RAD: Retrieval-Augmented Monocular Metric Depth Estimation for Underrepresented Classes

Supplementary Material

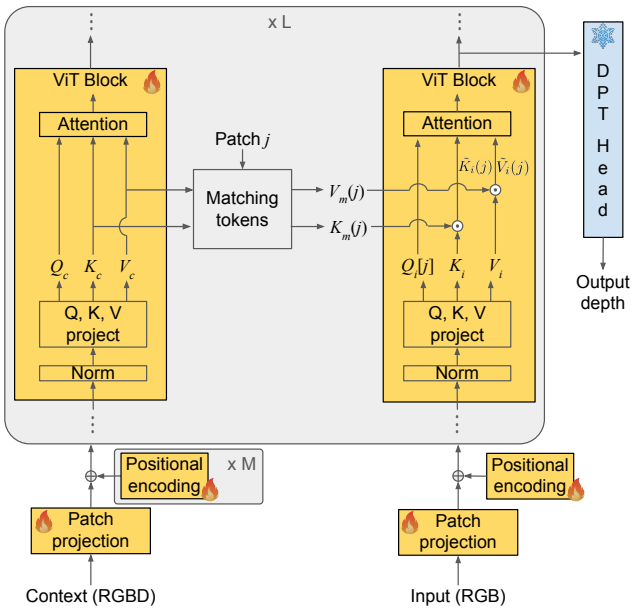


Figure 9. **Full network architecture.** The matched cross-attention mechanism allows the input stream to pull information from the context at the matched locations. Yellow blocks are optimized during training, while the blue block is frozen. Plus signs correspond to addition, the "dot" operation is concatenation.

A. Overall Network Architecture

Fig. 9 illustrates the complete network architecture employed in RAD. The design is based on the DepthAnything v2 framework [53], with one key modification: the original attention module is replaced by our matched cross-attention mechanism, which connects between the context and input streams. The remainder of the encoder's Vision Transformer (ViT) architecture remains unchanged.

The system comprises a dual-stream encoder, which is trainable during optimization, and a frozen DPT head [43]. For each token j in the input stream, the *matching tokens* block retrieves matched keys $K_m(j)$ and values $V_m(j)$, derived from the context keys K_c and values V_c at corresponding matching positions. These matched representations are concatenated with the input keys K_i and values V_i, forming the input to the standard scaled dot-product attention mechanism within the ViT. The input and context queries, $Q_i[j]$ and Q_c, are processed following the original DepthAnything v2 approach.

Importantly, each stream employs its own *positional encoding*, enabling the encoder to distinguish between context and input tokens. Additionally, each retrieved image is assigned a unique positional encoding, allowing the tokens to remain distinguishable within the upstream blocks of the ViT. Consequently, if M context samples are retrieved, the context stream utilizes M distinct positional encodings. All positional encoders are trainable to support this differentiation.

The *patch projection* for the input stream follows the original design of the DepthAnything v2 network. For the context stream, we introduce a fourth channel to process the sample's depth, as detailed in Sec. 3.1.3. The procedures for *normalization* and *Q, K, V projection* are identical to those employed in each ViT block of DepthAnything v2. Similarly, the *attention* mechanism remains unchanged, utilizing scaled dot-product attention.

B. Additional Qualitative Results

Additional qualitative results are presented in Fig. 10.

C. Mesh Creation for 3D Augmentation

During training, context samples may be generated through 3D augmentation (Sec. 3.1.1). In this process, the input image is rendered from a novel viewpoint using ground-truth depth and camera calibration parameters. A straightforward approach is to back-project each pixel into 3D and re-project it onto a new image plane; however, this often produces large regions without valid signal because the new viewpoint is misaligned with the original pixel grid. Such gaps degrade the realism of augmented samples and can negatively impact downstream learning.

To address this limitation, we reconstruct a continuous surface mesh from the back-projected 3D point cloud, enabling dense coverage and improved geometric consistency. For mesh generation, we employ NKSR [25], which has demonstrated strong performance across multiple domains. Figure 11 compares image re-projection using only the point cloud versus the full mesh, highlighting the substantial improvement in completeness and fidelity.

D. Additional Component Visualizations

D.1. Point Matching

Fig. 12 demonstrates the performance of the point-matching algorithm on three representative input images containing underrepresented classes (left) and their corresponding retrieved images (right) from Cityscapes. While point-matching methods are typically trained on multiple views

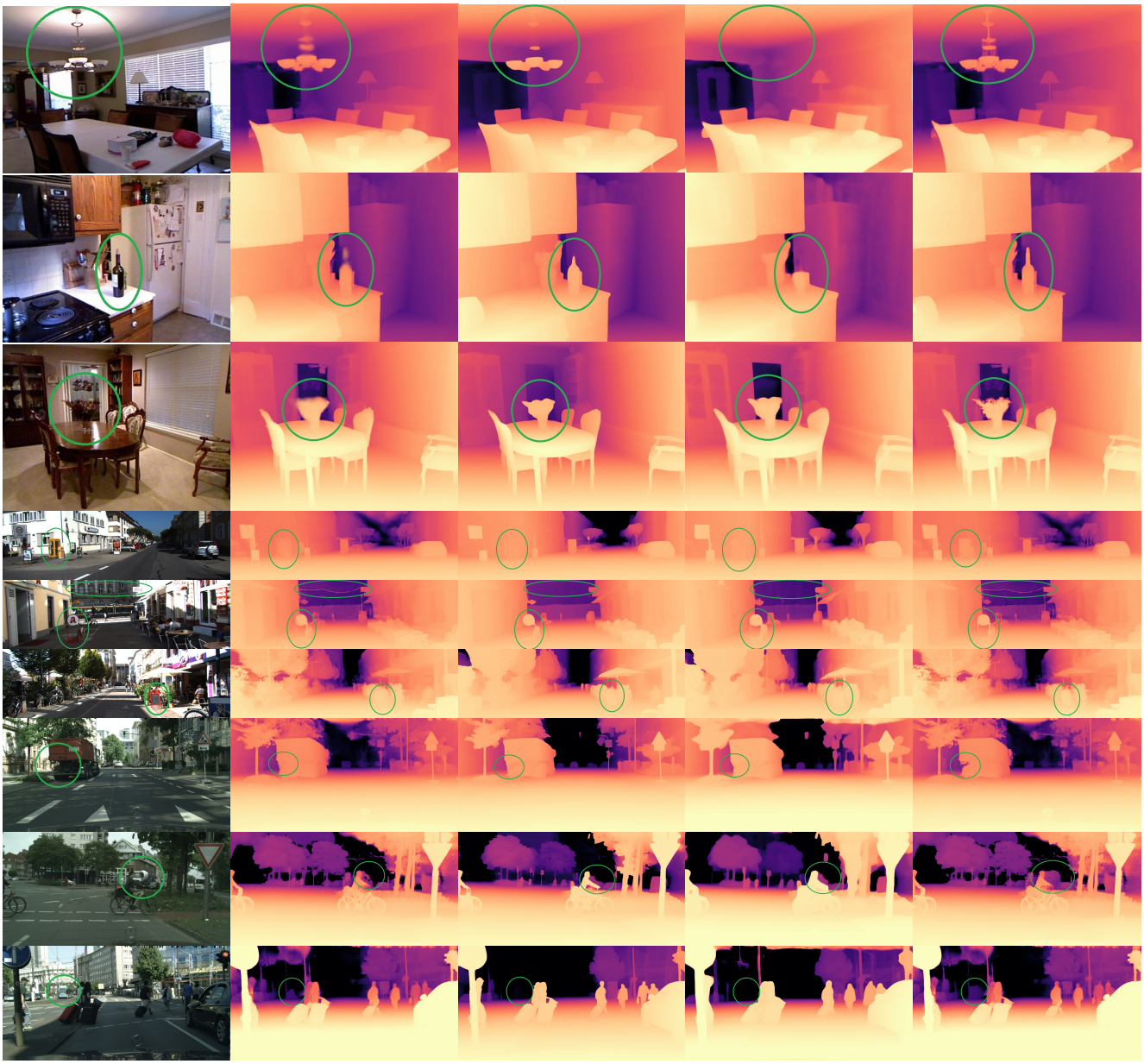
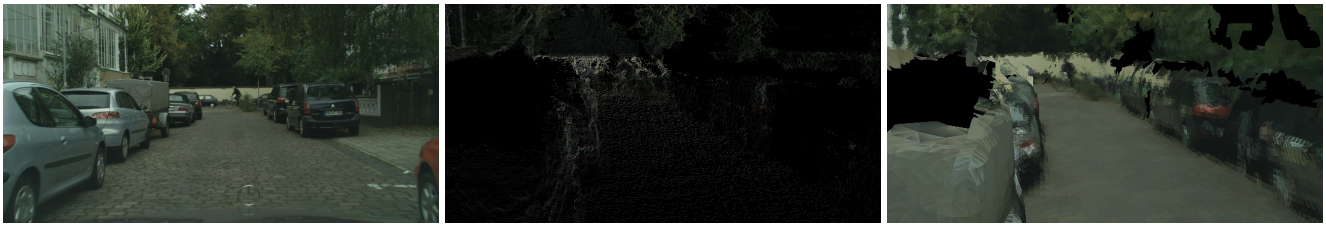


Figure 10. **Qualitative results** for NYU Depth v2 (top three rows), KITTI (middle three rows) and Cityscapes (bottom three rows). We compare our method (RAD) to baselines DepthAnything v2 [53], UniDepth v2 [39] and Metric3D v2 [22]. Best viewed zoomed in.

of the same scene, LightGlue [33] exhibits strong generalization to semantic similarities across different scenes, even under variations in scale and illumination. Notably, most matches occur on underrepresented objects, as these elements tend to be the most visually similar due to the uncertainty-aware retrieval mechanism described in Sec. 3.1.1.

D.2. Matched cross-attention

Fig. 13 illustrates additional results for matched cross-attention. Attention to input patches consistently concentrates on relevant geometric structures, whereas attention to context patches is pronounced when point correspondences are correct and markedly weaker when correspondences are incorrect. This behavior underscores the model's ability to leverage accurate matches for contextual reasoning, while reducing reliance on erroneous correspondences, thereby



RGB input | Point cloud projection | Mesh projection

Figure 11. **Mesh creation for 3D augmentation.** The input image (left) is projected from a new point of view. When using only the point cloud defined by the image pixels, projection yields large unmeasured regions (middle). When a mesh is used to reconstruct geometry, the scene is densely reconstructed.

Figure 12. **Point matches for underrepresented classes** (top: tram, middle: truck, bottom: bus). The left image is sampled from the validation set, while the right image is from the training set. Matches remain consistent despite variations in illumination and scale.

enhancing robustness.

E. Training and Implementation Details

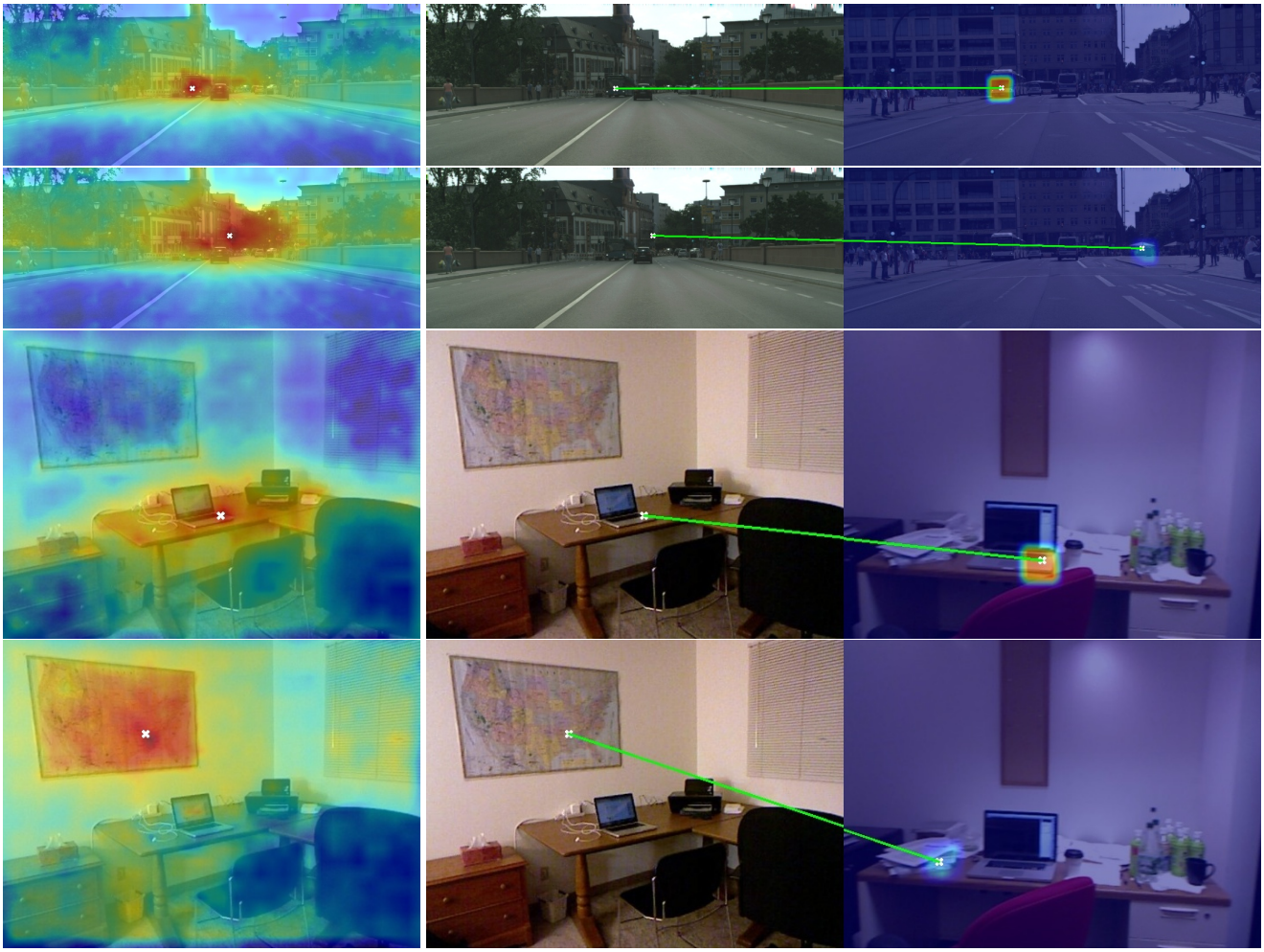
E.1. Network Optimization

Our training procedure was carried out in stages, as outlined below:

1. Start with pre-trained DepthAnything v2.
2. Fine-tune the DepthAnything v2 network on the training dataset to produce metric depth rather than relative depth. All weights are trained, with a significantly lower learning rate applied to the encoder, as recommended by the original authors.
3. Construct the context stream encoder by duplicating the fine-tuned encoder from Step 2. Modify the projection operation in the context ViT encoder to accept the depth channel (Sec. 3.1.3 in the main paper).
4. Freeze the decoder and fine-tune the dual-stream encoder. Optimization is performed on the training dataset using the complete retrieval-augmented pipeline (Sec. 3.1.1 in the main paper).

In Step 4 we optimized the positional encoding in both streams so that the network learns to differentiate between the two types of inputs.

As objective, we used the scale invariant log loss [11], also used by DepthAnything v2. We trained our models



(a) (b)

Figure 13. **Visualization of matched cross-attention**. For a selected patch in the input image, indicated by a white marker, and its corresponding matched point in the context image, we separately visualize the attention directed toward the input (a) and the context (b). When the match is correct, strong cross-attention emerges within the local neighborhood of the matched point in the context image. In contrast, incorrect matches yield weak cross-attention responses. Colormaps are kept consistent across images.

using the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 1×10^{-2}. The initial learning rate was set to 5×10^{-5} and decayed by a factor of 10 after 5 epochs of stagnation. Training was performed on four Nvidia L40S GPUs.

E.2. Uncertainty-aware Image Retrieval

To compute uncertainty, we generated 5 noisy variants of a given image image by adding Gaussian noise with a standard deviation $\sigma = 0.1$ (for normalized image values). The parameters h and q in Section 3.1.1 were set to 0.05 and 20%, respectively. Unless explicitly stated otherwise, in the experiments (Sec. 4) we retrieved 4 images and applied matched cross-attention using a 3×3 patch neighborhood. To perform efficient search we use the FAISS library [8].

E.3. Baseline Details

For each baseline in the experiments section (Sec. 4) we use the publicly available code and weights. Specifically, for ZoeDepth [4] we use the horizontal flip augmentation suggested by the authors. All methods are run according to their official instructions.

F. Runtime and Memory Analysis

We evaluate the model size, memory footprint, and inference runtime of RAD, and compare these metrics against those of the baseline architecture, DepthAnything v2 [53], as presented in Tab. 4. All measurements are conducted on Cityscapes images at a resolution of 2048×1024, using a single Nvidia L40S GPU and a RAD configuration that in-

corporates four context samples.

Table 4. **Parameter count, memory and runtime analysis.** Compared to DepthAnything v2 [53], RAD has approximately twice the parameter count and memory due to the second stream. Runtime is in seconds.

Method	Parameters	Memory	Runtime
DepthAny-Small	24.8M	94.55 MB	0.013 s
DepthAny-Base	97.5M	371.82 MB	0.017 s
DepthAny-Large	335.3M	1279.13 MB	0.019 s
RAD-Small	48.5M	184.99 MB	1.724 s
RAD-Base	187.4M	714.72 MB	1.738 s
RAD-Large	644.1M	2457.02 MB	1.780 s

Table 5. **Runtime breakdown for RAD-Large.** All values are reported in seconds.

Process	RAD-Small	RAD-Base	RAD-Large
Uncertainty	0.017 s	0.020 s	0.024 s
Segmentation	1.487 s	1.487 s	1.487 s
Point matching	0.191 s	0.191 s	0.191 s
KNN search	0.015 s	0.015 s	0.015 s
Feed-forward	0.014 s	0.025 s	0.063 s

A detailed breakdown of RAD's runtime is provided in Tab. 5. To estimate uncertainty, five inference operations are executed in parallel, corresponding to five noisy versions of the input image. Similarly, the four context samples and the input image are processed concurrently during RAD's feed-forward pass, synchronizing only to exchange information for matched cross-attention.

For RAD, both the parameter count and memory consumption are approximately doubled relative to DepthAnything v2, primarily due to the inclusion of the context stream. The inference time is substantially higher, rendering RAD unsuitable for real-time applications. This increase in runtime is largely attributable to the SAM2 segmentation [44] and LightGlue point matching [33] procedures. Consequently, the proposed approach would benefit significantly from future advances in segmentation and point-matching frameworks that combine accuracy with computational efficiency. Furthermore, it is possible to adapt the method to real-time applications by removing the segmentation process, and ussing only uncertainty, yielding a bit more noisy retrieval but improved runtime.

G. Limitations

RAD exhibits three primary limitations:

1. **High runtime.** The method's inference time is prohibitively large for real-time applications, with most of the overhead arising from image segmentation used for uncertainty-aware retrieval. A potential alternative is to employ pixel-wise uncertainty without coupling it to segmentation, enabling faster retrieval at the cost of increased noise, a trade-off that may be acceptable for real-time scenarios.
2. **Dependence on depth-annotated context data.** RAD requires a context dataset with ground-truth depth, which is not always available. While the context stream could operate on RGB-only inputs and leverage proxy multi-view cues to improve depth estimation, ground-truth depth is a strong signal for accurate performance.
3. **Sensitivity to retrieval and matching errors.** The approach relies on accurate image retrieval and point matching, as these components feed the matched cross-attention mechanism. Although attention partially mitigates this issue by down-weighting unreliable patches, errors in these stages can still propagate and degrade overall performance.