

# Tessellation GS: Neural Mesh Gaussians for Robust Monocular Reconstruction of Dynamic Objects

## Supplementary Material

### Camera Setup

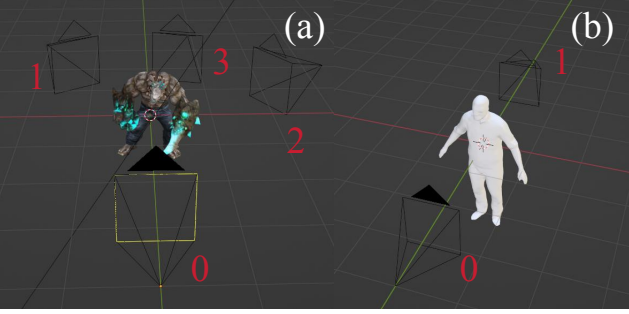


Figure 11. **Camera setup for SDNF and PSS.**(a): SDNF cameras. (b): PSS cameras.

We use blender coordinate system, and the character’s up direction aligns with the positive Z-axis.

For Smooth D-NeRF (SDNF), shown in Fig. 11 (a), camera 0 is a static validation view; camera 1, 2, and 3 rotate around the character (z-axis) but are mutually rigid. Camera 3 is training view, and camera 1 and 2 are test views, both 45 degrees azimuthal angles away from camera 3. Video supplementary material’s ”smooth\_dnerf\_validation”plementary material compares training and validation videos.

For People Snapshot dataset (PSS) shown in Fig. 11 (b), both training camera 0 and test camera 1 are static, and on the opposite sides of the character, 180 degree azimuthal angle to each others. We solely used video captured by camera 0 as input, and tested the results with camera 1. For the one cactus video from Unbiased4D (Ub4D), we could not provide rendering result from similarly novel views since only the front of the toy is captured in the training video.

### Stage One Loss Setup

For our proposed robust Chamfer loss  $\mathcal{L}_{RCD}$  in Eq. (2),  $U$  and  $V$  are sets of vertices in target and source meshes, and  $d$  is truncation distance. This setup helps the deformation model to not be interfered by large and sudden deformations between consecutive meshes, which are usually caused by temporal inconsistency and floating artifact of LRM. In addition, we define  $\mathcal{L}_{lap}$  as follows:

$$\mathcal{L}_{lap} = \frac{1}{N} \sum_{i=1}^N \|\mathbf{v}_i - \frac{1}{|\mathcal{E}_i|} \sum_{j \in \mathcal{E}_i} \mathbf{v}_j\|^2 \quad (13)$$

where  $\mathbf{v}_i$  is location of the  $i^{\text{th}}$  vertex and  $\mathcal{E}_i$  is the set of neighboring vertices of the  $i^{\text{th}}$  vertex.  $\mathcal{L}_n$  is normal consistency regularization defined as follows:

tency regularization defined as follows:

$$\mathcal{L}_n = \frac{1}{|\mathcal{E}|} \sum_{(i,j) \in \mathcal{E}} \|\mathbf{n}_i - \mathbf{n}_j\| \quad (14)$$

where  $\mathbf{n}_i$  is vertex normal vector of the  $i^{\text{th}}$  vertex and  $\mathcal{E}$  is the set of all neighboring vertex pairs.

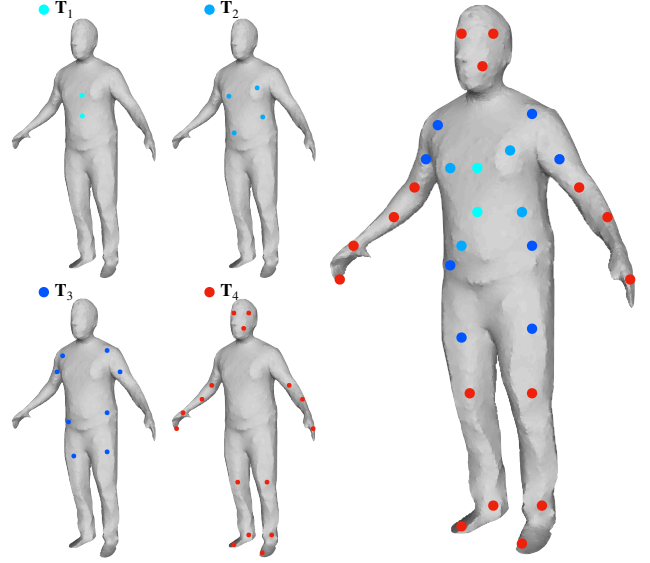


Figure 12. **Illustration of distribution of control points.** From  $T_1$  to  $T_4$ , each next temperature value is halved with  $T_4$  being 1.

### Deformation Model Setup

NPGS [5] uses a set of low-rank deformation basis to deform points. A fixed set of  $K$  learnable deformation basis is used for all timesteps, while each point has an individual learnable time-varying weight used to weighted sum the  $K$  deformation basis to get the displacement at timestep  $t$ . We observed that although it keeps the motion of the object low-rank, it doesn’t inherently keep local rigidity. Inspired by DynoSurf [48], we decided to make things opposite:  $K$  deformation control points provide time-varying deformation to drive deformation of mesh vertices. We express locations of each control point  $\mathbf{C}_k$  as weighted sum over all vertices in the canonical mesh with learnable weights  $\mathbf{m}_k$  as in Eq. (15), where  $\mathbf{V}_{t_0}$  is a matrix whose columns are positions of all vertices in the canonical mesh, this keeps

	Loss Term	Weight	Explanation
Stage One	$\mathcal{L}_{RCD}$	1	Our proposed robust Chamfer distance that truncate loss to zero above a threshold.
	$\mathcal{L}_{lap}$	0.5	Mesh Laplacian regularization ensures smoothness of mesh surface and uniformity of vertices.
	$\mathcal{L}_n$	0.001	Normal consistency regularization ensures smoothness of mesh surface.
Stage Two	$\mathcal{L}_1$	0.8	Rendered frames vs. input frames in L1 norm.
	$\mathcal{L}_{ssim}$	0.2	SSIM loss between rendered frames and input frames.
	$\mathcal{L}_{edge}$	0.2	L1 norm loss to penalize change in edge length of meshes after deformation.
	$\mathcal{L}_{lap}$	0.03	Same as in stage one.
	$\mathcal{L}_\alpha$	0.002	Mean opacity of all Gaussians to encourage the model to use fewer Gaussians.
	$\mathcal{L}_{normal}$	0.1 or 0	Predicted normal by DSINE [2] vs. rendered normal in L2 norm. Only used for static camera cases.
	$\mathcal{L}_{flow}$	0.01	Predicted optical flow by RAFT [41] vs. rendered optical flow in L1 norm.

Table 3. **Parameter weights for stage one and stage two.**

the control points within the convex hull of the mesh.

$$\begin{aligned}
\mathbf{C}_k &= \mathbf{V}_{t_0} \text{softmax}\left(\frac{\mathbf{m}_k}{T}\right) \\
&= \sum_{i=1}^N \left( \frac{e^{\frac{m_{k,i}}{T}}}{\sum_{j=1}^N e^{\frac{m_{k,j}}{T}}} \right) \mathbf{v}_{i,t_0}
\end{aligned} \quad (15)$$

We used hierarchical temperatures in softmax when deciding control point locations. High temperatures keep the control points close to the large-scale structures of the mesh, while low temperatures allow control points to spread out onto finer geometries of the mesh, analogous to SMPL’s tree structure. As illustrated in Fig. 12, we use 4 levels of granularities of control points, composed of in total 30 control points to drive the canonical mesh. To initialize the control nodes, we use farthest point sampling (FPS) to select 30 points, then we set the corresponding weight  $m_k$  to a large value to keep the result from FPS. Each vertex has a skinning weight over the 30 control points predicted by per control point MLP taking as input displacement from vertex to control point.

## Further Results

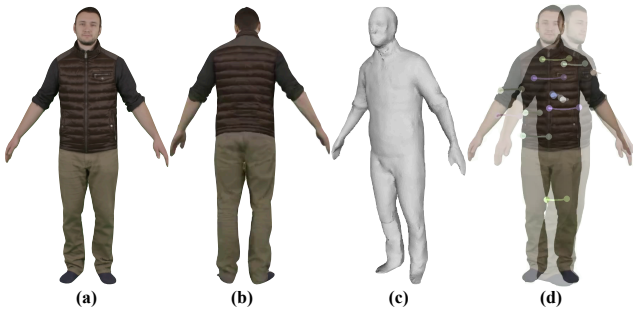


Figure 13. **People Snapshot result.** (a): input image. (b): novel view rendering result. (c): extracted mesh. (d): tracking result.

We have included an additional result on People Snapshot in Fig. 13. For complete tracking result, please refer to video results. We have also conducted further ablation study on the effectiveness of our stage two pipeline. We incorporated stage one of our pipeline with DG-Mesh [25], equipping DG-Mesh with LRM prior. Shown in Fig. 14

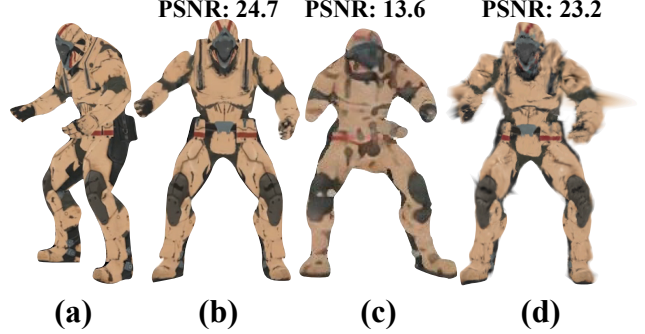


Figure 14. **Ablation and quantitative comparison on Tessellation GS.** (a): input image. (b): our method’s rendered novel view. (c): direct LRM rendering result. (d): DG-Mesh’s result after incorporating LRM prior.

(d), DG-Mesh still exhibits view-overfitting due to their relatively more free setup of mesh Gaussians. This proves that both stage one and stage two are crucial for the success of our pipeline.

## Optimization

In stage one, we used an exponentially decaying learning rate from  $1e-3$  to  $1e-5$  except for  $\mathbf{m}_k$  in Eq. (15), where we used a constant learning rate of  $1e-2$ . We train for 20000 steps until it converges. Together with LRM generation of mesh sequence, this step takes around 30 minutes. In stage two, we used an exponentially decaying learning rate for all MLPs, including motion MLPs, appearance decoders, and pose encoders, from  $1e-3$  to  $1e-5$ . We set constant learning rate of  $1e-3$  for Gaussian features on mesh vertices, Gaussian scales, and parent Gaussian opacities. We keep the constant learning rate of  $1e-2$  for  $\mathbf{m}_k$ . We train 400000 steps for the reconstruction to converge, which usually takes 60 minutes. In the initial 5000 steps, we train the model without Gaussian density control. After the initial 5000 steps, we use our adaptive density control technique mentioned in Sec. 3.2.4 to introduce new Gaussians and delete excessive Gaussians every 2000 steps until the final 5000 steps, when we stop the density control again.