EUROGRAPHICS 2024 / A. Bermano and E. Kalogerakis
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 2

Supplementary Material for HaLo-NeRF

Chen Dudai*[1], Morris Alper*[1], Hana Bezalel[1], Rana Hanocka[2], Itai Lang[2], Hadar Averbuch-Elor[1]

[1]Tel Aviv University [2]University of Chicago

1. HolyScenes – Additional Details

Landmarks and Categories Used.

Our benchmark spans three landmark building types (cathedrals, mosques, and a synagogue), from different areas around the world. We select scenes that have sufficient RGB imagery for reconstructing with [CZL*22]. The images were taken from IMC-PT 20 [Yi20] (*Notre-Dame Cathedral*, *St. Paul's Cathedral*), MegaDepth [LS18] (*Blue Mosque*), WikiScenes [WAESS21] (*Milan Cathedral*), and scraped from Wikimedia Commons using the WikiScenes data scraping procedure (*Badshahi Mosque* and *Hurva Synagogue*). The Notre-Dame cathedral has the most images in the dataset (3,765 images), and the *Hurva Synagogue* has the fewest (104 images). For semantic categories, we select diverse concepts of different scales. Some of these (such as *portal*) are applicable to all landmarks in our dataset while others (such as *minaret*) only apply to certain landmarks. As illustrated in Table 1, we provide segmentations of 3-4 semantic categories for each landmark; these are selected based on the relevant categories in each case (*e.g.* only the two mosques have minarets).

Annotation Procedure

We produce ground-truth binary segmentation maps to evaluate our method using manual labelling combined with correspondence-guided propagation. We first segment 110 images from 3-4 different categories from each of the six different scenes in our dataset, as shown in Table 1. We then estimate homographies between these images and the remaining images for these landmarks, using shared keypoint correspondences from COLMAP [SF16] and RANSAC. We require at least 100 corresponding keypoints that are RANSAC inliers; we also filter out extreme (highly skewed or rotated) homographies by using the condition number of the first two columns of the homography matrix. When multiple propagated masks can be inferred for a target image, we calculate each pixel's binary value by a majority vote of the warped masks. Finally, we filter these augmented masks by manual inspection. Out of 8,951 images, 6,195 were kept (along with the original manual seeds), resulting in a final benchmark size of 6,305 items. Those that were filtered are mostly due to occlusions and inaccurate warps. Annotation examples from our benchmark are shown in Figure 1.

Landmark	Portal	Window	Spire	Tower	Dome	Minaret	#Seed	#Seg
NDC	✓	✓	✗	✓	✗	✗	15	4048
MC	✓	✓	✓	✗	✗	✗	19	902
SPC	✓	✓	✗	✓	✓	✗	20	731
BAM	✓	✗	✗	✗	✓	✓	15	177
BLM	✓	✓	✗	✗	✓	✓	26	381
HS	✓	✓	✗	✗	✓	✗	15	66

Table 1: ***The HolyScenes Benchmark****, composed of the Notre-Dame Cathedral (NDC), Milan Cathedral (MC), St. Paul's Cathedral (SPC), Badshahi Mosque (BAM), Blue Mosque (BLM), and the Hurva Synagogue (HS). Above we report the set of semantic categories annotated for each landmark, chosen according to their visible structure. In the columns on the right, we report the number of initial manually segmented images (#Seed), and the final number of ground-truth segmentations after augmented with filtered warps (#Seg)*

2. Implementation Details

2.1. Augmenting the WikiScenes Dataset

The original WikiScenes dataset is as described in Wu *et al*. [WAESS21]. To produce training data for the offline stages of our system (LLM-based semantic distillation and V&L model semantic adaptation), we augment this cathedral-focused dataset with mosques by using the same procedure to scrape freely-available Wikimedia Commons collected from the root WikiCategory "Mosques by year of completion". The collected data contains a number of duplicate samples, since the same image may appear under different categories in Wikimedia Commons and is thus retrieved multiple times by the scraping script. In order to de-duplicate, we treat the image's filename (as accessed on Wikimedia Commons) as a unique identifier. After de-duplication, we are left with 69,085 cathedral images and 45,668 mosque images. Out of these, we set aside the images from landmarks which occur in HolyScenes (13,743 images total) to prevent test set leakage; the remaining images serve as our training data.

2.2. LLM-Based Semantic Distillation

To distill the image metadata into concise textual pseudo-labels, we use the instruction-tuned language model Flan-T5 [CHL*22], selecting the 3B parameter Flan-T5-XL variant. The model is given the image caption, related key-words, and filename, and outputs a

* Denotes equal contribution

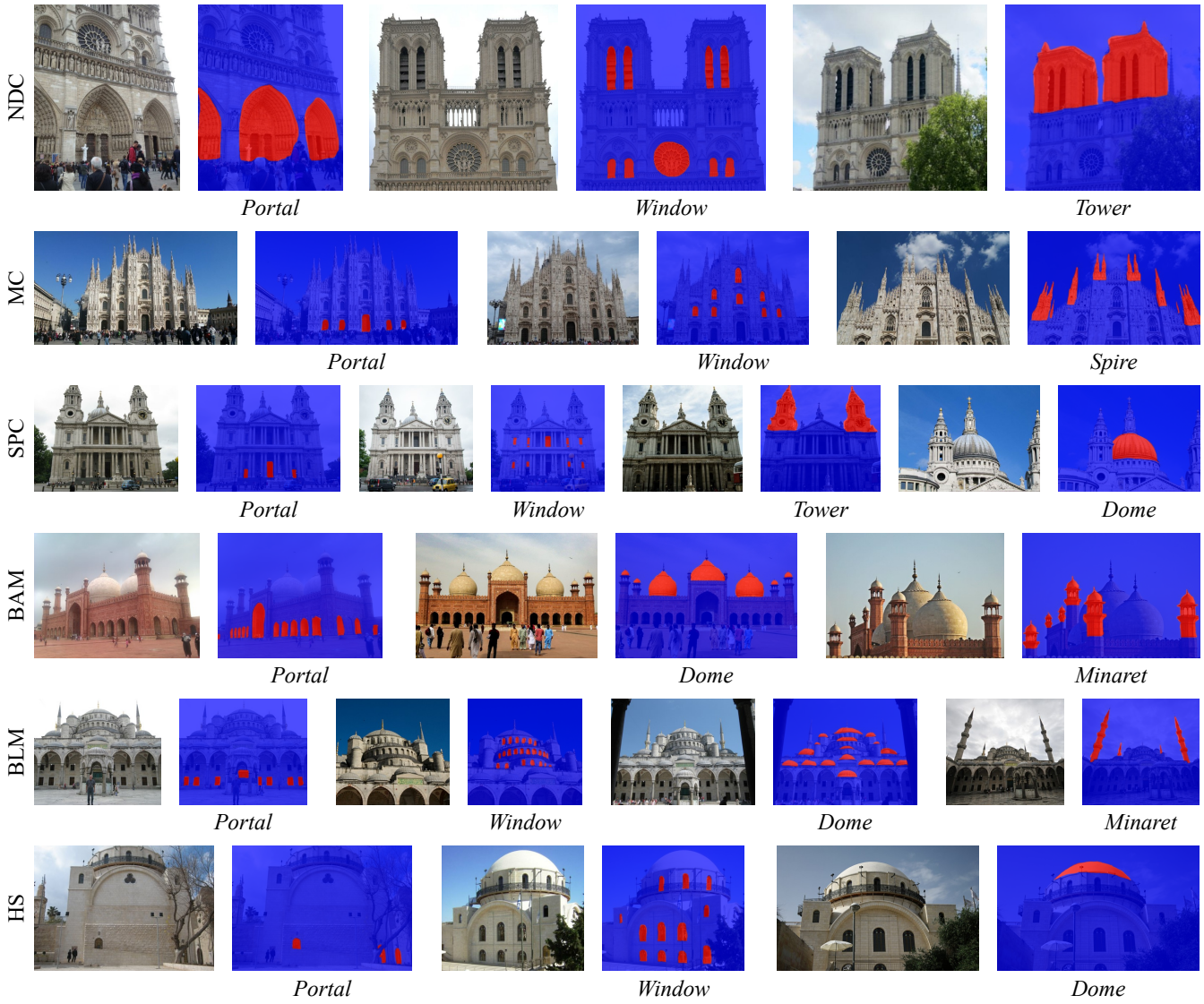

Figure 1: ***HolyScenes annotations.*** *We illustrate annotations for each category in each landmark in the HolyScenes dataset: Notre-Dame Cathedral (NDC), Milan Cathedral (MC), St. Paul's Cathedral (SPC), Badshahi Mosque (BAM), Blue Mosque (BLM), and Hurva Synagogue (HS).*

single word describing a prominent architectural feature within the image serving as its pseudo-label. Text is generated using beam search decoding with four beams. The prompt given to Flan-T5 includes the instruction to *Write "unknown" if it is not specified* (i.e. the architectural feature), in order to allow the language model to express uncertainty instead of hallucinating incorrect answers in indeterminate cases, as described in our main paper. We also find the use of the building's name in the prompt (*What architectural feature of* ⟨BUILDING⟩...) to be important in order to cue the model to omit the building's name from its output (e.g. *towers of the Cathedral of Seville* vs. simply *towers*).

To post-process these labels, we employ the following textual cleanup techniques. We (1) employ lowercasing, (2) remove outputs starting with "un-" ("unknown", "undefined" etc.), and (3) remove navigation words (e.g. "west" in "west facade") since these are not informative for learning visual semantics. Statistics on the final pseudo-labels are given in Section 3.1.

2.3. Semantic Adaptation of V&L Models

We fine-tune CLIP_{FT} on images and associated pseudo-labels, preprocessing by removing all pairs whose pseudo-label begins with "un-" (e.g. "unknown", "undetermined", etc.) and removing initial direction words ("north", "southern", "north eastern", etc.), as these are not visually informative. In total, this consists of 57,874 such pairs used as training data representing 4,031 unique pseudo-label values; this includes 41,452 pairs from cathedrals and 16,422

pairs from mosques. Fine-tuning is performed on CLIP initialized with the `clip-ViT-B-32` checkpoint as available in the `sentence-transformers` model collection on Hugging Face model hub, using the contrastive multiple negatives ranking loss as implemented in the Sentence Transformers library. We train for 5 epochs with learning rate 1e-6 and batch size 128.

To collect training data based on image correspondences for $\text{CLIPSeg}_{\text{FT}}$, we use the following procedure: Firstly, to find pairs of images in geometric correspondence, we perform a search on pairs of images (I_1, I_2) from each building in our train set along with the pseudo-label P of the first image, applying LoFTR [SSW*21] to such pairs and filtering for pairs in correspondence where I_1 is a zoomed-in image corresponding to category P and I_2 is a corresponding zoomed-out image. We filter correspondences using the following heuristic requirements:

- At least 50 corresponding keypoints that are inliers using OpenCV's `USAC_MAGSAC` method for fundamental matrix calculation.
- A log-ratio of at least 0.1 between the dispersion (mean square distance from centroid, using relative distances to the image dimensions) of inlier keypoints in I_1 and I_1.
- CLIP_{FT} similarity of at least 0.2 between I_1 and P, and at most 0.3 between I_2 and P. This is because I_1 should match P, while I_2, as a zoomed-out image, should contain P but not perfectly match it as a concept.
- At least 3 inlier keypoints within the region R_P of I_1 matching P. R_P is estimated by by segmenting I_1 with CLIPSeg and prompt P and binarizing with threshold 0.3.
- A low ratio of areas of the region matching P relative to the building's facade, since this suggests a localizable concept. This is estimated as follows: We first find the quadrilateral Q which is the region of I_2 corresponding to I_1, by projecting I_1 with the homography estimated from corresponding keypoints. We then find the facade of the building in I_2 by segmenting I_2 using CLIPSeg with the prompt *cathedral* or *mosque* (as appropriate for the given landmark), which outputs the matrix of probabilities M. Finally, we calculate the sum of elements of M contained in Q divided by the sum of all elements of M, and check if this is less than 0.5.

Empirically, we find that these heuristics succeed in filtering out many types of uninteresting image pairs and noise while selecting for the correspondences and pseudo-labels that are of interest. Due to computational constraints, we limit our search to 50 images from each landmark in our train set paired with every other image from the same landmark, and this procedure yields 3,651 triplets (I_1, I_2, P) in total, covering 181 unique pseudo-label categories. To use these correspondences as supervision for training segmentation, we segment I_1 using CLIPSeg with prompt P, project this segmentation onto I_2 using the estimated homography, and using the resulting segmentation map in the projected region as ground-truth for segmenting I_2 with P.

In addition to this data, we collect training data on a larger scale by searching for images from the entire training dataset with crops that are close to particular pseudo-labels. To do this, we run a search by randomly selecting landmark L and and one of its images I, selecting a random pseudo-label P that appears with L (not necessarily with the chosen image) in our dataset, selecting a random crop C of I, and checking its similarity to P with CLIP_{FT}. We check if the following heuristic conditions hold:

- C must have CLIP_{FT} similarity of at least 0.2 with P.
- C must have higher CLIP_{FT} similarity to P than I does.
- This similarity must be higher than the similarity between C and the 20 most common pseudo-labels in our train dataset (excluding P, if it is one of these common pseudo-labels).
- C when segmented using CLIPSeg with prompt P must have some output probability at least 0.1 in its central area (the central 280×280 region within the 352×352 output matrix).

If these conditions hold, we use the pair (I, P) along with the CLIPSeg segmentation of the crop C with prompt P as ground-truth data for fine-tuning our segmentation model. Although this search could be run indefinitely, we terminate it after collecting 29,440 items to use as training data.

For both sources of data (correspondence-based and crop-based), we further refine the pseudo-labels by converting them to singular, removing digits and additional direction words, and removing non-localizable concepts and those referring to most of the landmark or its entirety ("mosque", "front", "gothic", "cathedral", "side", "view").

We fine-tune CLIPSeg to produce $\text{CLIPSeg}_{\text{FT}}$ by training for 10 epochs with learning rate 1e-4. We freeze CLIPSeg's encoders and only train its decoder module. To provide robustness to label format, we randomly augment textual pseudo-labels by converting them from singular to plural form (e.g. "window" → "windows") with probability 0.5. At each iteration, we calculate losses using a single image and ground-truth pair from the correspondence-based data, and a minibatch of four image and ground-truth pairs from the crop-based data. We use four losses for training, summed together with equal weighting, as described in the main paper in Section 3.2.

CLIPSeg (and $\text{CLIPSeg}_{\text{FT}}$) requires a square input tensor with spatial dimensions 352×352. In order to handle images of varying aspect ratios during inference, we apply vertical replication padding to short images, and to wide images we average predictions applied to a horizontally sliding window. In the latter case, we use overlapping windows with stride of 25 pixels, after resizing images to have maximum dimension of size 500 pixels. Additionally, in outdoor scenes, we apply inference after zooming in to the bounding box of the building in question, in order to avoid attending to irrelevant regions. The building is localized by applying CLIPSeg with the zero-shot prompt *cathedral*, *mosque*, or *synagogue* (as appropriate for the building in question), selecting the smallest bounding box containing all pixels with predicted probabilities above 0.5, and adding an additional 10% margin on all sides. While our model may accept arbitrary text as input, we normalize inputs for metric calculations to plural form ("portals", "windows", "spires" etc.) for consistency.

2.4. 3D Localization

We build on top of the Ha-NeRF [CZL*22] architecture with an added semantic channel, similarly to Zhi *et al.* [ZLLD21]. This semantic channel consists of an MLP with three hidden layers (dimensions 256, 256, 128) with ReLU activations, and a final output

layer for binary prediction with a softmax activation. We first train the Ha-NeRF RGB model of a scene (learning rate 5e-4 for 250K iterations); we then freeze the shared MLP backbone of the RGB and semantic channels and train only the semantic channel head (learning rate 5e-5, 12.5K iterations). We train with batch size 8,192. When training the semantic channel, the targets are binary segmentation masks produced by $\text{CLIPSeg}_{\text{FT}}$ with a given text prompt, using the inference method described above. We binarize these targets (threshold 0.2) to reduce variance stemming from outputs with low confidence, and we use a binary cross-entropy loss function when training on them.

For indoor scenes, we use all available images to train our model. For outdoor scenes, we select 150 views with segmentations for building the 3D semantic field by selecting for images with clear views of the building's entire facade without occlusions. We find that this procedure yields comparable performance to using all the images in the collection, while being more computationally efficient. To select these images, we first segment each candidate image with CLIPSeg using one of the prompts *cathedral*, *mosque*, or *synagogue* (as relevant) , select the largest connected component C of the output binary mask (using thresold 0.5), and sort the images by the minimum horizontal or vertical margin length of this component from the image's borders. This prioritizes images where the building facade is fully visible and contained within the boundary of the visible image. To prevent occluded views of the building from being selected, we add a penalty using the proportion of overlap C and the similar binary mask C' calculated on the RGB NeRF reconstruction of the same view, since transient occlusions are not typically reconstructed by the RGB NeRF. In addition, we penalize images with less than 10% or more than 90% total area covered by C, since these often represent edge cases where the building is barely visible or not fully contained within the image. Written precisely, the scoring formula is given by $s = m + c - x$, where m is the aforementioned margin size (on a scale from 0 to 1), c is the proportion of area of C' overlapping C, and x is a penalty of 1.0 when C covers too little or much of the image (as described before) and 0 otherwise.

Runtime. A typical run (optimizing the volumetric probabilities for a single landmark) takes roughly 2 hours on a NVIDIA RTX A5000 with a single GPU. Optimizing the RGB and density values is only done once per landmark, and takes 2 days on average, depending on the number of images in the collection.

2.5. Baseline Comparisons

We provide additional details of our comparison to DFF [KMS22] and LERF [KKG*23]. We train these models on the same images used to train our model. We use a Ha-NeRF backbone; similarly to our method we train the RGB NeRF representations for 250K steps and then the semantic representations for an additional 150K steps. Otherwise follow the original training and implementation details of these models, which we reproduce here for clarity.

For DFF, we implement feature layers as an MLP with 2 hidden layers of 128 and ReLU activations. The input to the DFF model is the images and the corresponding features derived from LSeg, and we minimize the difference between the learned features and LSeg features with an L2 loss, training with batch size 1024.

For LERF, we use the official implementation which uses the Nerfacto method and the Nerfstudio API [TWN*23]. The architecture includes a DINO MLP with one hidden layer of dimension 256 and a ReLU activation; and a CLIP MLP consisting of with 3 hidden layers of dimension 256, ReLU activations, and a final 512-dimensional output layer. The input to this model consists of images, their CLIP embeddings in different scales, and their DINO features. We use the same loss as the original LERF paper: CLIP loss for the CLIP embeddings to maximize the cosine similarity, and MSE loss for the DINO features. The CLIP loss is multiplied by a factor of 0.01 similar to the LERF paper. We use an image pyramid from scale 0.05 to 0.5 in 7 steps. We train this model with batch size was 4096. We used also the relevancy score with the same canonical phrases as described in the LERF paper: "object", "things", "stuff", and "texture".

3. Additional Results and Ablations

3.1. Pseudo-Label Statistics

The pseudo-labels used in training consist of 4,031 unique non-empty values (over 58K images). The most common pseudo-labels are *facade* (5,380 occurrences), *dome* (3,084 occurrences), *stained class windows* (2,550 occurrences), *exterior* (2,365 occurrences), and *interior* (1,649 occurences). 2,453 pseudo-labels occur only once (61% of unique values) and 3,426 occur at most five times (79% of unique values). Examples of pseudo-labels that only occur once include: *spiral relief, the attic, elevation and vault, archevêché, goose tower, pentcost cross, transept and croisée*.

We note the long tail of pseudo-labels includes items shown in our evaluation such as *tympanum* (29 occurrences), *roundel* (occurs once as *painted roundel*), *colonnade* (230 occurrences), and *pediment* (3 occurrences; 44 times as plural *pediments*).

3.2. CLIPSeg Visualizations

As described in our main paper, we leverage the ability of CLIPSeg to segment salient objects in zoomed-in images even when it lacks fine-grained understanding of the accompanying pseudo-label. To illustrate this, Figure 4 shows several results of inputting the target text prompt *door* to CLIPSeg along with images that do not have visible doors. As seen there, the model segments salient regions which bear some visual and semantic similarity to the provided text prompt (*i.e.* possibly recognizing an "opening" agnostic to its fine-grained categorization as a door, portal, window, etc). Our fine-tuning scheme leverages this capability to bootstrap segmentation knowledge in zoomed-out views by supervising over zoomed-in views where the salient region is known to correspond to its textual pseudo-label.

Additionally, we find that 2D segmentation maps often show a bias towards objects and regions in the center of images, at the expense of the peripheries of scenes. This is seen for instance in Figure 4, where the windows on the center are better localized, in comparison to the windows on the sides of the building.

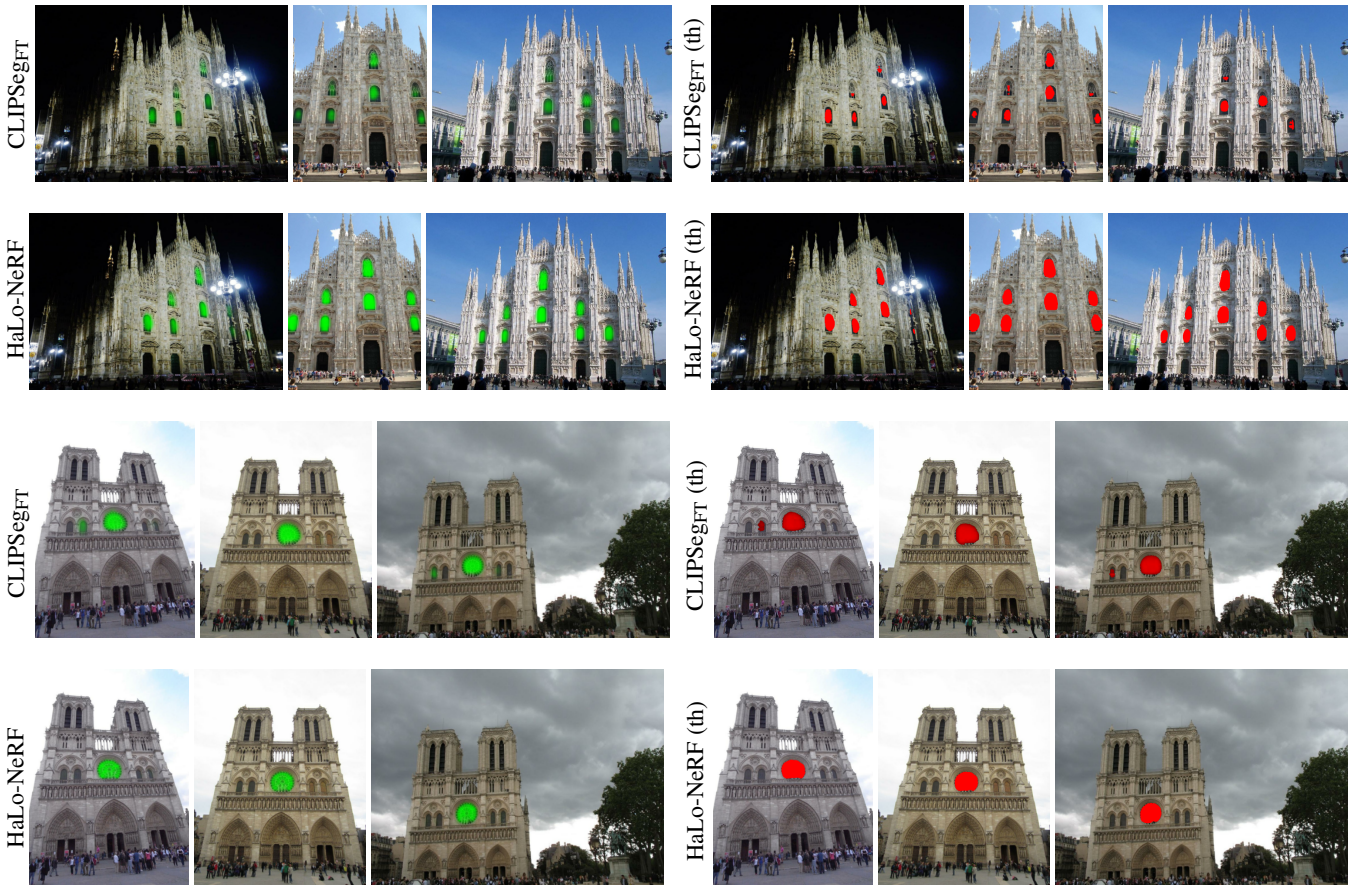

Figure 2: ***Results before and after 3D localization.*** *Segmentation results for the prompts windows and rose window are presented in the first and last pairs of rows, respectively. We show the results of CLIPSeg$_{FT}$ and HaLo-NeRF's projected localization in green, observing that HaLo-NeRF yields 3D-consistent results by fusing the 2D predictions of CLIPSeg$_{FT}$, which exhibit view inconsistencies. We also show binary segmentation (th) obtained with threshold* 0.5 *in red, seeing that inconsistencies are prominent when using these methods for binary prediction.*

3.3. HaLo-NeRF Visualizations

In Figure 2, we compare segmentation results before and after 3D localization. We see that HaLo-NeRF exhibits 3D consistency, while 2D segmentation results of CLIPSeg$_{FT}$ operating on each image separately exhibit inconsistent results between views. We also see that this effect is prominent when using these methods for binary segmentation, obtained by thresholding predictions.

In Figure 3, we demonstrate our ability to perform localization of multiple semantic concepts in a single view of a scene. By providing HaLo-NeRF with different text prompts, the user may decompose a scene into semantic regions to gain an understanding of its composition.

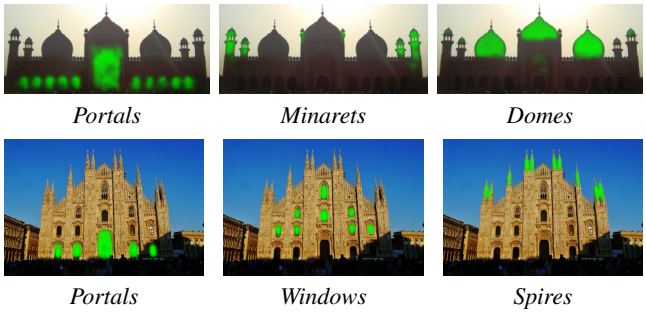

Figure 3: ***Different localization images for the same scene.*** *We show multiple semantic concept localizations of HaLo-NeRF for a single scene view. These results on Badshahi Mosque (first row) and Milan Cathedral (second row) illustrate how the user may provide HaLo-NeRF with multiple text prompts to understand the semantic decomposition of a scene.*

3.4. 2D Baseline Visualizations

In Figure 5, we visualize outputs of the two 2D baseline segmentation methods (LSeg and ToB) as well as CLIPSeg and our fine-tuned CLIPSeg$_{FT}$. We see that the baseline methods struggle to at-

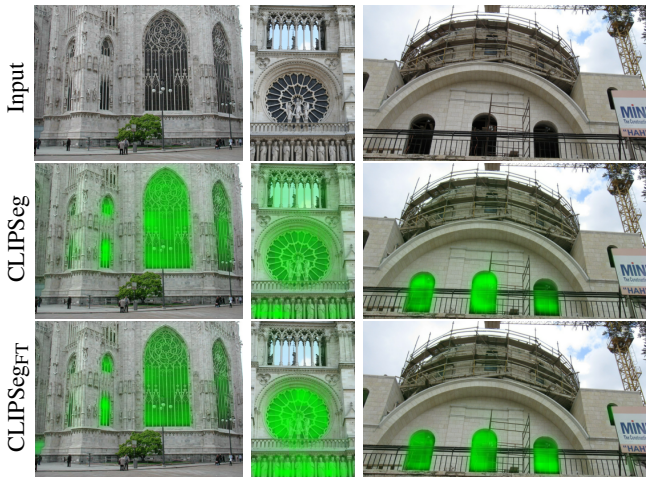

Figure 4: ***Providing text-based segmentation models with partially related text prompts***. *Above we provide the target prompt door to CLIPSeg (pretrained and fine-tuned) along with images that do not have visible doors. As seen above, the models instead segment more salient regions which bear some visual and semantic similarity to the provided text prompt (in this case, segmenting windows).*

tend to the relevant regions in our images, while CLIPSeg$_{FT}$ shows the best undestanding of these concepts and their localizations.

3.5. LLM Ablations

As an additional test of our LLM-based pseudo-labeling procedure, we ablate the effect of the LLM model size and prompt templates used. In particular, we test the following sizes of Flan-T5 [CHL*22]: XL (ours), Large, Base, and Small* These vary in size from 80M (Small) to 3B (XL) parameters. In addition, we test the following prompt templates: P_1 (our original prompt, including the phrase *...what architectural feature of...*), P_2 (*...what aspect of the building...*), and P_3 (*...what thing in...*).

We sample 100 random items from our dataset for manual inspection, running pseudo-labeling with our original setting (XL, P_1) as well as with alternate model sizes and prompts. Regarding model sizes, while the majority of non-empty generated pseudo-labels are valid as we show in the main paper, we consider how often empty or incorrect pseudo-labels are yielded when varying the model size. Considering this, 62/100 items receive an empty, poor or vague pseudo-label in our original setting, only one of these receives a valid pseudo-label with a smaller model, confirming the superior performance of the largest (XL) model. Regarding prompt variants, P_2 only yields 9/100 valid pseudo-labels (versus 38/100 for P_1), while P_3 yields 40/100 valid pseudo-labels (31 of these are in common with P_1). Thus, the best-performing prompt (P_3) is comparable to our original setting, suggesting that our original setting is well-designed to produce useful pseudo-labels.

* Available on Hugging Face Model Hub at the following checkpoints: `google/flan-t5-xl`, `google/flan-t5-large`, `google/flan-t5-base`, `google/flan-t5-small`.

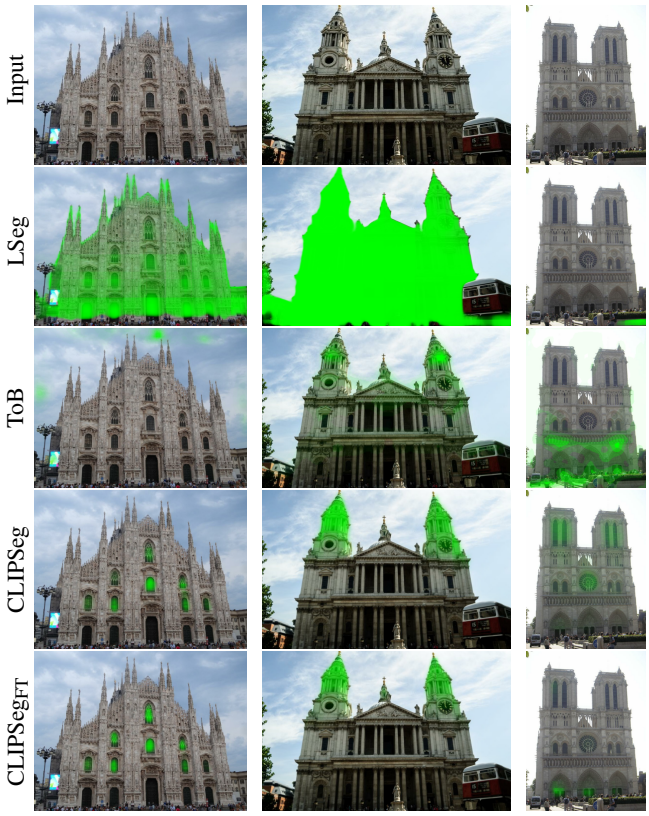

Figure 5: ***Illustration of baseline 2D segmentation methods.*** *As is seen above, the baseline methods (LSeg and ToB) struggle to attend to the relevant regions in the images, while CLIPSeg$_{FT}$ shows the best understanding of these concepts and their localizations, consistent with our quantitative evaluation.*

Method	R@1	R@5	R@10	R@16	R@32	R@64
CLIP	0.04	0.18	0.24	0.31	0.47	0.68
CLIP$_{FT}$	**0.08**	**0.30**	**0.44**	**0.52**	**0.66**	**0.79**

Table 2: ***Terminology Retrieval Evaluation***. *We evaluate image-to-text retrieval for finding relevant textual terms for test images. We report recall at $k \in \{1,5,10,16,32,64\}$, comparing our results (highlighted in the table) to the baseline CLIP model. Best results are highlighted in* ***bold****.*

3.6. CLIP$_{FT}$ Retrieval Results

In Table 2, we show quantitative results for the use of CLIP$_{FT}$ to retrieve relevant terminology, as described in our main paper. In particular, we fix a vocabulary of architectural terms found at least 10 times in the training data, and evaluate text-to-image retrieval on

Method	Thresh.	IoU	Precision	Recall
CLIPSeg	0.10	0.634	0.807	0.813
$\text{CLIPSeg}_{\text{FT}}$	0.10	**0.676**	**0.824**	**0.855**
CLIPSeg	0.15	0.650	0.815	0.827
$\text{CLIPSeg}_{\text{FT}}$	0.15	**0.690**	**0.833**	**0.860**
CLIPSeg	0.20	0.653	0.818	0.832
$\text{CLIPSeg}_{\text{FT}}$	0.20	**0.681**	**0.836**	**0.853**
CLIPSeg	0.25	0.649	0.817	0.832
$\text{CLIPSeg}_{\text{FT}}$	0.25	**0.652**	**0.836**	**0.833**

Table 3: *Results on Wikiscenes [WAESS21] for CLIPSeg before and after our fine-tuning procedure. Following Wu et al, we report IoU, precision, and recall scores. As these are threshold-dependent, we test multiple thresholds, finding that CLIPSeg_{FT} shows a performance boost overall.*

test images (from landmarks not seen during training) with pseudo-labels in this list. As seen in these results, our fine-tuning provides a significant performance boost to CLIP in retrieving relevant terms for scene views, as the base CLIP model is not necessarily familiar with fine-grained architectural terminology relevant to our landmarks out-of-the-box.

3.7. Additional $\text{CLIPSeg}_{\text{FT}}$ Results

To test the robustness of our CLIPSeg fine-tuning on additional datasets and preservation of pretraining knowledge, we evaluate segmentation results on two additional datasets: SceneParse150 [ZZP*16, ZZP*17] (general outdoor scene segmentation) and Wikiscenes [WAESS21] (architectural terminology).

On SceneParse150, we test on the validation split (2000 items), selecting a random semantic class per image (from among those classes present in the image's annotations). We segment using the class's textual name and measure average precision, averaged over all items to yield the mean average precision (mAP) metric. We observe a negligible performance degradation after fine-tuning, namely mAP 0.53 before fine-tuning and 0.52 afterwards, suggesting overall preservation of pretraining knowledge.

On Wikiscenes, fine-tuning improves all metrics reported by Wu et al. (IoU, precision, recall), as shown in Table 3. As these metrics are threshold-dependent, we test multiple threshold values, and see that $\text{CLIPSeg}_{\text{FT}}$ shows an overall improvement in performance (e.g. reaching IoU of 0.890 for the optimal threshold, while CLIPSeg without fine-tuning does not exceed 0.681). Thus, as expected, our model specializes in the architectural domain while still showing knowledge of general terms from pretraining.

3.8. LERF with CLIP_{FT}

In Table 4, we show quantitative results of LERF [KKG*23] using our CLIP_{FT}, and we compare it the results of LERF with CLIP without fine-tuning. We denote LERF with CLIP_{FT} as LERF_{FT}. In both cases, we use the Ha-NeRF backbone. We see that the results of LERF_{FT} are only slightly better than the results of LERF with the original CLIP, suggesting that using better features for regression is not sufficient in our problem setting.

Method	mAP	portal	window	spire	tower	dome	minaret
LERF*	0.14	0.16	**0.15**	0.18	**0.13**	0.10	0.09
LERF_{FT}*	**0.15**	**0.18**	**0.15**	**0.19**	0.12	**0.12**	**0.11**

*Using a Ha-NeRF backbone

Table 4: ***LERF Comparison***. *We report mean average precision (mAP; averaged per category) and per category average precision over the HolyScenes benchmark, comparing LERF with LERF_{FT}. Best results are highlighted in* ***bold***.

Method	mAP	portal	window	spire
DFF	0.07	0.04	0.05	0.12
LERF	0.19	0.32	0.06	0.2
HaLo-NeRF	**0.65**	**0.76**	**0.56**	**0.64**

Table 5: ***Constant Illumination Comparison***. *We report mean average precision (mAP; averaged per category) and per category average precision over the relevant categories in Milan Cathedral from Google-Earth using constant illumination. Best results are highlighted in* ***bold***.

3.9. Results in a Constant Illumination Setting

In Table 5, we show quantitative results of DFF, LERF, and HaLo-NeRF for a scene with constant illumination using a single camera. We used images rendered from Google Earth (following the procedure described in [XXP*21]) for the Milan Cathedral with the following three semantic categories: portal, window, and spire. Because the images were taken using a single camera with constant

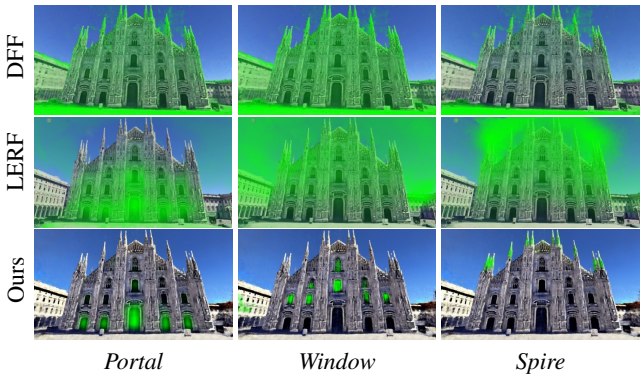

Figure 6: ***Localization comparison in a constant illumination setting***. *Above we show localization for DFF, LERF, and HaLo-NeRF (Ours) in a constant illumination setting, using input images rendered from Google Earth, as detailed further in the text. For this comparison, localization is performed using the original DFF and LERF implementations (and not our modified versions). As illustrated above, HaLo-NeRF outperforms the other methods also in a constant illumination setting.*

illumination, which adheres to the original DFF and LERF setting, we used their official public implementations. We produced ground-truth binary segmentation maps to evaluate the results by manual labelling of five images per category. As the table shows, the results of HaLo-NeRF are much better than those of DFF and LERF, even in the constant illumination case. These results further illustrate that these feature field regression methods are less effective for large-scale scenes. See Figure 6 for a qualitative comparison in this setting.

References

[CHL*22] CHUNG H. W., HOU L., LONGPRE S., ZOPH B., TAY Y., FEDUS W., LI E., WANG X., DEHGHANI M., BRAHMA S., ET AL.: Scaling Instruction-Finetuned Language Models. *arXiv preprint arXiv:2210.11416* (2022). 1, 6

[CZL*22] CHEN X., ZHANG Q., LI X., CHEN Y., FENG Y., WANG X., WANG J.: Hallucinated Neural Radiance Fields in the Wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), pp. 12943–12952. 1, 3

[KKG*23] KERR J., KIM C. M., GOLDBERG K., KANAZAWA A., TANCIK M.: Lerf: Language embedded radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (October 2023), pp. 19729–19739. 4, 7

[KMS22] KOBAYASHI S., MATSUMOTO E., SITZMANN V.: Decomposing NeRF for Editing via Feature Field Distillation. In *Advances in Neural Information Processing Systems (NeurIPS)* (2022). 4

[LS18] LI Z., SNAVELY N.: Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 2041–2050. 1

[SF16] SCHONBERGER J. L., FRAHM J.-M.: Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 4104–4113. 1

[SSW*21] SUN J., SHEN Z., WANG Y., BAO H., ZHOU X.: Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 8922–8931. 3

[TWN*23] TANCIK M., WEBER E., NG E., LI R., YI B., KERR J., WANG T., KRISTOFFERSEN A., AUSTIN J., SALAHI K., ET AL.: Nerfstudio: A modular framework for neural radiance field development. *arXiv preprint arXiv:2302.04264* (2023). 4

[WAESS21] WU X., AVERBUCH-ELOR H., SUN J., SNAVELY N.: Towers of Babel: Combining Images, Language, and 3D Geometry for Learning Multimodal Vision. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), pp. 428–437. 1, 7

[XXP*21] XIANGLI Y., XU L., PAN X., ZHAO N., RAO A., THEOBALT C., DAI B., LIN D.: Citynerf: Building nerf at city scale. *arXiv preprint arXiv:2112.05504* (2021). 7

[Yi20] YI K. M.: Image matching: Local features & beyond 2020. https://www.cs.ubc.ca/ kmyi/imw2020/data.html, 2020. https://www.cs.ubc.ca/ kmyi/imw2020/data.html. URL: https://www.cs.ubc.ca/~kmyi/imw2020/data.html, arXiv: https://www.cs.ubc.ca/~kmyi/imw2020/data.html. 1

[ZLLD21] ZHI S., LAIDLOW T., LEUTENEGGER S., DAVISON A. J.: In-Place Scene Labelling and Understanding with Implicit Scene Representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)* (2021), pp. 15838–15847. 3

[ZZP*16] ZHOU B., ZHAO H., PUIG X., FIDLER S., BARRIUSO A., TORRALBA A.: Semantic understanding of scenes through the ade20k dataset. *arXiv preprint arXiv:1608.05442* (2016). 7

[ZZP*17] ZHOU B., ZHAO H., PUIG X., FIDLER S., BARRIUSO A., TORRALBA A.: Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017). 7