3D Gaussian Splatting for Real-Time Radiance Field Rendering

Bernard Kerbl et al.
Presented by: Paolo Didier Alfano

Representing scenes

Representing scenes

Input Image

Representing scenes

Input Image

Train

Representing scenes

Input Image

Train

New views!

Amazing motivational slide

Amazing motivational slide

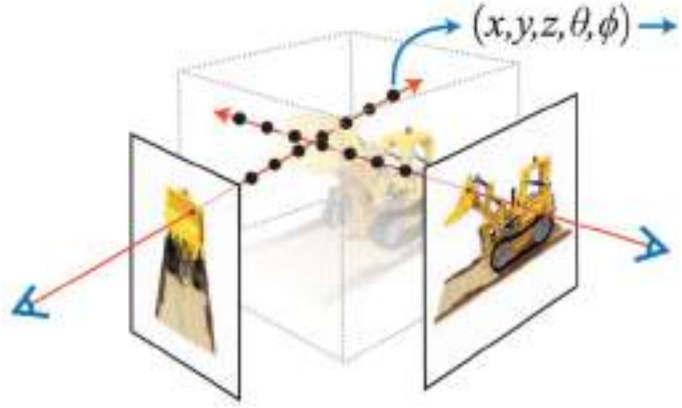
UniGe | MaLGa

NeRF recap

Gaussian splatting

Conclusions

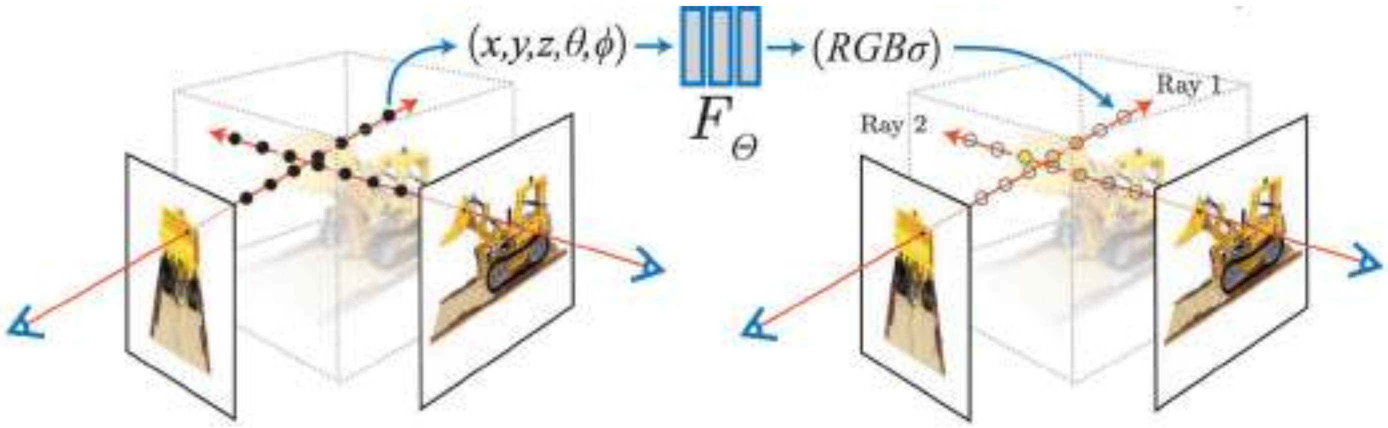
Pipeline: bird's eye view



a) NN input:

point from ray

Pipeline: bird's eye view



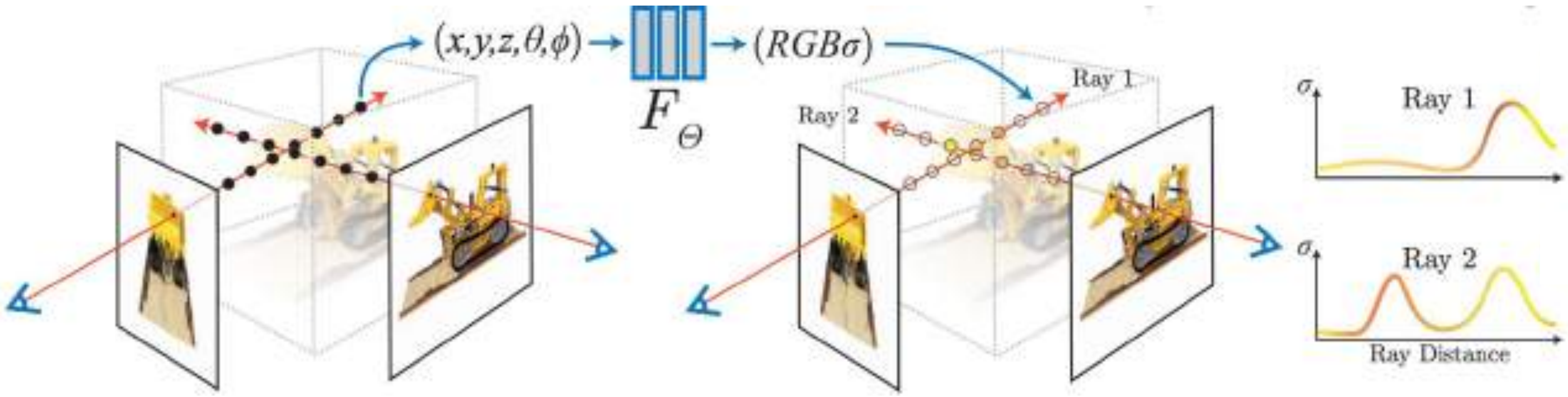
a) NN input:

point from ray

b) NN output:

Point color
and density

Pipeline: bird's eye view



a) NN input:

point from ray

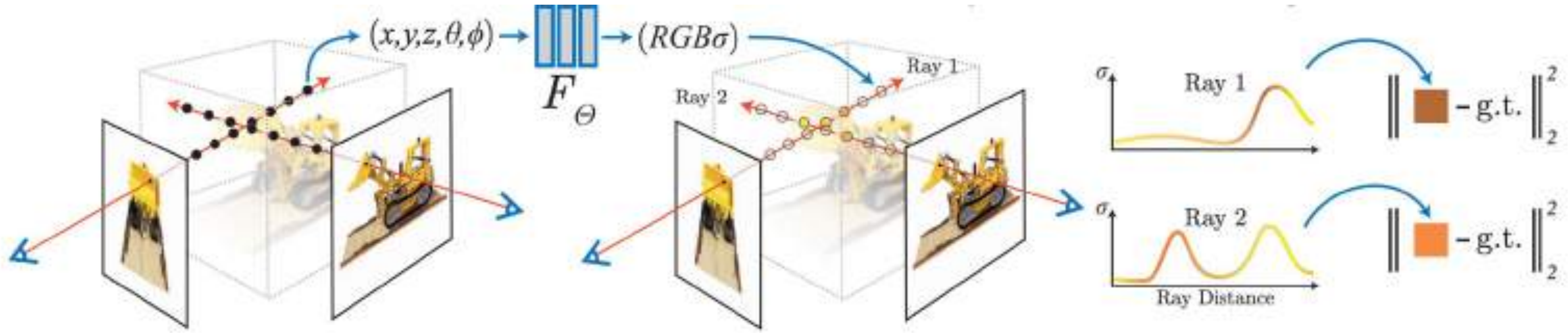
b) NN output:

Point color and density

c) Overall ray

Putting points together

Pipeline: bird's eye view



a) NN input:

point from ray

b) NN output:

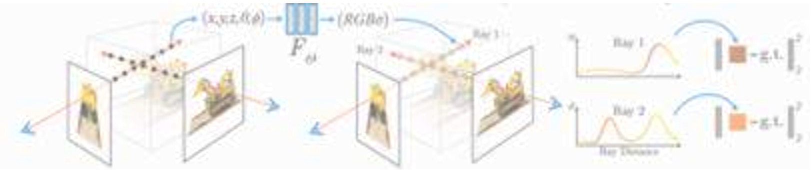
Point color and density

c) Overall ray

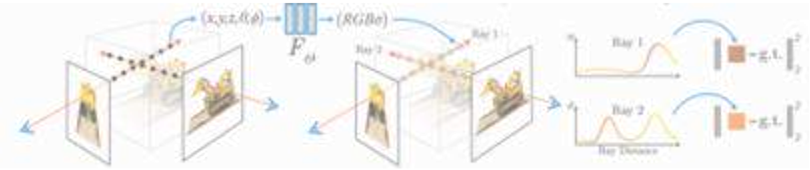
Putting points together

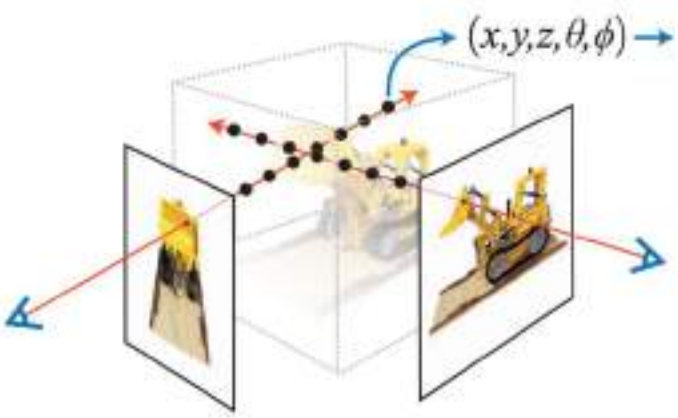
d) Loss

Images & rays



Images & rays



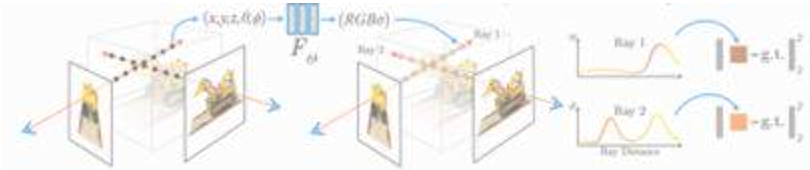


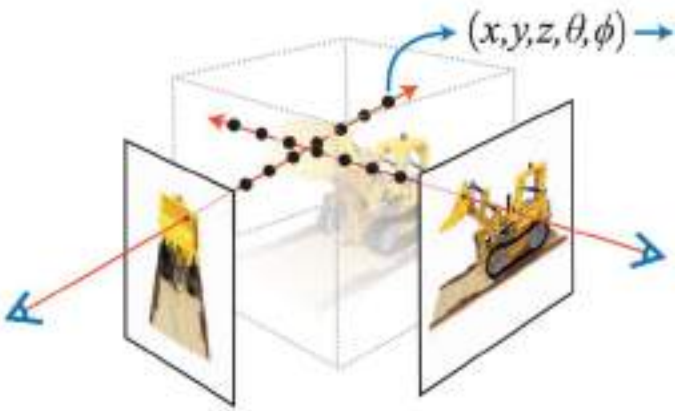
We have:

Images

Camera pose

Images & rays





We have:

Images

Camera pose

Dataset size? $\#\text{images} \cdot \#\text{pixels} \cdot \#\text{points}$

UniGe | MaLGa

NeRF recap

Gaussian splatting

Conclusions

Problems with NeRF?

Rendering efficiency?

NeRF: 0.071 fps

Problems with NeRF?

Rendering efficiency?

NeRF: 0.071 fps

Common problem:

Problems with NeRF?

Rendering efficiency?

NeRF:	0.071 fps

Common problem:

InstantNGP	9.2 fps
Plenoxels	8.2 fps

Problems with NeRF?

Training efficiency?

NeRF:	48 hrs

Problems with NeRF?

Training efficiency?

NeRF: 48 hrs

Why? $\#\text{images} \cdot \#\text{pixels} \cdot \#\text{points}$

Problems with NeRF?

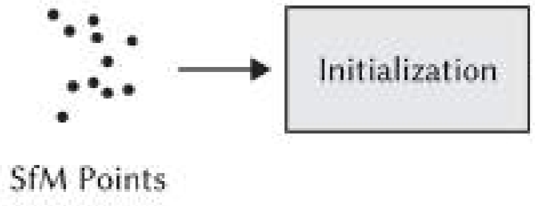
Training efficiency?

NeRF: 48 hrs

Why? $\#\text{images} \cdot \#\text{pixels} \cdot \#\text{points}$

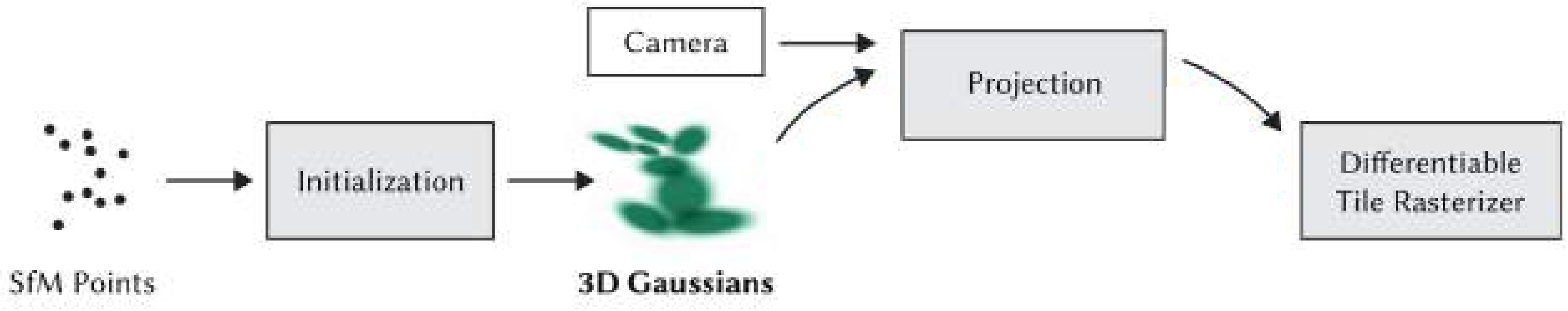
NN inference, for every sample

Gaussian splatting pipeline

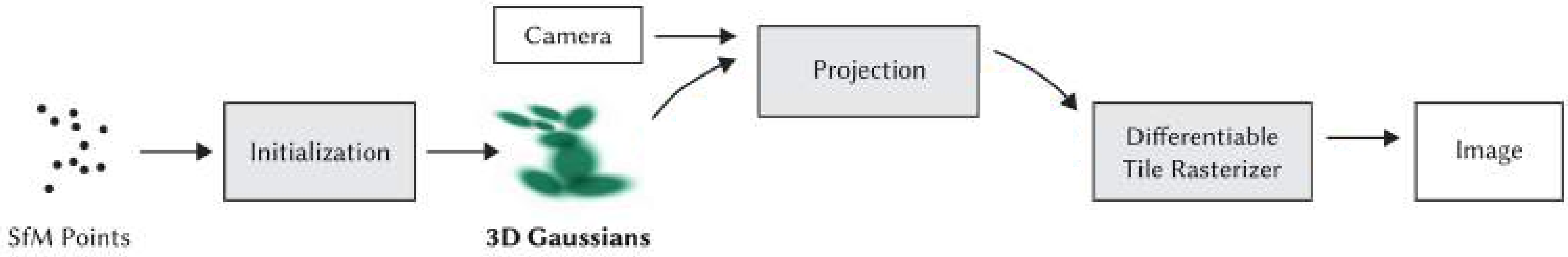


Gaussian splatting pipeline

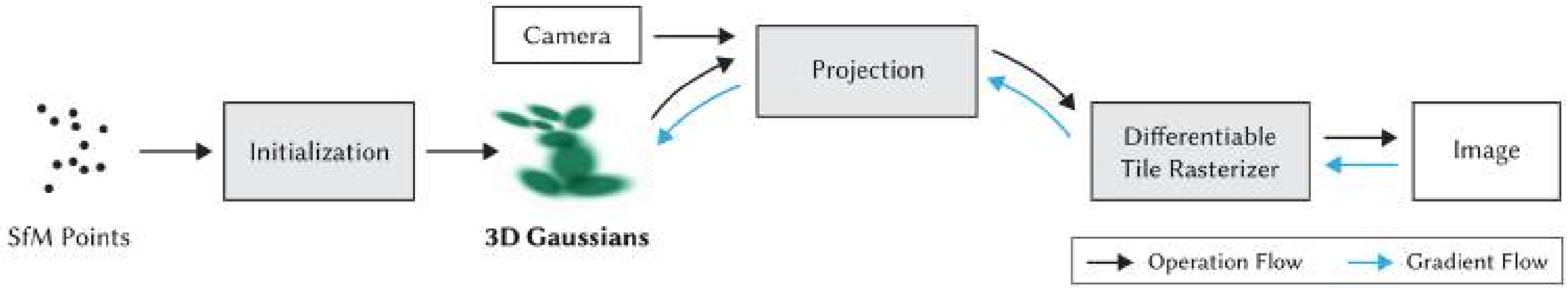
Gaussian splatting pipeline



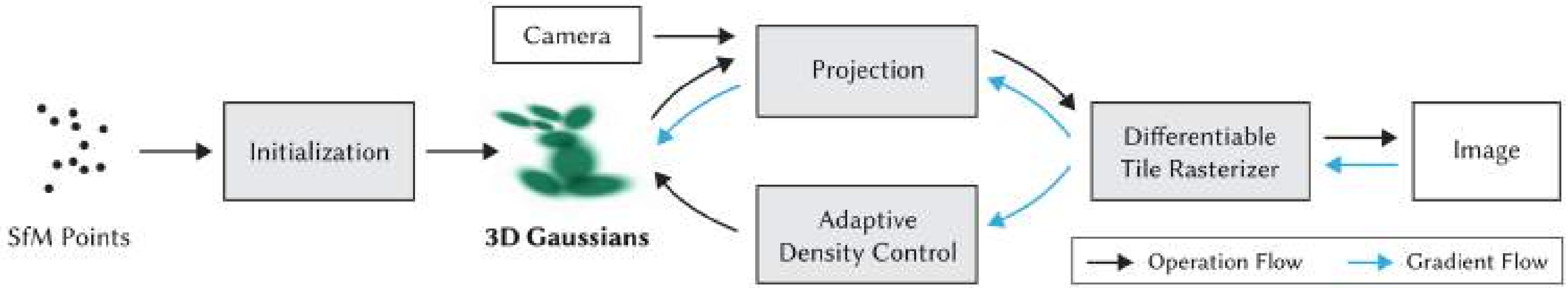
Gaussian splatting pipeline



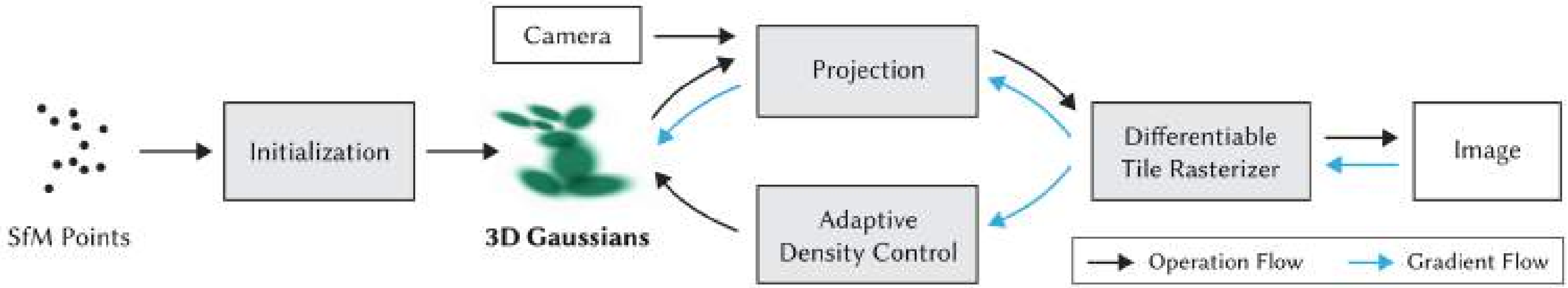
Gaussian splatting pipeline



Gaussian splatting pipeline

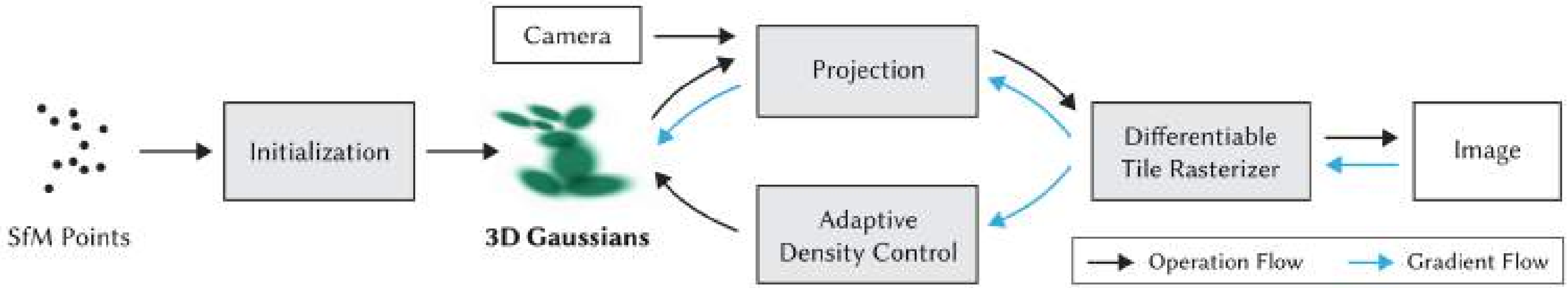


Gaussian splatting pipeline



No neural networks!

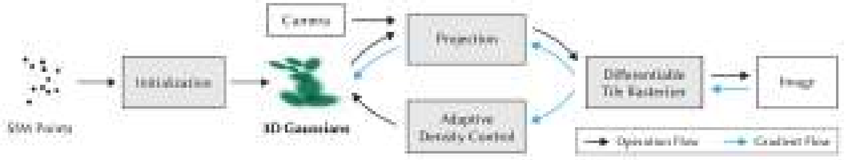
Gaussian splatting pipeline



No neural networks!

Modern ML + traditional computer vision methods

Gaussian splatting pipeline



Structure From Motion (SfM)

Input:

Multiple images

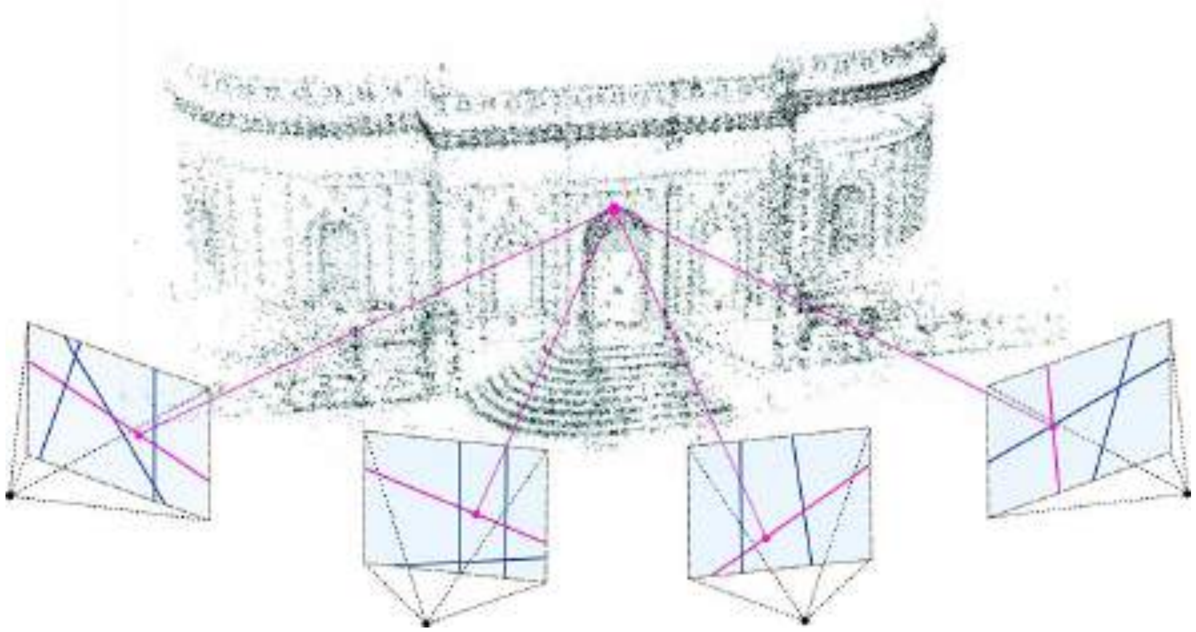
Structure From Motion (SfM)

Input:

Multiple images

Output:

Reconstruct camera poses



Structure From Motion (SfM)

Input:

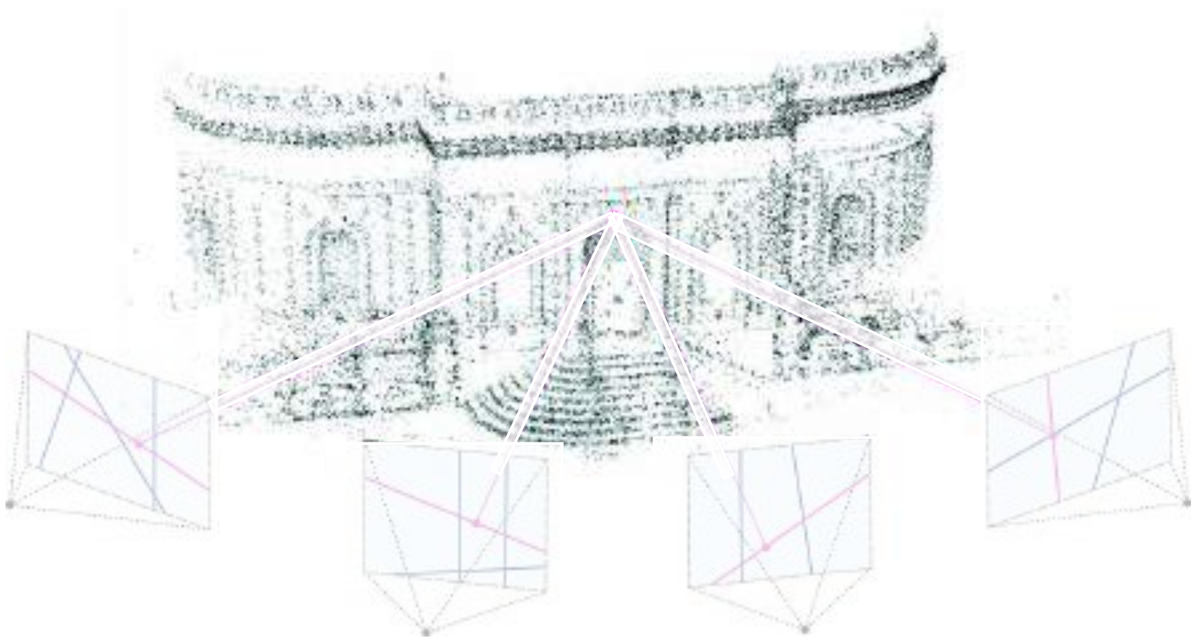
Multiple images

Output:

Reconstruct camera poses

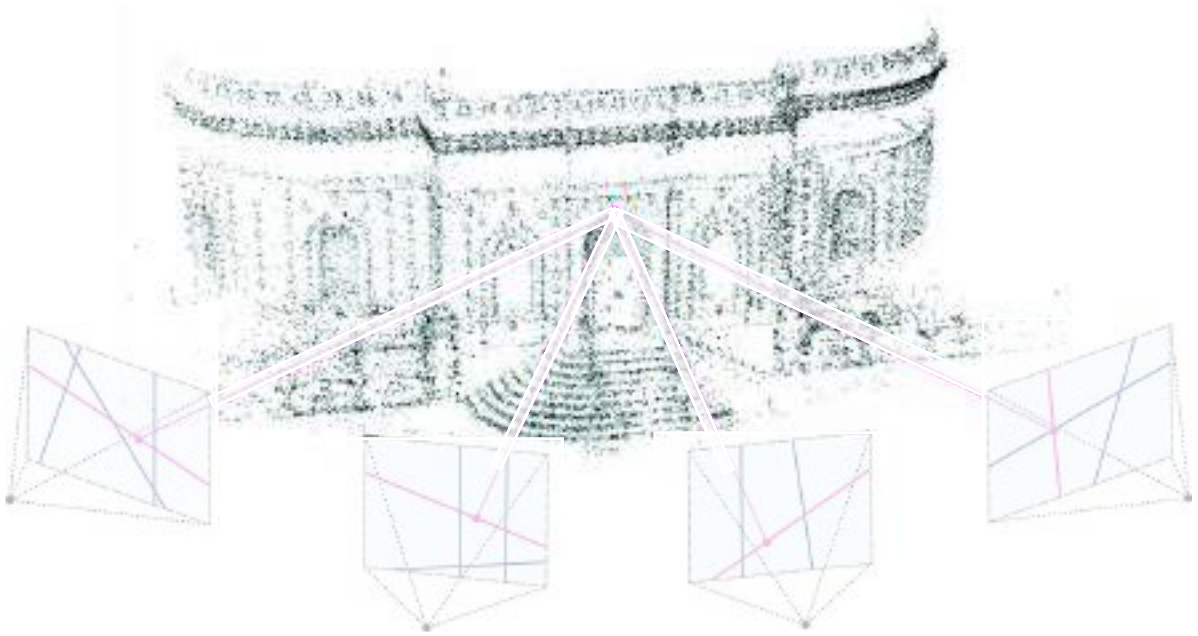
Side effect:

3D sparse point cloud



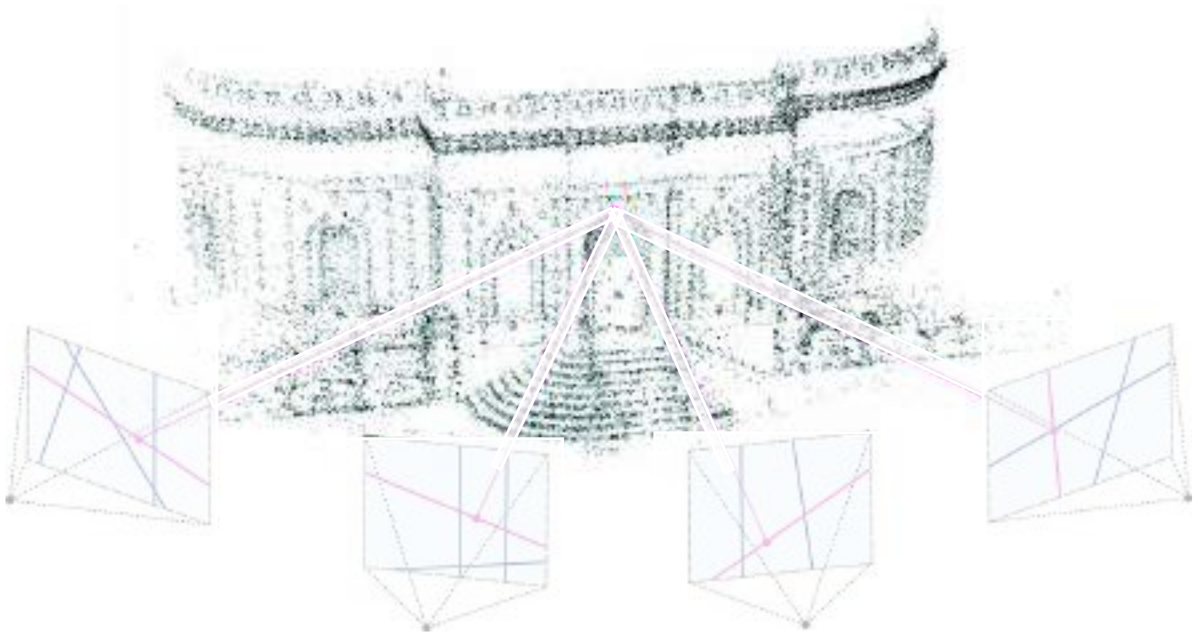
Structure From Motion (SfM)

Good for initialization



Structure From Motion (SfM)

Good for initialization

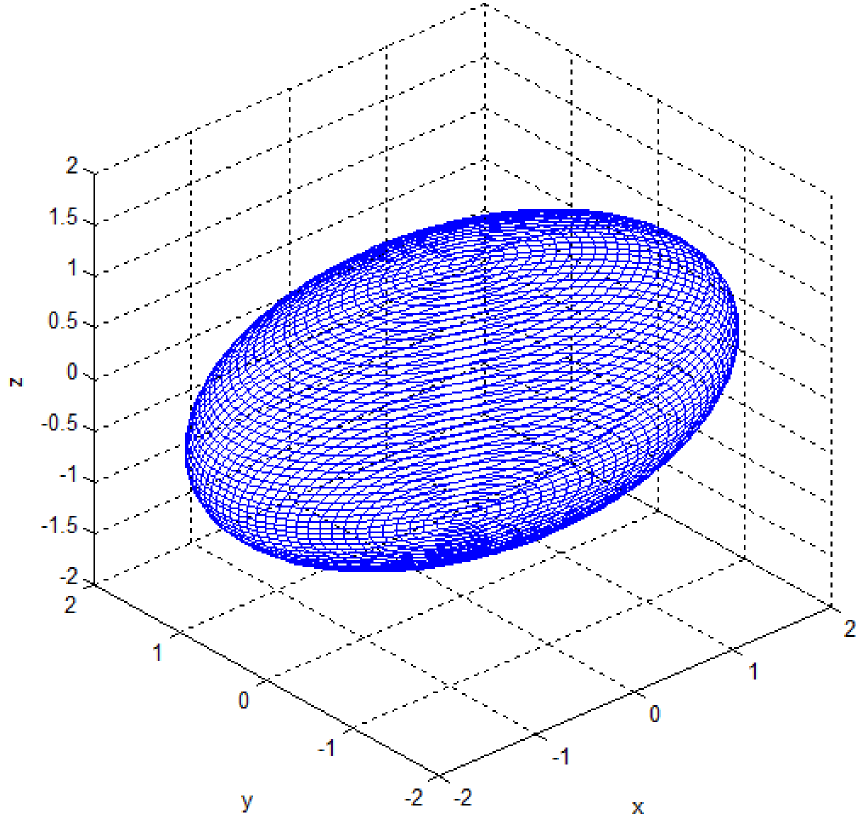


If no point cloud:

- Random initialization
- Worse loss values

What's a 3D gaussian?

Gaussian resembling an ellipsoid

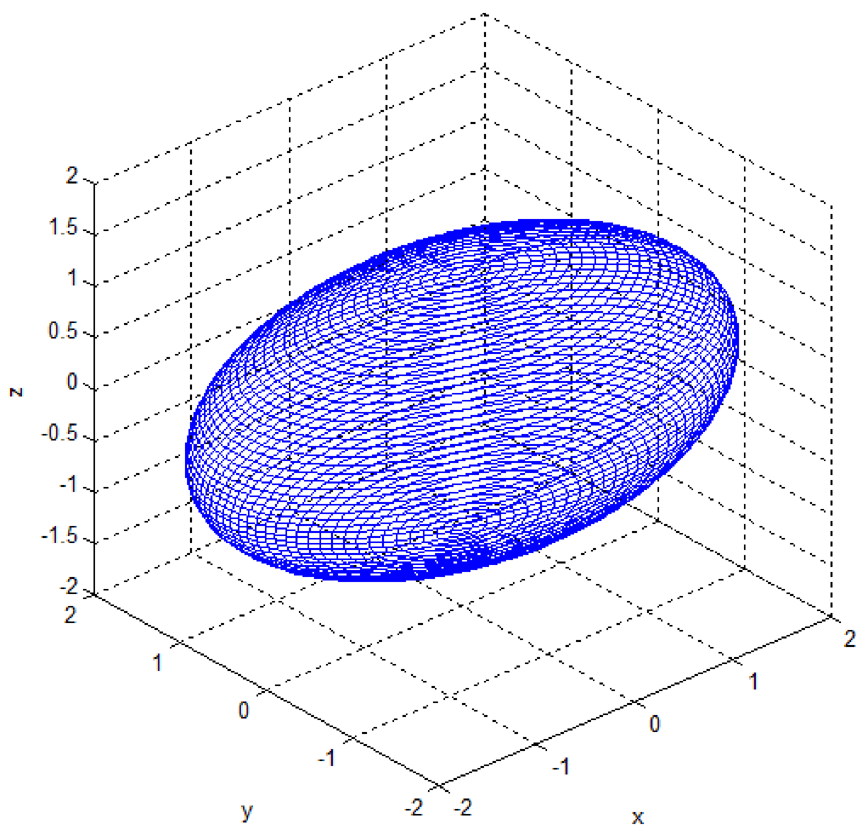


What’s a 3D gaussian?

Gaussian resembling an ellipsoid

Defined by

Position: X, Y, Z



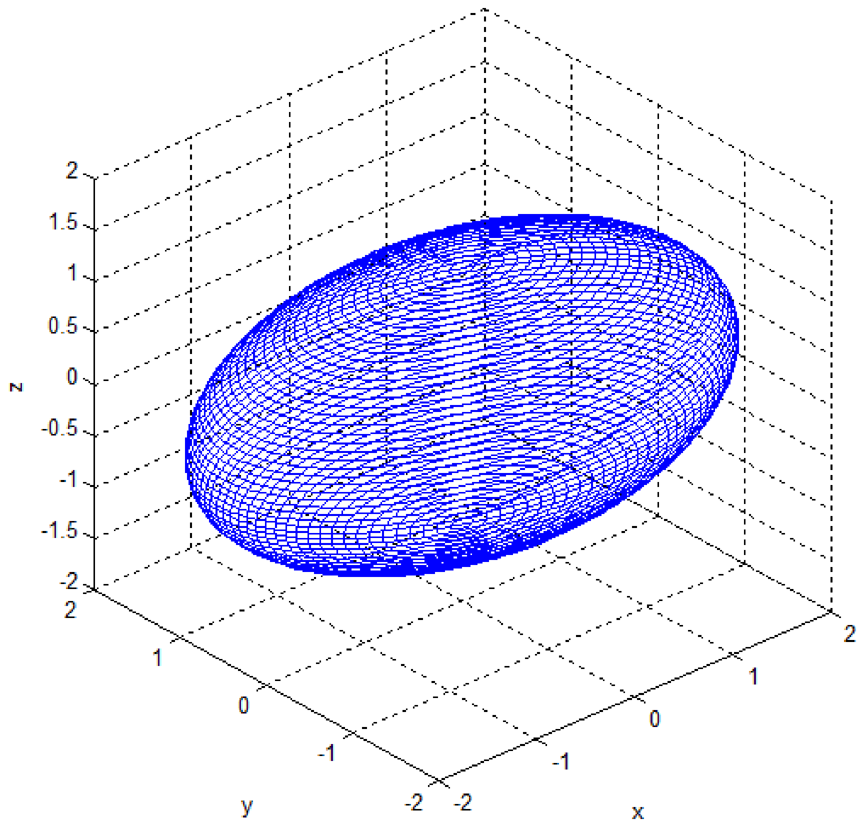
What’s a 3D gaussian?

Gaussian resembling an ellipsoid

Defined by

Position: X, Y, Z

Covariance: 3x3 matrix

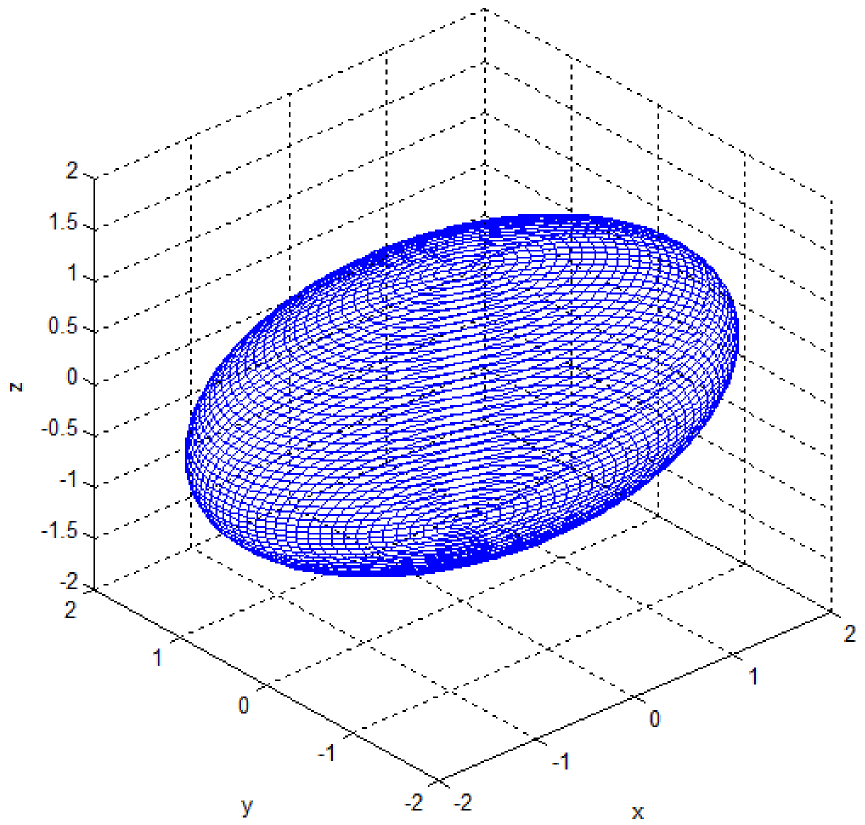


What's a 3D gaussian?

Gaussian resembling an ellipsoid

Defined by

- Position: X, Y, Z
- Covariance: 3x3 matrix
- Color: RGB

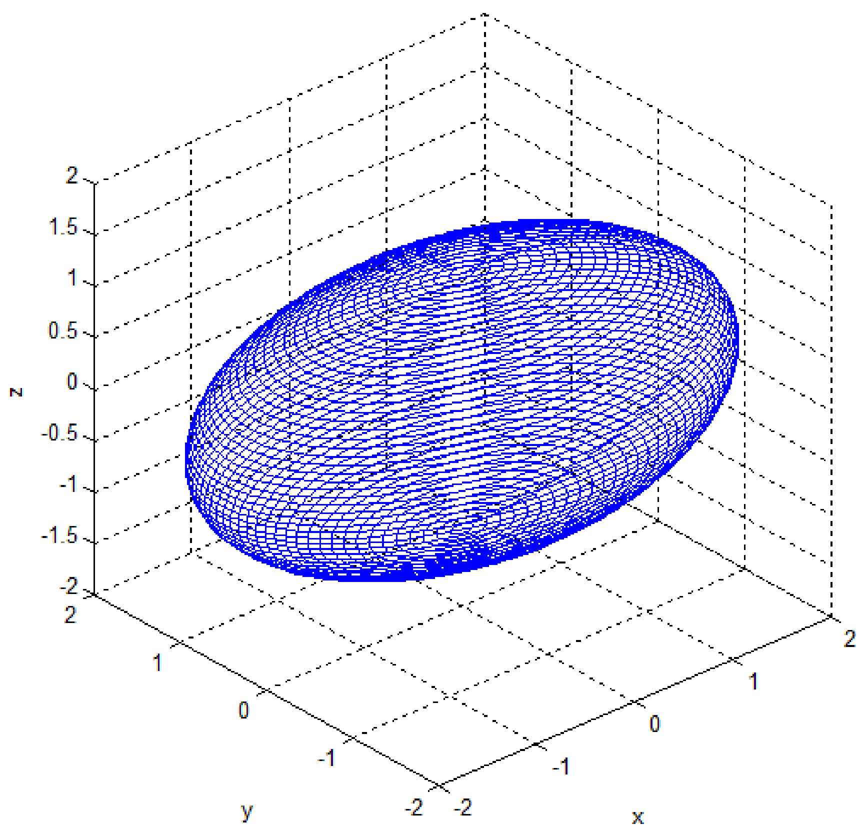


What's a 3D gaussian?

Gaussian resembling an ellipsoid

Defined by

- Position: X, Y, Z
- Covariance: 3x3 matrix
- Color: RGB
- Alpha transparency: α



Why 3D gaussian?

- Represent limited area in space

Why 3D gaussian?

- Represent limited area in space

- Theoretically infinite extent
  Defined everywhere

Why 3D gaussian?

- Represent limited area in space

- Theoretically infinite extent
  Defined everywhere

  Good for optimization

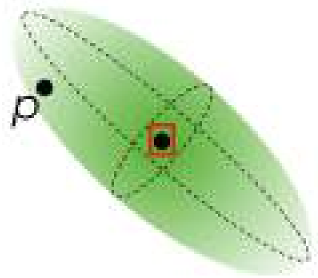
Why 3D gaussian?

- Represent limited area in space

- Theoretically infinite extent
  Defined everywhere

  Good for optimization

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



Initialization

Get point cloud / random init

Initialization

Get point cloud / random init

For every point

Place a 3D gaussian centered on it

Initialization

Get point cloud / random init

For every point

Place a 3D gaussian centered on it

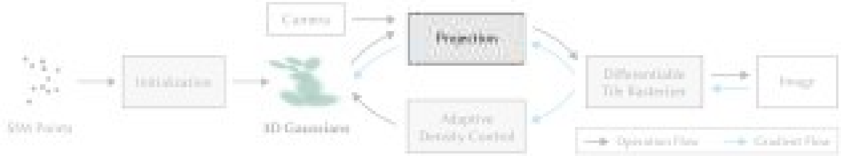
Only at initialization, isotropic gaussians

Initialization

Initialization

Initialization

Projection



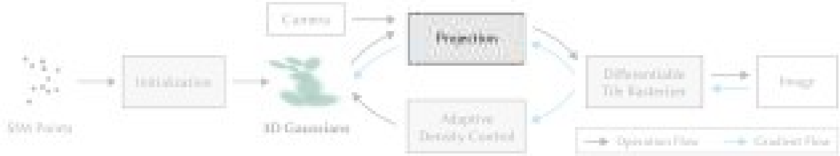
Set of 3D gaussian

Projection

Set of 3D gaussian

Project mean and covariance into 2D plane..

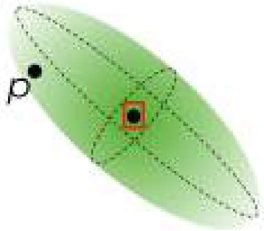
Projection



Set of 3D gaussian

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

Project mean and covariance into 2D plane..



..easy to compute impact to a certain pixel

Rendering: NeRF?

Rendering: NeRF?

Output from points on same ray

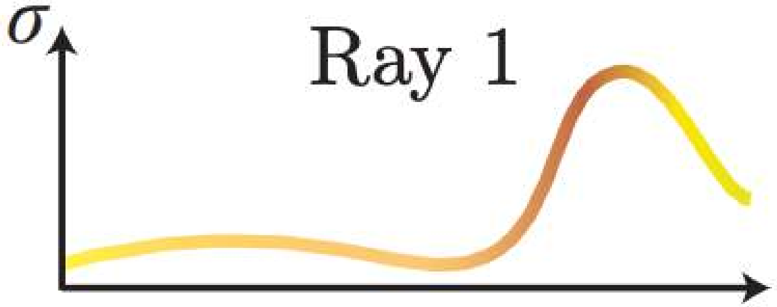
Suppose we have many

Rendering: NeRF?

Output from points on same ray

Suppose we have many

We can produce:

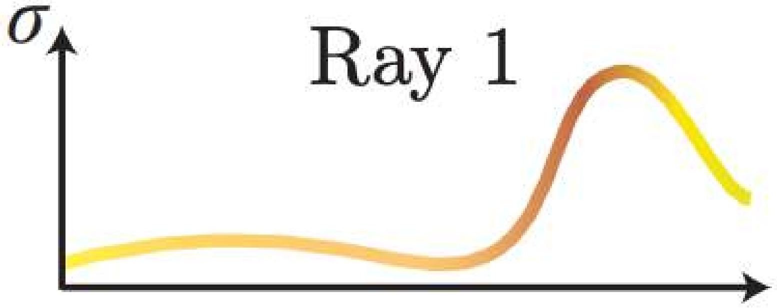


Rendering: NeRF?

Output from points on same ray

Suppose we have many

We can produce:



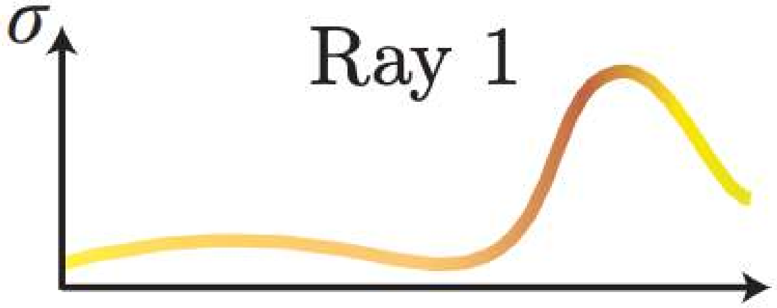
Aim: give color to whole ray

Rendering: NeRF?

Output from points on same ray

Suppose we have many

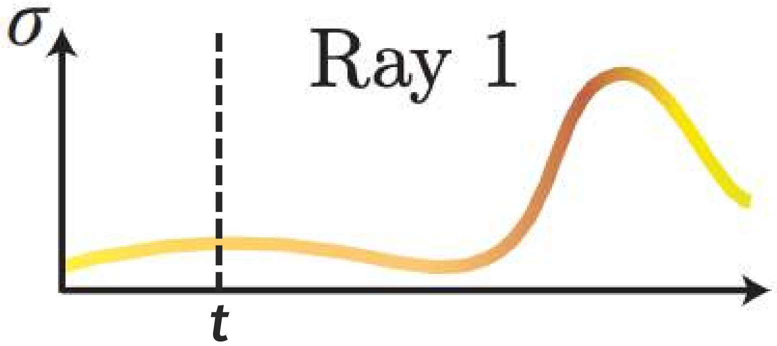
We can produce:



Aim: give color to whole ray
Why? determine pixel color associated to ray

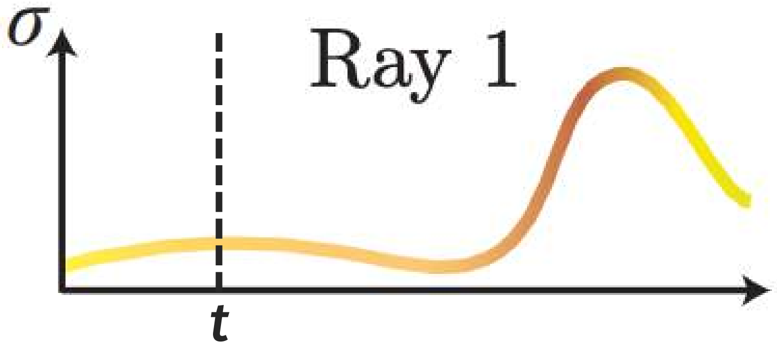
Rendering: NeRF?

Point t color



Rendering: NeRF?

Point t color

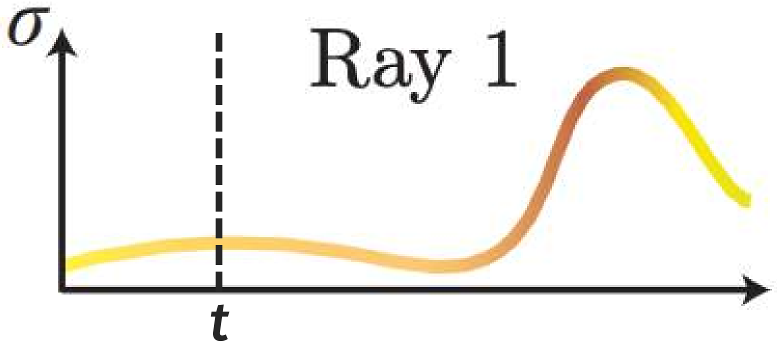


$$\mathbf{c}(\mathbf{r}(t), \mathbf{d})$$

Rendering: NeRF?

Point t color

Weighted on density (glass)



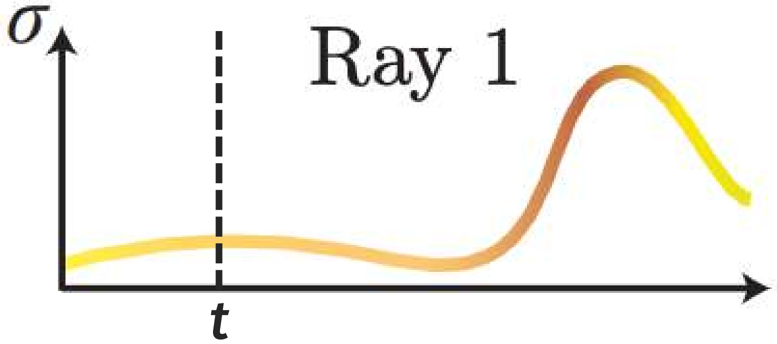
$$\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})$$

Rendering: NeRF?

Point t color

Weighted on density (glass)

Weighted on light already stopped



$$T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})$$

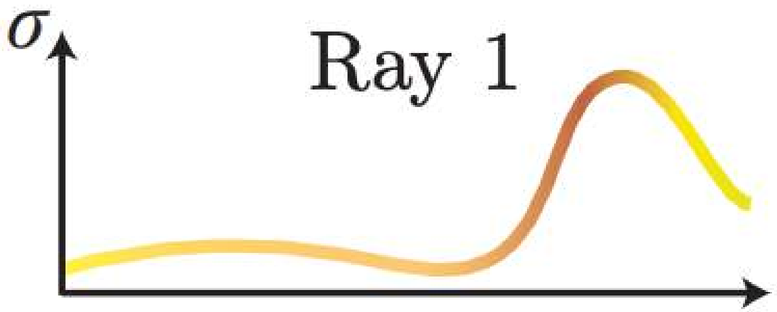
Rendering: NeRF?

Point *t* color

Weighted on density (glass)

Weighted on light already stopped

Sum of all points along ray



$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

Rendering: NeRF?

$$T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds \right)$$

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

Estimating the integral

Only 64 points per ray

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

Estimating the integral

Only 64 points per ray

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

Integral finite approximation:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} c_i(1 - \exp(-\sigma_i\delta_i))T_i =$$

$$\delta_i = t_{i+1} - t_i$$

Estimating the integral

Only 64 points per ray

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

Integral finite approximation:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} c_i(1 - \exp(-\sigma_i\delta_i))T_i = \sum_{i=1}^{N} c_i(1 - \exp(-\sigma_i\delta_i)) \exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j) =$$

$$\delta_i = t_{i+1} - t_i$$

Estimating the integral

Only 64 points per ray

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

Integral finite approximation:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} c_i(1 - \exp(-\sigma_i\delta_i))T_i = \sum_{i=1}^{N} c_i(1 - \exp(-\sigma_i\delta_i)) \exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j) =$$

$$= \sum_{i=1}^{N} c_i \underbrace{(1 - \exp(-\sigma_i\delta_i))}_{\alpha_i} \prod_{j=1}^{i-1} \underbrace{\exp(-\sigma_j\delta_j)}_{1-\alpha_j} =$$

$$\delta_i = t_{i+1} - t_i$$

Estimating the integral

Only 64 points per ray

$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt$$

Integral finite approximation:

$$\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} c_i(1 - \exp(-\sigma_i\delta_i))T_i = \sum_{i=1}^{N} c_i(1 - \exp(-\sigma_i\delta_i)) \exp(-\sum_{j=1}^{i-1} \sigma_j\delta_j) =$$

$$= \sum_{i=1}^{N} c_i \underbrace{(1 - \exp(-\sigma_i\delta_i))}_{\alpha_i} \prod_{j=1}^{i-1} \underbrace{\exp(-\sigma_j\delta_j)}_{1-\alpha_j} = \sum_{i=1}^{N} c_i\alpha_i \underbrace{\prod_{j=1}^{i-1}(1 - \alpha_j)}_{transmittance} \qquad \delta_i = t_{i+1} - t_i$$

Comparison

$$\sum_{i=1}^{N} c_i \alpha_i \underbrace{\prod_{j=1}^{i-1} (1 - \alpha_j)}_{transmittance}$$

NeRF

Comparison

$$\sum_{i=1}^{N} c_i \alpha_i \underbrace{\prod_{j=1}^{i-1} (1 - \alpha_j)}_{transmittance}$$

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

NeRF **Gaussian Splatting**

Efficiency

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

Efficiency

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

f_i^{2D} **2D Gaussian projection**

Efficiency

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

f_i^{2D} 2D Gaussian projection

Faster?

- No NN inference for every point   #pixels · #points

Efficiency

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

f_i^{2D} 2D Gaussian projection

Faster?

- No NN inference for every point #pixels · #points

- Every gaussian, one projection across pixels

Sorting algorithm

Sort gaussian by depth:

determine transmittance

Sorting algorithm

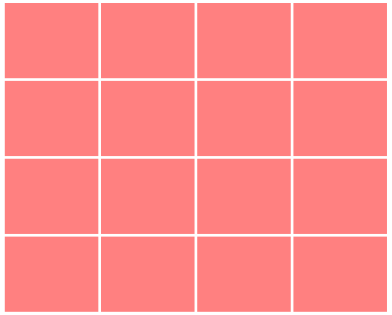
Sort gaussian by depth:

determine transmittance

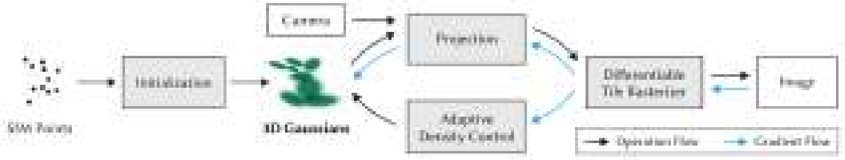
2D view:

tiled in 4x4 grid

parallelization

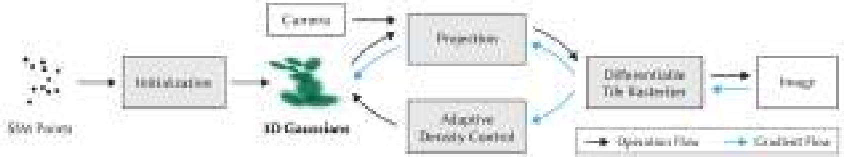


How does it look like?

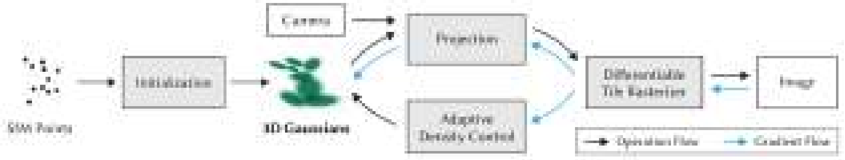


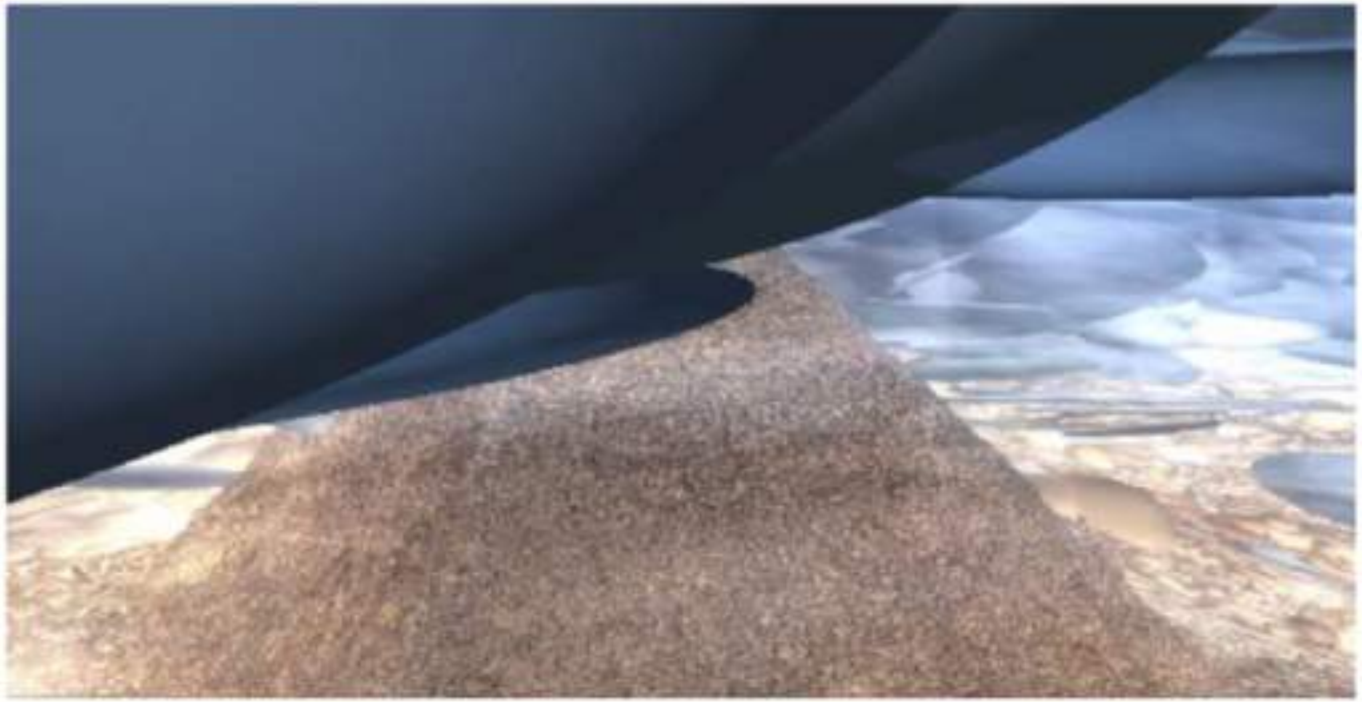
How does it look like?

How does it look like?



How does it look like?

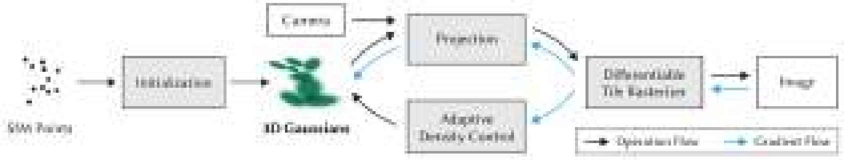




$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}$$

$$\lambda = 0.2$$

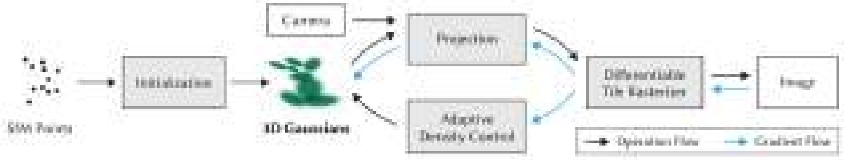
How does it look like?



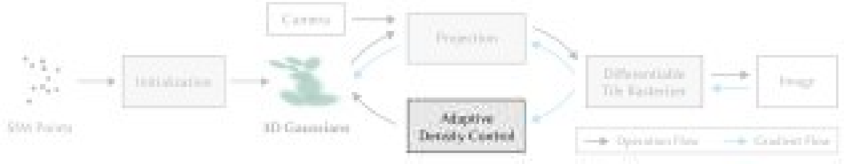
$$\mathcal{L} = (1 - \lambda)\mathcal{L}_1 + \lambda\mathcal{L}_{\text{D-SSIM}}$$

$$\lambda = 0.2$$

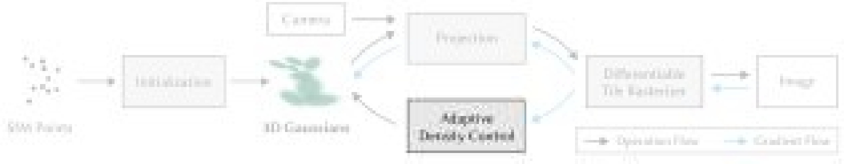
Flexibility?



Flexibility?

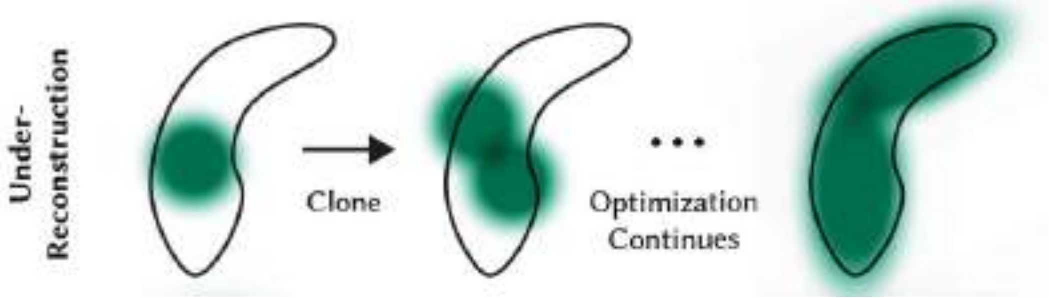


Flexibility?

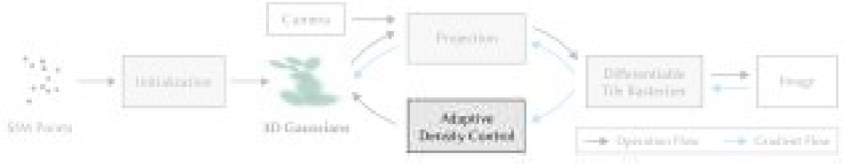


Every 100 iterations:

Under-Reconstruction check



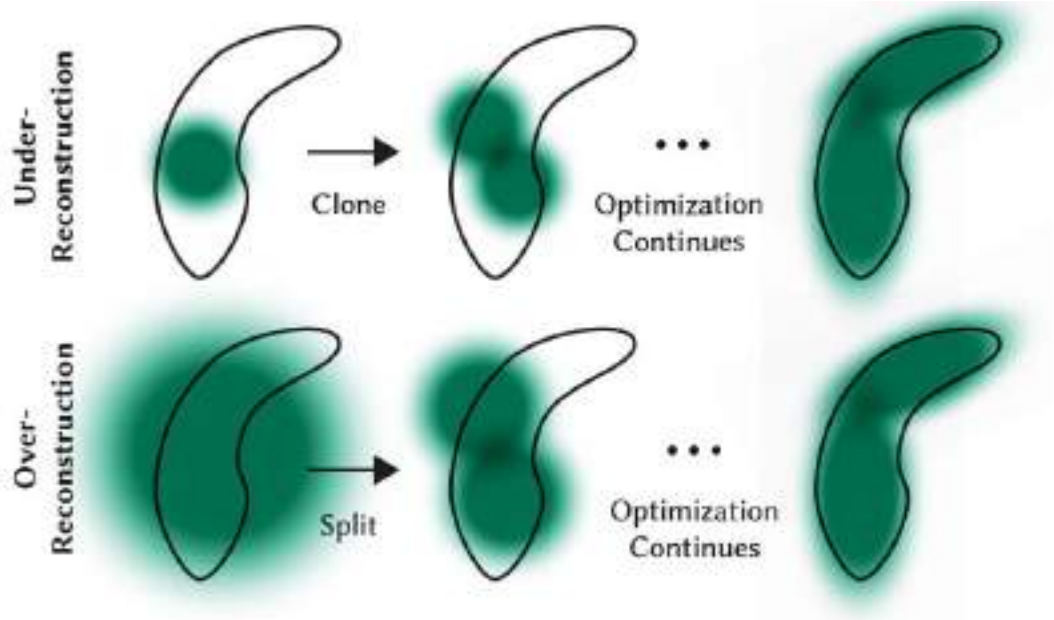
Flexibility?



Every 100 iterations:

Under-Reconstruction check

Over-Reconstruction check



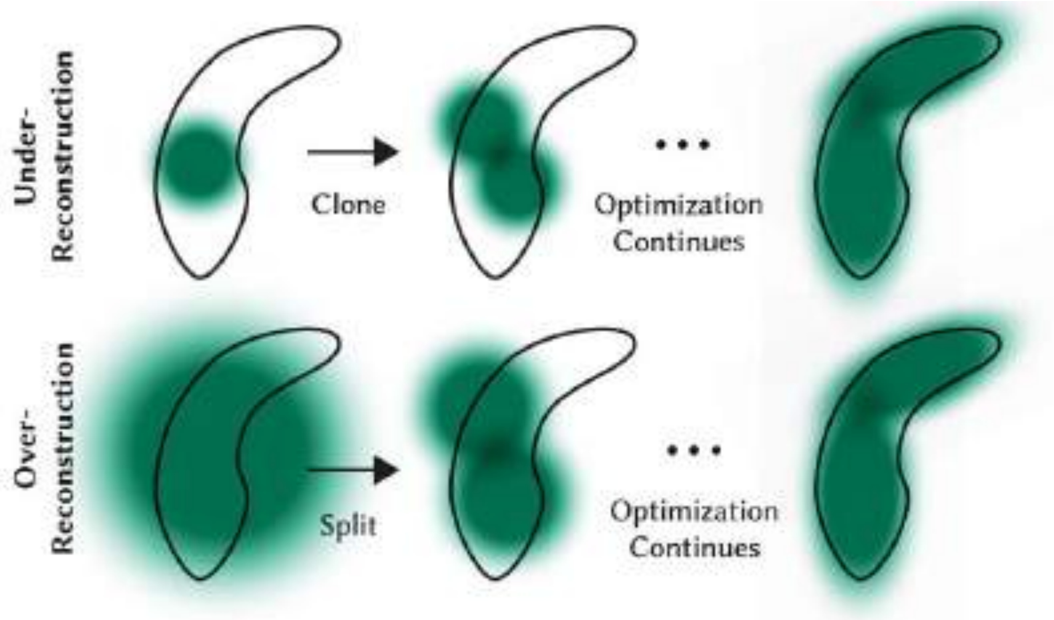
Flexibility?

Every 100 iterations:

Under-Reconstruction check

Over-Reconstruction check

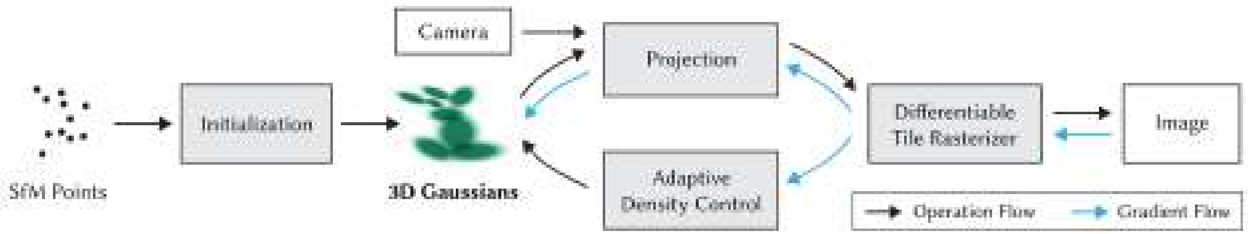
Alpha check



Recap

Algorithm 1 Optimization and Densification
w, h: width and height of the training images

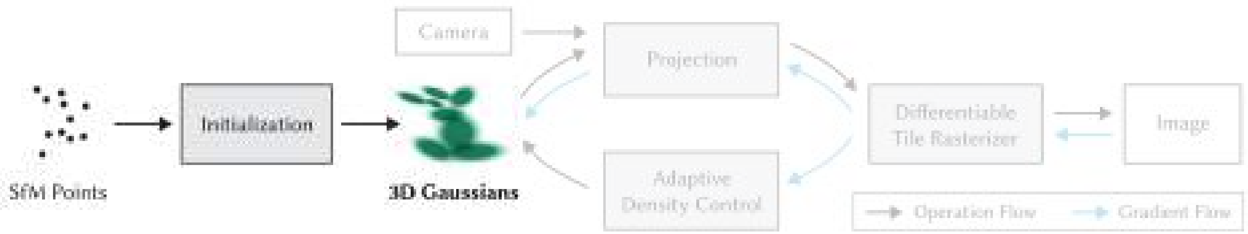
$M \leftarrow$ SfM Points ▷ Positions
$S, C, A \leftarrow$ InitAttributes() ▷ Covariances, Colors, Opacities
$i \leftarrow 0$ ▷ Iteration Count
while not converged **do**
  $V, \hat{I} \leftarrow$ SampleTrainingView() ▷ Camera V and Image
  $I \leftarrow$ Rasterize(M, S, C, A, V) ▷ Alg. 2
  $L \leftarrow Loss(I, \hat{I})$ ▷ Loss
  $M, S, C, A \leftarrow$ Adam(∇L) ▷ Backprop & Step
  if IsRefinementIteration(i) **then**
    for all Gaussians (μ, Σ, c, α) **in** (M, S, C, A) **do**
      if $\alpha < \epsilon$ or IsTooLarge(μ, Σ) **then** ▷ Pruning
        RemoveGaussian()
      end if
      if $\nabla_p L > \tau_p$ **then** ▷ Densification
        if $\|S\| > \tau_S$ **then** ▷ Over-reconstruction
          SplitGaussian(μ, Σ, c, α)
        else ▷ Under-reconstruction
          CloneGaussian(μ, Σ, c, α)
        end if
      end if
    end for
  end if
  $i \leftarrow i + 1$
end while



Recap

Algorithm 1 Optimization and Densification
w, h: width and height of the training images

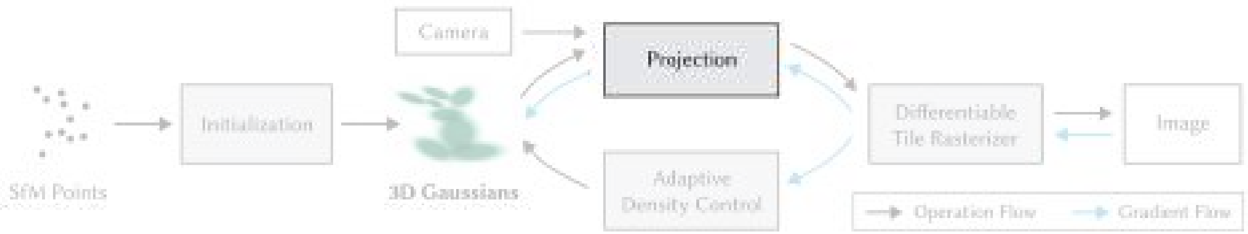
$M \leftarrow$ SfM Points ▷ Positions
$S, C, A \leftarrow$ InitAttributes() ▷ Covariances, Colors, Opacities
$i \leftarrow 0$ ▷ Iteration Count
while not converged **do**
  $V, \hat{I} \leftarrow$ SampleTrainingView() ▷ Camera V and Image
  $I \leftarrow$ Rasterize(M, S, C, A, V) ▷ Alg. 2
  $L \leftarrow Loss(I, \hat{I})$ ▷ Loss
  $M, S, C, A \leftarrow$ Adam(∇L) ▷ Backprop & Step
  if IsRefinementIteration(i) **then**
    for all Gaussians (μ, Σ, c, α) **in** (M, S, C, A) **do**
      if $\alpha < \epsilon$ or IsTooLarge(μ, Σ) **then** ▷ Pruning
        RemoveGaussian()
      end if
      if $\nabla_p L > \tau_p$ **then** ▷ Densification
        if $\|S\| > \tau_S$ **then** ▷ Over-reconstruction
          SplitGaussian(μ, Σ, c, α)
        else ▷ Under-reconstruction
          CloneGaussian(μ, Σ, c, α)
        end if
      end if
    end for
  end if
  $i \leftarrow i + 1$
end while



Recap

Algorithm 1 Optimization and Densification
w, h: width and height of the training images

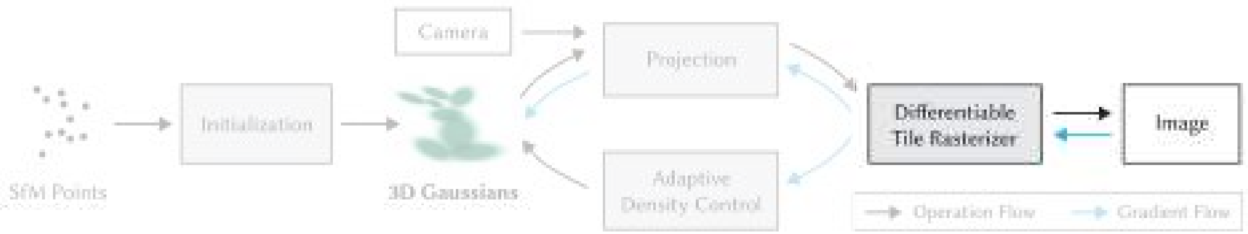
```
M ← SfM Points                                   ▷ Positions
S, C, A ← InitAttributes()                       ▷ Covariances, Colors, Opacities
i ← 0                                            ▷ Iteration Count
while not converged do
    V, Î ← SampleTrainingView()                  ▷ Camera V and 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
```



Recap

Algorithm 1 Optimization and Densification
w, h: width and height of the training images

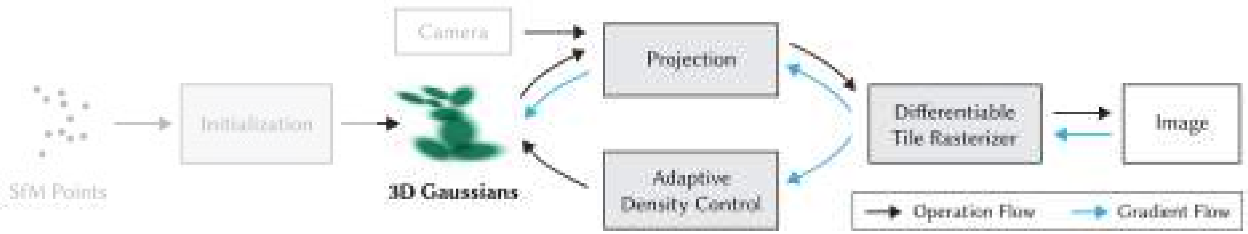
```
M ← SfM Points                                  ▷ Positions
S, C, A ← InitAttributes()   ▷ Covariances, Colors, Opacities
i ← 0                                      ▷ Iteration Count
while not converged do
    V, Î ← SampleTrainingView()          ▷ Camera V and Image
    I ← Rasterize(M, S, C, A, 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
```



Recap

Algorithm 1 Optimization and Densification
w, h: width and height of the training images

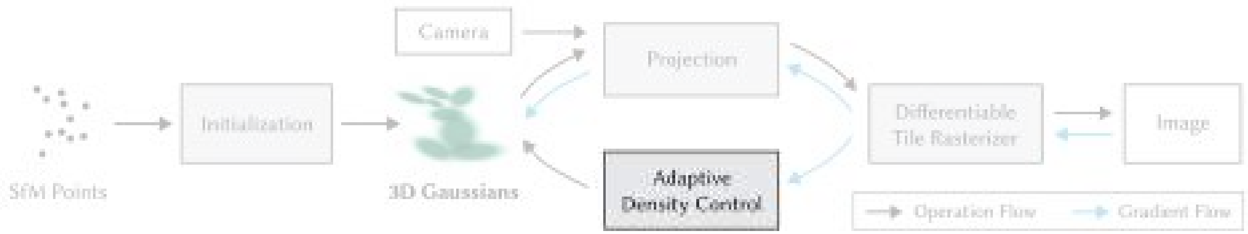
```
M ← SfM Points                                  ▷ Positions
S, C, A ← InitAttributes()        ▷ Covariances, Colors, Opacities
i ← 0                                           ▷ Iteration Count
while not converged do
    V, Î ← SampleTrainingView()             ▷ Camera V and Image
    I ← Rasterize(M, 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
```



Recap

Algorithm 1 Optimization and Densification
w, h: width and height of the training images

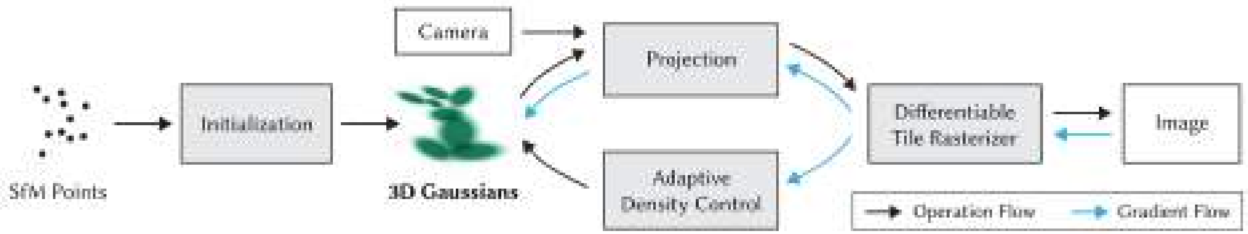
$M \leftarrow$ SfM Points ▷ Positions
$S, C, A \leftarrow$ InitAttributes() ▷ Covariances, Colors, Opacities
$i \leftarrow 0$ ▷ Iteration Count
while not converged **do**
  $V, \hat{I} \leftarrow$ SampleTrainingView() ▷ Camera V and Image
  $I \leftarrow$ Rasterize(M, S, C, A, V) ▷ Alg. 2
  $L \leftarrow Loss(I, \hat{I})$ ▷ Loss
  $M, S, C, A \leftarrow$ Adam(∇L) ▷ Backprop & Step
  if IsRefinementIteration(i) **then**
    for all Gaussians (μ, Σ, c, α) **in** (M, S, C, A) **do**
      if $\alpha < \epsilon$ or IsTooLarge(μ, Σ) **then** ▷ Pruning
        RemoveGaussian()
      end if
      if $\nabla_p L > \tau_p$ **then** ▷ Densification
        if $\|S\| > \tau_S$ **then** ▷ Over-reconstruction
          SplitGaussian(μ, Σ, c, α)
        else ▷ Under-reconstruction
          CloneGaussian(μ, Σ, c, α)
        end if
      end if
    end for
  end if
  $i \leftarrow i + 1$
end while



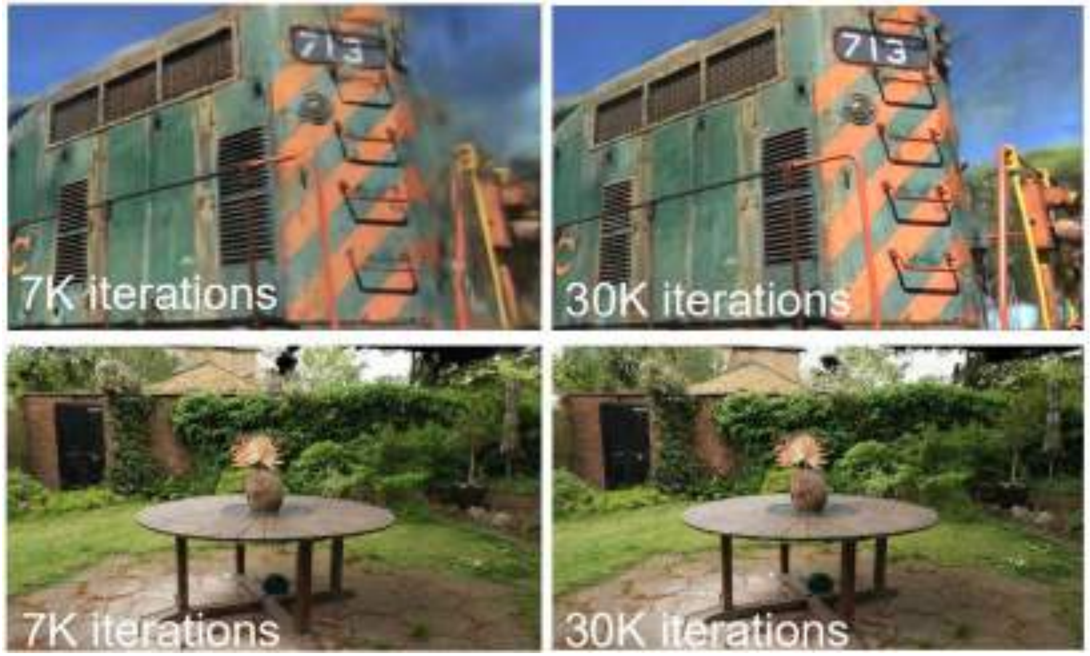
Recap

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
```



Quantitative results



Quantitative results

Dataset Method|Metric	Mip-NeRF360 $SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	Train	FPS	Mem
Plenoxels	0.626	23.08	0.463	25m49s	6.79	2.1GB
INGP-Base	0.671	25.30	0.371	5m37s	11.7	13MB
INGP-Big	0.699	25.59	0.331	7m30s	9.43	48MB
M-NeRF360	$0.792^{\dagger}$	$27.69^{\dagger}$	$0.237^{\dagger}$	48h	0.06	8.6MB
Ours-7K	0.770	25.60	0.279	6m25s	160	523MB
Ours-30K	0.815	27.21	0.214	41m33s	134	734MB

Quantitative results

Dataset	Tanks&Temples					
Method|Metric	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	Train	FPS	Mem
Plenoxels	0.719	21.08	0.379	25m5s	13.0	2.3GB
INGP-Base	0.723	21.72	0.330	5m26s	17.1	13MB
INGP-Big	0.745	21.92	0.305	6m59s	14.4	48MB
M-NeRF360	0.759	22.22	0.257	48h	0.14	8.6MB
Ours-7K	0.767	21.20	0.280	6m55s	197	270MB
Ours-30K	0.841	23.14	0.183	26m54s	154	411MB

Quantitative results

Dataset	Deep Blending					
Method|Metric	$SSIM^{\uparrow}$	$PSNR^{\uparrow}$	$LPIPS^{\downarrow}$	Train	FPS	Mem
Plenoxels	0.795	23.06	0.510	27m49s	11.2	2.7GB
INGP-Base	0.797	23.62	0.423	6m31s	3.26	13MB
INGP-Big	0.817	24.96	0.390	8m	2.79	48MB
M-NeRF360	0.901	29.40	0.245	48h	0.09	8.6MB
Ours-7K	0.875	27.78	0.317	4m35s	172	386MB
Ours-30K	0.903	29.41	0.243	36m2s	137	676MB

Qualitative results

Qualitative results

Qualitative results

UniGe | MaLGa

NeRF recap

Gaussian splatting

Conclusions

Conclusions

Gaussian splatting:

Conclusions

Gaussian splatting:

- Old meets new

Conclusions

Gaussian splatting:

- Old meets new

- SOTA results + Efficiency

Conclusions

Gaussian splatting:

- Old meets new
- SOTA results + Efficiency
- Interpretable

UniGe

MaLGa