Control4D: Efficient 4D Portrait Editing with Text

Supplementary Material

1. Implementation Details

1.1. GaussianPlanes

Spatial triplane decomposition. In 3D spatial space, we decompose the attributes of each point in the canonical Gaussian point cloud, including color, opacity, latent feature, and rotation. The decomposition of each attribute utilizes three corresponding feature planes. We employ a HashGrid [5] to represent each feature surface, with a hierarchy of resolutions at 16 levels, where the scale of each level is 1.3 times that of the preceding level. Each level contains 2 feature channels, and the encoded results are mapped to the corresponding attributes through a 256-unit MLP network.

4D flow decomposition. For the 4D flow, we have implemented a hierarchical decomposition using Tensor4D [9]. In our approach, we decompose the flow of each point's position and rotation at each moment within the Gaussian point cloud. This decomposition employs 9 feature planes. Each feature plane is represented using a HashGrid consistent with the spatial feature planes. Subsequently, the encoded results are first fused individually for the corresponding three planes using three 256-unit MLPs, and then a final attribute output is produced through another 256-unit MLP.

1.2. 4D Generator

Network Structure. We adopt a network architecture similar to pix2pixHD [10] for the GAN's generator and discriminator. In our generator, we introduce some modifications: the input features comprise both RGB and latent features, which are concatenated together as the input. Additionally, in the intermediate layers, we concatenate the feature with its global feature code. The architecture includes three downsample layers, three middle blocks, and five upsample layers, thereby achieving a 4x super-resolution in the final output. The number of the base feature channel in the network is 32. The input for the 4D generator is at a resolution of 256, and it outputs images at a resolution of 1024. As for the discriminator, we utilize the same architecture as the pix2pixHD discriminator.

Global Encoder. We utilize MobileNet [2] to extract the global code. Initially, the image is resized to a resolution of 224, followed by feature extraction through MobileNet's layers. We maps the final feature of MobileNet to a 64-dimensional global feature code.

Local Encoder. We employ an encoder similar to the VAE encoder used in Stable Diffusion [6] as our local encoder. Our local encoder compresses the original image to a quarter of its original size through two downsample layers, and the number of ”z_channels” is set to 4. The base channel number of our network is 32.

1.3. Diffusion-based Editor

We utilize ControlNet [11] as our diffusion-based editor. To achieve better control effects, we employ both normal and OpenPose as control signals. The control strength for normal is set at 0.5, while for OpenPose, it is 1.0. Additionally, we use the RealisticVision [8], an SD1.5 model, to obtain more realistic editing effects. Additionally, before feeding the images into ControlNet, we resize the 1024-resolution images down to a resolution of 512.

1.4. 4D Reconstruction based on GaussianPlanes

We initially reconstruct the 4D scenes using GaussianPlanes. Our experiments primarily focus on the Tensor4D dataset, and we also showcase some results in challenging scenes, including those from Neural3DVideo [3], ENeRF [4] and InstructNeRF2NeRF [1]. For the Tensor4D dataset, we employ a Gaussian sphere for initialization, with a point cloud size of 5,000 and a radius of 1. For the ENeRF, Neural3DVideo and InstructNeRF2NeRF datasets, we utilize the point cloud from the first frame of COLMAP [7] as the initialization.

During the training process, to ensure the stability of the canonical Gaussian point cloud, we adopt a weighted strategy for selecting training frames. There is a 50% probability that we choose all frames from the first moment and a 50% probability that we randomly select frames from other moments. This approach is designed to balance the representation of the initial frame with the dynamic aspects of the remaining video content. Simultaneously, we are also training the 4D generator in preparation for 4D editing.

During the training process, the learning rate for the point cloud positions linearly decays from 0.00016 to 0.0000016. The learning rates for scaling, color, opacity, and rotation are set at 0.005, while the learning rate for flow is 0.00025. The learning rate for the 4D generator is 0.001. The gradient threshold for splitting is set to 0.0002, and the interval of densification and pruning is 200. We employ L1 loss to train GaussianPlanes, with a weight of 1.0. For training the generator, we use L1 loss, perceptual loss, and GAN loss, with weights of 1.0, 1.0, and 0.01, respectively. The discriminator is trained using GAN loss and gradient penalty regularization, with respective weights of 1.0 and 0.01.

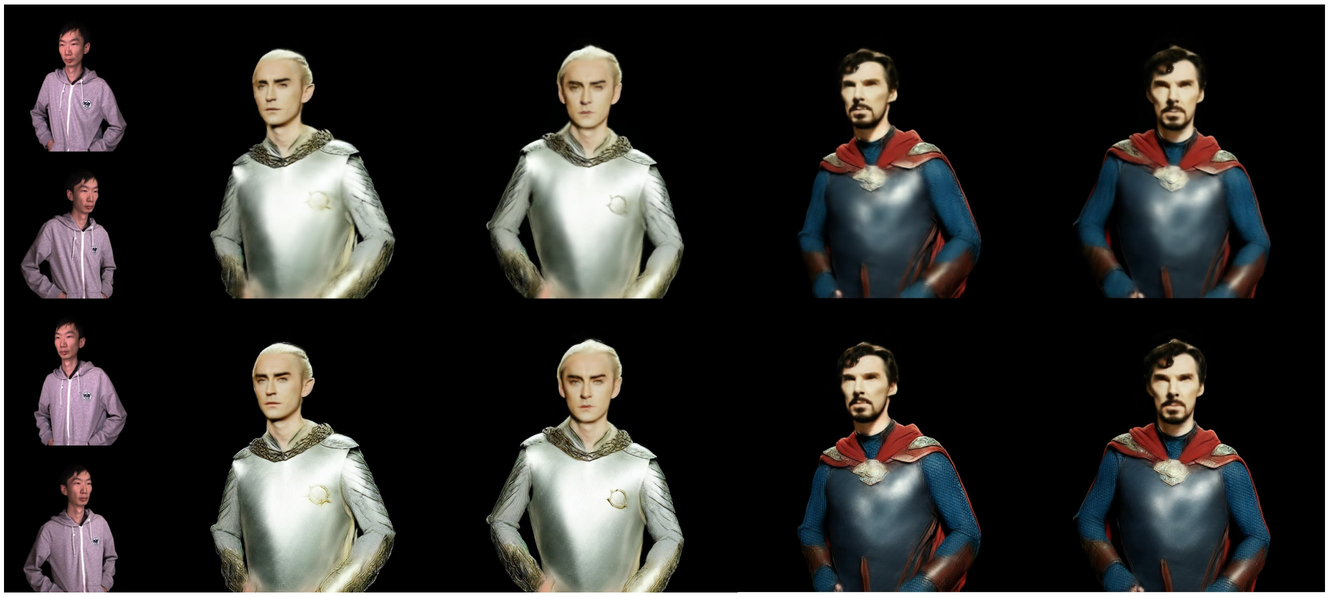

Figure 1. Ablation study of 4D generator. First row: Results utilizing only GaussianPlanes, second row: Results achieved by combining 4D generator. The prompts used here are ”Elf King” and ”Doctor Strange”.

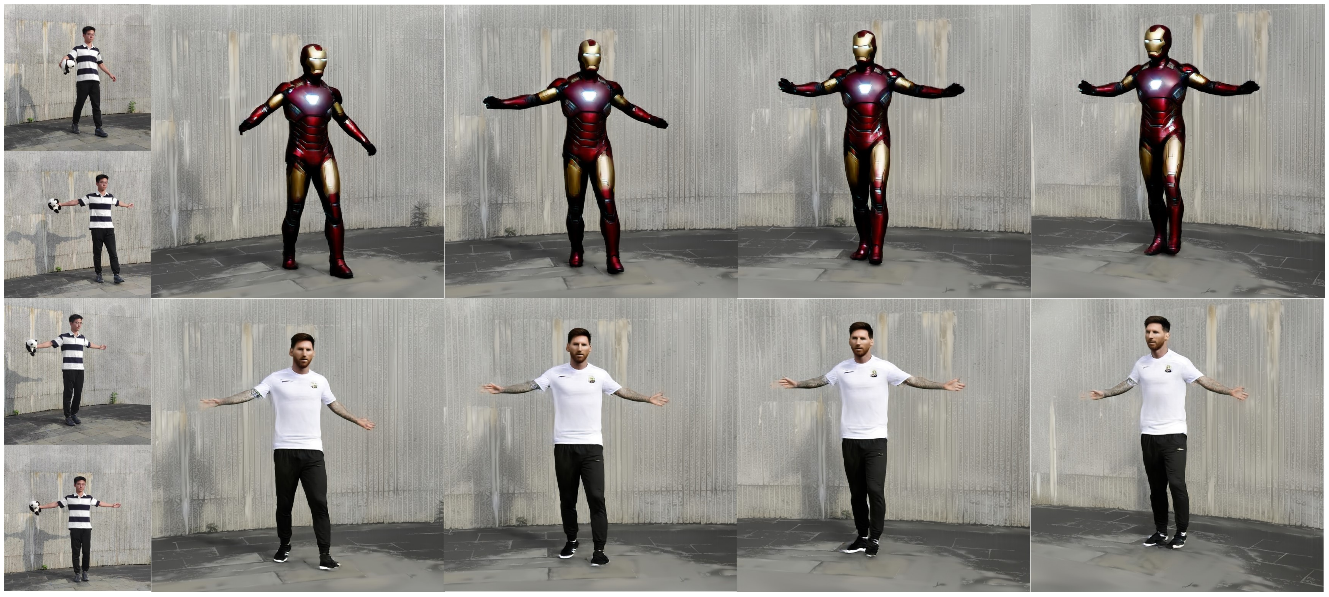

Figure 2. Control4D results on ENeRF dataset. The prompts are ”Iron Man” and ”Lionel Messi”.

1.5. 4D Editing Process

During the 4D editing process, we utilize two GPUs (RTX3090) for training. One GPU is dedicated to running edits for each image, while the other GPU is tasked with running GaussianPlanes and rendering images with the 4D generator. These two processes are executed in parallel. For complex multi-camera 4D scenes, including Neural3DVideo and ENeRF, we do not edit using all cameras. Instead, we use images from all cameras at the first moment and randomly select images from four cameras at other moments to form the dataset.

The first 1000 steps of our training are for static editing, followed by 4000 steps for dynamic editing. During static editing, the noise added to the diffusion-based editor is $U(0.02, 0.98)$, which is reduced to $U(0.02, 0.6)$ for dynamic editing. The steps of diffusion model is set to 20 and we use the DDIM scheduler. To enhance robustness, we lower the learning rates during the editing process. Specifi-

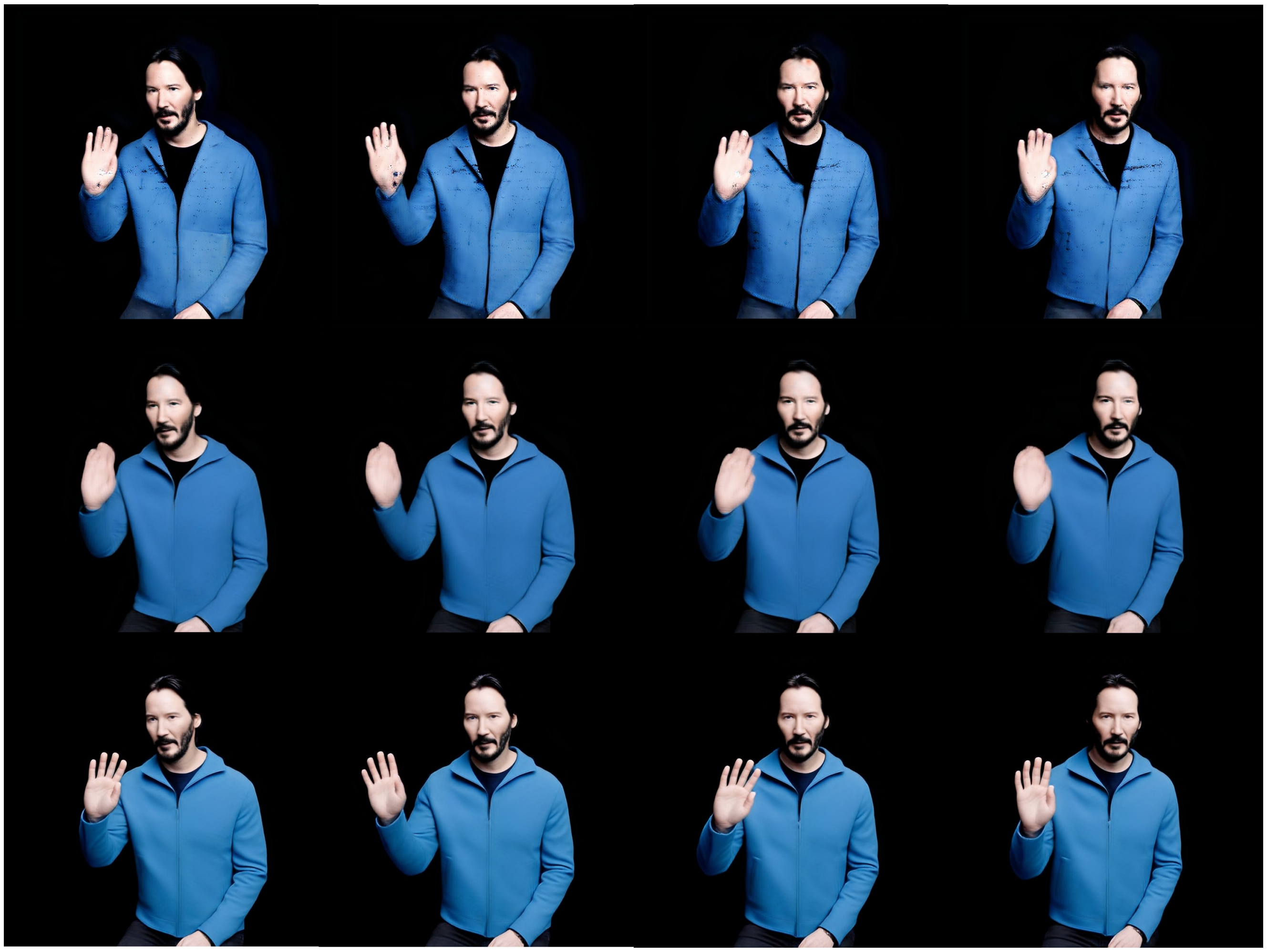

Figure 3. Ablation Study of GaussianPlanes. First row: w/o spatial triplane decomposition. Second row: w/o 4D flow decomposition. Third row: Control4D results.

cally, the learning rate for point positions is 0.000016, while the learning rates for scaling, color, opacity, and rotation remain at 0.005, and the learning rate for flow is 0.0001. The 4D generator's learning rate is set at 0.0001. In the editing process, we don't split or prune Gaussian points. In the multi-level guidance, the probability of selecting each level is equal. The weights of the various losses in 4D editing remain consistent with those in the reconstruction process.

2. More Comparisons

We conducted further comparisons with InstructNeRF2NeRF on their dataset. As shown in Fig. 9, our method noticeably surpasses InstructNeRF2NeRF in terms of realism and quality. Additionally, our optimization process is extremely fast, completing editing tasks in just 5 minutes, whereas InstructNeRF2NeRF requires at least about 5 hours. This makes our method 60 times more efficient than InstructNeRF2NeRF.

3. More Ablation Study

GaussianPlanes. We conducted more ablation studies on GaussianPlanes. As shown in Fig. 3, when the spatial triplane decomposition is not used, the results exhibit significant noise. Without the decomposition of flow, the edited results become noticeably blurred, with evident occurrences of jittering, which can be clearly observed in the supplementary video. This demonstrates that our proposed plane decomposition method makes Gaussian Splatting more structured and significantly enhances its robustness.

4D generator. More ablation experiments were conducted on our proposed 4D generator. As illustrated in Fig. 1, without the use of the generator and relying solely on GaussianPlanes, the images noticeably lose many high-quality de-

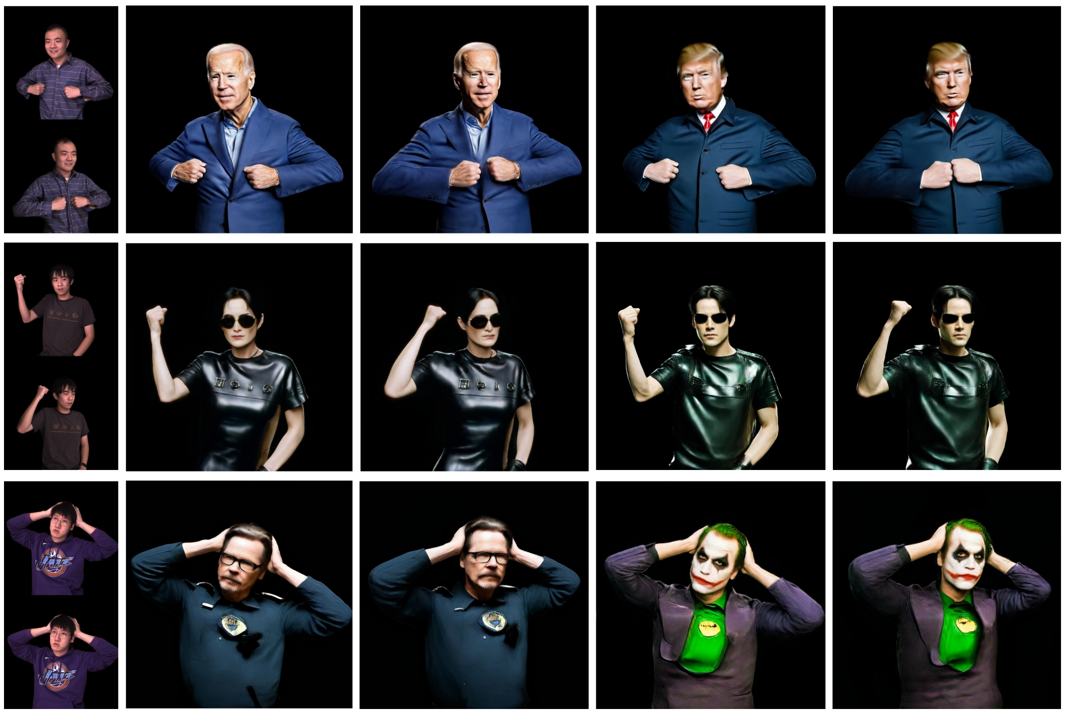
Figure 4. More Control4D results on Tensor4D dataset. The prompts are ”Joe Biden wearing suit”, ”Donald Trump wearing suit”, ”Trinity in The Matrix”, ”Neo in The Matrix”, ”James Gordon in Batman”, and ”Joker in Batman”.

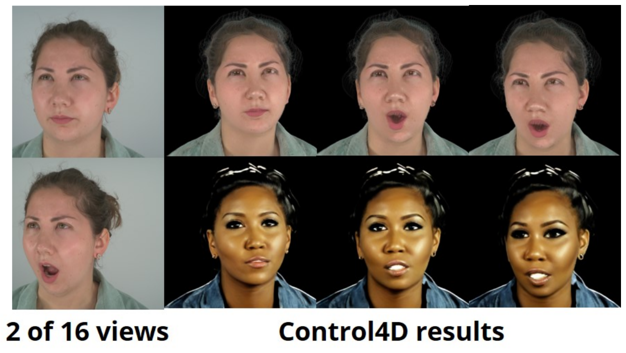

Figure 5. Experiment on facial expression. Text prompt is *”A black woman”*.

tails and appear blurry. This validates the role of our 4D generator in enhancing quality.

4. More Results

Our method is applicable not only to the editing of half-body and heads but also to complex 4D scenes and full-body human editing. More results are illustrated in Fig. **??**, 4, 7, and 2. Our method could also be applied to general dynamic scenes and control human facial expressions by landmark ControlNet as shown in Fig. 5 and 6. Please refer to our supplementary video for dynamic editing effects.

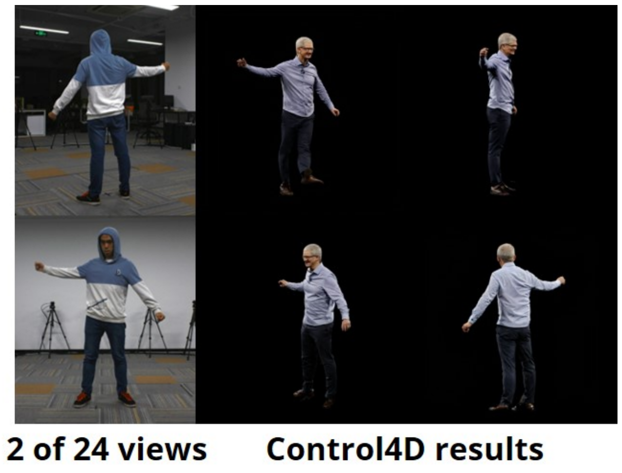

Figure 6. Experiment on 360-degree human avatar scene. Text prompt is *”Tim Cook”*.

5. Social Impact

The primary goal of our method is to provide users with an advanced tool for dynamic human editing in complex 4D scenes. While our approach enables intricate editing of full-body humans and facilitates creative expression in digital environments, it also raises concerns about potential misuse, such as creating deceptive or misleading content. This

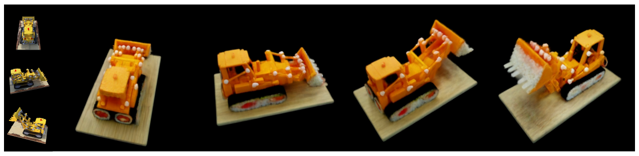

Figure 7. Experiment on general dynamic 360-degree cases. The text prompt is *"A sushi excavator"*.

challenge is not exclusive to our method but is a common issue across various generative modeling techniques. Additionally, in line with ethical considerations, our approach underscores the importance of diversity, including aspects of gender, race, and cultural representation. It is crucial for ongoing and future research in generative modeling to continuously engage with and reevaluate these ethical considerations to ensure responsible use and positive societal impact.

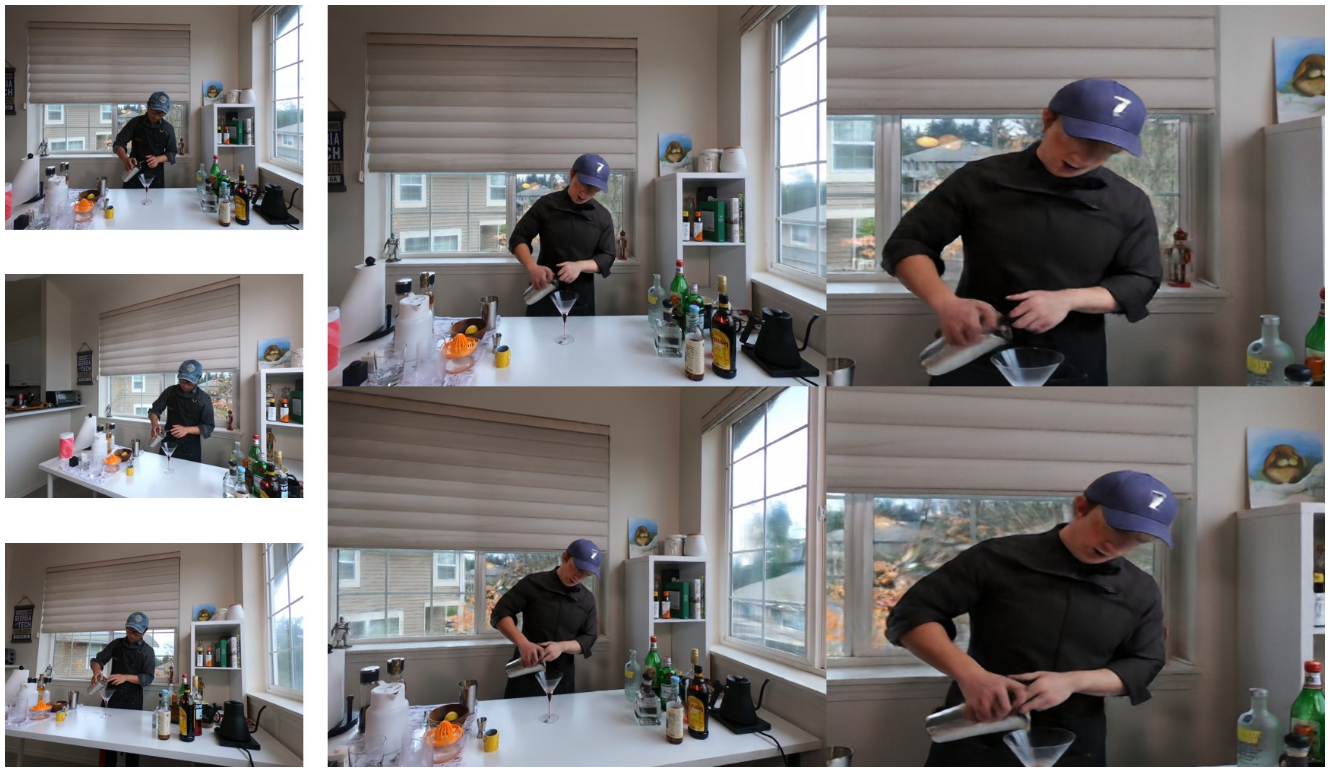
Figure 8. Control4D result on neural 3D video dataset. The prompt is ”Mark Zuckerberg”.

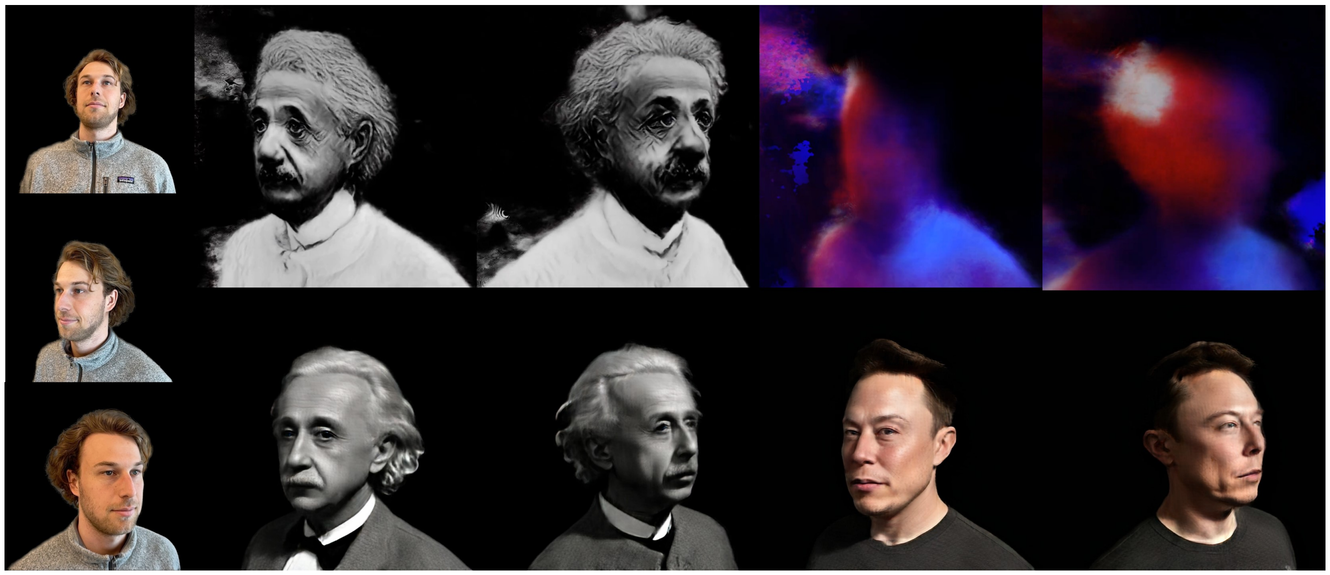
Figure 9. Comparison with InstructNeRF2NeRF on InstructNeRF2NeRF dataset. First row: InstructNeRF2NeRF results with prompts ”Turn him into Albert Einstein” and ”Turn him into Elon Musk”. Second row: Control4D results with prompts ”Albert Einstein” and ”Elon Musk”.

References

[1] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. *arXiv preprint arXiv:2303.12789*, 2023. 1

[2] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 1

[3] Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5521–5531, 2022. 1

[4] Haotong Lin, Sida Peng, Zhen Xu, Yunzhi Yan, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Efficient neural radiance fields for interactive free-viewpoint video. In *SIGGRAPH Asia Conference Proceedings*, 2022. 1

[5] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 1

[6] 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

[7] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1

[8] SG_161222. Realistic vision v5 stable diffusion checkpoint, 2023. 1

[9] Ruizhi Shao, Zerong Zheng, Hanzhang Tu, Boning Liu, Hongwen Zhang, and Yebin Liu. Tensor4d: Efficient neural 4d decomposition for high-fidelity dynamic reconstruction and rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16632–16642, 2023. 1

[10] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018. 1

[11] Lvmin Zhang and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. *arXiv preprint arXiv:2302.05543*, 2023. 1