LatentEditor: Text Driven Local Editing Using T2I Diffusion Models

Umar Khalid[1]*, Hasan Iqbal[2]*, Nazmul Karim[1]*, Muhammad Tayyab[1], Jing Hua[2], Chen Chen[1]
[1]University of Central Florida, [2]Wayne State University
*Equal Contribution

https://latenteditor.github.io/

European Conference on Computer Vision (ECCV) 2024
Milan, Italy

Wed 2 Oct 16:30-18:30 EDT, Poster # 69

Image Generation to Image Editing

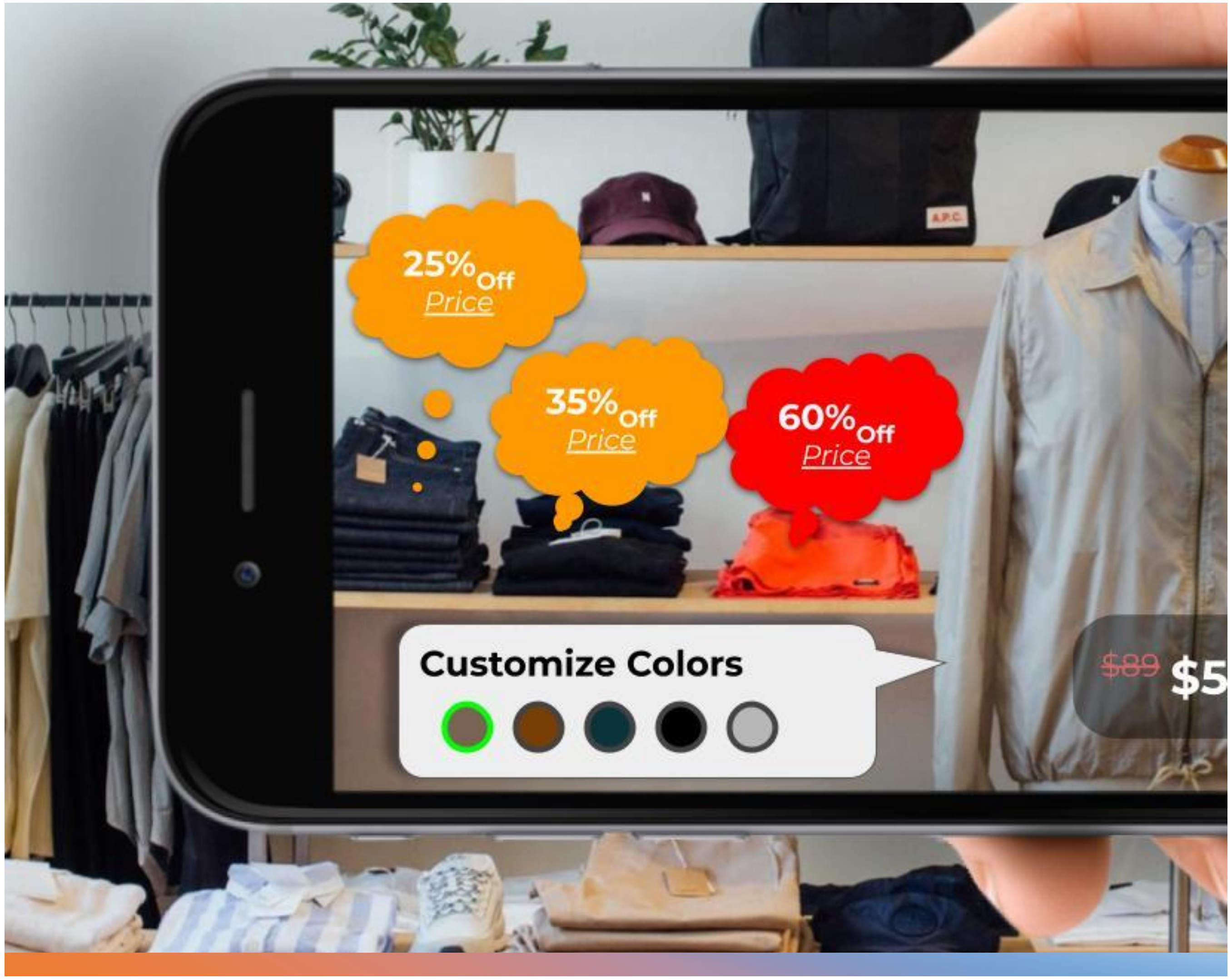
Tim Brooks et al. InstructPix2Pix. CVPR 2023

3D Scene Editing

- Application of Text2Image Diffusion Models
- Dataset Update with Edited Sample
- COLMAP Poses
- 3D Model Training (NeRF or Gaussain Splatting)

"Make it Autumn"

Haque, Ayaan, et al. "Instruct-nerf2nerf: Editing 3d scenes with instructions." *Proceedings of the IEEE/CVF ICCV*. 2023.



Significance of 3D Editing

- Film and Television
- Video Games
- Architecture and Interior Design
- Virtual Reality (VR) and Augmented Reality (AR)
- Fashion and Retail

3D Scene Editing Challenges

- High Training Cost
  - Large Training Time [24Hrs/scene- IN2N dataset]
  - Extensive Computational Resources
- Undesired/ Unrestricted Editing
  - Local Editing
  - Color Saturation
  - Single Object Editing
- Limited Applications:
  - Texture Editing mostly
  - Doesn't Support Geometric Editing

Introduction

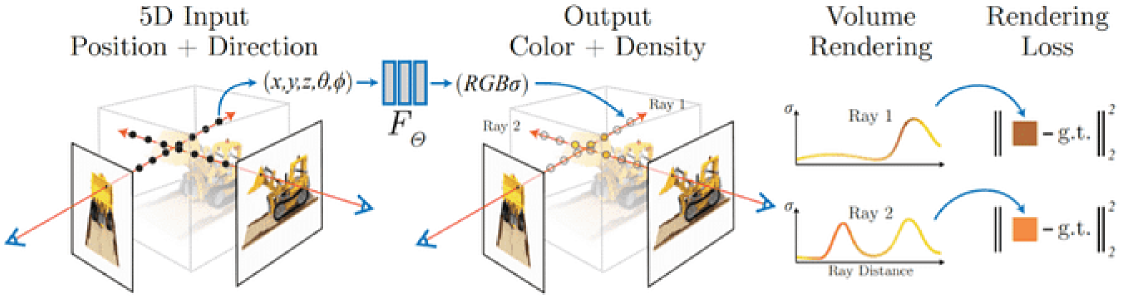
- Addressing limitations of InstructNeRF2NeRF
  - 3D Editing Efficiency
  - Quality of Editing
  - Multi-Attribute

Methods	Guidance			Editing Capacity			
	Pre-Computed Masks	Bounding Box	GAN Guidance	Text Driven	Style Transfer	Multi-Attribute Editing	Local Editing
Blend-NeRF [15]	✓	✗	✗	✗	✗	✗	✓
Blended-NeRF [6]	✗	✓	✗	✓	✗	✗	✓
DreamEditor [55]	✓	✗	✗	✓	✓	✗	✓
Control-4D [41]	✗	✗	✓	✓	✗	✗	✗
NeRF-Art [47]	✗	✗	✗	✓	✓	✗	✗
Instruct-N2N [8]	✗	✗	✗	✓	✓	✗	✗
LatentEditor (Ours)	✗	✗	✗	✓	✓	✓	✓

Tim Brooks et al. [Instruct-NeRF2NeRF](). ICCV 2023

NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis

- Predict the color and volume density for every viewing location and direction



- Generating a view from NeRF requires rendering all rays that pass through each pixel of the desired virtual camera

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t),\mathbf{d})dt\,, \text{ where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Expected color of a camera ray

Predicted Volume Density

Predicted Color

Probability that nothing has blocked the ray up to this point

- Training: Compute the squared error between rendered and true pixel colors

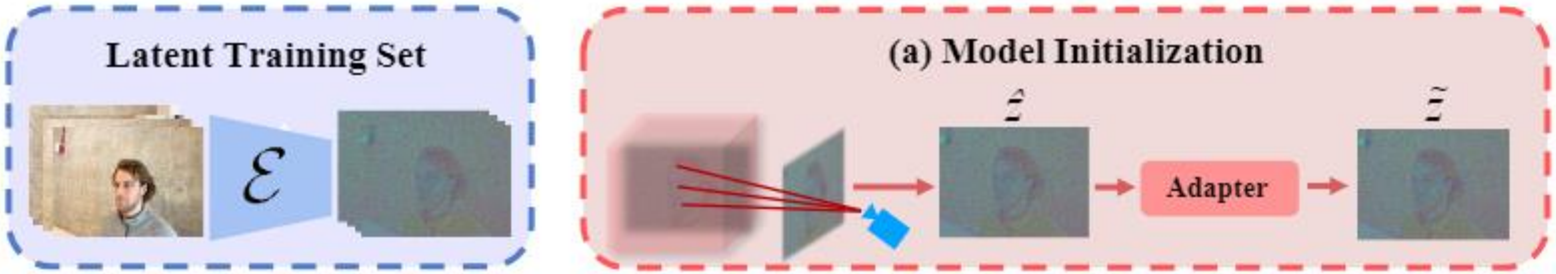
$$\mathcal{L} = \sum_{\mathbf{r}\in\mathcal{R}} \left[\left\| \hat{C}_c(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 + \left\| \hat{C}_f(\mathbf{r}) - C(\mathbf{r}) \right\|_2^2 \right]$$

LatentEditor Framework

RGB Images are converted into Latents using SD Encoder

$$z^n = \mathcal{E}(I^n) \in \mathbb{R}^{W' \times H' \times 4}$$

LatentEditor Framework



These latent feature maps Z serve as labels for LatentEditor NeRF initialization.

Volume Rendering: $$\hat{Z}(\mathbf{r}) = \int_{\tau_n}^{\tau_f} \mathbf{T}(\tau)\sigma(\mathbf{r}(\tau))\mathbf{z}(\mathbf{r}(\tau,\mathbf{d}))\,d\tau,$$

Reconstruction Loss: $$\mathcal{L}_r = \sum_{\mathbf{r}\in\mathcal{R}} \|\hat{Z}(\mathbf{r}) - Z(\mathbf{r})\|^2,$$

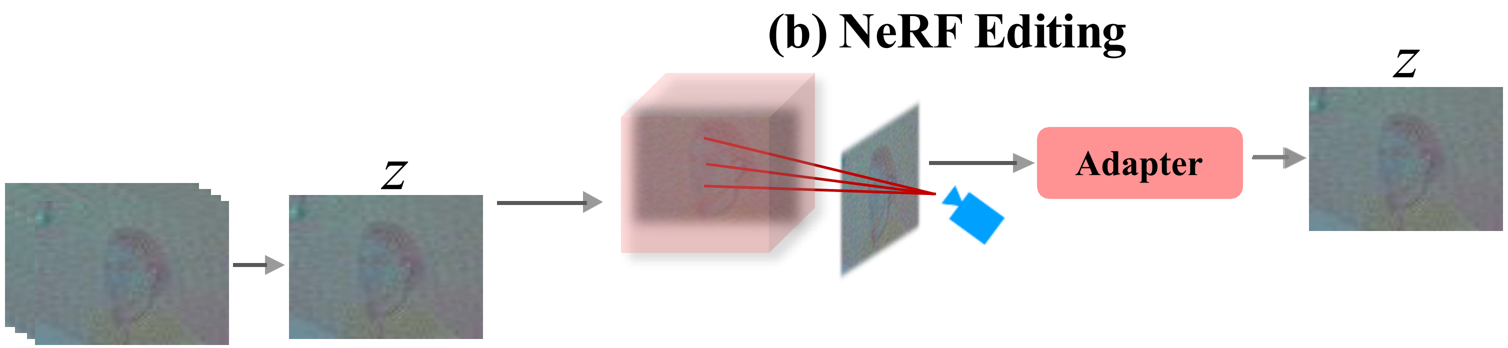
Refinement Loss: $$\mathcal{L}_f = \sum_{\mathbf{r}\in\mathcal{R}} \left\|\tilde{Z}^i(\mathbf{r}) - Z^i(\mathbf{r})\right\|^2$$

Refinement adapter mitigates the misalignment in the latent space and encompasses a trainable adapter with residual and self-attention mechanisms.

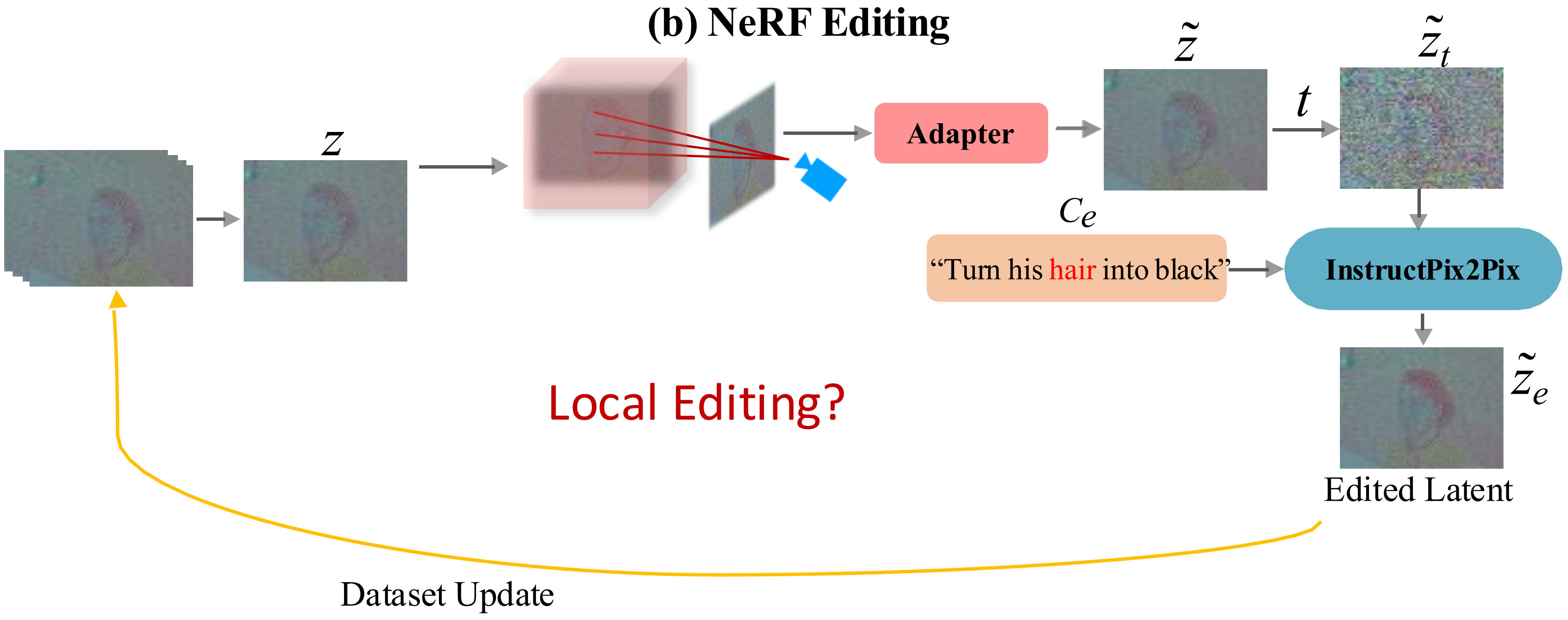
Training Loss: $$\mathcal{L}_T = \lambda_r\mathcal{L}_r + \lambda_f\mathcal{L}_f + \boxed{\lambda_p\mathcal{L}_{\text{reg}},}$$

Controls the camera parameters optimization.

LatentEditor Framework



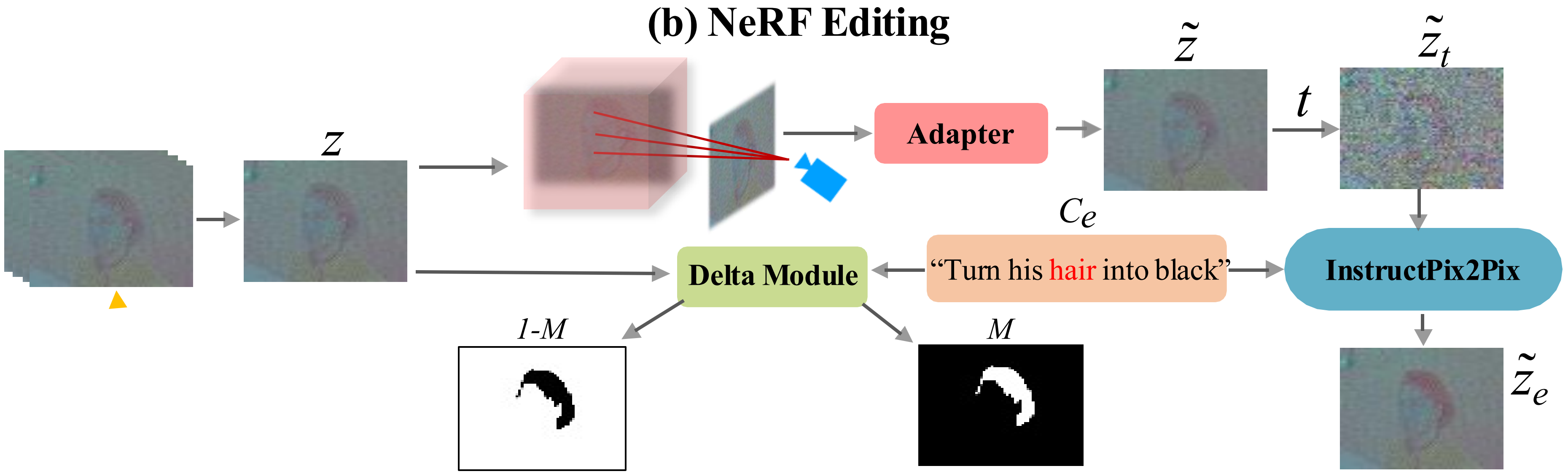
LatentEditor Framework



After initialization:

Consistently updates the training set with the edited latents, Ze.

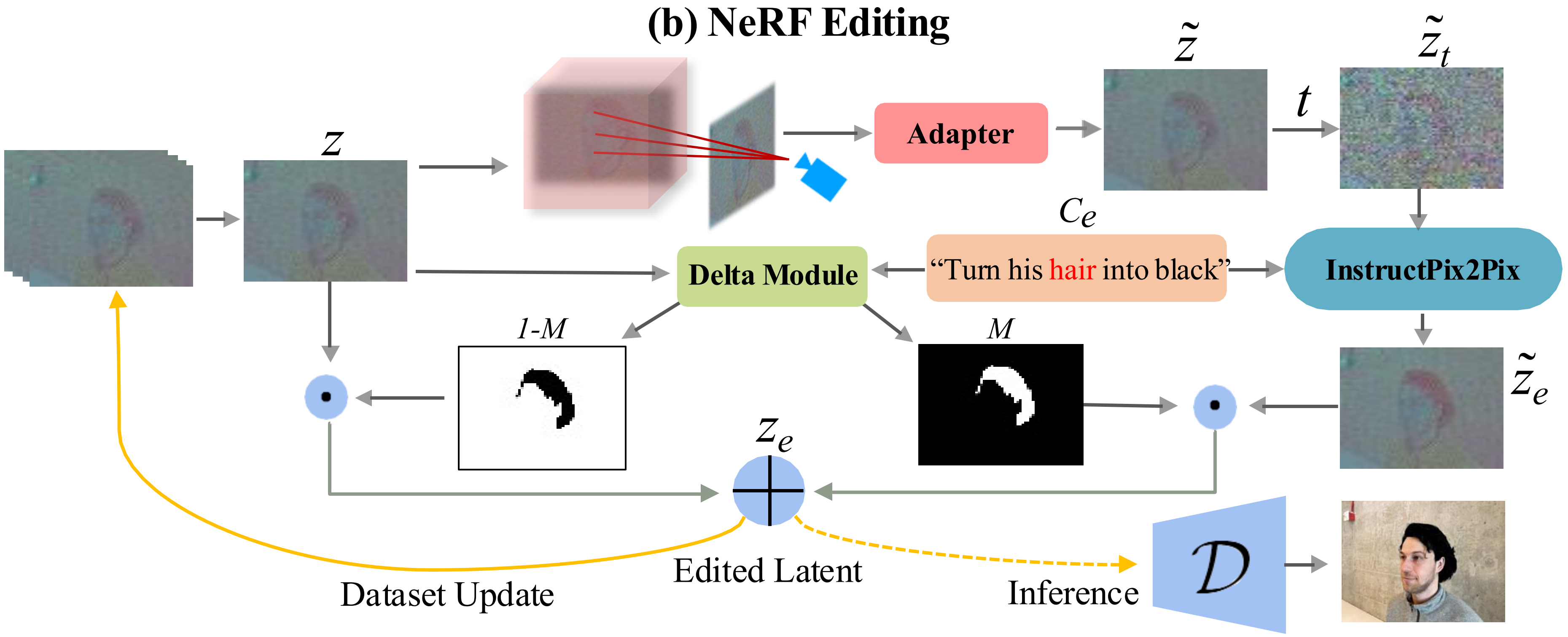
LatentEditor Framework



Prompt Aware Pixel Scoring:

- We design a delta module by modulating the IP2P diffusion process
- Aims to guide the editing by generating a mask

LatentEditor Framework



The final prediction IP2P prediction after complete denoising merges with the unedited latent using the mask M:

$$z_e^n = \tilde{z}_e^n \odot M + (1 - M) \odot z^n$$

LatentEditor Framework

Starting with noise addition to latent Z^n up to timestep Δt, we obtain the noisy latent $Z^n_{\Delta t}$

$$z^n_{\Delta t} = \sqrt{\beta_{\Delta t}} z^n + \sqrt{1 - \beta_{\Delta t}} \varepsilon,$$

IP2P's score estimation encompasses conditional and unconditional editing:

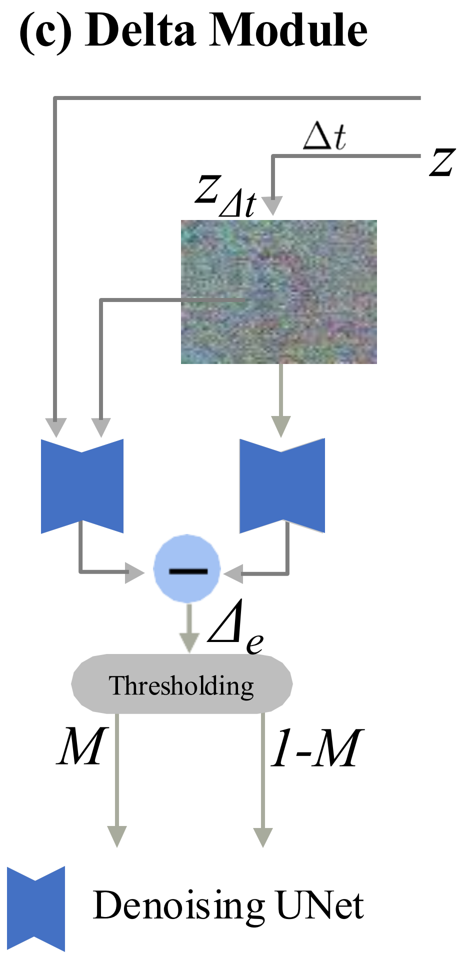
$$\begin{aligned} \tilde{\varepsilon}_\theta(z_t, I, C_e) &= \varepsilon_\theta(z_t, \varnothing_I, \varnothing_e) \\ &+ s_I \big(\varepsilon_\theta(z_t, I, \varnothing_e) - \varepsilon_\theta(z_t, \varnothing_I, \varnothing_e)\big) \\ &+ s_T \big(\varepsilon_\theta(z_t, I, C_e) - \varepsilon_\theta(z_t, I, \varnothing_e)\big). \end{aligned}$$

We calculate the delta scores using two noise predictions:

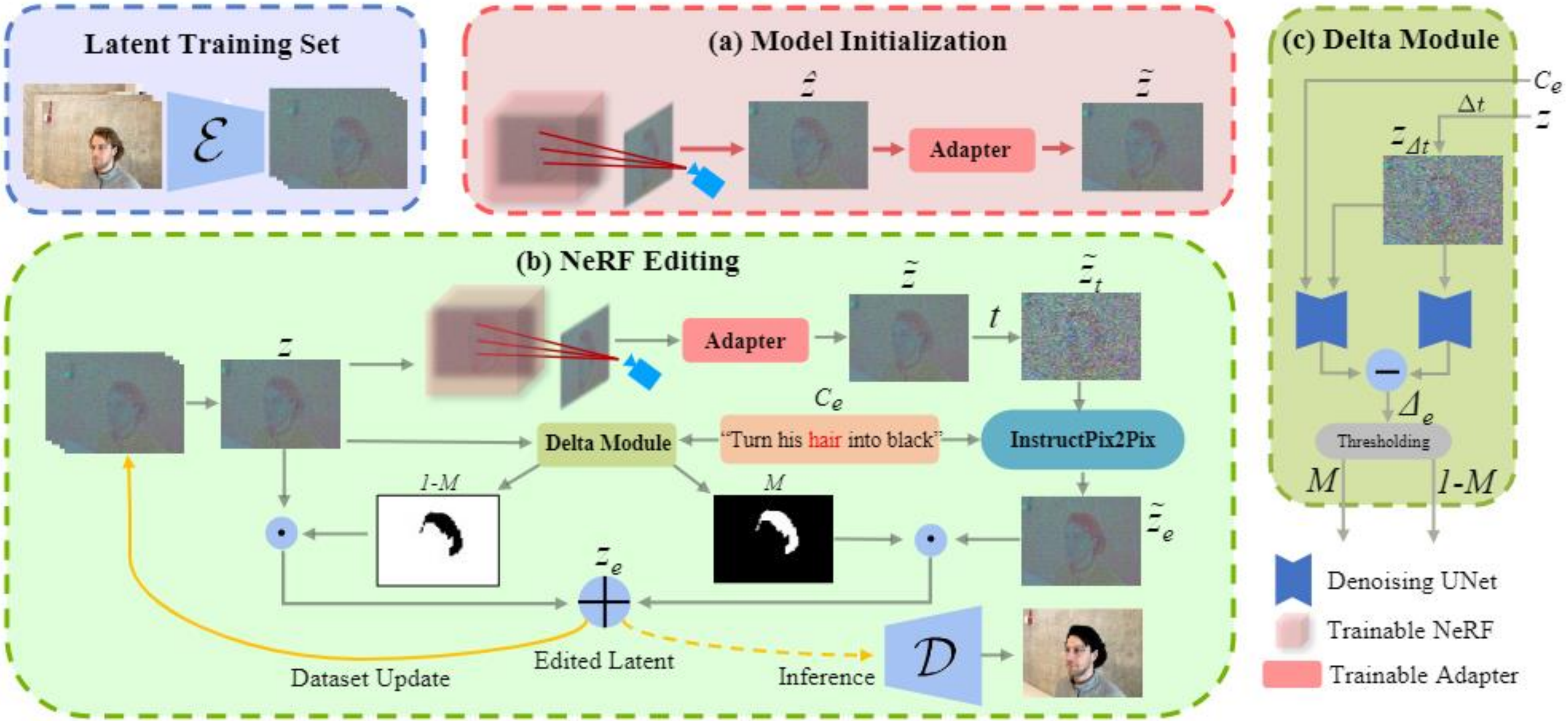
$$\Delta_\varepsilon = |\varepsilon_\theta(z^n_{\Delta t}, I, C_e) - \varepsilon_\theta(z^n_{\Delta t}, I, \varnothing_e)|$$

A binary mask, M can be generated by applying a threshold μ on Δ_ε as following:

$$M = \begin{cases} 1 & \text{if } \Delta_\varepsilon \geq \mu \\ 0 & \text{otherwise} \end{cases}$$



LatentEditor Framework



RGB image can be obtained by feeding the edited latent to the stable diffusion (SD) decoder D whereas E represents SD encoder.

Quantitative Results

Single Attribute Editing

Datasets:
1. IN2N
2. NeRF-Art
3. LLFF
4. NeRFstudio Dataset

Table 2: Quantitative evaluation of scene edits in terms of text alignment and frame consistency in CLIP space.

Method	CLIP Text-Image Direction Similarity↑	CLIP Direction Consistency↑	Edit PSNR↑
NeRF-Art [55]	0.2617	0.9188	21.04
Control4D [48]	0.2378	0.9263	19.85
DreamEditor [63]	0.2474	0.9312	20.67
IN2N [11]	0.2649	0.9358	24.07
Ours	**0.2661**	**0.9387**	**25.15**

Strong performance

Better Generalization

Quantitative Results

Multi-Attribute Editing

LLM Guided

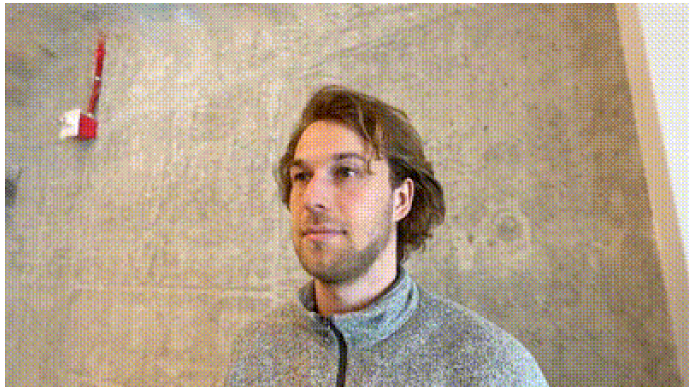
```
### Contexts
Break the following editing prompt into multiple parts with "
    and" as the key indicator of partition. Produce editing
    prompts based on the given input.
### Input
Make his hair red and give him blue jacket.
### Response
```

Table 2: Multi-attribute Editing. Quantitative evaluation of scene edits in terms of text alignment and frame consistency in CLIP space where our approach demonstrates the highest consistency.

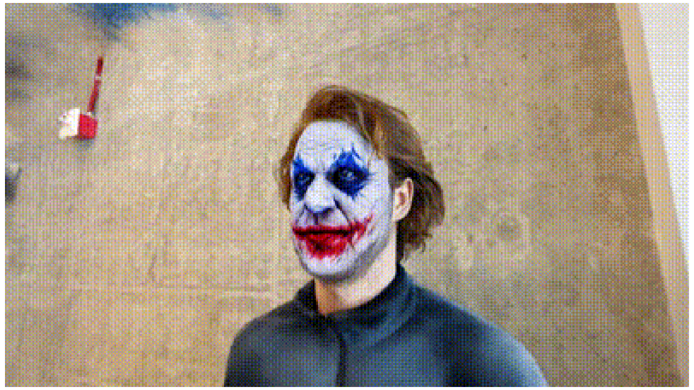
Method	CLIP Text-Image Direction Similarity↑	CLIP Direction Consistency↑	Edit PSNR↑
NeRF-Art [55]	0.2007	0.8564	16.54
Control4D [48]	0.1945	0.8664	14.45
DreamEditor [63]	0.2134	0.8702	18.33
IN2N [11]	0.2257	0.8846	20.07
Collaborative Score Distillation [20]	0.2134	0.8755	19.46
ViCA-NeRF [6]	0.2112	0.8636	18.76
Ours	**0.2611**	**0.9347**	**24.45**

No Method in the literature handles multi-attribute editing effectively.

LatentEditor Qualitative Results

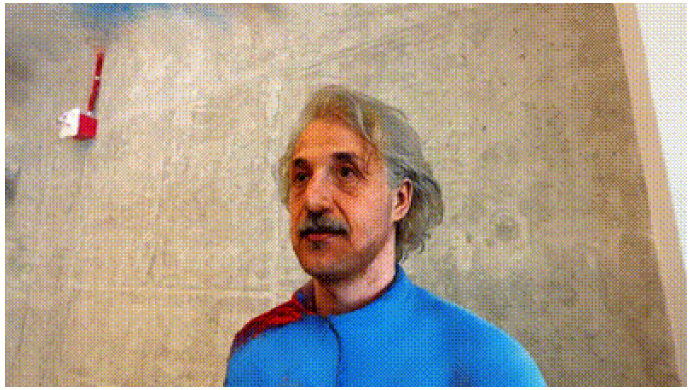


Original Scene



Turn his face into Joker and give his jacket batman suit touch

Give Him Moustache and Yellow Jacket



Turn him into Einstein and Give him Superman Shirt

LatentEditor Qualitative Results

Original Scene

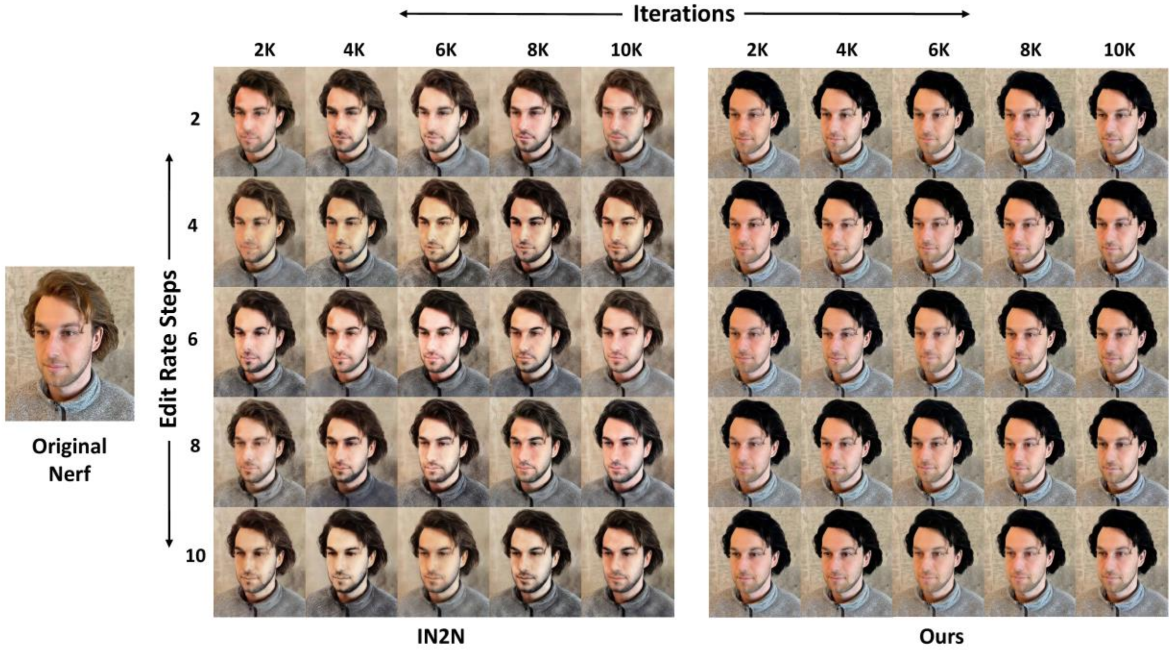
Turn Bulldozer into red

Turn cones into purple

Turn tyres into blue

Ablations

- Editing rate vs training iteration #
- Editing rate:
    - Dataset update frequency
- Lesser training iteration in comparison to RGB



"Turn his head hair into black"

Efficiency Analysis

- Model processes 64 times fewer rays than its RGB counterpart
  - Reduce the number of iterations by a factor of 64
- Nerfacto model
- End-to-End:
  - LatentEditor is **5-8** times faster than IN2N

Matrices	IN2N	LatentEditor
NeRF Feature Size	512×320	64×40
Model Initialization	<400s	>745s
GPU Consumption	1766MB	6778MB

"fangzhou-small(IN2N scene)"

Thanks