MODELING DEPTH AMBIGUITY: A MIXTURE-DENSITY REPRESENTATION FOR FLYING-POINT-FREE DEPTH ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite advances in depth estimation, *flying points* remain a persistent failure mode: near object boundaries, depth estimators often predict spurious 3D points in the empty space between foreground and background surfaces. We trace this artifact to a standard modeling choice: assigning each pixel a single depth hypothesis. At boundaries, a pixel can straddle a foreground and a background surface, so its true depth is ambiguous between the two. A model that predicts a single depth cannot keep both possibilities, so training instead pulls the prediction toward an intermediate depth that lies on neither surface. We address this with MDA, a mixture-density representation that lets the model predict multiple depth hypotheses and their associated probabilities for each pixel. Near boundaries, different hypotheses can align with different surfaces, and the decoded depth is selected from one of these hypotheses rather than placed in the empty space between them. Across different architectures, MDA substantially improves boundary reconstruction over each base model and largely removes flying-point artifacts even under severe input blur, while adding negligible runtime overhead. The same mixture-density framework naturally extends to transparent objects, where it predicts multiple depth layers at transparent pixels, and to sky regions, where a dedicated component separates the unbounded sky from finite-depth regions, producing flying-point-free skylines.

1 INTRODUCTION

Depth estimation from images has advanced remarkably in recent years (Eigen et al., 2014; Lin et al., 2026; Wang et al., 2025a; Xu et al., 2026; Wang et al., 2025c; Pavlovic et al., 2026): models can now recover accurate depth from a single frame or a handful of views. Yet one common failure remains widespread: *flying points*, 3D points that fall in empty space between foreground and background surfaces near object boundaries (Fig. 1). These artifacts corrupt reconstructed geometry and reduce the reliability of downstream applications such as novel-view synthesis and robotic manipulation. Importantly, flying points persist despite stronger backbones (Lin et al., 2026; Wang et al., 2025a; 2026a) and larger training datasets (Zhou et al., 2026; Ling et al., 2024; Pan et al., 2023; Wang et al., 2020; Roberts et al., 2021), suggesting that they are not simply a scaling problem.

We argue that flying points arise from a seemingly natural modeling choice: most depth estimators assign each pixel only one depth hypothesis. This works well on ordinary pixels, but becomes problematic at object boundaries. A boundary pixel can straddle an occlusion edge, so its image patch and features contain cues from both the foreground and background, making it difficult for the model to determine which surface the pixel belongs to. In reality, the correct point should lie on either the foreground or background surface, not between them. However, when the model is trained to predict a single value, it tends to compromise between the two plausible surface depths whenever it is unsure which surface the pixel belongs to, producing a point in empty space. This limitation is also reflected in the training objective. Standard ℓ_1 and ℓ_2 depth losses are negative log-likelihoods of Laplacian or Gaussian distributions centered at the predicted depth, thereby enforcing a unimodal per-pixel representation. This representation is well matched to smooth, unambiguous surfaces, but too restrictive at boundaries, where the plausible depth distribution is naturally multi-modal.

Existing fixes for flying points remain partial. PPD and PPVD (Xu et al., 2025; 2026) use generative models to refine the outputs of feed-forward depth estimators, but their multi-step denoising process

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

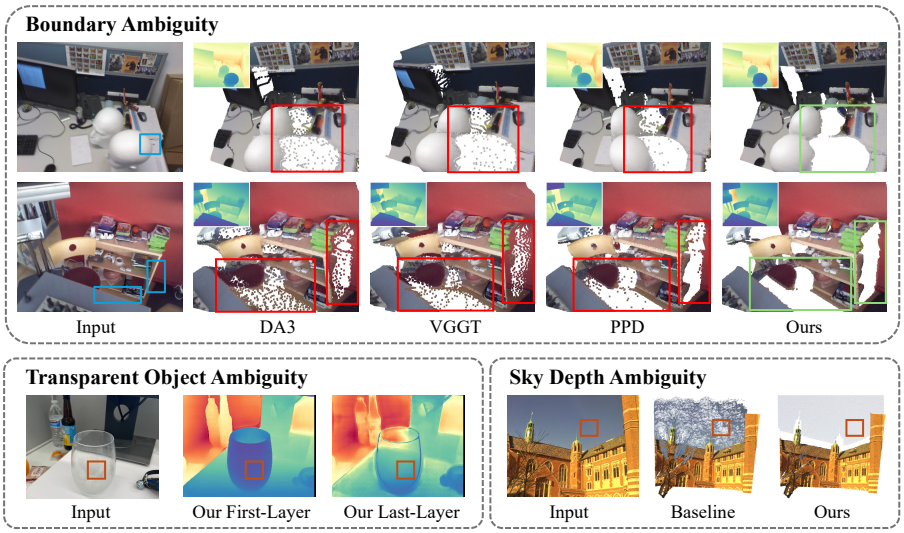


Figure 1: Overview of our approach. Existing depth estimators model each pixel as a unimodal distribution, producing flying points at boundaries. Our mixture-density model keeps multiple depth hypotheses, eliminating boundary artifacts, recovering layered depth behind transparent objects, and providing a clean skyline.

is slow and struggles when input images are blurry (Fig. 4). MoE3D (Wang et al., 2026b) routes different spatial regions to different depth heads but is still trained under a single-depth objective, so boundary ambiguity is re-distributed in different experts rather than resolved. SMDNet (Tosi et al., 2021) models stereo disparity as a two-component Laplacian mixture; we generalize this idea to K components in various monocular and multi-view estimators and to transparent objects and sky.

To address this issue, we propose Modeling Depth Ambiguity (MDA), a mixture-density depth representation that explicitly models the depth ambiguity near the boundary. For each pixel, the model predicts K Laplacian or Gaussian depth hypotheses together with their probabilities, and is trained with the corresponding mixture negative log-likelihood. On ordinary pixels, these hypotheses can agree, behaving like the original unimodal representation; near boundaries, they can instead represent the foreground and background surfaces separately. In this way, ambiguity is represented by probabilities over multiple depth hypotheses, rather than by shifting a single depth prediction into empty space. During training, different components specialize to different depth layers and remain anchored to valid surfaces. At inference, the decoded depth is selected from these hypotheses, so even imperfect boundary probabilities still yield points on existing surfaces instead of between surfaces.

MDA keeps the backbone unchanged and only modifies the final prediction layer(s): each component predicts a depth, a confidence score, and a mixture-weight logit (§3.2). It therefore adds only a small number of output channels and negligible inference overhead, and can be plugged into different architectures. We instantiate it on two feed-forward depth estimators, DA3 and VGGT, where it substantially improves boundary reconstruction (Fig. 5) and largely removes flying points. The same modification can also be applied to the depth decoder of the diffusion-based Marigold V2 (Pavlovic et al., 2026), improving its boundary quality.

Our mixture-density representation extends beyond object boundaries to broader forms of *depth ambiguity*, where a single pixel cannot be explained well by one depth value. For transparent objects, a camera ray may pass through glass and intersect multiple visible surfaces, making multiple depths physically valid at the same pixel. We adapt our representation so that multiple hypotheses can be active simultaneously, allowing the model to recover co-existing depth layers in transparent regions while preserving sharp boundaries elsewhere. For sky regions, where depth is effectively unbounded, we add a dedicated large-depth component. This yields sky segmentation and clean, flying-point-free skylines within the same prediction framework.

In summary, our contributions are:

- We identify flying points as a consequence of forcing depth estimators to predict a single depth value at ambiguous boundary pixels.

- We propose a lightweight K-component mixture-density depth representation that predicts multiple depth hypotheses and their probabilities, allowing boundary pixels to choose between foreground and background surfaces instead of landing between them.
- We instantiate the architecture-agnostic representation in the DA3 and VGGT depth heads and demonstrate substantially improved boundary reconstruction at negligible cost. We further show that it transfers to the diffusion-based Marigold V2 depth decoder and extends to transparent objects and sky regions.

2 Related Work

Monocular and Multi-View Depth Estimation. Early work on monocular depth estimation leveraged deep CNN regressors (Eigen et al., 2014) for single-image depth prediction. Subsequent work improved cross-dataset generalization (MiDaS (Ranftl et al., 2022)), adopted Vision Transformers for dense prediction (DPT (Ranftl et al., 2021)), and scaled the paradigm to millions of unlabeled images (Yang et al., 2024a;b). Researchers have also extended single-image depth estimation to multiple frames or views. DUSt3R (Wang et al., 2024) introduced a fully feed-forward framework for multi-view reconstruction, and MASt3R (Leroy et al., 2024) built on this foundation with 3D-aware feature matching for better multi-view accuracy. VGGT (Wang et al., 2025a), Pi-3 (Wang et al., 2025c), MapAnything (Keetha et al., 2025), Depth Anything 3 (Lin et al., 2026), and VGGT-Ω (Wang et al., 2026a) take this further by jointly predicting camera parameters, depth maps, or dense 3D points in a single transformer forward pass over arbitrary image collections. However, all these methods employ the unimodal depth modeling, limiting their ability to represent depth ambiguity at boundaries, transparent surfaces, and sky. Our method plugs directly into the final prediction head of such architectures and addresses this limitation with negligible computational overhead.

Depth Ambiguity at Boundaries, Transparent Surfaces, and Sky. Depth ambiguity is particularly pronounced at object boundaries, where flying-point artifacts degrade downstream performance. Pixel-Perfect Depth (Xu et al., 2025) and Marigold V2 (Pavlovic et al., 2026) use diffusion transformers to produce sharp depth boundaries from single images, while Pixel-Perfect Video Depth (Xu et al., 2026) extends this approach to video. InfiniDepth (Yu et al., 2026) represents monocular depth as a neural implicit field, with a decoder that predicts depth at continuous coordinates to recover fine-grained details. MoE3D (Wang et al., 2026b) uses a mixture-of-experts architecture to route spatial regions to specialized depth heads and is trained with a unimodal L2 loss. These methods leave many flying points unresolved, especially on blurry inputs. SMDNet (Tosi et al., 2021) is the closest in spirit to our formulation, modeling stereo disparity as a two-component Laplacian mixture, but it is restricted to stereo and does not extend to transparent surfaces or sky.

Depth ambiguity also arises on transparent surfaces, where multiple depths co-exist along a single ray. The Booster dataset (Ramirez et al., 2022) highlights the limitations of standard stereo methods on transparent and reflective materials, while LayeredDepth (Wen et al., 2025) introduces the first benchmark with multi-layer depth annotations. ClearGrasp (Sajjan et al., 2020) and SeeGroup (Wen & Deng, 2026) address transparent-object depth estimation through optimization and learning depth as unordered events, respectively. Despite these efforts, depth estimation for transparent objects remains a challenge. Sky pixels, whose true depth is effectively infinite, are typically handled outside the depth pipeline: Depth Anything 3 (Lin et al., 2026), for example, treats sky as a separate segmentation problem, training a new network alongside its depth estimator.

3 Methods

The introduction motivates our main idea: boundary pixels should be represented by multiple surface-valued depth hypotheses rather than by a single in-between depth. We now formalize this idea. We first show that the standard confidence-weighted depth loss corresponds to a unimodal likelihood (§3.1). We then replace this likelihood with a mixture-density representation that predicts multiple hypotheses and their probabilities (§3.2). Finally, we describe decoding, architecture, and the Gaussian-mixture variant used in our experiments.

3.1 Preliminary: Unimodal Depth Representation and Its Limitations

Modern depth estimators such as DA3 (Lin et al., 2026), VGGT (Wang et al., 2025a), and VGGT-Ω (Wang et al., 2026a) predict, for each pixel i, a depth $\hat{D}$ and a confidence C^D. The network is

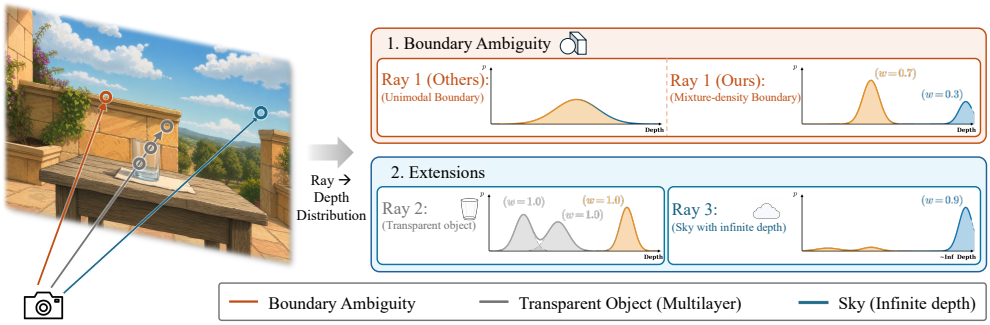


Figure 2: Three forms of depth ambiguity. **Ray 1 (Boundary):** the pixel straddles a foreground edge and a background surface, producing two depth hypotheses whose relative mixture weights encode the model's belief on which surface dominates. **Ray 2 (Transparent object):** the ray physically intersects multiple surfaces (e.g., the two sides of a glass cup and the background wall), and all of them are simultaneously valid depths. **Ray 3 (Sky):** the ray hits the sky (infinite depth in the end).

trained to minimize confidence-weighted L1 loss over all N pixels:

$$\mathcal{L}_{\text{Depth}} = \sum_{i=1}^{N} \left(C^D \|\hat{D} - D\|_1 - \alpha \log C^D \right), \tag{1}$$

where D is the ground-truth depth and $\alpha > 0$ prevents the trivial solution $C^D = 0$.

This loss has a probabilistic interpretation. Assume the ground-truth depth at pixel i follows a Laplacian distribution centered at the depth prediction $\hat{D}$ with a learned scale b, the probability of ground truth depth is: $p(D \mid \hat{D}, b) = \frac{1}{2b} \exp(-|\hat{D} - D|/b)$. With the reparameterization $C^D = \alpha/b$, the negative log-likelihood reduces to Eq. 1 up to a positive scale and an additive constant. Thus, the standard objective induces a unimodal per-pixel depth representation. The Gaussian case analogously yields a confidence-weighted ℓ_2 loss. Full derivations are provided in Appendix §A.1 and §A.2.

Reason from a Unimodal Representation to Flying Points. As discussed above, for standard depth regression models, each pixel is represented by one distribution centered at one depth. This is appropriate on smooth regions, but not at boundaries. Consider a boundary pixel that straddles a foreground surface at depth d_{fg} and a background surface at depth d_{bg}, with $d_{\text{bg}} \gg d_{\text{fg}}$. Its RGB observation and deep features can contain information from both surfaces, making the surface assignment ambiguous for the model. A unimodal predictor must nevertheless explain the pixel with one depth value. Under this ambiguity, the prediction often ends up at a compromise between d_{fg} and d_{bg}, which lies in empty space and becomes a flying point (Fig. 3, left; see the "Comparison to a Unimodal Head" paragraph in Appendix §A.4 for a detailed analysis).

3.2 Mixture-Density Depth Representation (MDA)

We replace the unimodal depth representation with a mixture-density representation that explicitly accounts for depth ambiguity (Fig. 3, right). Instead of requiring one predicted depth to explain every pixel, the model predicts multiple plausible depth hypotheses together with their probabilities. For each pixel, the prediction head outputs K depth hypotheses $\{\hat{D}_k\}_{k=1}^K$, scales $\{b_k\}_{k=1}^K$, and mixture weights $\{\pi_k\}_{k=1}^K$ with $\sum_{k=1}^K \pi_k = 1$ and $\pi_k \geq 0$. The mixture weights are produced by a softmax over per-component logits. The depth distribution can then be represented by:

$$p(D \mid \{\hat{D}_k, b_k, \pi_k\}_{k=1}^K) = \sum_{k=1}^{K} \pi_k \cdot \frac{1}{2b_k} \exp\left(-\frac{|\hat{D}_k - D|}{b_k} \right). \tag{2}$$

Each component represents one depth hypothesis. On ordinary pixels, all hypotheses may agree on the same surface. At boundaries, different hypotheses can specialize to different surfaces, e.g., foreground and background, while the mixture weights encode which surface is more likely.

Loss Derivation. Assuming independence across pixels, the training objective is the negative log-likelihood of the mixture density from Eq. 2 summed over all N pixels. We re-use the confidence reparameterization $C_k^D = \alpha/b_k$ from the unimodal case, and define a per-component loss $\mathcal{L}_k^{\text{Laplace}} =$

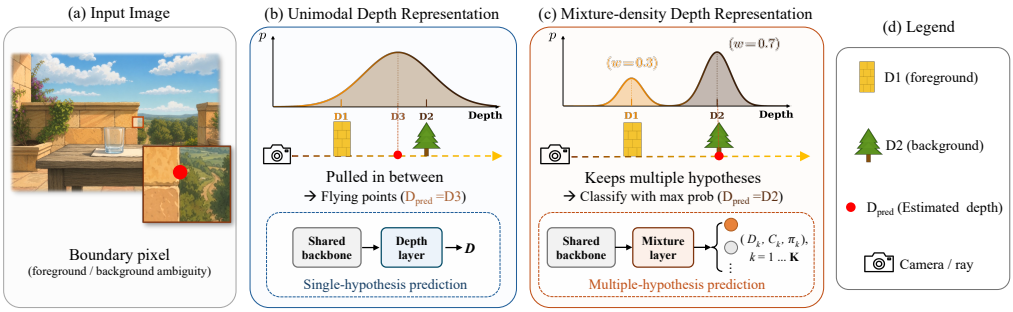


Figure 3: Unimodal versus mixture-density depth at an object boundary. (a) A boundary pixel may mix foreground and background evidence along the camera ray. (b) A unimodal predictor must output one depth, often ending up between the foreground (d_1) and background (d_2) at an intermediate estimate (d_3) that becomes a flying point. (c) Our mixture-density representation keeps multiple hypotheses and turns decoding into a selection among candidate surface depths, producing a boundary-aligned prediction.

$C_k^D|\hat{D}_k - D| - \alpha \log C_k^D$, which is exactly the loss of unimodal Laplacian (Eq. 1) applied to component k. Substituting everything into the mixture NLL and dropping constants yields:

$$\mathcal{L}_{\text{Mix}} = -\sum_{i=1}^{N} \log \sum_{k=1}^{K} \exp\left(\log \pi_k - \frac{1}{\alpha}\mathcal{L}_k^{\text{Laplace}}\right). \tag{3}$$

This representation naturally generalizes the unimodal case: when $K = 1$ with a single Laplace component, Eq. 3 reduces to Eq. 1. See the Appendix §A.1 for the full derivation.

3.3 Decoding Without Averaging

The mixture distribution represents several plausible depth hypotheses, but downstream applications usually require a single decoded depth map. At inference, we therefore select the component whose predicted depth is most likely under the learned mixture distribution. Concretely, for each candidate depth $\hat{D}_k$, we evaluate its density under Eq. 2 and choose the candidate with the highest score:

$$k^* = \underset{k\in\{1,\dots,K\}}{\text{argmax}} \sum_{j=1}^{K} \frac{\pi_j}{2b_j} \exp\left(-\frac{|\hat{D}_k - \hat{D}_j|}{b_j}\right), \qquad \hat{d} = \hat{D}_{k^*} \tag{4}$$

This requires only K density evaluations with nearly negligible overhead.

Discussion on Flying Points. At a boundary pixel, different mixture components can represent different candidate surfaces: one may explain the foreground depth, while another explains the background depth. At inference, even if the model is uncertain about which surface the pixel belongs to, the decoded depth is selected from the learned hypotheses. The output therefore lies on one of the candidate surfaces instead of floating between them. Appendix §A.4 provides a gradient analysis.

Discussion with MoE3D (Wang et al., 2026b). Our representation modifies the last layer of the depth predictor to produce per-component predictions and supervises them with our mixture NLL loss. This design has two entangled factors of improvement: higher network capability with additional predictions and our representation with mixture-density. Close to our work, MoE3D (Wang et al., 2026b) predicts multiple depths and averages them via softmax. Yet, MoE3D trains the model via a unimodal loss on the averaged depth, without mixture-density representation. We therefore compare with MoE3D and ablate the improvements in §5.1.3. The architectural change accounts for only a small portion of the improvement, while the mixture representation is the main driving force, demonstrating the effectiveness of our representation.

Network Architecture. Our mixture-density depth representation is architecture-agnostic: it can be plugged on top of most modern depth estimators (Lin et al., 2026; Wang et al., 2025a; Pavlovic et al., 2026) with minimal architectural change. We only modify the last output layer(s) of the depth head (e.g., the DPT head of DA3 and VGGT, or the VAE depth decoder of Marigold V2) to predict multiple depth hypotheses and their weights, leaving the backbone unchanged (see Appendix §B).

Table 1: Boundary quality analysis. DA3+Ours and VGGT+Ours cut boundary error over their baselines across all three datasets and run around 30× faster than diffusion-based PPD / PPVD. FPS is mean ± std over 5 runs.

Method	NRGBD				7Scenes				HiRoom				FPS↑
	Img		Seq		Img		Seq		Img		Seq		
	Acc↓	CD↓	Acc↓	CD↓	Acc↓	CD↓	Acc↓	CD↓	Acc↓	CD↓	Acc↓	CD↓	
PPD (Xu et al., 2025)	74.0	81.0	68.0	61.0	53.0	63.5	54.0	60.0	81.0	89.0	81.0	65.0	0.4 ±0.0
PPVD (Xu et al., 2026)	107.0	125.0	101.0	80.5	60.0	73.0	65.0	70.0	124.0	91.0	141.0	87.5	1.2 ±0.1
VGGT-Ω-1B (Wang et al., 2026a)	106.0	79.0	93.5	64.0	41.0	52.5	47.5	55.5	53.5	47.0	49.0	38.0	**53.3** ±7.9
VGGT (Wang et al., 2025a)	60.0	62.0	54.0	53.0	41.0	57.5	43.0	56.5	58.0	59.5	54.0	48.0	33.4 ±2.7
DA3 (Lin et al., 2026)	57.0	50.0	51.0	43.5	41.0	50.0	42.0	48.5	42.0	40.0	38.0	32.0	36.8 ±4.1
VGGT + Ours	28.0	38.0	27.0	34.0	**35.0**	42.0	37.0	42.5	45.0	50.0	42.0	40.5	34.1 ±2.9
DA3 + Ours (LMM)	**25.0**	35.5	**22.0**	**29.5**	37.0	42.0	38.0	42.0	**31.0**	34.5	**29.0**	**28.0**	33.3 ±2.6
DA3 + Ours (GMM)	**25.0**	**35.0**	24.0	30.5	**35.0**	**40.5**	**36.0**	**40.5**	**31.0**	**34.0**	30.0	**28.0**	33.3 ±2.6

3.4 Extension to a Gaussian Mixture Model

The Laplacian mixture above can be directly extended to a Gaussian Mixture Model (GMM) by replacing each Laplace component with a Gaussian distribution. The mixture NLL retains the same form as Eq. 3, with the per-component ℓ_1 term replaced by an ℓ_2 term; the full derivation is in §A.2. For training stability we apply the Gaussian mixture in log-depth space, following previous work (Xu et al., 2025; Keetha et al., 2025) (details in §A.3). All other design choices remain unchanged.

In our experiments, we find that the Gaussian Mixture Model outperforms the Laplacian Mixture Model on most benchmarks (see §5.1.3), likely because the ℓ_2 penalty provides stronger gradient signal for depth predictions at boundary areas and for non-dominant components. We therefore adopt the GMM as our default model unless otherwise stated.

4 Extensions to Other Depth Ambiguities

Our representation can be extended to other forms of depth ambiguities with minimal modifications. We study two forms of ambiguities in our paper: (I). Images of transparent objects (Fig. 2), where a pixel might indicate both the foreground glass and the background surface; (II). Images with sky (Fig. 2), where sky induces a special type of boundary.

4.1 Multi-Layer Depth for Transparent Objects

For transparent objects, a ray passing through glass physically intersects multiple surfaces. A pixel may therefore require multiple valid depth predictions: one for the visible transparent surface and another for the background behind it.

From Softmax to Sigmoid. The softmax mixture used for ordinary boundaries assumes that each pixel has one selected depth, since its weights are constrained to sum to one and the final prediction is decoded by selecting one component. To support transparent objects, we replace the softmax over mixture-weight logits with independent sigmoid weights, $\pi_k = \sigma(\pi'_k) \in (0, 1)$, so components are weighted independently rather than normalized against one another. Transparent pixels can then activate multiple components simultaneously, expressing co-existing depth layers along the ray. Opaque pixels are regularized to keep the weight sum close to one, retaining the boundary-handling behavior of the softmax variant. Full training and inference details are in §B.5.

4.2 Sky Estimation

Sky pixels have infinite depth, leading to extreme discontinuities at sky-scene boundaries and cause severe flying-point artifacts. Our representation can be extended to sky estimation by only adding an additional density component to account for the sky, leaving other components untouched.

Sky Component. The sky component has fixed mean μ_{sky} and scale b_{sky}, both set to large predefined constants and not updated during training. The resulting mixture is

$$p(D \mid \Theta) = \sum_{k=1}^{K} \pi_k \cdot \frac{1}{2b_k} \exp\left(-\frac{|\hat{D}_k - D|}{b_k}\right) + \pi_{K+1} \cdot \frac{1}{2b_{\text{sky}}} \exp\left(-\frac{|\mu_{\text{sky}} - D|}{b_{\text{sky}}}\right) \tag{5}$$

The finite-depth components continue to model scene geometry, while the sky component absorbs pixels whose depth should not be represented by a finite surface.

Table 2: Video Depth Evaluation. Our methods stay on par with their baselines (DA3 and VGGT) and substantially outperforms PPD/PPVD.

Method	Type	Sintel		Bonn		KITTI	
		Abs Rel ↓	δ<1.25 ↑	Abs Rel ↓	δ<1.25 ↑	Abs Rel ↓	δ<1.25 ↑
PPD (Xu et al., 2025)	Diff	0.473	39.8	0.315	53.2	0.221	62.8
PPVD (Xu et al., 2026)	Diff	0.330	51.6	0.164	72.1	0.069	97.6
MoE3D (Wang et al., 2026b)	FA	0.271	67.7	0.053	97.0	0.076	96.0
VGGT-Ω-1B (Wang et al., 2026a)	FA	**0.101**	**91.3**	**0.044**	97.2	0.063	97.6
VGGT (Wang et al., 2025a)	FA	0.297	68.8	0.055	97.1	0.073	96.5
DA3 (Lin et al., 2026)	FA	0.307	66.7	0.049	97.2	0.061	97.5
VGGT + Ours (GMM)	FA	0.241	67.4	0.076	94.2	0.047	**98.0**
DA3 + Ours (LMM)	FA	0.333	57.9	0.049	**97.4**	0.049	97.9
DA3 + Ours (GMM)	FA	0.223	67.0	0.053	97.2	**0.044**	97.7

Inference. We classify pixel i as sky when the sky component carries the largest mixture weight among all components, i.e., $\pi_{K+1} \geq \max_{k \leq K} \pi_k$. This gives a sky mask directly from the mixture weights without a separate segmentation network, and prevents finite-depth components from creating flying points along the skyline.

5 Experiments

We organize the experiments into two parts: §5.1 evaluates boundary and depth quality with feed-forward multi-view predictors and tests whether the representation transfers to a diffusion-based single-image decoder (§5.1.4), and §5.2 evaluates the transparent-object and sky-region variants.

5.1 Boundary and Depth Quality

5.1.1 Experimental Settings

Implementation Details For DA3 (Lin et al., 2026) and VGGT (Wang et al., 2025a), we replace only the final DPT prediction layer, using $K=4$ components unless stated otherwise, each predicting depth $\hat{D}_k$, confidence C_k^D, and mixture-weight π_k. All evaluated DA3 and VGGT models include the sky component (§4.2). We finetune the pretrained checkpoints for 10k steps; details in Appendix §B.

Training and Evaluation Datasets. For DA3 and VGGT, we train on a mixture of synthetic datasets and evaluate boundary quality on `NRGBD` (Azinović et al., 2022), `7Scenes` (Shotton et al., 2013), and `HiRoom` (Lin et al., 2026), and video depth on `Sintel` (Butler et al., 2012), `Bonn` (Palazzolo et al., 2019), and `KITTI` (Geiger et al., 2013). The full dataset list is provided in Appendix §B.3.

Evaluation Metrics. Following Pixel-Perfect-Depth (Xu et al., 2025), boundary quality uses edge-masked point clouds with Chamfer Distance (CD↓) and Accuracy (Acc↓) at image and sequence level. Video depth uses AbsRel↓ and δ<1.25↑ following Cut3r (Wang et al., 2025b) and Stream3r (Lan et al., 2025); FPS measures inference speed. Exact metric definitions, alignment rules, and the timing protocol are provided in Appendix §B.3.

5.1.2 Experimental Results

Boundary Quality. Table 1 shows that DA3+Ours achieves the lowest boundary CD (Chamfer Distance) and Acc across all datasets and granularities, often by a large margin. The gains are most visible in Acc, which measures the distance from each predicted point to its closest ground-truth point and is therefore particularly sensitive to flying points: such points lie far from both the foreground and background surfaces and incur large predicted-to-GT distances. Figure 5 provides qualitative comparisons. All the baselines produce flying-point artifacts at object boundaries, while our predictions stay on foreground or background surfaces.

Inference Speed. The last column of Table 1 shows that our head adds negligible overhead: VGGT+Ours and DA3+Ours run at 34.1 and 33.3 FPS versus 33.4 and 36.8 for their baselines, and run around 30× faster than the multi-step diffusion methods PPD and PPVD.

Video Depth Estimation. Our representation targets boundaries, which cover only a small fraction of each depth map, so standard depth accuracy should be preserved rather than improved. Table 2 reports video depth on Sintel, Bonn, and KITTI. VGGT-Ω-1B, which benefits from substantially

Table 3: Ablation of representation and architecture. The largest gains come from replacing the unimodal depth representation with a mixture-density representation.

Method	NRGBD				HiRoom			
	Img		Seq		Img		Seq	
	Acc↓	CD↓	Acc↓	CD↓	Acc↓	CD↓	Acc↓	CD↓
Single-head + Unimodal ℓ_1 (DA3)	57.0	50.0	51.0	43.5	42.0	40.0	38.0	32.0
Single-head + Unimodal ℓ_1 (Finetuned DA3)	54.0	49.0	48.0	42.5	38.0	36.0	35.0	30.0
Multi-head + Unimodal ℓ_1	50.0	46.5	44.0	40.0	39.0	36.5	36.0	30.5
Multi-head + Unimodal ℓ_2 + entropy (MoE3D-style)	56.0	49.0	49.5	45.0	46.5	46.0	44.0	37.0
Multi-head + Mixture-density ℓ_1 (Ours)	**25.0**	**35.5**	**22.0**	**29.5**	**31.0**	**34.5**	**29.0**	**28.0**

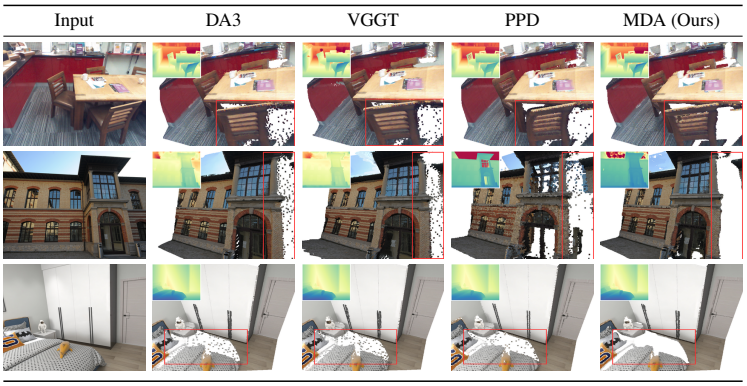


Figure 5: Qualitative boundary comparison. Baseline methods (DA3, VGGT, PPD) leave visible flying points on the boundaries, while our approach keeps the boundary clean.

larger training data, achieves the best overall accuracy. Measured against their own baselines, however, our models improve or stay on par: DA3+Ours lowers AbsRel on Sintel (0.223 vs. 0.307) and KITTI (0.044 vs. 0.061) and is slightly worse on Bonn (0.053 vs. 0.049), while VGGT+Ours improves on Sintel and KITTI with a drop on Bonn.

5.1.3 Experimental Analysis

Per-Component Analysis. We visualize each mixture component in Appendix Fig. 14. The components specialize spatially: each component dominates a different region of the scene. At boundaries, different components can account for the foreground or the background with a clean separation, reducing flying points. Additional scenes are provided in Appendix §C.1.3.

Robustness to Input Blur. Our mixture-density representation is especially useful when boundary evidence is weakened by blur. We simulate degraded inputs by downsampling each frame by factor s with area averaging and bicubic upsampling it back to the model resolution; larger s corresponds to stronger blur. We use the same evaluation protocol as above, and neither our model nor the baselines are trained with blur-specific augmentation. Figure 4 plots Acc on NRGBD as a function of s; qualitative results are provided in Appendix Fig. 7. Blur makes the surface assignment less reliable. For unimodal baselines, every uncertain pixel is predicted between the two surfaces; for our model, it is at worst assigned to the wrong surface. Consequently, our absolute error grows more slowly.

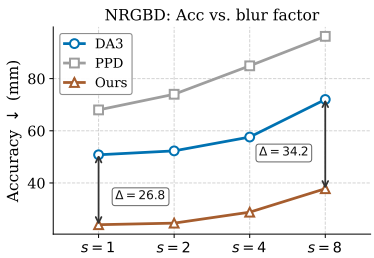
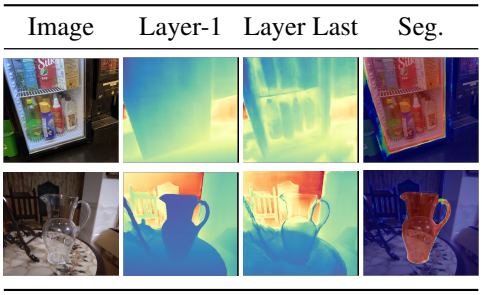
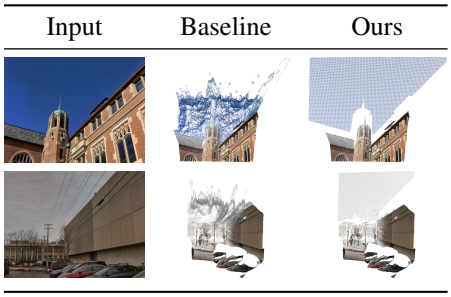


Figure 4: Boundary estimation Accuracy on NRGBD as a function of input blur s (Acc↓, mm). Our mixture model degrades less compared to baselines.

Ablation: Representation vs. Architecture. Our mixture-density formulation changes both the network architecture (single → multi-head) and the representation (unimodal → mixture density). To isolate their effects, Table 3 ablates the two design choices on the DA3 backbone. We also include a multi-head baseline with ℓ_2 loss and entropy regularization, following MoE3D, which is intended as a controlled architectural comparison rather than a reproduction of MoE3D. Except



(a) Our multi-layer depth prediction on transparent objects: input image, predicted first depth layer (visible transparent surface), last depth layer (occluded geometry behind it), and transparency segmentation.



(b) Sky comparison: input image, baseline trained without the sky component (flying points along the entire skyline), and our model with the sky component (clean sky boundaries).

Figure 6: Qualitative results on transparent objects and sky.

for the first row (the original DA3), all variants are trained with the same data and optimization setting. Finetuning and multi-head architecture bring modest gains, while replacing the unimodal representation with a mixture density accounts for most of the boundary improvement. This indicates that the representation, rather than the extra heads alone, is the main driver of performance.

5.1.4 Generalization to a Diffusion-Based Depth Decoder

Our representation only changes how depth is parameterized at the output layer, so it is architecture-agnostic and not tied to DPT-style regression heads. To test this, we instantiate it in the depth VAE decoder of Marigold V2 (Pavlovic et al., 2026), a latent-diffusion single-image depth estimator, keeping the diffusion backbone unchanged and following the Marigold V2 training recipe otherwise (details in Appendix §B.4). Following the Marigold V2 protocol, Table 4 reports Soft Edge Error (SEE$_k$ ↓, $k \in \{3, 5, 7\}$) on the HyperSim test set (Roberts et al., 2021), which compares each boundary prediction with ground-truth disparities in a local $k \times k$ neighborhood. The mixture decoder consistently improves over Marigold V2 at all neighborhood sizes and outperforms PPD and InfiniDepth.

Table 4: Single-image boundary quality (SEE$_k$ ↓) on HyperSim test with the Marigold V2 decoder.

Method	SEE$_3$ ↓	SEE$_5$ ↓	SEE$_7$ ↓
PPD (Xu et al., 2025)	0.404	0.385	0.371
InfiniDepth (Yu et al., 2026)	0.470	0.451	0.436
Marigold V2 (Pavlovic et al., 2026)	0.352	0.333	0.320
Marigold V2 + Ours	**0.329**	**0.310**	**0.298**

5.2 Experiments on Extensions

Transparent Object Depth. We evaluate multi-layer depth estimation on the LayeredDepth benchmark (Wen et al., 2025), which provides a synthetic validation set and a real-world validation set with human annotations; full quantitative results are deferred to §C.2.1. For transparent objects, our sigmoid-weighted mixture formulation (§4.1) recovers both the visible transparent surface and the occluded background behind it. Figure 6a shows multi-layer predictions on real LayeredDepth scenes: despite training only on synthetic supervision, the mixture head produces a clean first-layer depth aligned with the visible transparent surface and a plausible last-layer depth behind it.

Sky Estimation. To validate the dedicated sky component, we evaluate sky-segmentation quality on Sintel, reporting IoU against its semantic-segmentation ground truth; full quantitative results are deferred to §C.2.2. Figure 6b shows qualitative comparisons against a baseline identical to ours except for the missing sky component. Without the sky component, the baseline must place every sky pixel at a finite depth, producing flying points along the entire skyline. In contrast, our model assigns sky pixels to the sky component, producing clean sky boundaries.

6 Conclusion

We presented a mixture-density depth representation that removes *flying points* by letting ambiguous boundary pixels choose among surface-valued depth hypotheses instead of landing between them. Being architecture-agnostic, it improves boundary reconstruction over each base model (DA3, VGGT, and Marigold V2) at negligible cost, and extends to transparent surfaces and sky.

AI USE STATEMENT

We did not use generative AI tools for any task with required disclosure: they were not used to generate data, develop the method or its theoretical formulation, formulate or prove mathematical claims, propose hypotheses, design experiments, implement methods, translate, clean or analyze data, or interpret results. We used generative AI tools to edit the paper for readability, to draft parts of the experimental description and the appendix, to suggest the structure of the paper, and to create or edit auxiliary software code that is not part of the method implementation. We have reviewed all AI-assisted work: every AI-edited or AI-drafted sentence was checked by the authors against the original manuscript and the experimental results, and all AI-assisted code was verified and tested for correctness by all authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Full derivations of the Laplacian and Gaussian mixture objectives and the gradient analysis are given in Appendix §A. Architecture, training objectives, initialization, curriculum, and optimization settings for the DA3 and VGGT models are in Appendix §B; the Marigold V2 decoder adaptation in §B.4; and the transparent-object variant in §B.5. Training and evaluation datasets, metric definitions, alignment rules, and the timing protocol are listed in Appendix §B.3. Code and pretrained models will be released upon publication.

REFERENCES

Armen Avetisyan, Christopher Xie, Henry Howard-Jenkins, Tsun-Yi Yang, Samir Aroudj, Suvam Patra, Fuyang Zhang, Duncan Frost, Luke Holland, Campbell Orme, Jakob Engel, Edward Miller, Richard Newcombe, and Vasileios Balntas. SceneScript: Reconstructing scenes with an autoregressive structured language model. *arXiv preprint arXiv:2403.13064*, 2024.

Dejan Azinović, Ricardo Martin-Brualla, Dan B. Goldman, Matthias Nießner, and Justus Thies. Neural RGB-D surface reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Pierre Blanchard, Desmond J Higham, and Nicholas J Higham. Accurately computing the log-sum-exp and softmax functions. *IMA Journal of Numerical Analysis*, 41(4):2311–2330, 2021.

Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*, 2012.

Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2. *arXiv preprint arXiv:2001.10773*, 2020.

David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. *Advances in neural information processing systems*, 27, 2014.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, 2013.

Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. DynamicStereo: Consistent dynamic depth from stereo videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.

Nikhil Keetha, Norman Müller, Johannes Schönberger, Lorenzo Porzi, Yuchen Zhang, Tobias Fischer, Arno Knapitsch, Duncan Zauss, Ethan Weber, Nelson Antunes, et al. Mapanything: Universal feed-forward metric 3d reconstruction. *arXiv preprint arXiv:2509.13414*, 2025.

Yushi Lan, Yihang Luo, Fangzhou Hong, Shangchen Zhou, Honghua Chen, Zhaoyang Lyu, Shuai Yang, Bo Dai, Chen Change Loy, and Xingang Pan. Stream3r: Scalable sequential 3d reconstruction with causal transformer. *arXiv preprint arXiv:2508.10893*, 2025.

Vincent Leroy, Yohann Cabon, and Jérôme Revaud. Grounding image matching in 3d with mast3r. In *European conference on computer vision*, pp. 71–91. Springer, 2024.

Haotong Lin, Sili Chen, Jun Hao Liew, Donny Y. Chen, Zhenyu Li, Yang Zhao, Sida Peng, Hengkai Guo, Xiaowei Zhou, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth Anything 3: Recovering the visual space from any views. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=yirunib8l8.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22160–22169, 2024.

Emanuele Palazzolo, Jens Behley, Philipp Lottes, Philippe Giguère, and Cyrill Stachniss. ReFusion: 3d reconstruction in dynamic environments for RGB-D cameras exploiting residuals. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

Xiaqing Pan, Nicholas Charron, Yongqian Yang, Scott Peters, Thomas Whelan, Chen Kong, Omkar Parkhi, Richard Newcombe, and Yuheng Carl Ren. Aria digital twin: A new benchmark dataset for egocentric 3d machine perception. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 20133–20143, 2023.

Igor Pavlovic, Thiemo Wandel, Anton Obukhov, Luca Bartolomei, Andrey Davydov, Fabio Tosi, Matteo Poggi, Sabine Süsstrunk, and Dengxin Dai. Marigold V2: Revisiting diffusion transformers for monocular depth estimation. *arXiv preprint arXiv:2609.08084*, 2026.

Pierluigi Zama Ramirez, Fabio Tosi, Matteo Poggi, Samuele Salti, Stefano Mattoccia, and Luigi Di Stefano. Open challenges in deep stereo: the booster dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21168–21178, 2022.

René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 12179–12188, 2021.

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2022.

Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10912–10922, 2021.

Shreeyak Sajjan, Matthew Moore, Mike Pan, Ganesh Nagaraja, Johnny Lee, Andy Zeng, and Shuran Song. Clear grasp: 3d shape estimation of transparent objects for manipulation. In *2020 IEEE international conference on robotics and automation (ICRA)*, pp. 3634–3642. IEEE, 2020.

Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in RGB-D images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2013.

Fabio Tosi, Yiyi Liao, Carolin Schmitt, and Andreas Geiger. Smd-nets: Stereo mixture density networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8942–8952, 2021.

Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025a.

Jianyuan Wang, Minghao Chen, Shangzhan Zhang, Nikita Karaev, Johannes Schönberger, Patrick Labatut, Piotr Bojanowski, David Novotny, Andrea Vedaldi, and Christian Rupprecht. VGGT-Ω. *arXiv preprint arXiv:2605.15195*, 2026a.

Qianqian Wang, Yifei Zhang, Aleksander Holynski, Alexei A Efros, and Angjoo Kanazawa. Continuous 3d perception model with persistent state. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 10510–10522, 2025b.

Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20697–20709, 2024.

Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian Scherer. Tartanair: A dataset to push the limits of visual slam. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4909–4916. IEEE, 2020.

Yifan Wang, Jianjun Zhou, Haoyi Zhu, Wenzheng Chang, Yang Zhou, Zizun Li, Junyi Chen, Jiangmiao Pang, Chunhua Shen, and Tong He. π^3: Permutation-equivariant visual geometry learning. *arXiv preprint arXiv:2507.13347*, 2025c.

Zichen Wang, Ang Cao, Liam J. Wang, and Jeong Joon Park. MoE3D: A mixture-of-experts module for 3D reconstruction. *arXiv preprint arXiv:2601.05208*, 2026b.

Hongyu Wen and Jia Deng. SeeGroup: Multi-layer depth estimation of transparent surfaces via self-determined grouping. *arXiv preprint arXiv:2605.28735*, 2026.

Hongyu Wen, Yiming Zuo, Venkat Subramanian, Patrick Chen, and Jia Deng. Seeing and seeing through the glass: Real and synthetic data for multi-layer depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6715–6725, 2025.

Gangwei Xu, Haotong Lin, Hongcheng Luo, Xianqi Wang, Jingfeng Yao, Lianghui Zhu, Yuechuan Pu, Cheng Chi, Haiyang Sun, Bing Wang, et al. Pixel-perfect depth with semantics-prompted diffusion transformers. *arXiv preprint arXiv:2510.07316*, 2025.

Gangwei Xu, Haotong Lin, Hongcheng Luo, Haiyang Sun, Bing Wang, Guang Chen, Sida Peng, Hangjun Ye, and Xin Yang. Pixel-perfect visual geometry estimation. *arXiv preprint arXiv:2601.05246*, 2026.

Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10371–10381, 2024a.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. In *Advances in Neural Information Processing Systems*, volume 37, pp. 21875–21911, 2024b.

Hao Yu, Haotong Lin, Jiawei Wang, Jiaxin Li, Yida Wang, Xueyang Zhang, Yue Wang, Xiaowei Zhou, Ruizhen Hu, and Sida Peng. InfiniDepth: Arbitrary-resolution and fine-grained depth estimation with neural implicit fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26920–26930, June 2026.

Yang Zheng, Adam W. Harley, Bokui Shen, Gordon Wetzstein, and Leonidas J. Guibas. PointOdyssey: A large-scale synthetic dataset for long-term point tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023.

Yang Zhou, Yifan Wang, Jianjun Zhou, Wenzheng Chang, Haoyu Guo, Zizun Li, Kaijing Ma, Xinyue Li, Yating Wang, Haoyi Zhu, Mingyu Liu, Dingning Liu, Jiange Yang, Zhoujie Fu, Junyi Chen, Chunhua Shen, Jiangmiao Pang, Kaipeng Zhang, and Tong He. OmniWorld: A multi-domain and multi-modal dataset for 4D world modeling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=1y1YFKb9pp.

Supplementary Material

Modeling Depth Ambiguity: A Mixture-Density Representation for Flying-Point-Free Depth Estimation

This supplementary material is organized as below. §A provides the full derivations deferred from the main paper: including the complete Laplacian and Gaussian mixture derivation, the log-depth parameterization used at training, and a gradient analysis showing why the mixture representation is robust at depth boundaries. §B describes the architecture, training objective, and optimization for the DA3/VGGT boundary models, the Marigold V2 decoder adaptation (§B.4), and the transparent-object variant. §C reports additional experiments: boundary and depth quality (§C.1, including multi-view reconstruction, ablations on the number of components, and the inference rule, and extra qualitative results) and the two extensions (§C.2, including additional transparent-object and sky-component results). Finally, §D discusses failure cases and limitations of our method.

A Method Details

This section expands the derivations that were abbreviated in the main paper. We first show why the standard confidence-weighted ℓ_1 loss is a unimodal Laplacian NLL, then derive the Laplacian mixture loss (Eq. 3). We then present the unimodal and mixture Gaussian variants in one place (§3.1 and §3.4) and close with the log-depth parameterization used for the Gaussian variant.

A.1 Full Derivation of the Laplacian Mixture Representation

Unimodal Laplacian. For a single pixel i, the unimodal Laplacian model from §3.1 places a Laplace distribution centered at the predicted depth $\hat{D}$ with scale b:

$$p(D \mid \hat{D}, b) \;=\; \frac{1}{2b} \exp\left(-\frac{|D - \hat{D}|}{b} \right). \tag{6}$$

Assuming pixels are independent, the negative log-likelihood over all N pixels is:

$$\mathcal{L}_{\text{Uni}} = \sum_{i=1}^{N} \left(\frac{|\hat{D} - D|}{b} + \log b + \log 2 \right). \tag{7}$$

Using the confidence reparameterization $C^D = \alpha / b$, we have $b = \alpha / C^D$ and therefore:

$$\begin{aligned} \mathcal{L}_{\text{Uni}} &= \frac{1}{\alpha} \sum_{i=1}^{N} \Big(C^D |\hat{D} - D| - \alpha \log C^D + \alpha \log(2\alpha) \Big) \\ &= \frac{1}{\alpha} \mathcal{L}_{\text{Depth}} + N \log(2\alpha). \end{aligned} \tag{8}$$

Thus minimizing the confidence-weighted ℓ_1 loss in Eq. 1 is equivalent to minimizing the NLL of a unimodal Laplacian depth distribution, up to a positive scale and an additive constant.

Laplacian Mixture. The Laplacian mixture (§3.2, Eq. 2) replaces this single component with a convex combination of K Laplacian densities:

$$p(D \mid \{\hat{D}_k, b_k, \pi_k\}_{k=1}^{K}) \;=\; \sum_{k=1}^{K} \pi_k \cdot \frac{1}{2b_k} \exp\left(-\frac{|D - \hat{D}_k|}{b_k} \right), \tag{9}$$

with mixture weights $\pi_k \in [0, 1]$ satisfying $\sum_{k=1}^{K} \pi_k = 1$ and per-component scales $b_k > 0$. Each component represents one depth hypothesis weighted by its mixing probability.

Loss Derivation. We provide the full derivation of the Laplacian mixture loss (Eq. 3 in the main paper). Starting from the negative log of the mixture density above and applying the confidence reparameterization $C_k^D = \alpha / b_k$, the log-likelihood of a single Laplace component k becomes:

$$\log \frac{1}{2b_k} \exp\left(-\frac{|\hat{D}_k - D|}{b_k} \right) = -\frac{1}{\alpha} \left(C_k^D |\hat{D}_k - D| - \alpha \log C_k^D \right) - \log(2\alpha). \tag{10}$$

The term inside the parentheses is exactly the per-component version of the unimodal Laplacian loss from Eq. 1, now applied to each mixture component independently.

Let $\mathcal{L}_k^{\text{Laplace}} = C_k^D|\hat{D}_k - D| - \alpha \log C_k^D$ denote this per-component loss. Substituting back into the mixture NLL, taking the negative log, and dropping the constant $\log(2\alpha)$ yields the final loss (Eq. 3):

$$\mathcal{L}_{\text{Mix}} = -\sum_{i=1}^{N} \log \sum_{k=1}^{K} \exp\left(\log \pi_k - \frac{1}{\alpha}\mathcal{L}_k^{\text{Laplace}}\right). \tag{11}$$

LogSumExp Stabilization. Direct evaluation of $\log \sum_k \exp(a_k)$ is numerically unstable when any a_k is large. We stabilize it with the LogSumExp trick (Blanchard et al., 2021). Defining $a_k = \log \pi_k - \frac{1}{\alpha}\mathcal{L}_k^{\text{Laplace}}$ and $a^* = \max_k a_k$:

$$\log \sum_k e^{a_k} = a^* + \log \sum_k e^{a_k - a^*}, \tag{12}$$

where all terms $a_k - a^* \leq 0$, so the exponentials never overflow.

A.2 Full Derivation of the Gaussian Mixture Representation

Unimodal Gaussian. For a single pixel i, the unimodal Gaussian model places a Gaussian distribution centered at the predicted depth $\hat{D}$ with variance σ^2:

$$p(D \mid \hat{D}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\|\hat{D} - D\|^2}{2\sigma^2}\right). \tag{13}$$

Assuming pixels are independent, the negative log-likelihood over all N pixels is:

$$\mathcal{L}_{\text{NLL}} = \sum_{i=1}^{N} \left(\frac{\|\hat{D} - D\|^2}{2\sigma^2} + \tfrac{1}{2}\log \sigma^2 + \tfrac{1}{2}\log(2\pi)\right). \tag{14}$$

Using the confidence reparameterization $C^D = \alpha/\sigma^2$, we have $\sigma^2 = \alpha/C^D$ and therefore:

$$\begin{aligned}\mathcal{L}_{\text{NLL}} &= \frac{1}{2\alpha}\sum_{i=1}^{N}\Big(C^D\|\hat{D} - D\|^2 - \alpha \log C^D + \alpha\log(2\pi\alpha)\Big) \\ &= \frac{1}{2\alpha}\mathcal{L}_{\text{Gaussian}} + \tfrac{N}{2}\log(2\pi\alpha),\end{aligned} \tag{15}$$

where $\mathcal{L}_{\text{Gaussian}} = \sum_{i=1}^{N}\Big(C^D\|\hat{D} - D\|^2 - \alpha \log C^D\Big)$ is the confidence-weighted ℓ_2 loss. Compared with Eq. 1, the Gaussian formulation replaces the absolute error with the squared error but retains the same confidence-weighted data-fidelity plus log-barrier structure.

Gaussian Mixture. The Gaussian mixture (§3.4) replaces this single component with a convex combination of K Gaussian densities:

$$p(D \mid \{\hat{D}_k, \sigma_k^2, \pi_k\}_{k=1}^K) \;=\; \sum_{k=1}^{K} \pi_k \cdot \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{\|\hat{D}_k - D\|^2}{2\sigma_k^2}\right), \tag{16}$$

with mixture weights $\pi_k \in [0, 1]$ satisfying $\sum_{k=1}^K \pi_k = 1$ and per-component variances $\sigma_k^2 > 0$. As in the Laplacian mixture, each component represents one depth hypothesis; the Gaussian assumption changes the component shape from an ℓ_1-based density to an ℓ_2-based density.

Loss Derivation. Starting from the negative log of the Gaussian mixture density and applying the confidence reparameterization $C_k^D = \alpha/\sigma_k^2$, the log-likelihood of a single Gaussian component k becomes:

$$\log \frac{1}{\sqrt{2\pi\sigma_k^2}} \exp\left(-\frac{\|\hat{D}_k - D\|^2}{2\sigma_k^2}\right) = -\frac{1}{2\alpha}\left(C_k^D\|\hat{D}_k - D\|^2 - \alpha \log C_k^D\right) - \tfrac{1}{2}\log(2\pi\alpha). \tag{17}$$

Let $\mathcal{L}_k^{\text{Gaussian}} = C_k^D \|\hat{D}_k - D\|^2 - \alpha \log C_k^D$ denote the per-component Gaussian loss. Substituting back into the mixture NLL, taking the negative log, and dropping the constant $\frac{1}{2}\log(2\pi\alpha)$ yields the final loss:

$$\mathcal{L}_{\text{GMM}} = -\sum_{i=1}^{N} \log \sum_{k=1}^{K} \exp\left(\log \pi_k - \frac{1}{2\alpha}\mathcal{L}_k^{\text{Gaussian}}\right). \tag{18}$$

The only differences from the Laplacian mixture loss $\mathcal{L}_{\text{Mix}}$ are the squared error in $\mathcal{L}_k^{\text{Gaussian}}$ and the factor $1/(2\alpha)$ in the exponent, instead of $1/\alpha$ for the Laplacian case. The LogSumExp stabilization of §A.1 applies unchanged.

A.3 Log-Depth Parameterization for Gaussian Mixture

Raw depth values span a very large dynamic range across scenes (from centimeters at near-field objects to tens of meters outdoors), which makes a Gaussian likelihood in linear depth poorly calibrated: the ℓ_2 penalty disproportionately magnifies errors at far distances. Following (Xu et al., 2025; Keetha et al., 2025), we therefore apply the Gaussian mixture in log-depth space.

Concretely, every component mean $\hat{D}_k$, variance σ_k^2, and the ground-truth target are mapped through $f(D) = \log(D+\epsilon)$ before the loss is evaluated, where ϵ is a small constant (0.1 in our implementation) that prevents numerical instability for near-zero depth values. The inverse map $D = \exp(f) - \epsilon$ is applied at inference to recover linear depth. In the DA3/VGGT experiments, the Laplacian mixture uses the original depth space, since its ℓ_1 penalty is already robust to the heavy-tailed depth distribution and does not benefit from log-space reparameterization in those experiments.

A.4 Gradient Analysis: Why the Mixture-Density Representation Is Robust at Boundaries

This section explains the boundary robustness of our mixture representation from a **gradient perspective**. At ambiguous boundary pixels, some training labels may correspond to the foreground surface and others to the background surface, even when the local image evidence is similar. We show that the mixture NLL tolerates this ambiguity without dragging depth components into the empty space between foreground and background surfaces, thus avoiding flying points.

This robustness follows from the gradient structure of the mixture NLL: each component's update is gated by its posterior responsibility, so a component that already fits the foreground receives almost no depth gradient from a background label, and vice versa (Fig. 14). A unimodal head has no such mechanism: its single output implicitly encodes both the surface assignment and the depth, so changing the assignment drags the prediction through the empty space between surfaces. We give the derivation for the Laplacian mixture (LMM); the Gaussian mixture has the same form.

Setup. Consider a single pixel with ground-truth depth D and predicted Laplacian mixture

$$p(D) \;=\; \sum_{k=1}^{K} \pi_k \cdot \frac{1}{2b_k} \exp\left(-\frac{|D - \hat{D}_k|}{b_k}\right), \qquad \pi_k \;=\; \frac{\exp(\pi'_k)}{\sum_{j=1}^{K} \exp(\pi'_j)}, \tag{19}$$

where mixture weights π_k come from a softmax over per-component logits π'_k. The per-pixel NLL is $\mathcal{L} := -\log p(D)$. We define the *posterior responsibility* of component k as

$$\gamma_k \;:=\; \frac{\pi_k \frac{1}{2b_k} \exp(-|D - \hat{D}_k|/b_k)}{p(D)}, \qquad \sum_{k=1}^{K} \gamma_k \;=\; 1, \tag{20}$$

i.e. the posterior probability that component k explains the observed depth D under the current mixture parameters.

Gradients are Gated by γ_k. For any parameter θ_k of component k (i.e. $\hat{D}_k$ or b_k), differentiating $\mathcal{L}$ and using Eq. 20 gives

$$\frac{\partial \mathcal{L}}{\partial \theta_k} \;=\; -\gamma_k \cdot \frac{\partial \log p_k(D)}{\partial \theta_k}, \qquad \log p_k(D) \;=\; -\log(2b_k) - \frac{|D - \hat{D}_k|}{b_k}. \tag{21}$$

The mixture gradient on each component is exactly the single-Laplacian gradient on that component, scaled by the responsibility γ_k. In particular, for the depth prediction $\hat{D}_k$ we obtain

$$\frac{\partial \mathcal{L}}{\partial \hat{D}_k} \;=\; -\,\gamma_k \,\frac{\operatorname{sgn}(D-\hat{D}_k)}{b_k}. \tag{22}$$

Boundary-Robustness Consequence. Consider the two-component case (K=2) at a converged boundary pixel. Suppose component 1 has captured the foreground surface ($\hat{D}_1 \approx d_{\mathrm{fg}}$) and component 2 has captured the background surface ($\hat{D}_2 \approx d_{\mathrm{bg}}$), with $|d_{\mathrm{fg}} - d_{\mathrm{bg}}| \gg b_1, b_2$. If the ground-truth label at this pixel falls on the foreground ($D \approx d_{\mathrm{fg}}$), the density ratio in Eq. 20 becomes

$$\frac{p_1(D)}{p_2(D)} \;=\; \frac{b_2}{b_1}\,\exp\!\left(\frac{|D-d_{\mathrm{bg}}|}{b_2} - \frac{|D-d_{\mathrm{fg}}|}{b_1}\right),$$

which is exponentially large because $|D - d_{\mathrm{fg}}| \approx 0$ while $|D - d_{\mathrm{bg}}| \approx |d_{\mathrm{fg}} - d_{\mathrm{bg}}| \gg b_2$. Hence $\gamma_1 \approx 1$ and $\gamma_2 \approx 0$. Substituting $\gamma_2 \approx 0$ into Eq. 22 gives $\partial\mathcal{L}/\partial\hat{D}_2 \approx 0$: *the background component receives essentially no depth gradient*, so the mis-assigned label cannot drag $\hat{D}_2$ off the background surface and no flying point arises. The mixing-logit gradients $\partial\mathcal{L}/\partial\pi'_k = \pi_k - \gamma_k$ shift weight from π_2 toward π_1, while the depth values $\hat{D}_1, \hat{D}_2$ stay locked to their respective surfaces.

Comparison to a Unimodal Head. A unimodal head implicitly encodes the surface assignment and the depth in the same value $\hat{D}$. Switching a boundary pixel from the background to the foreground therefore means moving $\hat{D}$ from d_{bg} to d_{fg}, through the empty space between them. Since this implicit assignment is imperfect, flying points arise both during training and at test time.

Training. For the unimodal head, suppose $\hat{D} \approx d_{\mathrm{bg}}$ while the label is $D = d_{\mathrm{fg}}$. The gradient $\partial\mathcal{L}/\partial\hat{D} = -\operatorname{sgn}(D-\hat{D})/b$ pulls $\hat{D}$ off the background surface toward the foreground, so the prediction passes through the gap and stays at an intermediate depth until the assignment is fully corrected. For ambiguous boundary pixels, where the model is unsure which surface the pixel belongs to, this correction is slow and often incomplete, leaving the prediction in the gap.

Test time. On an unseen image, where the network is unsure which surface a boundary pixel belongs to, the unimodal prediction often lies between the two surfaces, which is a flying point.

Mixture. The mixture stores the assignment explicitly in π_k and the depths in $\hat{D}_k$. A mis-assigned label updates the mixing weights through $\partial\mathcal{L}/\partial\pi'_k = \pi_k - \gamma_k$, while the other component's depth receives almost no gradient ($\gamma_k \approx 0$), so no depth is dragged into the gap. For ambiguous boundary pixels, the uncertainty therefore affects only the mixing weights, while each depth stays on its own surface. At test time, an uncertain pixel still outputs one of the learned depths: a wrong assignment places the point on the wrong surface, never in between.

A Common Misunderstanding: Multi-Depth Supervision Is Not Required. There might be a misunderstanding that training a multi-depth representation requires multi-depth ground truth, e.g., one annotated depth per surface at boundary pixels. This is not the case. Although each training pixel carries only a single depth label, boundary pixels with similar local image evidence can receive *different* labels depending on subpixel foreground/background coverage. Under a unimodal loss, a single prediction must fit all of them, and whenever the prediction lies on the wrong surface, the label from the other surface pulls it into the gap. Under our mixture likelihood, different components instead explain different subsets of these labels through their posterior responsibilities γ_k (Eq. 20), as the gradient analysis above shows. The mixture therefore acquires multi-surface specialization from ordinary single-depth supervision, without any multi-layer or per-surface annotations.

B Extra Implementation Details

B.1 Model Architecture

Our mixture representation can be integrated into a direct depth head or a depth VAE decoder. In the DA3 and VGGT instantiations, we replace the final DPT output projection. For each of the K components, the head emits three per-pixel quantities: (i) a depth prediction $\hat{D}_k$, (ii) a confidence

$C_k^D = \alpha/b_k$, and (iii) a mixture-weight logit, normalized via softmax over the K components to produce π_k. The mixture parameters therefore require $3K$ scalar outputs per pixel, with $K{=}4$ by default for these two models. The Marigold V2 decoder adaptation is described separately in §B.4.

B.2 Implementation Details for DA3 and VGGT Boundary Models

Training Objective. The full training objective combines the mixture depth loss (Eq. 3) with a camera pose loss and point cloud loss inherited from DA3. For camera pose, we use the output of DA3's lightweight camera head rather than the camera-ray prediction from the Dual-DPT head, both during training and at inference. The camera pose loss is:

$$\mathcal{L}_{\text{pose}} = \sum_{t=1}^{N} \left(\|\hat{\boldsymbol{q}}_t - \boldsymbol{q}_t\|_2 + \left\| \frac{\hat{\boldsymbol{\tau}}_t}{\hat{s}} - \frac{\boldsymbol{\tau}_t}{s} \right\|_2 + \left\| \hat{\boldsymbol{f}}_t - \boldsymbol{f}_t \right\|_2 \right), \tag{23}$$

where $\boldsymbol{q}_t$, $\boldsymbol{\tau}_t$, and $\boldsymbol{f}_t$ denote the per-frame rotation quaternion, translation, and focal length, and s, $\hat{s}$ are the ground-truth and predicted global scale factors calculated similar to (Wang et al., 2025c).

Initialization, Curriculum, and Optimization. We instantiate the mixture head on the multiview depth estimator DA3-GIANT (Lin et al., 2026) and VGGT (Wang et al., 2025a), and initialize from the corresponding pretrained checkpoint. Because our mixture head replicates the original final prediction layer into K parallel branches, all K copies start with identical weights. To break the symmetry between mixture heads, we add zero-mean Gaussian noise to each duplicated weight tensor, with standard deviation $0.1 \cdot \text{mean}(|w|)$ where w is the corresponding original weight.

Training proceeds in three stages, each using the same optimizer (learning rate $1\text{e}{-4}$, effective batch size 48 via gradient accumulation, on 4 RTX Pro 6000 GPUs), for a total of $11{,}000$ steps: (i) we freeze the backbone and all DPT-head layers except the new mixture projection and train only the mixture-head weights for $1{,}000$ steps; (ii) we unfreeze the remaining DPT-head layers — the backbone is still frozen — and train for another $5{,}000$ steps; and (iii) we additionally unfreeze the local-attention layers of the backbone and train for $5{,}000$ steps. This curriculum stabilizes the early phase of training and preserves the backbone's pretrained capacity, letting us match the underlying model's depth quality with substantially less training data and compute. Following DA3, the base resolution is 504×504 and training image resolutions are randomly sampled from $\{504{\times}504$, $504{\times}378$, $504{\times}336$, $504{\times}280$, $504{\times}210$, $504{\times}154$, $378{\times}504$, $336{\times}504$, $280{\times}504$, $672{\times}504\}$, while the number of views is sampled uniformly from $[2, 24]$.

B.3 Experimental Protocol Details for DA3 and VGGT

Training and Evaluation Datasets. Following DA3, we train on a mix of synthetic datasets: `AriaSyntheticENV` (Avetisyan et al., 2024), `HyperSim` (Roberts et al., 2021), `MvsSynth` (Huang et al., 2018), `OmniWorld` (Zhou et al., 2026), `PointOdyssey` (Zheng et al., 2023), `TartanAir` (Wang et al., 2020), `vKitti2` (Cabon et al., 2020), `DynamicReplica` (Karaev et al., 2023), and `UnrealStereo4K` (Tosi et al., 2021). For evaluation, we use `NRGBD` (Azinović et al., 2022), `7Scenes` (Shotton et al., 2013), and `HiRoom` (Lin et al., 2026) for boundary quality, and `Sintel` (Butler et al., 2012), `Bonn` (Palazzolo et al., 2019), and `KITTI` (Geiger et al., 2013) for video depth estimation.

Evaluation Metrics. For **boundary quality**, we follow exactly the same setting as Pixel-Perfect-Depth (Xu et al., 2025): we extract edge masks from ground-truth depth maps with Canny operator and evaluate metrics on the masked point clouds. We report Chamfer Distance (CD↓) and Accuracy (Acc↓; mean predicted-to-GT distance) at two granularities: *per-image* (frame-level) and *per-sequence* (scene-level, aggregating point clouds across frames). We emphasize Acc because it is particularly sensitive to flying points: such points lie far from both foreground and background surfaces, resulting in large predicted-to-GT distances. For **inference speed**, we benchmark on a single L40S GPU at 504×384 resolution; the timing covers model inference and depth decoding. Each measurement is repeated five times on the same frames, and Table 1 reports the mean and standard deviation. For **video depth estimation**, we follow the per-frame protocol of Cut3r (Wang et al., 2025b) and Stream3r (Lan et al., 2025): we report Absolute Relative error (AbsRel↓, the mean of $\frac{|\hat{D}-D|}{D}$) and threshold accuracy $\delta{<}1.25$ (↑, the fraction of pixels with $\max\left(\frac{\hat{D}}{D}, \frac{D}{\hat{D}}\right) < 1.25$). Predictions are

aligned to the ground truth before evaluation: a single global scale per sequence for DA3, VGGT, and Ours, and a per-frame scale and shift for PPD and PPVD.

All VGGT-Ω comparisons use the 1B-parameter model (VGGT-Ω-1B), consistent with the main-paper experiments.

B.4 Marigold V2: Decoder Adaptation and Boundary Evaluation

Decoder Adaptation. We initialize from the Stage-2 Marigold V2 checkpoint (Pavlovic et al., 2026). The model maps a single RGB image to depth latents with a single-pass diffusion transformer (DiT) and decodes them into a depth map through a VAE decoder. The mixture head follows our DA3 and VGGT instantiations (§B), including the per-component outputs, $K{=}4$, and the initialization. The only difference is that each of the K parallel branches replicates the last two upsampling blocks of the VAE depth decoder rather than only the final prediction layer, giving about 30M trainable parameters. The VAE encoder, the DiT, and the earlier decoder layers remain frozen, so the new decoder tail is trained directly on the latents produced by the frozen DiT.

Training Objective. For this instantiation we use a Laplacian mixture, rather than the default Gaussian mixture used in the multi-view experiments. We otherwise follow the Stage-2 training recipe of Marigold V2 and retain its SinkLoss, adding the mixture NLL on top:

$$\mathcal{L}_{\text{Marigold}} = \lambda_{\text{mix}}\mathcal{L}_{\text{LapMix}} + \lambda_{\text{sink}}\mathcal{L}_{\text{SinkLoss}}. \tag{24}$$

Here $\mathcal{L}_{\text{LapMix}}$ is the mixture NLL of Eq. 3, evaluated in the decoder's supervised output space, and $\mathcal{L}_{\text{SinkLoss}}$ is the SinkLoss of Marigold V2 with its original settings. SinkLoss uses local matching to accommodate spatial uncertainty in depth supervision, while the mixture represents alternative depths at an individual pixel, so the two objectives are complementary.

Boundary Evaluation. We evaluate the mixture-decoder variant on the HyperSim test set, using the Soft Edge Error metrics of Table 2 in Marigold V2 (Pavlovic et al., 2026). For a boundary pixel i, SEE_k takes the minimum absolute disparity error between the prediction at i and valid ground-truth disparities in a $k \times k$ neighborhood, for $k \in \{3, 5, 7\}$. Ground-truth disparity discontinuities define the boundary pixels. This local tolerance evaluates compatibility with nearby surfaces without requiring exact pixelwise edge alignment. Qualitative examples are shown in Figure 8.

B.5 Implementation Details for Transparent Object Variant

Training Objective. Unlike the sky component, which is part of our main DA3 and VGGT models, the transparent-object variant is trained separately because it requires multi-layer supervision. For the transparent-object variant of §4.1, we use $K{=}2$ depth heads, with the first supervised on the visible (front) surface and the second on the occluded (back) surface for transparent pixels. The full per-pixel training loss combines a transparency-aware depth loss with a weight-regularization term:

$$\mathcal{L}_{\text{trans}} = \mathcal{L}_{\text{depth}} + \mathcal{L}_{\text{weights}}. \tag{25}$$

Depth loss. On opaque pixels, only one surface is visible and we keep the mixture NLL from the main paper (Eq. 3); the two heads remain free to maintain different depth hypotheses, retaining the boundary handling of the main model. On transparent pixels, both layers are visible and we instead supervise each head independently on its assigned ground-truth layer with a single-component Laplacian NLL:

$$\mathcal{L}_{\text{depth}} = \begin{cases} \mathcal{L}_1^{\text{Lap}}(D^{(1)}) \;+\; \mathcal{L}_2^{\text{Lap}}(D^{(L)}), & \text{if pixel is transparent,} \\ \mathcal{L}_{\text{mix}}(D), & \text{otherwise (opaque),} \end{cases} \tag{26}$$

where $D^{(1)}$ and $D^{(L)}$ are the first-layer (visible) and last-layer (occluded) ground-truth depths from LayeredDepth, $\mathcal{L}_k^{\text{Lap}}(g)$ is the single-component Laplacian NLL of §3.1 (Eq. 1) applied to head k with label g, and $\mathcal{L}_{\text{mix}}(D)$ is the mixture NLL of §3.2 (Eq. 3).

Weight-regularization loss. To drive the mixture weights toward the correct regime per pixel, we add

$$\mathcal{L}_{\text{weights}} = \begin{cases} \|\pi_0 - 1\|^2 + \|\pi_1 - 1\|^2, & \text{if pixel } i \text{ is transparent} \\ \|\pi_0 + \pi_1 - 1\|^2, & \text{otherwise.} \end{cases} \tag{27}$$

The first case drives both heads toward full activation, preserving both depth layers; the second recovers softmax-like sum-to-one behavior, so the boundary-aware selection from §3.3 still applies on opaque pixels.

Inference and Decoding. For simplicity, we set $K{=}2$ in this variant, modeling only two depth layers — the visible transparent surface and the background behind it. The weight-regularization loss $\mathcal{L}_{\text{weights}}$ encourages the mixture weights of both heads to be close to one on transparent pixels (so $\sum_k \pi_k \approx 2$) and the two weights to sum to one on opaque pixels (so $\sum_k \pi_k \approx 1$). At inference, we therefore use the weight sum as a soft transparency indicator and threshold it at the midpoint of the two regimes: when $\sum_k \pi_k > 1.5$, we treat the pixel as transparent and output both depth predictions as two separate layers, the first head giving the first (visible) layer and the second head the last (occluded) layer; otherwise we treat it as opaque and select a single depth via the component-selection rule of §3.3. The same transparent/opaque decision yields the transparency segmentation visualized in Fig. 6a, without any additional segmentation head. This produces multi-layer depth on transparent pixels while preserving sharp boundaries on opaque ones.

Initialization, Curriculum, and Optimization. We instantiate the mixture head on the single-view depth estimator DA3MONO-LARGE (Lin et al., 2026) and use the same initialization as the boundary-handling model. Training also follows the same three-stage curriculum, on 4 RTX Pro 6000 GPUs with learning rate $1\text{e}{-4}$ and batch size 128, for a total of $31{,}000$ steps: $1{,}000$ steps for the new mixture projection alone, $5{,}000$ steps with the rest of the DPT head unfrozen (backbone still frozen), and $25{,}000$ steps with the local-attention layers of the backbone additionally unfrozen. Following DA3, the base resolution is 504×504 and training image resolutions are randomly sampled from $\{504{\times}504,\ 504{\times}378,\ 504{\times}336,\ 504{\times}280,\ 378{\times}504,\ 336{\times}504,\ 672{\times}504\}$.

B.6 Implementation Details for Sky Component

The sky component uses $\mu_{\text{sky}} = 10^4$ and $b_{\text{sky}} = 1$ in linear depth; all other settings are identical to those of the DA3 and VGGT boundary models (§B.2). This component is included in all DA3 and VGGT models evaluated in this paper.

C Additional Experiments

This section reports additional results that complement §5.1 and §5.2 of the main paper. §C.1 contains additional results on boundary and depth quality (multi-view reconstruction, ablations, qualitative comparisons, per-component visualizations, and failure cases). §C.2 contains additional results on the transparent-object and sky-region variants.

C.1 Boundary and Depth Quality

Evaluation Metrics. We use two families of metrics. For **multi-view 3D reconstruction** on NRGBD, 7Scenes, and HiRoom we follow DA3 (Lin et al., 2026) and report Accuracy (Acc↓; mean distance from predicted to ground-truth points, in mm), Completeness (Comp↓; mean distance from ground-truth to predicted points, in mm), and Normal Consistency (NC↑; cosine similarity of surface normals). Each metric is reported as both mean and median across pixels. For **boundary quality** we follow Pixel-Perfect-Depth (Xu et al., 2025): we extract edge masks from the ground-truth depth with the Canny operator (low/high hysteresis thresholds 100/200) and compute Chamfer Distance (CD↓) and Accuracy (Acc↓) in millimeters on the resulting boundary point clouds at two granularities — *per-image* (frame-level) and *per-sequence* (scene-level, aggregating point clouds across frames in a sequence). To remove depth-scale bias, we align predictions to the ground truth before evaluation: for DA3, VGGT, and Ours, we fit a single global scale per sequence; for PPD and PPVD, we instead fit a per-frame scale and shift, which gives them the benefit of optimal local alignment. We emphasize Acc on boundary point clouds because it directly penalizes flying points: a point in the empty space between foreground and background lies far from *both* surfaces, so its predicted-to-GT distance grows with the foreground–background gap and is not absorbed by either alignment.

C.1.1 Experimental Results

Multi-View Reconstruction. Following DA3 (Lin et al., 2026), we evaluate multi-view 3D reconstruction on NRGBD, 7Scenes, and HiRoom, reporting accuracy (Acc), completeness (Comp),

Table 5: Multi-view 3D reconstruction. Our methods stays on par with their baselines (DA3 and VGGT) and substantially outperforms PPD/PPVD across datasets.

Method	NRGBD						7Scenes						HiRoom					
	Acc↓		Comp↓		NC↑		Acc↓		Comp↓		NC↑		Acc↓		Comp↓		NC↑	
	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.	Mean	Med.
PPD (Xu et al., 2025)	45.0	27.0	22.0	10.0	0.855	0.978	34.0	20.0	33.0	18.0	0.795	0.908	58.0	37.0	19.0	9.0	0.820	0.963
PPVD (Xu et al., 2026)	96.0	52.0	29.0	16.0	0.726	0.845	43.0	27.0	36.0	18.0	0.717	0.819	116.0	69.0	29.0	13.0	0.705	0.825
VGGT-Ω-1B (Wang et al., 2026a)	15.0	9.5	**8.0**	5.0	0.884	0.987	27.5	17.0	28.5	18.0	0.788	0.900	14.0	9.5	**7.5**	5.0	0.872	0.986
VGGT (Wang et al., 2025a)	20.0	13.0	15.0	6.0	0.882	0.988	29.0	16.0	29.0	16.0	0.764	0.878	28.0	18.0	15.0	7.0	0.859	0.985
DA3 (Lin et al., 2026)	**10.0**	**5.0**	11.0	**4.0**	**0.948**	**0.995**	24.0	13.0	27.0	14.0	0.805	0.917	**11.0**	**6.0**	8.0	**4.0**	**0.919**	**0.993**
VGGT + Ours (GMM)	16.0	10.0	15.0	6.0	0.886	0.988	24.0	13.0	25.0	13.0	0.805	0.913	24.0	15.0	14.0	7.0	0.855	0.982
DA3 + Ours (LMM)	12.0	7.0	11.0	**4.0**	0.904	0.987	**22.0**	**11.0**	**23.0**	**11.0**	0.791	0.901	13.0	8.0	8.0	**4.0**	0.869	0.980
DA3 + Ours (GMM)	11.0	7.0	12.0	**4.0**	0.912	0.990	23.0	**11.0**	25.0	12.0	**0.811**	**0.918**	14.0	8.0	9.0	**4.0**	0.878	0.985

Table 6: Ablation on the number of mixture components K. K=2 and K=4 yield comparable quality, while K=8 is slightly worse, likely because more components make the model harder to converge.

Method	NRGBD			HiRoom		
	CD↓	Video Boundary CD↓	Image Boundary CD↓	CD↓	Video Boundary CD↓	Image Boundary CD↓
Ours K=2	11.5	31.5	36.0	**11.5**	29.5	36.0
Ours K=4	11.5	**30.5**	**35.0**	**11.5**	**28.0**	**34.0**
Ours K=6	**10.5**	31.5	37.5	**11.5**	29.5	37.5
Ours K=8	12.5	33.0	38.0	12.0	35.0	44.5

and normal consistency (NC) with both mean and median values. As shown in Table 5, our mixture representation stays on par with the corresponding baseline on all three datasets: DA3+Ours achieves the best completeness on 7Scenes and the best or second-best accuracy on HiRoom, while remaining within a small margin on NRGBD. This confirms that the boundary improvements reported in the main paper come at little cost to standard scene reconstruction.

C.1.2 Experimental Analysis

Ablation: Number of Components. Table 6 varies the number of mixture components $K \in \{2, 4, 6, 8\}$. K=2 and K=4 yield comparable full-cloud and boundary CD, indicating that a small handful of components is already sufficient to capture the bimodal geometry at occlusion boundaries; further increasing K does not provide consistent gains and slightly hurts boundary quality on HiRoom at K=8, likely because more components make the model harder to converge. We therefore adopt K=4 as the default, since it matches the accuracy of larger K while using fewer head parameters.

Ablation: Inference Strategy. In §3.3 we adopt the *component-selection* rule at depth inference: the mixture density is evaluated at each component's mode, and the component with the largest value is returned as the point prediction. We compare it against two alternatives in Table 7. *(i) Mixture argmax* takes the argmax of the continuous mixture density directly. The two rules approximately coincide when the components are well-separated and yield near-identical boundary scores, but the continuous argmax requires per-pixel numerical optimization, making it much slower than the mode-selection variation. *(ii) Expectation* decodes the mixture by its mean, $\mathbb{E}[\hat{D}] = \sum_k \pi_k \hat{D}_k$. At a boundary pixel, the expectation lands in the empty space between the surfaces, recreating the flying-point failure. The boundary scores for this strategy are about four times worse than those of mode selection. We therefore adopt mode selection as the default.

C.1.3 Extra Qualitative Results

Boundary Comparisons. Figure 13 expands the boundary comparison of Figure 5 in the main paper. Each row shows the input image and results generated by the DA3 baseline, the VGGT baseline, PPD, and our model. Across every scene, the three baselines generate many flying points across object boundaries — along chair legs, table edges, doorways, and human silhouettes — while our mixture head preserves clean boundary geometry.

Per-Component Visualizations. Figure 14 visualizes the individual depth maps and mixture weights predicted by each of the K components. At boundary pixels, different components capture the foreground and background surfaces separately, while in smooth regions the components converge

Table 7: Ablation on inference strategy. Mode selection (our default) and Mixture argmax give nearly identical boundary scores, but mode selection runs roughly 2× faster. Decoding the mixture by its expectation performs much worse since it averages the foreground and background hypotheses and generates flying points. FPS is mean ± std over 5 runs.

Strategy	NRGBD				HiRoom				FPS↑
	Img		Seq		Img		Seq		
	Acc↓	CD↓	Acc↓	CD↓	Acc↓	CD↓	Acc↓	CD↓	
Mode selection (default)	**25.0**	**35.0**	**24.0**	**30.5**	**31.0**	**34.0**	**30.0**	**28.0**	33.3 ±2.6
Mixture argmax	26.0	35.5	**24.0**	**30.5**	**31.0**	**34.0**	**30.0**	**28.0**	22.3 ±6.9
Expectation	114.0	90.0	101.0	76.0	67.0	54.0	62.0	45.0	**34.5** ±1.9



Figure 7: Qualitative boundary reconstruction under input blur on six scenes. As s increases, baselines (DA3, PPD) accumulate thick bands of flying points at boundaries, while our model preserves clean boundary separation throughout.

to similar depths with one dominant weight — matching the qualitative behavior predicted by the mixture representation in §3.2.

Boundary Reconstruction Under Blur. Figure 7 provides qualitative results for the blur-robustness experiment of §5.1. The unimodal baselines accumulate thicker flying-point bands as s grows, because blur makes the surface assignment less reliable and every uncertain pixel is predicted between the two surfaces. Our mixture head keeps each component anchored to one surface (foreground or background), so the predicted boundary stays sharp even at s=8, consistent with the gradient-gating analysis of §A.4.

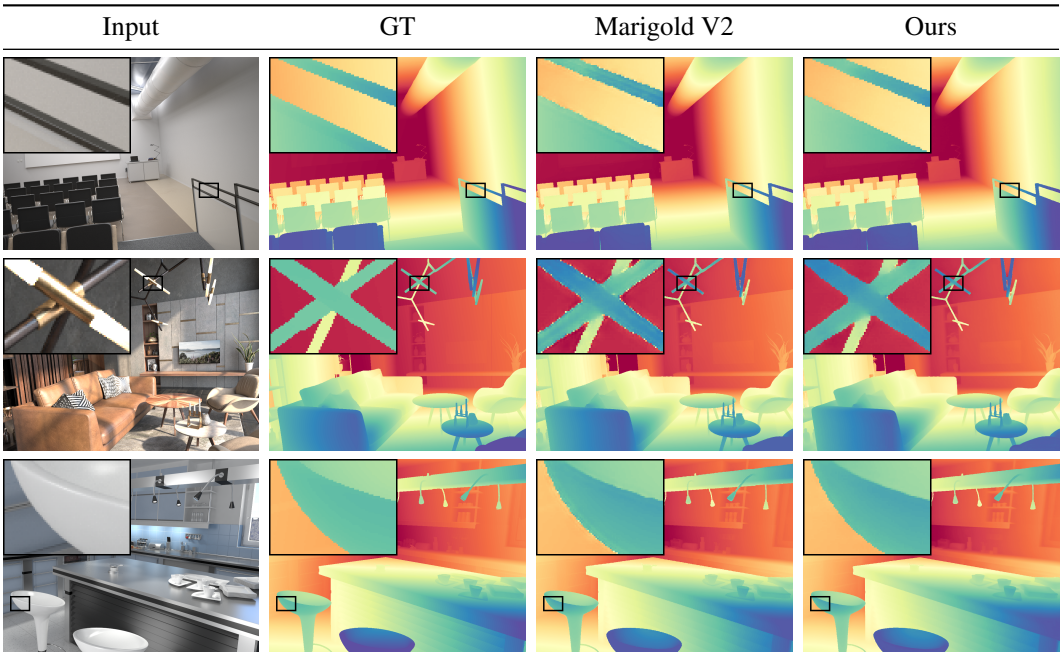


Figure 8: Qualitative single-image boundary comparison with the Marigold V2 decoder on the HyperSim test set; insets enlarge the boxed regions. The original decoder leaves speckles of intermediate depth along the handrail, lamp frame, and stool, while our mixture decoder keeps the discontinuity as sharp as the ground truth.

Single-Image Boundaries with the Marigold V2 Decoder. Figure 8 provides qualitative results for the diffusion-decoder experiment of §5.1.4. Each case shows the input image and zoomed-in depth crops from the ground truth, the original Marigold V2 decoder, and our mixture decoder. Trained with a unimodal objective, the original decoder leaves intermediate-depth pixels along object boundaries — on handrails, thin lamp frames, and stools — which become flying points once lifted to 3D, while our mixture decoder assigns each boundary pixel to one of the two adjacent surfaces, consistent with the SEE improvements in Table 4.

C.2 Experiments on Extensions

Evaluation Metrics. For **transparent object depth** on the LayeredDepth benchmark (Wen et al., 2025) we follow the protocol of Wen et al. (2025). On the synthetic validation set we report AbsRel (↓) and δ<1.25 (↑) for the first (visible) and last (occluded) depth layers, and edge-aware boundary Accuracy (Acc; ↓, mm) and Chamfer Distance (CD; ↓, mm) computed on the first layer alone and across all layers. On the real-world validation set, which provides human-annotated multi-layer depth orderings, we report ordering accuracy (↑) at three granularities — pairwise, triplet, and quadruplet. For **sky estimation** on Sintel we report sky-segmentation IoU (↑) computed against the Sintel semantic-segmentation ground truth, using the sky mask obtained by argmax over the mixture weights (§4.2).

C.2.1 Transparent Object Depth

We evaluate multi-layer depth estimation on the LayeredDepth benchmark (Wen et al., 2025), which provides a synthetic validation set and a real-world validation set with human annotations.

Synthetic. Following (Wen et al., 2025), on the synthetic validation set we report AbsRel and δ<1.25 accuracy for the first and last depth layers, and we additionally report edge-aware boundary Accuracy (Acc) and Chamfer Distance (CD) to quantify how cleanly transparent surfaces are reconstructed (Table 8). Our sigmoid-weighted mixture formulation improves over the DA3 baseline on every column. The GMM variant also improves on all columns over the stronger DA3-Multilayer baseline, which predicts two depth layers without the mixture loss, while the LMM variant is slightly worse on the last layer.

Table 8: Transparent object depth on the LayeredDepth synthetic validation set. Our GMM variant outperforms both DA3 and DA3-Multilayer baselines.

Method	First Layer		Last Layer		Boundary Acc↓		Boundary CD↓	
	AbsRel↓	δ<1.25 ↑	AbsRel↓	δ<1.25 ↑	First	All	First	All
DA3 (Lin et al., 2026) (monocular)	0.158	0.771	0.290	0.546	80.4	72.0	89.8	131.1
DA3-Multilayer (monocular)	0.118	0.872	0.202	0.734	72.9	75.5	81.1	95.6
SeeGroup (Wen & Deng, 2026) (monocular)	0.118	0.875	0.282	0.699	83.4	164.0	81.5	125.0
Ours (monocular, LMM)	0.107	0.879	0.213	0.718	**59.9**	70.3	74.0	**90.3**
Ours (monocular, GMM)	**0.100**	**0.896**	**0.182**	**0.758**	60.1	**68.8**	**71.7**	90.6

Table 9: Multi-layer depth ordering on the LayeredDepth real-world validation set with human annotations. All methods are evaluated on the same filtered subset, using only the first and last depth layers (§C.2.1). Bold and underline mark the best and second-best scores.

Method	Pairs ↑	Triplets ↑	Quadruplets ↑
DA3 (Lin et al., 2026) (monocular)	0.789	<u>0.620</u>	<u>0.314</u>
DA3-Multilayer (monocular)	0.752	0.550	0.256
SeeGroup (Wen & Deng, 2026) (monocular)	**0.816**	**0.635**	**0.320**
Ours (monocular, LMM)	<u>0.807</u>	0.606	0.269
Ours (monocular, GMM)	0.806	0.619	0.268

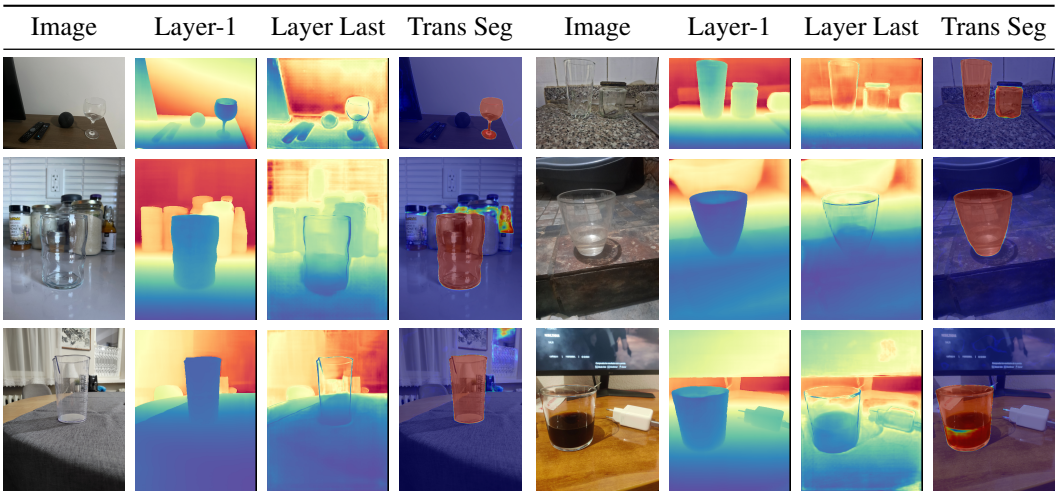


Figure 9: Qualitative multi-layer depth on the LayeredDepth real-world set (Wen et al., 2025). For each scene we show, left to right: the input image, the predicted first depth layer (visible transparent surface), the predicted last depth layer (occluded geometry behind it), and the transparency segmentation derived from the mixture-weight sum (§4.1).

Real-World. Table 9 reports pairwise, triplet, and quadruplet ordering accuracy on 300 real-world LayeredDepth validation images (Wen et al., 2025), evaluating only the first and last depth layers. Many of the original annotations are too noisy to verify: even human observers cannot determine the number of glass layers or their depth order. We therefore keep only the images for which two annotators independently agree that the depth order can be determined visually, and evaluate all methods on this same subset. We will release the filtered test set. Ground-truth layer 1 is mapped to the predicted first depth, while layers 3, 5, and 7 are mapped to the predicted last depth; this differs from the official protocol requiring four independently predicted layers. A tuple is correct only if its predictions strictly increase in annotated order; no scale or shift alignment is applied. SeeGroup achieves the highest scores on all three metrics. Our LMM and GMM variants improve pair accuracy over DA3 (0.789 to 0.807 and 0.806, respectively), but have lower triplet and quadruplet accuracy. Both variants outperform DA3-Multilayer on all three metrics.

Qualitative Examples. Figure 9 shows additional multi-layer predictions on real LayeredDepth scenes, expanding on the two examples in Figure 6a of the main paper.

Table 10: Sky segmentation on three outdoor Sintel sequences. "DA3-nested" uses the dedicated sky-segmentation network shipped alongside DA3 (Lin et al., 2026); "Ours" obtains sky masks from the dedicated sky component (§4.2) by taking the argmax over mixture weights.

Method	`alley_2`	`temple_2`	`temple_3`
DA3-nested (Lin et al., 2026)	**0.961**	0.823	**0.715**
Ours	0.951	**0.826**	1e-5

C.2.2 Sky Estimation

Quantitative. We evaluate the dedicated sky component on Sintel sequences (only three sequences have a clear and separable sky: `alley_2`, `temple_2`, `temple_3`). Table 10 reports sky-segmentation IoU obtained from the sky component's argmax (§4.2) and compares against the dedicated sky-segmentation network shipped alongside DA3 (Lin et al., 2026).

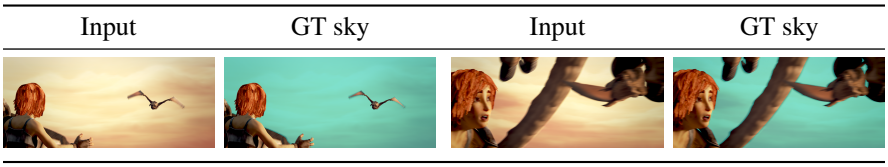


Figure 10: Two representative frames from Sintel `temple_3`: the input RGB and the ground-truth sky mask (green = sky). Sky pixels dominate most of the frame — a sky-dominant configuration that does not appear in our synthetic training mix — which explains the near-zero IoU on this sequence reported in Table 10.

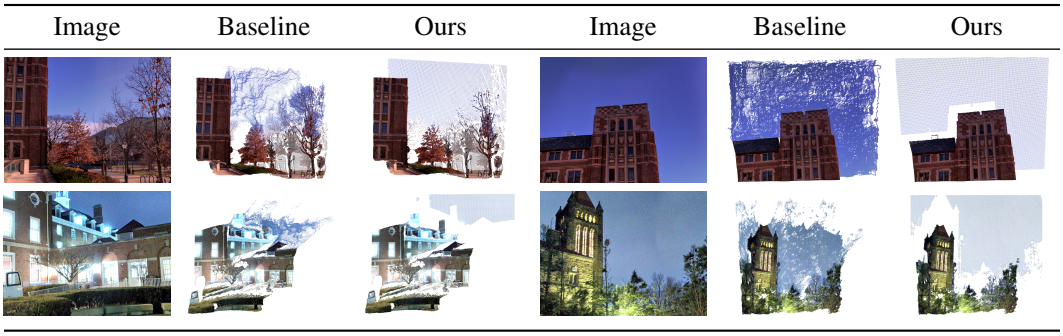


Figure 11: Qualitative comparison on sky regions. Without a dedicated sky component, the baseline produces flying points along the entire skyline; our model assigns sky pixels to the sky component, producing clean sky boundaries.

The near-zero IoU on `temple_3` in Table 10 is a training-distribution artifact rather than a failure of the formulation. As Figure 10 illustrates, sky pixels occupy most of the frame in this sequence — a sky-dominant configuration that does not appear in our synthetic training mix. DA3's nested sky-segmentation network does not disclose its training data, but is presumably trained on a substantially more diverse image distribution that includes such scenes. We expect this gap to close given comparable training coverage of sky-dominant outdoor data, and we leave a more comprehensive sky-training mix to future work.

Qualitative Examples. Figure 11 shows additional sky-handling comparisons (input / baseline / ours) on Sintel sequences, expanding on the two examples in Figure 6b of the main paper.

D Failure Cases and Limitations

Figure 12 highlights two characteristic failure modes of our mixture representation. *First* (Figure 12, left), although our method produces far fewer flying points than prior approaches, some depth artifacts still occasionally appear on object surfaces oriented nearly parallel to the camera viewing direction. At such grazing angles the appearance cue degenerates — a small image patch projects to a long range of depths, so some artifacts still occasionally appear on these surfaces. *Second* (Figure 12, right), we inherit the failure modes of the DA3 backbone itself: depth predictions on small thin

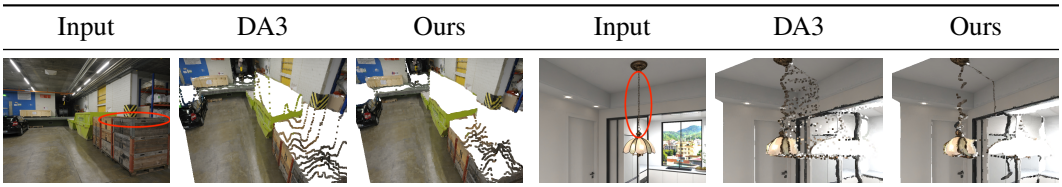


Figure 12: Failure cases. *Left:* Although our method produces far fewer flying points than the baselines overall, depth artifacts still appear on surfaces oriented nearly parallel to the camera viewing direction, where the grazing-angle appearance gives weak depth cues. *Right:* We inherit failure modes of the DA3 backbone: depth on small thin structures is locally distorted.

structures are locally distorted. Since our model is finetuned from the DA3 checkpoint, it is hard to rescue these predictions.

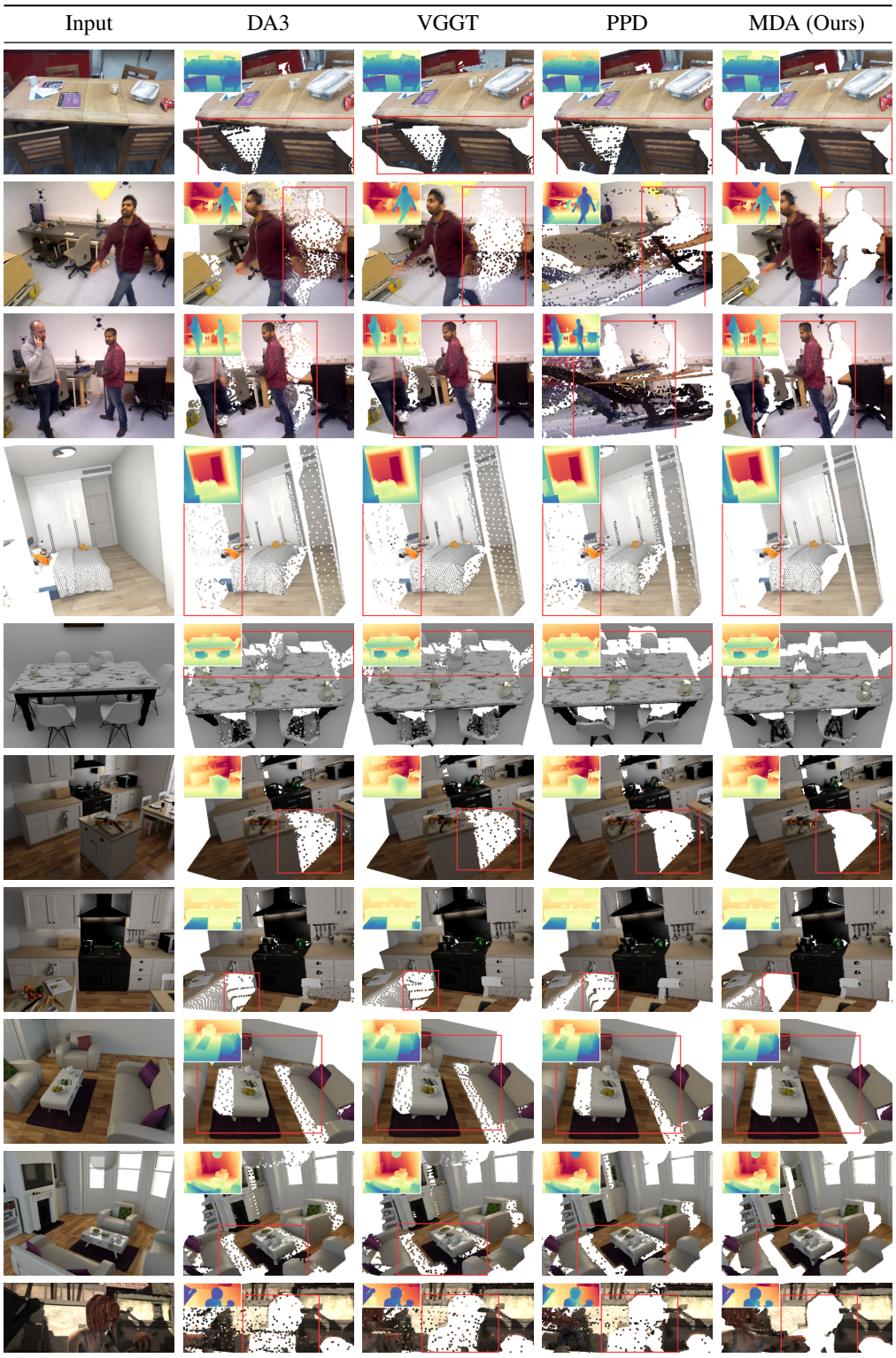


Figure 13: Additional qualitative boundary comparison across nine scenes drawn from 7Scenes, Bonn, HiRoom, NRGBD, and Sintel. Baseline methods (DA3, VGGT, PPD) leave visible flying points wherever depths differ sharply, while our approach keeps the boundary clean across all scenes.

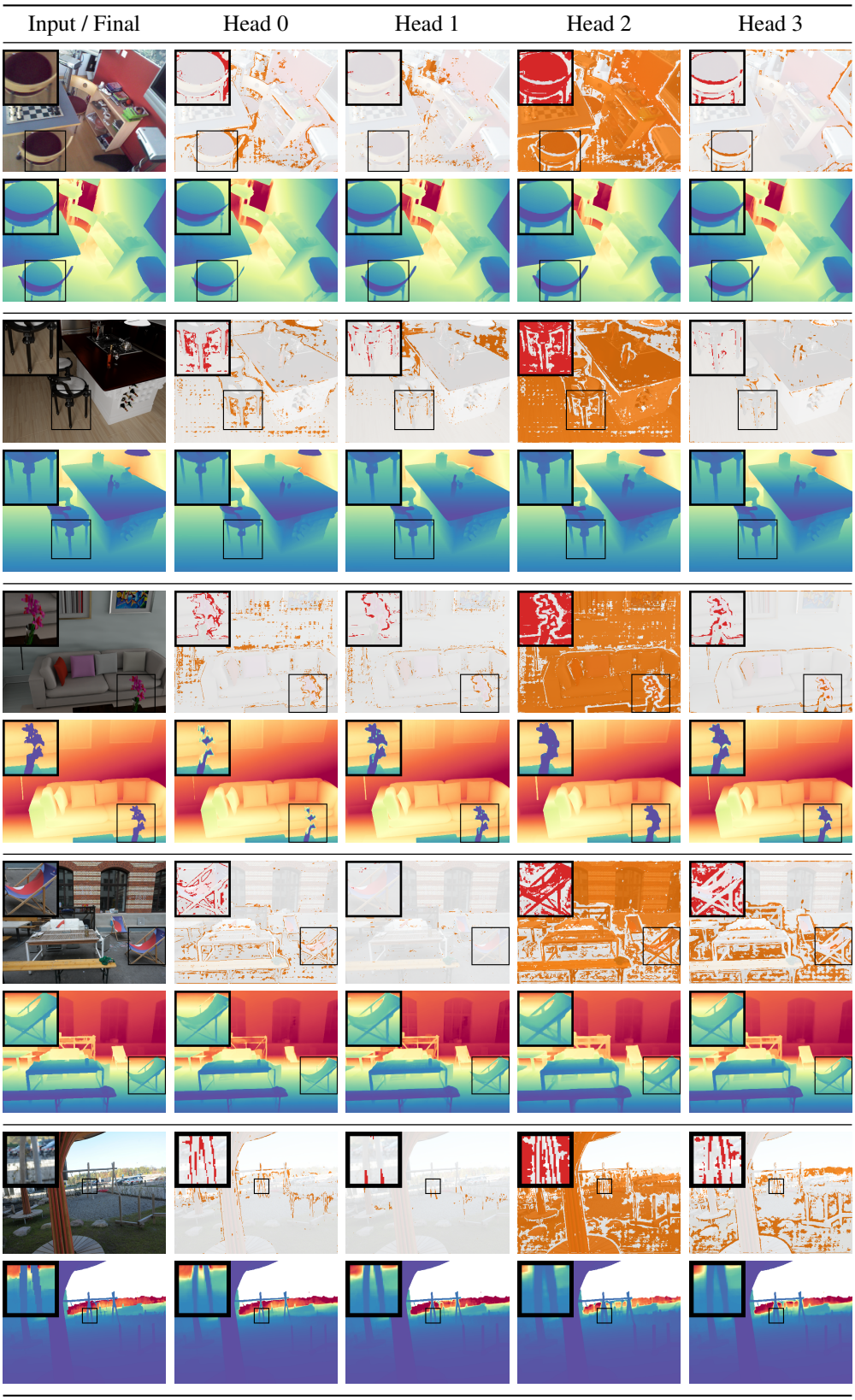


Figure 14: Per-component visualization of heads with $K{=}4$. For each scene, the leftmost column shows the input image (top) and our final fused depth (bottom); the four right columns show the per-pixel mixture weight π_k (top) and the corresponding component mean depth D_k (bottom). Components specialise spatially: at occlusion boundaries different heads lock onto the foreground and background surfaces, whereas in smooth regions one head carries almost all the weight while the others converge to similar depths.