Mind The Edge: Refining Depth Edges in Sparsely-Supervised Monocular Depth Estimation

Lior Talker[1]   Aviad Cohen[1]   Erez Yosef[1,2]   Alexandra Dana[1]   Michael Dinerstein[1]

[1]Samsung Israel R&D Center, Tel Aviv, Israel   [2]Tel Aviv University, Israel

{lior.talker,aviad.cohen,alex.dana,m.dinerstein}@samsung.com   erez.yo@gmail.com
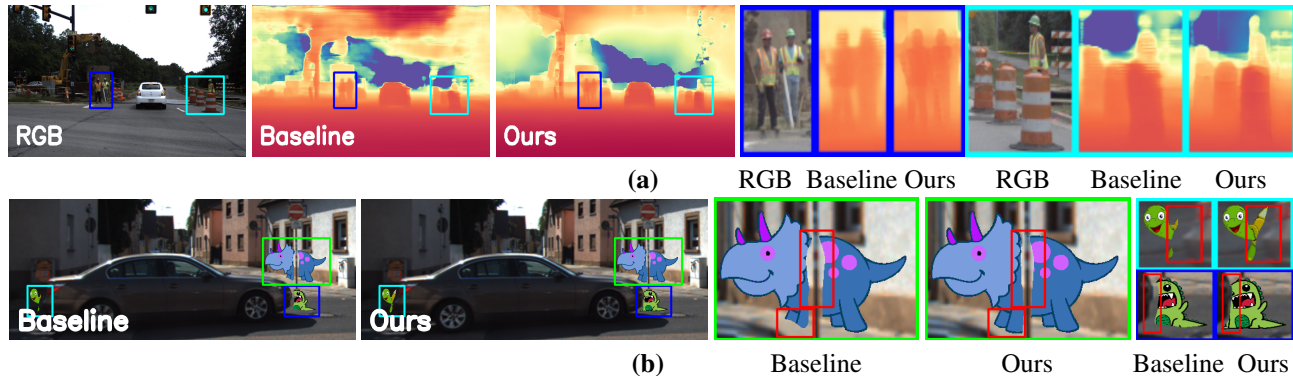
Figure 1. **Refining depth edges with our method when using Packnet-SAN [18] as an MDE baseline.** (a) Depth estimation (DDAD dataset). Zoom-in on the crops (on the right) to see the improvement in the 2D localization of the depth edges between the baseline and our method. (b) Augmented reality. An example of virtual objects planted in a scene from the KITTI dataset for AR applications. Zoom-in and inspect the boundaries for the best impression of the depth edges accuracy.

Abstract

Monocular Depth Estimation (MDE) is a fundamental problem in computer vision with numerous applications. Recently, LIDAR-supervised methods have achieved remarkable per-pixel depth accuracy in outdoor scenes. However, significant errors are typically found in the proximity of depth discontinuities, i.e., depth edges, which often hinder the performance of depth-dependent applications that are sensitive to such inaccuracies, e.g., novel view synthesis and augmented reality. Since direct supervision for the location of depth edges is typically unavailable in sparse LIDAR-based scenes, encouraging the MDE model to produce correct depth edges is not straightforward. To the best of our knowledge this paper is the first attempt to address the depth edges issue for LIDAR-supervised scenes. In this work we propose to learn to detect the location of depth edges from densely-supervised synthetic data, and use it to generate supervision for the depth edges in the MDE training. To quantitatively evaluate our approach, and due to the lack of depth edges GT in LIDAR-based scenes, we manually annotated subsets of the KITTI and the DDAD datasets with depth edges ground truth. We demonstrate significant gains in the accuracy of the depth edges with comparable per-pixel depth accuracy on several challeng-

ing datasets. Code and datasets are available at `https://github.com/liortalker/MindTheEdge`.

1. Introduction

Monocular Depth Estimation (MDE) aims to recover the depth of each pixel in a single RGB image. It is used in many important applications, such as robotic navigation [10], novel view synthesis [6] and Augmented Reality (AR) [27]. Nonetheless, MDE is an ill-posed problem; that is, a single RGB image may be generated from many possible scenes. However, in recent years, many MDE methods, based on Convolutional Neural Networks (CNNs) have shown remarkable results. CNN-based supervised methods are trained with depth Ground Truth (GT) which is usually dense for indoor scenes, e.g., acquired using a RGBD camera [41], and sparse for outdoor scenes, e.g., acquired using a LIDAR sensor [15].

It was observed in several papers [29, 49, 55] that the 2D locations of depth discontinuities, which we refer to as *depth edges* in this paper, are often poorly estimated, resulting in thick smooth depth gradients or incorrectly-localized edges (see the baseline in Fig. 1). Some applications that use predicted depth maps are highly sensitive to errors in depth edges, which are often part of the silhouette of ob-
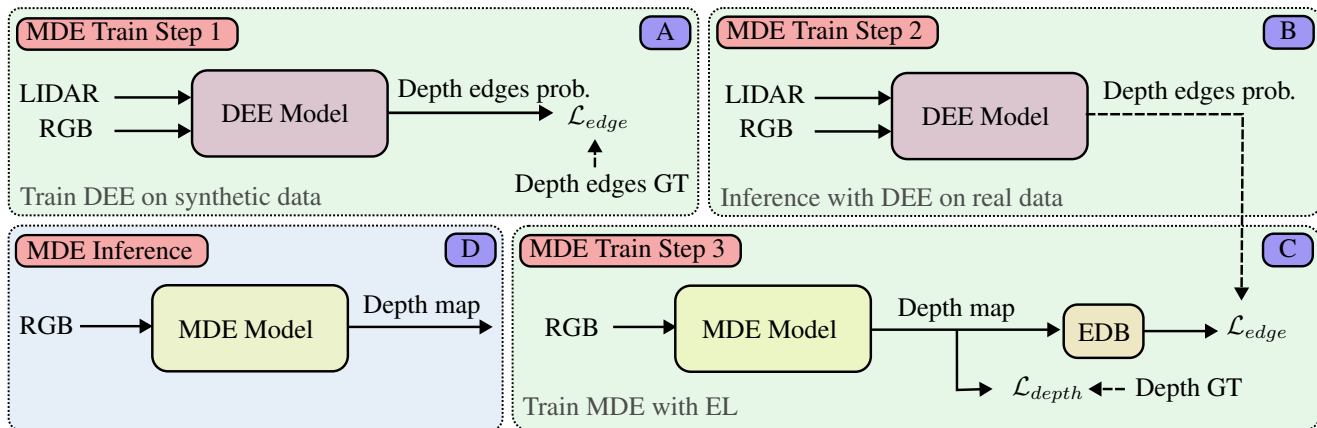
Figure 2. **Overview of our proposed MDE training method.** (A) Training the DEE model on synthetic data. (B) Inferring depth edges on the training set of the real data using the trained DEE model. (C) Training the MDE model on real data with the Edge Loss (EL) using the supervision from the previous step. (D) Inference using the MDE model on the real data. Solid and broken lines represent the dataflow and the GT used in loss functions, respectively.

jects. One example of such an application is Novel View Synthesis (NVS) – generating a new view of a scene captured from one or more views. In NVS methods that use depth explicitly [6], these type of localization errors in the 2D location of depth edges, may result in wrongly localized object parts in another newly generated view. Another important application that often uses predicted depth is virtual object rendering for AR [27], which computes for each pixel the closest occluding object from the point of view of the user. When relying on inaccurate depth edges in the computation of the occlusion, significant artifacts and unrealistic appearance may occur (see Fig. 1).

Roughly speaking, the MDE network solves two subtasks: (i) depth edge estimation, and (ii) continuous surface depth estimation. Our method is based on the observation that MDE networks tend to focus on the latter much more than the former. The underlying reasons for this behaviour are probably twofold. The first is the small impact of the depth edges on the network's loss since they occupy only a tiny portion of the image. The second, which is partially discussed in [9, 51], is due to alignment errors between the RGB image and the LIDAR signal. Specifically, LIDAR measurements are often wrongly associated with objects although belonging to the background, or absent in occluded areas that are revealed in the time gap between the RGB and the LIDAR data acquisition (Fig. 3b). This phenomenon is also depicted in Fig. 3a for the LIDAR of the KITTI dataset [15], where regions close to depth edges suffer from a lower density of LIDAR measurements.

In this paper we propose to improve the accuracy of the depth edges in MDE methods by directly encouraging the depth edges of the predicted depth map to be well-localized. Assuming that depth edges GT is available, a dedicated *depth edges loss* that encourages the network to generate

depth discontinuities in the correct locations may be used. However, due to the sparsity of the depth GT in typical outdoor scenes (e.g., KITTI), which are considered in this paper as our target domain, obtaining accurate depth edges GT is a challenging task. To this end, we are the first to propose a method to tackle this problem in sparsely-supervised MDE methods. We note that another aspect of depth edges accuracy, namely edge *sharpness* [7], is not considered in this paper, and is complementary to our work.

To this end, we propose to train a Depth Edge Estimation (DEE) network (Sec. 3.5), which predicts the probability that a pixel is located on a depth edge, using an accurate dense depth GT of a synthetic dataset [22] (Fig. 2A). Using the DEE network we infer a probabilistic map of depth edges on the training set of the target real domain (Fig. 2B). These maps are then used to guide the proposed edge loss in the training of the MDE network (Fig. 2C). Although training on synthetic data and inferring on real data is known to lead to a performance decrease due to a 'domain gap' [32, 38], our experiments (Sec. 4) demonstrate that the DEE network performs very well in practice. In particular, the DEE network's obtained depth edges are significantly more accurate than the depth edges 'naturally' obtained in the standard MDE training (Fig. 5). This is probably, as discussed above, due to the low attention paid to estimating the correct locations of the depth edges in the MDE training. Importantly, the MDE, which is trained only on the target dataset, benefits from the depth edges predictions of the DEE network despite their imperfection.

Due to the lack of depth edges GT for evaluation in LIDAR-supervised real-world datasets, we manually annotated depth edges in two evaluation sets of 102 and 50 images from the KITTI and DDAD datasets, respectively (see an example in the top of Fig. 6). We show on these newly
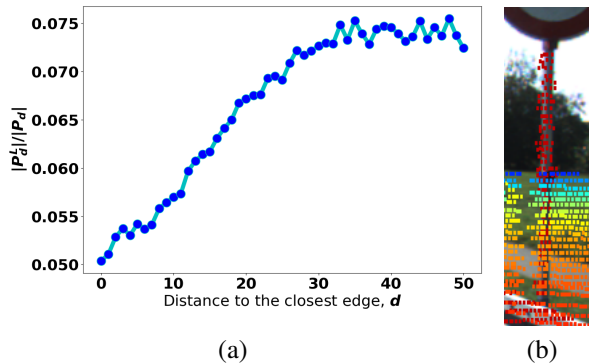
Figure 3. **The density of the LIDAR near edges in a partial set of the KITTI dataset (our proposed KITTI-DE dataset).** Denote the set of all pixels with a distance d to the closest edge as P_d, and the set of pixels, p, in $p \in P_d$ with LIDAR measurement as P_d^L. (a) The ratio of LIDAR measurements, $|P_d^L|/|P_d|$, out of all pixels in P_d, as a function of d. (b) An example from the KITTI dataset of a gap in the LIDAR measurments (left of the pole) and an infiltration of LIDAR measurements from the background to the pole (right of the pole). For visualization purposes the LIDAR measurements are dilated.

annotated datasets that our approach generates depth with significantly more accurate depth edges than the MDE baseline, while maintaining a similar depth accuracy.

Our main contributions are twofold:

- A novel method to improve the localization of depth edges in MDE methods while preserving their per-pixel depth accuracy.
- A benchmark of human-annotated depth edges to evaluate their quality in sparsely-supervised MDE algorithms.

2. Previous Work

Depth estimators can be trained using full supervision with LiDAR or other absolute depth measurements [1, 4, 18, 26, 30, 31, 42], by self-supervision using two or more RGB images [16, 17, 19, 46, 54] or using semi-supervision [3, 21, 25, 47]. The main objective of these methods is to reduce the overall mean absolute relative error (ARE), recently achieving state-of-the art (SOTA) result with ARE close to 5% on the KITTI dataset [1, 18, 31]. However, specific applications such as AR require that the edges of the predicted depth are also accurate.

To achieve this, various depth estimation models were trained in a fully-supervised manner using dense depth GT in datasets such as NYU-2 [41], IBims-1 [24] and Middlebury [39]. Dense outdoor depth maps include the small scale ETH3D [40] dataset and large-scale DIODE [44]. SharpNet [34] improved depth prediction on object edges in indoor scenes by predicting occluding contours. Their solution was first pretrained to predict occluding contours on synthetic data and then fine-tuned on the indoor NYUv2 dataset to constrain normal, depth and occluding contours.

In their subsequent work [33] the authors predicted displacement fields of pixels with poorly predicted depth values of dense depth maps. DCTNet [53] boosted the resolution of depth maps from low-resolution ones using an edge attention and high-resolution RGB images, where they assume co-occurrence between the texture edges of RGB images and depth edges. These methods require training on dense GT collected using expensive short-range laser scanners which are mostly ineffective for outdoor scenes.

Other works utilized the partial consistency between the edges of semantic classes and depth edges to improve the depth predictions on these regions. A dedicated loss was developed [55] which slightly improved the ARE near the edges of the segmentation masks. However, the edges between semantic classes do not necesserily match the depth edges; for example, depth edges between the instances of the same class (e.g., buildings). [37] and [49] used stereo images in addition to panoptic segmentation maps to refine depth edges. Recently, [8] attempted to solve a phenomenon of 'edge-fattening' by redesigning the triplet loss for depth estimation. However, the solution is designed for self-supervised approaches, and their per-pixel accuracy is low. In contrast, our work improves depth edges without using additional segmentation maps from the target domain, while achieving comparable ARE to SOTA methods.

In an attempt to overcome the sparsity limitation of ground-truth depth maps for outdoor scenes, disparity maps were generated from the 3D Movies Dataset [36]. However, stereo matching is a non-trivial task, which is especially inaccurate around depth edges [36]. Moreover, without knowing the exact baseline and other camera characteristics, the estimated depth maps units cannot be metric-reconstructed as needed for applications that require absolute depth. MiDAS [36] and DPT [35] were also trained on this dataset, achieving visually striking depth maps. However, the lack of absolute scale limits their usage.

In recent works, [29] and [11] introduced methods to merge two depth maps at different resolutions, trained on dense depth datasets. The accuracy of [29] was evaluated using order ranking around depth edges, but did not measure the absolute depth metrics on sparsely-supervised datasets, while [11] did measure absolute depth metrics but not the accuracy around depth edges. To the best of our knowledge we are the first to consider the problem of depth edges in *sparse LIDAR-based* scenes; therefore, no 'natural' candidates for comparison exist. Nevertheless, due to their impressive depth edges and their similar goal, we compare our method to [29] and [11] (BoostingDepth and Gradient-Fusion, respectively) and show their limitations (Sec. 4.4).

3. Method

The flow of our MDE training method can be described in three steps as illustrated in Fig. 2A-C. In the first step, the

DEE network is trained on the source synthetic dataset with the depth edges GT, to predict a probabilistic map of depth edges for a given RGB image and the corresponding LIDAR measurement (Fig. 2A and Sec. 3.5). In the second step, inference is applied on the training set of the target real data using the trained DEE network, resulting in (approximate) depth edge labels, $\tilde{E}_{GT}$ (Fig. 2B). In the last step we train the MDE network with an Edge Detection Block (EDB) to improve the localization of the depth edges (Fig. 2C and Sec. 3.2). The training is carried out using a straightforward supervised depth loss with LIDAR as GT, and the proposed edge loss with $\tilde{E}_{GT}$ as approximate GT. In the following sections we describe our method and the used steps in detail.

3.1. Monocular Depth Estimation

MDE models are commonly trained to regress per-pixel depth, $D(I)$, from an RGB image, I, given a depth GT, D_{GT}, in S scales, using a loss function:

$$\mathcal{L}_{depth}^{ms} = \frac{1}{S} \sum_s \mathcal{L}_{depth}(D^s(I), D_{GT}^s), \qquad (1)$$

where $D^s(I)$ and D_{GT}^s indicate the predicted depth and GT depth in scale s, respectively. For brevity we omit the scale indexes for the rest of the paper, although, unless explicitly stated, all losses are multi-scale.

3.2. Edge Detection Block (EDB)

To encourage the model to produce depth edges at the correct locations, we use a differentiable layer that computes the per-pixel probability of depth edges, $\tilde{E}(D(I))$, from the predicted depth map, $D(I)$, where I is the input RGB image. Given $D(I)$, the EDB computes the magnitude of the spatial image gradient, $|\nabla D(I)|$, and then transforms it into an 'edge-ness' probability score:

$$\tilde{E}(D(I)) = sigmoid(|\nabla D(I)| - t_{grad}) \qquad (2)$$

by shifting it with the paramater t_{grad} and passing it through a sigmoid function.

In practice, when using the standard image gradient

$$|\nabla D(I)| = \sqrt{\left(\frac{dD(I)}{dx}\right)^2 + \left(\frac{dD(I)}{dy}\right)^2},$$

some cyclic gradient patterns may emerge since both $dD(I)/dx$ and $dD(I)/dy$ are unconstrained (see depiction in the supplementary material). Therefore, we compute the gradient as a derivative in the direction perpendicular to the edge. To this end, the normal direction to the edge is first computed from the depth edge GT (estimated using the DEE network - Sec. 3.5), $\tilde{E}_{GT}$, by:

$$\theta = atan2\left(\frac{d\tilde{E}_{GT}}{dy}, \frac{d\tilde{E}_{GT}}{dx}\right)$$

The derivative in the direction perpendicular to the edge for every pixel (x, y) is therefore given by:

$$\nabla_\theta D_I(x, y) = D_I(x + \cos\theta, y + \sin\theta) - \\ D_I(x - \cos\theta, y - \sin\theta),$$

where $D_I = D(I)$ and the coordinates $x \pm \cos\theta$ and $y \pm \sin\theta$ are rounded in practice. The predicted edge probability, $\tilde{E}(D(I))$, is then used by the depth edges loss as described in the following section.

3.3. Depth Edges Loss

Given the depth edges GT, $\tilde{E}_{GT}$, and the output of the EDB, $\tilde{E}(D(I))$, we use the following loss to encourage depth discontinuity at $\tilde{E}_{GT}$:

$$\mathcal{L}_{edge}(\tilde{E}(D(I)), \tilde{E}_{GT}) = BBCE\left(\tilde{E}(D(I)), \tilde{E}_{GT}\right) \quad (3)$$

where the $BBCE$ [50] is the Balanced Binary Cross Entropy loss where the positives (edge pixels) and the negatives (non-edge pixels) are reweighted in a standard BCE loss so they have equal contribution.

3.4. Total Loss

Our MDE model is trained with a linear combination of the edge loss, $\mathcal{L}_{edge}$, and the standard depth loss, $\mathcal{L}_{depth}^{ms}$ (Eq. 1). The total loss is given by

$$\mathcal{L} = \mathcal{L}_{depth}^{ms}(D(I), D_{GT}) + \alpha\mathcal{L}_{edge}(\tilde{E}(D(I)), \tilde{E}_{GT}), \tag{4}$$

where α is a parameter to balance between the two losses, and D_{GT} is the depth GT.

3.5. Depth Edges Estimation (DEE)

Since the actual depth edges GT is unavailable for the target real data we use the DEE network, which predicts depth edges, $\tilde{E}(I, D')$, from RGB, I, and LIDAR, D'. We note that the LIDAR measurements have significant impact on the performance of the DEE network when given as input in addition to the RGB (ablation study is presented in the supplementary material). In order to train the DEE network we use a synthetic dataset with dense depth and LIDAR that are available for each RGB image. To extract depth edges GT, E_{GT}, we use the Canny edge detector [5] on the dense depth GT. The DEE network is trained with the depth edges loss as presented in Eq. 3; that is, $\mathcal{L}_{edge}(\tilde{E}(I, D'), E_{GT})$.

The predicted depth edges, $\tilde{E}(I, D')$, obtained from the DEE network are dense with Gaussian-like distributions. To produce one-pixel wide thin depth edges, we use two standard edge-detection post-processing operations [5]: (a) Non-Maximum Suppression (NMS), and (b) hysteresis (with 0.85 and 0.9 as low and high parameters).

4. Experiments

In the following we describe experiments that demonstrate the effectiveness of our suggested method and compare it to the SOTA and other relevant methods.

4.1. Datasets

To train the DEE network we use the GTA-PreSIL dataset [22], which is rendered from the Grand Theft Auto (GTA) video game. It consists of ∼44K images for training and ∼7K images for validation and testing, where each image has a corresponding dense depth GT. To generate the GT for the depth edges, we use Canny edge detector [5] with low and high thresholds of 4 and 5 meters, respectively. We note that the DEE network could have been trained with Virtual KITTI [13], probably achieving better performance on KITTI (details below) due to a smaller domain gap between the datasets. However, since we aim to demonstrate a general-purpose DEE network, GTA-PreSIL was the better choice since its structures and styles are significantly different from the target real domains.

We demonstrate our method on three different datasets: Synscapes [48], KITTI [43] and DDAD [20]. The data split for the highly realistic Synscapes dataset, where each image has a corresponding dense depth, is similar to [45]. We have chosen this dataset since (1) it provides perfect depth edges GT to evaluate our method, and (2) it has dense depth that allows us to fully analyze the impact of our method on the per-pixel depth accuracy. For KITTI we use the Eigen split [12], and for the DDAD dataset we use the official split.

KITTI & DDAD Depth Edges Evaluation Sets: To enable direct evaluation of the depth edges of MDE networks on the KITTI and DDAD datasets, we introduce the KITTI Depth Edges (KITTI-DE) and the DDAD Depth Edges (DDAD-DE) evaluation sets which consist of depth edges annotations of 102 images from the KITTI dataset and 50 images from DDAD dataset, respectively. To ease the manual annotation process, we started from the corresponding images of the Semantic and Instance Segmentation evaluation benchmark of KITTI [2] and the validation set of DDAD. To get an initial approximation of the depth edges we derive the edges of the instance segmentation masks for the relevant classes, and the semantic segmentation maps for the other classes. Then, false depth edges were manually removed (e.g. car wheels point of contact on the road), and missing depth edges were added (e.g., edges between building). See the supplementary material for additional details of the annotation process. We note that the objective of the proposed depth edges evaluation sets is to complement the standard depth evaluation sets, not to replace it, thus allowing to inspect another important aspect of MDE.
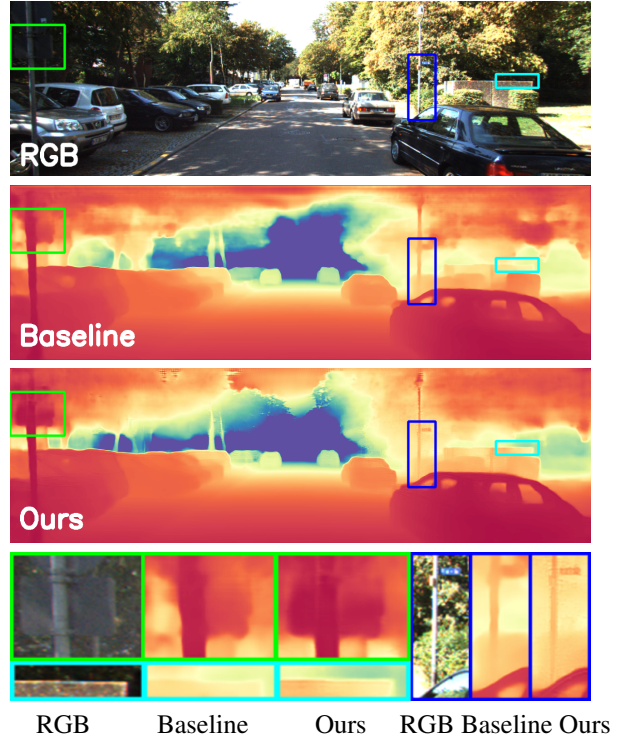

Figure 4. **Examples of depth predictions in the KITTI-DE dataset.** The depth predictions for the baseline and our method correspond to Packnet-SAN and Packnet-SAN+EL, respectively.

4.2. Metrics

To evaluate the performance of the MDE networks, we use some of the common per-pixel MDE depth metrics, as well as metrics to evaluate the quality of the edges of the predicted depth. Both depth and depth edges are evaluated in the bottom 60% of the image where LiDAR is commonly available, as defined in Garg et al. [14], for all datasets.

Depth metrics: We use the common Absolute Relative Error (ARE) to measure the per-pixel depth error, given by $ARE(\tilde{d}, d) = |\tilde{d} - d|/d$, where $\tilde{d}$ and d are the predicted depth and GT depth, respectively. Note that the ARE is computed only over the sparse LIDAR measurements in the KITTI and the DDAD datasets, which are very sparse near edges (Fig. 3), making the ARE a poor metric to measure the quality of the 2D localization of depth edges.

Depth edges metrics: To evaluate the edge quality of the predicted depth, one option is the ORD metric [49], which measures the percentage of order disagreements between pairs of depth points in the predicted depth against the GT depth. We argue that this metric suffers from a similar problem as the ARE - since LIDAR measurements are sparse, many possible 2D edges can obtain the same ORD value.

Therefore, in addition to the ORD metric, we use the most common metric from the edge detection literature, BSDS [28]. It consists of finding a bijective matching,

$\mu(\tilde{E}, E)$, between the edge pixels of the predicted depth, $\tilde{E}$, and the edge pixels of the GT, E. Each match $(\tilde{e}, e) \in \mu(\tilde{E}, E)$ between edge pixel, $\tilde{e} \in \tilde{E}$, and edge pixel GT, $e \in E$, is obtained such that the locations of the pixels are similar; that is, $||\tilde{e}, e||_2 \leq t_e$ where t_e is a small threshold (we use $t_e = 2$). The precision, $PR(\tilde{E}, E)$, and recall, $RE(\tilde{E}, E)$, are then given by $|\mu(\tilde{E}, E)|/|\tilde{E}|$, and $|\mu(\tilde{E}, E)|/|E|$, respectively.

In practice, $\tilde{E}$ is obtained by extracting edges from the predicted depth d using Canny edge detector with low and high thresholds of th_{low} and th_{high}, respectively. Since there is an inherent tradeoff between the precision and recall, we invoke multiple runs of a Canny edge detector with different parameters to obtain a precision-recall curve. Finally, to quantify the depth edges quality in a single number, we compute the Area Under the Curve (AUC). Since a large part of the graph is uncovered by all MDE algorithms, we report both the AUC for a partial representative range and for all range ([0,1]). Note that the edge metrics used in the I-BIMS1 dataset [24], ϵ_{DBE}^{acc} and ϵ_{DBE}^{comp}, define a matching between the predicted edges and the GT edges, similarly to our AUC metric. However, they suffer from a significant limitation in comparison to our AUC metric since their matching is not bijective and allows multiple-to-multiple matching. For example, broken and non-consecutive predicted edge can be perfectly matched to a consecutive GT edge, and yield the best score under these metrics.

4.3. Results

We use our method with three SOTA sparsely-supervised MDE methods as baselines: Packnet-SAN [18], AdaBins [4] and PixelFormer [1]. For all methods we use the publicly available code and weights and resume training for ten more epochs with our proposed loss (Sec. 3.3), and without it as a baseline for the comparison. In the following sections, we refer to the original methods by Packnet-SAN, AdaBins and PixelFormer, and refer to these methods with the addition of our edge loss (Sec. 3) as {method name} + EL (ours), where EL is abbreviation for 'Edge Loss'. We further show on the supplementary material that training simoultaneously (or sequentially) on both source (GTA-PreSIL) and target (KITTI or DDAD) datasets is inferior to our method. We note that, since Packnet-SAN exhibits better depth edges than both other methods, we use it as our main baseline in all experiments.

4.3.1 The KITTI dataset

In Fig. 4 we present an image from the KITTI-DE dataset alongside the depth predictions of the baselines and our method. It can be seen, for example on the crops in the bottom row, that our method has considerably more accurate depth edges than the baseline in many parts of the scene.

The precision-recall curves for the KITTI-DE dataset,

which are presented in the leftside of Fig. 5, are computed for the baselines and competing methods (solid curves) and for our method with the edge loss (non-red dashed curves). The precision-recall curves of our method in comparison to the corresponding baselines, when compared using the same precision (same x coordinate), have a significantly higher recall. In Tab. 1, the AUC metric for the edge quality is presented, where our method achieves an AUC of 61.87%, 53.47% and 46.23% in comparison to the Packnet-SAN, AdaBins and PixelFormer baselines that achieve an AUC of 47.56%, 41.23% and 32.79%, respectively, indicating a relative improvement between 30% and 40%.

The quality of the per-pixel depth is also evaluated on the KITTI-DE dataset, and the ARE is presented in Tab. 1. The Packnet-SAN, AdaBins and PixelFormer baselines achieve an ARE of 3.45%, 3.14% and 3.00%, respectively, in comparison to our Packnet-SAN + EL, AdaBins + EL and PixelFormer + EL that achieve an ARE of 3.61%, 3.11% and 2.94%, respectively, which indicates a comparable performance. We also evaluate the methods on the standard KITTI test set (Tab. 1), where only the per-pixel depth quality can be estimated. The Packnet-SAN, AdaBins and PixelFormer baselines achieve an ARE of 6.17%, 6.28% and 5.46%, respectively, in comparison to our Packnet-SAN + EL, AdaBins + EL and PixelFormer + EL that achieve an ARE of 6.50%, 6.21% and 5.59%, respectively, which also indicates a comparable performance. We note that, interestingly, the quality of edges is not highly correlative to the per-pixel depth quality, which demonstrates the need for the depth edges annotated datasets we propose.

4.3.2 The DDAD dataset

In Fig. 6 we present an image from the DDAD-DE dataset, alongside the depth predictions of Packnet-SAN and our method with Packnet-SAN as baseline. Furthermore, the depth edges (extracted similarly to KITTI), which are laid on top of the RGB images, are presented in the top row. It can be seen that our method has considerably more accurate edges that fit more tightly on objects' silhouette in comparison to the baseline. Moreover, some missing edges are added and some clear false positives are discarded in our method in comparison to the baseline.

The precision-recall curves for the DDAD-DE dataset are presented in the right side of Fig. 5, where our method, when compared with the same precision (same x coordinate), has significantly higher recall than the Packnet-SAN baseline. In Tab. 2, the AUC metric for the edge quality is presented, where our method achieves an AUC of 48.32% in comparison to Packnet-SAN that achieves an AUC of 31.52%, indicating around 50% of relative improvement.

The quality of the per-pixel depth is also evaluated on the DDAD-DE dataset, and the ARE is presented in Tab. 2. The Packnet-SAN baseline and our methods achieve an ARE of
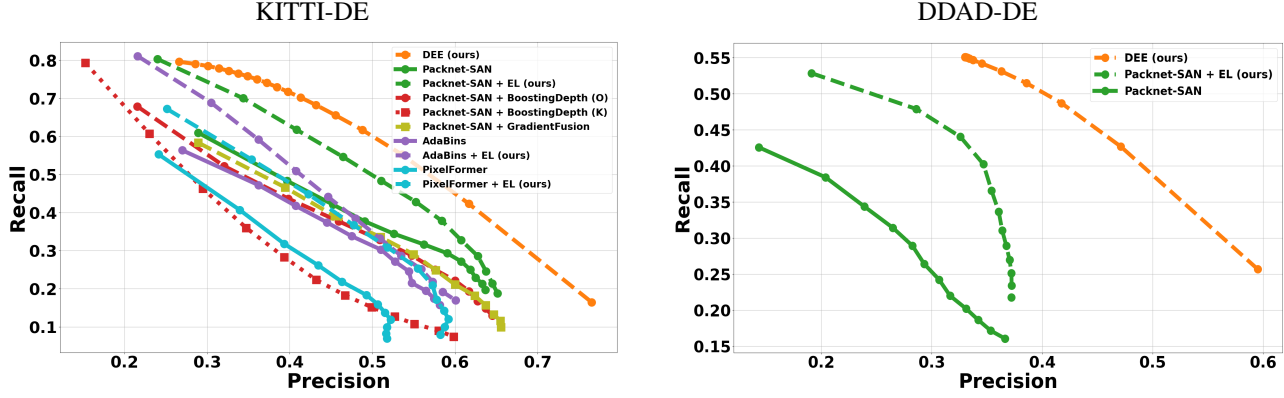
Figure 5. **Precision and recall of the depth edges on the KITTI-DE and DDAD-DE evaluation sets**. Each point on the graphs of the MDE methods is generated with different parameters of the Canny edge detector. Each of the points on the graphs that correspond to the DEE method is generated by thresholding the depth edge probability in the range $(0, 1)$.

Method	KITTI-DE				KITTI test		
	AUC (edges) ↑	ORD ↓	ARE ↓	$\delta < 1.25$ ↑	ORD ↓	ARE ↓	$\delta < 1.25$ ↑
Packnet-SAN	47.56% (39.40%)	7.68%	3.45%	98.66%	12.40%	6.17%	95.39%
Packnet-SAN + BoostingDepth (O)	46.04% (37.07%)	10.35%	9.32%	88.90%	12.63%	11.10%	86.41%
Packnet-SAN + BoostingDepth (K)	36.19% (31.27%)	9.47%	7.24%	93.62%	11.45%	8.33%	91.99%
Packnet-SAN + GradientFusion	44.51% (34.10%)	9.18%	5.93%	95.66%	11.15%	7.18%	94.17%
Packnet-SAN + EL (ours)	**61.87% (49.02%)**	7.75%	3.61%	98.53%	12.48%	6.50%	95.06%
AdaBins	41.23% (34.11%)	7.69%	3.14%	98.78%	10.14%	6.28%	95.85%
AdaBins + EL (ours)	**53.47% (44.00%)**	7.64%	3.11%	98.79%	10.13%	6.21%	95.87%
PixelFormer	32.79% (26.44%)	7.47%	3.00%	98.79%	7.56%	5.45%	96.98%
PixelFormer + EL (ours)	**46.23% (35.33%)**	7.53%	2.94%	98.80%	7.58%	5.59%	96.72%

Table 1. **Results on the KITTI dataset.** The AUC is given for the range where at least one MDE method has valid measurement: [0.12,0.65]. In parentheses we also report the AUC of the full [0,1] range. In BoostingDepth, O is for the original training (dense data) by the authors, and K is for our training (KITTI data).

8.89% and 8.99%, respectively, which indicates a comparable performance. On the standard DDAD evaluation set (Tab. 2), the ARE of the Packnet-SAN and our method are 9.49% and 10.0%, respectively, indicating, a 5% decrease in per-pixel depth accuracy of our method.

4.3.3 The Synscapes dataset

We present the results of an additional experiment on the Synscapes dataset in Tab. 2 and in the supp. material, where we show that an increase in the depth edges accuracy yields an increase in the per-pixel depth accuracy (e.g., ARE) for dense depth GT. It further suggests that the slight decrease in the per-pixel depth accuracy, which is sometimes observed in KITTI and DDAD, might be due to the sparseness of their depth GT, especially near depth edges.

4.4. Comparison to Merging Methods [11, 29]

To the best of our knowledge there are no other methods to improve the depth edges of sparsely-supervised MDE methods; however, BoostingDepth [29] and GradientFusion [11]

share a similar aim to ours, even though these methods were designed for indoor scenes with dense GT. To this end, we use the predictions of Packnet-SAN with BoostingDepth's original depth merger for comparison with our method. Additionally, we train the depth merger on the KITTI dataset for a fair comparison (see details in the supplementary material). The fusion-net in GradientFusion was trained solely on the high-res HR-WSI dataset [49]. Their method is based on a preprocessing of HR-WSI's images with a guided filter to produce high and low resolutions that allow the self-supervised training. We therefore use their original fusion-net that was used for all datasets and MDE methods. Two experiments are presented: one with LeRes [52], which was the main backbone for GradientFusion (see supp. material), and one with Packnet-SAN.

For both BoostingDepth (with the original weights) and GradientFusion, the AUC metric presented in Tab. 1 and Fig. 5 shows a slightly worse performance than the Packnet-SAN baseline, and a significantly worse performance than Packnet-SAN with our method. One possible reason for the
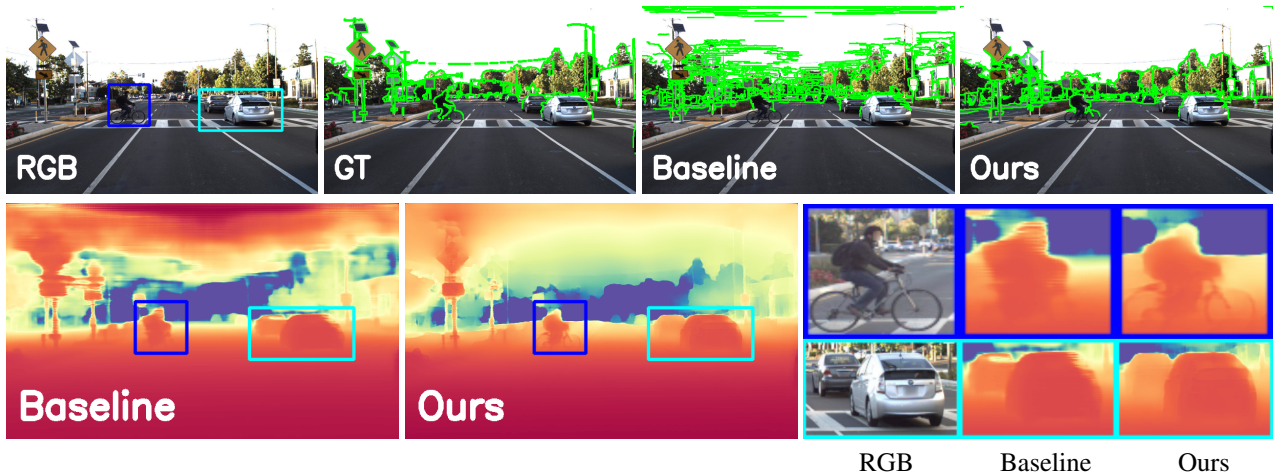
Figure 6. **Examples of depth predictions of Packnet-SAN and Packnet-SAN + EL (ours) on images from the DDAD-DE dataset.**
The top row, from left to right: RGB, depth edges GT (manually annotated), depth edges extracted (using Canny) from the baseline and our depth, respectively. The bottom row, from left to right: the predicted depth of the baseline and our method, respectively, followed by two zoom-in crops with the RGB, baseline and our predicted depth, respectively.

Method	DDAD-DE				DDAD test			Synscapes			
	AUC (edges) ↑	ORD ↓	ARE ↓	δ_1 ↑	ORD	ARE	δ_1	AUC	ORD	ARE	δ_1
Packnet-SAN	31.52% (23.32%)	8.03%	8.89%	91.6%	8.95%	9.49%	90.7%	61.17%	30.8%	5.43%	96.2%
Packnet-SAN + EL (ours)	**48.32% (32.29%)**	8.38%	8.99%	91.4%	9.43%	10.0%	89.5%	**65.38%**	21.1%	4.85%	96.5%

Table 2. **Results on the DDAD and Synscapes datasets.** The AUC (for DDAD) is given for the range where at least one MDE method has valid measurement: [0.14,0.37]. In parentheses we also report the AUC of the full [0,1] range.

performance is probably due to the domain gap between the datasets used to train the depth mergers and KITTI. Additional reasons for BoostingDepth might be: (i) The depth merger can process only square images, which deviates significantly from KITTI's aspect ratio, and (ii) It is highly sensitive to the network's Receptive Field (RF), where Packnet-SAN has a large RF of 1028, close to the size of the image. The ARE performance is significantly worse than both other methods, probably due to the usage of the GAN-based pix2pix [23] to train the depth merger which is known to create results that 'look' realistic but not necessarily accurate. The performance of GradientFusion for KITTI is low since self-supervision using the guided filter strongly depends on image resolution (as stated in their paper also), which is low in datasets like KITTI.

BoostingDepth with the depth merger that was trained on KITTI presents worse edges (AUC) than the original weights, but better ARE. We hypothesize that the basic assumption of BoostingDepth, in which an MDE network produces more fine details in high resolution (but worse overall shape) breaks for LIDAR-based scenes due to the lack of depth GT near depth edges. See supp. material for a more detailed discussion, visual results and training details.

4.5. Application: AR occlusions

We present an experiment in rendering virtual objects for AR applications. We use the predicted depth of the Packnet-SAN baseline and our method to compute the occluded parts of the virtual objects from the point of view of the camera. In Fig. 1b three animated characters are planted in the scene. To observe the difference in the depth edges accuracy, zoom-in on the sides of the vehicle and on the 'no-entry' post. The occlusions of the character in our method seems much more realistic than the baseline.

5. Conclusion

We have presented a method to improve the localization of the depth edges in sparsely-supervised MDE methods, while preserving the per-pixel depth accuracy. The method is based on the observation that detecting the location of the depth edges can be learned effectively from synthetic data. To evaluate our method on real data, we introduced two depth edges evaluation sets for the KITTI and the DDAD datasets by manual annotation. The proposed method can be further improved by considering the images (and LIDAR) of the target dataset (e.g., KITTI) in the training procedure, possibly using methods from the unsupervised domain adaptation literature.

References

[1] Ashutosh Agarwal and Chetan Arora. Attention attention everywhere: Monocular depth prediction with skip attention. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5861–5870, 2023. 3, 6

[2] Hassan Alhaija, Siva Mustikovela, Lars Mescheder, Andreas Geiger, and Carsten Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *International Journal of Computer Vision (IJCV)*, 2018. 5

[3] Ali Jahani Amiri, Shing Yan Loo, and Hong Zhang. Semi-supervised monocular depth estimation with left-right consistency using deep neural network. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 602–607. IEEE, 2019. 3

[4] 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*, pages 4009–4018, 2021. 3, 6

[5] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986. 4, 5

[6] Ang Cao, Chris Rockwell, and Justin Johnson. Fwd: Real-time novel view synthesis with forward warping and depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15713–15724, 2022. 1, 2

[7] Chuangrong Chen, Xiaozhi Chen, and Hui Cheng. On the over-smoothing problem of cnn based disparity estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8997–9005, 2019. 2

[8] Xingyu Chen, Ruonan Zhang, Ji Jiang, Yan Wang, Ge Li, and Thomas H Li. Self-supervised monocular depth estimation: Solving the edge-fattening problem. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 5776–5786, 2023. 3

[9] Xuelian Cheng, Yiran Zhong, Yuchao Dai, Pan Ji, and Hong-dong Li. Noise-aware unsupervised deep lidar-stereo fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6339–6348, 2019. 2

[10] Diogo Santos Ortiz Correa, Diego Fernando Sciotti, Marcos Gomes Prado, Daniel Oliva Sales, Denis Fernando Wolf, and Fernando Santos Osório. Mobile robots navigation in indoor environments using kinect sensor. In *2012 Second Brazilian Conference on Critical Embedded Systems*, pages 36–41. IEEE, 2012. 1

[11] Yaqiao Dai, Renjiao Yi, Chenyang Zhu, Hongjun He, and Kai Xu. Multi-resolution monocular depth map fusion by self-supervised gradient-based composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 488–496, 2023. 3, 7

[12] 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. 5

[13] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4340–4349, 2016. 5

[14] Ravi Garg, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European conference on computer vision*, pages 740–756. Springer, 2016. 5

[15] 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*, pages 3354–3361. IEEE, 2012. 1, 2

[16] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019. 3

[17] Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8977–8986, 2019. 3

[18] Vitor Guizilini, Rares Ambrus, Wolfram Burgard, and Adrien Gaidon. Sparse auxiliary networks for unified monocular depth prediction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11078–11088, 2021. 1, 3, 6

[19] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2485–2494, 2020. 3

[20] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 5

[21] Akhil Gurram, Ahmet Faruk Tuna, Fengyi Shen, Onay Urfalioglu, and Antonio M López. Monocular depth estimation through virtual-world supervision and real-world sfm self-supervision. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 3

[22] Braden Hurl, Krzysztof Czarnecki, and Steven Waslander. Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2522–2529. IEEE, 2019. 2, 5

[23] 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*, pages 1125–1134, 2017. 8

[24] Tobias Koch, Lukas Liebel, Friedrich Fraundorfer, and Marco Korner. Evaluation of cnn-based single-image depth estimation methods. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018. 3, 6

[25] Yevhen Kuznietsov, Jorg Stuckler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6647–6655, 2017. 3

[26] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and

Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019. 3

[27] Marcio C de F Macedo and Antonio L Apolinario. Occlusion handling in augmented reality: Past, present and future. *IEEE Transactions on Visualization and Computer Graphics*, 2021. 1, 2

[28] David R Martin, Charless C Fowlkes, and Jitendra Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE transactions on pattern analysis and machine intelligence*, 26(5):530–549, 2004. 5

[29] S Mahdi H Miangoleh, Sebastian Dille, Long Mai, Sylvain Paris, and Yagiz Aksoy. Boosting monocular depth estimation models to high-resolution via content-adaptive multi-resolution merging. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9685–9694, 2021. 1, 3, 7

[30] Chao Ning and Hongping Gan. Trap attention: Monocular depth estimation with manual traps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5033–5043, 2023. 3

[31] Luigi Piccinelli, Christos Sakaridis, and Fisher Yu. idisc: Internal discretization for monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21477–21487, 2023. 3

[32] Koutilya PNVR, Hao Zhou, and David Jacobs. Sharingan: Combining synthetic and real data for unsupervised geometry estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13974–13983, 2020. 2

[33] 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*, pages 14648–14657, 2020. 3

[34] Michael Ramamonjisoa and Vincent Lepetit. Sharpnet: Fast and accurate recovery of occluding contours in monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019. 3

[35] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12179–12188, 2021. 3

[36] 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*, 2020. 3

[37] Faraz Saeedan and Stefan Roth. Boosting monocular depth with panoptic segmentation maps. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3853–3862, 2021. 3

[38] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3752–3761, 2018. 2

[39] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German conference on pattern recognition*, pages 31–42. Springer, 2014. 3

[40] Thomas Schops, Johannes L Schonberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3260–3269, 2017. 3

[41] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pages 746–760. Springer, 2012. 1, 3

[42] Minsoo Song, Seokjae Lim, and Wonjun Kim. Monocular depth estimation using laplacian pyramid-based depth residuals. *IEEE transactions on circuits and systems for video technology*, 31(11):4381–4393, 2021. 3

[43] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant cnns. In *International Conference on 3D Vision (3DV)*, 2017. 5

[44] Igor Vasiljevic, Nick Kolkin, Shanyi Zhang, Ruotian Luo, Haochen Wang, Falcon Z Dai, Andrea F Daniele, Mohammadreza Mostajabi, Steven Basart, Matthew R Walter, et al. Diode: A dense indoor and outdoor depth dataset. *arXiv preprint arXiv:1908.00463*, 2019. 3

[45] Yufeng Wang, Yi-Hsuan Tsai, Wei-Chih Hung, Wenrui Ding, Shuo Liu, and Ming-Hsuan Yang. Semi-supervised multi-task learning for semantics and depth. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2505–2514, 2022. 5

[46] Jamie Watson, Oisin Mac Aodha, Victor Prisacariu, Gabriel Brostow, and Michael Firman. The temporal opportunist: Self-supervised multi-frame monocular depth. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1164–1174, 2021. 3

[47] Felix Wimbauer, Nan Yang, Lukas Von Stumberg, Niclas Zeller, and Daniel Cremers. Monorec: Semi-supervised dense reconstruction in dynamic environments from a single moving camera. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6112–6122, 2021. 3

[48] Magnus Wrenninge and Jonas Unger. Synscapes: A photorealistic synthetic dataset for street scene parsing. *arXiv preprint arXiv:1810.08705*, 2018. 5

[49] Ke Xian, Jianming Zhang, Oliver Wang, Long Mai, Zhe Lin, and Zhiguo Cao. Structure-guided ranking loss for single image depth prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 611–620, 2020. 1, 3, 5, 7

[50] Saining Xie and Zhuowen Tu. Holistically-nested edge detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1395–1403, 2015. 4

[51] Yan Xu, Xinge Zhu, Jianping Shi, Guofeng Zhang, Hujun Bao, and Hongsheng Li. Depth completion from sparse lidar data with depth-normal constraints. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2811–2820, 2019. 2

[52] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. Learning to recover 3d scene shape from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 204–213, 2021. 7

[53] Zixiang Zhao, Jiangshe Zhang, Shuang Xu, Zudi Lin, and Hanspeter Pfister. Discrete cosine transform network for guided depth map super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5697–5707, 2022. 3

[54] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1851–1858, 2017. 3

[55] 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*, pages 13116–13125, 2020. 1, 3