# GPS-Gaussian: Generalizable Pixel-wise 3D Gaussian Splatting for Real-time Human Novel View Synthesis

Supplementary Material

We present the visualization of opacity maps (Sec. 7) and scaling maps (Sec. 8), performance under randomly placed camera setup (Sec. 9), run-time comparison (Sec. 10), network architecture (Sec. 11) and live demo setting (Sec. 12).

## 7. Visualization of Opacity Maps

As mentioned in Sec. 5.4, the joint regression with Gaussian parameters eliminates the outliers by predicting an extremely low opacity for the Gaussian points centered at these positions. The visualization of opacity maps is shown in Fig. 5. Since the depth prediction works on low resolution and upsampled to full image resolution, the drastically changed depth in the margin areas causes ambiguous predictions (*e.g.* the front and rear placed legs of the girl and the crossed arms of the boy in Fig. 5). These ambiguities lead to rendering noise on novel views when using a point cloud rendering technique. Thanks to the learned opacity map, the low opacity values make the outliers invisible in novel view rendering results, as shown in Fig. 5 (e).

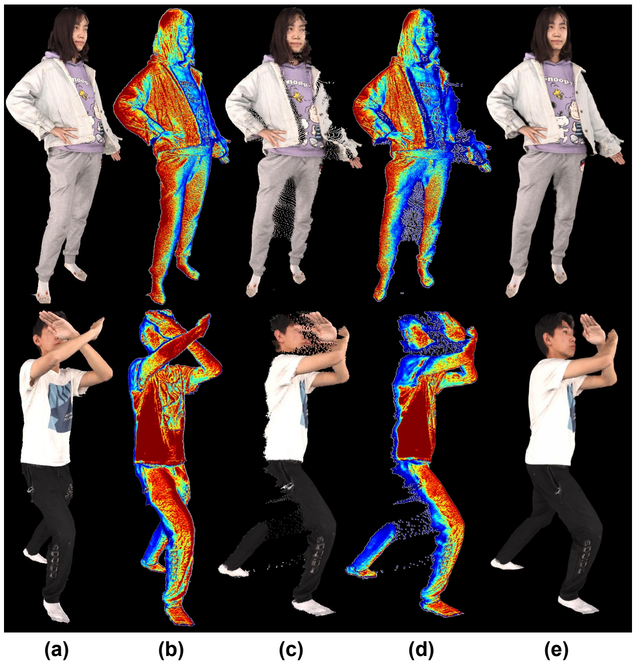

Figure 5. **Visualization of opacity maps.** (a) One of the source view images. (b) The predicted opacity map related to (a). (c)/(d) The directly projected color/opacity map at novel viewpoint. (e) Novel view rendering results. A cold color in (b) and (d) represents an opacity value near 0, while a hot color near 1. The low opacity values predicted for the outliers make them invisible.

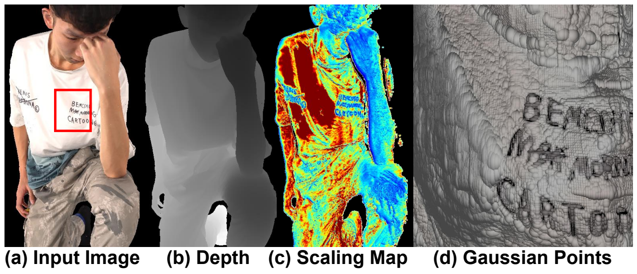

Figure 6. **Visualization of scaling map and the shape of Gaussian points.** (a) One of the source view images. (b) The depth of (a). (c) The scaling map shown in heat map, where a hotter color represents a larger value. (d) The zoom-in Gaussian points of the boxed area in (a). The depth and scaling map are normalized.

## 8. Visualization of Scaling Maps

The visualization of the scaling map (mean of three axes) in Fig. 6 (c) indicates that the Gaussian points with lower depth roughly have smaller scales than the distant ones. However, the scaling property is also impacted by comprehensive factors. For example, as shown in Fig. 6 (c) and (d), fine-grained textures or high-frequency geometries lead to small-scaled Gaussians.

## 9. Randomly Placed Camera Setup

We test our method with a randomly placed camera setup in Fig. 7. The model trained under a uniformly placed 8-camera setup in Sec. 5 shows a strong generalization capability to random camera setup with a pitch in range of $[-20°, +20°]$ and yaw in range of $[-25°, +25°]$. However, rendering additional synthetic data covering more general camera setups to re-train the model is a better choice for achieving improved performance in such cases.

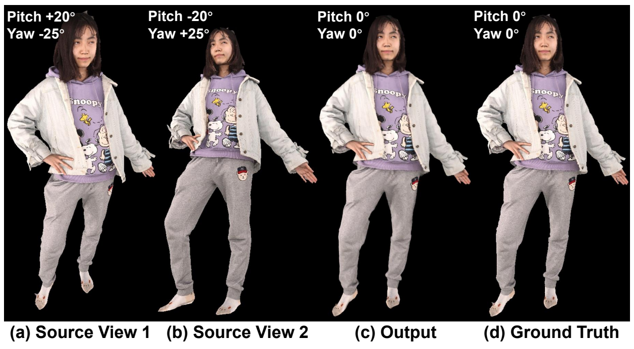

Figure 7. **Result on randomly placed camera setup.** (a) and (b) are the source view images with an extreme pitch and yaw. (c) is the novel view rendering result. (d) is the novel view ground truth.

## 10. Run-time Comparison

We compare the run-time of the proposed GPS-Gaussian with baseline methods. As illustrated in Table 4, the overall run-time can be generally divided into two parts which correlate to the source views and the desired novel view respectively. The source view correlated computation in FloRen [44] refers to coarse geometry initialization while the key components, the depth and flow refinement networks, operate on novel viewpoints. IBRNet [53] uses transformers to aggregate multi-view cues at each sampling point aggregated to the novel view image plane, which is time-consuming. ENeRF [17] constructs two cascade cost volumes on the targeted novel viewpoint, then predicts the novel view depth followed by a depth-guided sampling for volume rendering. Once the target viewpoint changes, these methods need to recompute the novel view correlated modules. However, the computation on source views dominates the run-time of GPS-Gaussian, which includes binocular depth estimation and Gaussian parameter map regression. Given a target viewpoint, it takes only 0.8 ms to render the 3D Gaussians to the desired novel view. This allows us to render multiple novel views simultaneously, which caters to a wider range of applications such as holographic displays. Suppose that $n = 10$ novel views are required concurrently, it takes our method $T = T_{src} + n \times T_{novel} = 35ms$ to synthesize, while $124ms$ for FloRen and $1261ms$ for ENeRF.

Table 4. **Run-time comparison.** We report the run-time correlated to the source views and each novel view on an RTX 3090 GPU. All methods take two $1024 \times 1024$ source images as input. Our method can render multiple novel views concurrently in real-time.

| Methods | Source view | Novel view (per view) |
|---|---|---|
| FloRen [44] | 14 ms | 11 ms |
| IBRNet [53] | 5 ms | 4000 ms |
| ENeRF [17] | 11 ms | 125 ms |
| Ours | 27 ms | 0.8 ms |

## 11. Network Architecture

As shown in Fig. 8, the network architecture of the proposed GPS-Gaussian is composed of (1) image encoder, (2) depth estimator, and (3) Gaussian parameter predictor.

**Image Encoder.** The image encoder $\mathcal{E}_{img}$ is applied to both source images and maps each of them to a set of dense feature map $\{\mathbf{f}_l^s\}_{s=1}^S$ as in Eq. 5. $\mathcal{E}_{img}$ has a similar architecture to the feature encoder in RAFT-Stereo [19]. We change the kernel size of the first convolution layer from $7 \times 7$ to $5 \times 5$ and replace all batch normalization with group normalization. Residual blocks and downsampling layers produce image features in 3 levels at $1/2$, $1/4$ and $1/8$ the input image resolution, with 32, 48 and 96 channels, respectively. The extracted features are further used to construct the correlation volume and regress the Gaussian parameters.

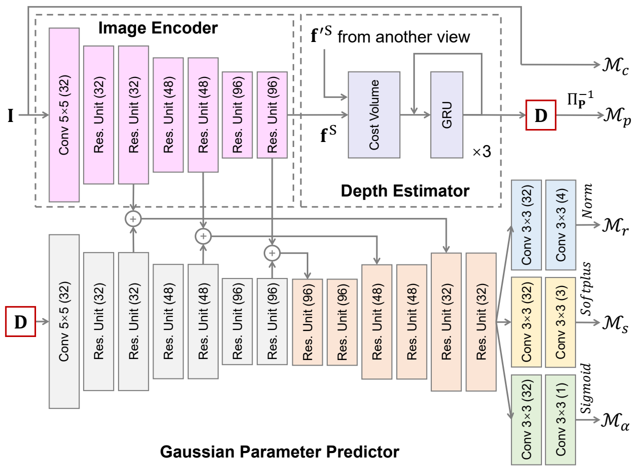

Figure 8. **Network architecture.** The proposed framework takes a source image as input to regress Gaussian parameter maps.

**Depth Estimator.** As mentioned in Sec. 4.1, The classic binocular stereo methods only estimate the depth for 'reference view'. while the 'target view' feature is only used to construct the cost volume but is not involved in further depth estimation or iterative refinement. By making the image encoder independent from the depth estimator, and re-indexing the correlation volume $\mathbf{C}$ for both lookup procedures, we realize a compact and parallelized implementation that results in a decent efficiency increase exceeding 30%. For the refinement module, we set $T = 3$ considering the trade-off between the performance and the cost.

**Gaussian Parameter Predictor.** This module is composed of a depth encoder $\mathcal{E}_{depth}$ and a U-Net like Gaussian parameter decoder $\mathcal{D}_{parm}$. $\mathcal{E}_{depth}$ takes the predicted depth as input and has an identical architecture to the image encoder. Image features concatenated with depth features are aggregated to the Gaussian parameter decoder via skip connections. The decoded pixel-wise Gaussian feature $\mathbf{\Gamma}$ passes through three specific prediction heads to get rotation map $\mathcal{M}_r$, scaling map $\mathcal{M}_s$ and opacity map $\mathcal{M}_\alpha$, respectively. Meanwhile, the position map $\mathcal{M}_p$ is determined by the predicted depth map $\mathbf{D}$ and the color map $\mathcal{M}_c$ directly borrows from the RGB value of the input image.

## 12. Live Demo Setting

We present live demos on our project page, in which we capture source view RGB streams and synthesize novel views in one system. Due to the memory limit of RTX 3090 GPU, we connect the front 6 cameras (facing human subjects) to the computer, which are uniformly positioned in a circle of a 2-meter radius. GPS-Gaussian enables real-time high-quality rendering, even for challenging hairstyles and human-object or multi-human interactions. For a more in-depth exploration of our results, please visit our homepage: `shunyuanzheng.github.io/GPS-Gaussian`.