GauHuman: Articulated Gaussian Splatting from Monocular Human Videos

— Supplementary Material

Shoukang Hu  Tao Hu  Ziwei Liu
S-Lab, Nanyang Technological University

1. Implementation Details

Implementation Details of GauHuman Our pose refinement module consists of 4 fully connected layers, i.e., an input layer, 2 hidden layers, and one output layer. Each layer is followed by a ReLU activation. The dimension of the hidden layer is 128, while the input and output dimension of the pose refinement module is 69. The LBS offset module adopts 5 fully connected layers with an input layer, 3 hidden layers, and one output layer. The ReLU activation is used after each layer. Positional encoding is applied to the 3D Gaussian positions before they are fed into the input layer. The input and output dimensions of the LBS offset module are 63 and 24 (number of joints). We use Adam optimizer with a learning rate \(10^{-5}\) to optimize the above two modules. We set the threshold of KL divergence as 0.4 to perform split/clone operations. Other training details for 3D Gaussians are the same as [5].

Evaluation Metrics. To quantitatively evaluate the quality of rendered novel view and novel pose images, we report the peak signal-to-noise ratio (PSNR) [15], structural similarity index (SSIM) [17] and Learned Perceptual Image Patch Similarity (LPIPS) [21].

Details of Comparable Methods. 1). Subject-specific optimization-based methods. Neural Body (NB) [12] encodes latent codes in SMPL vertex points and uses them to learn the neural radiance fields. Animatable NeRF (AN) [11] learns a canonical human NeRF through skeleton-driven deformation and learned blend weight fields. AS[13] further extends [11] by learning a signed distance field and a pose-dependent deformation field for residual information and geometric details of dynamic 3D humans. HumanNeRF [18] incorporates a pose refinement module, LBS field, and non-rigid deformation module to optimize a volumetric representation of 3D humans in the canonical space. DVA [14] extends mixtures of volumetric primitives [9] to articulated 3D humans for high-quality telepresence. InstantNVR [1] and InstantAvatar [4] propose to use multi-hashing encoding for fast training of 3D humans. 2). Generalizable methods. PixelNeRF [20] learns a neural network to infer the radiance field based on the input image. Neural Human Performer (NHP) [7] aggregates pixel-aligned features at each time step and temporally-fused features to learn generalizable neural radiance fields. For generalizable methods, we evaluate each subject (e.g., one subject of MonoCap) by first pre-training the model on the other data set (e.g., ZJU_Mocap data set) and then fine-tuning it on the evaluated subject.

Efficient Implementation of KL Divergence The Kullback–Leibler (KL) divergence of two 3D Gaussians is computed as follows:

\[
KL(G(x_0)|G(x_1)) = \frac{1}{2}(tr(\Sigma_1^{-1}\Sigma_0) + \ln \frac{\det \Sigma_1}{\det \Sigma_2} + (p_1 - p_0)^T \Sigma_1^{-1}(p_1 - p_0) - 3),
\]

where \(p_0, \Sigma_0, p_1, \Sigma_1\) are the position and covariance matrix of two 3D Gaussians \(G(x_0)\) and \(G(x_1)\).

As the covariance matrix is decomposed into the product of rotation and scaling matrices \(\Sigma = RSS^T R^T\), we simplify the computation of matrix inverse and determinant operations, i.e.,

\[
\Sigma_1^{-1} = (RSS^T R^T)^{-1} = RS^{-1}S^{-1} R^T,
\]

where \(S = \det(RSS^T R^T) = \det(S)\).

Since scaling matrix \(S\) is a diagonal matrix, the inverse and determinant of a diagonal matrix can be easily derived by inversing and prodding the diagonal elements respectively. Meanwhile, the inverse of the orthogonal rotation matrix is the transpose of the original matrix. The above simplification saves the computation time for matrix inverse and determinant operation.

2. Details of Loss Functions

Photometric Loss. Given the ground truth target image \(C\) and predicted image \(\hat{C}\), we apply the photometric loss as follows:

\[
L_{\text{color}} = \|\hat{C} - C\|_2.
\]

Mask Loss. We also leverage the human region masks for Human NeRF optimization. The mask loss is defined as:

\[
L_{\text{mask}} = \|\hat{M} - M\|_2,
\]
where $\hat{M}$ is the accumulated volume density and $M$ is the ground truth binary mask label.

**SSIM Loss.** We further employ SSIM to ensure the structural similarity between ground truth and synthesized images, i.e.,

$$L_{SSIM} = SSIM(\hat{C}, C).$$

(5)

**LPIPS Loss.** The perceptual loss LPIPS is also utilized to ensure the quality of rendered image, i.e.,

$$L_{LPIPS} = LPIPS(\hat{C}, C).$$

(6)

In summary, the overall loss function contains four components, i.e.,

$$L = \ell_{\text{color}} + \lambda_1 \ell_{\text{mask}} + \lambda_2 L_{SSIM} + \lambda_3 L_{LPIPS},$$

(7)

where $\lambda$’s are loss weights. Empirically, we set $\lambda_1 = 0.5$, $\lambda_2 = \lambda_3 = 0.01$ to ensure the same magnitude for each loss.

3. **Rotating Spherical Harmonic coefficients**

When transforming 3D Gaussians from canonical space to posed space, the SH coefficients should also be rotated for view-dependent color effects. The above is achieved by first computing a Wigner D-matrix [19] and then rotating SH coefficients with the Wigner D-matrix. In our implementation, we find that rotating SH coefficients has little effect on the final performance\(^1\), so we do not consider it in our work.

4. **Further Analysis**

**Evaluations on novel poses.** For each subject in ZJU_MoCap [12] and MonoCap [2, 3, 13], we collect 20 frames for novel pose synthesis by sampling 1 frame every 10 frames. We show the performance comparison of novel pose synthesis as follows. As shown in the table, GauHuman outperforms baselines. Note that PixelNeRF, NeuralBody, and InstantNVR are unsuitable for novel pose synthesis.

Table 1. Quantitative Novel Pose evaluation of our GauHuman and baseline methods on the ZJU_MoCap and MonoCap data sets. LPIPS* = 1000 $\times$ LPIPS. For a fair comparison, we do not conduct test-time optimization of SMPL parameters with images from the test set on InstantAvatar [4].

<table>
<thead>
<tr>
<th>Method</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>LPIPS↓</th>
<th>PSNR↑</th>
<th>SSIM↑</th>
<th>LPIPS↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>AN</td>
<td>28.64</td>
<td>0.952</td>
<td>47.74</td>
<td>30.67</td>
<td>0.981</td>
<td>19.14</td>
</tr>
<tr>
<td>AS</td>
<td>30.42</td>
<td>0.963</td>
<td>37.70</td>
<td>32.78</td>
<td>0.984</td>
<td>16.27</td>
</tr>
<tr>
<td>HumanNeRF</td>
<td>30.47</td>
<td>0.962</td>
<td>27.31</td>
<td>31.63</td>
<td>0.983</td>
<td>14.18</td>
</tr>
<tr>
<td>DVA</td>
<td>29.31</td>
<td>0.955</td>
<td>38.79</td>
<td>32.27</td>
<td>0.982</td>
<td>16.44</td>
</tr>
<tr>
<td>InstantAvatar</td>
<td>29.50</td>
<td>0.934</td>
<td>76.37</td>
<td>27.75</td>
<td>0.945</td>
<td>68.20</td>
</tr>
<tr>
<td>GauHuman</td>
<td>31.29</td>
<td>0.965</td>
<td>29.89</td>
<td>33.00</td>
<td>0.984</td>
<td>13.95</td>
</tr>
</tbody>
</table>

\(^1\)We also find that the implementation of Wigner D-matrix using Pytorch [10] is time-consuming due to the matrix exponential operation.

**Scale to in-the-wild datasets.** We generalize GauHuman to an in_the_wild monocular online video. We use EasyMoCap [16] to predict SMPL pose parameters and SAM [6] to predict human masks for the monocular online video. We sample 1 frame every 5 frames and collect 100 frames for training. To evaluate novel view and novel pose synthesis results, we follow the same setting as ZJU_Mocap and MonoCap. As shown in Fig. 1, GauHuman produces plausible results and surpasses the state-of-the-art InstantAvatar baseline. Note that the pose refinement module can help refine human pose parameters (e.g., foot) for accurate 3D human reconstruction. One example is shown in Fig. 2.

**Effects of the number of 3D Gaussians on quality and speed.** According to our experiments, increasing the number of 3D Gaussians leads to limited performance improvement, but consumes more computation time. For example, it takes 1 hour for 100k Gaussian points to converge to a 28.01 PSNR; while our GauHuman only needs 55s to achieve a 28.08 PSNR with 13k Gaussians.

**Number of 3D Gaussians across different datasets.** According to our analysis, the number of 3D Gaussians varies on different datasets, depending on the resolution and accuracy of SMPL and camera parameters. For six subjects
Using constant KL to perform only splitting and cloning operations. We experiment on sequence 386 of the Zju_Mocap dataset. If only constant KL is used to perform split and clone operations, this will lead to a large number (up to 200k) of 3D Gaussians with a final performance PSNR of 27.88 (vs 28.08 PSNR of our GauHuman). The magnitude of the scaling matrix and gradients of positions are also important metrics for performing split or clone operations.

Convergence speed of initializing the scene with 13k points from SMPL (initial points can share same points from SMPL vertices) without performing splitting, cloning, and pruning operations. We ablate the experiments on sequence 386 of the Zju_Mocap dataset and find that it takes about 6 times more time to converge to a worse performance (PSNR: 27.35) than ours (PSNR: 28.08) when initializing the scene with 13k points without performing splitting, cloning, and pruning operations.

Additional experiments on DNA-Rendering We further evaluate the performance of our GauHuman and two representative baseline methods on a DNA-Rendering data set. We select two sequences (0012_09 and 0025_11 from part 1) from the DNA-Rendering data set and collect 100 frames for training. Similar to ZJU_MoCap and MonoCap, one camera is used for training. For evaluation purposes, we use four nearby camera views as testing views. As shown in Tab. 2 and Fig. 3, AS [13] and InstantAvatar [4] struggle to produce photorealistic renderings due to the complex clothing and fast-moving human actors recorded on the DNA-Rendering data set. In comparison, our GauHuman learns high-quality 3D human performers with fast training and rendering speed, which verifies the flexibility and efficiency of 3D Gaussian Splatting.

Table 2. Quantitative comparison of our GauHuman and baseline methods on the DNA-Rendering data set. LPIPS$^*$ $= 1000 \times$ LPIPS. Frames per second (FPS) is measured on an RTX 3090. For a fair comparison, we do not conduct test-time optimization of SMPL parameters with images from the test set on GART [8].

![Figure 3. Novel view synthesis results produced by our GauHuman and baseline methods on DNA Rendering data set. The bottom lines show the training time and rendering speed of each method on the DNA Rendering data set. Zoom in for the best view.](image)

### Table 3. Quantitative comparison of our GauHuman and GART on the ZJU_MoCap data set. LPIPS$^*$ $= 1000 \times$ LPIPS. Frames per second (FPS) is measured on an RTX 3090.

<table>
<thead>
<tr>
<th>Method</th>
<th>ZJU_MoCap (Avg)</th>
<th>DNA-Rendering</th>
<th>SSIM↑</th>
<th>LPIPS$^*$↓</th>
<th>Train FPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>GART</td>
<td>30.91</td>
<td>0.9615</td>
<td>31.83</td>
<td>53.4k</td>
<td>3m 77</td>
</tr>
<tr>
<td>GauHuman (Ours)</td>
<td>31.34</td>
<td>0.9647</td>
<td>30.51</td>
<td>11.8k</td>
<td>1m 189</td>
</tr>
<tr>
<td>GART</td>
<td>31.90</td>
<td>0.9747</td>
<td>18.8</td>
<td>55.0k</td>
<td></td>
</tr>
<tr>
<td>GauHuman (Ours)</td>
<td>32.24</td>
<td>0.9757</td>
<td>18.9</td>
<td>12.6k</td>
<td></td>
</tr>
<tr>
<td>GART</td>
<td>33.50</td>
<td>0.9669</td>
<td>29.9</td>
<td>51.4k</td>
<td></td>
</tr>
<tr>
<td>GauHuman (Ours)</td>
<td>33.72</td>
<td>0.9693</td>
<td>29.0</td>
<td>13.1k</td>
<td></td>
</tr>
<tr>
<td>GART</td>
<td>27.74</td>
<td>0.9518</td>
<td>40.3</td>
<td>52.9k</td>
<td></td>
</tr>
<tr>
<td>GauHuman (Ours)</td>
<td>28.19</td>
<td>0.9564</td>
<td>39.3</td>
<td>9.9k</td>
<td></td>
</tr>
<tr>
<td>GART</td>
<td>31.92</td>
<td>0.9637</td>
<td>32.6</td>
<td>51.6k</td>
<td></td>
</tr>
<tr>
<td>GauHuman (Ours)</td>
<td>32.27</td>
<td>0.9669</td>
<td>30.2</td>
<td>11.3k</td>
<td></td>
</tr>
<tr>
<td>GART</td>
<td>29.34</td>
<td>0.9540</td>
<td>37.9</td>
<td>51.7k</td>
<td></td>
</tr>
<tr>
<td>GauHuman (Ours)</td>
<td>30.24</td>
<td>0.9584</td>
<td>35.2</td>
<td>11.0k</td>
<td></td>
</tr>
<tr>
<td>GART</td>
<td>31.08</td>
<td>0.9577</td>
<td>31.5</td>
<td>57.7k</td>
<td></td>
</tr>
<tr>
<td>GauHuman (Ours)</td>
<td>31.42</td>
<td>0.9611</td>
<td>30.6</td>
<td>12.8k</td>
<td></td>
</tr>
</tbody>
</table>

Comparison with concurrent work GART [8] Our concurrent work GART [8] also extends Gaussian Splatting to 3D human modelling with monocular videos. It achieves comparable novel view synthesis performance when compared with state-of-the-art baseline methods on ZJU_MoCap data set while improving the rendering speed to 77 FPS with the efficient 3D Gaussian Splatting technique. We reproduce the result of GART with their released code and show the comparison results in Tab. 3. For a fair comparison, we do not conduct test-time optimization of SMPL parameters...
with images from the test set on GART \cite{8}. In comparison with GART, our GauHuman produces slightly better novel view synthesis performance with both faster training (1m vs. 3m) and rendering (189FPS vs. 77FPS) speed. Specifically, we achieve fast optimization of GauHuman by initializing and pruning 3D Gaussians with 3D human prior, while splitting/cloning via KL divergence guidance, along with a novel merge operation for further speeding up. Notably, without sacrificing rendering quality, GauHuman can fast model the 3D human performer with \sim 13k 3D Gaussians.

References


\[20\] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelNeRF: Neural radiance fields from one or few images. In CVPR, 2021.