PoseGaussian: Pose-Driven Novel View Synthesis for Robust 3D Human Representation

Anonymous Author(s)
Affiliation
Address
email

Supplementary Materials

This supplementary document provides additional resources to complement the main paper. It includes a narrated demo video, source code for reproducibility, and comprehensive comparisons with state-of-the-art approaches focused on temporal coherence—an aspect not fully detailed in the main paper. These materials offer deeper insight into the effectiveness and robustness of our proposed framework. Additional **visualization results** and **source code** are available on our anonymous project website:

`https://anonymous.4open.science/r/PoseGaussian/`

Due to IRB restrictions, subject test data is currently withheld but will be released upon approval prior to final publication.

To further justify the role of pose guidance in improving temporal coherence, we conduct quantitative evaluations on both public and custom datasets. Specifically, we use the ZJU-Mocap [4] benchmark to ensure comparability with prior work and introduce our own captured sequences involving more dynamic human motions. These sequences simulate real-world challenges such as rapid limb movement and self-occlusion. By analyzing metrics including ΔSSIM, ΔPSNR, and ΔLPIPS between consecutive frames, we demonstrate that our pose-informed framework achieves more consistent view synthesis over time.

Table 1 presents a quantitative evaluation of temporal coherence on selected dynamic sequences from the ZJU-Mocap dataset. These sequences (S377, S386, S387, S392, S393, S394) feature notable motions such as arm swings, punches, and kicks, which are more challenging than the sequences typically reported in prior work. Consequently, absolute SSIM, PSNR, and LPIPS scores are generally lower than those in the original benchmark. Nevertheless, our method demonstrates strong temporal consistency across frames. In particular, we achieve the best performance across all Δ-metrics, indicating smoother transitions and improved coherence. Specifically, we report an average ΔPSNR of 0.38, ΔSSIM of 0.024, and ΔLPIPS of 0.016, outperforming state-of-the-art baselines. While our method does not yield the highest absolute PSNR, it consistently leads in Δ-metrics, highlighting its robustness in preserving visual stability under motion. A similar trend is observed on our custom dataset, as shown in Table 2. While our method does not achieve the highest absolute PSNR, it consistently outperforms all baselines in other key metrics, including ΔPSNR, ΔSSIM, and ΔLPIPS, highlighting its superior temporal stability. Notably, we observe that all evaluated methods perform worse overall on our custom dataset compared to ZJU-Mocap (Table 1), likely due to the increased motion complexity and more frequent self-occlusions present in our sequences. This performance drop underscores the difficulty of our dataset and further demonstrates the robustness of our approach under more challenging, real-world conditions.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

Method	PSNR↑	μ (ΔPSNR)	σ (ΔPSNR)	SSIM↑	μ (ΔSSIM)	σ (ΔSSIM)	LPIPS↓	μ (ΔLPIPS)	σ (ΔLPIPS)
AS [TPAMI'24] [8]	**23.88**	0.39	0.31	0.882	0.025	0.019	0.042	0.029	0.017
HumanNeRF [arXiv'22] [6]	21.05	0.74	0.56	0.81	0.033	0.029	0.058	0.042	0.038
InstantNVR [CVPR'23] [1]	22.72	0.63	0.37	0.822	0.049	0.037	0.043	0.029	0.024
IBRNet [CVPR'21] [5]	22.83	0.62	0.37	0.827	0.036	0.028	0.052	0.031	0.023
GPS-Gaussian [CVPR'24] [7]	23.65	0.60	0.42	0.853	0.041	0.025	0.039	0.023	0.015
3D-GS [[ACMTOG'23]] [3]	20.74	0.97	0.61	0.808	0.049	0.035	0.045	0.038	0.026
GauHuman [CVPR'24] [2]	23.70	0.61	0.29	0.876	0.032	0.018	0.035	0.021	0.016
Ours	23.85	**0.38**	**0.23**	**0.890**	**0.024**	**0.013**	**0.032**	**0.016**	**0.010**

Table 1: Temporal coherence comparison on ZJU-MoCap [4].

Method	PSNR↑	μ (ΔPSNR)	σ (ΔPSNR)	SSIM↑	μ (ΔSSIM)	σ (ΔSSIM)	LPIPS↓	μ (ΔLPIPS)	σ (ΔLPIPS)
AS [TPAMI'24] [8]	22.5	0.61	0.53	0.815	0.028	0.022	0.053	0.028	0.020
HumanNeRF [arXiv'22] [6]	19.7	0.89	0.73	0.762	0.050	0.037	0.056	0.040	0.031
InstantNVR [CVPR'23] [1]	20.5	0.83	0.45	0.755	0.057	0.046	0.051	0.037	0.025
IBRNet [CVPR'21] [5]	20.3	0.65	0.58	0.694	0.045	0.035	0.064	0.042	0.026
GPS-Gaussian [CVPR'24] [7]	22.7	0.56	0.45	0.708	0.040	0.028	0.065	0.031	0.018
3D-GS [[ACMTOG'23]] [3]	20.3	0.72	0.43	0.746	0.052	0.039	0.049	0.037	0.027
GauHuman [CVPR'24] [2]	**22.8**	0.58	0.42	0.791	0.030	0.022	0.058	0.029	0.019
Ours	22.5	**0.43**	**0.31**	**0.825**	**0.023**	**0.015**	**0.046**	**0.023**	**0.015**

Table 2: Temporal coherence comparison on our custom dataset.

References

[1] Chen Geng, Sida Peng, Zhen Xu, Hujun Bao, and Xiaowei Zhou. Learning neural volumetric representations of dynamic humans in minutes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8759–8770, 2023.

[2] Shoukang Hu and Ziwei Liu. Gauhuman: Articulated gaussian splatting from monocular human videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20418–20431. IEEE, 2024.

[3] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (TOG)*, 42(4): 1–14, 2023.

[4] Sida Peng, Yuanqing Zhang, Yinghao Xu, Qianqian Wang, Qing Shuai, Hujun Bao, and Xiaowei Zhou. Neural body: Implicit neural representations with structured latent codes for novel view synthesis of dynamic humans. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9054–9063, 2021.

[5] Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4690–4699, 2021.

[6] Chung-Yi Weng, Brian Curless, Pratul P. Srinivasan, Jonathan T. Barron, and Ira Kemelmacher-Shlizerman. Humannerf: Free-viewpoint rendering of moving people from monocular video, 2022.

[7] Shunyuan Zheng, Boyao Zhou, Ruizhi Shao, Boning Liu, Shengping Zhang, Liqiang Nie, and Yebin Liu. Gps-gaussian: Generalizable pixel-wise 3d gaussian splatting for real-time human novel view synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19680–19690, 2024.

[8] Xiaowei Zhou, Sida Peng, Zhen Xu, Junting Dong, Qianqian Wang, Shangzhan Zhang, Qing Shuai, and Hujun Bao. Animatable implicit neural representations for creating realistic avatars from videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(6):4147–4159, 2024.

Appendix

3D Gaussian Splatting Preliminary The proposed *PoseGaussian* approach builds upon the core principles of 3D Gaussian Splatting (3D-GS) (Kerbl et al. 2023). In 3D-GS, a static scene is modeled using a collection of spatial primitives, where each primitive is parameterized as an anisotropic Gaussian distribution centered at a 3D location. The density at a spatial point $\mathbf{x} \in \mathbb{R}^3$ due to the i-th Gaussian is given by:

$$g_i(\mathbf{x}) = \frac{1}{(2\pi)^{3/2}|\Sigma_i|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^T\Sigma_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right),$$

where $\boldsymbol{\mu}_i \in \mathbb{R}^3$ is the center of the Gaussian, and $\Sigma_i \in \mathbb{R}^{3\times3}$ is the covariance matrix that encodes both the shape and orientation. This covariance can be decomposed as $\Sigma_i = R_i S_i R_i^T$, where R_i is a rotation matrix and S_i is a diagonal scaling matrix representing variances along the local principal axes.

When the 3D Gaussians are projected into 2D image space, their covariances are transformed via a view transformation matrix W and a Jacobian matrix J from an affine approximation of the camera projection. The resulting 2D covariance becomes:

$$\Sigma_i' = JW\Sigma_i W^T J^T,$$

preserving the geometric effects of both rotation and scale during projection.

In our approach, we predict a dense 2D map of Gaussian parameters per pixel location x, denoted as:

$$\mathbf{G}(x) = \{\mathcal{M}_\tau(x)\}, \quad \tau \in \{\mathrm{p}, \mathrm{c}, \mathrm{r}, \mathrm{s}, \alpha\},$$

where $\mathcal{M}_\mathrm{p}(x)$ gives the projected center position, $\mathcal{M}_\mathrm{c}(x)$ the color, $\mathcal{M}_\mathrm{r}(x)$ the rotation, $\mathcal{M}_\mathrm{s}(x)$ the scale, and $\mathcal{M}_\alpha(x)$ the opacity. These parameters are designed to mirror the structure of 3D-GS after projection:

- $\mathcal{M}_c$ represents the pixel color, which is directly retrieved as the mean color value at each pixel location.
- $\mathcal{M}_p$ denotes the projected pixel position, computed by transforming the estimated depth map through camera intrinsics and projecting it into the virtual view.
- $\mathcal{M}_\mathrm{r}(x)$ corresponds to the local rotation matrix R_i used in the covariance decomposition;
- $\mathcal{M}_\mathrm{s}(x)$ encodes per-axis scale values analogous to the diagonal entries in S_i;
- $\mathcal{M}_\alpha(x)$ defines the pixel-level opacity, which is used in Gaussian alpha blending as described in the following color blending equation.

The final rendered color at each pixel x is then computed using front-to-back compositing across multiple overlapping Gaussians, sorted by depth:

$$C(x) = \sum_i \alpha_i(x)\, c_i(x) \prod_{j<i} (1-\alpha_j(x)),$$

where $\alpha_i(x) = \mathcal{M}_\alpha^{(i)}(x)$ represents the pixel-level opacity, and $c_i(x) = \mathcal{M}_c^{(i)}(x)$ denotes the color directly obtained from the pixel color map $\mathcal{M}_c$. This blending formulation enables smooth, differentiable rendering and effectively captures both appearance and temporal continuity.

Therefore these five output maps—$\mathcal{M}_\mathrm{p}(x)$, $\mathcal{M}_\mathrm{c}(x)$, $\mathcal{M}_\mathrm{r}(x)$, $\mathcal{M}_\mathrm{s}(x)$, and $\mathcal{M}_\alpha(x)$—are visually illustrated in Fig. 5, showing how each component contributes to the Gaussian-based rendering process.

Implementation Details This is the architecture of our encoder and decoder:

As depicted in Fig. 5, the three parallel encoders share an identical structure: an initial 3×3 convolutional layer with 32 channels, followed by six residual units with progressively increasing channel dimensions. To enhance modality-specific encoding and channel selectivity, each residual unit includes a Squeeze-and-Excitation (SE) block. These SE modules adaptively recalibrate channel-wise responses, mitigating modality noise and reinforcing semantically important features across diverse inputs. The choice of a 3×3 kernel size and progressive channel widths (32, 64, 96, 128) balances efficient local feature extraction with scalable semantic representation. Skip connections are formed by concatenating intermediate encoder features at corresponding layer indices and forwarding them to the decoder. This skip alignment enriches the decoder with multi-level semantic information throughout the reconstruction process.

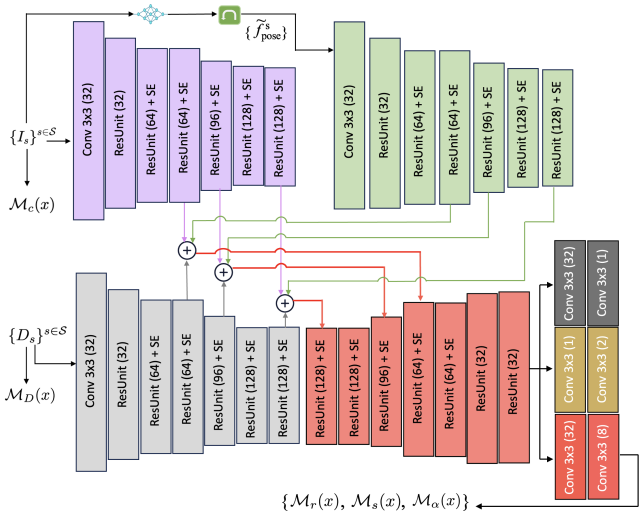

Figure 5: Architecture of the pose encoder-decoder network.

The decoder mirrors the encoder structure to complete the U-Net architecture, progressively upsampling and refining the feature maps. At the final decoder stage, three output heads are attached: a scale-rotation-opacity (SR-Opacity) branch for predicting Gaussian parameters, a depth branch to enforce depth consistency regularization, and an uncertainty branch for predicting per-pixel confidence maps. The depth branch outputs a normalized depth map via a Conv3x3(1) layer followed by a Tanh activation. The SR-Opacity branch applies a Conv3x3(8) layer, splitting outputs into 3 channels for scale (Softplus activation), 3 channels for rotation (normalization), and 2 channels for opacity (Sigmoid activation).

The uncertainty branch outputs 2 channels via a Conv3x3(2) layer followed by Sigmoid activations, where Channel 1 corresponds to depth confidence and Channel 2 to SR-Opacity confidence.

Detailed Configuration Variants and Their Impact

- **Config 1: Simplified Architecture**
  This variant uses only a single 5×5 Conv2D layer with 32 channels followed by one Residual Block. Due to its shallow depth and limited capacity, this configuration struggles to capture complex features, resulting in lower perceptual quality (LPIPS: 0.13) and reduced efficiency.
- **Config 2: Deeper Residual Learning with Skip Connection**
  This setup enhances the residual block structure by stacking two Conv2D layers with a skip connection between them. This deeper residual learning significantly improves image reconstruction, achieving the highest SSIM (0.92), lowest LPIPS (0.09), and PSNR of 26.0. These metrics indicate better preservation of structural and perceptual details compared to Config 1.
- **Config 3: Adding Batch Normalization and ReLU Activation**
  Building on Config 2, this variant introduces Batch Normalization layers and ReLU activations within the residual blocks. These additions improve feature representation and training stability, further enhancing the network's ability to generalize.
- **Config 4: No Downsampling Layers**
  In this variant, downsampling operations (e.g., strided convolutions or pooling) are removed to maintain spatial resolution throughout the encoder. While this results in a good SSIM score (0.91) and moderate PSNR (25.1), the LPIPS metric slightly degrades (0.11), reflecting a small perceptual quality drop compared to Config 2 and 3.

System Setup for Runtime Performance All experiments were conducted on a desktop workstation with an AMD Ryzen 9 5900X CPU, 64 GB RAM, and an NVIDIA RTX 4080 GPU (16 GB VRAM). All input images are resized to 512×512 pixels, and our pipeline achieves interactive performance with a per-frame processing time of approximately 100 milliseconds. The end-to-end processing is decomposed as follows: the Gaussian Splatting (GS) rendering module consumes $\sim$38 ms; pose feature extraction using a lightweight pose encoder takes $\sim$12 ms; image feature extraction via a ResNet-based backbone requires $\sim$20 ms; and our Temporal Pose Stabilizer (TPS), which adjusts view blending based on inter-frame pose dynamics, contributes $\sim$18 ms. The remaining $\sim$12 ms accounts for view-dependent shading, memory overhead, and post-processing (e.g., alpha blending and tone mapping). All components are implemented using PyTorch with CUDA acceleration, and timings were averaged over 300 consecutive frames. This configuration preserves the core efficiency of standard Gaussian Splatting pipelines while enabling temporally robust rendering in dynamic human scenes.

Baseline Evaluation Protocols and Setup Standardization To enable a fair and meaningful comparison across baseline methods that originally adopt differing evaluation protocols and input assumptions, we took the following standardization steps:

- Wherever possible, we re-implemented or adapted baseline methods to align with our evaluation pipeline. We also utilized official pretrained models or public code released by the authors when available to minimize reimplementation discrepancies.
- Baseline methods often exclude different cameras during testing (e.g., *HumanNeRF* omits Camera 1 and *InstantNVR* omits Camera 4). To unify the testing configuration, we fixed the number of input views to 8 cameras, uniformly sampled across the rig. Unless otherwise specified, this view configuration was used consistently across all evaluations.
- Input data for all methods were synchronized to share the same temporal frame sequences and test splits. In cases where re-evaluation of pretrained models was not feasible (e.g., due to code dependency or training pipeline complexity), we reported the published results verbatim, clearly annotating any protocol differences for transparency.
- Some baseline methods did not report perceptual metrics such as LPIPS. In those cases, we attempted to run their released models within our standardized setup to estimate the missing metrics. When this was not possible, we explicitly mark the metric as unavailable to avoid misleading comparisons.
- The baseline methods differ in input modality assumptions. For example, *HumanNeRF*, *InstantNVR*, and *MonoHuman* consume monocular video, while *GPS-Gaussian* and *NHP* rely on multi-view images. Our method requires synchronized multi-view video. To adapt monocular video baselines for comparison, we extracted single-camera trajectories from our multi-view dataset to form monocular sequences, ensuring input compatibility with their original settings. Evaluation metrics were then computed from the same view across all methods to ensure alignment.

While some residual discrepancies in evaluation protocols remain unavoidable due to fundamental differences in model design and data requirements, the above measures help clarify relative performance trends. This rigorous standardization enhances the credibility of our comparative analysis and supports the validity of our reported improvements.