

DAGSM: Disentangled Avatar Generation with GS-enhanced Mesh

Supplementary Material

In this document, we provide more results of ablation studies (A) and implementation details of our method (B).

A. More ablation studies

A.1. Ablation study on view-consistent refinement.

We conduct an ablation study to demonstrate the benefits of three components in the view-consistent refinement step: cross-view attention (CV-Att), IAW-DE, and using a weighted MSE loss with weighted map \mathcal{W} (wMSE). To better demonstrate the texture, we render a single layer of the white lace dress (i.e., the layer not occluded by the body) on a black background.

As shown in Fig. 8(1), directly using original SDEidt as in [87], the results have inconsistent texture style. Moreover, the overlapping texture regions of two rendered views are blurred due to view inconsistency. Without cross-view attention (Fig. 8(3)), the consistency of the texture decreases after enhancement, proving the importance of cross-view attention in maintaining texture style consistency. Without IAW-DE and the weighted MSE loss (Fig. 8(2)), the overlapping texture regions of two rendered views can easily get blurred even if their textures are slightly inconsistent. Fig. 8 (4) shows that applying a weighted MSE loss partially alleviates the blurred texture in overlapping regions caused by view inconsistency. However, IAW-DE addresses this issue more effectively (4 vs. 5) by allowing each view to focus on enhancing the most directly observed regions. Although the weighted MSE loss offers limited improvement (Fig. 8(5)), we also include it for its implementation simplicity.

A.2. Ablation study on different 3D representations.

In Fig. 9, we compare our GSM with the traditional mesh representation to generate a white lace dress while keeping all the other settings the same. As demonstrated in Fig. 9, our GSM can handle transparent fabric texture while traditional mesh with texture cannot automatically learn texture transparency.

B. Implementation Details

We provide more implementation details that cannot be included in the main paper due to space.

B.1. Details of IAW-DE

As described in Algorithm. 1, we input the weight map \mathcal{W} , the rendered image \mathcal{V} , the rendered image \mathcal{V} , the text prompt y , and refine strength $n = 0.4$ to Stable Diffusion

Algorithm 1 Incident-angle-weighted denoising (IAW-DE)

Input: The weight map \mathcal{W} ; the rendered image \mathcal{V} ; the text prompt y ; refine strength n (i.e., number of denoising steps)
Output: The refined image $\hat{\mathcal{V}}$.

```

1:  $z = \text{SD3\_encode}(\mathcal{V})$ 
2:  $\hat{z}_n = \text{addnosie}(z, n)$ ,  $z_n = \text{denosie}(\hat{z}_n, n, y)$ 
3: for  $t = n - 1$  to 0 do
4:    $\hat{z}_t = \text{addnosie}(z, t)$ 
5:   for  $\mathcal{W}_j$  in  $\mathcal{W}$  do
6:     if  $\mathcal{W}_j > \frac{t}{n}$  then
7:        $\hat{z}_{t,j} = z_{t+1,j}$ 
8:     end if
9:   end for
10:   $z_t = \text{denosie}(\hat{z}_t, t, y)$ 
11: end for
12:  $\hat{\mathcal{V}} = \text{SD3\_decode}(z_0)$ 
13: return  $\hat{\mathcal{V}}$ 

```

3 (SD3) [16], obtaining the refined image $\hat{\mathcal{V}}$ via incident-angle-weighted denoising (IAW-DE).

B.2. Formula of the regularizations in Eq. 6

In body generation (Sec. 4.2) and texture generation (Sec. 4.3), we add regularizations on the positions (\mathcal{L}_p), scales (\mathcal{L}_s), and rotations (\mathcal{L}_r) into Eq. 6 and Eq. 10 to constrain the movement of the Gaussians.

Positions regularization \mathcal{L}_p constrains the projection point of the Gaussian center on the bound triangle to be within the boundaries of the triangle, with a height offset below 0.1:

$$\mathcal{L}_p = \mathcal{L}_\lambda(\lambda_1) + \mathcal{L}_\lambda(\lambda_2) + \mathcal{L}_\lambda(1 - \lambda_1 - \lambda_2) + \|\max(z, 0.1)\|_2^2$$

$$\mathcal{L}_\lambda = \begin{cases} -\lambda, & \text{if } \lambda < 0 \\ 0, & \text{if } 0 < \lambda < 1 \\ \lambda - 1, & \text{if } \lambda > 1 \end{cases} \quad (12)$$

Following GaussianAvatars [56], we constrain the local scale of each 2D Gaussian to remain below 0.6:

$$\mathcal{L}_p = \|\max(\mathbf{s}, 0.6)\|_2^2 \quad (13)$$

Rotations regularization \mathcal{L}_r constrains the normal direction of the 2D Gaussian disk to be consistent with the normal direction \vec{n} of the bound triangle:

$$\mathcal{L}_r = \text{cosine}(\mathbf{R} \mathbf{r} \mathbf{n}, \vec{n}) \quad (14)$$

where \mathbf{n} is vector $[0, 0, 1]$.

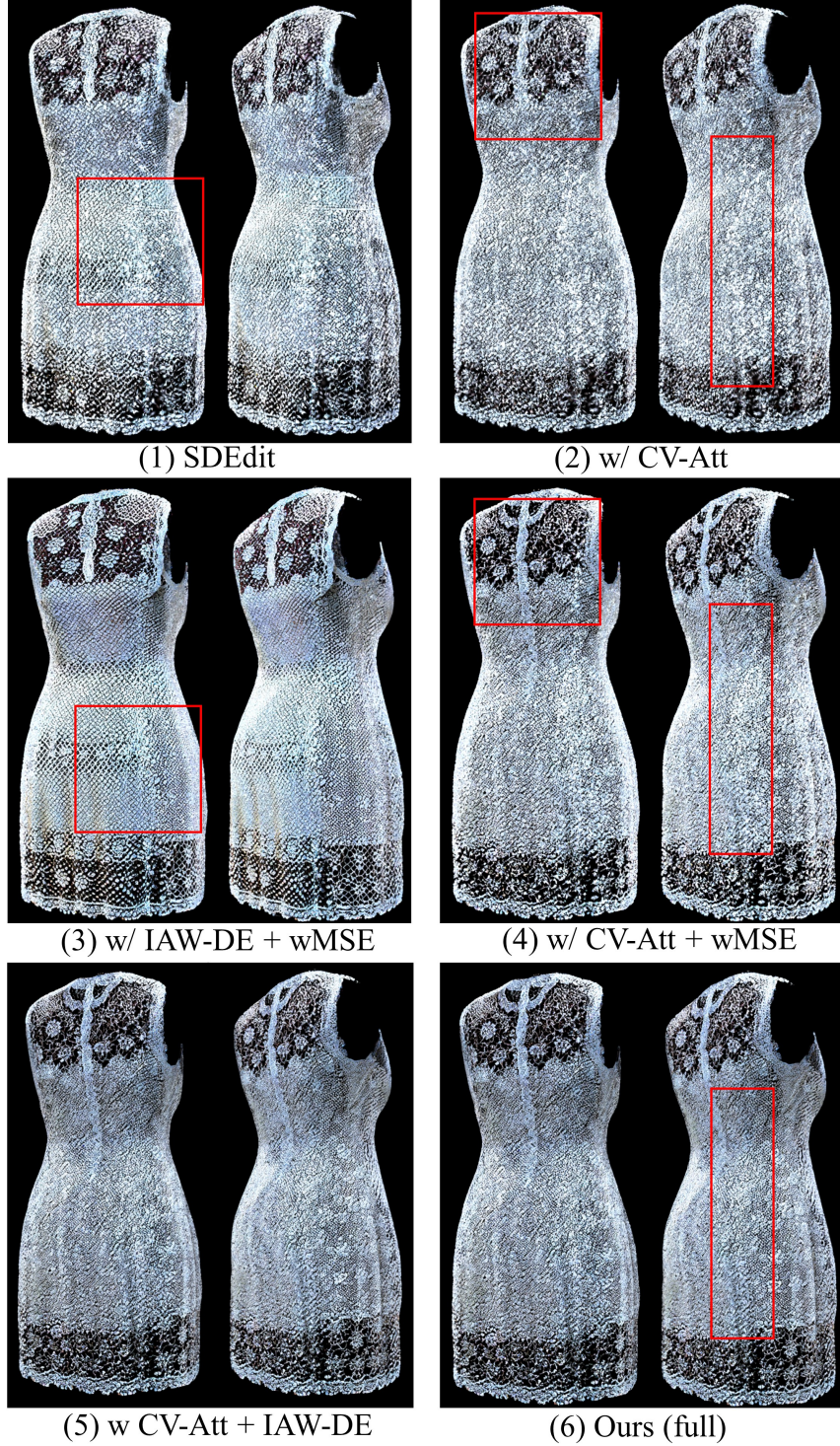


Figure 8. Ablation study on three components in view-consistent refinement: cross-view attention (CV-Att), IAW-DE, and using a weighted MSE loss with weighted map \mathcal{W} (wMSE). Using the original SDEdit (1) results in inconsistent texture styles and thus blurred overlapping regions due to view inconsistency. Removing cross-view attention (3) reduces texture consistency, proving the importance of cross-view attention in maintaining texture style consistency. Without IAW-DE and the weighted MSE loss (2), the overlapping texture regions of two rendered views can easily get blurred even if their textures are only slightly inconsistent. Applying a weighted MSE loss (4) partially alleviates the blurred texture in overlapping regions caused by view inconsistency. However, our IAW-DE addresses this issue more effectively (4 vs. 5) by allowing each view to focus on enhancing the most directly observed regions. Although the weighted MSE loss offers limited improvement (5 vs. Ours), we also include it for its implementation simplicity.



Figure 9. Ablation study on different 3D representations to show the advantage of GSM in handling transparent fabric texture. We compare our GSM with the traditional mesh representation to generate a white lace dress while keeping all the other settings the same. Our GSM can handle transparent fabric texture while traditional mesh fails to learn the transparent texture using existing differentiable engine (e.g. Nvdiffrast [35]) automatically.

Table 2. Selected body regions to initialize the garment 2DGS.

Types	Selected regions
t-shirt	Spines, Shoulders, Arms
long-shirt	Spines, Shoulders, Arms, ForeArms
hoodie	Spines, Shoulders, Arms, ForeArms
down coat	Spines, Shoulders, Arms, ForeArms, Hips
coat	Spines, Shoulders, Arms, ForeArms, Hips, UpLegs
shots	Hips, UpLegs
pants	Hips, UpLegs, Legs
shoes	Foots, ToeBases
sleeveless dress	Spines, Shoulders, Hips, UpLegs
long sleeve long dress	Spines, Shoulders, Arms, ForeArm, Hips, UpLegs, Legs

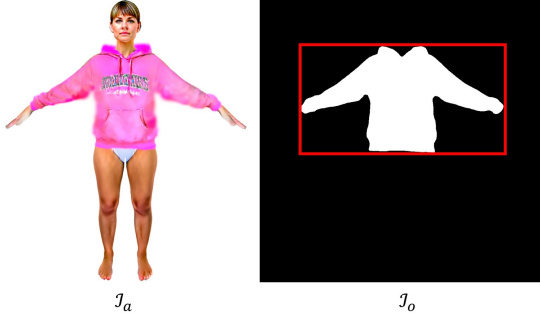


Figure 10. Inputs to SAM: the clothed human image I_a and the bounding box (red) of the Garment 2DGS (i.e. the hoodie) as SAM’s prompt.

B.3. Details of initializing 2D Gaussians in Sec. 4.3

As shown in Tab. 2, we use SMPL-X [53] part segments to select the 2DGS from the corresponding body regions to initialize the garment 2DGS for different clothing types.

B.4. Details of using SAM to obtain semantic masks

We implement SAM [33] via Huggingface [72]. Using the bounding box (the red box in Fig. 10) of the rendered gar-

Table 3. Learning rates of different parameters.

Parameter	Learning rate
β	0.01
\mathcal{D}	0.0001
u	0.0002→0.00002
s, r	0.005
\mathcal{U}_c, c	0.01
$\mathcal{U}_\alpha, \alpha, o$	0.1

ment’s 2DGS as the box prompts, we input it along with the clothed human image I_a to SAM to obtain the semantic mask \mathcal{M} of the garment.

B.5. More implementation details

We use the official code [23] to implement 2D Gaussian Splatting. We use the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\text{weightdecay} = 0$, and $\text{epsilon} = 10^{-15}$ for optimization and the learning rates of different parameters can be found in Tab. 3. We set $\lambda_p = \lambda_s = 10$, $\lambda_r = 1$ in Eq.6&10 and $\lambda_{dis} = 1$, $\lambda_{smooth} = 100$ in Eq.9. The rendered size is 1024×1024 and the RFDS loss is calculated on Sta-

ble Diffusion 3 [16], with the CFG weight 100 and time steps $t \sim \mathcal{U}(0.02, 0.98)$. All experiments are conducted on an NVIDIA A100 (40 GB). The body generation takes 3K iterations, consuming roughly 60 minutes. In the cloth generation, *original 2DGS generation* and *texture generation* require 2K iterations each, taking 30 and 20 minutes, respectively. In body generation and cloth generation, the view angles range from -15° to 30° in elevation and -180° to 180° in azimuth. View-consistent refinement takes 16 minutes, with 8 views around the object at 45° intervals, and the 0° view is the canonical view.