From Sparse to Dense: Camera Relocalization with Scene-Specific Detector from Feature Gaussian Splatting

Supplementary Material

A. Feature Gaussian Training

Our training strategy references Feature 3DGS [70]. To adapt the feature field for the localization task and improve robustness, we make the following modifications:

1. Thanks to the development of the CUDA accelerated rasterization tool gsplat [63], we can efficiently render feature maps while preserving the original feature dimensions, eliminating the need for the speed-up module proposed in Feature 3DGS for upsampling feature channels after rasterization. Specifically, the feature f stored in the Gaussian primitive g has the same dimension D as the feature map $F_t(I)$ extracted using the general local feature extractor. This also enables direct matching between the 2D query features and the 3D features of the Gaussian primitives.
2. The rendering process for the radiance field of Feature Gaussian is based on the alpha blending rasterization method [33]. Let $\mathcal{N}$ denote the set of Gaussians associated with a pixel, sorted in front-to-back order. The pixel color C is computed by blending the color c of Gaussians as follows:

$$C = \sum_{i \in \mathcal{N}} c_i \alpha_i T_i, \tag{8}$$

where T_i is the transmittance factor accounting for the accumulated opacity α of all preceding Gaussians, defined as:

$$T_i = \prod_{j=1}^{i-1} (1 - \alpha_j). \tag{9}$$

This alpha blending approach is also applied to render the feature field. However, due to the vector triangle inequality, directly accumulating features is unsuitable. To address this, we introduce L2 normalization into the alpha blending formula. Specifically, we normalize the Gaussian feature f before rasterization to mitigate the influence of feature magnitude, and we further normalize the accumulated features after rasterization. The final rendered feature F_s is therefore expressed as:

$$F_s = \text{norm}\left(\sum_{i=1}^{n} \text{norm}(f_i)\alpha_i T_i\right), \tag{10}$$

where $\text{norm}(\cdot)$ denotes the L2 normalization operation. This two-step normalization process ensures stability and robustness in the feature field training and rendering.

Module	Time (ms)
Feature Extraction	3.7
S.S.D.	6.4
Sparse Matching	17.4
Pose Estimation (Sparse)	15.8
Rasterization	23
Dense Matching	13.2
Pose Estimation (Dense)	72.8
Total	152.3

Table A. **Detailed Time Consumption Analysis.**

B. Matching-Oriented Sampling Algorithm

The algorithm of the matching-oriented sampling strategy is illustrated in Algorithm 1.

C. Qualitative Analysis

Challenging Cases. In Fig. B Fig. C, we present the localization results of STDLoc in challenging scenarios involving weak texture and varying illumination conditions. For illumination changes, we demonstrate sample cases from both the Cambridge Landmarks dataset and real scene, where STDLoc achieves precise localization results. In the weak texture scenario, we provide a comparative analysis with HLoc in the Stairs scenario from the 7-Scenes dataset. STDLoc extracts denser matches, enabling more accurate pose estimation.

Failure Cases. As shown in Tab. 5, the recall of (5m, 10°) in the dense stage is slightly lower than that in the sparse stage. The localization results for these failure cases are illustrated in Fig. A. The first column is the query image, the second column is the sparse matching result between the query image and landmarks, and the third column is the dense feature map of the query image and the feature map rendered based on the sparse stage localization pose.

The sparse stage successfully provides an accurate pose estimation, but the feature map rendered in the dense stage is indistinct, leading to the failure of dense matching. This lack of distinguishability in the dense feature map is caused by floaters in the scene, which is a common issue in 3DGS [21] scenes. Therefore, reducing floaters in the scene can be effective in minimizing these failure cases. In addition, this situation reflects the robustness of our sparse stage, which can effectively eliminate the influence of these

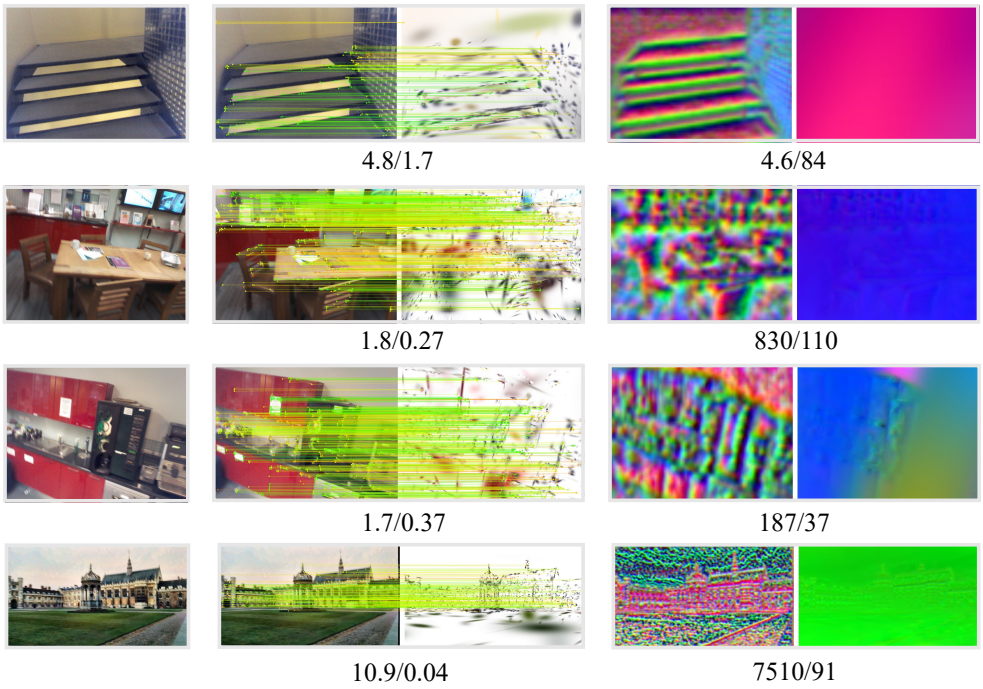

Figure A. **Failure Cases Visualization.** Visualization results of some examples where localization is successful in the sparse stage but failed in the dense stage. The translation error (*cm*) and rotation error (°) are indicated below the corresponding stage.

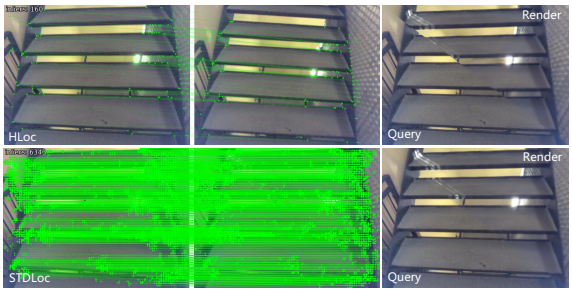

Figure B. **Comparison with HLoc in Weak Texture Scenario.**

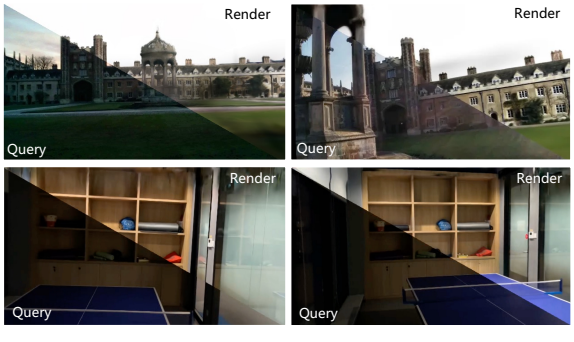

Figure C. **Localization Results in Illumination Changes Scenarios.**

floaters through the matching-oriented sampling strategy.

More Visualization Results. Fig. D presents the localization visualization results across all scenes for both the Cambridge Landmarks and 7-Scenes datasets. From left to right, each column shows the query image, its corresponding dense feature map, the sparse matching result between the query image and landmarks, the rendered feature map from the final dense stage, and the stitched result of the query image with the rendered image using the final pose. The visualization of the sparse matching results is achieved by rendering Gaussian landmarks based on the pose estimated in the sparse stage, followed by drawing the matches.

The third column of the figure demonstrates that our sparse stage achieves robust 2D-3D matching results. This is attributed to our proposed matching-oriented sampling strategy and scene-specific detector. Furthermore, the second-to-last column demonstrates the capability to learn the feature field using Feature Gaussian. As shown in the last column, the rendered image aligns precisely with the query image, highlighting the high accuracy of our localization method. By leveraging the learned feature field, our approach exhibits remarkable robustness against illumination changes and weak texture.

Algorithm 1 Matching-Oriented Sampling Algorithm

Require: Gaussians $\mathcal{G}$, training images $\{I\}$, feature maps $\{F_t(I)\}$, anchor number n, nearest neighbors k
Ensure: Sampled landmarks $\tilde{\mathcal{G}}$

for each Gaussian $g \in \mathcal{G}$ **do** ▷ Assign scores for each Gaussian
  $f \leftarrow \text{norm}(g.feature)$ ▷ Normalize Gaussian feature
  $\mathcal{V} \leftarrow \{\text{The set of images where } g \text{ is visible}\}$
  $s \leftarrow 0$
  for each Image $I \in \mathcal{V}$ **do**
    $(u, v) \leftarrow \text{Project}(g.center, I)$ ▷ The pixel coordinates of the projected Gaussian center
    $f_{\text{img}} \leftarrow \text{norm}(\text{GridSample}(F_t(I), (u, v)))$ ▷ Extract 2D feature using bilinear interpolation
    $s \leftarrow s + \langle f, f_{\text{img}} \rangle$
  end for
  $g.score \leftarrow s/|\mathcal{V}|$ ▷ Average score across images
end for
$\mathcal{A} \leftarrow \text{RandomSampling}(\mathcal{G}, n)$ ▷ Randomly sample anchors
$\tilde{\mathcal{G}} \leftarrow \emptyset$
for each anchor $a \in \mathcal{A}$ **do** ▷ Anchor-guided selection
  $\mathcal{N}_a \leftarrow \text{FindkNearestNeighbors}(a, \mathcal{G}, k)$
  $g^* \leftarrow \arg\max_{g \in \mathcal{N}_a} g.score$ ▷ Select Gaussian with the highest score among neighbors
  $\tilde{\mathcal{G}} \leftarrow \tilde{\mathcal{G}} \cup \{g^*\}$
end for

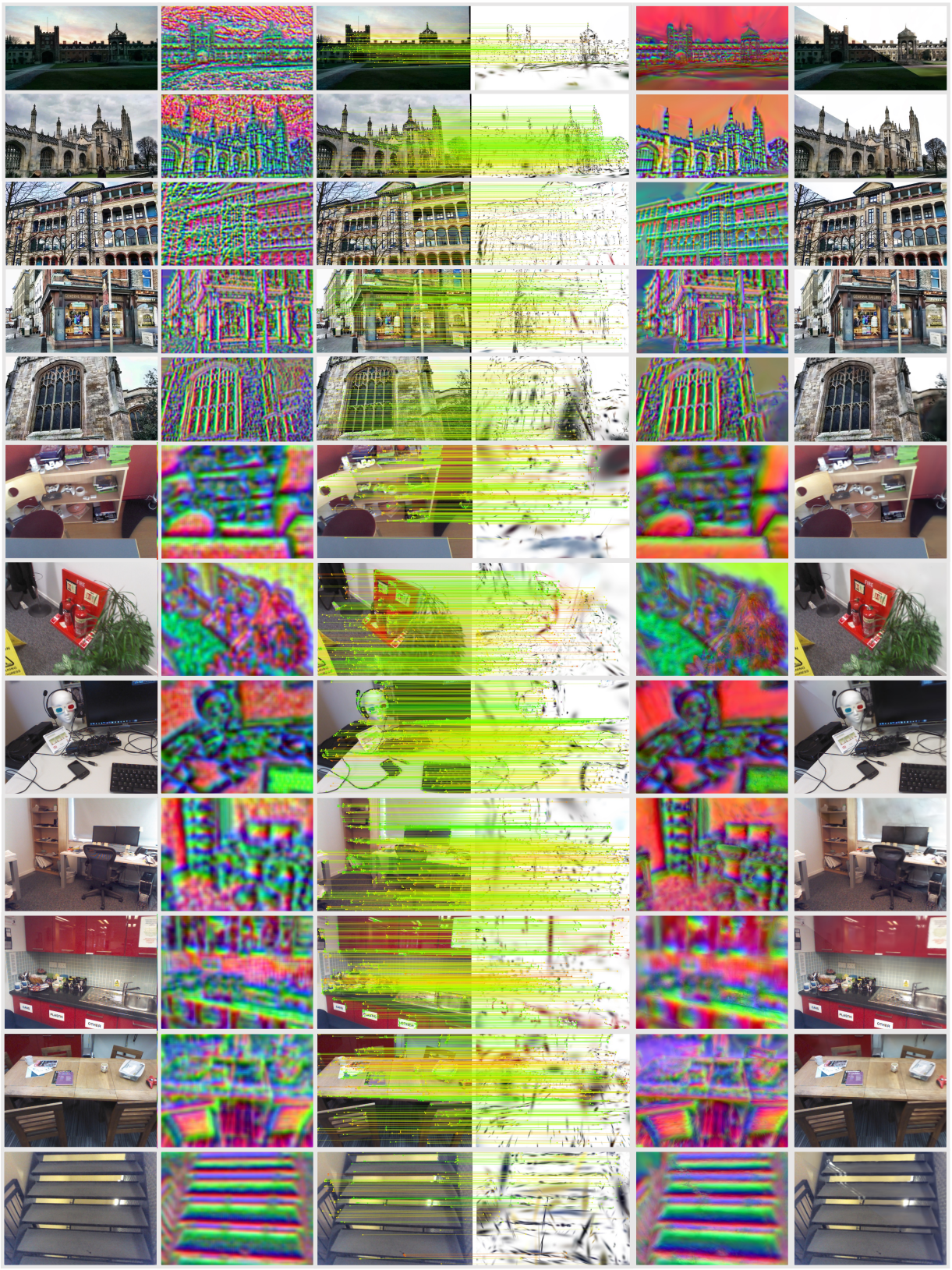

Figure D. **More Visualizations.** We show all scenes on both the Cambridge Landmarks and 7-Scenes datasets.