KAIST [1] ETH zürich [2]

SurFhead

Affine Rig Blending for Geometrically Accurate 2D Gaussian Surfel Head Avatars

ICLR 2025

Jaeseong Lee[1*] Taewoong Kang[1*] Marcel C. Bühler[2] Min-Jung Kim[1]
Sungwon Hwang[1] Junha Hyung[1] Hyojin Jang[1] Jaegul Choo[1]

* Indicates equal contribution

summertight.github.io/SurFhead

ICLR
International Conference On Learning Representations

Teaser

Geometrical Head Avatars

High Fidelity Normals & Depths

Geometrical Head Avatars

1 Relighting

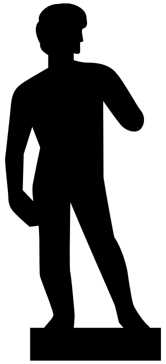
High Fidelity Normals & Depths

Geometrical Head Avatars

High Fidelity Normals & Depths

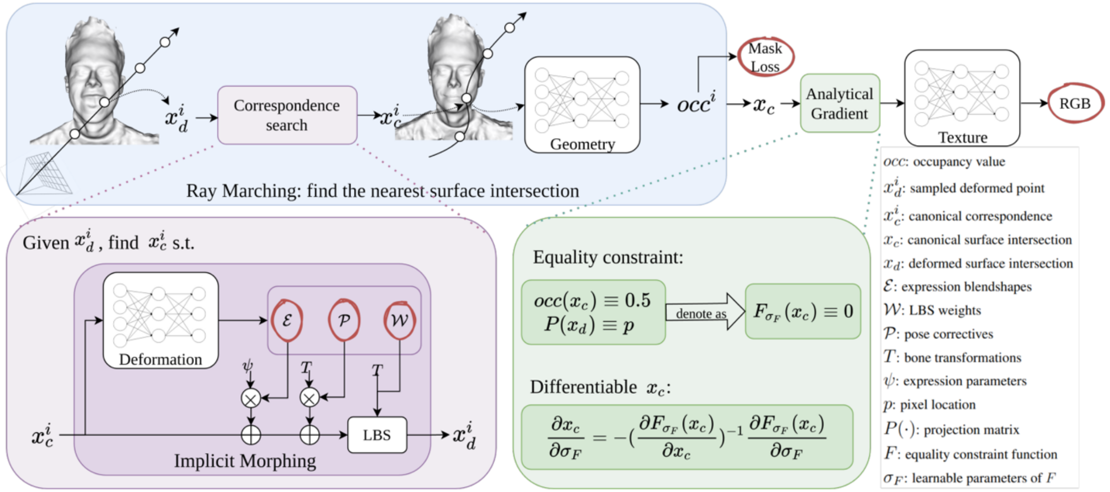
1 Relighting

2 Meshing

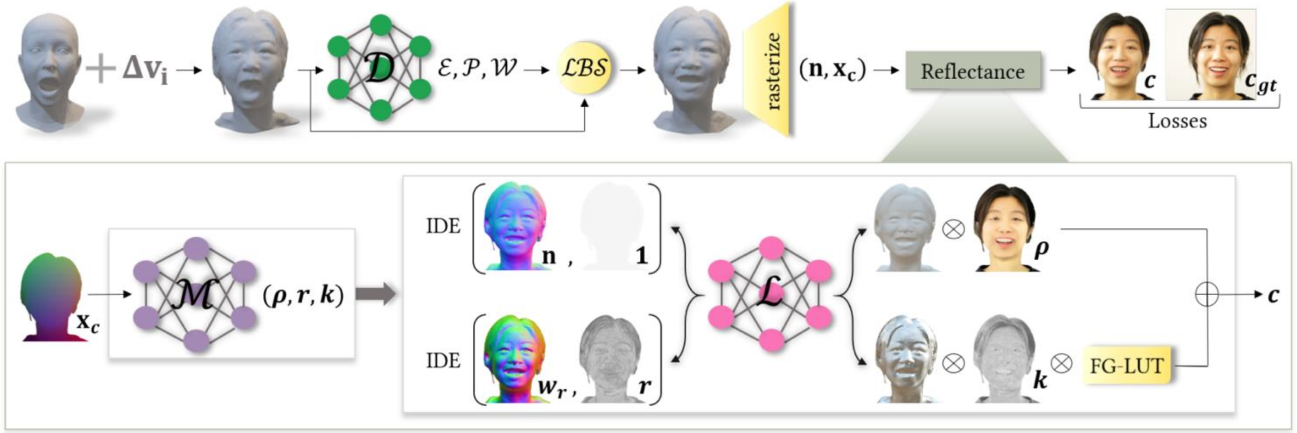


Geometrical Head Avatars – Previous art

IMAvatar (Zheng et al., CVPR 2022)

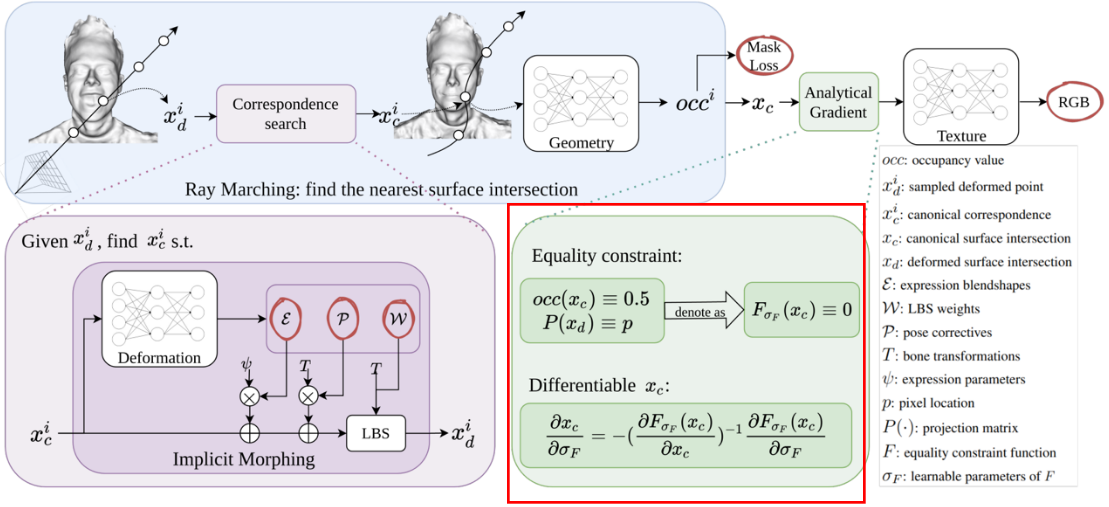


FLARE (Bharadwaj et al., TOG 2023)



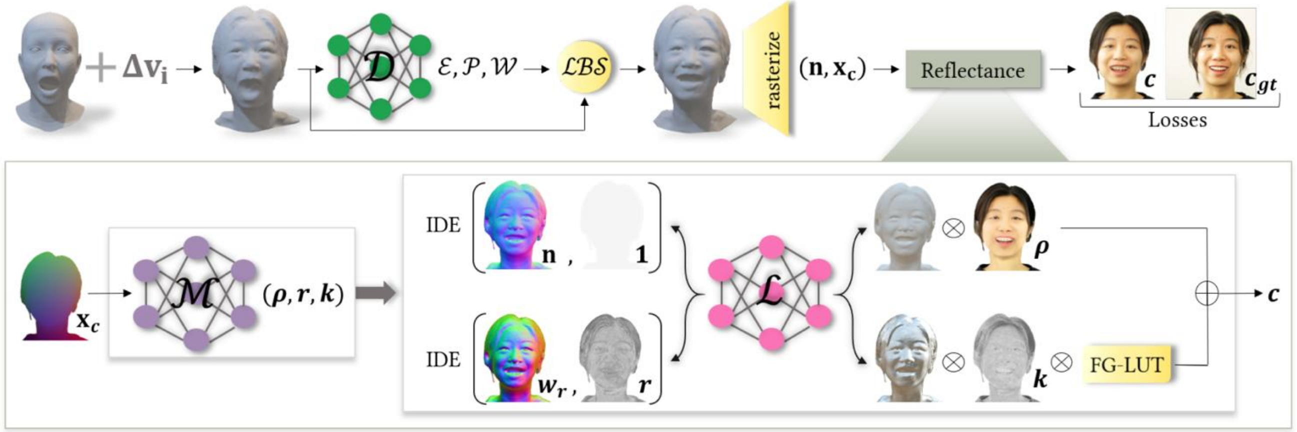
Geometrical Head Avatars – Previous art

IMAvatar (Zheng et al., CVPR 2022)



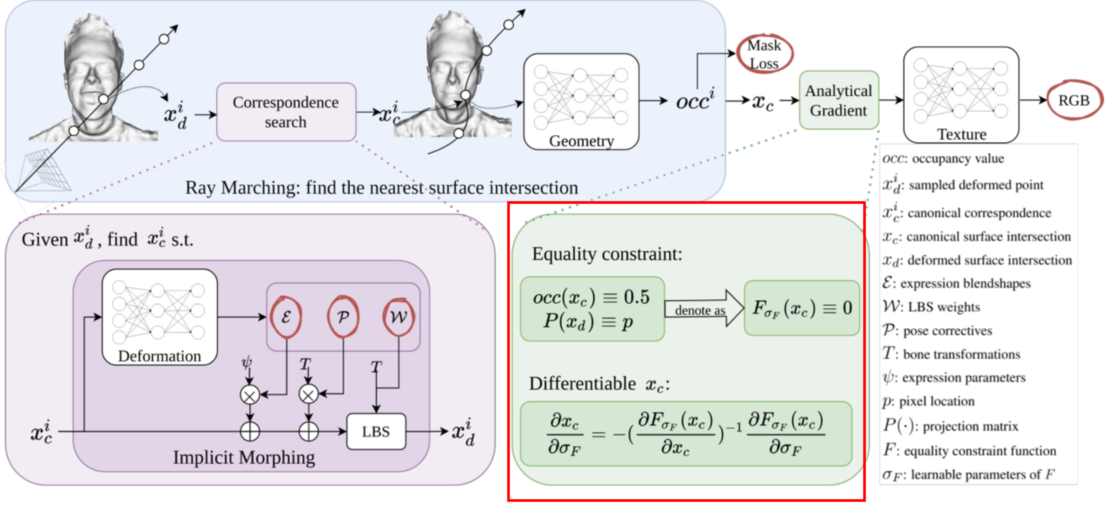
Exhaustive Numerical Surface Search

FLARE (Bharadwaj et al., TOG 2023)



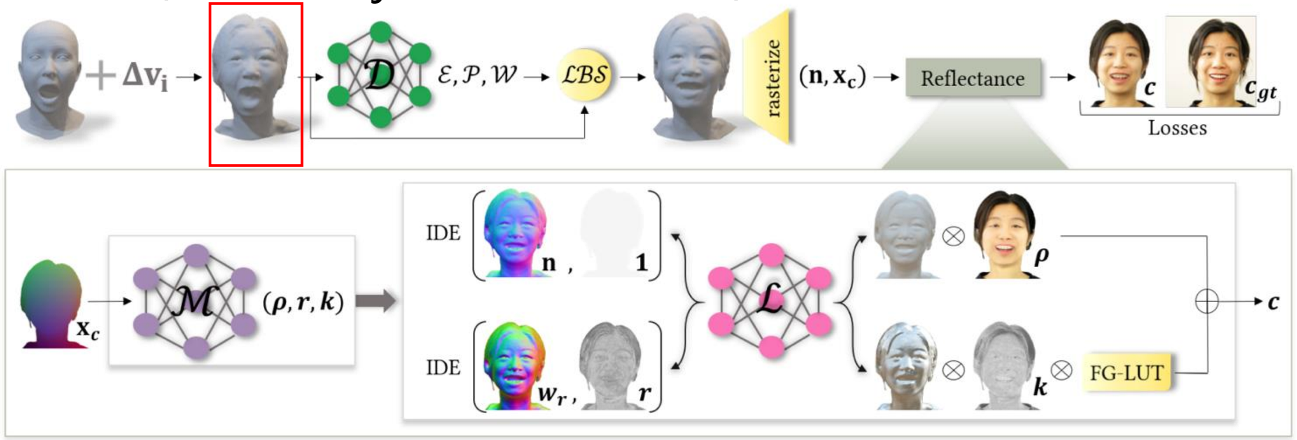
Geometrical Head Avatars – Previous art

IMAvatar (Zheng et al., CVPR 2022)



Exhaustive Numerical Surface Search

FLARE (Bharadwaj et al., TOG 2023)



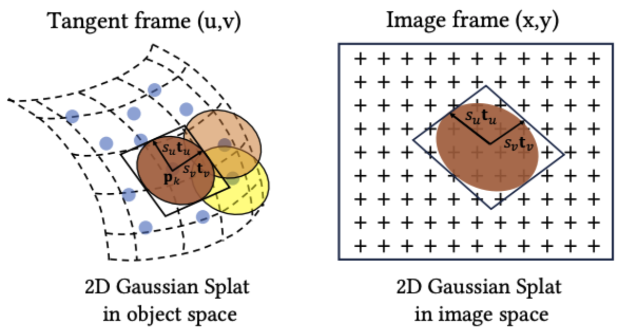
Unstable Training-time Remeshing

How to design ‘geometrical’ & ‘dynamic’ **Gaussian** head avatars?

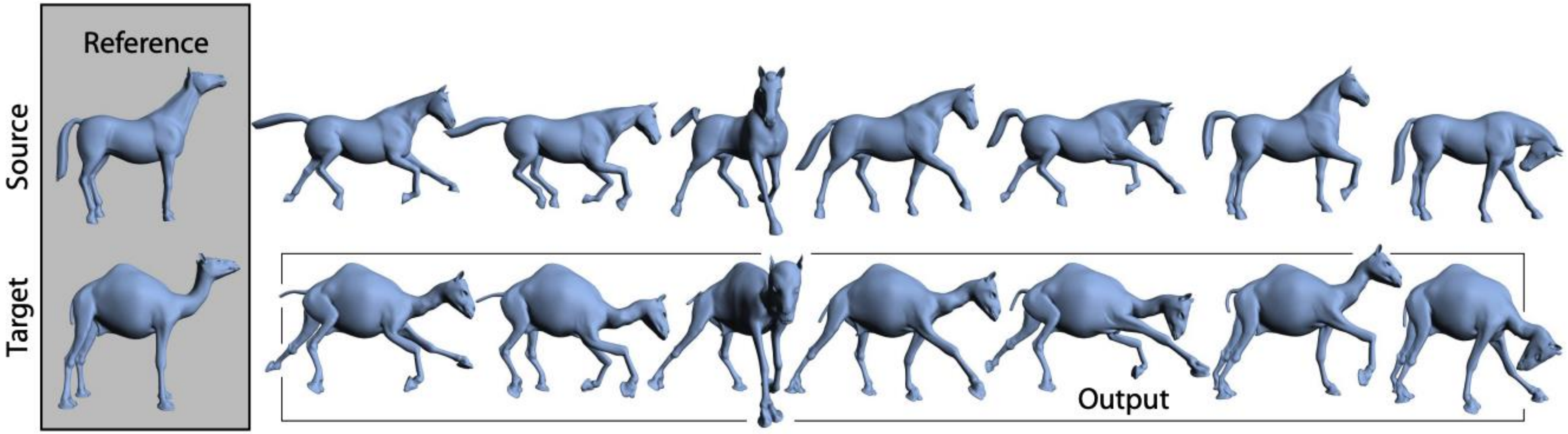
3DGS (Kerbl et al., SIGGRAPH 2023)

How to design ‘geometrical’ & ‘**dynamic**’ Gaussian head avatars?

To achieve accurate normal and depth**, we utilize 2D surfels**
Also accurate rigging states, inspired by DTF**, rigging Gaussians with Jacobian gradients.**



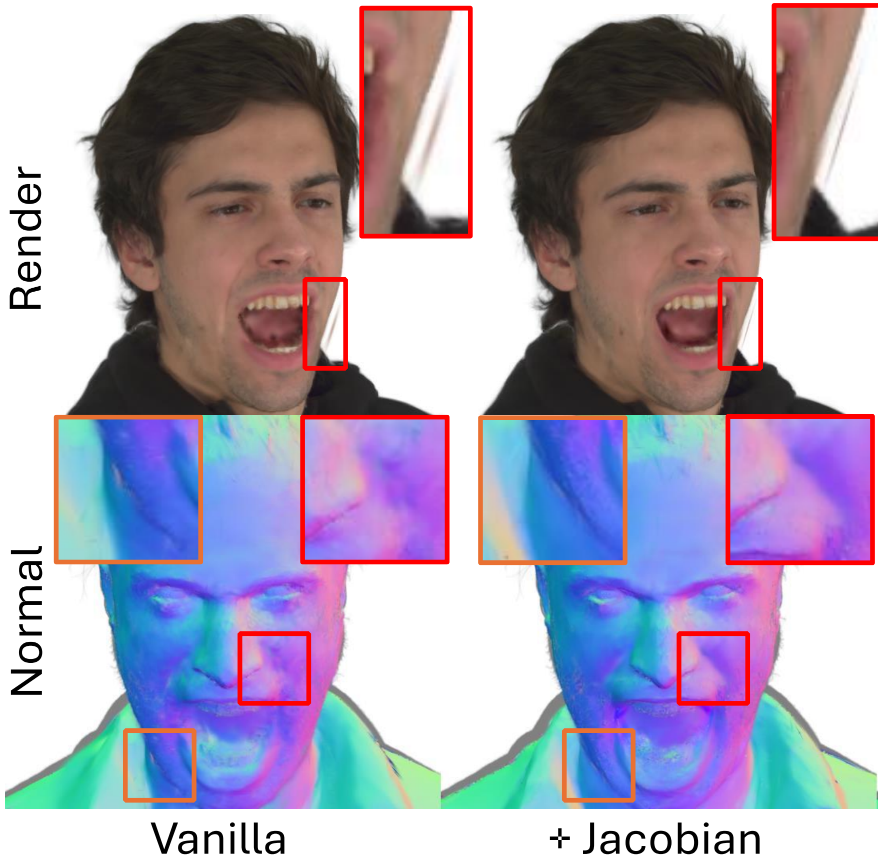
2DGS (SIGGRAPH 2024, Huang et al.)



DTF (SIGGRAPH 2000, Sumner and Popovic)

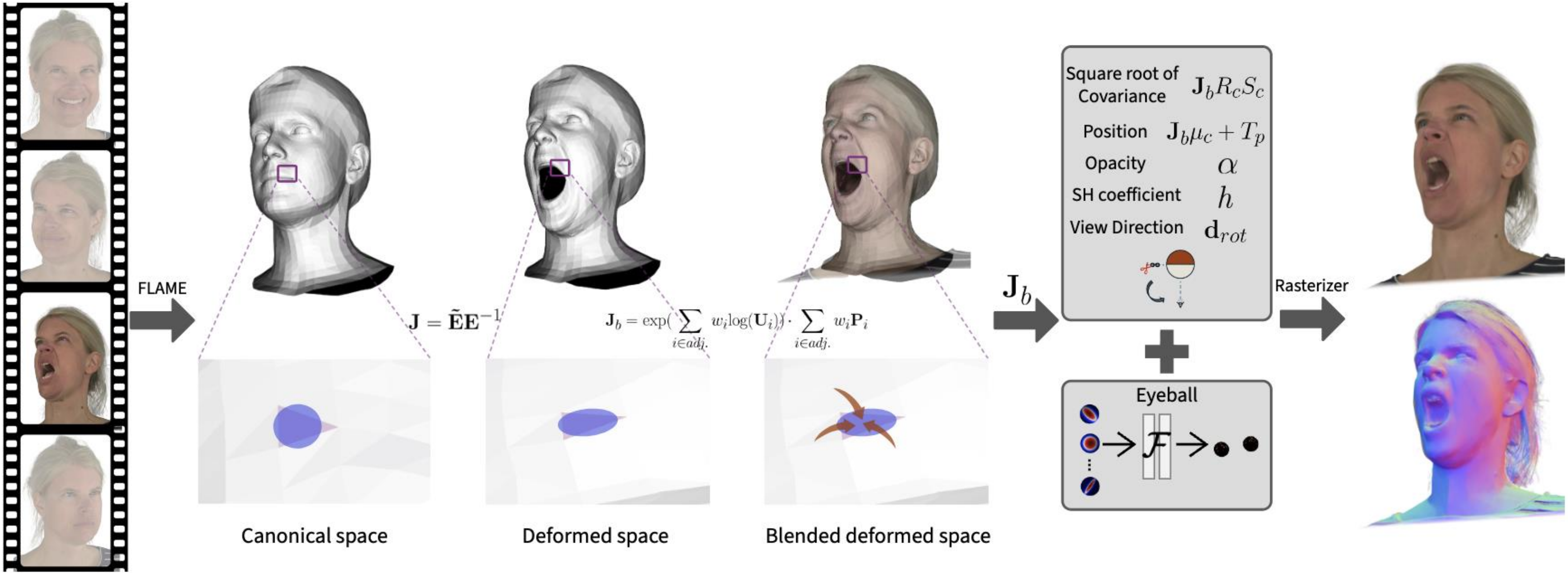
However,

Only using 2DGS and Jacobian Gradients lead to **floating artefacts** and **ambiguous normal.**



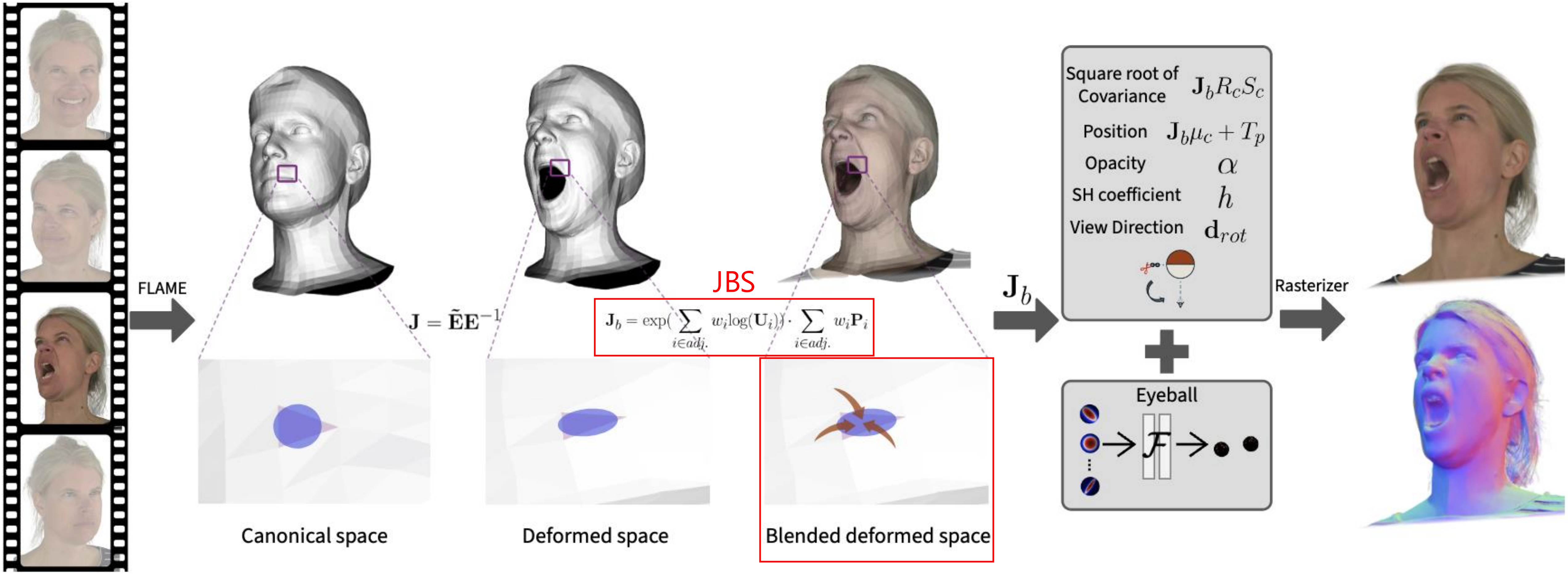
To handle these,

We propose Jacobian Blend Skinning (JBS).



To handle these,

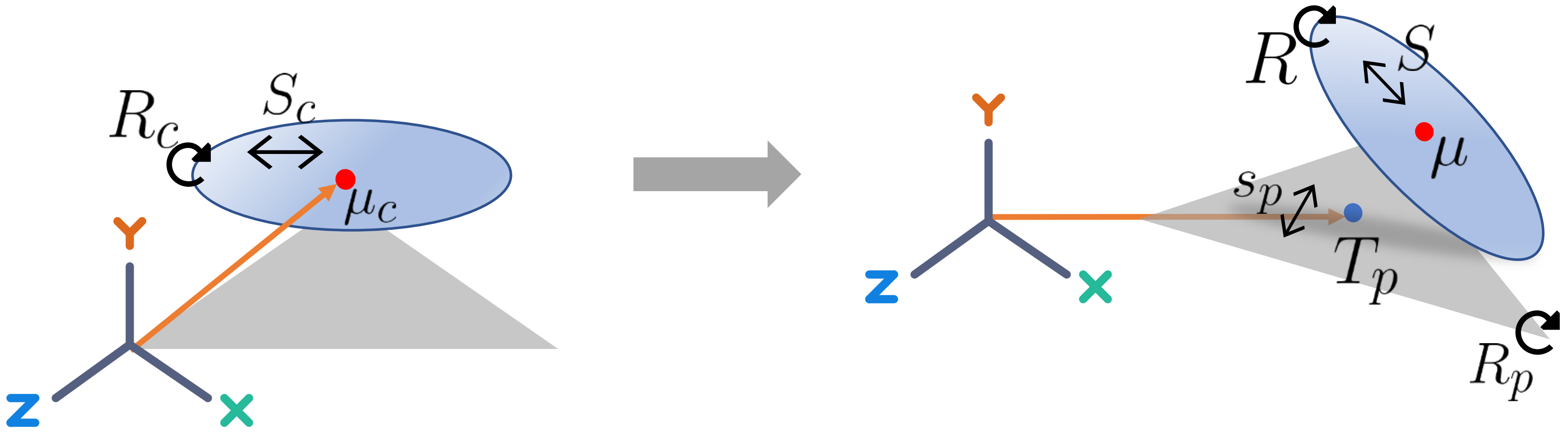
We propose Jacobian Blend Skinning (JBS).



Mesh Binding Inheritance

We follow 3DMFM binding inheritance strategy of GaussianAvatars (CVPR 2024, Qian et al.)

$$R = R_p R_c, \mu = s_p R_p \mu_c + T_p, S = s_p S_c$$



Canonical
Local Space

Deformed
Metric Space

Similarity to Jacobian Transforms

Following DTF, we replace the GaussianAvatars' similarity transforms to Jacobian transforms.

Similarity to Jacobian Transforms

Following DTF we replace the GaussianAvatars' similarity transforms to Jacobian transforms.

$$\underline{R = R_p R_c}, \mu = s_p R_p \mu_c + T_p, \underline{S = s_p S_c}$$

$$\Leftrightarrow$$

$$\Sigma^{1/2} = s_p R_p R_c S_c$$

Similarity to Jacobian Transforms

Following DTF (Sumner and Popovic), we replace the GaussianAvatars' similarity transforms to Jacobian transforms.

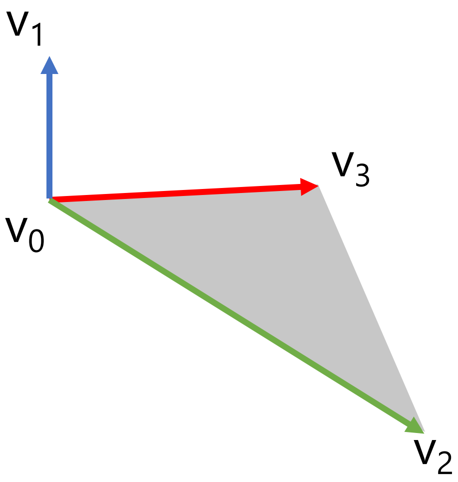
$$\underline{R = R_p R_c}, \mu = s_p R_p \mu_c + T_p, \underline{S = s_p S_c}$$

$$\Sigma^{1/2} = \boxed{s_p R_p} R_c S_c$$

$$\Sigma^{1/2} = \boxed{J} R_c S_c$$

Jacobian Deformation Transfer

Jacobian can represent stretch and shear transforms unlike similarity transforms.

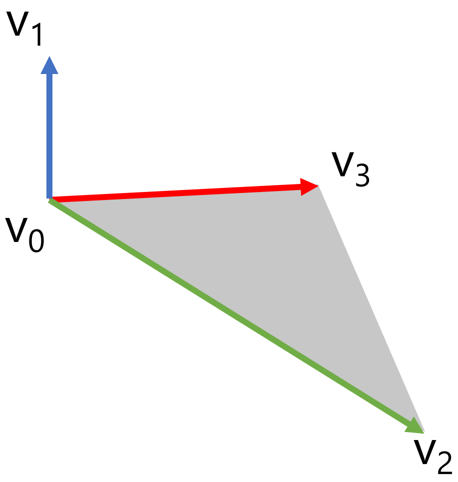


$E = [v_1 - v_0, v_2 - v_0, v_3 - v_0]$

Canonical
Metric Space

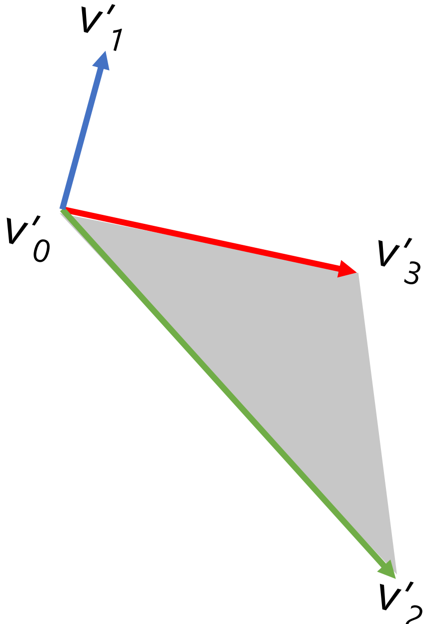
Jacobian Deformation Transfer

Jacobian can represent stretch and shear transforms unlike similarity transforms.



$E = [v_1 - v_0, v_2 - v_0, v_3 - v_0]$

Canonical
Metric Space

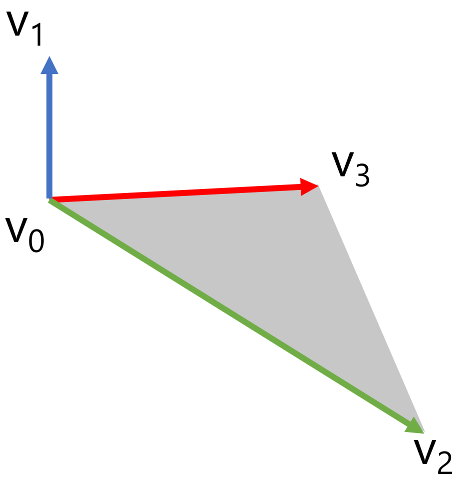


$E' = [v'_1 - v'_0, v'_2 - v'_0, v'_3 - v'_0]$

Deformed
Metric Space

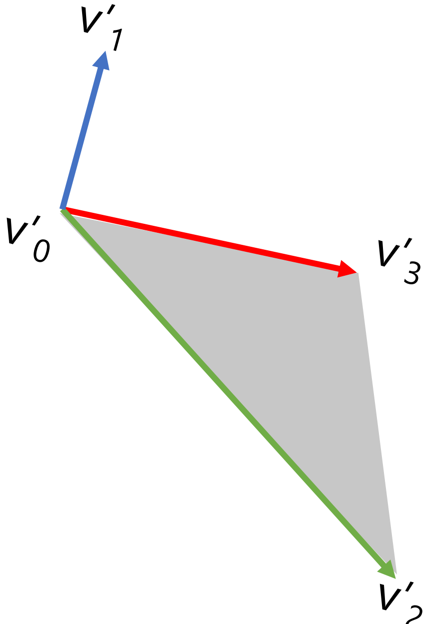
Jacobian Deformation Transfer

Jacobian can represent stretch and shear transforms unlike similarity transforms.



$E = [v_1 - v_0, v_2 - v_0, v_3 - v_0]$

Canonical
Metric Space



$E' = [v'_1 - v'_0, v'_2 - v'_0, v'_3 - v'_0]$

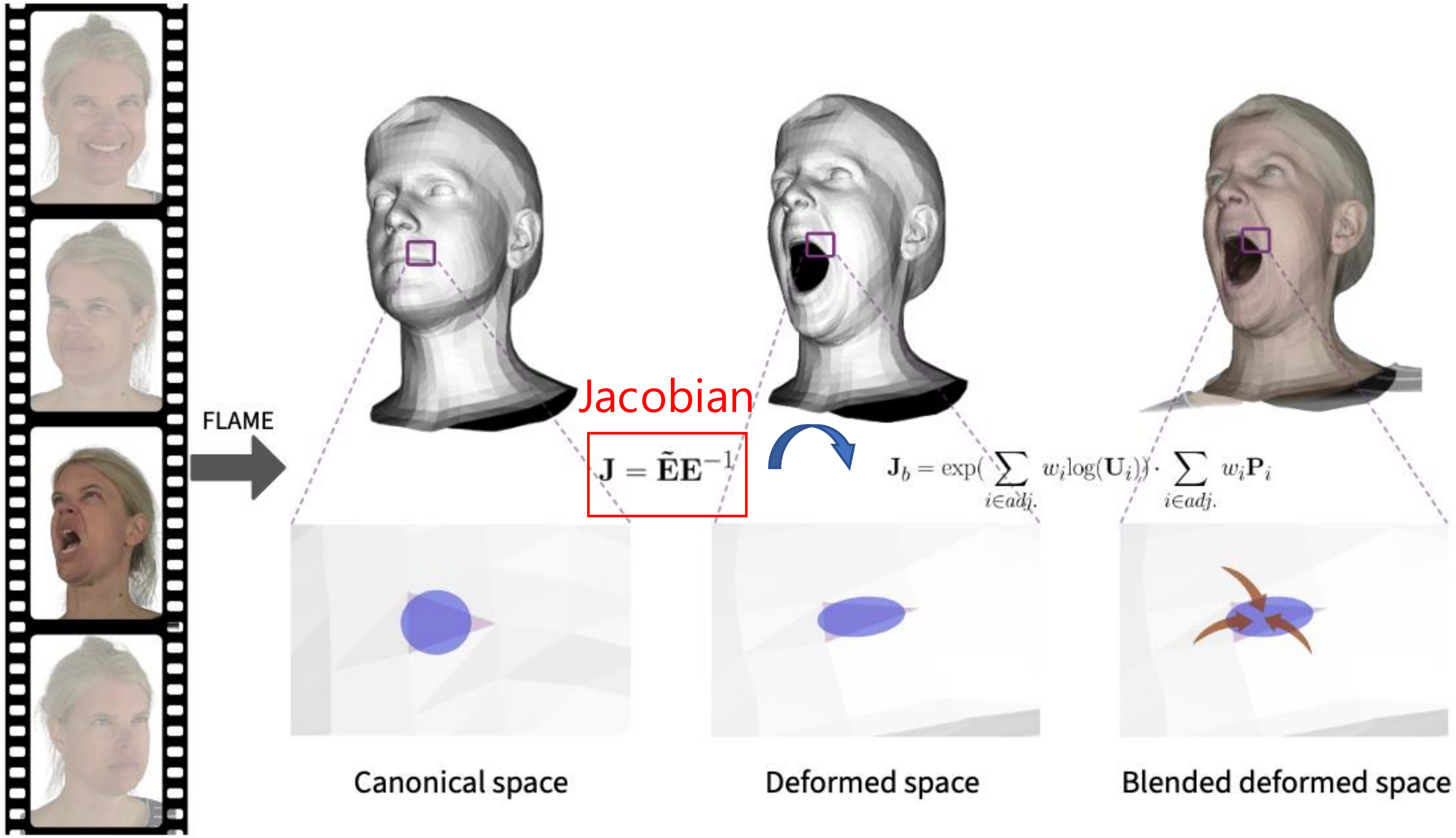
Deformed
Metric Space

$$J = E'E^{-1}$$

DTF (SIGGRAPH 2000, Sumner and Popovic.)

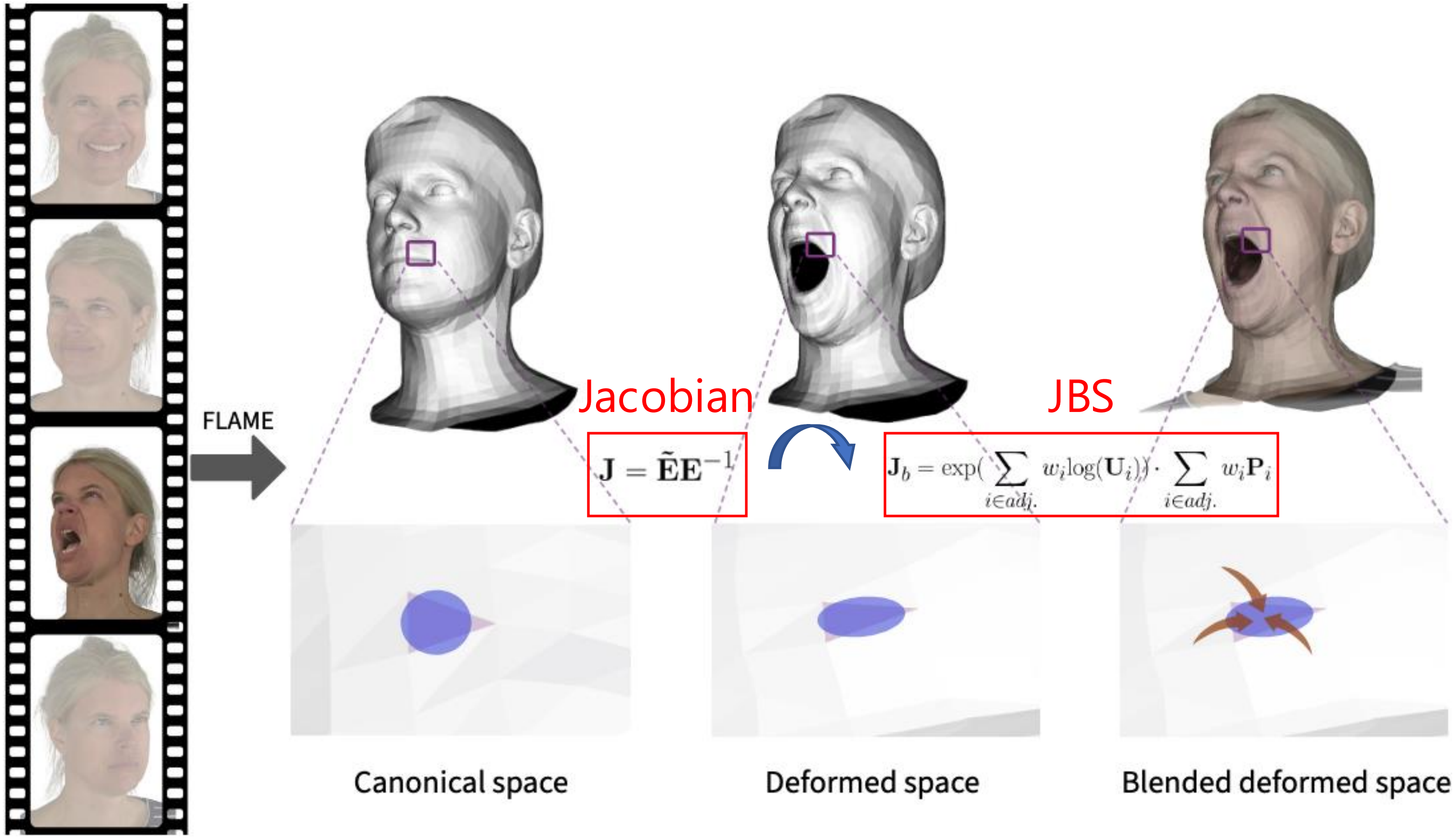
Jacobian to JBS

We replace again it with our JBS, blending Jacobians with adjacent transforms.



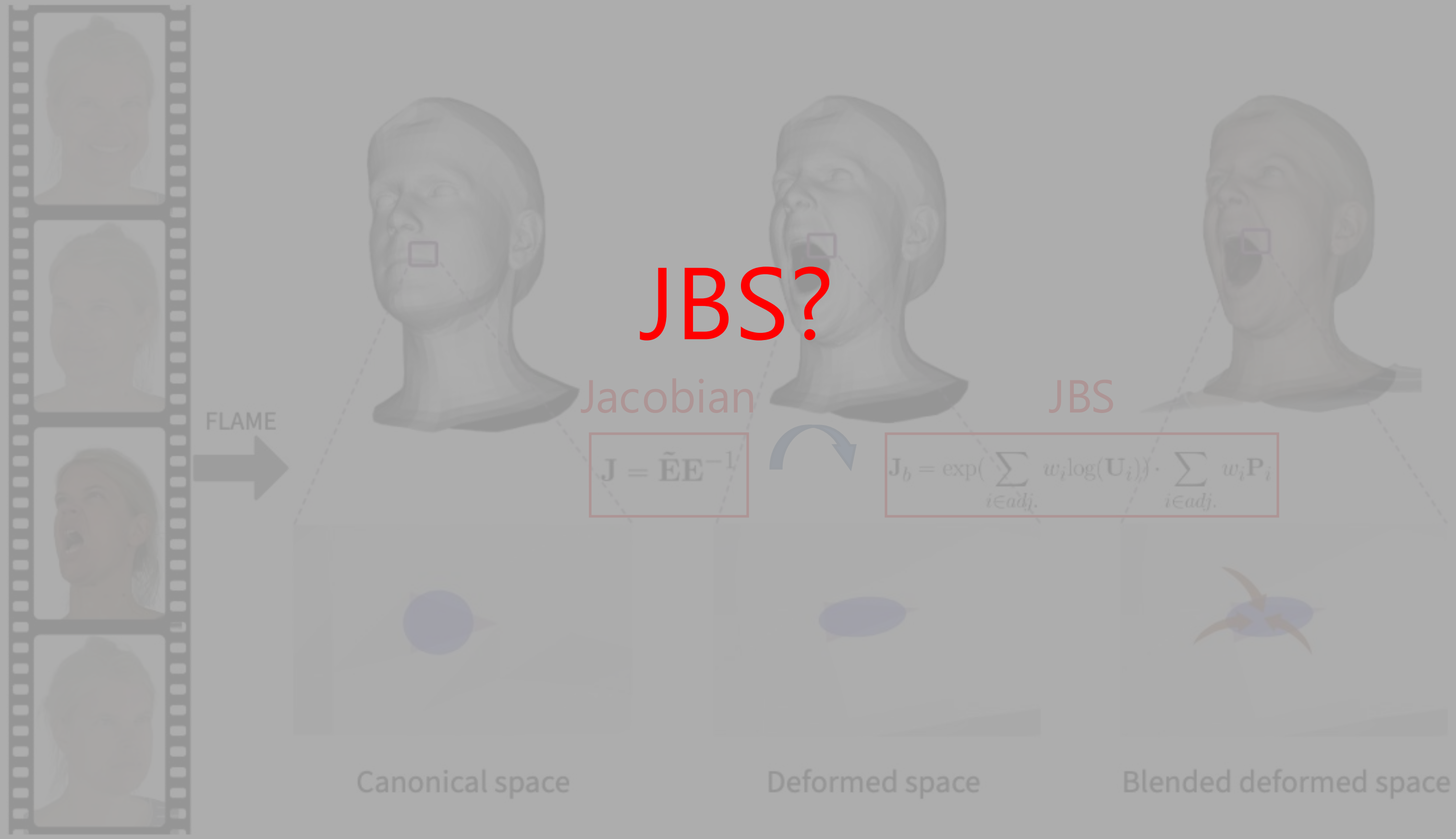
Jacobian to JBS

We replace again it with our JBS, blending Jacobians with adjacent transforms.



Jacobian to JBS

We replace again it with our JBS, blending Jacobians with adjacent transforms.



Let’s delve into JBS step-by-step

We replace again it with our JBS, blending Jacobians with adjacent transforms.

$$\mathbf{J}_b := \mathrm{JBS}(\mathbf{J}, w)$$

Let’s delve into JBS step-by-step

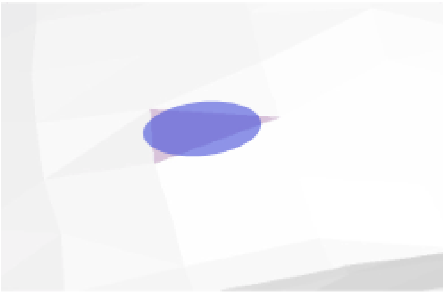
We replace again it with our JBS, blending Jacobians with adjacent transforms.

$$\mathbf{J}_b := \mathrm{JBS}(\mathbf{J}, w) = \underbrace{\exp(\sum_{i \in adj.} w_i \log(\mathbf{U}_i))}_{\mathbf{U}_b} \cdot \underbrace{\sum_{i \in adj.} w_i \mathbf{P}_i}_{\mathbf{P}_b}$$

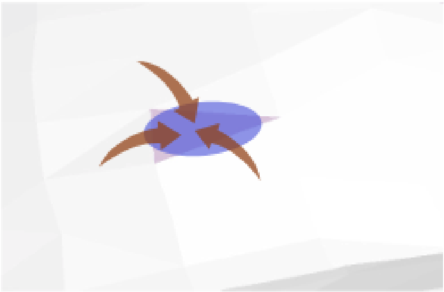
Let’s delve into JBS step-by-step

We replace again it with our JBS, blending Jacobians with adjacent transforms.

$$\mathbf{J}_b := \mathrm{JBS}(\mathbf{J}, w) = \boxed{\underbrace{\exp\Big(\sum_{i \in adj.} w_i \log(\mathbf{U}_i)\Big)}_{\mathbf{U}_b} \cdot \underbrace{\sum_{i \in adj.} w_i \mathbf{P}_i}_{\mathbf{P}_b}}$$



Deformed space



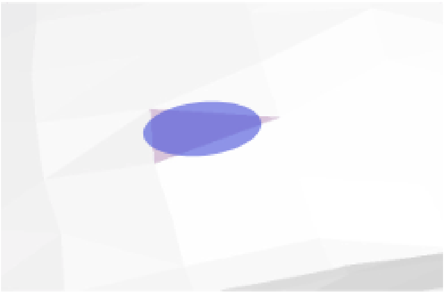
Blended deformed space

Let's delve into JBS step-by-step

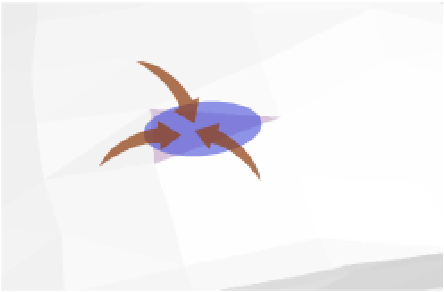
We replace again it with our JBS, blending Jacobians with adjacent transforms.

In matrix-logarithm space

$$\mathbf{J}_b := \mathrm{JBS}(\mathbf{J}, w) = \underbrace{\exp\left(\boxed{\sum_{i \in adj.} w_i \log(\mathbf{U}_i)}\right)}_{\mathbf{U}_b} \cdot \underbrace{\sum_{i \in adj.} w_i \mathbf{P}_i}_{\mathbf{P}_b}$$



Deformed space



Blended deformed space

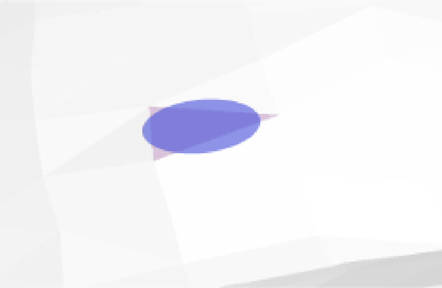
Let's delve into JBS step-by-step

We replace again it with our JBS, blending Jacobians with adjacent transforms.

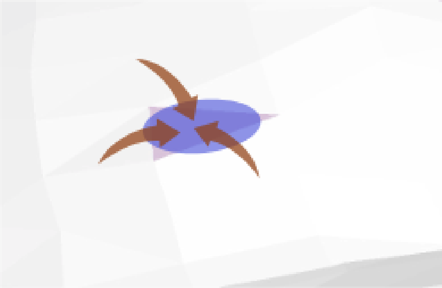
In matrix-logarithm space

In matrix-linear space

$$\mathbf{J}_b := \mathrm{JBS}(\mathbf{J}, w) = \underbrace{\exp(\boxed{\sum_{i \in adj.} w_i \log(\mathbf{U}_i)})}_{\mathbf{U}_b} \cdot \underbrace{\boxed{\sum_{i \in adj.} w_i \mathbf{P}_i}}_{\mathbf{P}_b}$$



Deformed space



Blended deformed space

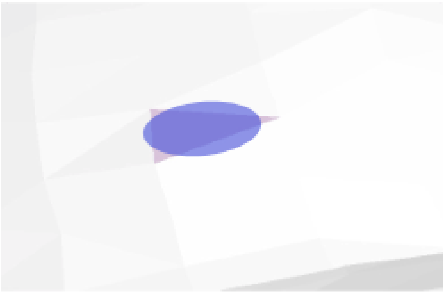
Let's delve into JBS step-by-step

We replace again it with our JBS, blending Jacobians with adjacent transforms.

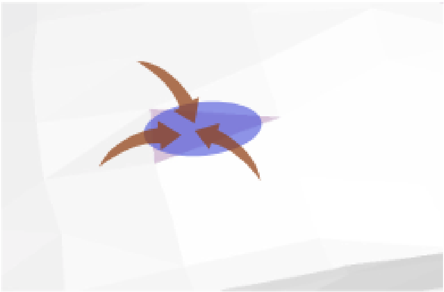
In matrix-logarithm space

In matrix-linear space

$$\mathbf{J}_b := \mathrm{JBS}(\mathbf{J}, w) = \underbrace{\exp\Big(\boxed{\sum_{i \in adj.} w_i \log(\mathbf{U}_i)}\Big)}_{\mathbf{U}_b} \cdot \underbrace{\boxed{\sum_{i \in adj.} w_i \mathbf{P}_i}}_{\mathbf{P}_b}$$



Deformed space



Blended deformed space

Let's delve into JBS step-by-step

$$\Sigma^{1/2} = \boxed{s_p R_p} R_c S_c \quad \mu = \boxed{s_p R_p} \mu_c + T_p$$

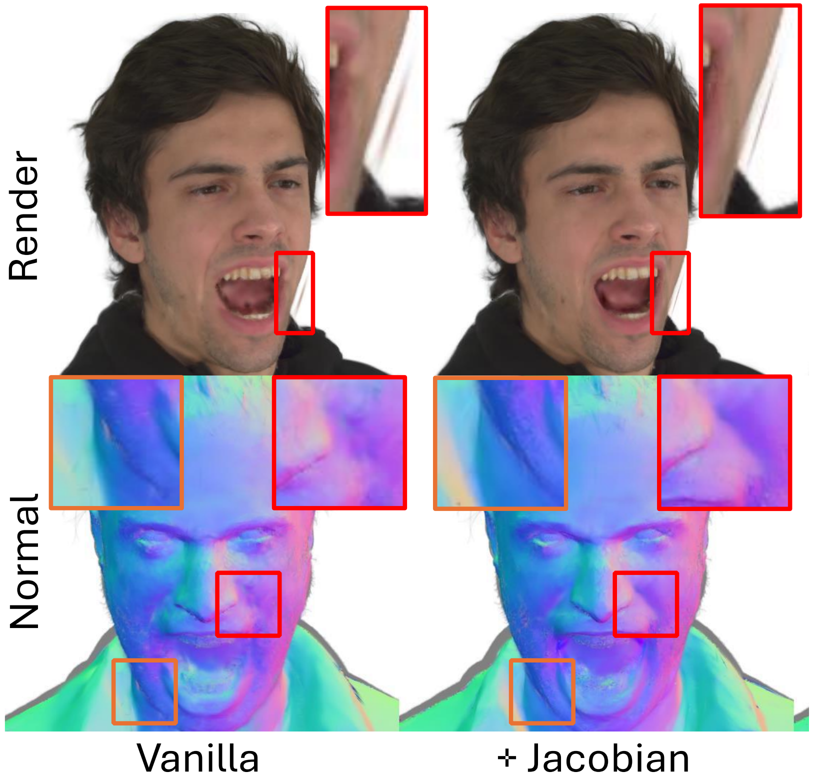
Let’s delve into JBS step-by-step

$$\Sigma^{1/2} = \boxed{s_p R_p} R_c S_c \qquad \mu = \boxed{s_p R_p} \mu_c + T_p$$

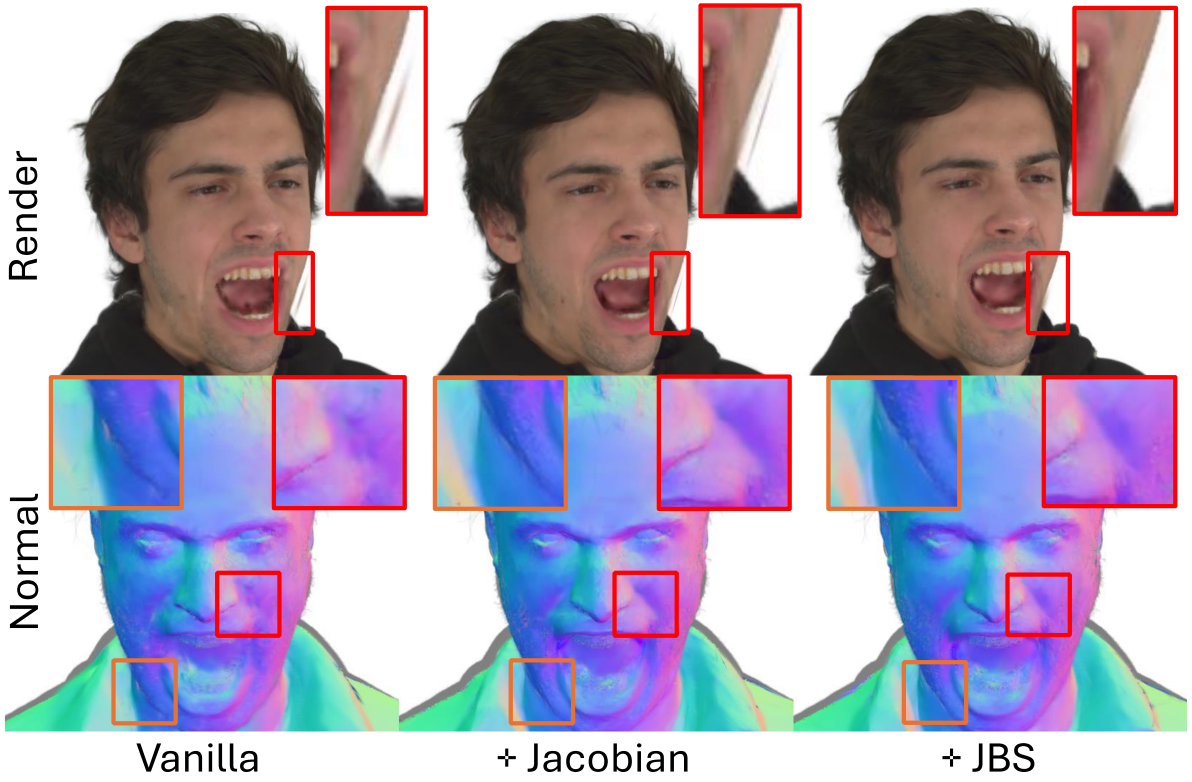
$$\Downarrow$$

$$\Sigma^{1/2} = \boxed{J_b} R_c S_c \qquad \mu = \boxed{J_b} \mu_c + T_p$$

Finally



Finally



Resolving hollow-illusion

Resolving hollow-illusion

Stop Gradient

Rotation R
Position μ

$+$

$$\mathcal{L}_{\text{eye}} = \sum_{i \in E} (1 - \alpha_i)^2$$

Resolving hollow-illusion

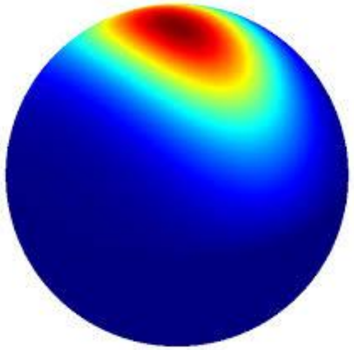
Stop Gradient

Rotation R
Position μ

+

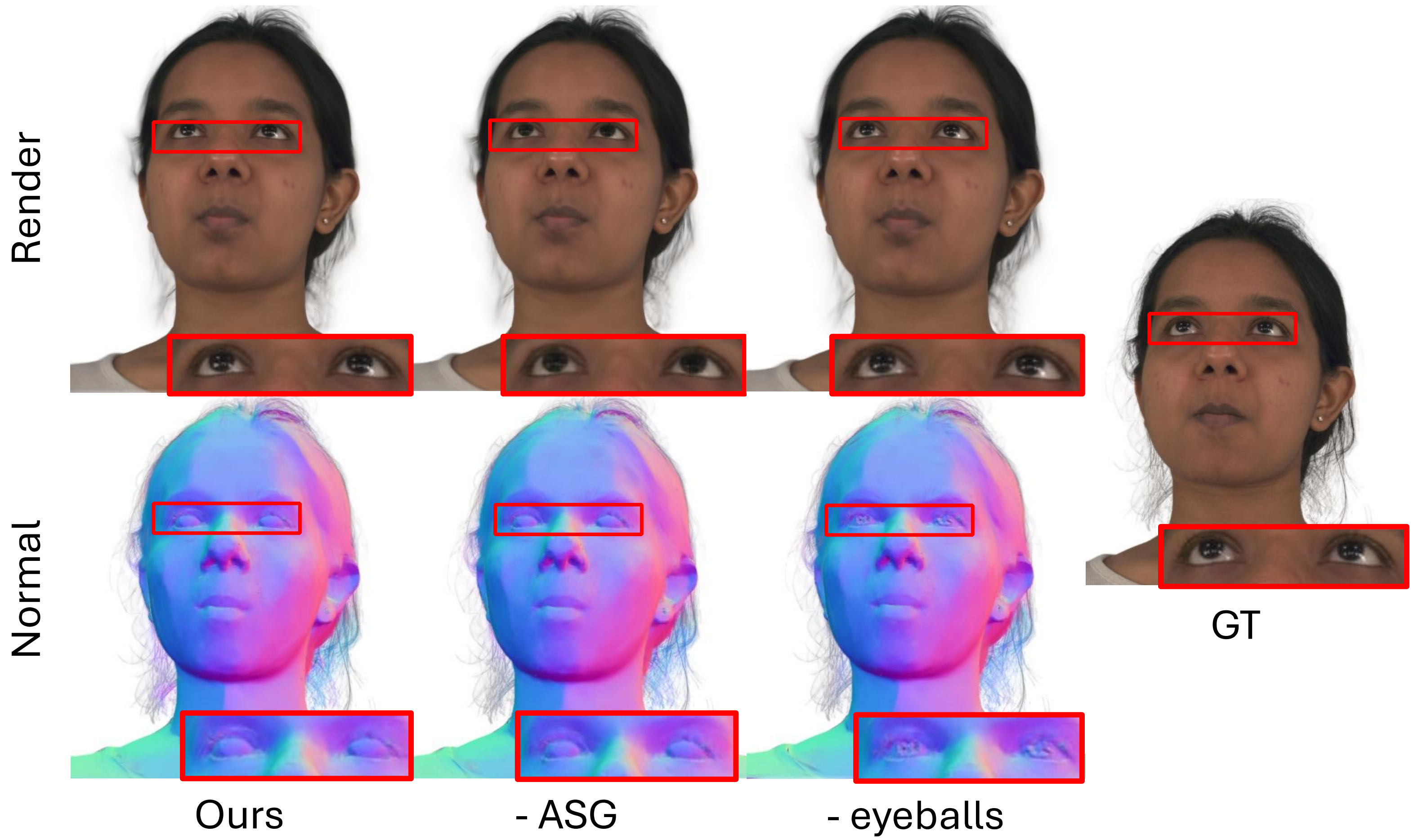
$$\mathcal{L}_{\text{eye}} = \sum_{i \in E} (1 - \alpha_i)^2$$

+

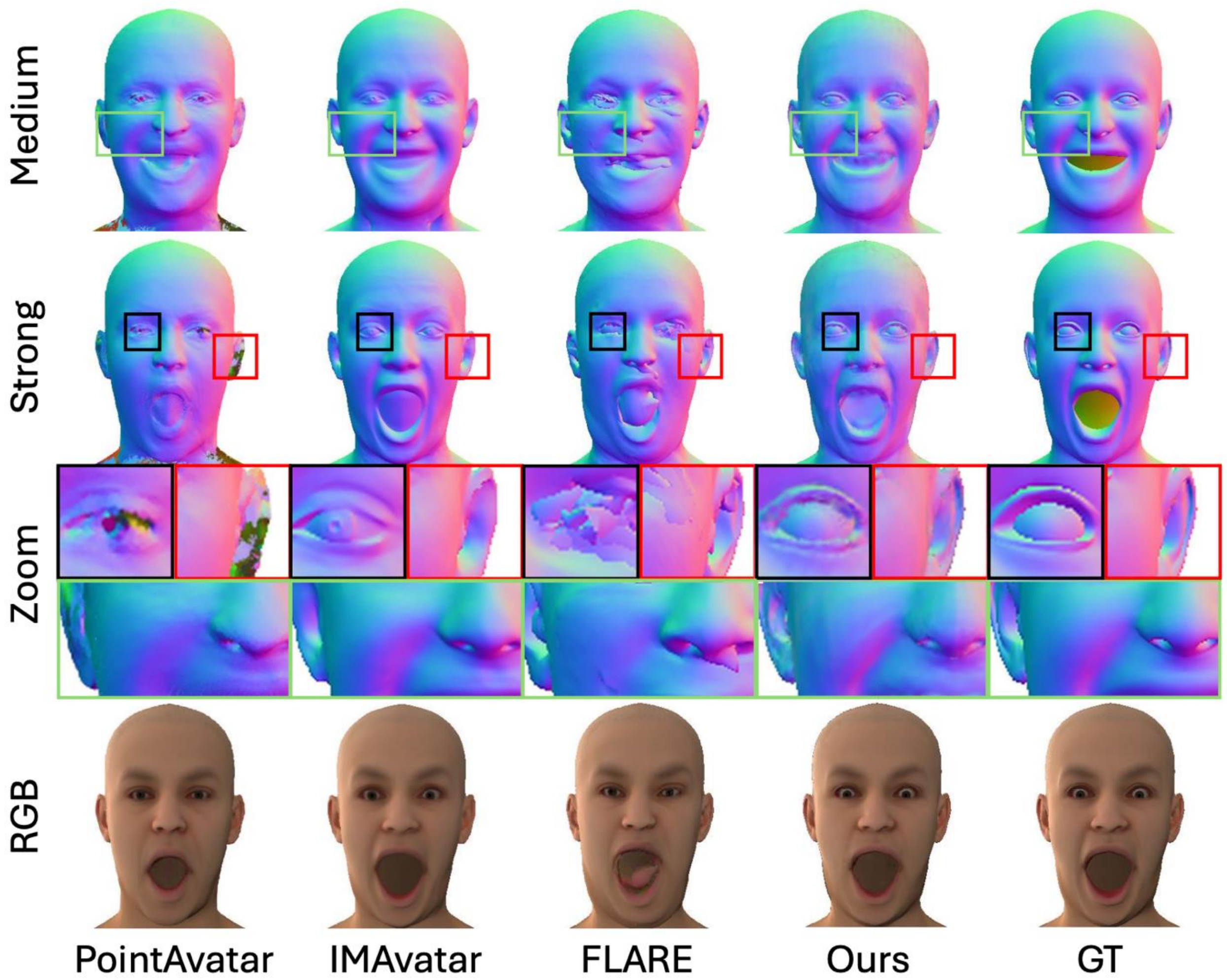


ASG (TOG 2013, Xu et al.)

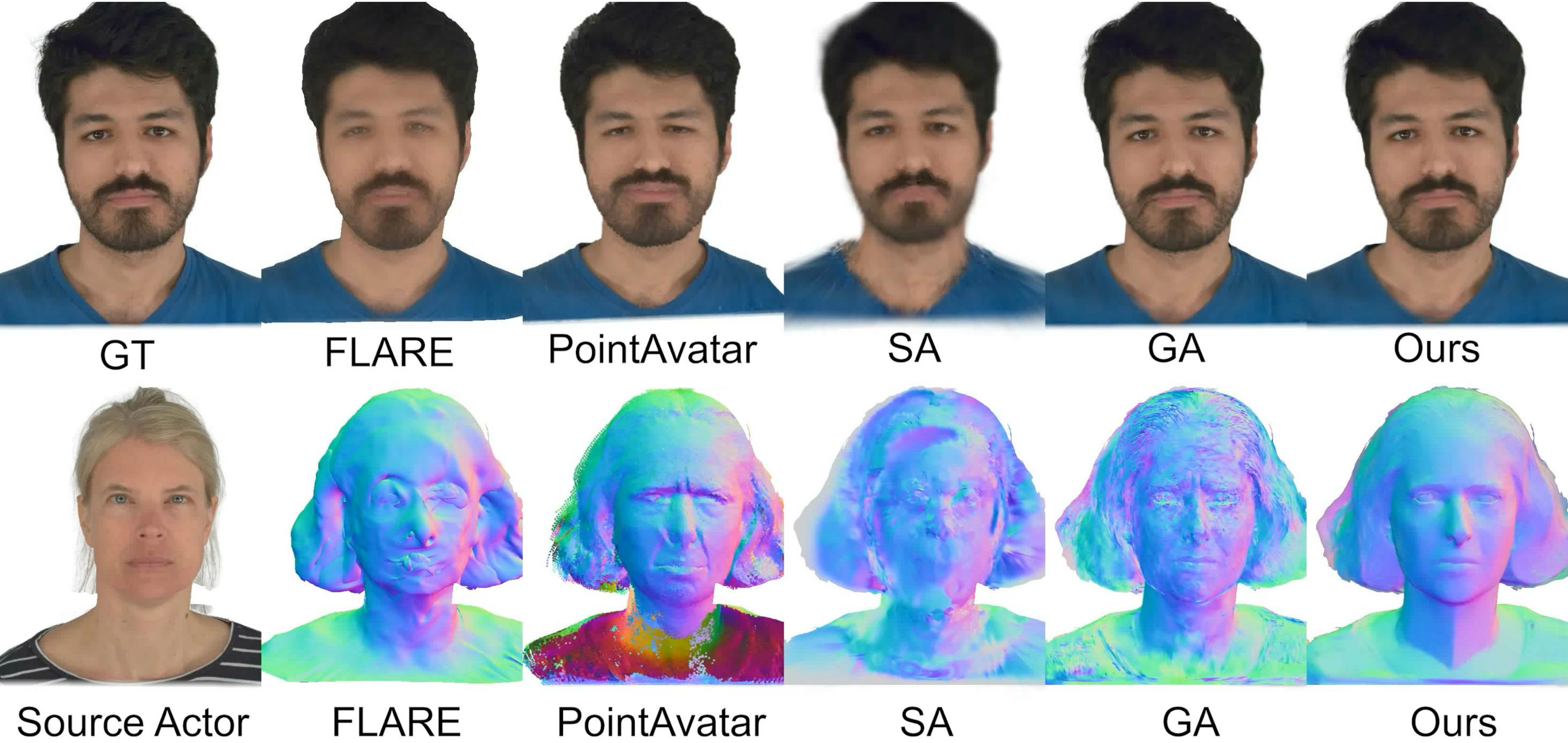
Resolving hollow-illusion



Comparison on FaceTalk (CVPR 2023, Zheng et al.)



Comparison on NeRSemble (SIGGRAPH 2023, Kirschstein et al.)



Application #1 **Relighting**
using GaussianShader (CVPR 2024, Jiang et al.)

Application #2 **Meshing**
using Truncated Signed Distance Function (TSDF)

Thank you for watching!

For more details visit summertight.github.io/SurFhead