SparsePose: Sparse-View Camera Pose Regression and Refinement

Suppemental Material

A. Further Ablation study

A.1. LSTM iterations

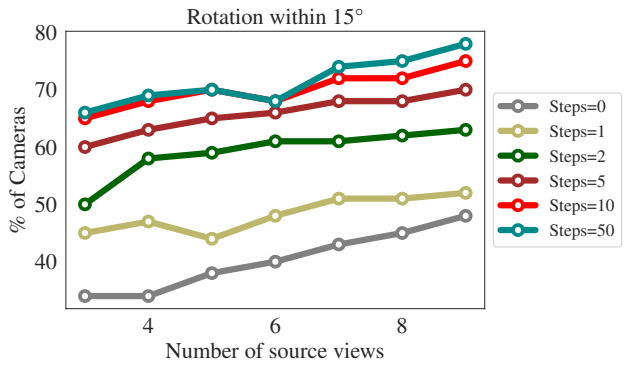

Figure 10. **Ablation results varying the number of LSTM steps.**

We add an additional ablation experiment over the number of steps required for the LSTM. We vary the number of LSTM iterations between 0 and 50, and report the percentage of cameras that were predicted between 15° of ground truth. We report the results in Figure 10. As previously noted, all the experiments were performed with 10 LSTM iterations, which balances out speed and accuracy of predictions. While we observe slight improvements with 50 LSTM iterations, overall, using 10 LSTM iterations performs similarly.

A.2. Timing Analysis

	Time (seconds)
HLOC [56]	38s
COLMAP + SIFT [39, 58]	18s
Pix. Perfect SFM [37]	55s
RelPose [79]	48s
SRT [55]	**2.7s**
MetaPose [64]	**2.6s**
SparsePose	**3.6s**

Table 1. **Time (in seconds) to perform registration on a single sequence with 9 source images.** To enable fair comparison between all methods, only sequences where *all* baseline methods were able to register all the source images were included in the analysis. Each of the methods are run on the same NVIDIA A6000 for fair comparison.

A.3. Different rotation threshold

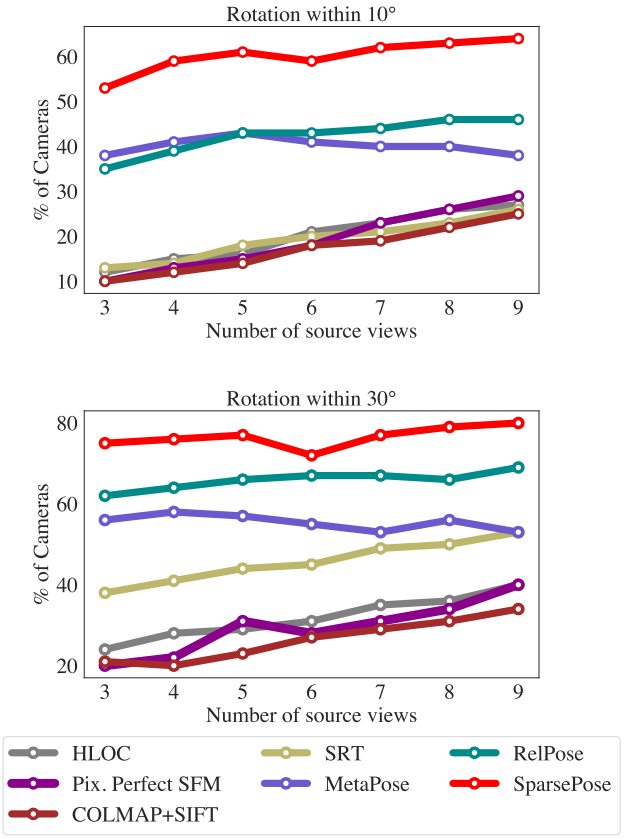

Figure 11. **Evaluating on the percentage of cameras accurately predicted with 10° and 30° thresholds.**

B. Baseline details

MetaPose. For the MetaPose baseline [64], we used the initialization from SparsePose, and adapt the MetaPose architecture in the officially released code to perform pose updates on the current updates. We do not utilize the human-specific information proposed, since our data is more general than the data used to evaluate MetaPose. We train MetaPose on the same subset of CO3D [52] that was used in training SparsePose.

Scene Representation Transformer (SRT). SRT [55] proposes to learn a prior over the 3D geometry from data implicitly by learning a "set-latent scene representation" from sparse or dense images of the scene using transformer encoder and decoder layers. Although SRT does not learn a

direct 3D geometry of the scene, it does learn a prior over the 3D geometry, as it can perform novel-view synthesis. To adapt SRT to our evaluation protocol, we add an additional 3-layer MLP that is trained to predict the relative rotations and translations for the input image sequence. We train the "unposed" version of SRT, and add an additional MLP (with 3-hidden layers) to predict rotations and translations, and we train the method with the the same training dataset and loss as SparsePose. For training, we use the same hyperparameters as suggested in the original paper.

Heirarchical Localization (HLOC). For HLOC [56], we use the officially released code from https://github.com/cvg/Hierarchical-Localization, which uses SuperPoint [14] for generating correspondances and SuperGlue [57] for image matching.

COLMAP + SIFT. For the COLMAP baseline with SIFT features, we used the officially released code from HLOC https://github.com/cvg/Hierarchical-Localization, which supports SIFT image features.

RelPose. For the RelPose baseline, we used the officially released code from https://github.com/jasonyzhang/relpose, and trained on the same dataset used to train SparsePose. We use the default RelPose hyperparameters.

Pixel Perfect SFM. For the Pixel Perfect SFM [37], we used the officially released codebase from https://github.com/cvg/pixel-perfect-sfm.

C. More implementation details

Hyperparameter	Value
Number of training steps	500,000
Number of source views during step	$\mathcal{U}[3, 9]$
Number of sequences sampled per step	1
Choice of $\mathcal{E}_{\text{init}}$	DINO [8]
Architecture of $\mathcal{E}_{\text{init}}$	ViT-B/8 [15]
Number of heads $\mathcal{T}_{\text{init}}$	8
Number of heads $\mathcal{T}_{\text{refine}}$	2
Number of hidden dim. $\mathcal{T}_{\text{init}}$, $\mathcal{T}_{\text{refine}}$	2048
Number of hidden layers $\mathcal{N}_{\text{init}}$, $\mathcal{N}_{\text{pose}}$	3
Number of hidden dim. $\mathcal{N}_{\text{init}}$, $\mathcal{N}_{\text{pose}}$	512
Activation for $\mathcal{N}_{\text{init}}$, $\mathcal{T}_{\text{init}}$, $\mathcal{T}_{\text{refine}}$, $\mathcal{N}_{\text{pose}}$	GELU
Number of LSTM steps	10
Optimizer	Adam [30]
Learning rate	10^{-4}
Learning rate decay iterations	250,000
Learning rate decay factor	10

Table 2. **Hyperparameters and implementation details.** These hyperparameters are shared through all experiments for SparsePose, unless stated otherwise.

D. Further evaluation of results

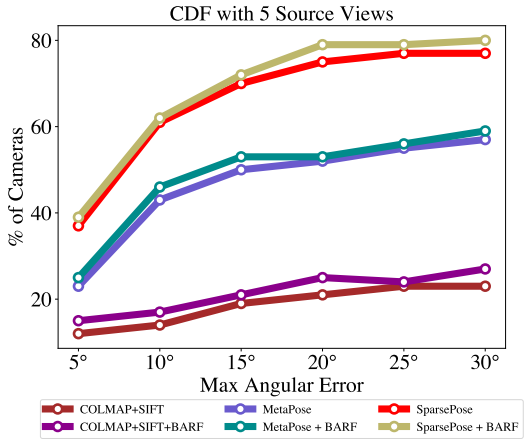

Figure 12. CDF of the % of cameras within a max angular error before and after finetuning the camera poses with BARF.

Figure 12 % of cameras within the max angular error before and after finetuning with BARF [34] for 5 source views, using a pretrained category-centric NeRFormer [52]. BARF finetuning generally improves poses across the board, though the same general trends remain.

Furthermore, we also plot the CDF showing the % of cameras predicted that are within maximum angular error, for all considered source views, in Figure 13. We can see that SparsePose continues to outperform classical and learning based SfM methods by a large margin.

E. Qualitative results

In Figure 14 we provide additional qualitative results with different numbers of source views and visualize the predicted camera poses by our method compared to baselines. We also include additional qualitative novel-view synthesis results for different categories over different numbers of source views in Figure 15 and Figure 16. In both cases, we see that SparsePose predicts more accurate camera poses, resulting in higher quality novel-view synthesis compared to other baselines.

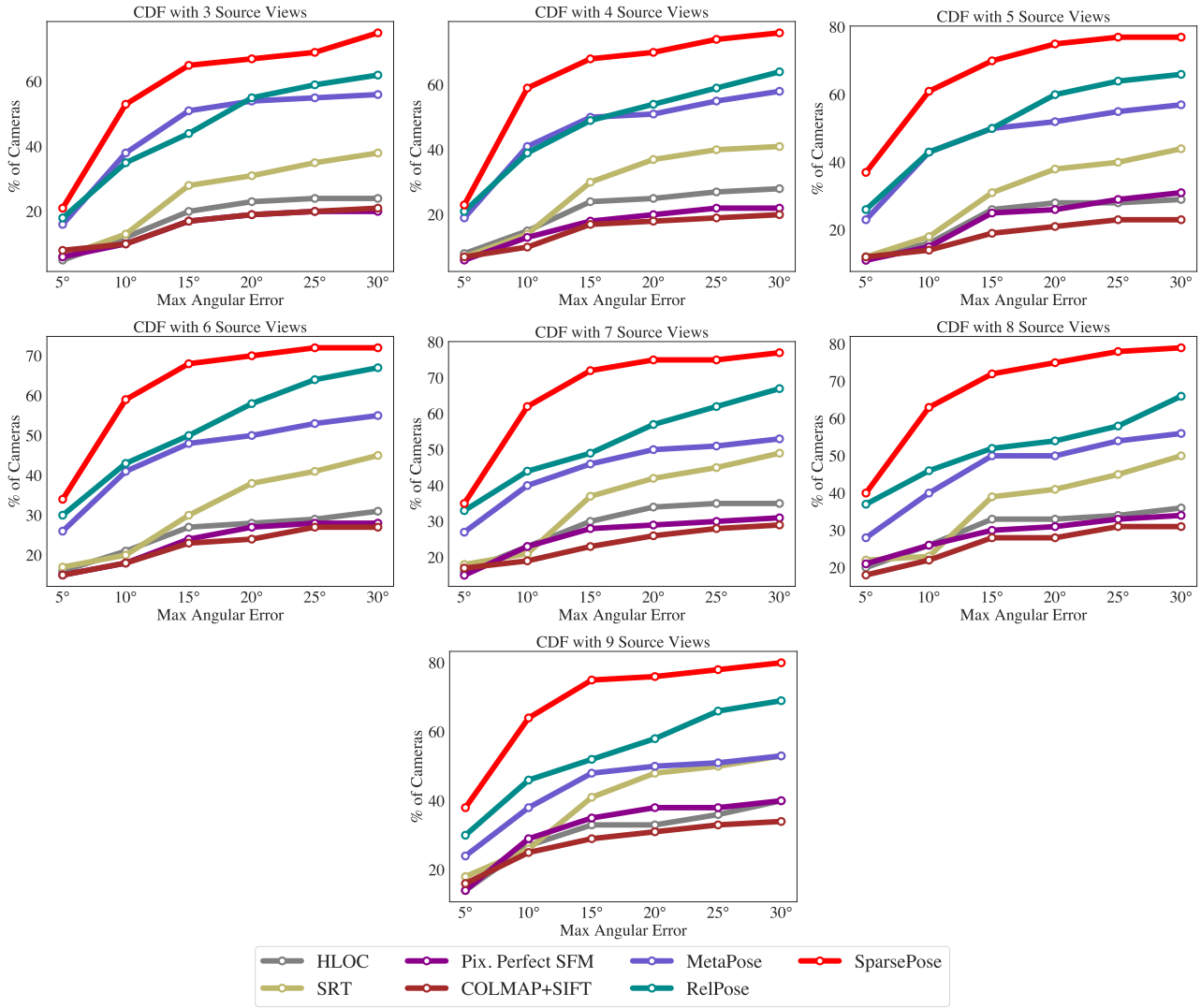

Figure 13. CDF for different number of source views and the % of cameras less than the maximum angular error.

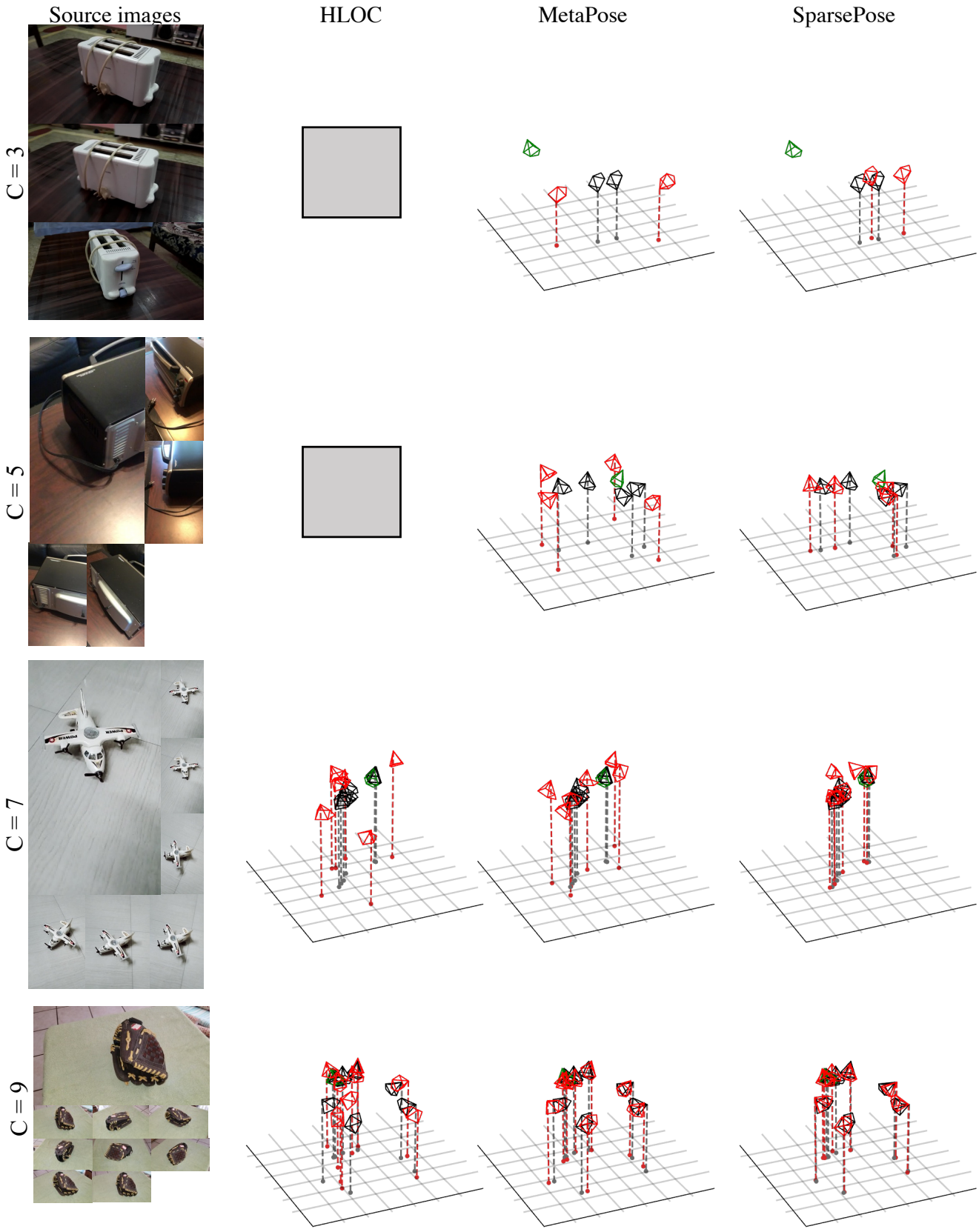

Figure 14. **More qualitative results for the predicted camera poses.** The camera centers are projected to the $x - y$ plane for easy visual comparison. The ground truth poses are shown in black, predicted poses in red, and the first camera for each sequence (used to align predictions) in green. Gray boxes indicate failure to converge.

Figure 15. **More qualitative renders from a sparse set of unposed images.**

Figure 16. **More qualitative renders from a sparse set of unposed images.**