$$C^n = \{\sqrt{\bar{\alpha}_{t_n}} z_{ref} + \sqrt{1 - \bar{\alpha}_{t_n}} \epsilon_i\}_{i=1}^M, \tag{3}$$

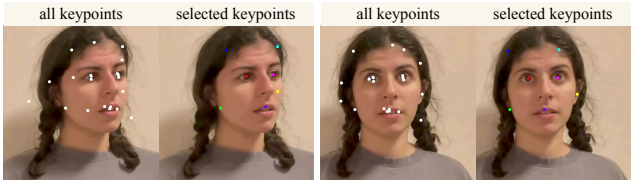

Figure 1. The implicit 3D keypoints used in our hybrid motion control.

where $\epsilon_i \sim \mathcal{N}(0, I)$. Subsequently, we interpolate the motion signals between the reference image and the first driving frame. Let $m_{f,s}$ and $m_{f,d}^1$ denote the implicit facial motion embeddings extracted from I_R and I_D^1, respectively. For the i-th frame in the initial window:

$$m_{f,i} = (1 - \omega_i) m_{f,s} + \omega_i m_{f,d}^1, \tag{4}$$

where $\omega_i = \frac{i-1}{MN-1}$ is the interpolation factor. For implicit 3D keypoints, we interpolate the 3D transformation parameters:

$$\begin{aligned} R_i &= R\big((1 - \omega_i)\theta_s + \omega_i \theta_d^1\big), \\ s_i &= (1 - \omega_i) s_s + \omega_i s_d^1, \\ t_i &= (1 - \omega_i) t_s + \omega_i t_d^1, \end{aligned} \tag{5}$$

where $\theta = (\text{pitch}, \text{yaw}, \text{roll})$ denotes Euler angles and $R(\theta)$ denotes the rotation matrix constructed from Euler angles. The interpolated keypoints are then computed as:

$$k_i = s_i \cdot k_{c,s} R_i + t_i. \tag{6}$$

Inference procedure and speed. Algorithm 3 outlines the complete inference pipeline. The reported inference speed and latency are end-to-end measurements encompassing all computational overheads. Specifically, generating 1,000 frames on a single NVIDIA H100 GPU requires 63.11 s in total (Preprocess: 8.05 s, Denoise: 39.54 s, Decode: 15.52 s). The average per-frame latency is ∼63 ms. With micro-chunk size $M = 4$, the output latency is ≈0.25 s.

Details of LV100. For long-term portrait animation evaluation, we collect 100 unseen videos ($\geq$ 1 minute, 25 FPS) from various online platforms, including YouTube, TikTok, and BiliBili. Additionally, we compile 100 in-the-wild reference portraits from ChatGPT-5, Doubao, and Pexels, covering a broad range of facial structures, appearances, and styles. Representative examples from the LV100 benchmark are shown in Fig. 2.

C. More Ablations

In this section, we provide additional ablations on some network components and hyperparameters.

Hybrid motion signals. As shown in Fig. 3, the implicit 3D keypoints k_d control global head movements, including rotation, translation, and scale. In contrast, the implicit facial

Algorithm 3: Micro-chunk Streaming Inference

Input: Reference image I_R; Streaming chunks $\{V_s^{1:M}\}_{s\geq 1}$; VAE encoder/decoder $\mathcal{V}_e, \mathcal{V}_d$; Pre-trained models $\mathcal{R}, \mathcal{D}, \mathcal{E}_m, \mathcal{E}_p$; Micro-chunk size M; Sampling schedule $\{t_i\}_{i=1}^{N}$; Motion threshold τ

Output: Generated video stream

`/* Initialization phase */`

Set $z_{ref} = \mathcal{V}_e(I_R)$

Set $h_{ref}, m_{ref}, p_{ref} \leftarrow \{\mathcal{R}(z_{ref})\}, \mathcal{E}_m(I_R), \mathcal{E}_p(I_R)$

Initialize motion bank $\mathcal{B}_{mot} = \{m_{rer}\}$

for $n = 1$ *to* $N-1$ **do**

  Sample $\epsilon \sim \mathcal{N}(0, I)$

  Set $C^n \leftarrow \Psi\big(\text{Repeat}(z_{ref}, M), \epsilon, t_n\big)$

Initialize window $W_0 = \{C^1, C^2, \ldots, C^{N-1}\}$

`/* Streaming inference loop */`

for $s = 1, 2, 3, \ldots$ **do**

  `/* Preprocessing */`

  Set $m_s^{1:M}, p_s^{1:M} \leftarrow \mathcal{E}_m(V_s^{1:M}), \mathcal{E}_p(V_s^{1:M})$

  Sample $\epsilon \sim \mathcal{N}(0, I)$

  Set $C^N \leftarrow \Psi(Repeat(z_{ref}, M), \epsilon, t_N)$

  Set current window input $W_s \leftarrow \{W_{s-1}, C^N\}$

  if $s = 1$ **then**

    Using MII strategy to construct motion window $m_s^{1:MN}$ and pose window $p_s^{1:MN}$

  else

    Set $m_s^{1:MN} \leftarrow \{m_{s-1}^{M+1:MN}, m_s^{1:M}\}$

    Set $p_s^{1:MN} \leftarrow \{p_{s-1}^{M+1:MN}, p_s^{1:M}\}$

  `/* Denoising */`

  Predict clean latents $\hat{W}_s$ via $\mathcal{D}(W_s, t_{1:N}, h_{ref}, m_s^{1:MN}, p_s^{1:MN})$

  $d \leftarrow \min_{m \in \mathcal{B}_{mot}} \|m_s^1 - m\|_2$

  if $d > \tau$ **then**

    Set $h_{ref} \leftarrow \{h_{ref}, \mathcal{R}(\hat{C}_1^1)\}$

    Set $\mathcal{B}_{mot} \leftarrow \mathcal{B}_{mot} \cup \{m_s^1\}$

  `/* Decoding */`

  Decode $\hat{C}^1$ with $\mathcal{V}_d$ and **Emit** video chunk

  `/* Sliding */`

  Set $W_s \leftarrow \{\hat{C}^2, \hat{C}^3, \ldots, \hat{C}^N\}$

  Sample $\epsilon_s \sim \mathcal{N}(0, I)$

  Set $W_s \leftarrow \Psi(W_s, \epsilon_s, t_{1:N-1})$

motion embedding $m_{f,d}$ primarily controls fine-grained facial expressions. Although $m_{f,d}$ contains some pose-related cues, these signals have lower priority compared to the implicit 3D keypoints, as reflected in the result of $k_s + m_{f,d}$.

Motion threshold. We evaluate the effect of the motion threshold τ in our historical keyframe mechanism. As shown

Figure 2. Examples from LV100.

Figure 3. Effect of implicit 3D keypoints and facial motion embedding.

in Table 1, a smaller τ triggers more frequent history bank updates, providing richer historical information that helps stabilize long-term temporal consistency (lower FVD and tLP). However, more historical frames weaken the influence of the reference image I_R, leading to slight ID drift and consequently lower ID-SIM. Conversely, a larger τ better preserves identity but slightly degrades temporal stability due to fewer historical keyframes. Overall, we set $\tau = 17$ as it offers the best trade-off between identity preservation and temporal coherence.

VAE decoder. We further analyze the impact of different VAE decoders. As shown in Table 2, TinyVAE [2] significantly accelerates inference (up to 20 FPS) but introduces noticeable degradation in visual quality. SVD VAE [1] provides no improvement in temporal consistency and even

Table 1. Ablation study on motion threshold τ.

τ	ID-SIM↑	AED↓	APD↓	FVD↓	tLP↓
15	0.6924	0.7043	0.0309	510.9	12.69
16	0.6940	0.7039	0.0306	516.6	12.78
17	0.6983	0.7028	0.0305	520.6	12.83
18	0.7015	0.7047	0.0306	522.9	12.93
19	0.7097	0.7084	0.0304	526.5	13.05
20	0.7159	0.7099	0.0303	529.2	13.18

Table 2. Ablation study on VAE decoder.

decoder	ID-SIM↑	AED↓	APD↓	tLP↓	FPS↑
SVD VAE [1]	0.6920	0.7452	0.0489	12.97	11.4
SD VAE [4]	**0.6983**	**0.7028**	**0.0305**	**12.83**	15.8
TinyVAE [2]	0.6758	0.7593	0.0489	14.66	**20.0**

Table 3. Ablation on sampling schedules.

Schedule	ID↑	AED↓	APD↓
$[999, 666, 333, 0]$	**0.763**	0.714	0.033
$[999, 749, 499, 249]$	0.759	0.714	0.033

reduces runtime efficiency. In contrast, SD VAE [4] achieves the best overall performance while maintaining competitive inference speed.

Noise schedule. We empirically compare different noise schedules during the appearance distillation stage. As reported in Table 3, the schedule $[999, 666, 333, 0]$ yields better identity preservation compared to $[999, 749, 499, 249]$ while maintaining identical motion accuracy.

D. More experimental results

User study. To further evaluate the qualitative performance of our method against existing baselines, we conducted a user study. The study involved 30 participants who were presented with 20 diverse video sets. For each set, participants were asked to select the best video by answering four specific questions focused on: **A**ppearance, **M**otion, **T**emporal, and **O**verall quality. As shown in Fig. 4, PersonaLive performs comparably to the state-of-the-art diffusion model, X-NeMo, across the evaluated dimensions. Conversely, the GAN-based method, LivePortrait, consistently ranked the lowest in user preference.

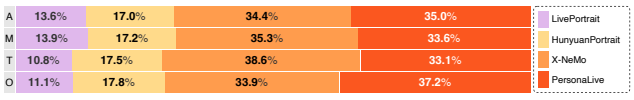

Figure 4. User study results.

Broader evaluations on hardware platforms. While streaming video generation methods are typically evaluated on a single high-end platform, we further assess PersonaLive across diverse hardware setups to demonstrate its practical applicability. Specifically, we evaluate the inference speed on an NVIDIA H100, an RTX 4090, and an RTX A6000 GPU, reporting frames per second (FPS) both with and without TensorRT (TRT) acceleration. Without TRT, our method runs at 15.8 FPS on the H100, 7.93 FPS on the RTX 4090, and 4.73 FPS on the RTX A6000. When utilizing TRT acceleration, the inference speed is significantly boosted to an impressive 24.3 FPS on the H100, maintains a real-time capability of 13.2 FPS on the consumer-grade RTX 4090, and reaches 6.87 FPS on the RTX A6000.

Visualization results. As shown in Fig. 5, Fig. 7, Fig. 9, and Fig. 11, we present additional visualization results under self-reenactment and cross-reenactment settings, further demonstrating the robustness and generalization ability of PersonaLive. Fig. 6, Fig. 8, Fig. 10, and Fig. 12 show long avatar videos synthesized by PersonaLive, highlighting its stability and consistency over long-term sequences.

E. Ethics Statement

Our work focuses on advancing portrait animation technology and is developed solely for academic and creative research. While the method itself is not intended for malicious use, we acknowledge its potential misuse in generating deceptive or non-consensual synthetic media. To promote transparency and responsible use, all generated content should be clearly marked as artificial, and the technology should be applied in accordance with ethical and legal standards.

References

[1] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023. 3, 4

[2] Ollin Boer Bohan. Tinyvae. 2023. 3, 4

[3] Jianzhu Guo, Dingyun Zhang, Xiaoqiang Liu, Zhizhou Zhong, Yuan Zhang, Pengfei Wan, and Di Zhang. Liveportrait: Efficient portrait animation with stitching and retargeting control. *arXiv preprint arXiv:2407.03168*, 2024. 1

[4] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022. 1, 4

[5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1

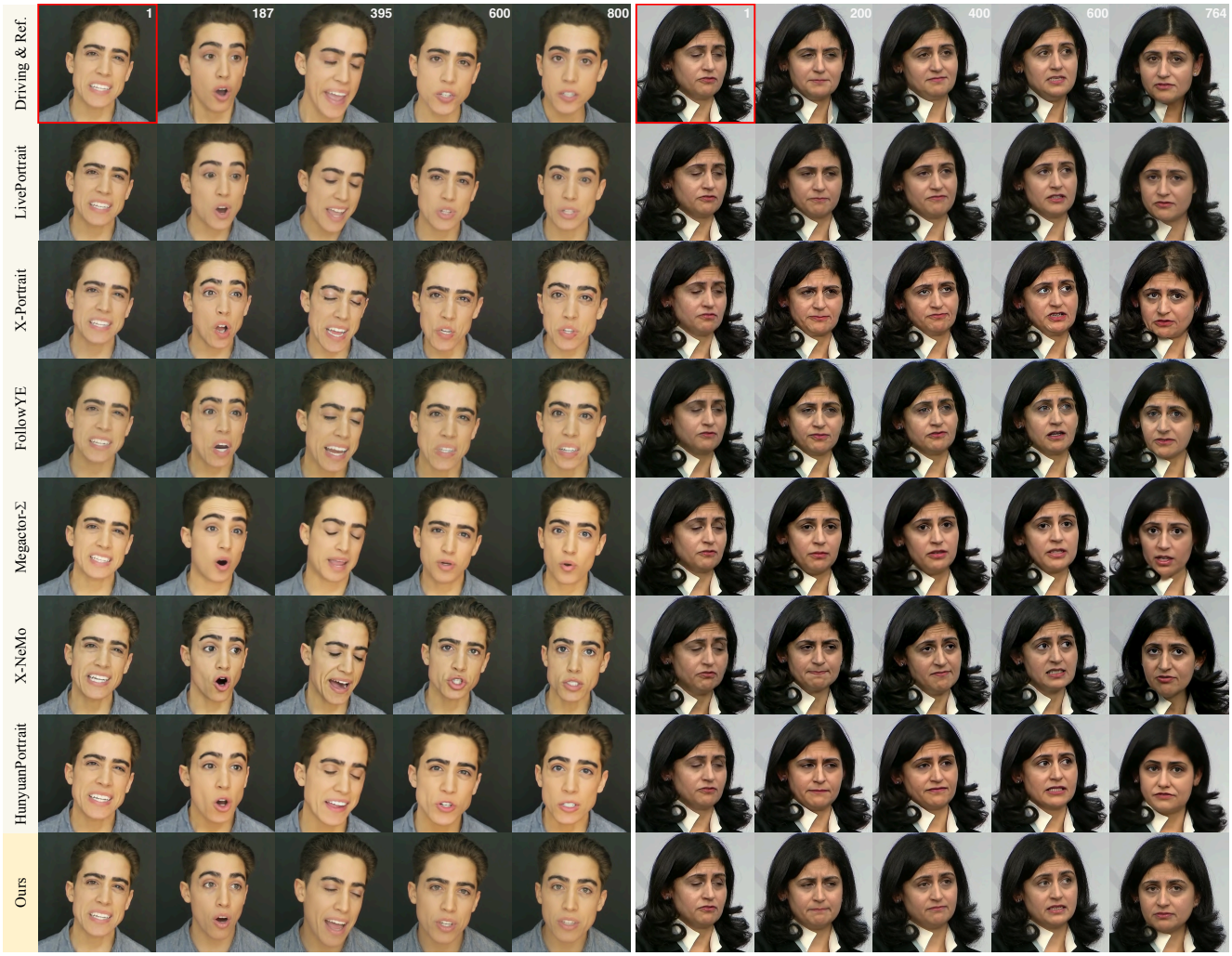

Figure 5. More visualizations of self-reenactment comparison (1/2). The images with red borders are the reference images.

Figure 6. Long avatar video results (1/4).

Figure 7. More visualizations of self-reenactment comparison (2/2). The images with red borders are the reference images.

Figure 8. Long avatar video results (2/4).

Figure 9. More visualizations of cross-reenactment comparison (1/2).

Figure 10. Long avatar video results (3/4).

Figure 11. More visualizations of cross-reenactment comparison (2/2).

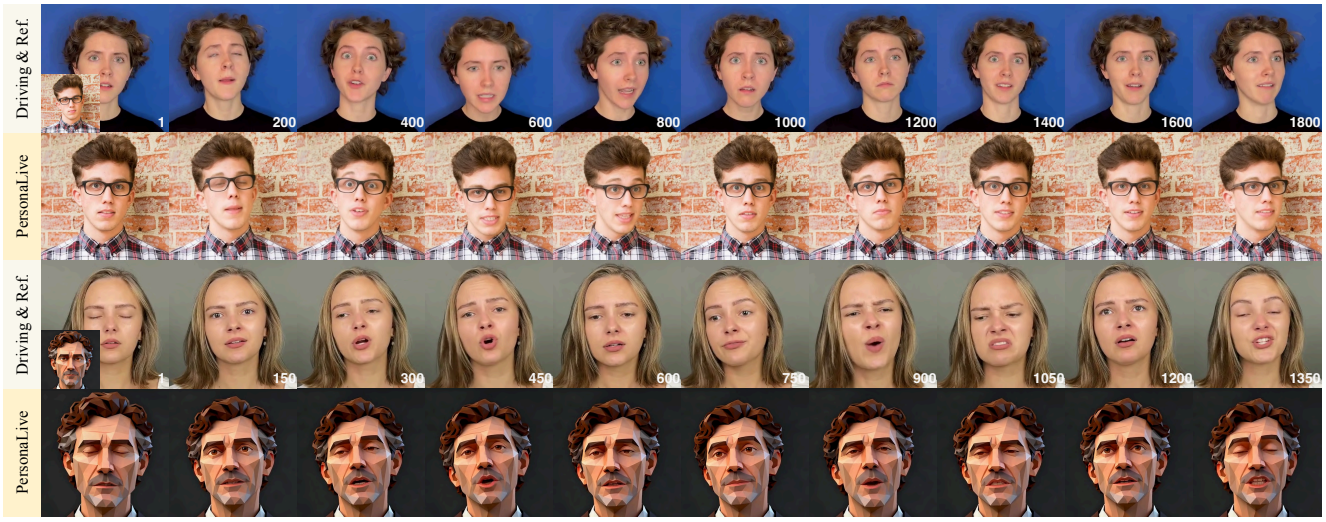

Figure 12. Long avatar video results (4/4).