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

ACM Transactions on Graphics, 2023

Source: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/


Menu
Views
Capture
3D Gaussians
Camera Point view
Metrics
57.56 (17.37 ms)
VSync On

Point-Based Rendering and Radiance Fields

$$C = \sum_{i=1}^{N} T_i(1 - \exp(-\sigma_i \delta_i))\mathbf{c}_i \qquad T_i = \exp\left(-\sum_{j=1}^{i-1} \sigma_j \delta_j\right)$$

transmittance

density

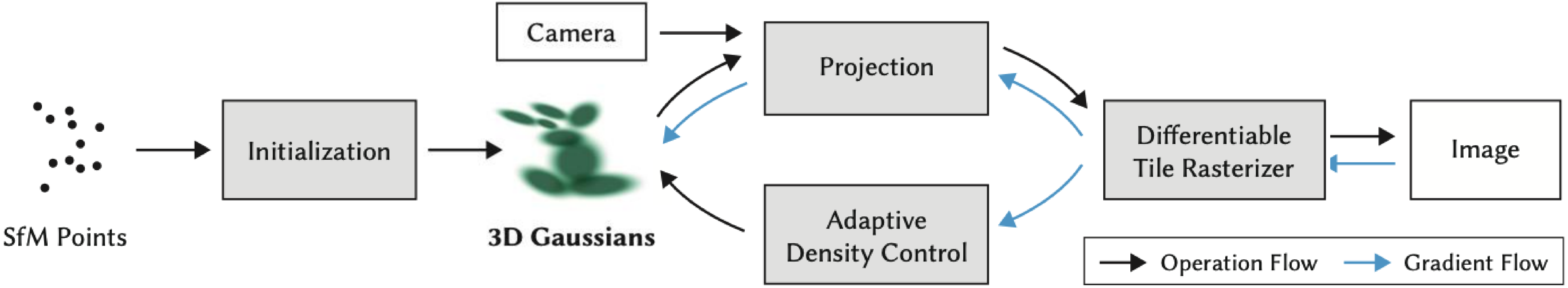
color

intervals

$$C = \sum_{i=1}^{N} T_i \alpha_i \mathbf{c}_i \qquad \alpha_i = (1 - \exp(-\sigma_i \delta_i)) \qquad T_i = \prod_{j=1}^{i-1}(1 - \alpha_i)$$

neural point-based $$C = \sum_{i \in \mathcal{N}} c_i \alpha_i \prod_{j=1}^{i-1}(1 - \alpha_j)$$

3D Gaussian Splatting for Real-Time Radiance Field Rendering



introduce 3D Gaussians

Reasonably
Compact, Unstructured, Precise
representation of the scene

optimization of the properties

1. 3D position
2. opacity α
3. anisotropic covariance
4. spherical harmonic (SH) coefficients

real-time rendering

- fast GPU sorting algorithms
- tile-based rasterization

DIFFERENTIABLE 3D GAUSSIAN SPLATTING

→ model the geometry as a set of 3D Gaussians

$$G(x) = e^{-\frac{1}{2}(x)^T \Sigma^{-1}(x)}$$

3D covariance matrix
(world space)

$$\Sigma = RSS^T R^T$$

rotation matrix
(quaternion q)

scaling matrix
(3D vector s)

→ 3D Gaussians to 2D projection for rendering

$$\Sigma' = JW \, \Sigma \, W^T J^T$$

Jacobian

covariance matrix
(camera coordinates)

viewing transformation

OPTIMIZATION WITH ADAPTIVE DENSITY CONTROL OF 3D GAUSSIANS

1. 3D position
2. opacity α
3. anisotropic covariance
4. spherical harmonic (SH) coefficients

Optimize (Render and Compare)

3D to 2D projection → incorrectly placed geometry

Create/destroy/move geometry

large homogeneous areas → small number of large anisotropic Gaussians

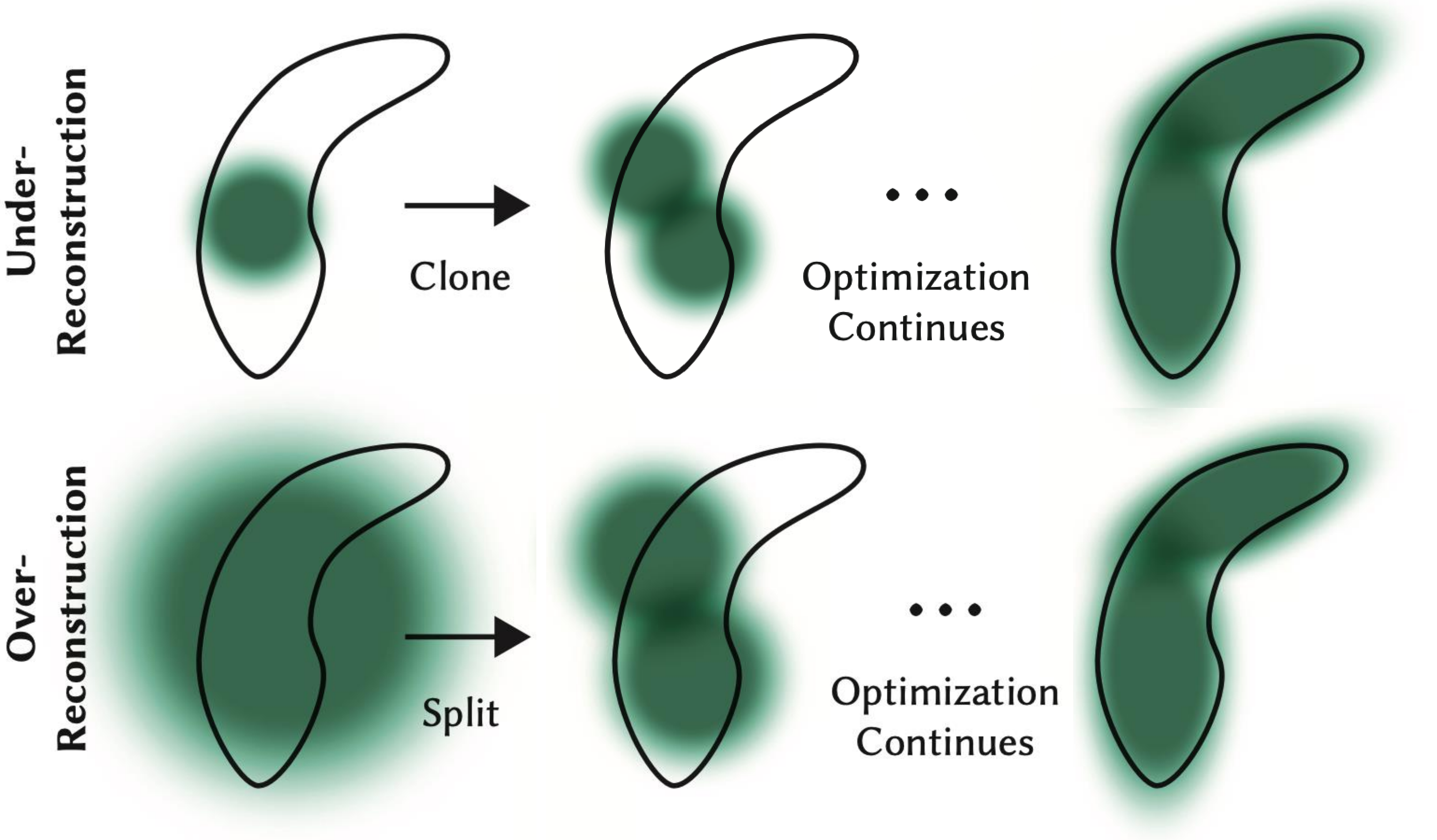
Adaptive Control of Gaussians (number & density/unit volume)

Sparser → Denser

Populate empty areas → focuses on regions
- missing geometric features (under-reconstruction)
- Gaussians cover large areas (over-reconstruction)

Remove transparent Gaussians

Remove Gaussians (large in worldspace/viewspace)


Under-
Reconstruction
Clone
Optimization
Continues
Over-
Reconstruction
Split
Optimization
Continues

Algorithm 1 Optimization and Densification

w, h: width and height of the training images

```
M ← SfM Points                                          ▷ Positions
S, C, A ← InitAttributes()          ▷ Covariances, Colors, Opacities
i ← 0                                             ▷ Iteration Count
while not converged do
    V, Î ← SampleTrainingView()               ▷ Camera V and Image
    I ← Rasterize(M, S, C, A, V)                          ▷ Alg. 2
    L ← Loss(I, Î)                                          ▷ Loss
    M, S, C, A ← Adam(∇L)                       ▷ Backprop & Step
    if IsRefinementIteration(i) then
        for all Gaussians (μ, Σ, c, α) in (M, S, C, A) do
            if α < ε or IsTooLarge(μ, Σ) then             ▷ Pruning
                RemoveGaussian()
            end if
            if ∇_p L > τ_p then                     ▷ Densification
                if ||S|| > τ_S then           ▷ Over-reconstruction
                    SplitGaussian(μ, Σ, c, α)
                else                         ▷ Under-reconstruction
                    CloneGaussian(μ, Σ, c, α)
                end if
            end if
        end for
    end if
    i ← i + 1
end while
```

FAST DIFFERENTIABLE RASTERIZER FOR GAUSSIANS

Goals: fast overall rendering and fast sorting
- **allow** approximate α-blending
- **avoid** hard limits on the number of splats (receive gradients)

→ tile-based rasterizer for Gaussian splats

- splitting the screen into 16×16 tiles
- cull 3D Gaussians against the view frustum and each tile
  - only keep Gaussians with a 99% confidence interval intersecting the view frustum

- instantiate each Gaussian (number of overlapping tiles) → combines (view space depth & tile ID)
- assign each instance a key
- sort Gaussians based on these keys
- produce a list for each tile

Algorithm 2 GPU software rasterization of 3D Gaussians

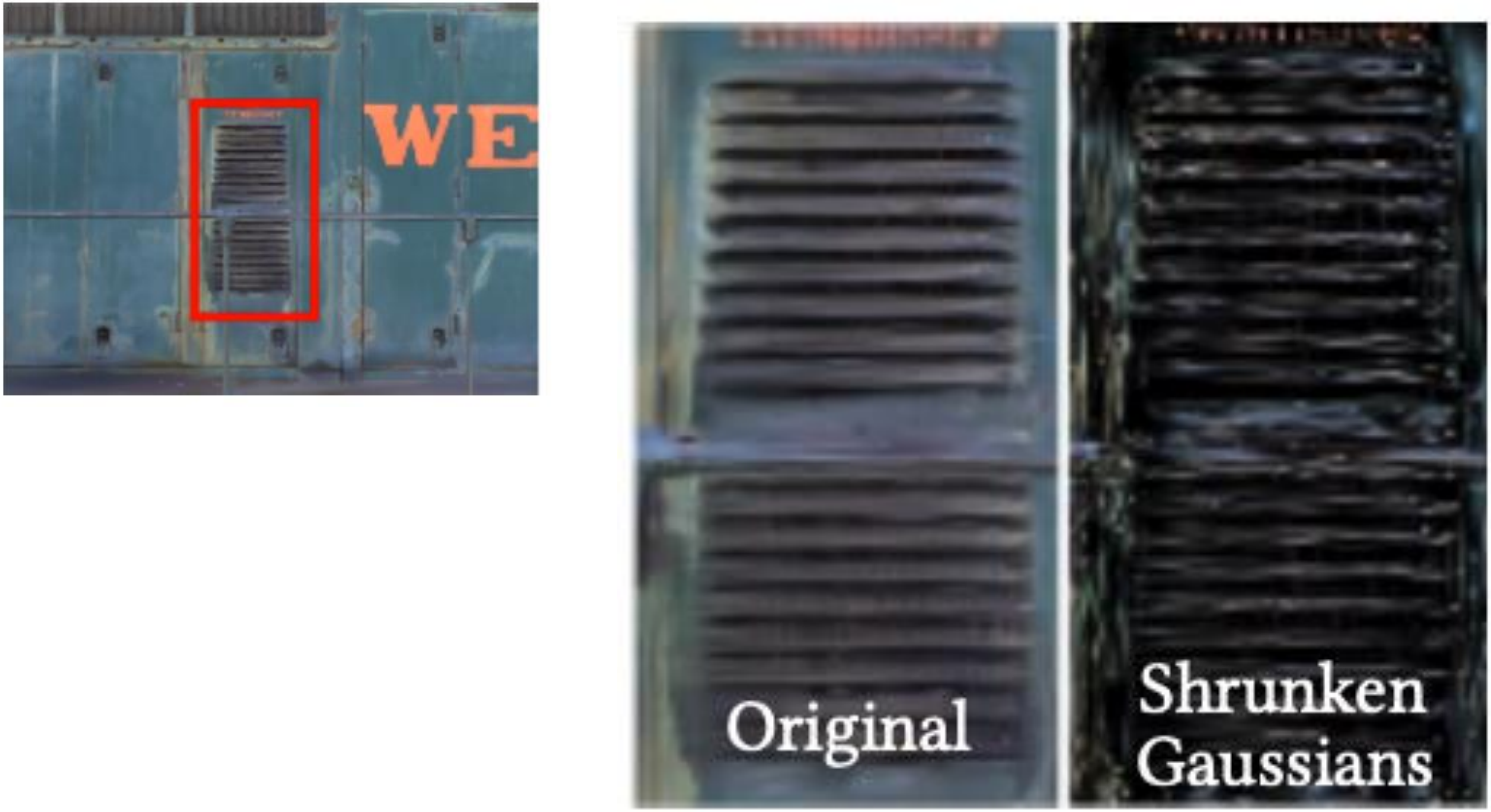
w, h: width and height of the image to rasterize

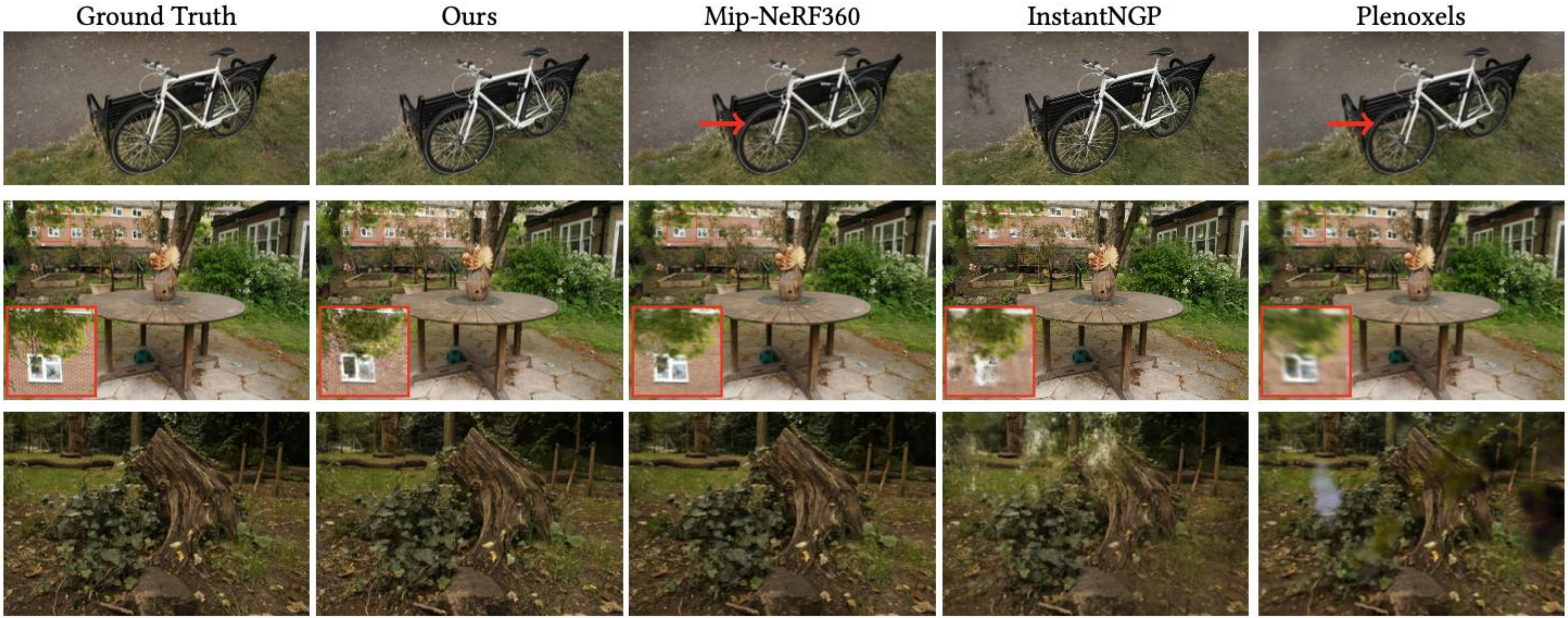
M, S: Gaussian means and covariances in world space

C, A: Gaussian colors and opacities

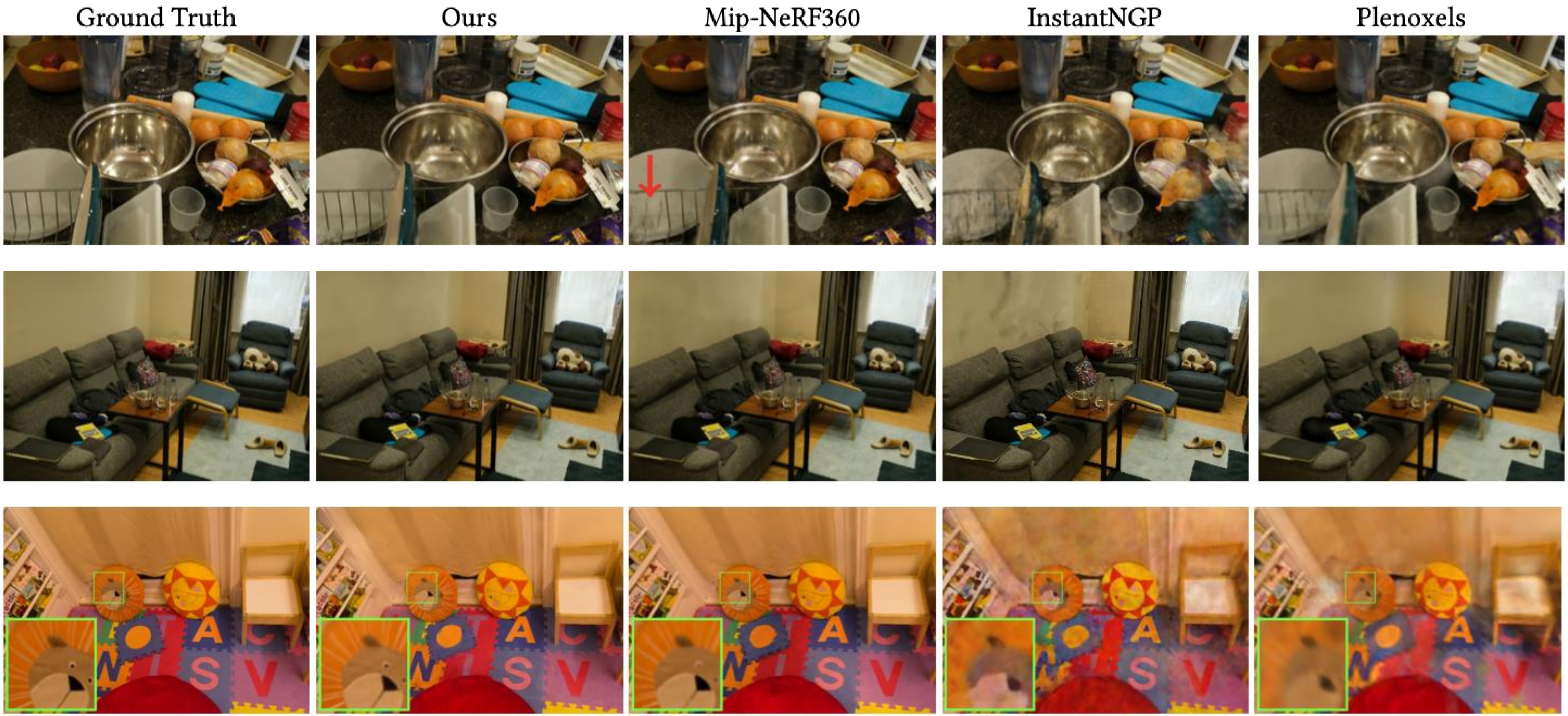
V: view configuration of current camera

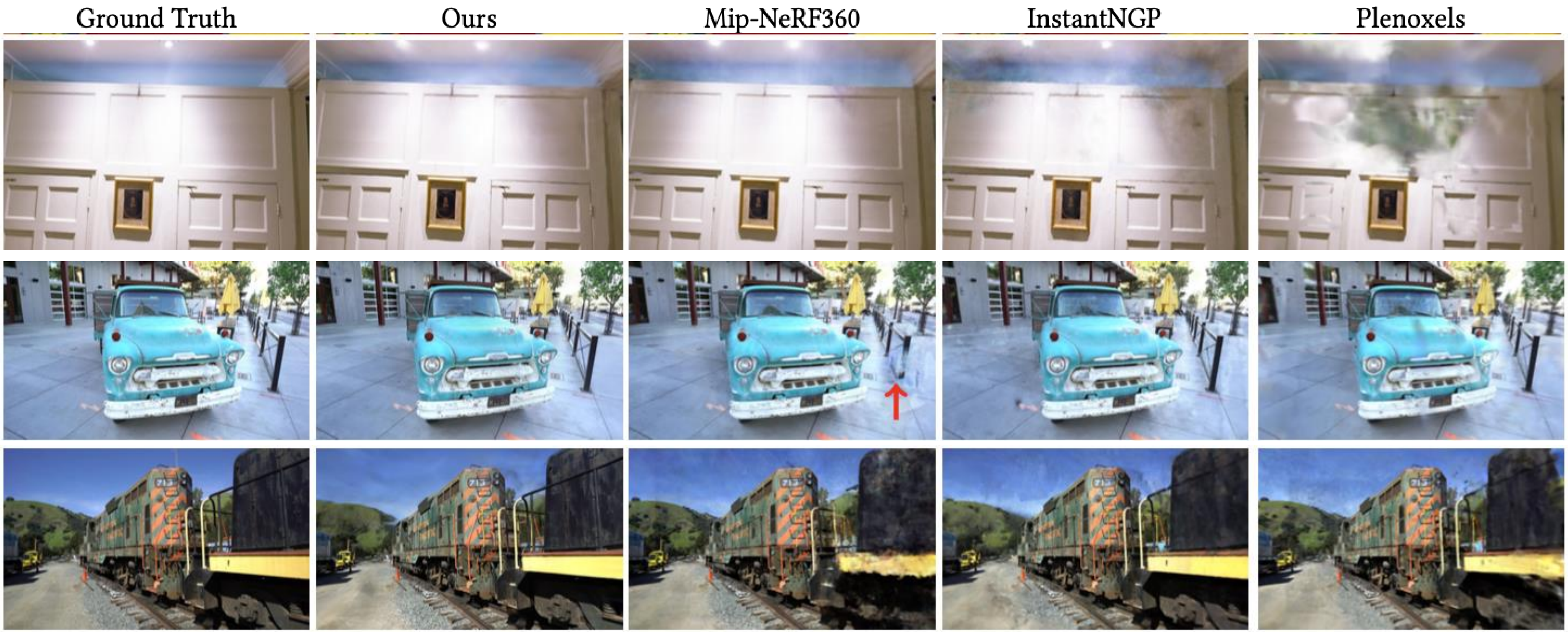
```
function RASTERIZE(w, h, M, S, C, A, V)
    CullGaussian(p, V)                              ▷ Frustum Culling
    M', S' ← ScreenspaceGaussians(M, S, V)          ▷ Transform
    T ← CreateTiles(w, h)
    L, K ← DuplicateWithKeys(M', T)                 ▷ Indices and Keys
    SortByKeys(K, L)                                ▷ Globally Sort
    R ← IdentifyTileRanges(T, K)
    I ← 0                                           ▷ Init Canvas
    for all Tiles t in I do
        for all Pixels i in t do
            r ← GetTileRange(R, t)
            I[i] ← BlendInOrder(i, L, r, K, M', S', C, A)
        end for
    end for
     return I
end function
```


WE
Original
Shrunken
Gaussians



Ground Truth
Ours
Mip-NeRF360
InstantNGP
Plenoxels


Ground Truth
Ours
Mip-NeRF360
InstantNGP
Plenoxels

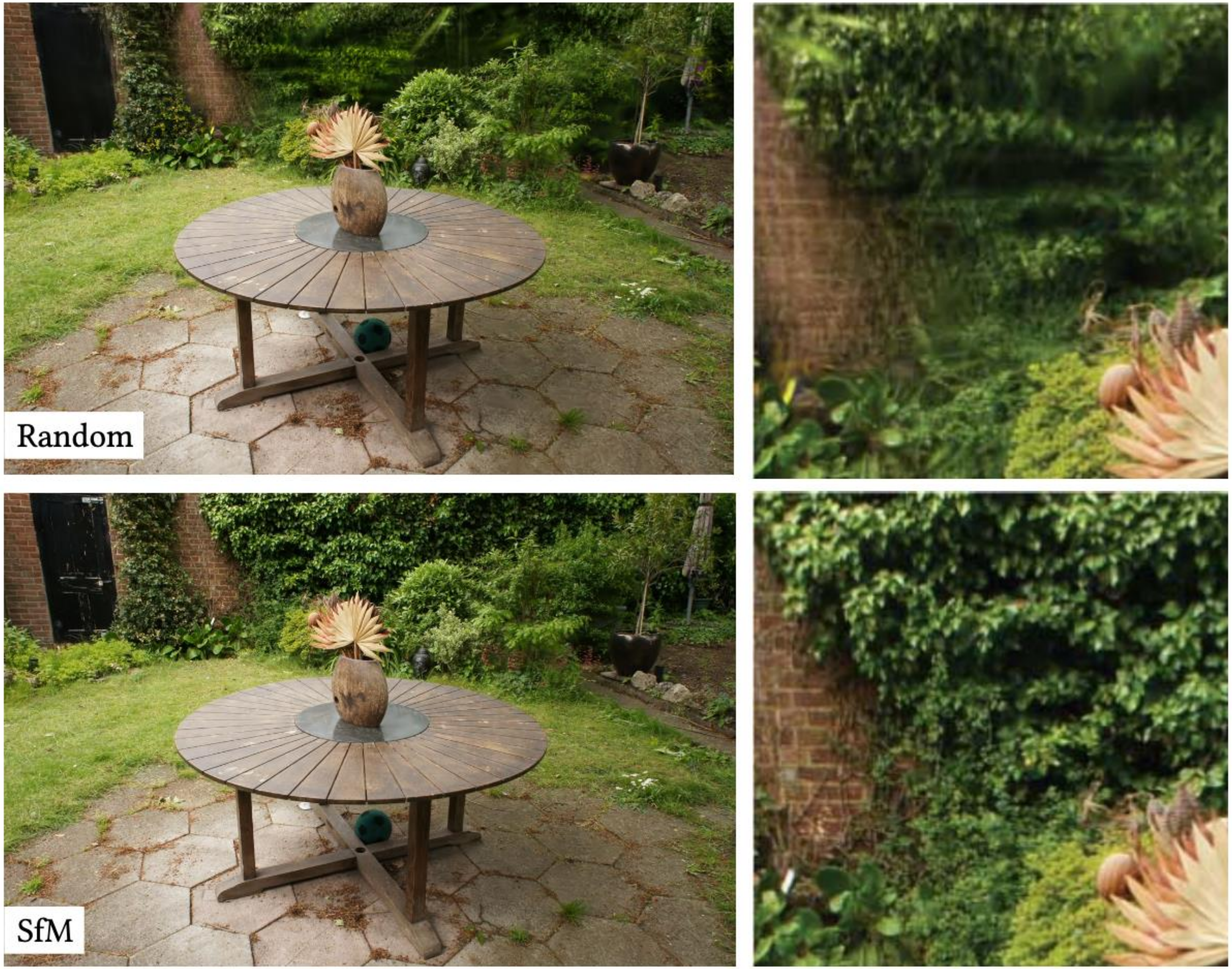

Ground Truth
Ours
Mip-NeRF360
InstantNGP
Plenoxels

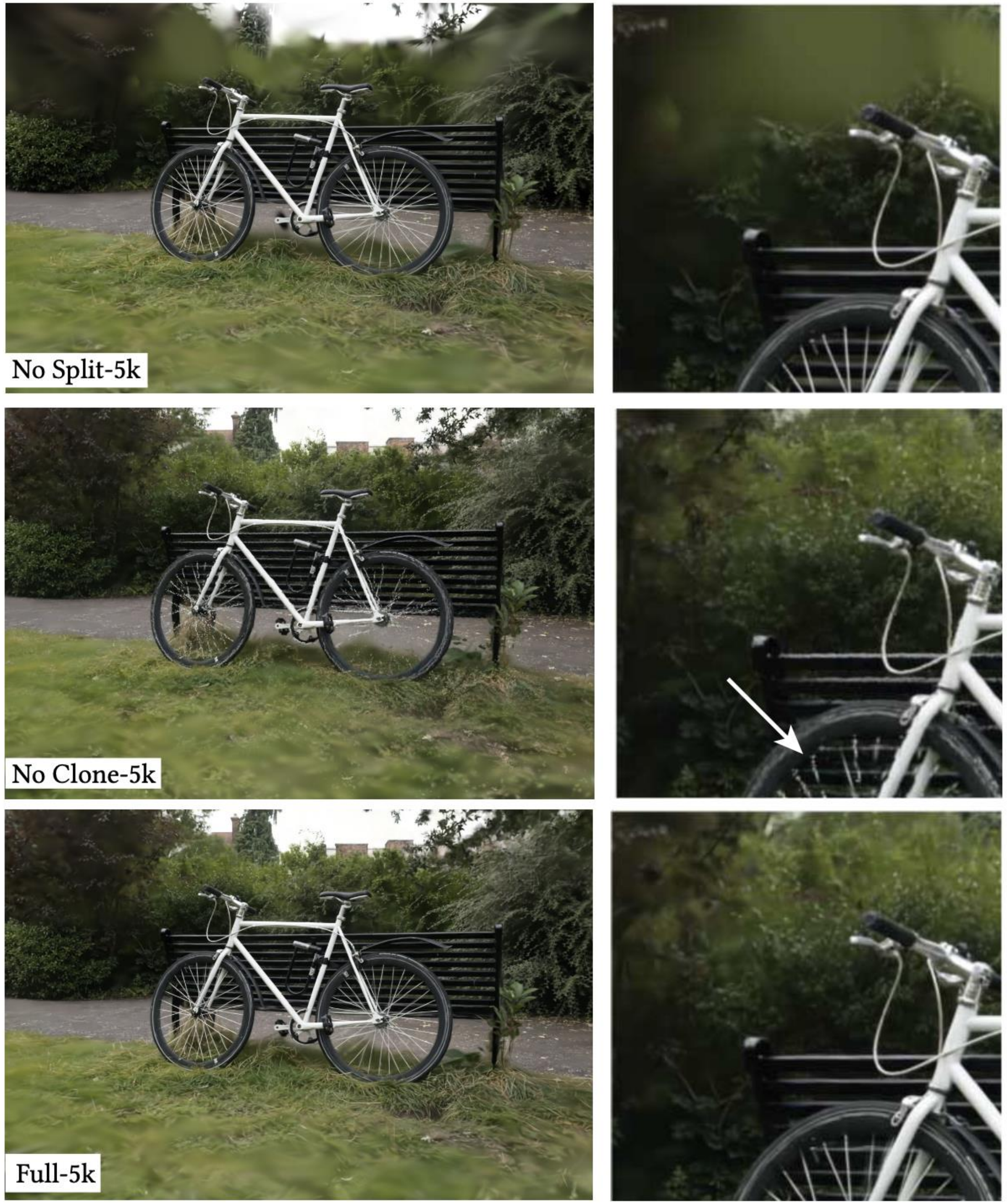
Dataset	Mip-NeRF360						Tanks&Temples						Deep Blending					
Method|Metric	*SSIM*↑	*PSNR*↑	*LPIPS*↓	Train	FPS	Mem	*SSIM*↑	*PSNR*↑	*LPIPS*↓	Train	FPS	Mem	*SSIM*↑	*PSNR*↑	*LPIPS*↓	Train	FPS	Mem
Plenoxels	0.626	23.08	0.463	25m49s	6.79	2.1GB	0.719	21.08	0.379	25m5s	13.0	2.3GB	0.795	23.06	0.510	27m49s	11.2	2.7GB
INGP-Base	0.671	25.30	0.371	5m37s	11.7	13MB	0.723	21.72	0.330	5m26s	17.1	13MB	0.797	23.62	0.423	6m31s	3.26	13MB
INGP-Big	0.699	25.59	0.331	7m30s	9.43	48MB	0.745	21.92	0.305	6m59s	14.4	48MB	0.817	24.96	0.390	8m	2.79	48MB
M-NeRF360	0.792†	27.69†	0.237†	48h	0.06	8.6MB	0.759	22.22	0.257	48h	0.14	8.6MB	0.901	29.40	0.245	48h	0.09	8.6MB
Ours-7K	0.770	25.60	0.279	6m25s	160	523MB	0.767	21.20	0.280	6m55s	197	270MB	0.875	27.78	0.317	4m35s	172	386MB
Ours-30K	0.815	27.21	0.214	41m33s	134	734MB	0.841	23.14	0.183	26m54s	154	411MB	0.903	29.41	0.243	36m2s	137	676MB


Menu
Views
Capture
▶ 3D Gaussians
▶ Camera Point view
▼ Metrics
59.97 (16.67 ms)
VSync On

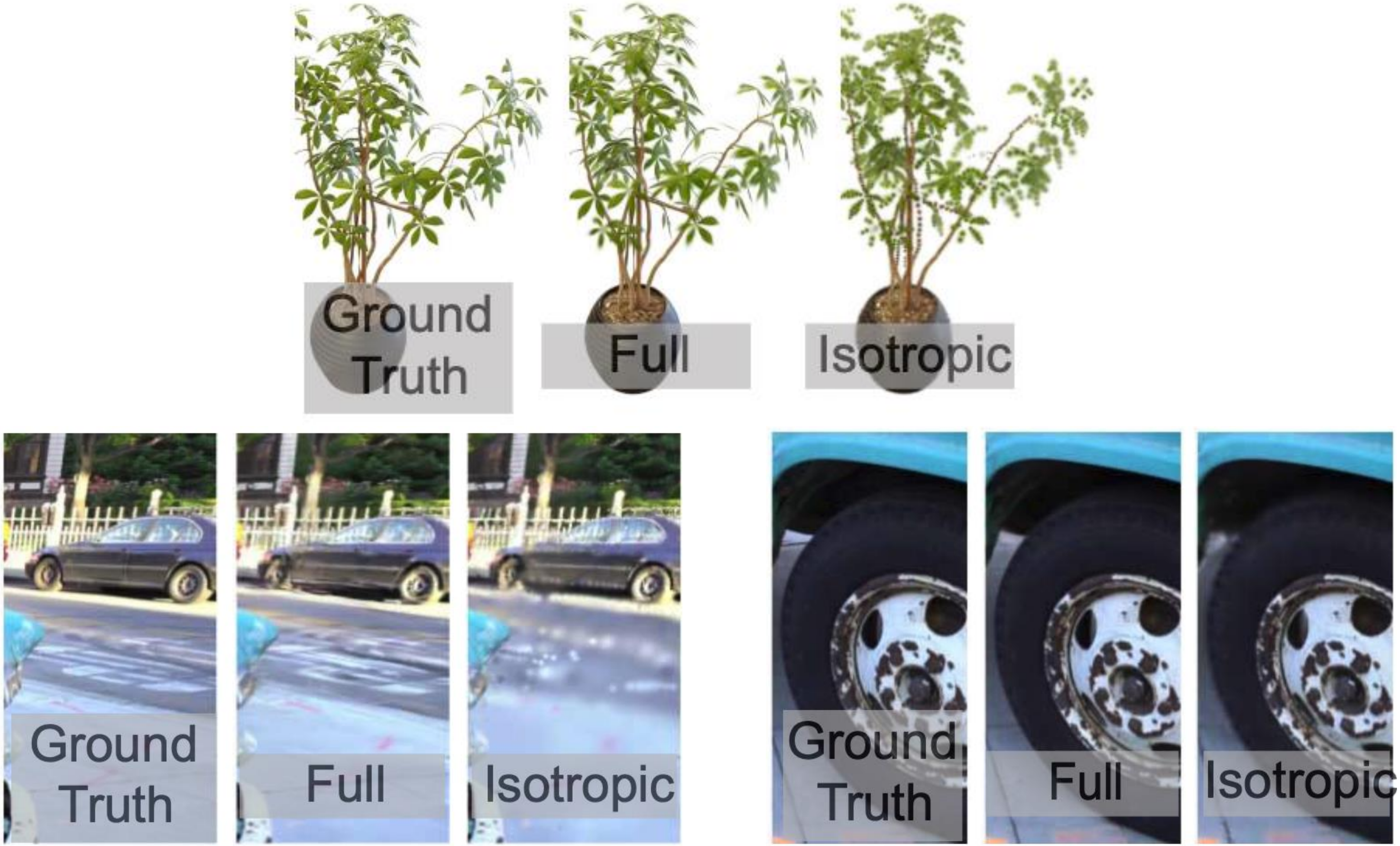
	Mic	Chair	Ship	Materials	Lego	Drums	Ficus	Hotdog	Avg.
Plenoxels	33.26	33.98	29.62	29.14	34.10	25.35	31.83	36.81	31.76
INGP-Base	36.22	35.00	31.10	29.78	36.39	26.02	33.51	37.40	33.18
Mip-NeRF	36.51	35.14	30.41	30.71	35.70	25.48	33.29	37.48	33.09
Point-NeRF	35.95	35.40	30.97	29.61	35.04	26.06	36.13	37.30	33.30
Ours-30K	35.36	35.83	30.80	30.00	35.78	26.15	34.87	37.72	33.32

	Truck-5K	Garden-5K	Bicycle-5K	Truck-30K	Garden-30K	Bicycle-30K	Average-5K	Average-30K
Limited-BW	14.66	22.07	20.77	13.84	22.88	20.87	19.16	19.19
Random Init	16.75	20.90	19.86	18.02	22.19	21.05	19.17	20.42
No-Split	18.31	23.98	22.21	20.59	26.11	25.02	21.50	23.90
No-SH	22.36	25.22	22.88	24.39	26.59	25.08	23.48	25.35
No-Clone	22.29	25.61	22.15	24.82	27.47	25.46	23.35	25.91
Isotropic	22.40	25.49	22.81	23.89	27.00	24.81	23.56	25.23
Full	22.71	25.82	23.18	24.81	27.70	25.65	23.90	26.05


Random
SfM


No Split-5k
No Clone-5k
Full-5k


Menu
Views
Capture
▶ 3D Gaussians
▶ Camera Point view
Die Aufnahme wurde begonnen
▼ Metrics
78.72 (12.70 ms)


Ground Truth
Full
Isotropic
Ground Truth
Full
Isotropic
Ground Truth
Full
Isotropic


713

Q&A

Source: https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/