Instruct 4D-to-4D: Editing 4D Scenes as Pseudo-3D Scenes Using 2D Diffusion

Linzhan Mou[1†] Jun-Kun Chen[2†] Yu-Xiong Wang[2]
[1]Zhejiang University [2]University of Illinois Urbana-Champaign [†]Equal Contribution
moulz@zju.edu.cn {junkun3,yxw}@illinois.edu

Supplementary Material

Our supplementary material includes implementation details and some additional experiments.

A. Demo Video & Code

We provide a demo video `demo.mp4`, visualizing our editing results as videos.

We also provide some source code in the `code` folder, which contains the code of our main pipeline.

B. Implementation Details

B.1. 4D Representation

The primary input of our method is a 4D NeRF representation, acquired through the NeRFPlayer [7]. Our framework is general, and therefore, any 4D scene representation adopting RGB observations as supervision can be used. In the implementation of the multi-camera scenes, we use TensoRF [2]-based NeRFPlayer as NeRF backbone and follow the same setting as in their experiments on the multi-camera DyNeRF [4] scenes. For our experiments, we extract 50 frame segments, from the full-length videos and downsample images to 1352×1014 for 4D representation and editing. Furthermore, to show the capabilities of our method in long-term videos(pseudo scenes), we also use full-length 300-frame videos with 676×507 resolution for additional evaluation, both trained for 100,000 iterations per scene. In the implementation of the monocular scenes, we use Instant-NGP [5]-based NeRFPlayer for HyperNeRF [6] dataset, trained for 60, 000 iterations per scene, and Nerfacto-based NeRFPlayer from NeRFStudio [8] for DyCheck [3] dataset, trained for 30, 000 iterations per scene.

B.2. Anchor-Aware Attention

We introduce the anchor-aware attention module to augment IP2P, enhancing the appearance control by the reference anchor frame. Specifically, each latent feature calculates the corresponding key and value features based on the concatenation of the anchor frame z_{v_a} and the current frame z_{v_i}. The definition of anchor-aware attention is as follows:

$$Q = W^Q z_{v_i}, K = W^K \left[z_{v_a}; z_{v_i}\right], V = W^V \left[z_{v_a}; z_{v_i}\right]$$

where W are projection layers in attention shared across space and time, and $[\cdot]$ denotes concatenation operation. Besides, we employ the original spatial self-attention weights as initialization. In each pseudo-view editing process, we use the first frame which is edited as the anchor frame to provide appearance reference.

Fig. B.3-(a) shows that anchor-aware attention clearly improves the editing consistency across views and batches.

B.3. Sliding Window-Based Pseudo-View Editing Method

As visualized in Fig. B.3-(b), we filter inferior flow predictions by leveraging the forward-backward consistency constraint. Also, the anchor-aware IP2P is well designed both to inpaint occluded areas and to repaint the whole part based on the appearance of the anchor frame. This leads to reasonable editing results even when the RAFT prediction is not accurate.

B.4. 4D Editing Procedure

We re-initialize the optimizers in the trained NeRFPlayer model and utilize Anchor-Aware Instruct-Pix2Pix [1] as our 2D editing model. For the diffusion model, varying hyperparameters are applied at distinct phases. During the anchor frames editing stage, the input diffusion timestep t decays from 0.98 to 0.7 in a cosine annealing manner. We employ 20 diffusion steps for this phase. During the inpainting stage after optical flow warping, we set timestep t to 0.6, and only used 3 diffusion steps. As optical flow

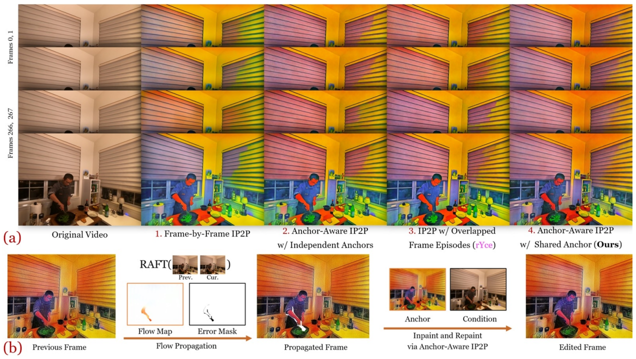
Frames 0, 1
Frames 266, 267
(a)
Original Video
1. Frame-by-Frame IP2P
2. Anchor-Aware IP2P w/ Independent Anchors
3. IP2P w/ Overlapped Frame Episodes (rYce)
4. Anchor-Aware IP2P w/ Shared Anchor (**Ours**)
RAFT( )
Prev.
Cur.
Flow Map
Error Mask
Flow Propagation
Anchor
Condition
Inpaint and Repaint via Anchor-Aware IP2P
(b)
Previous Frame
Propagated Frame
Edited Frame

Figure B.3. (a) The usage of anchor-aware attention clearly improves the editing consistency across views and batches. (b) A more detailed visualization of our sliding window pseudo-view editing method through an optical flow-based propagation and anchor-aware inpainting and repainting.

warping extensively propagates across most areas, the diminished timestep and fewer diffusion steps contribute to efficient inpainting while preserving the original data distribution. To control the extent of alterations for specific edits, we calibrate the classifier-free guidance weights for each scene, defined as S_T and S_I for the text instruction and original image, respectively. For object-focused editing tasks, We set S_I= 1.5 and S_T = 7.5, whereas, for style transfer tasks, the settings are S_I= 1.5 and S_T = 9.5. The number of iterations varies in different scenes shown in the paper. Due to the parallelization strategy, we don't need to trade off between NeRF training and image editing, thus we use 15,000 iterations for monocular scenes and 25,000 iterations for multi-camera scenes (with over 1000 images each). While each experiment is conducted on 2 NVIDIA A40 GPUs, most scenes converge to a fine-grained edited scene within 1.5 hours.

C. Discussion

C.1. Limitations

The major limitation of our Instruct 4D-to-4D is rooted in the limitation of IP2P [1] – given that Instruct 4D-to-4D edits scenes by distilling from IP2P, its editing capability is capped by IP2P. We will fail in the failure cases of IP2P, and perform poorly if IP2P does so. In addition, as we are using the original IP2P without fine-tuning, we lose the ability to leverage the per-scene information to facilitate editing. On the other hand, we benefit from the high efficiency of such a training-free pipeline.

Moreover, without input of 3D geometry information or 4D movement information, IP2P is unaware of any 3D/4D information, including position, geometry, and timestep. It can only infer the correlation between frames using cross-attention modules based on the RGB images, which might be inaccurate and lead to inconsistent editing results. Note that the source of consistency in Instruct 4D-to-4D is primarily the cross-attention module, which is a soft mechanism without supervision or enforcement. While Fig. **??** shows that our IP2P can generate consistent editing results under certain situations, this is not always guaranteed.

Some instructions may indicate shape editing. Instruct 4D-to-4D could only perform simple shape editing where the modification is near the surface, *e.g.*, 'change the cat to a fox' which slightly changes the head shape. Instruct 4D-to-4D does not support aggressive shape editing, *e.g.*, 'remove the cat,' like most of the instruction-guided 3D scene editing methods, or editing the movement of an object.

C.2. Future Directions

One possible future direction is to support per-scene training, *e.g.*, fine-tuning RAFT for more accurate optical flow prediction, augmenting IP2P to support 3D and 4D information, *etc*. This could lead to a more powerful IP2P towards more consistent 4D editing.

References

[1] Tim Brooks, Aleksander Holynski, and Alexei A Efros. InstructPix2Pix: Learning to follow image editing instructions. In *CVPR*, 2023. 1, 2

[2] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. TensoRF: Tensorial radiance fields. In *ECCV*, 2022. 1

[3] Hang Gao, Ruilong Li, Shubham Tulsiani, Bryan Russell, and Angjoo Kanazawa. Monocular dynamic view synthesis: A reality check. *NeurIPS*, 2022. 1

[4] 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 *CVPR*, 2022. 1

[5] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *TOG*, 2022. 1

[6] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M Seitz. Hypernerf: A higher-dimensional representation for topologically varying neural radiance fields. *arXiv preprint arXiv:2106.13228*, 2021. 1

[7] Liangchen Song, Anpei Chen, Zhong Li, Zhang Chen, Lele Chen, Junsong Yuan, Yi Xu, and Andreas Geiger. NeRFPlayer: A streamable dynamic scene representation with decomposed neural radiance fields. *TVCG*, 2023. 1

[8] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *SIGGRAPH*, 2023. 1