SHED Light on Segmentation for Dense Prediction

Seung Hyun Lee
University of Michigan

Sangwoo Mo
Pohang University of Science and Technology (POSTECH)

Stella X. Yu*
University of Michigan

Reviewed on OpenReview: *https://openreview.net/forum?id=gsRlLeHiiI*

Abstract

Dense prediction infers per-pixel values from a single image and underlies 3D perception and robotics. Real-world scenes are highly structured, yet standard dense prediction models typically decode spatial outputs without explicitly tying reconstruction to a learned hierarchy of image regions. We propose SHED, an encoder–decoder that constructs an explicit hierarchy of image segments and reverses this hierarchy during decoding. The encoder groups image regions into progressively coarser segment tokens, and the decoder unpools coarse representations back to finer regions through the learned assignments rather than generic feature upsampling. The hierarchy is trained end-to-end from the dense prediction objective alone, without segmentation labels. SHED improves mean monocular-depth accuracy over a matched DPT baseline, with the largest relative gain under HyperSim-to-NYUv2 transfer without target scale or shift alignment. Beyond per-pixel accuracy, boundary, object-wise, and Chamfer evaluations show smaller mean differences in geometric structure. Representations learned under depth supervision also improve alignment with held-out PartNet part annotations. Finally, applying the reverse hierarchy to RAFT reduces end-point error under matched optical-flow training on FlyingChairs.

1 Introduction

Dense prediction tasks, such as monocular depth estimation and semantic segmentation, serve as the foundation for spatial reasoning in robotics (Liu et al., 2024). At their core, these tasks require an understanding of how 2D image projections relate to the underlying 3D structure of the world. Modern models based on Vision Transformers (ViTs) (Dosovitskiy, 2020) capture dependencies across image locations. However, standard grid-based decoding does not explicitly tie the reconstruction of dense outputs to a learned hierarchy of image regions. This suggests a region-aware reconstruction mechanism that connects coarse scene structure to local predictions, with the goal of preserving object boundaries and surface geometry (Figure 1, row 1).

Hierarchical accounts of human vision motivate this bidirectional design (Hochstein & Ahissar, 2002). Reverse hierarchy theory describes how a high-level scene interpretation guides access to finer visual details. Motivated by this perspective, we organize dense prediction through a hierarchy that groups fine-grained image regions into progressively coarser segments. This structural scaffold then guides the refinement of local details. Coarse segment features are propagated back to finer regions, allowing broader scene context to inform local depth estimates. This coarse-to-fine reconstruction aims to improve boundary alignment and within-region depth consistency (Figure 1, row 2).

To realize this idea, we propose **SHED**, **S**egment **H**ierarchy for **D**ense Prediction, a novel architecture that reconceptualizes dense prediction as a process of hierarchical grouping and reconstruction. Starting from

*Corresponding author.

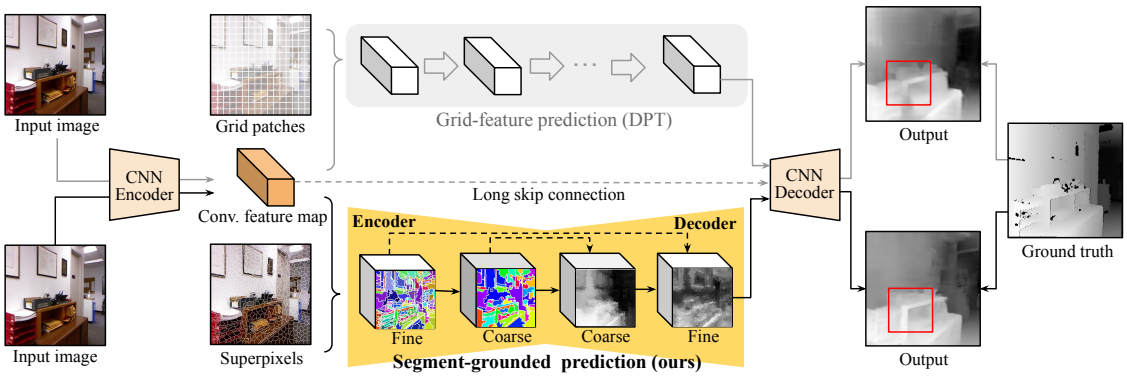


Figure 1: **Segment hierarchy for dense prediction (SHED).** Conventional methods such as DPT (Ranftl et al., 2021) combine contextualized grid features with convolutional decoding. SHED uses a hierarchy of segment tokens to guide dense prediction. In the illustrated segment-token configuration, the encoder pools superpixel tokens into coarser segment tokens, and the decoder propagates coarse segment features back to finer regions by reusing the learned pooling assignments, producing depth maps with structural coherence.

RGB-derived superpixels, SHED learns an explicit hierarchy of segments that adaptively decompose the scene. The hierarchy requires no segmentation annotations. Its grouping assignments are learned end-to-end, guided solely by the dense prediction objective. While encoder-side grouping builds on prior work (Ke et al., 2024), our central contribution is reverse structural decoding for dense prediction. The decoder propagates coarse segment information back to finer regions through the same learned assignments used to construct the hierarchy, directly tying reconstruction to the image-dependent grouping structure. Coarse regional context and fine-level features are then combined to produce dense outputs.

The technical core of SHED lies in its bidirectional flow. While the encoder iteratively groups fine regions into increasingly abstract segment tokens, the decoder reverses this flow by propagating features from coarse to fine regions. To produce a structured dense map, SHED's decoder unpools features from coarser segments to finer ones using probabilistic assignments computed during the encoding phase. This reverse hierarchy, which has no counterpart in encoder-only segmentation frameworks, enables coarse structural context to directly inform fine-grained predictions. By distributing segment-level features across associated regions, the model couples predictions within each region through shared features, while retaining fine-level information for local detail.

Our contributions are summarized as follows:

- **Reverse structural decoding.** We introduce a dense prediction architecture that couples a learned forward segment hierarchy with coarse-to-fine decoding through the same pooling assignments. The decoder transports coarse features to finer regions, combines them with encoder skips, and integrates the decoded hierarchy states through multi-level spatial readout.

- **Segment-consistent depth estimation.** SHED improves mean depth accuracy under NYUv2 (Nathan Silberman & Fergus, 2012) training and cross-domain transfer, with the clearest relative gains under HyperSim (Roberts et al., 2021)-to-NYUv2 transfer. Boundary, object-wise, and 3D reconstruction evaluations further characterize the geometric structure of its predictions.

- **Structure-aware representations.** Representations learned under depth supervision align with held-out PartNet (Mo et al., 2019) annotations and support image retrieval. Supervised semantic-segmentation fine-tuning further evaluates transfer beyond depth. Separately, applying the reverse hierarchy to RAFT (Teed & Deng, 2020) improves motion correspondence under matched optical-flow training.

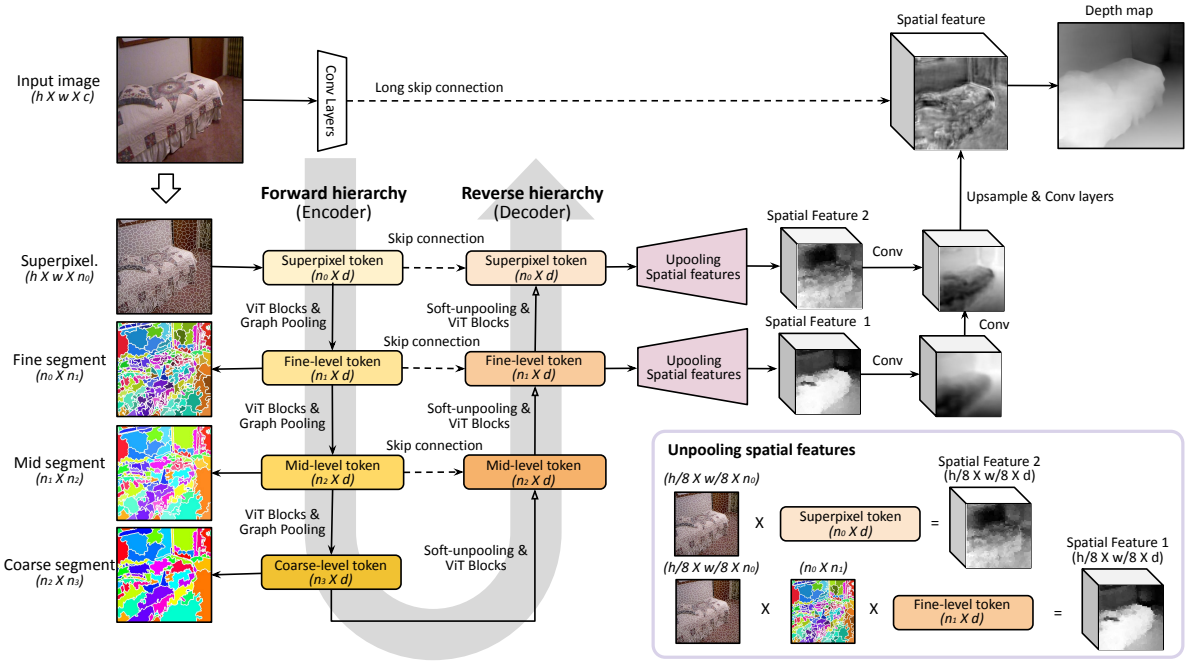


Figure 2: **SHED introduces a bidirectional segment hierarchy for dense prediction.** SHED decomposes dense prediction into two complementary processes: structural abstraction and structural inversion. **1)** The encoder constructs a forward hierarchy by grouping superpixel tokens into increasingly abstract segments. **2)** The decoder inverts this hierarchy by unpooling segment representations from coarse to fine, enabling global structural context to directly inform pixel-level predictions. Decoded segment tokens are projected into region-aligned 2D feature maps and fused across hierarchy levels. These features are then combined with early convolutional features to recover fine details and produce the final dense map. The spatial-unpooling inset illustrates two hierarchy levels; in the full model, the readout uses all four decoded hierarchy states aligned through the composed assignments.

2 Related Work

Dense prediction is a core problem in computer vision, aiming to assign pixel-level outputs across an image (Forsyth & Ponce, 2002). It includes tasks such as semantic segmentation (Long et al., 2015), depth estimation (Eigen et al., 2014), optical flow (Teed & Deng, 2020), and image editing (Isola et al., 2017). Modern approaches typically adopt encoder-decoder architectures, such as U-Net (Ronneberger et al., 2015) and DPT (Ranftl et al., 2021), trained using task-specific supervision. Large-scale systems such as Segment Anything (Ravi et al., 2025) and Depth Anything (Yang et al., 2024) have further advanced dense visual prediction. However, recent analyses suggest that strong benchmark performance does not necessarily imply structural consistency (El Banani et al., 2024; Man et al., 2024).

Monocular depth estimation is a representative dense prediction task that infers per-pixel depth from a single image. It is widely used in 3D reconstruction (Song et al., 2017), autonomous driving (Geiger et al., 2012), and robotic perception (Tateno et al., 2017). Structural cues in depth estimation have been extensively explored to enhance geometric coherence. Existing approaches can be broadly categorized into four types: **1)** Representation-based approaches modify how depth is encoded, such as by discretizing depth values (Fu et al., 2018; Bhat et al., 2021; Li et al., 2024) or modeling spatial dependencies (Liu et al., 2015; Cheng et al., 2018; Yuan et al., 2022). **2)** Regularization imposes geometric constraints through loss functions that promote smooth surfaces (Godard et al., 2017; Zhan et al., 2018; Bian et al., 2019), consistent normals (Yang et al., 2018), or planar regions (Yin et al., 2019; Watson et al., 2019). **3)** Multi-task learning jointly estimates depth with auxiliary signals, such as scene geometry (Eigen & Fergus, 2015; Yin & Shi, 2018) or semantics (Mousavian et al., 2016; Kendall et al., 2018; Chen et al., 2019; Guizilini et al., 2020; Zhu

et al., 2020). **4)** Post-processing refines predictions using off-the-shelf techniques (Krähenbühl & Koltun, 2011; Chen et al., 2016).

Several multi-task approaches have explored segmentation as an auxiliary signal to improve depth estimation. Early works used segmentation as an additional supervision signal (Mousavian et al., 2016; Kendall et al., 2018), while more recent ones leveraged segment regions or boundaries to guide depth discontinuities (Chen et al., 2019; Guizilini et al., 2020; Zhu et al., 2020). In contrast, SHED integrates learned segment structure directly into the model's internal representation, organizing dense prediction through a learned hierarchical decomposition of the image. Moreover, the segment hierarchy in SHED is learned through depth supervision, without additional segmentation annotations.

Perceptual grouping is a key mechanism in human vision that organizes low-level elements into coherent global structures (Wertheimer, 1938; Marr, 2010). This principle has inspired a broad range of computer vision research, including perception (Locatello et al., 2020; Mo et al., 2021; Kang et al., 2022; Deng et al., 2023; Ranasinghe et al., 2023), segmentation (Arbeláez et al., 2012; Hwang et al., 2019; Ke et al., 2022; Xu et al., 2022), and generation (Hong et al., 2018; Mo et al., 2018; He et al., 2022). In particular, CAST (Ke et al., 2024) recently applied it to ViTs for concurrent segmentation and recognition. However, most of these methods, including CAST, consider only a *forward hierarchy*, constructing representations and segmentations in a bottom-up manner. For dense prediction, an additional question is how the learned grouping assignments can guide the reconstruction of fine-grained spatial outputs. To address this, we adopt the complementary concept of a *reverse hierarchy* (Hochstein & Ahissar, 2002), where global structures guide and refine local parts through top-down feedback. While prior works (Anderson et al., 2018; Shi et al., 2023; Eftekhar et al., 2023) have explored top-down visual processing in high-level visual tasks, they do not address hierarchical reconstruction for dense prediction. Other studies (Eslami et al., 2016; Sajjadi et al., 2022; Seitzer et al., 2022) apply related ideas to encoder-decoder architectures, but focus on object-centric representations rather than explicit segment hierarchies for dense prediction. SHED differs from these approaches by using the same learned segment assignments for both forward hierarchical grouping and reverse coarse-to-fine reconstruction within a ViT-based dense prediction model.

3 SHED: Segment Hierarchy for Dense Prediction

We propose SHED, a dense prediction architecture that explicitly models a bidirectional segment hierarchy (see Figure 2). The encoder groups image regions into segment tokens and progressively pools them into coarser representations, while the decoder reverses this process by unpooling segment features from coarse to fine. By propagating segment-level features back toward pixel-level predictions, SHED uses coarse regional context to guide dense reconstruction. Unlike generic feature upsampling, the decoder transports coarse features back to finer regions through the same learned assignments that formed the forward hierarchy.

3.1 Encoder: Grouping segments via forward hierarchy

The encoder constructs a forward segment hierarchy that abstracts fine-grained regions into increasingly coarse structural units. This hierarchy serves as the structural backbone of SHED, linking fine and coarse image regions. While similar hierarchical grouping strategies have been explored for image-level recognition, we use the resulting hierarchy and its learned assignments for coarse-to-fine feature reconstruction in dense prediction. To construct the forward hierarchy, we adopt a segment-based tokenization strategy that groups image features into superpixel tokens and progressively clusters them based on feature similarity. This grouping mechanism follows prior work on hierarchical segmentation (Ke et al., 2024), and provides the intermediate segment features and learned assignments used by SHED's decoder for coarse-to-fine reconstruction.

Unlike encoder-only hierarchical segmentation models, the forward hierarchy in SHED serves as an intermediate structural representation that is subsequently inverted by the decoder for dense prediction.

Tokenization. Given an image $X \in \mathbb{R}^{h \times w \times c}$, the encoder produces hierarchical segmentations $S_0, S_1, \ldots$ and corresponding embeddings $Z_0, Z_1, \ldots$, ordered from fine to coarse. We first divide the image into n_0 superpixels using SEEDS (Van den Bergh et al., 2012), a fixed preprocessing step based only on RGB. We extract a convolutional feature map $F_{\text{conv}} \in \mathbb{R}^{(h_0 \cdot w_0) \times d}$ with spatial stride 8 ($h_0 = h/8$, $w_0 = w/8$). We resize

the superpixel label map to this feature resolution using nearest-neighbor interpolation and represent it as a one-hot assignment matrix $S_0 \in \mathbb{R}^{(h_0 \cdot w_0) \times n_0}$, which maps each feature-grid location to a superpixel. We then add fixed sinusoidal positional embeddings to F_{conv} and average-pool features within each superpixel to obtain initial embeddings $Z_0 \in \mathbb{R}^{n_0 \times d}$. The default 384×384 token schedule is $(n_0, n_1, n_2, n_3) = (576, 256, 128, 64)$, matching DPT's 24×24 token grid. For global context modeling, we prepend a learnable class token to form $\bar{Z}_0 \in \mathbb{R}^{(n_0+1) \times d}$ and feed it into the first ViT block.

Hierarchical clustering. Coarser segments are obtained by alternating ViT blocks and graph pooling (Ke et al., 2024). At each level l, given Z_{l-1} and S_{l-1} from the previous layer, we prepend the current class token to form $\bar{Z}_{l-1}$, apply ViT blocks, and obtain updated features, excluding the class token. Below, Z_{l-1} denotes these updated segment features. To form coarser tokens $Z_l \in \mathbb{R}^{n_l \times d}$, we first select n_l seed indices $c_{l,j}$ from Z_{l-1} via farthest point sampling (Qi et al., 2017). The corresponding seed features are $C_l[j] = Z_{l-1}[c_{l,j}]$. Let $Q_l = G_l(Z_{l-1})$ denote the features produced by the learned attention-based grouping transformation. We compute the soft assignment matrix $P_l \in \mathbb{R}^{n_{l-1} \times n_l}$ using a row-wise softmax over scaled cosine similarities:

$$P_l(i \to j) = \frac{\exp(\alpha \ \mathrm{sim}(Q_l[i], Q_l[c_{l,j}]))}{\sum_{k \in [n_l]} \exp(\alpha \ \mathrm{sim}(Q_l[i], Q_l[c_{l,k}]))}.$$

Here, $[n] := \{0, \ldots, n-1\}$. Rows correspond to fine segments and columns to coarse segments; each row sums to one.

We refine the seed features by aggregating transformed fine-level features, normalizing by the total assignment weight of each coarse segment, and adding a residual connection:

$$Z_l = C_l + g_{l,2}\big(\big(P_l^\top g_{l,1}(Z_{l-1})\big) \oslash \big(P_l^\top \mathbf{1}\big)\big).$$

Here, $g_{l,1}$ and $g_{l,2}$ are learned pointwise transformations before and after aggregation, respectively, and $\oslash$ denotes element-wise division with the denominator broadcast across channels.

To propagate segment assignments through the hierarchy, we compute coarser segmentations by composing the assignment matrices:

$$S_l = S_{l-1}\bar{P}_l, \quad l = 1, 2, \ldots, l_{\max},$$

where $\bar{P}_l$ is a hard assignment matrix obtained by taking the argmax over each row of P_l. The soft assignments P_l are retained for feature pooling and reverse unpooling.

3.2 Decoder: Predicting outputs via reverse hierarchy

The decoder reconstructs spatial feature maps by reversing the encoder's segment hierarchy, progressively propagating features from coarse to fine segment levels. This involves two steps: **1)** computing decoder features Z'_l by unpooling from Z'_{l+1} and fusing them with encoder features Z_l via skip connections; and **2)** projecting the decoded hierarchy features into the spatial domain and fusing information from all hierarchy levels to produce the final dense prediction.

Unpooling segment tokens. Formally, the reverse hierarchy distributes coarse segment features to finer segments through the learned encoder assignments. We initialize the coarsest decoder feature as $Z'_{l_{\max}} = Z_{l_{\max}}$. At each level $l = l_{\max} - 1, \ldots, 0$, we reuse the encoder assignment $P_{l+1} \in \mathbb{R}^{n_l \times n_{l+1}}$ to transport coarse features to the corresponding fine segments:

$$U_l = P_{l+1} Z'_{l+1} \in \mathbb{R}^{n_l \times d}.$$

As each row of P_{l+1} sums to one, the unpooled feature for each fine segment is a weighted combination of coarse-segment features. We then fuse the transported features with the encoder features at the same level and refine them using MLP and ViT blocks:

$$Z'_l = \mathrm{ViT}_l\left(\mathrm{MLP}_l\left(U_l + Z_l\right)\right).$$

The encoder skip supplies fine-level information, while the transported features provide context from coarser segments. The MLP acts independently on each segment token, whereas the ViT blocks jointly update the segment tokens and the class token, which is omitted from the equation for clarity.

Spatial readout. We convert the decoded segment tokens into spatial feature maps by composing the hierarchy assignments:

$$P_{0\to l} = P_1 P_2 \cdots P_l \in \mathbb{R}^{n_0 \times n_l}.$$

The decoded state at each hierarchy level is first aligned with the finest segment space:

$$\widetilde{Z}'_0 = Z'_0, \qquad \widetilde{Z}'_l = P_{0\to l} Z'_l, \quad l = 1, \ldots, l_{\max}.$$

Accordingly, the spatial feature map contributed by level l is

$$F_l = S_0 \widetilde{Z}'_l = S_0 P_{0\to l} Z'_l, \qquad F_l \in \mathbb{R}^{(h_0 w_0) \times d},$$

where $P_{0\to 0} = I$. Thus, every decoded hierarchy level contributes a spatial feature map through the same assignments that formed the encoder hierarchy.

To combine these multi-level features, we concatenate all aligned hierarchy states in the finest segment space and project them back to the decoder feature dimension:

$$Z_{\text{hier}} = Z'_0 + \text{Proj}\left(\text{Concat}\left[\widetilde{Z}'_0, \widetilde{Z}'_1, \ldots, \widetilde{Z}'_{l_{\max}}\right]\right).$$

The fused hierarchy representation is then projected to the spatial domain using the initial superpixel assignment:

$$F_{\text{hier}} = S_0 Z_{\text{hier}}.$$

This operation fuses the decoded hierarchy states in the common finest-segment space before projecting the resulting representation to the spatial domain. The resulting feature map is combined with the stride-8 and stride-4 convolutional features and refined through convolution and upsampling to produce the final dense prediction.

A common design in dense prediction is to build a spatial hierarchy by assembling feature maps from multiple representation levels. DPT (Ranftl et al., 2021) is a representative example of this approach. It reassembles features from different ViT layers into regular-grid feature maps at multiple spatial resolutions and progressively fuses them through a convolutional decoder.

In SHED, the hierarchy is defined over learned segment regions. The number of segment tokens progressively decreases as the encoder forms coarser regions, and the same learned assignments guide coarse-to-fine feature transport during decoding. The decoded hierarchy states are projected onto a common $h_0 \times w_0$ feature grid at stride 8 and fused across abstraction levels. The subsequent convolutional decoder incorporates stride-4 features and upsamples the prediction to the input resolution.

4 Experiments

We evaluate SHED along three dimensions. First, we assess depth prediction and cross-domain transfer through per-pixel accuracy, occlusion-boundary quality, object-wise depth accuracy, and 3D reconstruction. Second, we analyze the structure captured by representations learned under depth supervision through learned region hierarchies, part-level alignment, and image retrieval. Third, we evaluate SHED on additional dense prediction tasks through supervised semantic-segmentation fine-tuning and matched optical-flow training. Controlled ablations further examine tokenization, learned grouping, reverse reconstruction, and multi-level spatial readout. We also report computational cost.

4.1 Setup

Following the hybrid CNN–Transformer design of DPT-Hybrid (Ranftl et al., 2021), we use a ResNet-50/ViT-Small baseline for controlled comparisons with SHED. Our configuration uses the stem and first two residual stages of ResNet-50 (He et al., 2016), a 1×1 feature projection, and a 12-block, 384-dimensional ViT-Small trunk (Dosovitskiy, 2020). We refer to this configuration as DPT throughout the paper. In the primary NYUv2 comparison, DPT and SHED share the supervised ImageNet-1K initialization for corresponding backbone and trunk weights, training data, optimization, and evaluation protocol.

For in-domain depth evaluation, we train and evaluate on NYUv2 (Nathan Silberman & Fergus, 2012), a standard benchmark for indoor depth estimation. For cross-domain evaluation, we train both DPT and SHED on the synthetic HyperSim dataset (Roberts et al., 2021) and assess their *synthetic-to-real transfer* to the real-world NYUv2 dataset. Throughout the paper, DPT is our primary *controlled* baseline. This comparison evaluates the overall model design; decoder alternatives and multi-level spatial readout are examined further in the controlled ablations. Released relative and metric-HyperSim checkpoints of Depth Anything v2 Small (Yang et al., 2024) are included as contextual references because their architecture and pretraining differ from those of the controlled DPT–SHED comparison.

Tokenization. We generate superpixels from RGB images using the SEEDS algorithm (Van den Bergh et al., 2012). SEEDS is fixed and non-differentiable and uses neither semantic nor instance annotations. The primary NYUv2 comparison uses 480×640 inputs with a segment-token schedule of $1200 \to 512 \to 256 \to 128$, matching DPT's 30×40 grid at the finest level. The decoder/readout controls and computational measurements use 384×384 inputs with $576 \to 256 \to 128 \to 64$ tokens, matching a 24×24 grid. Features are extracted from intermediate ResNet-50 blocks at 1/4 and 1/8 of the input resolution; both are passed to the final decoder via skip connections. In the interleaved segment-token configuration, SHED initializes segment tokens by pooling the stride-8 convolutional features within superpixels before ViT processing. The primary NYUv2 comparison and the decoder/readout controls use a variant that instead initializes the segment hierarchy by pooling the output features of a 12-block grid-token ViT within the superpixels.

Architecture. The interleaved segment-token configuration described in Section 3 is used for the HyperSim-trained SHED model. The encoder consists of three stages, each with two ViT blocks followed by graph pooling, progressively reducing the number of segment tokens to 256, 128, and 64 at 384×384 input resolution. The decoder mirrors this with unpooling and receives skip connections from the corresponding encoder stages. Each decoder stage contains two ViT blocks, giving six encoder blocks and six decoder blocks in total. We use this token-count schedule as a fixed multiscale scaffold. The pooling assignments and the content routed into each segment remain learned and image-dependent. At each reverse stage, the learned pooling assignment transports coarse features back to the corresponding finer segments. These features are added to the encoder skip and processed by an MLP followed by the stage's ViT blocks. The four hierarchy states are combined by the multi-level spatial readout described in Section 3.

The primary NYUv2 comparison and the decoder/readout controls use a grid-token configuration. A 12-block ViT first processes the CNN-derived grid tokens. Its output features are pooled within SEEDS superpixels to initialize a segment hierarchy with three forward transitions. A shared pooling module constructs the hierarchy, and a shared reverse-refinement module propagates coarse features back through the learned assignments. The primary NYUv2 configuration uses all four hierarchy states for spatial readout. The decoder/readout controls additionally retain a shared DPT-style spatial pyramid to evaluate reconstruction while preserving progressive spatial upsampling.

Metric-depth output. For the primary NYUv2 comparison, DPT predicts $\widehat{d} = 10\,\mathrm{sigmoid}(z)$ and SHED predicts $\widehat{d} = 10\,\mathrm{sigmoid}(z + b_{\mathrm{cls}})$, with depth expressed in meters. Here z is the spatial output logit, and b_{cls} is a learned, image-dependent scalar bias obtained by averaging the four CLS-conditioned level logits used in the hierarchy readout. Neither model uses target-fitted scale or shift at inference.

For the primary NYUv2 comparison in Table 1, DPT and the SHED configuration with a grid-token encoder are trained for 40,000 optimizer steps at 480×640 resolution with a batch size of 16. Both use AdamW with weight decay 0.02 and a one-cycle learning-rate schedule. The maximum learning rates are 2×10^{-5} for pretrained parameters and 2×10^{-4} for newly initialized parameters. Both models use the same training images and augmentations, with a scale-invariant logarithmic loss (SILog, $\lambda = 0.85$) computed against ground-truth depth.

We evaluate the final-step checkpoint from each run on the official 654-image NYUv2 test split using the same crop and valid-depth range. Results are reported as mean $\pm$ sample standard deviation over training seeds $0, 1, 2$, with paired-seed confidence intervals reported for the primary comparison. The 384×384 control experiments restore predictions to 480×640 by bilinear interpolation before evaluation. For synthetic-to-real transfer, the HyperSim-trained DPT and SHED models are evaluated on NYUv2 without test-time scale, shift, or affine alignment. Their results are reported over training seeds $0, 1, 2$.

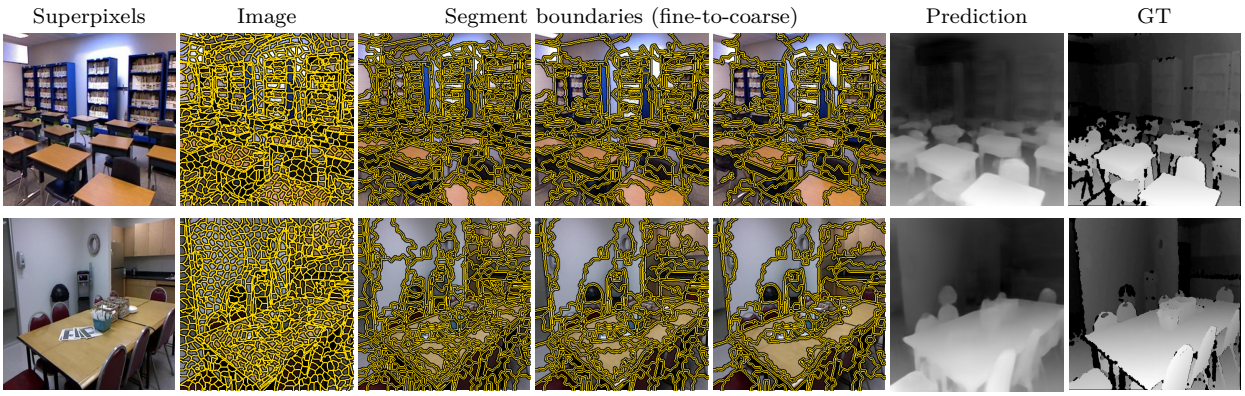


Figure 3: **SHED produces structurally coherent depth predictions aligned with scene layout.** We visualize the learned fine-to-coarse segment hierarchy alongside the final depth prediction and ground-truth (GT) depth on NYUv2 test examples. Superpixels and learned segment boundaries are shown as yellow overlays on the same input image at every hierarchy level, allowing the fine-to-coarse grouping to be compared directly. In the first example, the learned regions delineate classroom desks, with the corresponding depth prediction separating them from the background. In the second, SHED partitions the table into multiple regions while preserving a smooth depth variation toward the back.

4.2 Segment-consistent depth estimation

We evaluate the accuracy and geometric structure of SHED's depth predictions. We first visualize the learned region hierarchies together with their corresponding depth maps. We then report per-pixel accuracy and cross-domain transfer, alongside contextual comparisons with released depth foundation models. Finally, we assess structural quality through occlusion-boundary localization, object-wise depth accuracy, and 3D scene reconstruction.

Qualitative depth and region structure. Figure 3 shows the learned segment hierarchies together with their corresponding depth predictions. The examples illustrate region boundaries around classroom desks and the partitioning of a tabletop into multiple regions. We display the superpixels and all subsequent hierarchy levels using the same boundary-overlay style on the input image, facilitating comparison across levels. The examples illustrate region boundaries around classroom desks and the partitioning of a tabletop into multiple regions, alongside the corresponding depth predictions.

Per-pixel metrics & generalization. Table 1 reports standard per-pixel depth metrics under both in-domain and cross-domain settings. Under matched NYUv2 training, mean results favor SHED on all reported metrics. The paired 95% confidence interval over training-seed differences is above zero for δ_1. The NYUv2 evaluation split was also used for configuration selection; this interval therefore quantifies variation across training seeds conditional on that selection. Under HyperSim-to-NYUv2 transfer, SHED improves all reported mean metrics over the HyperSim-trained DPT baseline. Both models are trained only on HyperSim and evaluated without test-time scale, shift, or affine alignment.

Table 1 also reports NYUv2-to-SUN-RGBD transfer using the same NYUv2-trained checkpoints without target fine-tuning or calibration. Across the transfer settings, the largest relative gains occur in AbsRel and RMSE for HyperSim-to-NYUv2 transfer.

Comparison with foundation models. Table 3 places our primary NYUv2 depth results alongside the released Depth Anything v2 ViT-S model (Yang et al., 2024). DPT and SHED use matched supervised ImageNet-1K initialization and NYUv2 training, and are evaluated without test-time scale or shift alignment. Their results are reproduced from Table 1.

In block A, we evaluate the released relative-depth checkpoint after fitting a least-squares scale and shift in depth space separately for each image using valid NYUv2 ground-truth pixels. In block B, we evaluate the released metric-HyperSim checkpoint without target-derived calibration. SHED achieves better mean depth

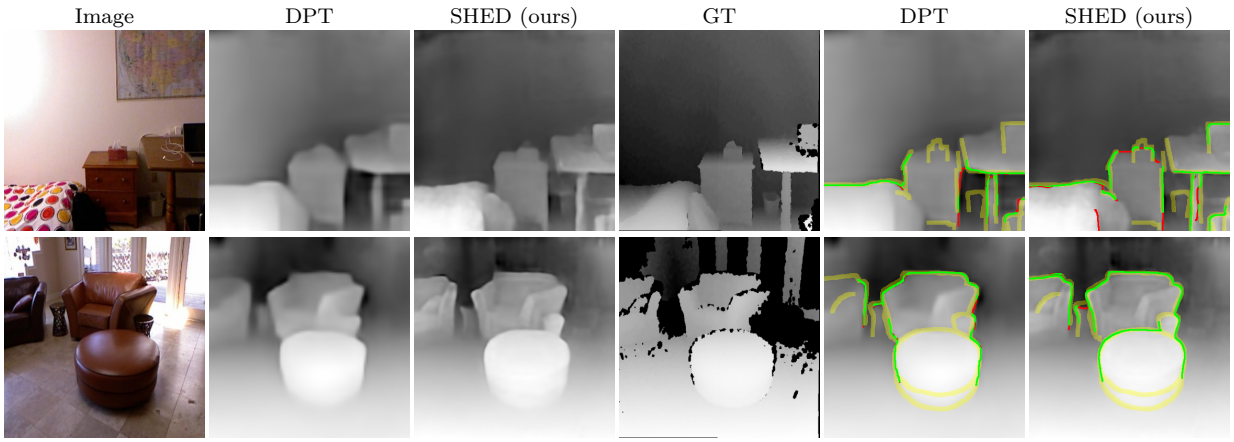


Figure 4: **SHED shows sharper object contours and clearer occlusion boundaries in these examples.** We compare depth maps (cols. 2–4) and occlusion boundaries (cols. 5–6) from DPT and SHED on the NYUv2-OC++ dataset. Boundaries are extracted using a Canny edge detector and compared with the reference annotations. Ground-truth edges are shown in yellow, matched predicted edges in green, and unmatched predicted edges in red. These examples highlight clearer separation around furniture in the bedroom (row 1) and around the armchair and ottoman (row 2).

Table 1: **SHED improves mean depth accuracy across in-domain and cross-domain evaluations.** We evaluate on the 654-image NYUv2 test split and the 5,050-image SUN-RGBD test split. DPT and SHED are compared under matched training and evaluation protocols within each block. The NYUv2-trained SHED uses the grid-token configuration; the HyperSim-trained model uses the interleaved segment-token configuration. All predictions are evaluated without target-derived scale or shift alignment. The NYUv2 evaluation split was used for configuration selection. SUN-RGBD evaluation uses the same NYUv2 checkpoints without target fine-tuning or calibration. Results are mean ± sample standard deviation over training seeds $0, 1, 2$. Bold denotes the better displayed mean within each block.

		Depth accuracy			Depth error		
Method	Train	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	AbsRel↓	RMSE↓	$\log_{10} \downarrow$
Trained on NYUv2, tested on NYUv2							
DPT	NYUv2	0.8644±0.0005	0.9785±0.0008	0.9949±0.0004	0.1205±0.0007	0.4144±0.0031	0.0510±0.0004
SHED (ours)	NYUv2	**0.8687±0.0008**	**0.9799±0.0012**	**0.9951±0.0004**	**0.1189±0.0012**	**0.4088±0.0018**	**0.0501±0.0003**
Trained on HyperSim, tested on NYUv2							
DPT	HyperSim	0.4261±0.0088	0.7810±0.0046	0.9354±0.0046	0.3617±0.0092	1.0381±0.0054	0.1303±0.0020
SHED (ours)	HyperSim	**0.4764±0.0170**	**0.8033±0.0115**	**0.9356±0.0035**	**0.3272±0.0164**	**0.9501±0.0264**	**0.1231±0.0032**
Trained on NYUv2, tested on SUN-RGBD							
DPT	NYUv2	0.7608±0.0016	0.9449±0.0001	**0.9858±0.0003**	0.1677±0.0010	0.4932±0.0020	0.0698±0.0002
SHED (ours)	NYUv2	**0.7637±0.0012**	**0.9461±0.0011**	**0.9858±0.0005**	**0.1670±0.0018**	**0.4887±0.0021**	**0.0693±0.0003**

metrics than DPT within each matched comparison. Comparisons with the released Depth Anything v2 checkpoints additionally reflect differences in architecture, pretraining, supervision, and calibration protocol.

Boundary accuracy. We assess the structural quality of SHED by comparing occlusion boundaries against DPT on the NYUv2-OC++ dataset (Ramamonjisoa et al., 2020). Figure 4 shows predicted depth maps and their occlusion boundaries, extracted using a Canny edge detector (Canny, 1986).

For quantitative boundary evaluation, we use the matched 384×384 configuration and fixed normalized Canny thresholds of 0.15/0.30. We report ε_a, the mean localization error of predicted edges satisfying the strict 10-pixel eligibility rule, and ε_c, the symmetric 10-pixel-clipped Chamfer error normalized by the combined edge count. The complete evaluation protocol is specified in Section A.

Table 2: **SHED achieves lower mean boundary errors and better mean object-wise depth metrics.** Boundary evaluation uses the matched 384×384 configuration; predictions are bilinearly restored to the native 480×640 evaluation grid before computing ε_a and ε_c with fixed Canny thresholds of 0.15/0.30. Object-wise depth evaluation uses the primary 480×640 NYUv2 checkpoints and computes instance-macro metrics using the original instance annotations. DPT and SHED use matched training and evaluation protocols within each evaluation block. Results are mean ± sample standard deviation over training seeds $0, 1, 2$. Bold denotes the better displayed mean.

	Boundary		Object-wise depth		
Method	$\varepsilon_a \downarrow$	$\varepsilon_c \downarrow$	$\delta_1 \uparrow$	RMSE↓	$\log_{10} \downarrow$
DPT	5.5148±0.0425	9.5716±0.0073	0.8487±0.0025	0.4471±0.0044	0.0540±0.0005
SHED (ours)	**5.3988±0.0782**	**9.5466±0.0195**	**0.8506±0.0029**	**0.4451±0.0036**	**0.0537±0.0004**

Table 3: **Comparison with matched DPT baselines and released Depth Anything v2 references.** All results use the 654-image NYUv2 test split. DPT and SHED results are mean ± sample standard deviation over three training seeds; DAv2-S uses fixed released weights. Bold denotes the better displayed mean within each matched DPT–SHED comparison.

	Depth accuracy			Depth error		
Method	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	AbsRel ↓	RMSE ↓	$\log_{10} \downarrow$
A. In-domain comparison with a target-aligned relative-depth reference						
DPT	0.8644±0.0005	0.9785±0.0008	0.9949±0.0004	0.1205±0.0007	0.4144±0.0031	0.0510±0.0004
SHED (ours)	**0.8687±0.0008**	**0.9799±0.0012**	**0.9951±0.0004**	**0.1189±0.0012**	**0.4088±0.0018**	**0.0501±0.0003**
DAv2-S (relative)	0.8747	0.9680	0.9907	0.1131	0.4417	0.0472
B. Synthetic-to-real comparison without target calibration						
DPT	0.4261±0.0088	0.7810±0.0046	0.9354±0.0046	0.3617±0.0092	1.0381±0.0054	0.1303±0.0020
SHED (ours)	**0.4764±0.0170**	**0.8033±0.0115**	**0.9356±0.0035**	**0.3272±0.0164**	**0.9501±0.0264**	**0.1231±0.0032**
DAv2-S (metric-HyperSim)	0.3991	0.8747	0.9763	0.3285	0.9196	0.1192

At the primary Canny setting, SHED has lower mean ε_a and lower mean ε_c. Table 10 shows the same mean ordering at three fixed thresholds. Predicted-edge density is slightly higher for SHED, so the mean error reductions are not explained by predicting fewer edges. The paired 95% confidence intervals over training-seed differences include zero for both boundary metrics. We therefore report a repeated mean ordering, not a statistically established boundary improvement.

Object-wise depth accuracy. Beyond boundary localization, we evaluate depth accuracy within annotated object regions on NYUv2 (Nathan Silberman & Fergus, 2012), using ground-truth object masks as structural references. As shown in Table 2, SHED achieves higher mean δ_1 and lower mean RMSE and $\log_{10}$ error than DPT. The paired 95% confidence intervals over the three training seeds include zero for all three metric differences. The within-object gains are modest and vary by error measure.

3D scene reconstruction. We evaluate the geometric structure of predicted depth by reconstructing 3D point clouds on NYUv2 (Nathan Silberman & Fergus, 2012). Predicted and ground-truth depths are back-projected using NYUv2 camera intrinsics. DPT and SHED use the same intrinsics and valid-pixel masks. Depth values are expressed in meters, and evaluation is performed without test-time scale or pose alignment.

Figure 5 shows frontal and bird's-eye views of point clouds reconstructed from the same RGB image. In this example, SHED produces straighter wall contours than DPT, illustrating how differences in predicted depth affect the reconstructed surface geometry.

We quantify reconstruction performance with the Chamfer distance (Fan et al., 2017) in both directions. Each direction measures the mean Euclidean distance from points in one cloud to their nearest neighbors in the other. As shown in Table 4, SHED has lower mean Chamfer distances in both directions, corresponding to closer aggregate agreement between the predicted and ground-truth surfaces. Figure 6 illustrates how



Figure 5: **SHED produces structured 3D reconstructions.** We visualize 3D point clouds reconstructed from single-view depth maps, using predictions from DPT and SHED, together with ground-truth depth, for an NYUv2 example. Frontal views (cols. 2–4) and bird's-eye views (cols. 5–7) show the reconstructed scene from different viewpoints. In this example, SHED produces flatter wall geometry than DPT, as seen most clearly in the bird's-eye views.

Table 4: **SHED achieves lower mean Chamfer distance in both directions.** We evaluate point clouds reconstructed from predicted and ground-truth depth on the 654-image NYUv2 test split. Both directed distances are reported in meters, without test-time scale or pose alignment. Results are mean ± sample standard deviation over training seeds $0, 1, 2$. Bold denotes the lower displayed mean in each direction.

Method	Pred.→GT ↓	GT→Pred. ↓
DPT	0.1738±0.0015	0.1912±0.0010
SHED (ours)	0.1726±0.0026	0.1892±0.0047

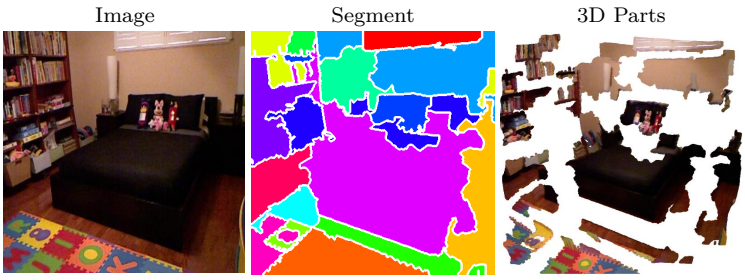


Figure 6: **SHED lifts learned segments into structured 3D regions.** We show the input image, learned segment partition, and corresponding 3D region decomposition obtained using SHED-predicted depth. segmentation or part annotations. The segments are learned with depth supervision, without semantic

learned image regions can be lifted with predicted depth into 3D region decompositions. The model learns these regions through depth supervision, without semantic segmentation or part annotations. The depth and region predictions thus provide complementary descriptions of scene geometry.

4.3 Structure-aware representations

We examine the structure captured by SHED representations learned under depth supervision through region-hierarchy visualization, part-level alignment, and image retrieval. These analyses assess whether the learned representations encode scene and object structure beyond per-pixel depth accuracy.

Learned region hierarchies. We analyze the segment hierarchy learned by SHED by comparing it to CAST, an encoder trained for image recognition using segment-based representations. Specifically, we use CAST, trained on ImageNet (Deng et al., 2009) with the self-supervised MoCo-v3 objective (Chen et al., 2021). For this visualization, we configure SHED to produce 64, 32, and 16 segments from 196 initial superpixels at 224×224, matching the hierarchy sizes of CAST.

Figure 7 shows the resulting partitions at three levels of granularity. In these examples, several boundaries of the depth-trained SHED hierarchy follow object and surface outlines visible in the RGB and depth references. Both methods progressively form coarser region groupings. The illustrated region structure emerges during depth training without region or part annotations.

Part-level structural alignment on PartNet. We evaluate whether depth-trained representations align with held-out 3D part annotations from PartNet (Mo et al., 2019). We use a custom-rendered subset of the official test split containing 807 shapes from 17 categories and 6,456 fixed views. Four shapes have no valid rendered part annotations in any view, leaving 6,424 views for scoring. PartNet annotations are used only for evaluation, never for training, clustering, checkpoint selection, or hyperparameter tuning.

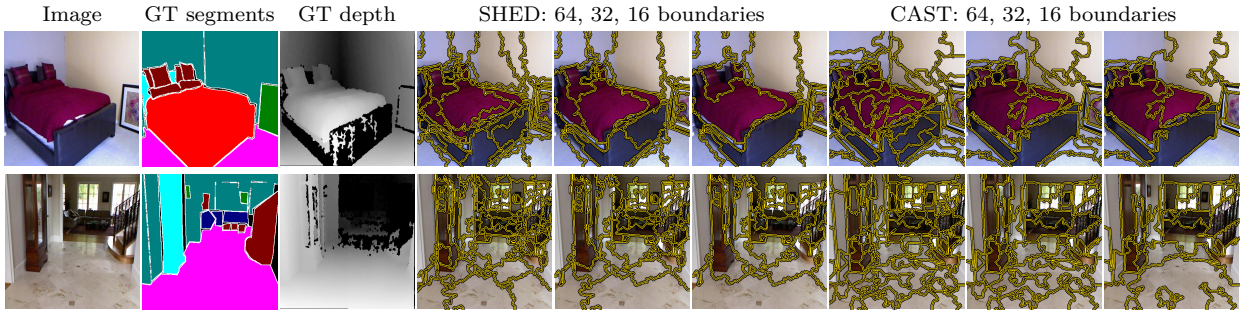


Figure 7: **SHED learns depth-aware segment hierarchies.** We compare segmentations from SHED and CAST (Ke et al., 2024) at the same hierarchy levels: 64, 32, and 16 segments. Both methods are visualized at 224×224 resolution with 196 initial superpixels; levels are ordered from fine to coarse within each method. All predicted partitions are shown as yellow boundary overlays on the same input image. Ground-truth segments and depth are included for reference. In these examples, SHED boundaries follow the bed and its components (row 1) and form progressively coarser groupings across the floor (row 2). SHED is trained with depth supervision; the CAST reference is trained on ImageNet using the self-supervised MoCo-v3 objective.

Table 5: **SHED achieves higher mean part-level structural alignment than DPT on PartNet.** We score 6,424 nonempty views from 6,456 renderings of 807 test shapes across 17 categories, using part annotations only for evaluation. Metrics are averaged over views within each shape, shapes within each category, and then equally over categories. Coarse and fine ARI use 17 categories; the other metrics use nine categories with middle-level annotations. Hier. denotes aggregation over coarse, middle, and fine levels; Boundary F is also aggregated over these levels. DPT and SHED report mean ± sample standard deviation over training seeds $0, 1, 2$. SEEDS and CAST-S are fixed single-run references. Bold denotes the highest displayed mean in each column.

Method	Hier. ARI ↑	Coarse ARI ↑	Middle ARI ↑	Fine ARI ↑	Hier. oracle mIoU ↑	Boundary F ↑
DPT	0.1183 ± 0.0026	0.1195 ± 0.0019	0.1409 ± 0.0037	0.0825 ± 0.0028	0.3040 ± 0.0032	0.1518 ± 0.0004
SEEDS	0.1305	0.1335	0.1508	0.1159	0.2799	**0.2408**
CAST-S	0.1512	0.1667	**0.2012**	0.0817	**0.3637**	0.2269
SHED (ours)	$\mathbf{0.1640 \pm 0.0044}$	$\mathbf{0.1781 \pm 0.0025}$	0.1925 ± 0.0048	$\mathbf{0.1300 \pm 0.0035}$	0.3557 ± 0.0128	0.2158 ± 0.0030

We render eight fixed 384×384 views per shape at 45° azimuth intervals, 20° elevation, and camera distance 3. We use the same frozen NYUv2-trained DPT and SHED checkpoints as in the primary depth evaluation. For DPT, we cluster final-block patch tokens; for SHED, we cluster fine-level features after reverse reconstruction. For both models, features are ℓ_2-normalized before deterministic spherical k-means at $K \in \{16, 32, 64\}$. All methods use the same renderings, camera parameters, and valid-point masks. Rendered ground-truth depth lifts the resulting regions into 3D, isolating representation alignment from depth-prediction error. Quantitative results are reported in Table 5.

SHED improves all reported mean metrics over DPT. Among the evaluated methods, SHED also achieves the highest mean hierarchical, coarse, and fine ARI. CAST-S leads on middle-level ARI and hierarchical oracle mIoU, and SEEDS leads on boundary F-score. Fixed qualitative examples are shown in Figure 8. These results indicate that part-level structure is reflected in features learned under depth supervision, without PartNet supervision.

Image retrieval. We additionally evaluate representations learned under depth supervision through image retrieval using mean-pooled encoder patch tokens for both DPT and SHED. The evaluation protocol and results are reported in Section C. SHED achieves higher mean retrieval scores with this shared readout. Both models use the same readout without a model-specific retrieval head.

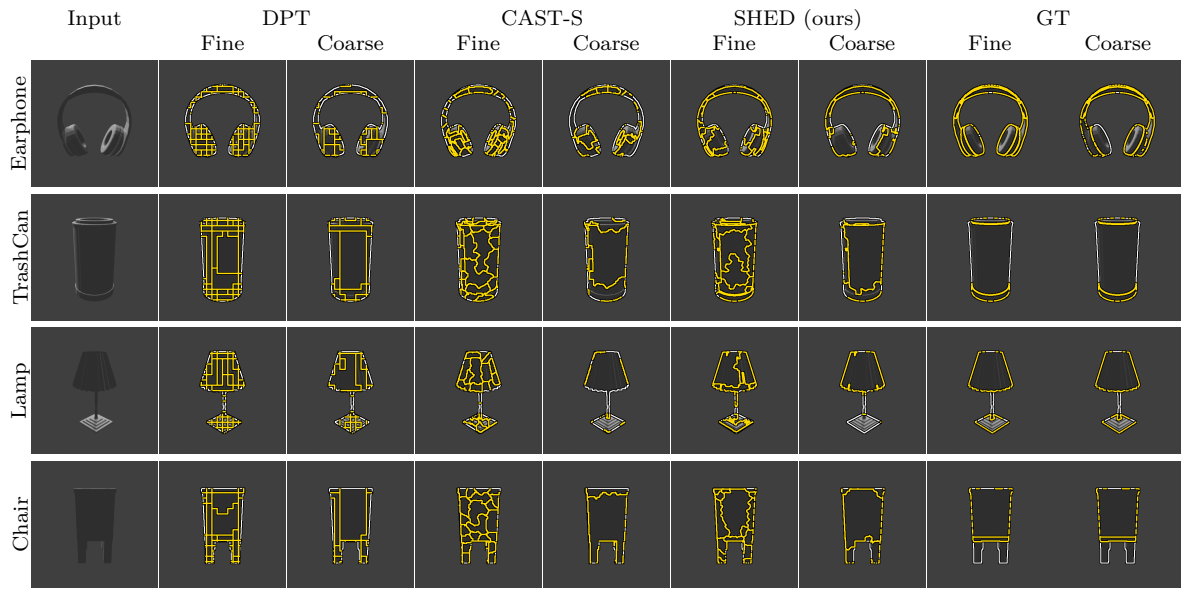


Figure 8: **SHED captures visible part structure without PartNet supervision.** We compare DPT, CAST-S, and SHED using the same rendered views, with the corresponding PartNet annotations shown at the right. Fine and coarse partitions are displayed as yellow boundary overlays on the input image for every method. PartNet annotations are used only for evaluation. White contours show the common rendered-object silhouette; yellow contours show each method's fine and coarse partition boundaries.

Table 6: **SHED achieves higher mIoU and boundary F-score than CAST on ADE20K.** We fine-tune SHED on the ADE20K training set using semantic labels for 240 epochs with a batch size of 16 and evaluate on the official validation split. ViT-Small and CAST values are reported reference results from Ke et al. (2024). Bold denotes the highest reported value in each column.

Method	mIoU ↑	Boundary F-score ↑
ViT-Small	41.7	–
CAST	43.1	36.5
SHED (ours)	**44.5**	**37.7**

4.4 Additional dense prediction tasks

Semantic segmentation. We evaluate supervised transfer to ADE20K by fine-tuning SHED on the training set using semantic labels for 240 epochs with batch size 16. Following the CAST evaluation protocol, we report region-level mIoU and boundary F-score on the official validation split in Table 6. SHED achieves higher reported mIoU than the ViT-Small and CAST references, together with a higher reported boundary F-score than CAST. The gains over CAST extend to both region overlap and boundary alignment.

Optical flow. We further evaluate assignment-tied reverse-hierarchy features for optical flow with RAFT (Teed & Deng, 2020). Both variants use the same ResNet-50/12-block Transformer backbone initialized from the same supervised ImageNet-1K DPT-12E checkpoint. They also instantiate and execute the same online SEEDS, hierarchy, and reverse-processing modules. Neither variant uses a depth-trained checkpoint, depth predictions, a teacher, or pseudo-labels.

We denote the matched grid-feature control as RAFT-DPT. It supplies the final block-12 Transformer grid to RAFT's correlation module. Although the control executes the hierarchy branch for compute matching, its output selector retains the DPT grid. RAFT-SHED instead supplies the grid refined by the assignment-tied reverse-hierarchy residual. The comparison therefore isolates whether reverse-hierarchy information is included in the correlation features.

Table 7: **RAFT-SHED achieves lower optical-flow EPE in all three training seeds.** We compare RAFT using grid features with RAFT using SHED reverse-hierarchy features. Both variants share the backbone, initialization, training data, and optimization recipe, and are trained on FlyingChairs for 5,000 steps with a batch size of 8. We evaluate the final checkpoints on 640 validation image pairs using 24 RAFT update iterations. We report end-point error (EPE) in pixels for each training seed and the mean ± sample standard deviation across seeds. Bold denotes the lower displayed EPE in each column.

Method	Seed 0	Seed 1	Seed 2	Mean ± SD
RAFT-DPT (grid features)	1.8840	1.7864	1.7934	1.8213±0.0545
RAFT-SHED (ours)	**1.7805**	**1.7540**	**1.7530**	**1.7625±0.0156**

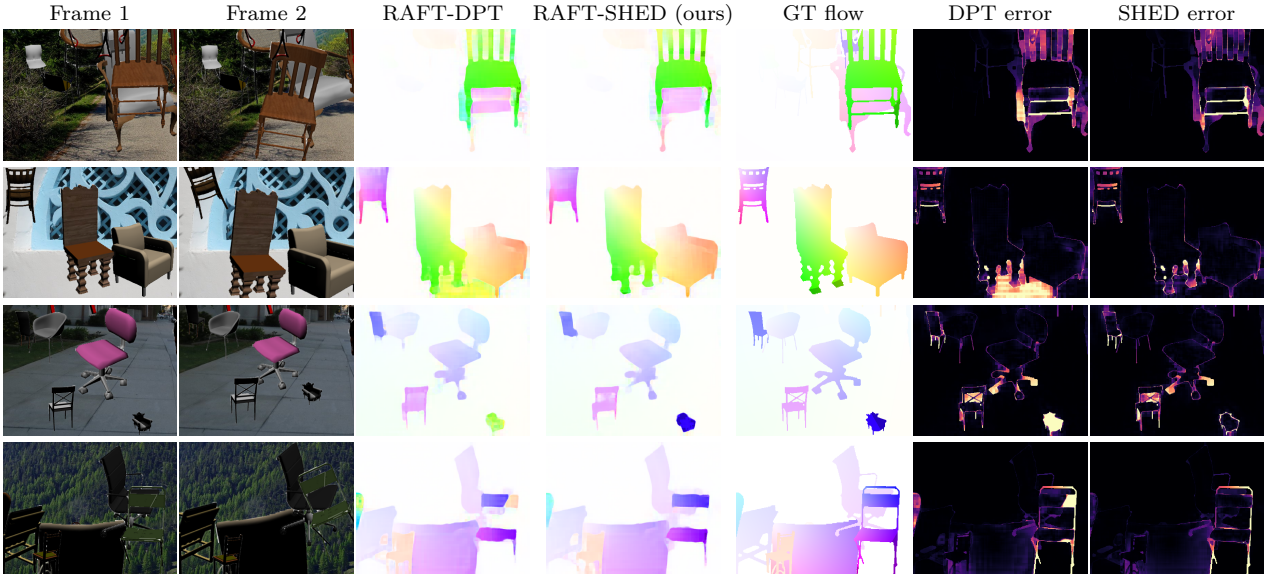


Figure 9: **Qualitative optical-flow comparison with the DPT-grid control.** Both methods use the matched 5,000-step protocol. The last two columns show per-pixel end-point error under the same color scale; darker pixels indicate lower error. Across the full 640-pair validation set, RAFT-SHED obtains lower error on 542 pairs (84.7%).

We train both variants on FlyingChairs for 5,000 steps with a batch size of 8 and training seeds $0, 1, 2$. Training data, crops, optimizer, learning-rate schedule, and the remaining RAFT components are shared. We evaluate the exact final checkpoint from every run on all 640 validation image pairs using 24 RAFT update iterations.

As shown in Table 7, RAFT-SHED achieves lower end-point error (EPE) than RAFT-DPT in every paired training seed. Figure 9 shows the corresponding seed-0 RAFT-DPT and RAFT-SHED predictions. The displayed examples illustrate cases in which the reverse-hierarchy features better preserve object-motion structure and reduce errors around motion boundaries. The complete per-image evaluation for the same seed also favors RAFT-SHED on a large majority of the validation pairs. The reverse-hierarchy features therefore improve motion correspondence under the tested flow-supervised training protocol.

4.5 Ablation studies and computational cost

Decoder control. Table 8 compares SHED with ConvCap+Spatial after 2,500 training steps. Both models retain the grid-token encoder, SEEDS tokenizer, learned forward hierarchy, and DPT-style spatial pyramid. SHED uses assignment-tied reverse processing and all-level readout, whereas ConvCap+Spatial reconstructs fine features from global summaries of the forward hierarchy before spatial decoding. This comparison evaluates the combined reverse hierarchy and readout while preserving progressive spatial upsampling. Both

Table 8: **SHED achieves lower mean depth and boundary errors in the 2,500-step decoder control.** We compare SHED with ConvCap+Spatial on NYUv2 while retaining SEEDS, the learned forward hierarchy, and a DPT-style spatial pyramid in both models. SHED uses assignment-tied reverse processing and all-level readout; ConvCap+Spatial reconstructs fine features from global hierarchy summaries. Both models use 384×384 inputs and Canny thresholds of 0.15/0.30. Results are mean ± sample standard deviation over seeds $0, 1, 2$. Bold denotes the better displayed mean in each column.

Method	$\delta_1 \uparrow$	AbsRel↓	RMSE↓	$\log_{10} \downarrow$	$\varepsilon_a \downarrow$	$\varepsilon_c \downarrow$
ConvCap+Spatial	0.7705±0.0026	0.1638±0.0012	0.5660±0.0019	0.0679±0.0004	5.723±0.120	9.623±0.018
SHED (ours)	0.7705±0.0009	**0.1631±0.0007**	**0.5655±0.0014**	**0.0677±0.0002**	**5.554±0.082**	**9.573±0.013**

Table 9: **All-level versus finest-only readout from frozen NYUv2 checkpoints.** Both settings retain the complete reverse hierarchy. Only the readout configuration is changed at inference, without retraining. Results are mean ± sample standard deviation over the same three fixed checkpoints. Bold denotes the better mean between the two settings.

Readout	$\delta_1 \uparrow$	$\delta_2 \uparrow$	AbsRel↓	RMSE↓
Finest only (L_0)	0.8673±0.0007	0.9794±0.0011	0.1195±0.0017	0.4106±0.0010
All four levels (L_0, L_1, L_2, L_3)	**0.8687±0.0008**	**0.9799±0.0012**	**0.1189±0.0012**	**0.4088±0.0018**

models use 384×384 inputs and seeds $0, 1, 2$. SHED achieves lower mean errors on all reported error metrics, while δ_1 is tied at the displayed precision. The training-seed-level evidence for an improvement in ε_c remains inconclusive. Within this training budget, the combined reverse hierarchy and readout show their largest relative mean error reduction in boundary localization.

Tokenizer and grouping controls. The complete 2,500-step five-arm study additionally evaluates regular-grid initial regions and fixed grouping under the same recipe. Its results are reported in Table 11. The tokenizer and grouping comparisons show small depth-accuracy differences and more noticeable changes in mean boundary localization within the tested configurations.

Multi-level spatial readout. Table 9 compares all-level and finest-only readout using the three fixed primary NYUv2 checkpoints without retraining. Both settings retain the complete reverse hierarchy; only the readout configuration changes at inference. All-level readout yields better mean values on all four reported metrics than finest-only readout. For these fixed checkpoints, directly reading the coarser states complements their propagation through the reverse hierarchy.

Computational characteristics. Table 12 reports the inference cost of DPT and the grid-token encoder configuration of SHED at 384×384, with batch size 1 and BF16 on an NVIDIA H100. We report both model-only and end-to-end latency, with the latter including online SEEDS with 50 refinement iterations.T he measured grid-token implementation has a similar parameter count and peak GPU memory usage to DPT, with additional latency from both neural processing and online SEEDS.

5 Conclusion

We presented SHED, a dense prediction architecture that couples a learned forward segment hierarchy with reverse coarse-to-fine decoding through the same grouping assignments. The hierarchy is learned end-to-end from the task objective without segmentation labels. Under matched training, SHED achieves better mean monocular-depth metrics than DPT, with the largest relative gains under HyperSim-to-NYUv2 transfer without target calibration. Representations learned under depth supervision also show stronger part-level alignment on PartNet, and applying the reverse hierarchy to RAFT consistently reduces optical-flow error across training seeds. Boundary localization, object-wise depth, and 3D reconstruction show smaller mean differences that further characterize the geometric effects of the model. Taken together, these results suggest that tying dense reconstruction to a learned region hierarchy is a useful inductive bias for dense prediction.

Reproducibility. Implementation details are in Section A. Code and trained models are available at https://github.com/lsh3163/SHED.

Acknowledgments

This project was supported in part by U.S. National Science Foundation (NSF) grant 2313151, with additional computing resources provided through the NAIRR Pilot (CIS250430, CIS240421, and CIS240431), and by the South Korean Ministry of Science and ICT (MSIT) under grant RS-2026-25524173. We thank the TMLR Action Editor and anonymous reviewers for their constructive feedback.

References

Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6077–6086, 2018.

Pablo Arbeláez, Bharath Hariharan, Chunhui Gu, Saurabh Gupta, Lubomir Bourdev, and Jitendra Malik. Semantic segmentation using regions and parts. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3378–3385. IEEE, 2012.

Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4009–4018, 2021.

Jiawang Bian, Zhichao Li, Naiyan Wang, Huangying Zhan, Chunhua Shen, Ming-Ming Cheng, and Ian Reid. Unsupervised scale-consistent depth and ego-motion learning from monocular video. *Advances in neural information processing systems*, 32, 2019.

John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

Po-Yi Chen, Alexander H Liu, Yen-Cheng Liu, and Yu-Chiang Frank Wang. Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pp. 2624–2632, 2019.

Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. *Advances in neural information processing systems*, 29, 2016.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9640–9649, 2021.

Xinjing Cheng, Peng Wang, and Ruigang Yang. Depth estimation via affinity learned with convolutional spatial propagation network. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 103–119, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Zhiwei Deng, Ting Chen, and Yang Li. Perceptual group tokenizer: Building perception with iterative grouping. *arXiv preprint arXiv:2311.18296*, 2023.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Ainaz Eftekhar, Kuo-Hao Zeng, Jiafei Duan, Ali Farhadi, Ani Kembhavi, and Ranjay Krishna. Selective visual representations improve convergence and generalization for embodied ai. *arXiv preprint arXiv:2311.04193*, 2023.

David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE international conference on computer vision*, pp. 2650–2658, 2015.

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.

Mohamed El Banani, Amit Raj, Kevis-Kokitsi Maninis, Abhishek Kar, Yuanzhen Li, Michael Rubinstein, Deqing Sun, Leonidas Guibas, Justin Johnson, and Varun Jampani. Probing the 3d awareness of visual foundation models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21795–21806, 2024.

SM Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. *Advances in neural information processing systems*, 29, 2016.

Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 605–613, 2017.

David A Forsyth and Jean Ponce. *Computer vision: a modern approach*. prentice hall professional technical reference, 2002.

Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2002–2011, 2018.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3354–3361. IEEE, 2012.

Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 270–279, 2017.

Vitor Guizilini, Rui Hou, Jie Li, Rares Ambrus, and Adrien Gaidon. Semantically-guided representation learning for self-supervised monocular depth. *arXiv preprint arXiv:2002.12319*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Xingzhe He, Bastian Wandt, and Helge Rhodin. Ganseg: Learning to segment by unsupervised hierarchical image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1225–1235, 2022.

Shaul Hochstein and Merav Ahissar. View from the top: Hierarchies and reverse hierarchies in the visual system. *Neuron*, 36(5):791–804, 2002.

Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. Inferring semantic layout for hierarchical text-to-image synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7986–7994, 2018.

Jyh-Jing Hwang, Stella X Yu, Jianbo Shi, Maxwell D Collins, Tien-Ju Yang, Xiao Zhang, and Liang-Chieh Chen. Segsort: Segmentation by discriminative sorting of segments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7334–7344, 2019.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

Hyunwoo Kang, Sangwoo Mo, and Jinwoo Shin. Oamixer: Object-aware mixing layer for vision transformers. *arXiv preprint arXiv:2212.06595*, 2022.

Tsung-Wei Ke, Jyh-Jing Hwang, Yunhui Guo, Xudong Wang, and Stella X Yu. Unsupervised hierarchical semantic segmentation with multiview cosegmentation and clustering transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2571–2581, 2022.

Tsung-Wei Ke, Sangwoo Mo, and X Yu Stella. Learning hierarchical image segmentation for recognition and by recognition. In *The Twelfth International Conference on Learning Representations*, 2024.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7482–7491, 2018.

Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Advances in neural information processing systems*, 24, 2011.

Zhenyu Li, Xuyang Wang, Xianming Liu, and Junjun Jiang. Binsformer: Revisiting adaptive bins for monocular depth estimation. *IEEE Transactions on Image Processing*, 2024.

Fayao Liu, Chunhua Shen, Guosheng Lin, and Ian Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE transactions on pattern analysis and machine intelligence*, 38(10): 2024–2039, 2015.

Minghuan Liu, Zixuan Chen, Xuxin Cheng, Yandong Ji, Ri-Zhao Qiu, Ruihan Yang, and Xiaolong Wang. Visual whole-body control for legged loco-manipulation. *arXiv preprint arXiv:2403.16967*, 2024.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in neural information processing systems*, 33:11525–11538, 2020.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

Yunze Man, Shuhong Zheng, Zhipeng Bao, Martial Hebert, Liangyan Gui, and Yu-Xiong Wang. Lexicon3d: Probing visual foundation models for complex 3d scene understanding. *Advances in Neural Information Processing Systems*, 37:76819–76847, 2024.

David Marr. *Vision: A computational investigation into the human representation and processing of visual information*. MIT press, 2010.

Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 909–918, 2019.

Sangwoo Mo, Minsu Cho, and Jinwoo Shin. Instagan: Instance-aware image-to-image translation. *arXiv preprint arXiv:1812.10889*, 2018.

Sangwoo Mo, Hyunwoo Kang, Kihyuk Sohn, Chun-Liang Li, and Jinwoo Shin. Object-aware contrastive learning for debiased scene representation. *Advances in Neural Information Processing Systems*, 34: 12251–12264, 2021.

Arsalan Mousavian, Hamed Pirsiavash, and Jana Košecká. Joint semantic segmentation and depth estimation with deep convolutional networks. In *2016 Fourth International Conference on 3D Vision (3DV)*, pp. 611–619. IEEE, 2016.

Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

Michael Ramamonjisoa, Yuming Du, and Vincent Lepetit. Predicting sharp and accurate occlusion boundaries in monocular depth estimation using displacement fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14648–14657, 2020.

Kanchana Ranasinghe, Brandon McKinzie, Sachin Ravi, Yinfei Yang, Alexander Toshev, and Jonathon Shlens. Perceptual grouping in contrastive vision-language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5571–5584, 2023.

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.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *ICLR*, 2025.

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.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241. Springer, 2015.

Mehdi SM Sajjadi, Daniel Duckworth, Aravindh Mahendran, Sjoerd Van Steenkiste, Filip Pavetic, Mario Lucic, Leonidas J Guibas, Klaus Greff, and Thomas Kipf. Object scene representation transformer. *Advances in neural information processing systems*, 35:9512–9524, 2022.

Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, et al. Bridging the gap to real-world object-centric learning. *arXiv preprint arXiv:2209.14860*, 2022.

Baifeng Shi, Trevor Darrell, and Xin Wang. Top-down visual attention from analysis by synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2102–2112, 2023.

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1746–1754, 2017.

Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 6243–6252, 2017.

Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pp. 402–419. Springer, 2020.

Michael Van den Bergh, Xavier Boix, Gemma Roig, Benjamin De Capitani, and Luc Van Gool. Seeds: Superpixels extracted via energy-driven sampling. In *Computer Vision–ECCV 2012: 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VII 12*, pp. 13–26. Springer, 2012.

Jamie Watson, Michael Firman, Gabriel J Brostow, and Daniyar Turmukhambetov. Self-supervised monocular depth hints. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 2162–2171, 2019.

Max Wertheimer. Laws of organization in perceptual forms. 1938.

Jiarui Xu, Shalini De Mello, Sifei Liu, Wonmin Byeon, Thomas Breuel, Jan Kautz, and Xiaolong Wang. Groupvit: Semantic segmentation emerges from text supervision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 18134–18144, 2022.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *arXiv preprint arXiv:2406.09414*, 2024.

Zhenheng Yang, Peng Wang, Wei Xu, Liang Zhao, and Ramakant Nevatia. Unsupervised learning of geometry from videos with edge-aware depth-normal consistency. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Wei Yin, Yifan Liu, Chunhua Shen, and Youliang Yan. Enforcing geometric constraints of virtual normal for depth prediction. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 5684–5693, 2019.

Zhichao Yin and Jianping Shi. Geonet: Unsupervised learning of dense depth, optical flow and camera pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1983–1992, 2018.

Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. Neural window fully-connected crfs for monocular depth estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3916–3925, 2022.

Huangying Zhan, Ravi Garg, Chamara Saroj Weerasekera, Kejie Li, Harsh Agarwal, and Ian Reid. Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 340–349, 2018.

Shengjie Zhu, Garrick Brazil, and Xiaoming Liu. The edge of depth: Explicit constraints between segmentation and depth. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13116–13125, 2020.

A Implementation Details

A.1 Training details

The NYUv2 training settings and input resolutions are specified in the main experimental setup. RGB images are resized using bilinear interpolation, while depth maps and valid-depth masks use nearest-neighbor interpolation. Raw depth values are converted from millimeters to meters.

Both models use the same augmentation policy. Horizontal flipping is applied with probability 0.5. Geometric transformations are applied jointly to RGB, depth, and valid-depth masks, while photometric transformations affect only RGB. The policy includes brightness, contrast, saturation, gamma, and hue adjustments, together with translation, scaling, and rotation.

Tokenization. For SHED, SEEDS (Van den Bergh et al., 2012) operates on the spatially augmented RGB image, converted to Lab, before photometric augmentation and ImageNet normalization. OpenCV SEEDS uses one internal level, prior 2, five histogram bins, and 50 refinement iterations. The primary 480×640 NYUv2 configuration uses 1,200 initial superpixels; the 384×384 control experiments use 576. The corresponding hierarchy schedules are specified in the main text.

A.2 Depth and structural evaluation

We evaluate on the 654-image NYUv2 test split (Nathan Silberman & Fergus, 2012) at 480×640 resolution, bilinearly resizing lower-resolution predictions when needed. Crop coordinates below are zero-based and half-open, with rows specified before columns.

Per-pixel depth metrics. We use the crop $[45{:}471, 41{:}601]$ and valid ground-truth depths satisfying $0.01 < d < 10.0$ meters. Predictions are clamped to $[0.01, 10.0]$ meters. Threshold accuracy δ_k is the fraction of valid pixels satisfying $\max(\widehat{d}/d, d/\widehat{d}) < 1.25^k$, for $k = 1, 2, 3$. We also report AbsRel, RMSE, and mean absolute $\log_{10}$ error. Each metric is computed per image and then averaged over images.

NYUv2-OC++ boundary protocol. Boundary evaluation uses the crop $[20{:}460, 24{:}616]$. Zero depth is invalid. Predicted depth is min–max normalized over valid support in float32 before applying `skimage.feature.canny` with $\sigma = \sqrt{2}$ and thresholds 0.15/0.30. ε_a is the mean distance to the nearest ground-truth boundary over predicted boundary pixels whose distance is strictly below 10 pixels. ε_c sums nearest-boundary distances in both directions, clipped at 10 pixels, and divides by the combined predicted and ground-truth boundary-pixel count. If the eligibility set for ε_a is empty, we assign both ε_a and ε_c a value of 10 pixels.

Object-wise depth metrics. We compute the depth metrics separately within each annotated object mask intersected with valid-depth support. For each training seed, scores are averaged uniformly over all valid object instances across the evaluation set.

3D evaluation. Predicted and ground-truth depths are back-projected using the same NYUv2 camera intrinsics and valid-pixel masks. Chamfer distances are computed per image in both directions and averaged over the 654 test images.

A.3 Efficiency measurement protocol

We measure inference on one NVIDIA H100 in BF16 with batch size one at 384×384. Each of five fresh-process repeats includes 50 warm-up iterations and 500 measured iterations with explicit CUDA synchronization; we report the median over repeats. Model-only latency covers the GPU forward pass. End-to-end latency starts from an in-memory RGB image and includes tensor preparation, online SEEDS, host-to-device transfer, and depth decoding; disk decoding is excluded. SEEDS uses 576 initial regions, 50 refinement iterations, and one CPU worker for this measurement. Peak allocated GPU memory excludes CPU memory used by SEEDS, and neural FLOPs exclude CPU preprocessing. Measured costs are reported in Table 12.

Table 10: **SHED lowers both boundary-error means across three fixed Canny settings on NYUv2-OC++.** The same frozen matched-384 predictions are evaluated at every displayed threshold pair. Predictions are bilinearly restored to the native 480×640 evaluation grid before scoring. Results are mean $\pm$ sample standard deviation over training seeds $0, 1, 2$. Density is the percentage of cropped image pixels detected as boundaries and is reported as a diagnostic with no preferred direction. Bold denotes the lower displayed mean boundary error within each threshold pair.

Thresholds	Method	$\varepsilon_a \downarrow$	$\varepsilon_c \downarrow$	Density (%)
0.10/0.20	DPT	4.8419±0.0279	9.1683±0.0233	0.2783±0.0100
	SHED (ours)	**4.7787±0.0432**	**9.1489±0.0407**	0.2912±0.0127
0.15/0.30	DPT	5.5148±0.0425	9.5716±0.0073	0.1292±0.0037
	SHED (ours)	**5.3988±0.0782**	**9.5466±0.0195**	0.1388±0.0057
0.20/0.40	DPT	6.8345±0.0561	9.7842±0.0067	0.0599±0.0033
	SHED (ours)	**6.5790±0.0786**	**9.7652±0.0103**	0.0677±0.0027

B Additional Controls and Robustness Analyses

B.1 Boundary-threshold and edge-density sensitivity

Table 10 evaluates the same frozen matched-384 predictions used for the boundary block of Table 2. Predictions are bilinearly restored to the native 480×640 evaluation grid before scoring. Following the reviewer's request, we report predicted-edge density together with a sensitivity analysis over three fixed Canny threshold pairs spanning the tested range. The middle pair, 0.15/0.30, is the primary setting, and the sweep is not used to select a favorable threshold for either method.

Across the three fixed Canny settings, SHED has lower mean ε_a and ε_c than DPT at every setting. The reductions in ε_a are 1.3%, 2.1%, and 3.7% at 0.10/0.20, 0.15/0.30, and 0.20/0.40, respectively. The corresponding reductions in ε_c are also preserved at all three operating points. This repeated mean ordering shows that the reported boundary result is not tied to a single Canny threshold.

SHED produces denser edge maps, with absolute density increases of 0.0129, 0.0096, and 0.0078 percentage points across the three settings. We report these values explicitly because edge density can affect boundary metrics. Notably, the density increase is not accompanied by worse predicted-to-reference localization: ε_a, which penalizes spurious or mislocalized predicted edges, is lower for SHED at every threshold. Moreover, the improvement occurs simultaneously in ε_a and the symmetric error ε_c throughout the tested threshold range. These results support a favorable localization–coverage trade-off rather than indiscriminate edge proliferation.

B.2 Factor-isolation controls

Table 11 reorganizes the complete five-arm, 2,500-step experiment into three direct factor contrasts. All variants use the same training data, initialization protocol, optimization, 384×384 input resolution, training budget, evaluation inventory, and three training seeds. Within each contrast, the named component is changed while the remaining hierarchy and prediction pipeline are retained.

Tokenizer control. The RH and Grid+RH arms differ only in their initial regions: RH uses SEEDS regions, whereas Grid+RH uses a regular 24×24 token grid. Their depth results are nearly identical. RH has lower ε_a by 0.046, whereas Grid+RH has lower ε_c by 0.013. Thus, the reverse-hierarchy model retains comparable depth and boundary behavior after the hand-crafted SEEDS boundary prior is removed. The structural behavior therefore cannot be explained solely by SEEDS snapping the initial regions to RGB gradients.

Learned-grouping control. The RH and Fixed+RH arms retain the same tokenizer, hierarchy sizes, reverse-processing path, and training recipe. Fixed+RH replaces the learned fine-to-coarse assignments with

Table 11: **Matched factor controls isolate the tokenizer, learned grouping, and assignment-tied reverse reconstruction.** All controls use 384×384 inputs, 2,500 training steps, and training seeds $0, 1, 2$. Each row changes only the factor named in the first column. We report the gain produced by the proposed factor: for δ_1, gain is proposed minus control, whereas for error metrics it is control minus proposed. Positive values therefore favor the proposed factor. The tokenizer comparison is a stress test of the SEEDS prior: replacing SEEDS with regular grid tokens produces only small, mixed changes, showing that the reverse hierarchy does not depend on the hand-crafted boundary prior. Learned assignments and assignment-tied reverse reconstruction improve both boundary-error directions. The boundary intervals are 95% paired-image bootstrap intervals conditioned on the three fixed training seeds.

Factor isolated	Controlled replacement	Gain in δ_1	Gain in AbsRel	Gain in ε_a	Gain in ε_c
Tokenizer dependence	Grid tokens → SEEDS	+0.0004	+0.0000	+0.046	−0.013
Learned grouping	Fixed → learned assignments	+0.0003	+0.0000	**+0.216**	**+0.035**
Reverse reconstruction	ConvCap → assignment-tied reverse	+0.0000	**+0.0007**	**+0.169**	**+0.050**

fixed spatial assignments. Restoring the learned assignments reduces ε_a from 5.550 to 5.334 and ε_c from 9.574 to 9.539, corresponding to improvements of 0.216 and 0.035. The conditional paired-image 95% confidence intervals are $[0.131, 0.305]$ and $[0.023, 0.046]$, respectively. This controlled comparison isolates a favorable contribution from learning the hierarchy assignments rather than merely imposing fixed multiscale grouping.

Reverse-reconstruction control. The SHED+Spatial and ConvCap+Spatial arms retain SEEDS, the learned forward hierarchy, the same DPT-style spatial pyramid, and identical total and trainable parameter counts. ConvCap+Spatial replaces assignment-tied reverse transport and all-level hierarchy readout with reconstruction conditioned on global hierarchy summaries. Restoring the proposed reverse reconstruction improves AbsRel from 0.1638 to 0.1631 and reduces ε_a from 5.723 to 5.554 and ε_c from 9.623 to 9.573. The corresponding boundary improvements are 0.169 and 0.050, with conditional paired-image 95% confidence intervals of $[0.074, 0.263]$ and $[0.037, 0.063]$. Table 8 is the 2,500-step decoder control. The primary 40,000-step comparison remains SHED versus DPT in Table 1.

Together, these contrasts separate the three principal architectural explanations. Removing SEEDS does not remove the structural behavior, learned assignments improve both boundary errors over fixed grouping, and assignment-tied reverse reconstruction improves AbsRel and both boundary errors over the matched reconstruction control. The paired-image intervals quantify consistency across the fixed 654-image evaluation set and are reported separately from training-seed-level inference.

B.3 Computational cost

Table 12 includes online SEEDS in the end-to-end inference path. Under this protocol, SHED requires more neural FLOPs, latency, and peak allocated GPU memory than DPT.

Table 12: **Inference cost at 384×384.** Latencies are milliseconds per image under the protocol in Section A. Neural FLOPs exclude CPU preprocessing. End-to-end (E2E) latency includes online 50-iteration SEEDS. GPU memory is peak allocated memory for the end-to-end path.

Method	Params (M)	GFLOPs	SEEDS (ms)	Model (ms)	E2E (ms)	GPU (MB)
DPT	32.32	122.34	–	8.84	9.10	242.9
SHED	32.52	132.29	133.84	58.87	193.41	245.3

Recorded training cost. Recorded task-training runs account for 311.95 GPU-hours, including auxiliary experiments but excluding evaluation and unrecorded or failed development runs. Optical-flow runs have a separate recorded service-time upper bound of 11.44 GPU-hours, including evaluation and service overhead.

C Layout-aware Retrieval

C.1 Scene and frame retrieval

We evaluate scene-level and frame-level retrieval on NYUv2 using the depth-trained representations of DPT and SHED. For each image, we extract the class token from the final ViT layer, apply ℓ_2 normalization, and rank the other images by cosine similarity. Scene-level retrieval treats other frames from the same annotated scene as relevant matches. Frame-k retrieval evaluates matches to temporally neighboring frames within range k. We report Top-K retrieval accuracy for $K \in \{1, 3, 5\}$.

Figure 10 presents the original scene-level and frame-level results, together with Top-5 retrieval examples. SHED achieves higher retrieval accuracy than DPT at all three reported ranks in both settings. Figure 11 provides additional Top-1 examples. These CLS-based evaluations allow same-scene and temporal matches; the following evaluation uses a common patch-token representation and cross-scene candidates to control for these factors.

a) Top-5 retrieval examples



b) Top-K retrieval accuracy (%)

Method	Top-1	Top-3	Top-5
Scene retrieval			
DPT	45.2	69.7	77.2
SHED (ours)	**60.5**	**78.7**	**87.0**
Frame retrieval (k=5)			
DPT	18.5	31.0	38.3
SHED (ours)	**30.5**	**42.3**	**48.3**

Figure 10: **SHED learns layout-aware representations through depth supervision.** We evaluate image retrieval on NYUv2 based on cosine similarity between class tokens from the final ViT block. **a)** Top-5 results (ranked left to right), with similarity scores shown below. SHED retrieves images with similar layouts, such as a central desk and a rear bookshelf, while DPT retrieves unrelated scenes. **b)** Top-K accuracy at the scene and frame level ($k = 5$), where the targets are different views from the same scene or nearby frames. SHED achieves higher retrieval accuracy than DPT at all three reported ranks in both scene and frame retrieval.



Figure 11: **Comparison of top-1 image retrieval results.** SHED retrieves samples that are more structurally similar to the query, indicating that its global embedding effectively captures scene layouts. In contrast, DPT focuses more on visual appearance, as shown in column 3, where it retrieves an image from a different scene that shares a similar color of blue.

C.2 Retrieval with mean-pooled patch tokens

We evaluate DPT and SHED using mean-pooled encoder patch tokens to compare their representations with a common readout. For each image, we average the 576 patch tokens from the final encoder block, apply ℓ_2 normalization, and rank gallery images by cosine similarity. Both models produce 384-dimensional descriptors. We evaluate the three NYUv2 depth-training seeds $0, 1, 2$. To control for scene identity, we use disjoint query and gallery sets from the 654-image NYUv2 test inventory. All gallery candidates come from a different recording scene from the query, excluding same-scene and temporal matches.

We construct retrieval relevance labels from the available ground-truth depth maps. A model-independent, scale-neutral descriptor of ground-truth log depth and gradients defines geometric similarity. For each query, we select five geometry-nearest images among appearance-dissimilar candidates as positives. We additionally select 59 appearance-nearest images among geometry-dissimilar candidates as negatives, forming a fixed 64-image candidate set. Thus, positives are defined by a depth-based similarity criterion rather than manually annotated image-pair labels. The candidate sets and relevance labels are identical for DPT and SHED and are fixed before model evaluation. Ground-truth depth is used to construct the evaluation targets; the models rank candidates solely using their image representations.

Table 13: **Retrieval with mean-pooled patch tokens.** Values are percentages, reported as mean ± sample standard deviation over three training seeds. AP is average precision and MRR is mean reciprocal rank. Bold denotes the higher mean.

Method	AP ↑	R@1 ↑	R@3 ↑	R@5 ↑	MRR ↑
DPT	18.52±0.57	14.92±1.28	32.82±2.49	47.63±1.98	30.57±1.23
SHED	**20.03±0.33**	**19.24±1.70**	**37.24±2.66**	**49.69±1.60**	**34.18±1.28**

SHED achieves higher means on all five retrieval metrics in Table 13.

D Additional Visualizations

D.1 More visualizations of depth maps



Figure 12: **Comparison of depth maps.** SHED captures more accurate shapes with sharper boundaries, whereas DPT produces blurrier results.

D.2 More visualizations of 3D reconstruction

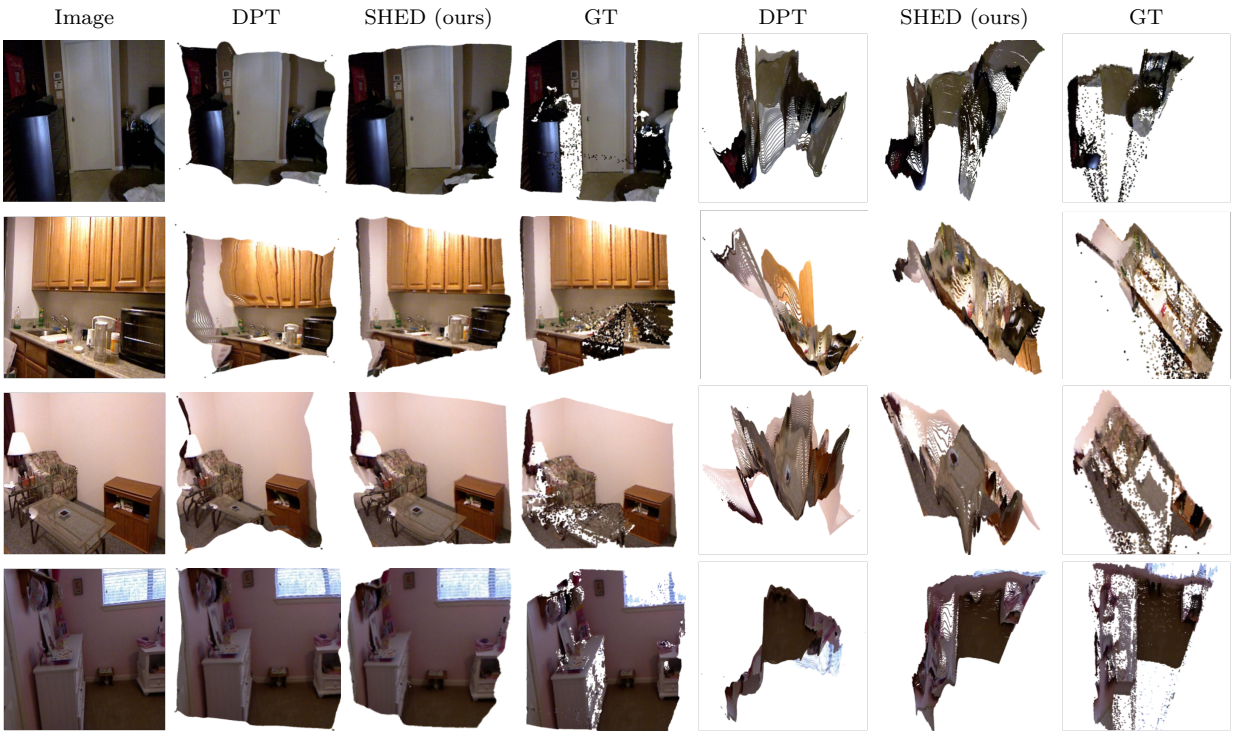


Figure 13: **Comparison of 3D reconstruction results.** Frontal views (cols 2–4) and bird's-eye views (cols 5–7). DPT yields curved wall boundaries in the frontal views, which lead to distorted 3D reconstructions visible in the bird's-eye views. In contrast, SHED produces sharp depth edges that preserve straight object contours and more faithfully represent the ground-truth 3D geometry.

E Limitations and Broader Impacts

E.1 Limitations

SHED uses a fixed token-count schedule, although the learned grouping assignments remain image-dependent. We have not established robustness to arbitrary schedules or tokenizers, and content-adaptive token counts remain a possible extension. SEEDS is non-differentiable and contributes substantial CPU latency under the measured protocol. The reported controls cover specific configurations and do not show uniform gains from every hierarchy component. Evaluation on the selected datasets does not establish reliability under all scene, sensor, or domain shifts.

E.2 Broader impacts

Structured depth may benefit mapping and robotics, but SHED has not been validated for safety-critical use. More detailed scene reconstruction can also facilitate surveillance or expose private spatial information about people and locations, so deployment requires appropriate data-access and privacy safeguards.

ImageNet pretraining and the depth datasets may encode geographic, demographic, scene, and sensor biases; we have not measured performance across demographic groups. Recorded training costs and their exclusions are reported in Section B. The supplement includes model implementations, NYUv2 training configurations and drivers, and result summaries. It does not include every task-specific training and evaluation driver or all trained checkpoints.