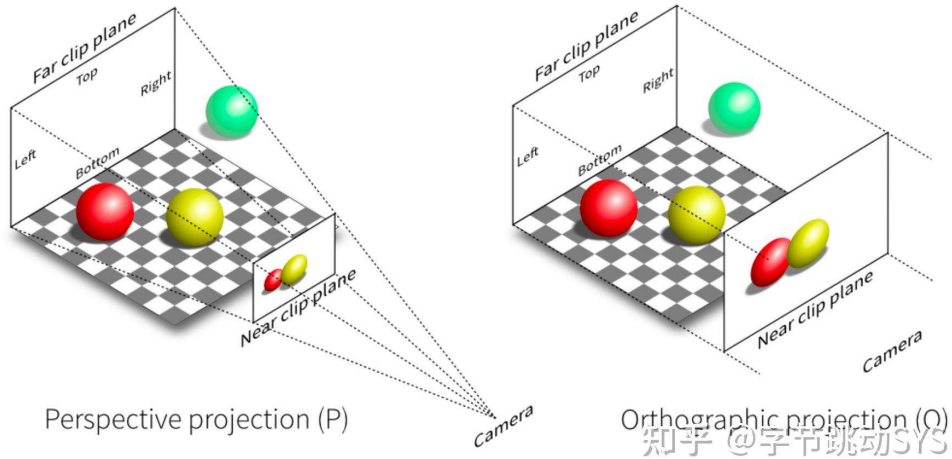
GaussianEditor: Editing 3D Gaussians Delicately with Text Instructions

Jiemin Fang[*] Junjie Wang[*] Xiaopeng Zhang, Lingxi Xie, Qi Tian
Huawei Inc.
{jaminfong, is.wangjunjie, 198808xc, zxphistory}@gmail.com tian.qi1@huawei.com
GaussianEditor.github.io

Gaussian Splatting

3D Gaussian Splatting for Real-Time Radiance Field Rendering

BERNHARD KERBL*, Inria, Université Côte d'Azur, France
GEORGIOS KOPANAS*, Inria, Université Côte d'Azur, France
THOMAS LEIMKÜHLER, Max-Planck-Institut für Informatik, Germany
GEORGE DRETTAKIS, Inria, Université Côte d'Azur, France



$$\Theta_i = \{x_i, s_i, q_i, \alpha_i, c_i\}$$

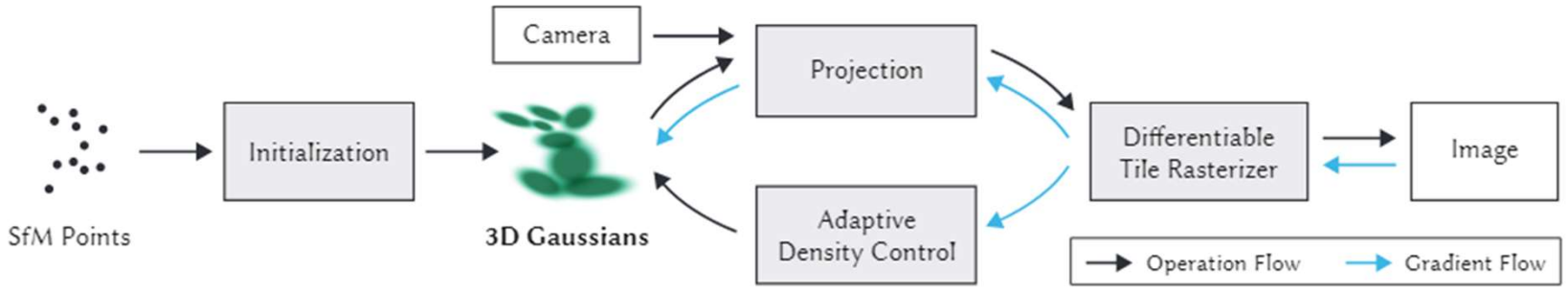


Fig. 2. Optimization starts with the sparse SfM point cloud and creates a set of 3D Gaussians. We then optimize and adaptively control the density of this set of Gaussians. During optimization we use our fast tile-based renderer, allowing competitive training times compared to SOTA fast radiance field methods. Once trained, our renderer allows real-time navigation for a wide variety of scenes.

Gaussian Editor

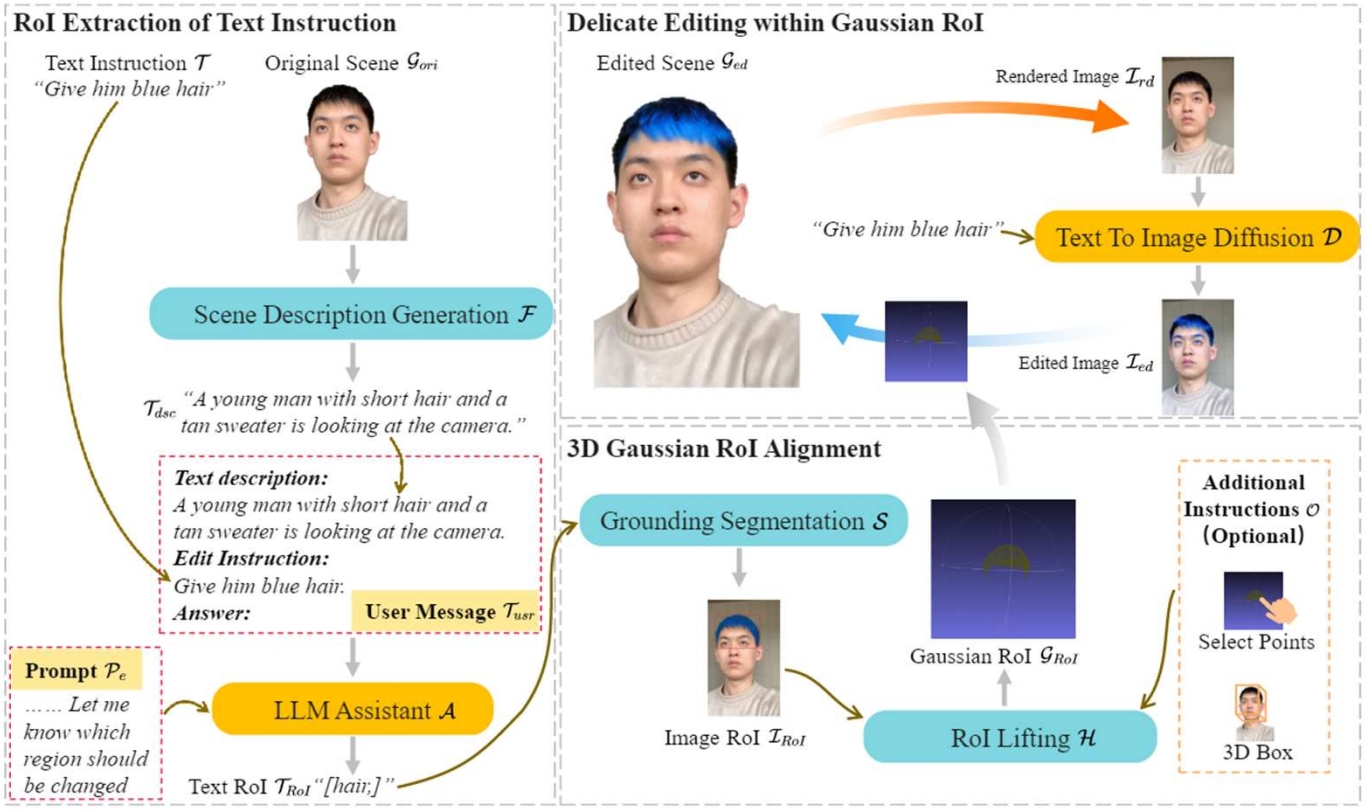
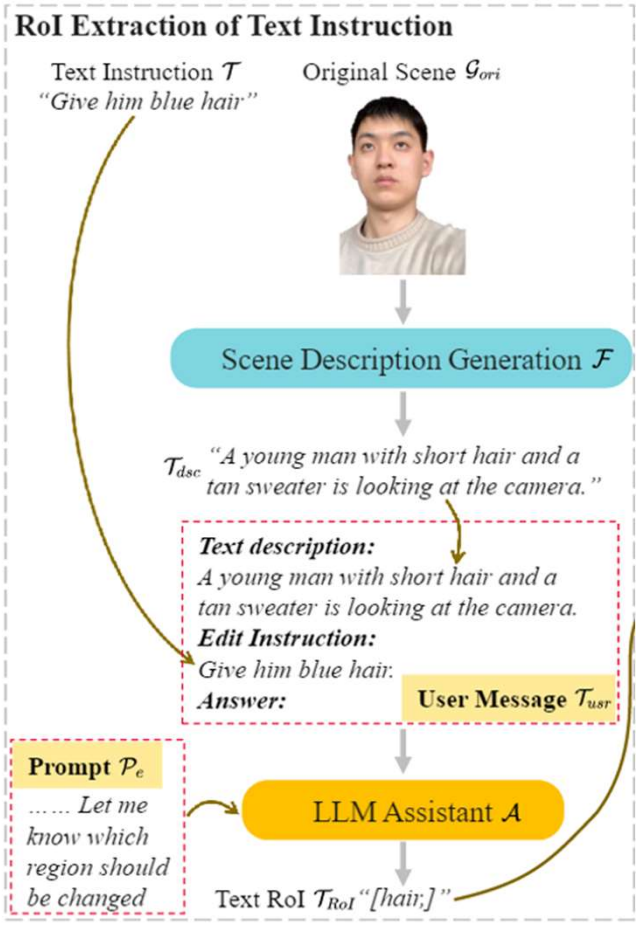


Figure 2. Our framework, named GaussianEditor, consists of three key steps. First, the scene generation $\mathcal{F}$ was performed to get the scene description $\mathcal{T}_{dsc}$ from original 3D Gaussians $\mathcal{G}_{ori}$. The description $\mathcal{T}_{dsc}$ and the text instruction $\mathcal{T}$ provided by the user are then combined using a template $\mathcal{T}_{tmp}$ to get user message $\mathcal{T}_{usr}$. The $\mathcal{T}_{usr}$ and a pre-defined prompt $\mathcal{P}_e$ are fed into an LLM assistant $\mathcal{A}$ to obtain the instruction RoI $\mathcal{T}_{RoI}$. Second, a grounding segmentation model $\mathcal{S}$ is used to convert $\mathcal{T}_{RoI}$ to image RoI $\mathcal{I}_{RoI}$, which is then lifted to 3D Gaussians RoI $\mathcal{G}_{RoI}$ by RoI lifting $\mathcal{H}$, where additional user instructions $\mathcal{O}$ can be incorporated. Third, following the user instruction $\mathcal{T}$, rendered image $\mathcal{I}_{rd}$ from randomly chosen views is edited by a 2D diffusion model $\mathcal{D}$. The loss between $\mathcal{I}_{rd}$ and edited one $\mathcal{I}_{ed}$ is calculated. Finally, gradient backpropagation and optimization are performed within the Gaussian RoI $\mathcal{G}_{RoI}$ to get the edited scene $\mathcal{G}_{ed}$.

RoI Extraction of Text



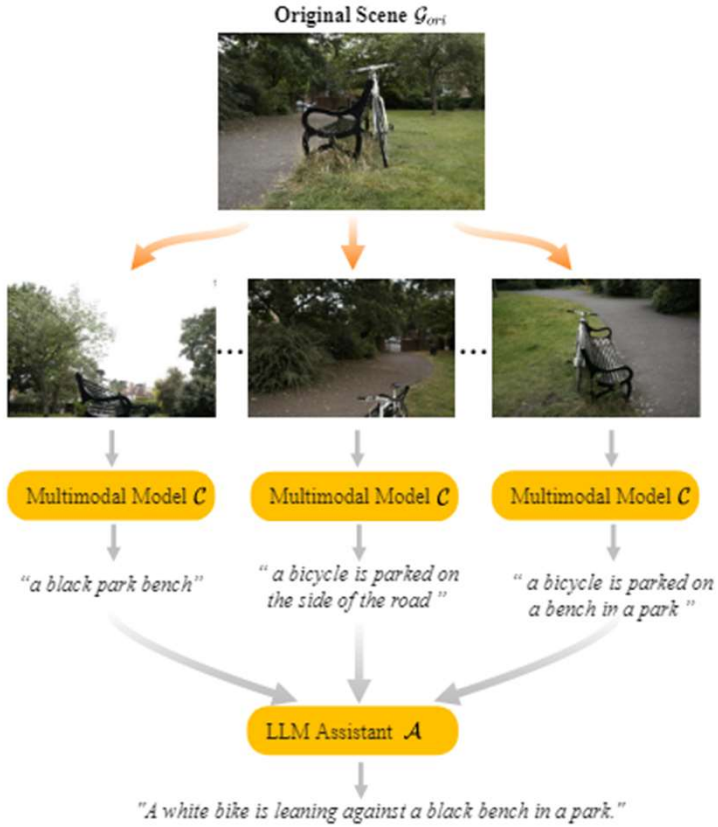
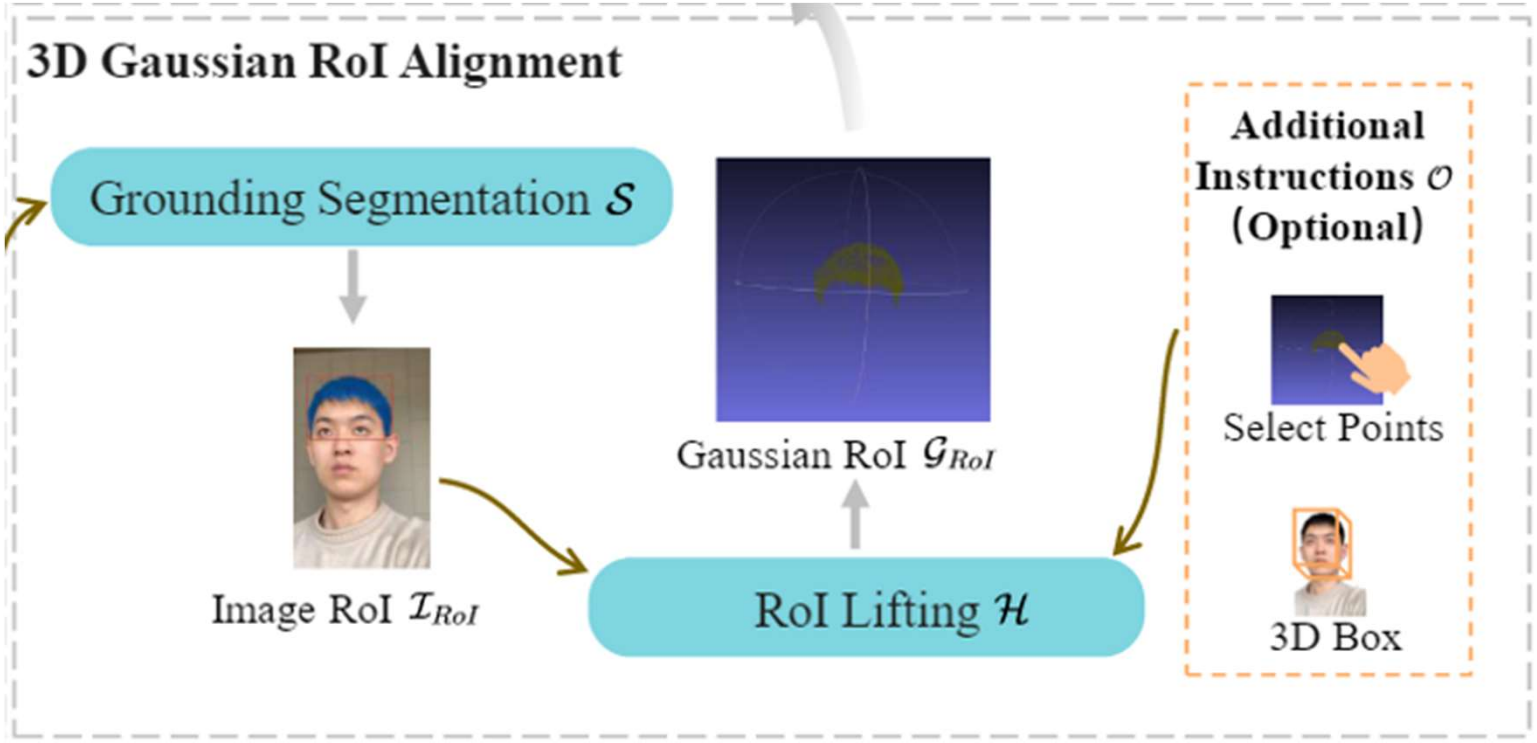


Figure 3. The process of obtaining scene description.

the regions specified by the RoI can receive corresponding gradients during the back-propagation process. Finally, optimization is executed based on these gradients. Through several rounds of iterative optimization, the final optimized scene representation $\mathcal{G}_{ed}$ is obtained.

3D Gaussian RoI Alignment

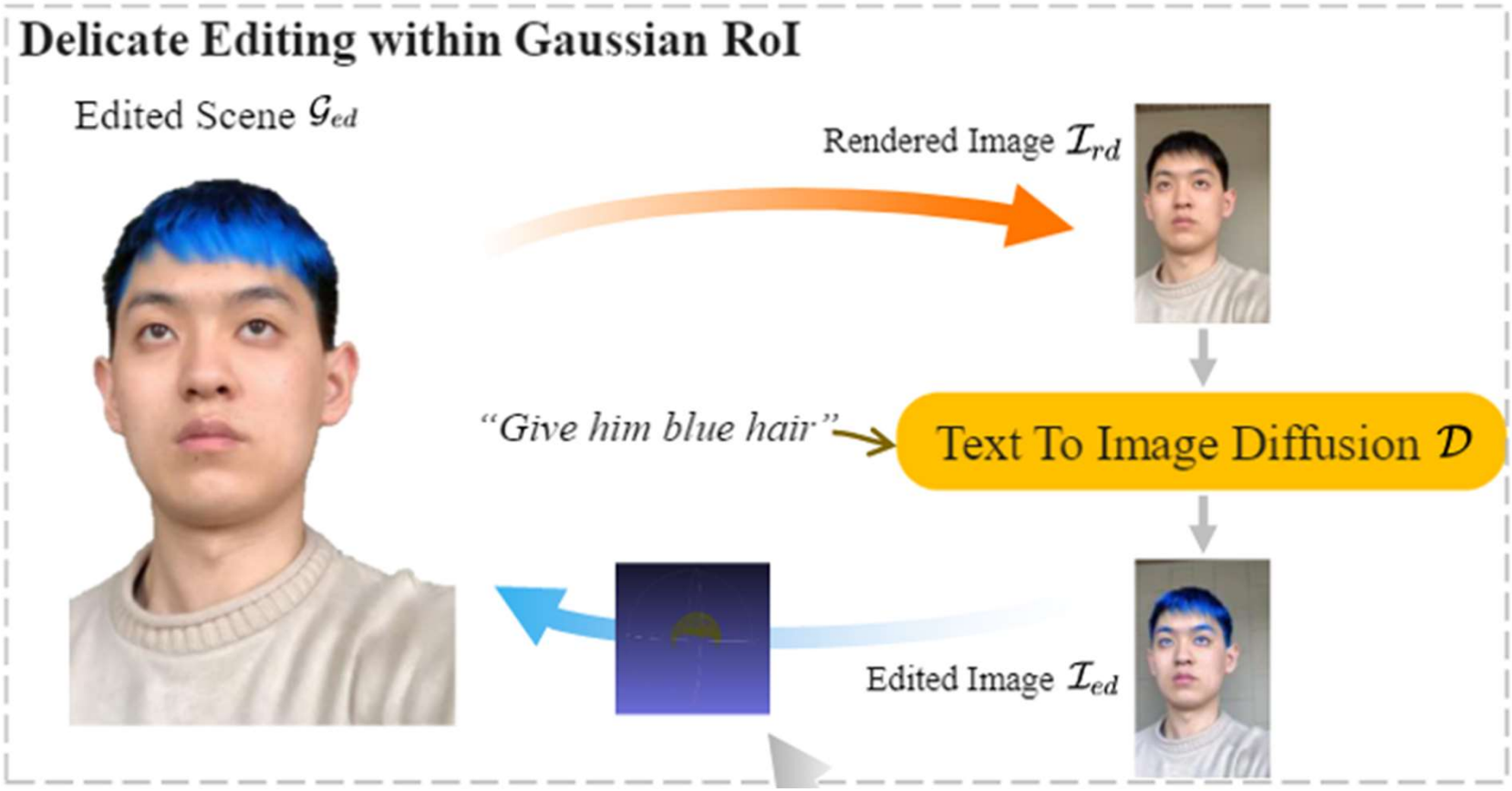


$$\mathcal{I}_{RoI}^{rd} = \sum_{i \in \mathcal{N}} r_i \sigma_i \prod_{j=1}^{i-1} (1 - \alpha_j).$$

$$\Theta_i = \{x_i, s_i, q_i, \alpha_i, c_i\}$$

$$\mathcal{L}_{proj} = \lambda_1 \sum (\mathcal{I}_{RoI}^{rd} \cdot \mathcal{I}_{RoI}) + \lambda_2 \sum ((1 - \mathcal{I}_{RoI}^{rd}) \cdot \mathcal{I}_{RoI}),$$

Delicate Editing within Gaussian RoI



Quality Evaluation

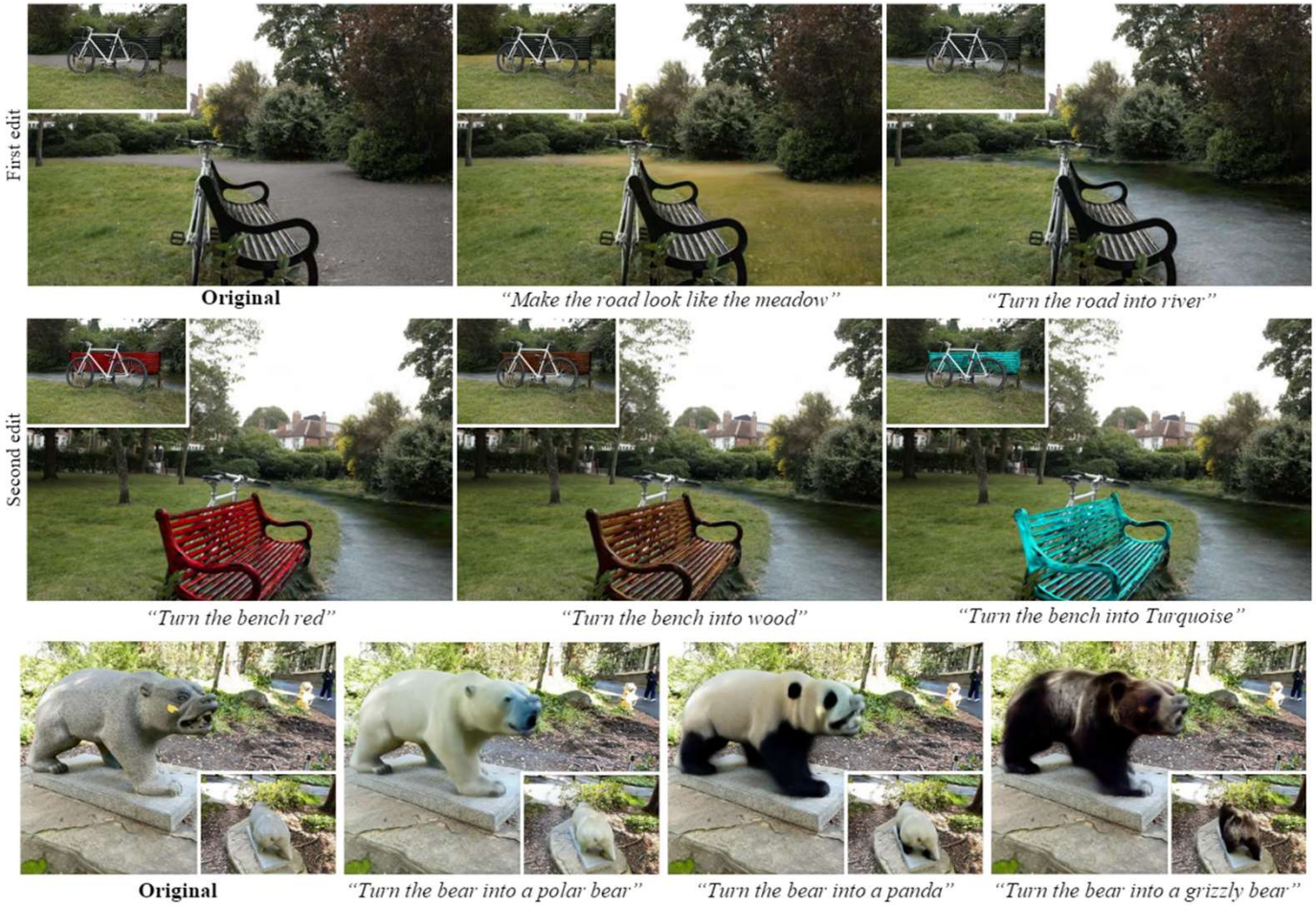


Figure 4. Qualitative results on outdoor scenes. Our method supports separate foreground and background editing in real-world scenes.

Gaussian Splatting

Instruct-NeRF2NeRF: Editing 3D Scenes with Instructions

Ayaan Haque, Matthew Tancik, Alexei A. Efros, Aleksander Holynski, and Angjoo Kanazawa

UC Berkeley

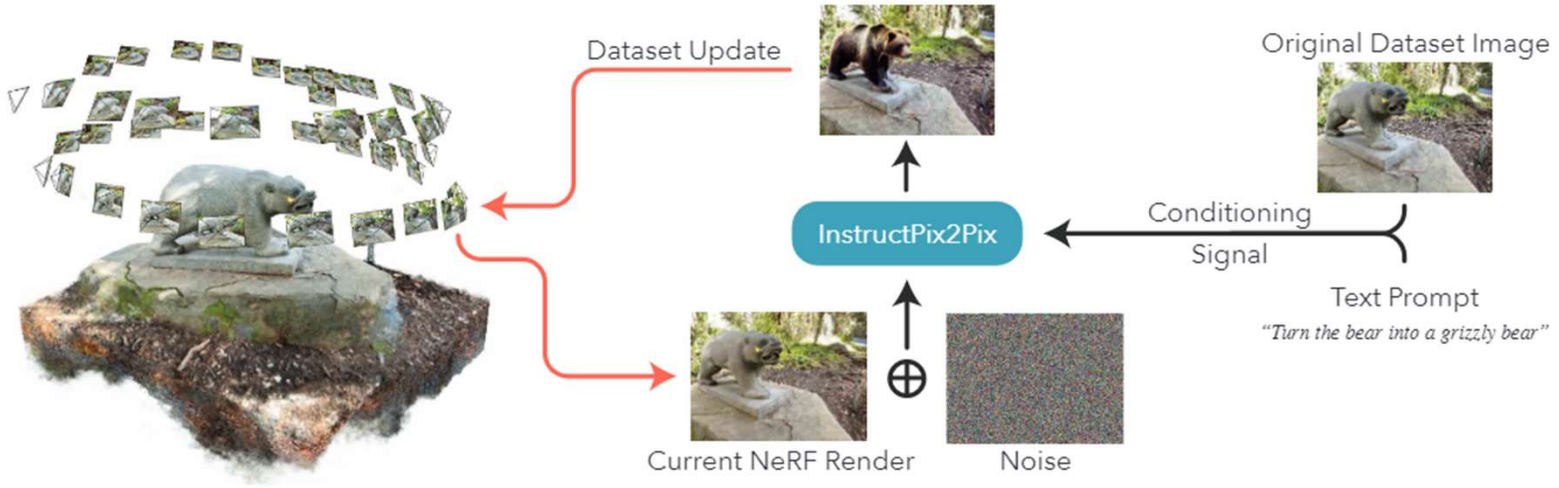


Figure 2: **Overview**: Our method gradually updates a reconstructed NeRF scene by iteratively updating the dataset images while training the NeRF: (1) an image is rendered from the scene at a training viewpoint, (2) it is edited by InstructPix2Pix given a global text instruction, (3) the training dataset image is replaced with the edited image, and (4) the NeRF continues training as usual.

Quality Evaluation

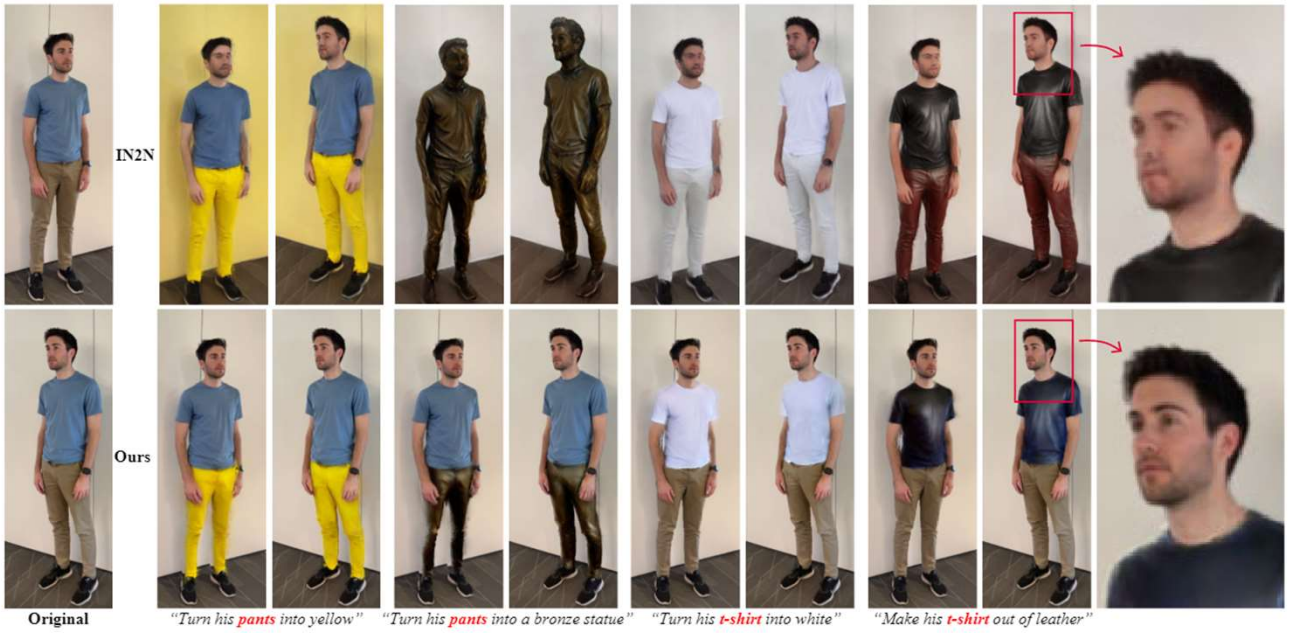


Figure 5. Comparisons with Instruct-NeRF2NeRF (IN2N) [11]. We use IN2N as a baseline and compare our method with the scenes presented in their paper.

Figure 6. Qualitative results on complex multi-object scenes. The background "`desk`", the foreground "`flower pot`", and the multi-view blocked foreground "`rolling pin`" are edited separately.

View-Consistent 3D Editing with Gaussian Splatting

Yuxuan Wang[1], Xuanyu Yi[1], Zike Wu[1], Na Zhao[2], Long Chen[3], and Hanwang Zhang[1,4]

[1] Nanyang Technological University
[2] Singapore University of Technology and Design
[3] Hong Kong University of Science and Technology
[4] Skywork AI

VC 3d Editing

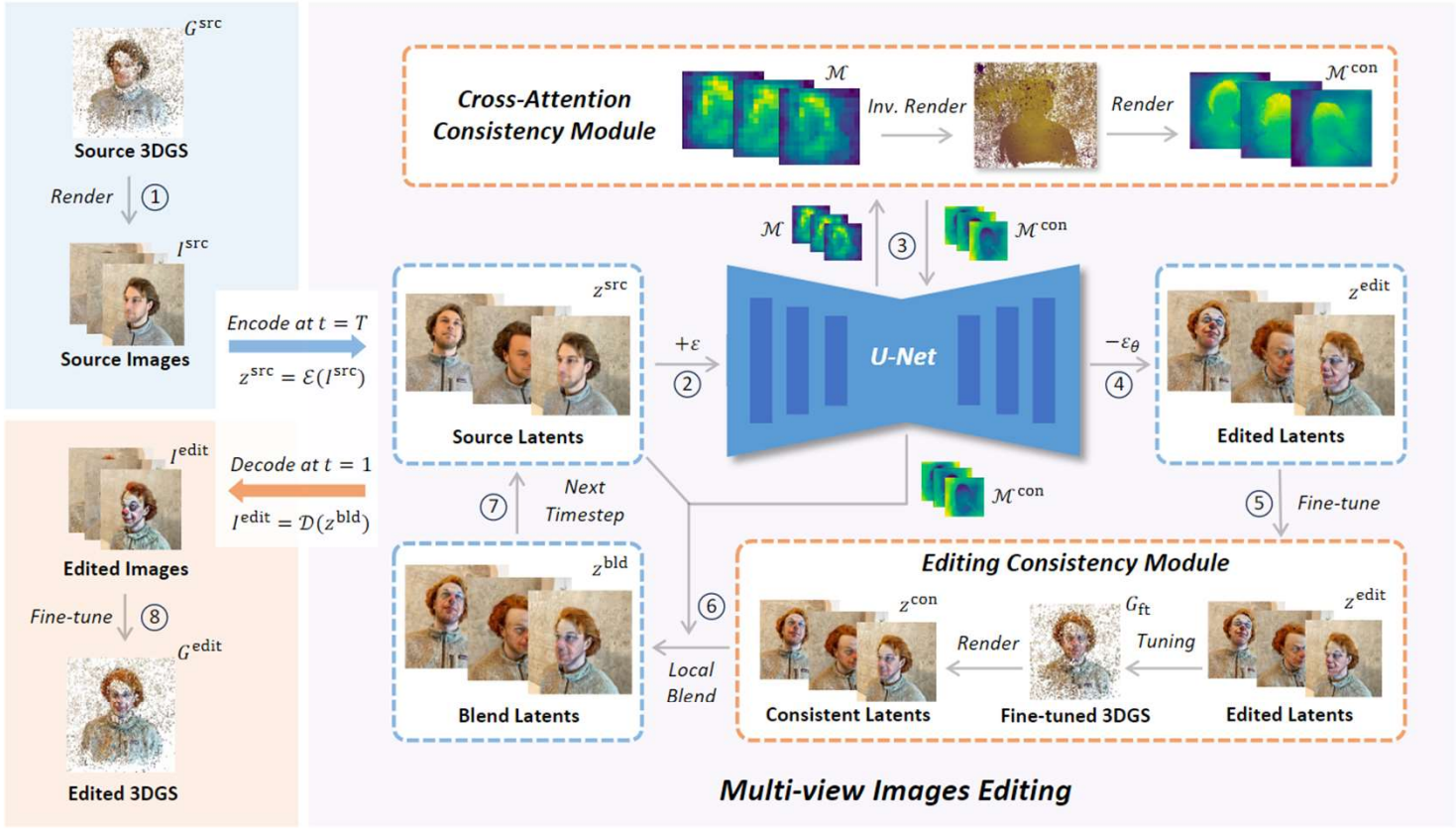


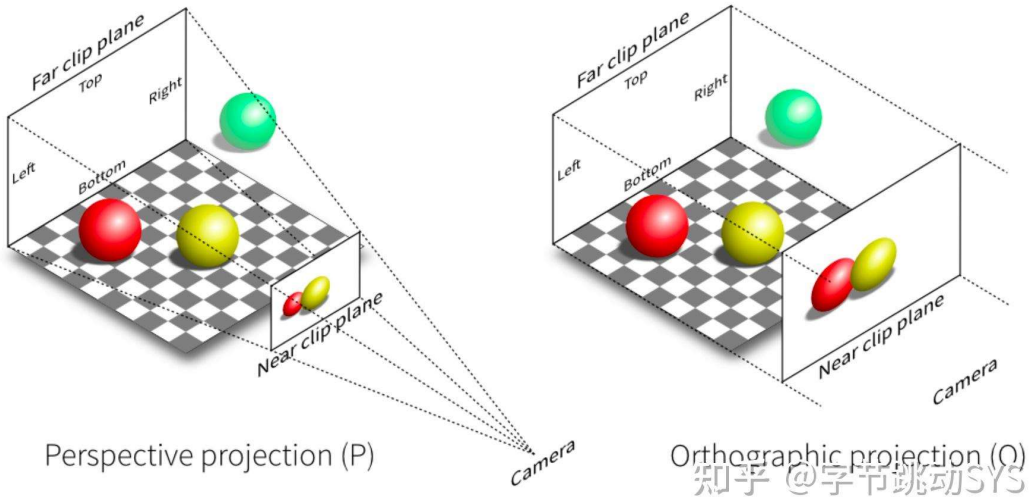
Fig. 3: The pipeline of our VcEdit: VcEdit employs an image-guided editing pipeline. In the image editing stage, the Cross-attention Consistency Module and Editing Consistency Module are employed to ensure the multi-view consistency of edited images. We provide a detailed overview in Sec. 4.1.

VC 3d Editing

GaussianEditor: Swift and Controllable 3D Editing with Gaussian Splatting

Yiwen Chen[*1,2] Zilong Chen[*3,5] Chi Zhang[2] Feng Wang[3] Xiaofeng Yang[2]
Yikai Wang[3] Zhongang Cai[4] Lei Yang[4] Huaping Liu[3] Guosheng Lin[**1,2]

[1]S-Lab, Nanyang Technological University
[2]School of Computer Science and Engineering, Nanyang Technological University
[3]Department of Computer Science and Technology, Tsinghua University
[4]SenseTime Research [5]ShengShu



$$\Theta_i = \{x_i, s_i, q_i, \alpha_i, c_i\}$$

$$w_i^j = \sum o_i(\boldsymbol{p}) * T_i^j(\boldsymbol{p}) * \mathcal{M}^j(\boldsymbol{p}),$$

New attribute m, where mij represents the semantic Gaussian mask for the i-th Gaussian point and the j-th semantic label

VC 3d Editing

Fig. 4: Qualitative comparison with the DDS [16] and the GSEditor [6]: The *topmost* row demonstrate the original views, while the *bottom* rows show the rendering view of edited 3DGS. VcEdit excels by effectively addressing the multi-view inconsistency, resulting in superior editing quality. In contrast, other methods encounter challenges with mode collapse and exhibit flickering artifacts.