3D Gaussian Splatting for Real-Time Radiance Field Rendering

Algorithm

Input – set of images of static scene

1. Get sparse point cloud by SfM

2. Create a set of 3D Gaussians

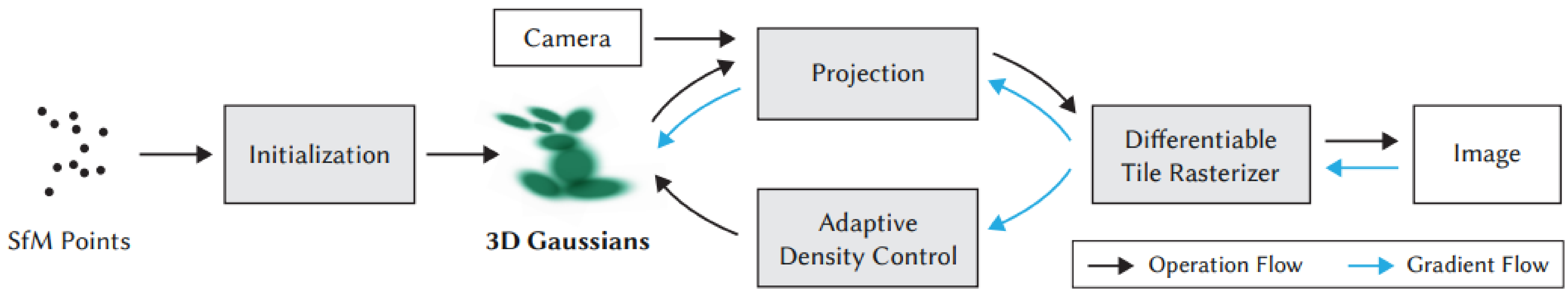
3. Optimize parameters of Gaussians with gradient descent

4. Render an image

Sparse point cloud

Pipeline

3D Gaussian Splatting for Real-Time Radiance Field Rendering • 1:5



Parameters of a Gaussian

- Position (mean)

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

- Covariance matrix (3x3)

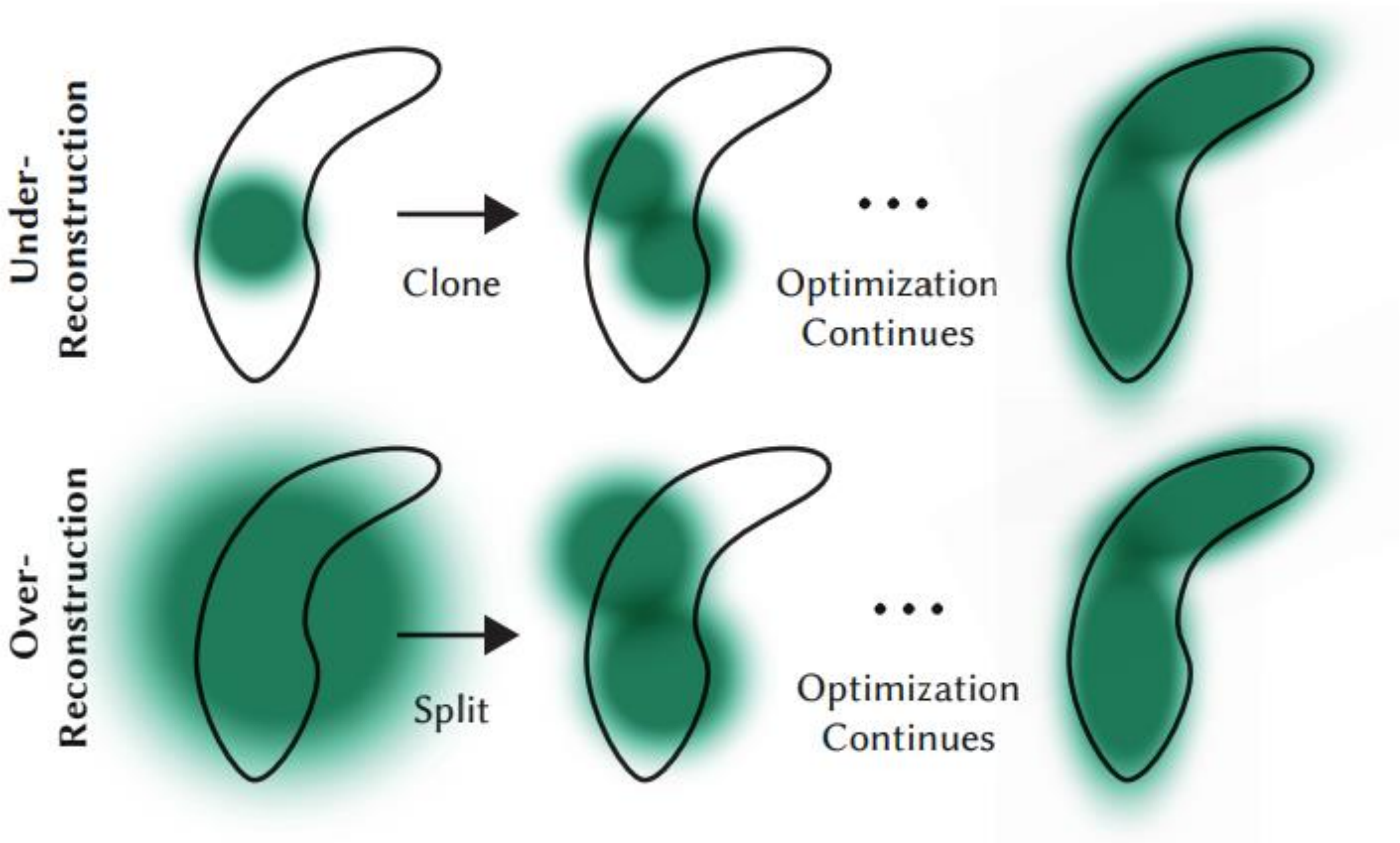
$$\Sigma = RSS^T R^T$$ - should be positive semi-definite

R – rotation matrix
S – scale matrix

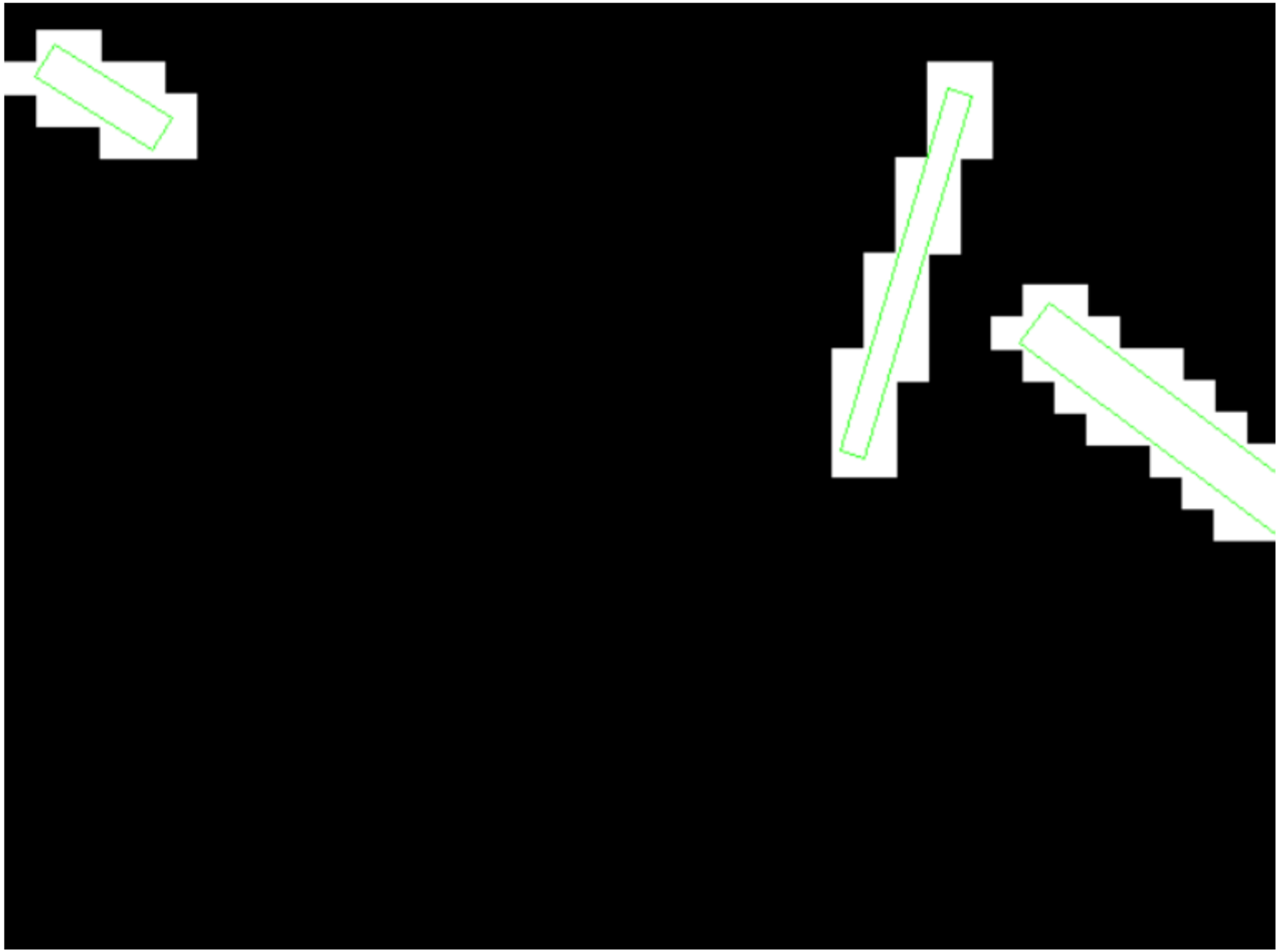
- Color (RGB)

- Opacity ([0, 1])

Gaussian control



Rasterizer



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

α_i - opacity of i-th point

c_i - color of i-th point

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
```

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
```

Metrics

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

	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


Ground Truth
Full
Isotropic

Problems

- Artifacts in regions where the scene is not well observed

- Memory consumption

References

- https://arxiv.org/abs/2308.04079v1 - main article
- https://github.com/graphdeco-inria/gaussian-splatting - implementation
- https://github.com/joeyan/gaussian_splatting/blob/main/MATH.md - math of gaussian splatting
- https://huggingface.co/blog/gaussian-splatting - hugging-face review
- https://habr.com/ru/articles/768590/ - habr review