

Supplementary Materials

Anonymous Submission (#432)

1 More Pipeline Diagrams

1.1 Data Generation Pipeline Diagram

Our training data generation pipeline is shown in Figure 1, which mainly contains scene Structure-from-Motion (SfM) reconstruction and training waypoints visual localization..

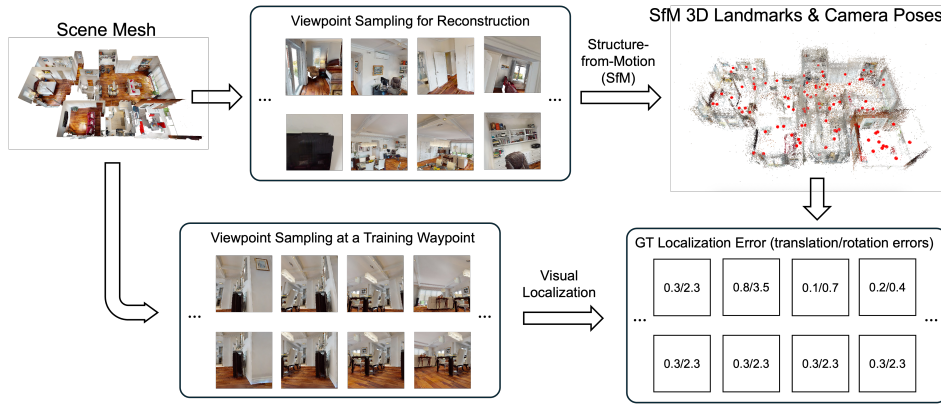


Figure 1: Data Generation.

1.2 Data Preprocessing Diagram

The data preprocessing pipeline is demonstrated in Figure 2. This pipeline first removes highly uncertain points from the 3D landmarks and implicitly encodes the training waypoint coordinate into the data via egocentric transformation. Finally, it applies a bounding box cropping on the point cloud at the training waypoint to enforce the model to focus on nearby information, and also works as a normalization.

2 Additional Experiments

2.1 Experiments on Sparsified Data

We conducted experiments on data sparsified at various levels to compare our method (ActLoc-Bin) with the baseline Learning-Where-to-Look (LWL) [1]. As shown in Table 1, ActLoc-Bin outperforms LWL across most sparsification levels, with the greatest advantage under extreme sparsification.

3 Qualitative Results

3.1 Viewpoint Selection Visualization

Some visualizations of the viewpoint selection task are presented in Figure 3, where red cells stand for bad viewpoints and blue cells for good viewpoints.

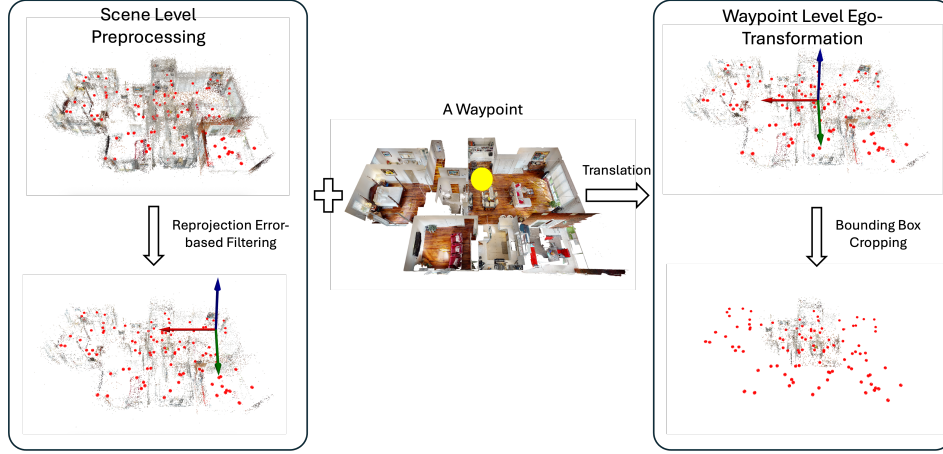


Figure 2: Data Preprocessing.

Sparsification Level	Method	0.1 m/1°	0.25 m/2°	0.5 m/5°	5 m/10°
0%	LWL	90.04	90.98	91.17	91.92
	ActLoc-Bin	92.11	92.48	92.86	93.23
10%	LWL	89.47	90.23	90.41	91.54
	ActLoc-Bin	91.17	91.73	92.11	92.29
20%	LWL	87.78	88.72	89.66	90.04
	ActLoc-Bin	90.79	91.54	92.29	92.67
30%	LWL	88.53	89.66	90.23	90.79
	ActLoc-Bin	89.66	90.23	90.79	91.17
40%	LWL	88.72	89.66	90.60	91.73
	ActLoc-Bin	90.23	90.98	91.17	91.54
50%	LWL	89.85	90.41	90.98	91.35
	ActLoc-Bin	89.29	90.04	90.98	91.54
60%	LWL	89.10	90.23	90.79	90.79
	ActLoc-Bin	88.35	89.66	90.04	90.41
70%	LWL	87.78	88.72	89.29	89.85
	ActLoc-Bin	87.22	88.72	89.10	89.29
80%	LWL	83.83	86.09	86.84	87.59
	ActLoc-Bin	86.84	87.59	88.16	88.91
90%	LWL	83.08	84.02	84.77	85.53
	ActLoc-Bin	86.84	87.59	88.16	88.91

Table 1: **Localization accuracy (%) under different sparsification levels** (percentage of camera poses and associated points removed) on 10 HM3D [2] scenes.

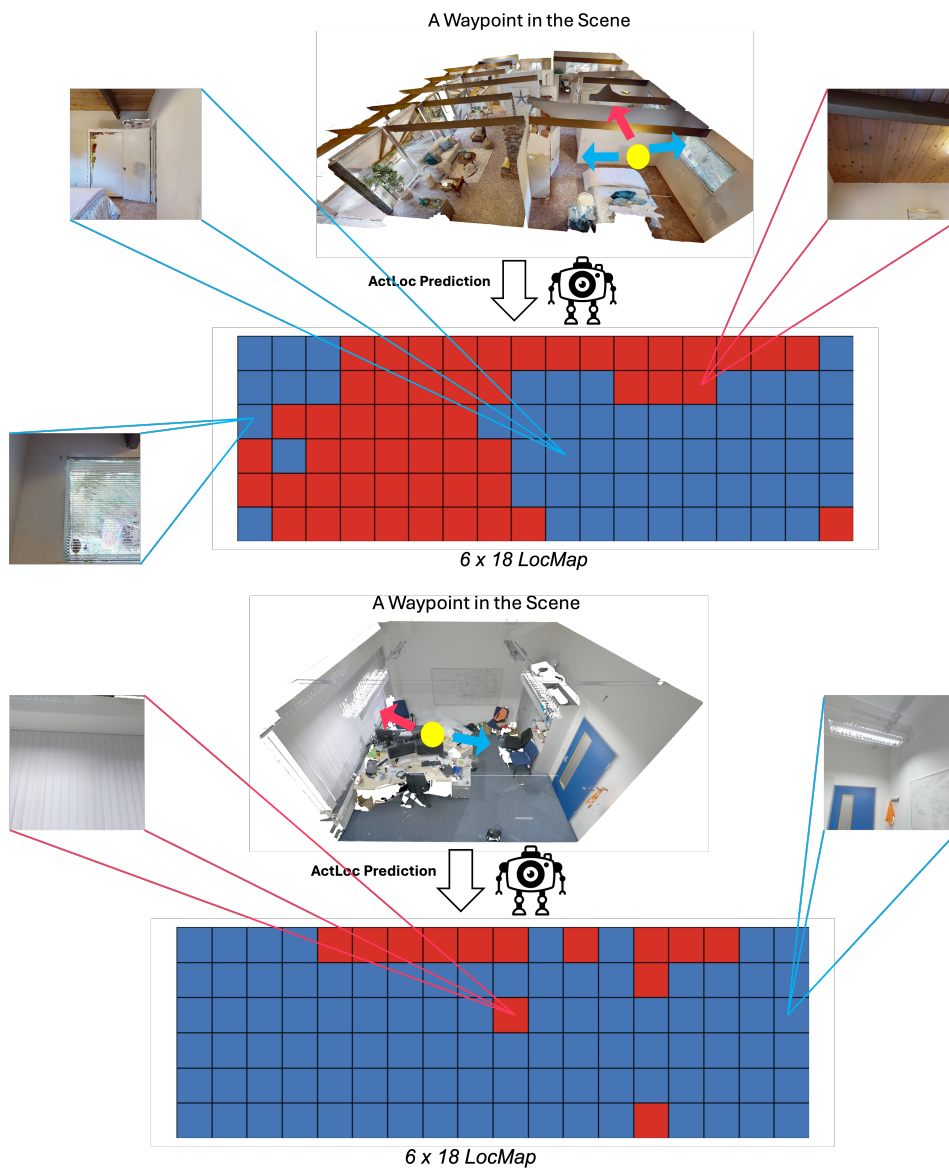


Figure 3: **Viewpoint Selection Visualization from the ScanNet [3].**

3.2 Path Planning Visualization

One of the path planning results in our HM3D test scenes is visualized in Figure 4.

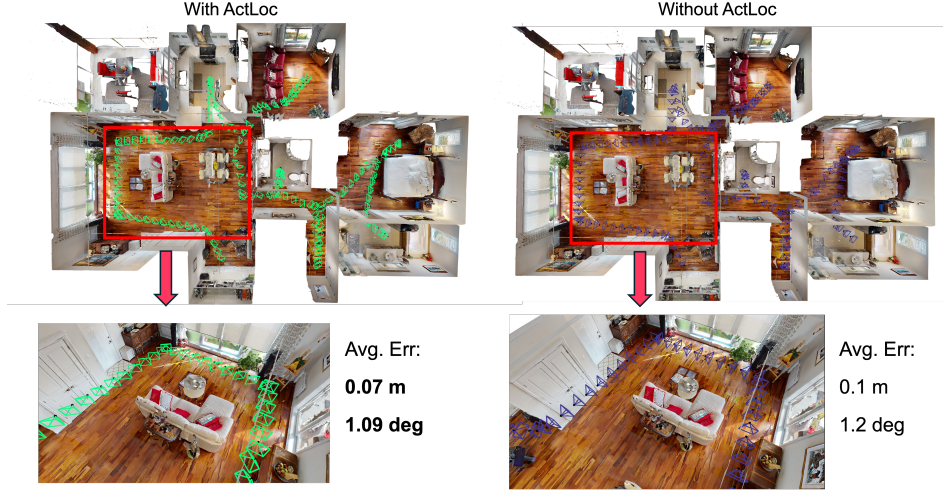


Figure 4: Path Planning Result of a HM3D Scene.

3.3 Inference Time Comparison Plot

Figure 5 compares the inference speed of our model and the baseline under three different prediction grid resolutions. ActLoc-Bin completes a forward pass in about 108 ms at all three resolutions, with under 2 ms of run-to-run variation. In contrast, LWL requires approximately 8.3 s, 27.5 s and 104.3 s on average for the 6×18 , 12×36 , and 24×72 grids, respectively. This corresponds to a speed-up of two to three orders of magnitude.

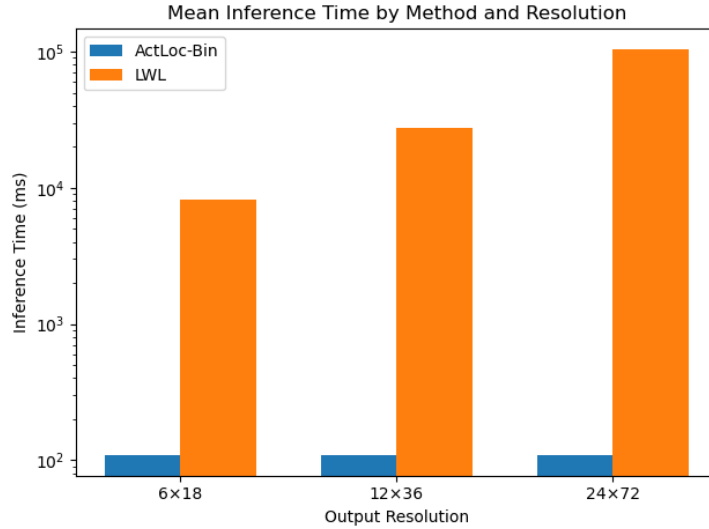


Figure 5: Comparison of mean inference time (ms) for ActLoc-Bin vs. LWL across output resolutions, plotted on a log scale.

3.4 Global Localization Score Map

ActLoc generates a localization map (LocMap) in a single forward pass, allowing rapid precomputation over an entire scene to support three-dimensional path planning. To illustrate, we sample dense waypoints at a height of 0.5 m, run LocMap for each, average the top five grid scores, and visualize the results as a heatmap in Fig. 6. The heatmap exposes clear spatial variations in localization difficulty, which a planner can exploit to guide the robot toward areas that are easier to localize.

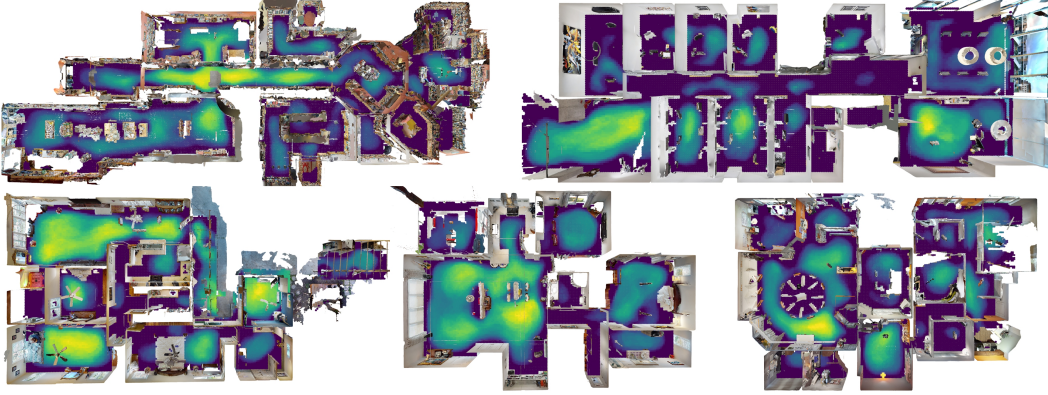


Figure 6: Overall Localization Score of Different Scenes.

4 Implementation Details

4.1 Model Architecture Specs

This subsection provides a more detailed specification of our network architecture, complementing the overview presented in the main paper. As described, our model processes egocentrically transformed camera pose sequences and landmark point clouds to predict visual localization accuracy.

1. **Input Embedding:** The raw camera pose features (7-dimensional: 3D center + 4D quaternion) and landmark features (6-dimensional: 3D coordinates + 3D normalized RGB) are independently projected into a shared 256-dimensional embedding space (where $d_{model} = 256$) using modality-specific linear layers.
2. **Dual-Branch Self-Attention Encoders:** Following embedding, each modality stream is processed by a dedicated Transformer [4] encoder stack to refine its representation through self-attention.
 - *Structure:* Each encoder stack consists of 6 identical Transformer encoder layers ($num_encoder_layers = 6$).
 - *Encoder Layer Configuration:* Each layer contains:
 - A multi-head self-attention mechanism with 8 attention heads ($nhead = 8$). The model dimension throughout the attention mechanism is 256.
 - A point-wise feed-forward network with an input and output dimension of 256, and an inner dimension of 1024 ($dim_feedforward = 1024$). The activation function used within this network is ReLU.
 - Layer normalization is applied *after* both the self-attention and feed-forward sub-modules (Post-LN). The epsilon value for layer normalization is $1e-5$.
 - Dropout with a rate of 0.1 ($dropout = 0.1$) is applied within the feed-forward network and after the attention output.
 - These encoders are designed to efficiently process variable-length sequences, accommodating differing numbers of camera poses or landmarks per scene.

3. **Bidirectional Cross-Attention Fusion:** To integrate information between the two modalities, the outputs from the self-attention encoders are fed into a bidirectional cross-attention block.
 - *Structure:* This block is composed of 4 stacked cross-attention layers (`num_cross_layers = 4`).
 - *Cross-Attention Layer Configuration:* In each layer, the pose features attend to the landmark features, and concurrently, the landmark features attend to the pose features. This is facilitated by multi-head attention mechanisms, also configured with 8 attention heads and operating on 256-dimensional features. Projections for query, key, and value are modality-specific. Similar to the self-attention encoders, layer normalization (Post-LN, with epsilon $1e-5$) and dropout (0.1) are applied.
4. **Aggregation and Multi-Layer Perceptron (MLP) Head:** As mentioned in the main paper, masked mean pooling is applied to the output sequences from the cross-attention block for each modality, yielding two fixed-size vectors. These vectors are then concatenated. The subsequent MLP head is structured as follows:
 - An initial linear layer takes the concatenated $2 \times 256 = 512$ -dimensional features and projects them to a 256-dimensional representation. A bias term is included in this layer.
 - This is followed by a ReLU activation function.
 - Layer normalization (with epsilon $1e-5$) is then applied to the activated features.
 - A final linear layer (the classifier) maps this 256-dimensional representation to the output logits. The total number of output elements from this layer is $N_{\text{classes}} \times 6 \times 18$, where N_{classes} is the number of accuracy levels predicted and is task-dependent (e.g., $N_{\text{classes}} = 2$ for binary classification). A bias term is included in this final layer.
 - These flat logits are then reshaped to the final output tensor of shape (`batch_size`, N_{classes} , 6, 18), corresponding to the predicted accuracy for each of the N_{classes} levels across the 6×18 viewing direction grid.

The model uses `bfloat16` precision during training and inference to leverage hardware acceleration for the attention computations, which is particularly beneficial for the underlying FlashAttentionV2 [5] implementation.

For an even clearer overview, the key architectural details and hyperparameters of our model are summarized in Table 4.1.

4.2 Training Setup

All model training and experiments were conducted on an NVIDIA RTX 4090 GPU with 24 GB of VRAM. The training objective was to minimize a weighted cross-entropy loss. The per-cell loss is defined as

$$\mathcal{L}_{\text{instance}}(\mathbf{z}, y) = -w_y \log\left(\frac{\exp(z_y)}{\sum_{c=0}^{K-1} \exp(z_c)}\right) \quad (1)$$

where

- $\mathbf{z} = (z_0, \dots, z_{K-1})$ are the model’s logits for the K classes,
- $y \in \{0, \dots, K-1\}$ is the true class index,
- w_y is the precomputed weight for class y , and
- $\mathcal{L}_{\text{instance}}$ is the loss for a single grid cell.

Component/Stage	Parameter/Aspect	Value/Specification
Input Processing		
Input Features	Camera Poses	7-dimensional (3D center + 4D quaternion)
	Landmark Point Cloud	6-dimensional (3D coordinates + 3D normalized RGB)
Input Embedding	Embedding Dimension (d_model)	256
	Projection	Modality-specific Linear Layers
Self-Attention Encoders (per modality branch)		
Encoder Stack	Number of Layers (num_encoder_layers)	6
Transformer Encoder Layer	Model Dimension (d_model)	256
	Attention Heads (nhead)	8
	Feed-Forward Inner Dimension (dim_feedforward)	1024
	Feed-Forward Activation	ReLU
	Layer Normalization	Post-LN
	Layer Normalization Epsilon	1e-5
	Dropout Rate	0.1
Cross-Attention Fusion Block		
Cross-Attention Stack	Number of Layers (num_cross_layers)	4
Cross-Attention Layer	Model Dimension (d_model)	256
	Attention Heads (nhead)	8
	Layer Normalization	Post-LN
	Layer Normalization Epsilon	1e-5
	Dropout Rate	0.1
Aggregation		
Method	Per Modality Stream	Masked Mean Pooling
MLP Head (Post-Aggregation & Concatenation)		
Concatenated Input	Dimension	$2 \times 256 = 512$
Fusion Linear Layer 1	Output Dimension	256
	Activation	ReLU
Layer Normalization	Epsilon	1e-5
Classifier Linear Layer	Output Dimension	$N_{\text{classes}} \times 6 \times 18$
Output		
Final Output Shape	Per Sample	$(N_{\text{classes}}, 6, 18)$ grid
General		
Numerical Precision	Training & Inference	bfloat16

Table 2: **Summary of Model Architecture Details and Hyperparameters.** N_{classes} denotes the task-dependent number of predicted accuracy levels.

Class weights were calculated from the inverse frequency of each class in the training set and then normalized to reduce class imbalance. Specifically, if

$$n_i = |\{m : y^{(m)} = i\}|$$

is the number of samples in class i , then

$$w_i = K \frac{1/n_i}{\sum_{j=0}^{K-1} (1/n_j)}, \quad i = 0, 1, \dots, K-1 \quad (2)$$

The key training hyperparameters are detailed in Table 4.2.

Category	Hyperparameter	Value/Setting
<i>General Training</i>		
	Number of Epochs	100
	Batch Size	4
	Number of Output Classes	2 or 4 (configurable)
	Automatic Mixed Precision (AMP)	Enabled
<i>Optimizer</i>		
	Type	AdamW
	Initial Learning Rate	1e-4
	Weight Decay	1e-2
<i>Learning Rate Scheduler</i>		
	Type	ReduceLROnPlateau
	Patience	3 epochs
	Reduction Factor	0.8
	Monitored Metric	Validation accuracy
<i>Early Stopping</i>		
	Patience	15 epochs

Table 3: **Training Hyperparameters.**

References

- [1] L. Di Giammarino, B. Sun, G. Grisetti, M. Pollefeys, H. Blum, and D. Barath. Learning where to look: Self-supervised viewpoint selection for active localization using geometrical information. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part LXXXVI*, page 188–205, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73015-3. doi:10.1007/978-3-031-73016-0_12. URL https://doi.org/10.1007/978-3-031-73016-0_12.
- [2] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. Turner, E. Underlander, W. Galuba, A. Westbury, A. X. Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*, 2021.
- [3] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.
- [4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. NIPS’17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.
- [5] T. Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.