3DEgo: 3D Editing on the Go!

Umar Khalid[1]*, Hasan Iqbal[2]*, Azib Farooq , Jing Hua[2], Chen Chen[1]
[1]University of Central Florida, [2]Wayne State University
*Equal Contribution

https://3dego.github.io/

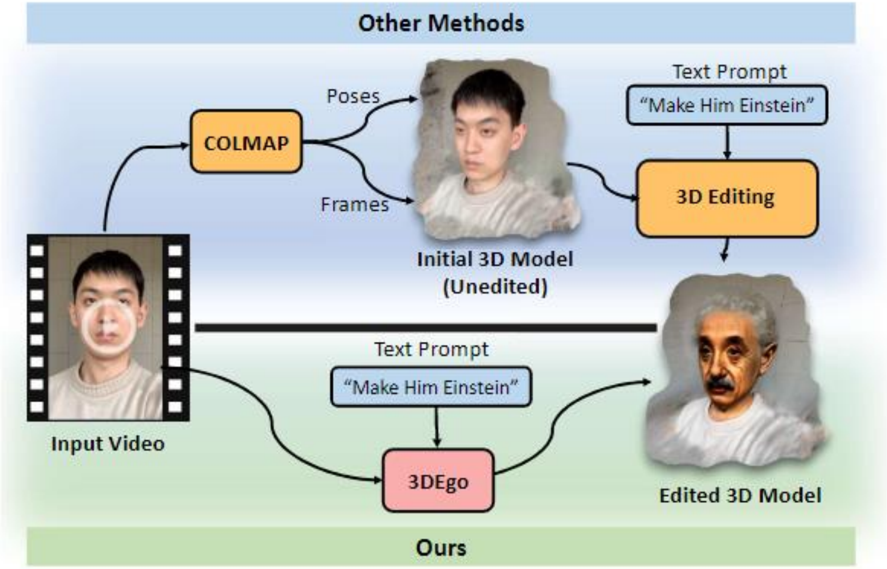
European Conference on Computer Vision (ECCV) 2024
Milan, Italy

Thu 3 Oct 4:30 a.m. EDT -6:30 a.m. EDT, Poster # 60

Introduction

- Merge a three-stage workflow into a singular, comprehensive framework.

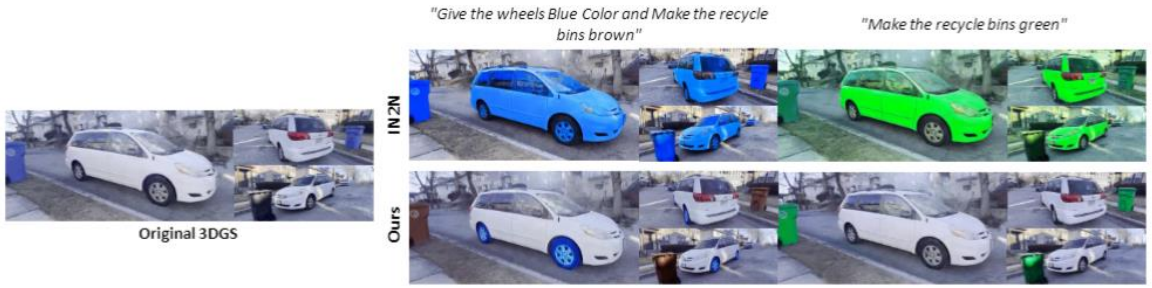
- This efficiency is achieved by:
  - Bypassing the need for COLMAP
  - Avoiding model initialization



- Integrates Segment Anything Model with LLM to achieve local editing

Contributions

- Text-conditioned Pose-Free 3D synthesis
  - Gaussian Splatting trained on a casually recorded video
- Autoregressive Editing:
  - Preserving consistency across multiple views
  - Conditioned on already edited adjacent frames
  - Mask Generation using LLM and SAM models
- GS25 Datastet
  - 25 casually captured monocular videos


"Give the wheels Blue Color and Make the recycle bins brown"
"Make the recycle bins green"
IN2N
Ours
Original 3DGS

Van Scene

Consistent Multi-View 2D Editing

- Mask Generation using LLM and SAM models
- Per-Frame Editing using IP2P
- Autoregressive Editing:
  - Editing Conditioned on already edited adjacent frames

Single Frame IP2P

$$\tilde{\varepsilon}_\theta(e_t, f, \mathcal{T}) = \varepsilon_\theta(e_t, \varnothing_f, \varnothing_\mathcal{T}) + s_f\big(\varepsilon_\theta(e_t, f, \varnothing_\mathcal{T}) - \varepsilon_\theta(e_t, \varnothing_f, \varnothing_\mathcal{T})\big) + s_\mathcal{T}\big(\varepsilon_\theta(e_t, f, \mathcal{T}) - \varepsilon_\theta(e_t, f, \varnothing_\mathcal{T})\big)$$

- Image-conditional noise estimation, ε_θ (e_t, E, $Ø_T$) across all frames in W:

$$\bar{\varepsilon}_\theta(e_t, \varnothing_\mathcal{T}, W) = \sum_{n=1}^{w} \beta_n \varepsilon_\theta^n(e_t, E_n, \varnothing_\mathcal{T})$$

Autoregressive IP2P

$$\varepsilon_\theta(e_t, f, \mathcal{T}, W) = \gamma_f \tilde{\varepsilon}_\theta(e_t, f, \mathcal{T}) + \gamma_E \bar{\varepsilon}_\theta(e_t, \varnothing_\mathcal{T}, W)$$

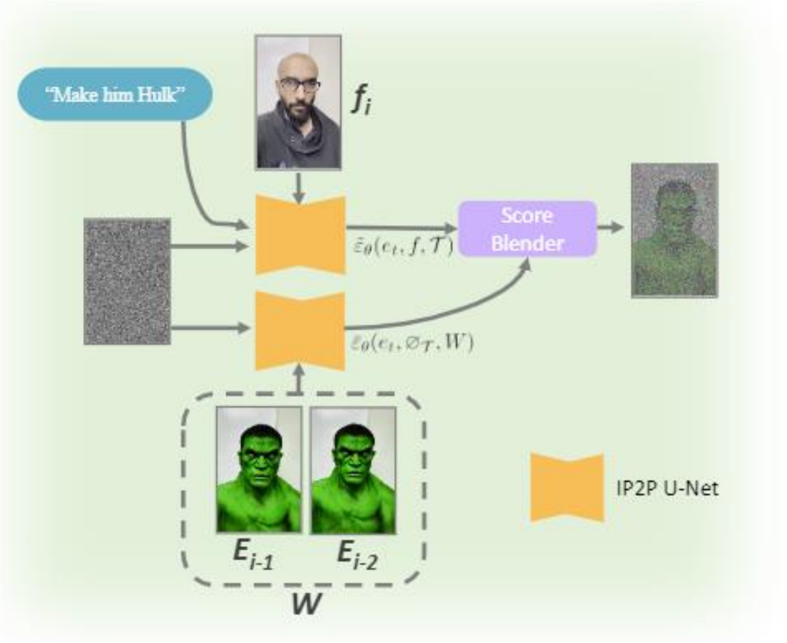
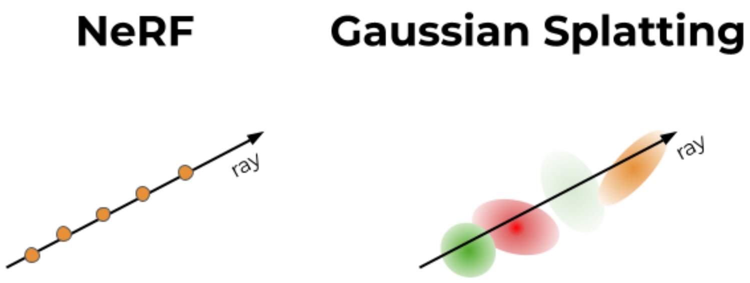

"Make him Hulk"
f_i
$\varepsilon_\theta(e_t, f, \mathcal{T})$
Score Blender
$\bar{\varepsilon}_\theta(e_t, \varnothing_\mathcal{T}, W)$
E_{i-1}
E_{i-2}
W
IP2P U-Net

Fig. 3: Autoregressive Editing. At each denoising step, the model predicts $w + 1$ separate noises, which are then unified via weighted noise blender (Eq. 4) to predict $\varepsilon_\theta(e_t, f, \mathcal{T}, W)$.

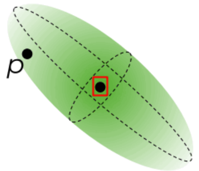
Recap - 3D Gaussian Splatting: Representing Scenes as Gaussians

- Each 3D Gaussian is parametrized by:
  - Mean **μ** interpretable as location x, y, z;
  - Covariance **Σ**;
  - Opacity **σ(α)**,
  - **Color parameters**



- The impact of a 3D Gaussian *i* on an arbitrary 3D point *p* in 3D is defined as:

$$f_i(p) = \sigma(\alpha_i)\exp(-\frac{1}{2}(p-\boxed{\mu_i})\Sigma_i^{-1}(p-\boxed{\mu_i}))$$

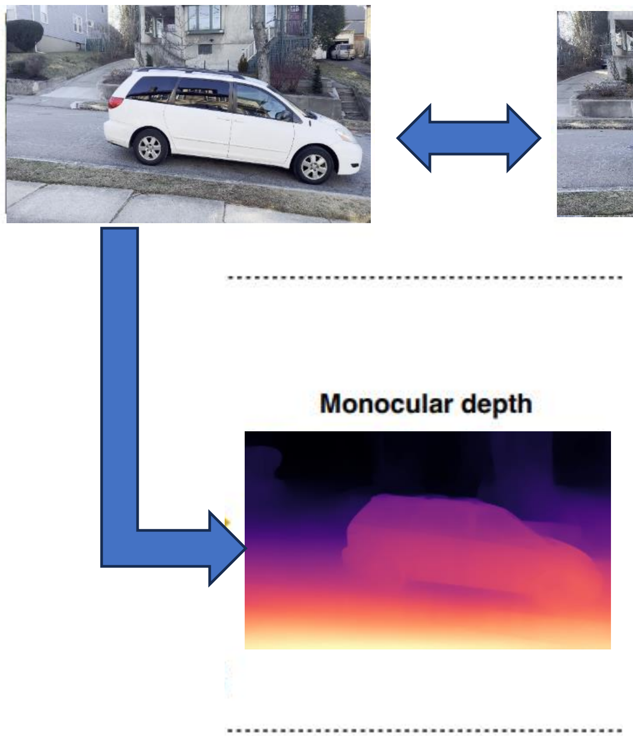


- The image formation model of Gaussian splatting:

$$C(p) = \sum_{i\in N} c_i f_i^{2D}(p) \underbrace{\prod_{j=1}^{i-1}(1-f_j^{2D}(p))}_{transmittance}$$

Kerbl, Bernhard, et al. "3D Gaussian Splatting for Real-Time Radiance Field Rendering." *ACM Trans. Graph.* 42.4 (2023): 139-1.

Training Mechanism



- Parameterizing Gaussians:

  Gaussian point, $h = \{\mu, \Sigma, c, \alpha, m\}$
- A pre-trained Depth Estimator to initialize point clouds
- 3D point cloud based Gaussians initialization

Training Mechanism

- Relative Pose Estimation:

$$\mathcal{H}_i^* = \arg \min_{c,\Sigma,\mu,\alpha} \mathcal{L}_{rgb}(\mathcal{R}(\mathcal{H}_i), E_i) + \arg \min_{m} \mathcal{L}_{KEA}(\mathcal{R}(\mathcal{H}_i), M_i),$$

- A learnable SE-3 affine transformation:

$$\mathcal{M}_i{}^* = \arg \min_{\mathcal{M}_i} \mathcal{L}_{rgb}(\mathcal{R}(\mathcal{M}_i \odot \mathcal{H}_i), E_{i+1}),$$

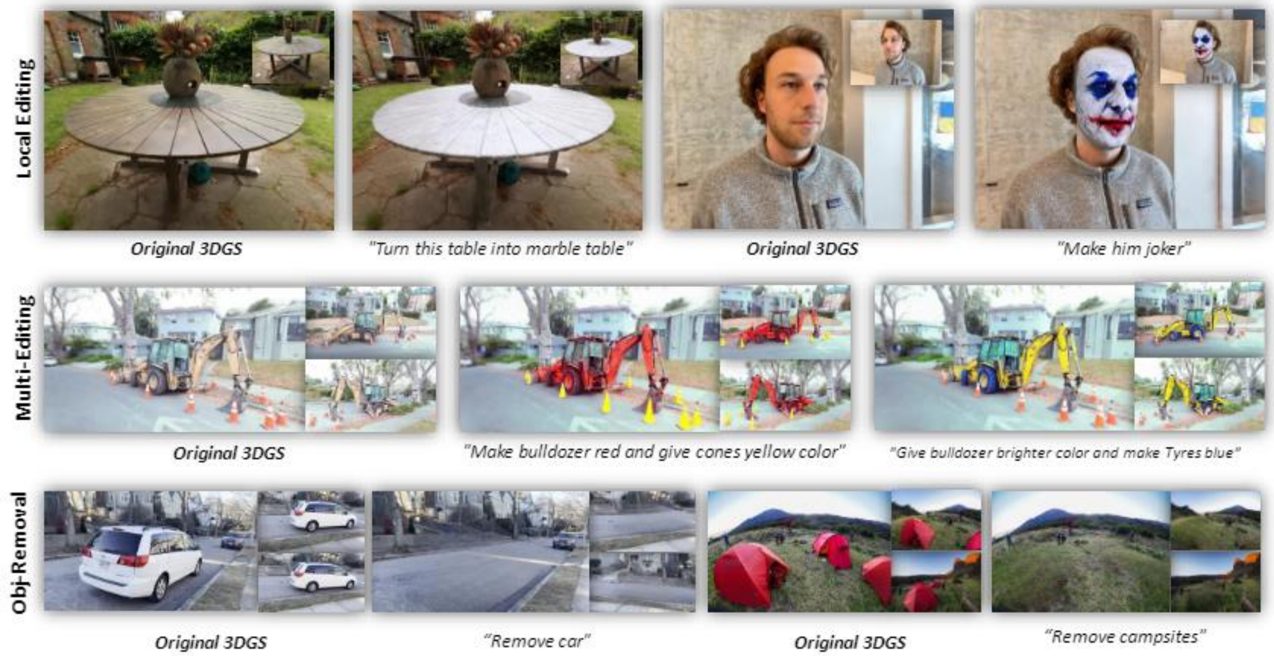
- Gradual 3D Scene Expansion
  - We increase the density of the Gaussians currently under reconstruction as new frames are introduced

    Regularize the Estimated Pose:

$$\mathcal{L}_{pc} = D_{\text{Chamfer}}(\mathcal{M}_i^* \mathcal{H}_i^*, \mathcal{H}_{i+1}^*)$$

Qualitative Results

- Local Editing
- Background is intact
- Geometric Editing



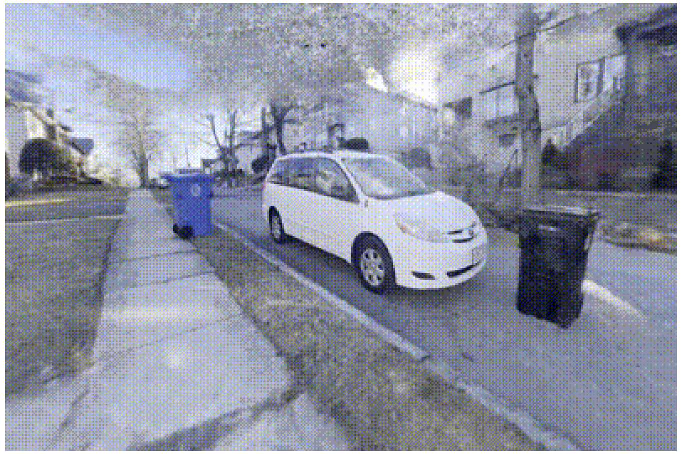
Original 3DGS — "Turn this table into marble table" — Original 3DGS — "Make him joker"

Original 3DGS — "Make bulldozer red and give cones yellow color" — "Give bulldozer brighter color and make Tyres blue"

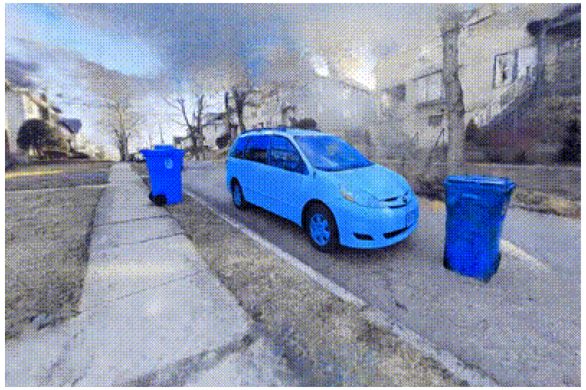
Original 3DGS — "Remove car" — Original 3DGS — "Remove campsites"

Fig. 2: 3DEgo offers rapid, accurate, and adaptable 3D editing, bypassing the need for original 3D scene initialization and COLMAP poses. This ensures compatibility with videos from any source, including casual smartphone captures like the **Van** 360-degree scene. The above results identify three cases challenging for IN2N [11], where our method can convert a monocular video into customized 3D scenes using a streamlined, single-stage reconstruction process.

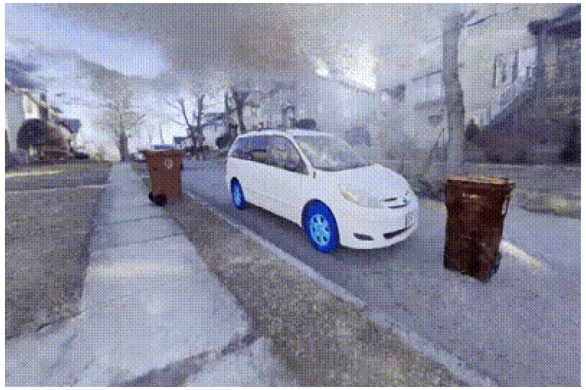
3D Editing Comparison



Give the wheels Blue Color and Make the recycle bins brown.

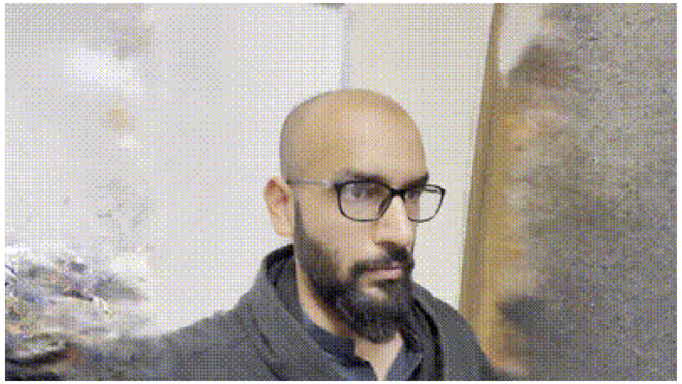


IN2N

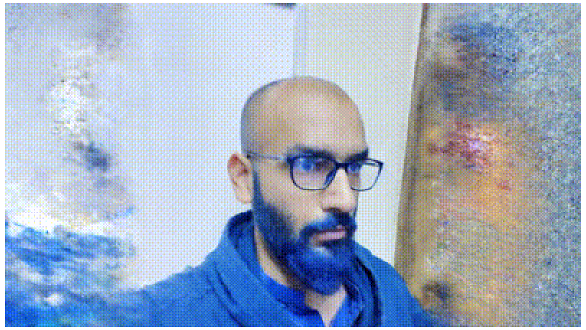


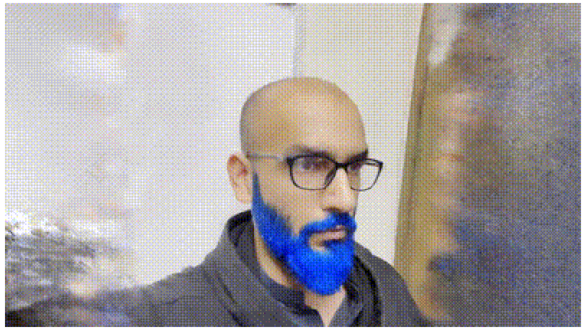
3DEgo

Original Scene



Turn his beard into blue.





Qualitative Results

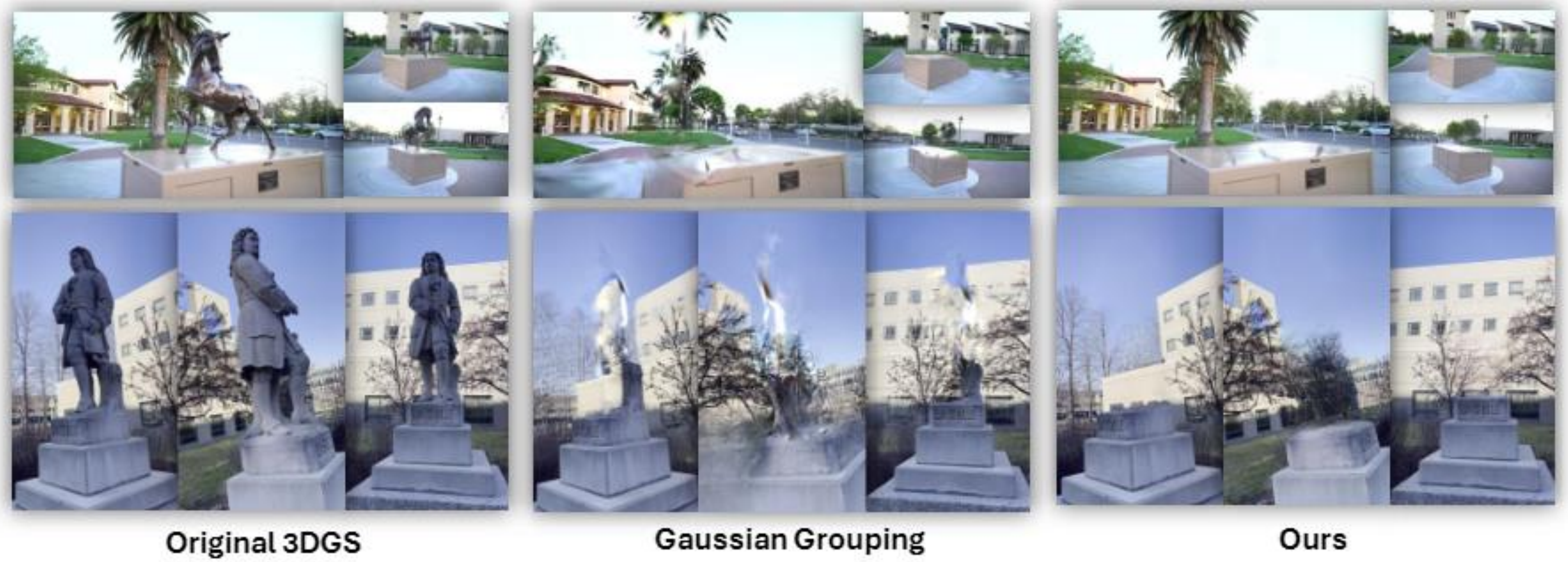


Fig. 5: Our approach surpasses Gaussian Grouping [50] in 3D object elimination across different scenes from GS25 and Tanks & Temple datasets. *3DEgo* is capable of eliminating substantial objects like statues from the entire scene while significantly minimizing artifacts and avoiding a blurred background.

GS25 Dataset Contribution

- Comprises of 25 scenes casually recorded from phone
  - No Stabilizer
  - No calibrated cameras
- Variety of scenes:
  - Indoor & Outdoor
  - Single & Multi-object
  - 360 & 180 degree views
- Public dataset with and w/o COLMAP poses
  - https://3dego.github.io/

three_people_standing

William_statue

Bear_and_girl

Quantitative Results

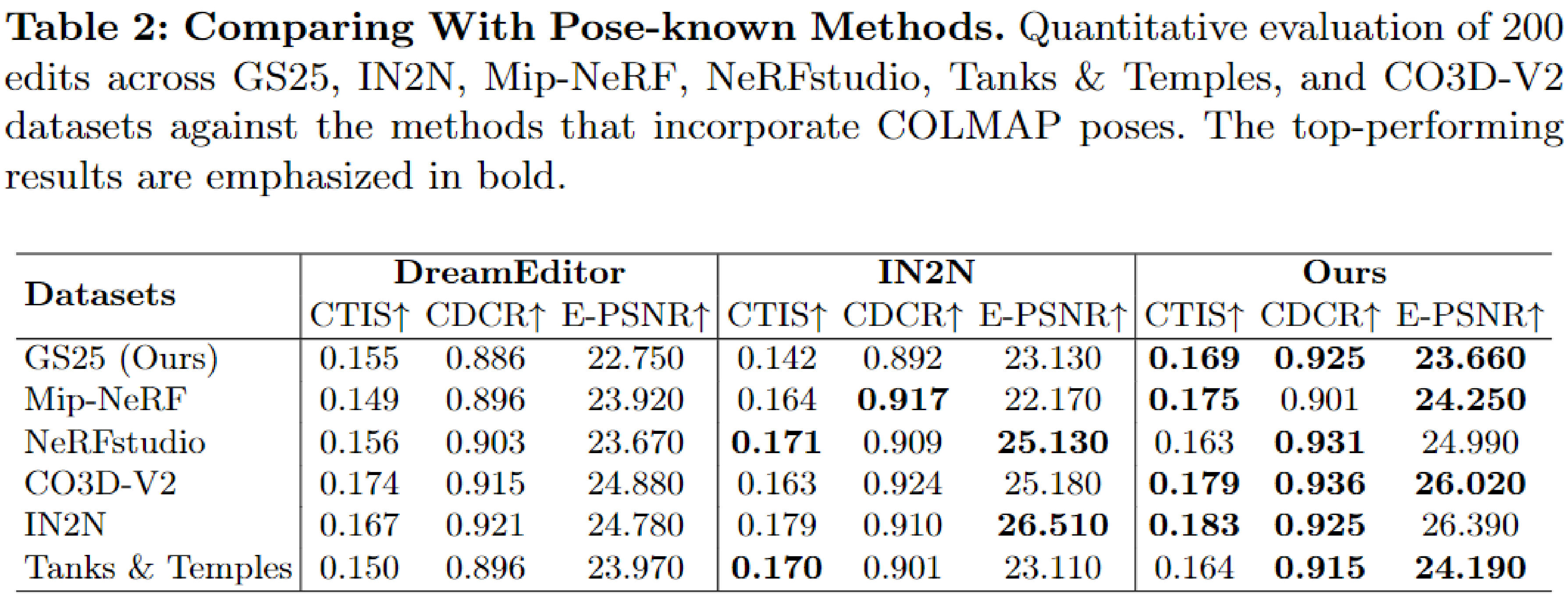


Table 2: Comparing With Pose-known Methods. Quantitative evaluation of 200 edits across GS25, IN2N, Mip-NeRF, NeRFstudio, Tanks & Temples, and CO3D-V2 datasets against the methods that incorporate COLMAP poses. The top-performing results are emphasized in bold.

Datasets	**DreamEditor**			**IN2N**			**Ours**		
	CTIS↑	CDCR↑	E-PSNR↑	CTIS↑	CDCR↑	E-PSNR↑	CTIS↑	CDCR↑	E-PSNR↑
GS25 (Ours)	0.155	0.886	22.750	0.142	0.892	23.130	**0.169**	**0.925**	**23.660**
Mip-NeRF	0.149	0.896	23.920	0.164	**0.917**	22.170	**0.175**	0.901	**24.250**
NeRFstudio	0.156	0.903	23.670	**0.171**	0.909	**25.130**	0.163	**0.931**	24.990
CO3D-V2	0.174	0.915	24.880	0.163	0.924	25.180	**0.179**	**0.936**	**26.020**
IN2N	0.167	0.921	24.780	0.179	0.910	**26.510**	**0.183**	**0.925**	26.390
Tanks & Temples	0.150	0.896	23.970	**0.170**	0.901	23.110	0.164	**0.915**	**24.190**

Beats Pose-known methods under most settings

Quantitative Results

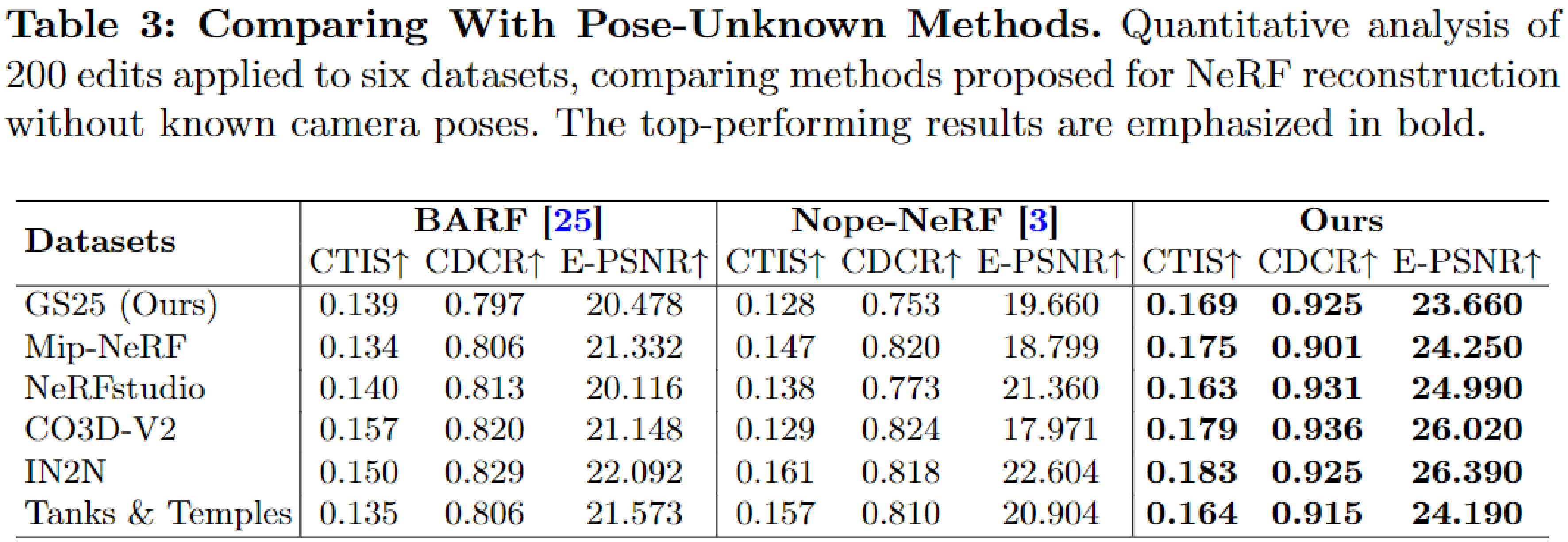


Table 3: Comparing With Pose-Unknown Methods. Quantitative analysis of 200 edits applied to six datasets, comparing methods proposed for NeRF reconstruction without known camera poses. The top-performing results are emphasized in bold.

Datasets	**BARF [25]**			**Nope-NeRF [3]**			**Ours**		
	CTIS↑	CDCR↑	E-PSNR↑	CTIS↑	CDCR↑	E-PSNR↑	CTIS↑	CDCR↑	E-PSNR↑
GS25 (Ours)	0.139	0.797	20.478	0.128	0.753	19.660	**0.169**	**0.925**	**23.660**
Mip-NeRF	0.134	0.806	21.332	0.147	0.820	18.799	**0.175**	**0.901**	**24.250**
NeRFstudio	0.140	0.813	20.116	0.138	0.773	21.360	**0.163**	**0.931**	**24.990**
CO3D-V2	0.157	0.820	21.148	0.129	0.824	17.971	**0.179**	**0.936**	**26.020**
IN2N	0.150	0.829	22.092	0.161	0.818	22.604	**0.183**	**0.925**	**26.390**
Tanks & Temples	0.135	0.806	21.573	0.157	0.810	20.904	**0.164**	**0.915**	**24.190**

Beats Pose-Unknown NeRF Methods: Implicit vs Explicit Modeling

Recap and Limitations

- Eliminated COLMAP requirement
- Model initialization on unedited images is not necessary
- Text Conditioned 3D scene from a monocular video
- LLM guided SAM has been applied into 3D scene editing
- Limitations:
  - 3D model training is required on edited scenes
  - Not a one-shot editor – Gaussians are learnt for a specific scene

Thanks