Bitrate-Controlled Diffusion for Disentangling Motion and Content in Video – Supplemental Material –

Xiao Li[1†] Qi Chen[1,2,3†] Xiulian Peng[1] Kai Yu[2] Xie Chen[2,3✉] Yan Lu[1✉]
[1]Microsoft Research Asia [2]Shanghai Jiaotong University [3]Shanghai Innovation Institute
{cq1073554383, kai.yu, chenxie95}@sjtu.edu.cn {xili11, xipe, yanlu}@microsoft.com

This document contains additional implementation details for our training and evaluation as well as more results and ablations. We also provide a video demo for our results in a separate MPEG-4 file along with this document and we encourage readers to watch it.

A. Implementation Details

A.1. Video Representation Learning

Diffusion Model Preliminaries Diffusion models [6] have emerged as a powerful class of generative models that achieve state-of-the-art performance in various generation tasks. These models operate by defining a stochastic Markov process that progressively transforms data into noise and then learns to reverse this process to generate realistic samples. Formally, a diffusion model consists of a forward process (a.k.a., the diffusion process) and a reverse process (a.k.a., the denoising process). The forward process is defined as a Markov chain that iteratively adds Gaussian noise to a data sample $\mathbf{z}_0 \sim q(\mathbf{z})$ over T steps*:

$$q(\mathbf{z}_t|\mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t; \sqrt{\alpha_t}\mathbf{z}_{t-1}, (1-\alpha_t)\mathbf{I}), \tag{1}$$

where $\{\alpha_t\}$ are variance schedule parameters controlling the noise level at each step. By applying this process iteratively, the data distribution gradually approaches an isotropic Gaussian distribution. The reverse process, parameterized by a neural network $\boldsymbol{\theta}$, aims to learn the conditional distribution:

$$p_{\boldsymbol{\theta}}(\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{C}) = \mathcal{N}(\mathbf{z}_{t-1}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\mathbf{z}_t, t, \mathbf{C}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\mathbf{z}_t, t, \mathbf{C})). \tag{2}$$

where C is the condition signal to guide the denoising process. In our case, the conditional signal comes from our disentangle encoder $\mathcal{T}$, i.e., $C = (m, c) = T(z_0)$. [6] has revealed that by doing simplification and using parametrization trick, we can derive a closed form estimation for z_0 from Eq. (1) and Eq. (2):

$$\mathbf{z}_0 \approx \frac{1}{\sqrt{\bar{\alpha}_t}}(\mathbf{z}_t - \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{z}_t, t, \mathbf{C})), \tag{3}$$

where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. Sampling is performed by iteratively denoising from $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ back to $\mathbf{x}_0$.

The added gaussian noise $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t, \mathbf{C})$ is predicted by the denoising decoder $\mathcal{D}$.

The model is trained by predicting the noise $\boldsymbol{\epsilon}$ added at each step. Specifically, the training loss is formulated as:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{\mathbf{x}_0, \boldsymbol{\epsilon}, t}\Big[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\mathbf{x}_t, t)\|_2^2\Big], \tag{4}$$

where $\mathbf{x}_t$ is obtained by perturbing $\mathbf{x}_0$ with noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ according to the forward process. This training objective effectively reduces to a denoising autoencoder loss, enabling the model to learn an implicit score function that guides the reverse process.

Data pre-processing. We further apply a 2x temporal down-sampling to all video data during training. This is mainly to increase the maximum motion range the model can see during training under limited computational resources. All input video sequences are cropped and split into 4-second clips. Our cross-driven strategy (see main text) of splitting a clip in half assumes minimal content changes within each training clip, which can be ensured through data preprocessing.

Model Architecture and Hyper-parameters. For our disentangle encoder which extracts motion and content simultaneously, we adopt the T5 Encoder equipped with relative positional encoding. For diffusion denoiser, we use DiT architecture and insert temporal blocks in certain layers (as listed in temporal block indices of Tab. A4 and Tab. A5), the temporal blocks and spatial blocks follow the original design of DiT blocks [7]. The motion feature m is inserted into the decoder D by modulating the activations of each layer h using the Adaptive Group Normalization layers in

*We omit the video timestep here for simplicity. The subscript t denoted the denoising timestep.

each DiT block:

$$AdaGN(h, m, t) = m_s(t_s GroupNorm(h) + t_b) + m_b \tag{5}$$

where (t_s, t_b) and (z_s, z_b) are obtained by linear projections from the sinusoidal timestep embedding of denoising timestep t and the motion feature m respectively. The content feature c is directly concatenated to the input z_t.

Hyper-parameters for our representation learning model are listed in Tab. A4 and Tab. A5. Given the distinct data domains and dataset scales between the talking head dataset LRS3 and the synthetic cartoon characters dataset Sprites, we employ different hyperparameters for the respective models. Additionally, instead of utilizing a pre-trained VAE in the Sprites model, we directly process the video in the pixel space.

A.2. Auto-regressive motion generation

Data processing. Our auto-regressive motion generation model is trained on sequences of motion tokens; we generate these tokens from our pre-trained BCD model. The clip-wise content token is prepended to motion token sequences as the first sequence element, serving as a content prompt. To increase the motion diversity for both large motion and subtle motion, we implement data augmentation by downsampling videos temporally by a factor of 2 with a 50% probability.

Implementation. We use the GPT2 implementation from huggingface [10]. We modify the output head of GPT2 to match our grouped codebook outputs, i.e., we output the probability distribution of the next motion token across multiple codebooks, resulting in multi-group probability logits. We calculate the cross entropy loss between these probability distributions for each codebook with the ground truth motion tokens (in terms of codebook IDs) of the next timestep. The average of these cross-entropy losses serves as the loss function for our model. For inference, we sample predicted logits with the largest probability. Output video frames are generated by running our diffusion denoiser with predicted motion sequences and content features. Hyper-parameters for our motion generation model are listed in Tab. A6.

B. Additional Results

B.1. Motion Transfer and Generation

Our demo video showcases a comparison of cross-identity motion transfer results alongside motion generation outcomes. For a detailed view, please refer to our accompanying video file. We show more motion transfer results on Sprites in Fig. A1. Following previous methods, we also quantitatively assess disentanglement quality by measuring the attribute accuracy of videos generated by our model, using the pre-trained video classification network from [1]. Tab. A1 reports the motion and content accuracy for each attribute category, demonstrating that our method produces correct disentanglement results and achieves state-of-the-art performance on par with C-DSVAE [1].

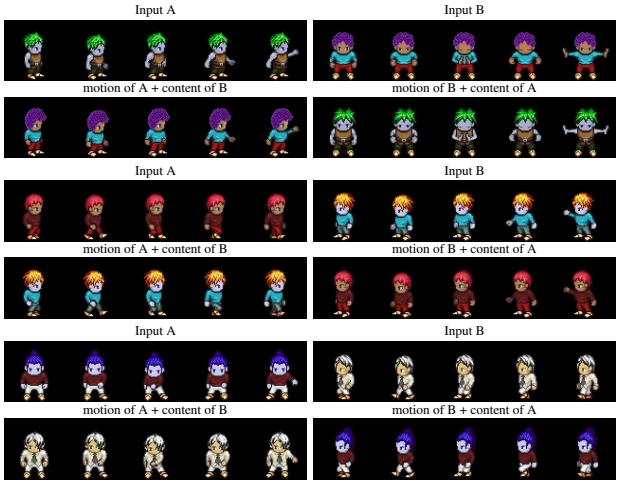

Figure A1. Additional motion transfer results on Sprites test set.

Table A1. Cross-driven attributes accuracies(%) of different methods on Sprites test set. "Action" represents the motion attribute and the others are content attributes.

Methods	Action↑	Skin↑	Pants↑	Top↑	Hair↑
C-DSVAE	100.00	100.00	100.00	100.00	100.00
Ours	100.00	100.00	100.00	100.00	100.00

Table A2. Quantitative comparison for motion transfer with different content feature frames.

Frames	FID↓	CSIM↑	Shape error↓ $\times 10^{-1}$	Motion error↓ $\times 10^{-2}$	Cross error↓ $\times 10^{-2}$
1	87.9	**0.692**	0.47	3.13	3.81
5	86.0	0.685	0.43	3.50	4.21
15	**85.6**	0.685	0.43	3.47	4.14
20	86.6	0.688	0.44	3.20	3.78
all	86.0	**0.692**	**0.41**	**3.13**	**3.67**

Table A3. Ablation study of the usage of bitrate loss.

Method	FID↓	CSIM↑	Shape error↓ $\times 10^{-1}$	Motion error↓ $\times 10^{-2}$	Cross error↓ $\times 10^{-2}$
w/o bitrate loss	92.7	**0.70**	0.59	4.81	6.21
Ours	**86.0**	0.69	**0.41**	**3.13**	**3.67**

B.2. Additional Ablation Studies

Number of frames for content extraction. Our method is able to extract content information from an arbitrary

Figure A2. Ablation study on target bitrate selection. *Column 1-2*: Content and motion reference. *Column 3-7*: Motion transfer results under different target bitrate. Bitrate 4kbps achieves the best tradeoff between image fidelity and disentanglement.

length of video. We analyzed the result of extracting content with different lengths of input and the results are shown in Tab. A2. Overall, the overall motion transfer quality is quite robust to the number of content frames with slight differences in terms of error metrics, while using more content reference frames improves the identity preservation and cross-identity motion transfer effect.

Effects of different bitrate. In the main paper, we quantitatively analyzed the effect of different bitrate. We additionally qualitatively visualize the effect in Figure A2: results under a lower bitrate (e.g. 2kbps) leads to degradation of some motion, while results under larger bitrate (>8kbps) demonstrate worse disentanglement of identity.

The usage of bitrate loss. We introduced bitrate loss during training to improve bitrate convergence. Ablation studies (see Tab. A3) show that removing it degrades performance, confirming its effectiveness.

C. Evaluation Metrics

To provide a more thorough evaluation of our method and other baselines on the cross-identity motion transfer task, we proposed additional metrics. These metrics are used together with existing metrics such as CSIM for evaluation results in the main paper. Here we explain the motivation for these additional metrics and their implementation details.

3D Face Fitting. All additional metrics rely on a dense reconstruction of 3D face meshes from both target and input videos. For this purpose, we utilize the method described in Wood et al.[5, 12] for several reasons. Firstly, it employs an optimization-based approach using dense landmark observations, ensuring accurate face reconstructions. In contrast, single-pass feed-forward reconstruction methods like those in Deng et al.[2] lack a per-video optimization process and fail to guarantee accurate reconstructions across videos with significant head pose and motion variations. Secondly, the parametric face model in [12] incorporates detailed blendshape expression bases within a continuous space, facilitating the assessment of expression transfer. Lastly, the dense landmark predictions and the parametric 3D face model from Wood et al. [12] are trained using high-quality synthetic face images covering a wide range of identities, expressions, head poses, genders, and races, minimizing potential ethical biases and privacy concerns during evaluation. In practice, the average fitting error of the face reconstruction method [12], as measured by reprojection error in pixel space on our test videos, is less than 1.5 pixels—indicative of high reconstruction quality.

The 3D face reconstruction of a given input video sequence V with n frames is characterized by identity coefficients $\beta \in R^c$, per-frame expression coefficients $\psi \in R^{n \times m}$, per-frame pose vector (in Euler angle) $\theta \in$

$R^{n\times 3}$, and per-frame translation vector $t \in R^{n\times 3}$. The computation of 3D mesh vertices from these coefficients leverages the parametric mesh model described in [11]: $\mathcal{M}(\beta,\theta,\psi,t) : R^{c\times m\times 3} \rightarrow R^{v\times 3}$. For a comprehensive mathematical explanation, we direct readers to the work of [11, 12].

Shape Error. The shape error e_s is calculated as the mean squared error (MSE) distance between the **identity meshes** of the content reference video and the target video. An **identity mesh** M_{id} is derived by setting the pose θ, expression ψ, and translation t to zero, i.e.,

$$M_{identity} = \mathcal{M}(\beta, 0, 0, 0) \tag{6}$$

$$e_s = MSE(M^{ref}_{identity}, M^{dst}_{identity}) \tag{7}$$

Thus, the shape error e_s isolates and quantifies the identity discrepancy between the target and the reference input, excluding all other variables.

Motion Error. Likewise, the motion error e_m is determined by calculating the average MSE distance between **motion meshes** of the motion reference video and the target video. This calculation is achieved by setting the identity coefficients to zero:

$$M_{motion} = \mathcal{M}(0, \theta, \phi, t) \tag{8}$$

$$e_m = MSE(M^{ref}_{motion}, M^{dst}_{motion}) \tag{9}$$

It is important to note that head poses are also incorporated into the motion mesh calculation.

Cross Transfer Error. Finally, the cross transfer error e_c is calculated to evaluate the overall quality of the cross-identity motion transfer task by utilizing the fully fitted meshes:

$$M_{full} = \mathcal{M}(\beta, \theta, \phi, t) \tag{10}$$

$$e_c = MSE(M^{ref}_{full}, M^{dst}_{full}) \tag{11}$$

Discussion. Existing methods evaluate identity preservation using cosine similarity (CSIM) [3, 4, 8, 9]. However, we have found these methods to be often unreliable and unpredictable for evaluating motion on a larger scale. CSIM tends to underestimate identity similarity in cases of significant head poses and overestimate it when face images exhibit warping distortions that alter facial shape. Figure A3 presents some illustrative examples: images of the same individual with large poses show low CSIM scores, while images with noticeable warping artifacts receive high CSIM scores. Our metric for measuring shape error complements the CSIM metric and offers greater robustness against pose variations.

Evaluating motion accuracy in cross-identity motion transfer poses a significant challenge due to the difficulty in obtaining ground truth results for this generative task. Existing methods have resorted to indirect metrics, with ARD (average rotation distance)[3, 4] and AUH (average facial action unit hamming distance)[3, 9] being the most commonly used. The ARD metric focuses exclusively on head poses, particularly head rotations, whereas the AUH metric, as proposed in [3], quantifies the binary hamming distance between two boolean vectors derived from a subset of the facial action coding system (FACS). Our motion error metric offers a comprehensive assessment of head poses, facial expressions, and additional motion nuances.

Finally, we have observed that each metric proposed in previous studies is capable of evaluating only a single aspect of cross-identity motion transfer quality. Our cross-transfer error metric addresses this gap by assessing the overall quality of cross-identity motion transfer.

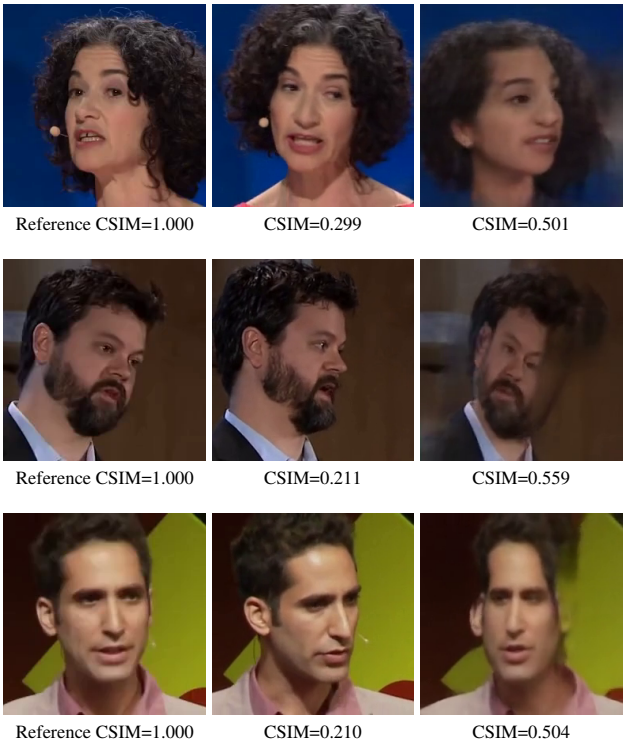

Figure A3. CSIM analysis. Images with the same identities but different poses compared to the reference images get low CSIM, but images with obvious warping artifacts achieve high CSIM, indicating the probably unreliable evaluation results of the CSIM metric.

Table A4. Hyperparameters of representation learning model on LRS3 dataset.

Hyperparameter	Value
Disentangle Encoder	
Architecture	T5 Encoder
Transformer hidden size	512
Transformer feed-forward dimension	2048
Transformer layers	12
Attention heads	8
Denoise Decoder	
Architecture	DiT-B/4
Resolution	32
Input channels	4
Patch size	4
Transformer hidden size	768
Transformer feed-forward dimension	3072
Transformer layers	12
Attention heads of spatial block	12
Attention heads of temporal block	12
Temporal block indices	[0, 2, 4, 6, 8, 10]
Classifier-free guidance masking rate	0.1
Sampling	
Strategy	EDM
Classifier-free guidance	1.5
Diffusion denoising steps	128
Bitrate-controlled Vector Quantization	
Codebook numbers	64
Code dimension per codebook	16
Number of entries per codebook	32
Target bitrate	4kbps
Maximum of gumbel-softmax temperature	1.5
Minimum of gumbel-softmax temperature	0.1
Temperature decaying rate	0.9999972
Optimization	
Optimizer	AdamW
Learning Rate	2e-4 for denoise decoder, 1e-4 for others
Batch size	32 (4×A100 GPU×8)
Weight Decay	0.01
β	(0.9,0.999)

Table A5. Hyperparameters of representation learning model on Sprites dataset.

Hyperparameter	Value
Disentangle Encoder	
Architecture	T5 Encoder
Transformer hidden size	384
Transformer feed-forward dimension	1536
Transformer layers	12
Attention heads	6
Denoise Decoder	
Architecture	DiT-S/8
Resolution	64
Input channels	3
Patch size	8
Transformer hidden size	384
Transformer feed-forward dimension	1536
Transformer layers	12
Attention heads of spatial block	6
Attention heads of temporal block	6
Temporal block indices	[0, 2, 4, 6, 8, 10]
Classifier-free guidance masking rate	0.1
Sampling	
Strategy	EDM
Classifier-free guidance	1.5
Diffusion denoising steps	50
Bitrate-controlled Vector Quantization	
Codebook numbers	1
Code dimension per codebook	384
Number of entries per codebook	64
Target bitrate	150bps
Maximum of gumbel-softmax temperature	2.0
Minimum of gumbel-softmax temperature	0.5
Temperature decaying rate	0.9999972
Optimization	
Optimizer	AdamW
Learning Rate	5e-5
Batch size	128
Weight Decay	0.01
β	(0.9,0.999)

Table A6. Hyperparameters of motion generation model on LRS3 dataset.

Hyperparameter	**Value**
Model	
Architecture	GPT-2
Hidden size	1024
Layers	12
Heads	8
Optimization	
Optimizer	AdamW
Learning Rate	0.0001
Batch size	256 (8×H100 GPU × 32)
Weight Decay	0.01
β	(0.9,0.999)

References

[1] Junwen Bai, Weiran Wang, and Carla P Gomes. Contrastively disentangled sequential variational autoencoder. *Advances in Neural Information Processing Systems*, 34: 10105–10118, 2021. 2

[2] Yu Deng, Jiaolong Yang, Sicheng Xu, Dong Chen, Yunde Jia, and Xin Tong. Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019. 3

[3] Michail Christos Doukas, Stefanos Zafeiriou, and Viktoriia Sharmanska. Headgan: One-shot neural head synthesis and editing. In *Proceedings of the IEEE/CVF International conference on Computer Vision*, pages 14398–14407, 2021. 4

[4] Yue Gao, Yuan Zhou, Jinglu Wang, Xiao Li, Xiang Ming, and Yan Lu. High-fidelity and freely controllable talking head video generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5609–5619, 2023. 4

[5] Charlie Hewitt, Fatemeh Saleh, Sadegh Aliakbarian, Lohit Petikam, Shideh Rezaeifar, Louis Florentin, Zafiirah Hosenie, Thomas J Cashman, Julien Valentin, Darren Cosker, and Tadas Baltrušaitis. Look ma, no markers: holistic performance capture without the hassle. *ACM Transactions on Graphics (TOG)*, 43(6), 2024. 3

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 1

[7] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. 1

[8] Aliaksandr Siarohin, Stéphane Lathuilière, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. First order motion model for image animation. *Advances in neural information processing systems*, 32, 2019. 4

[9] Ting-Chun Wang, Arun Mallya, and Ming-Yu Liu. One-shot free-view neural talking-head synthesis for video conferencing. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10039–10049, 2021. 4

[10] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019. 2

[11] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Thomas J Cashman, and Jamie Shotton. Fake it till you make it: face analysis in the wild using synthetic data alone. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3681–3691, 2021. 4

[12] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Matthew Johnson, Jingjing Shen, Nikola Milosavljević, Daniel Wilde, Stephan Garbin, Toby Sharp, Ivan Stojiljković, et al. 3d face reconstruction with dense landmarks. In *European Conference on Computer Vision*, pages 160–177. Springer, 2022. 3, 4