Abstract

Depth prediction is at the core of several computer vision applications, such as autonomous driving and robotics. It is often formulated as a regression task in which depth values are estimated through network layers. Unfortunately, the distribution of values on depth maps is seldom explored. Therefore, this paper proposes a novel framework combining contrastive learning and depth prediction, allowing us to pay more attention to depth distribution and consequently enabling improvements to the overall estimation process. Purposely, we propose a window-based contrastive learning module, which partitions the feature maps into non-overlapping windows and constructs contrastive loss within each one. Forming and sorting positive and negative pairs, then enlarging the gap between the two in the representation space, constraints depth distribution to fit the feature of the depth map. Experiments on KITTI and NYU datasets demonstrate the effectiveness of our framework.

1. Introduction

Depth prediction aims at recovering the 3D structure of a scene and is a crucial task at the core of various computer vision applications, such as robot/vehicle navigation and augmented reality, to name a few. Methods for dense depth prediction can be divided into two main classes: active and passive techniques. Representatives of the former are LiDAR, structured light, and time-of-flight sensors. They perturb the sensed environment with a signal, according to different technologies, and measure its behavior or appearance to infer the depth of the sensed scene. For instance, LiDAR and Time-of-flight measure the traveling time a signal needs from the emitter to the receiver, both located in the sensing device. However, due to their intrinsic limitations, they generally generate sparse/low-resolution measurements. Thus, various approaches to densify such data have been proposed, typically in setups coupling such sensors with high-resolution RGB images [11, 18, 27, 29, 38, 47, 77, 81]. On the other hand, deep learning enabled pure passive sensing techniques to rely on single images [17, 21, 36] without any auxiliary aid to achieve the outlined goal despite being very challenging due to its ill-posed nature [52].

Regardless of the adopted setup, little effort in the literature focuses on analyzing the distribution of depth data in a statistical manner, as we propose in this paper. In the real world, the depth changes smoothly within adjacent pixels belonging to the same portion of the object, while this is not always the case for depth predicted by neural networks. Best viewed in color.

Figure 1. Imaging system and depth prediction process. The depth changes smoothly within adjacent pixels belonging to the same portion of the object, while this is not always the case for depth predicted by neural networks. Best viewed in color.
ideal depth maps for two smooth surfaces and one discontinuity – i.e. common structures in the real world – and construct their depth histogram, as shown in Fig. 3. We can observe that the distributions of depth values in the histograms are much more regular. Therefore, we argue that regularizing such distributions can yield higher-quality depth maps. Accordingly, by focusing on the structure of objects from a microscopic perspective, any object can be seen as the composition of several small, smooth surfaces interleaved by depth discontinuities, whose depth distributions can be regularized.

Thus, we inquire whether a learning method can regularize the distribution of the depth predictions by a deep network. The recent contrastive learning approaches [4, 9, 24] proposed to contrast positive pairs – i.e., elements sharing the same label – against negative pairs – elements with different labels – in the representations space, which looks suitable for our purpose. Therefore, we introduce a contrastive learning framework tailored for depth prediction. Specifically, we propose a Window-based Contrastive Learning module (WCL) which segments the depth feature maps into non-overlapping windows and computes a contrastive loss only within each one, similarly to how recent Vision Transformers [44] compute self-attention on local windows. It allows us, in a more tractable manner rather than acting on the whole depth features, to contrast the depth values in small regions and expand their distribution locally by constructing positive and negative pairs in the depth representation and enlarging the gap between them. To the best of our knowledge, we are the first to apply contrastive learning for depth prediction, focusing on expanding depth distribution.

Our main contributions can be summarized as follows:

- We propose a novel method combining contrastive learning with depth prediction, contrasting depth distribution to improve accuracy.
- We propose a Window-based Contrastive Learning (WCL) module for depth prediction by learning similarity, forming positive and negative pairs of features, and computing contrastive loss within a window.
- Experimental results and a detailed ablation study with different depth prediction models demonstrate the effectiveness of our proposal.

2. Related Work

2.1. Depth prediction

Depth prediction is a challenging computer vision task, faced according to different methodologies and setups such as depth completion, depth estimation, and self-supervised depth estimation with stereo pairs and videos. Depth completion recovers dense depth maps from sparse measurements and a high-resolution image. Uhrig et al. [62] propose a sparsity-invariant convolution layer to consider the location of missing data while addressing data sparsity within deep networks. Ma et al. [45, 46] utilize early fusion to combine sparse depth with a color image and feed them into an encoder-decoder CNN, which boosts the performance of depth completion. Multi-branch network architecture is an effective approach to fuse multi-modal data as reported in [18, 32, 38, 55, 63]. Spatial propagation networks (SPN) [43] is another popular depth-refinement approach [10–12, 42, 48]. DeepLiDAR [55] introduces pixel-
wise surface normals as geometric constraints and proposes multiple branches to generate dense depth maps jointly. GuideFormer [56] and CompletionFormer [78] introduce transformers in this task as well. In [7, 73, 81], graph representation is utilized to model more helpful information from sparse and irregular point clouds.

Regarding depth estimation, supervised monocular approaches gained much interest in the literature in recent years. Eigen et al. [16, 17] proposed a multi-stage, coarse-to-fine network to estimate depth from a single image, and Laina et al. [35] a fully convolutional architecture for depth estimation. Some works introduce attention mechanisms to achieve significant performance improvements [1, 2, 37, 39]. Other works estimate depth by predicting probability distributions and discrete bins [5, 40, 59]. A high-order geometric constraint is also employed to reconstruct depth prediction in [36, 74]. Some multi-task works predict depth maps by jointly learning with other vision tasks [41, 54, 79]. A high-order geometric constraint is also employed to reconstruct depth prediction in [36, 74]. Some multi-task works predict depth maps by jointly learning with other vision tasks [41, 54, 79].

DANet [59] proposes to utilize depth distribution as supervision for the prediction.


### 2.2. Contrastive learning

Contrastive learning has achieved remarkable progress employing discriminative learning by contrasting positive pairs against negative pairs in representations space [23] and some works target visual representation learning [9, 15, 24, 33, 69]. SimCLR [9] implements contrastive learning in a simple network framework, where positive pairs are from data augmentation of the same image, while negative ones are from different images. MoCo [24] maintains a queue of negative samples and turns one branch of a Siamese network into a momentum encoder to improve the queue consistency. Some recent works [6, 68, 72] have introduced contrastive learning for dense prediction. ReSim [71] learns regional representations from a pair of views originating by sliding windows from the same image. DenseCL [68] optimizes a pairwise contrastive loss at the pixel level between two different image views. Ke et al. [31] propose a weakly supervised segmentation method that utilizes contrastive relationships between pixels and segments in the feature space. Alonso et al. [3] utilize a memory bank to contrast the features from labeled and unlabeled data employing end-to-end training. Some works [70, 76] use a contrastive loss to generate high-frequency details for image super-resolution tasks. Shen et al. [58] propose contrastive differential learning in image translation and use it for depth-to-depth synthesis.

### 2.3. Window-based Approaches

Long-range dependency is a notable cue and Transformers [64], Markov Random Fields (MRFs) [57] and Conditional Random Fields (CRFs) [8, 67] use it to boost their learning ability. However, these methods have a severe drawback in the computational complexity, increasing quadratically with image size. Purposely, some works aim at addressing this issue. Dosovitskiy et al. [14] apply a transformer on sequences of image patches for image classification tasks. Pyramid ViT [65] uses a progressive shrinking pyramid and spatial-reduction attention to reduce computations of the transformer on large feature maps. Swin Transformer [44] proposes a novel architec-
contrastive loss in depth prediction tasks are challenging, and some of the main issues about deploying a different representations and depth labels. Indeed, distinguishing even pixels from the same object category may have the target task is semantic segmentation. However, this assumption consists in regularizing such distribution. For this purpose, applying contrastive learning by clustering the different pixels in the representation space seems a promising strategy for possibly regularizing the depth distribution. Moreover, it has been recently applied in an unsupervised manner [6,9,24,68,72], thus not making use of labels. Some recent works [6,26,31,66] focus on dense predictions, such as low-level image similarity and feature affinity in weakly supervised segmentation. Chaitanya et al. [6] use contrastive learning at the level of local and global features. All these methods assume that pixels belonging to the same object share the same representation and label since the two are close in distance and representation and the set (A, D) as a negative pair being far in distance and representation. However, for the set (B, C), it is hard to define whether they are positive or negative pairs since they are close in distance but belong to different objects.

- Constructing and computing a global relationship graph for all pixels is highly resource-consuming. For instance, if we consider an $H \times W$ image, its relationship map results in size $HW \times HW$, thus forming positive and negative pairs and contrasting them yields massive memory footprints, computation, and time.

- It is hard to contrast further the long-range sets, such as set (A, D), even in low-quality maps since there is already a significant distance between the two points.

For the reasons outlined, employing contrastive learning on the whole image is challenging. Purposely, we introduce the concept of local similarity in our method. As already pointed out, the output depth of a deep network changes smoothly within a small region. Therefore, a pixel has more similarities with its surrounding pixels. The closer the two pixels are, the stronger the similarity. For instance, the similarity between A and B is stronger than the similarity between A and others in Fig. 4. To deal with these issues, inspired by some works [13, 44, 75], we propose for depth prediction a cost-effective window-based approach for contrastive learning. We limit the similarity calculation and construct the positive and negative pairs within small regions rather than on the entire feature map, which is a more reasonable and resource-saving strategy. Thus, we set pairs with strong similarity as positive pairs and ones with weak similarity as negative pairs. Enlarging the gap between them also makes depth distribution better regularized and yields more accurate results.

3. Contrastive learning for Depth Prediction

In this section, we first present the motivation for our work and how WCL can improve depth prediction.

3.1. Motivation

In the imaging system, the image is a perspective projection of a 3D scene, and the corresponding depth map reflects the distance between each point in the scene and the viewpoint. Almost all surfaces of observed objects have discrete depth values and adjacent pixels, even within small regions, have similar but different depth values. As shown in Fig. 2, the histograms of different depth images with different accuracy change dramatically, and high-quality depth maps have a more regular depth distribution than low-quality ones. Hence, an intuitive idea to model this assumption consists in regularizing such distribution. For this purpose, applying contrastive learning by clustering the different pixels in the representation space seems a promising strategy for possibly regularizing the depth distribution. Moreover, it has been recently applied in an unsupervised manner [6,9,24,68,72], thus not making use of labels. Some recent works [6,26,31,66] focus on dense predictions, such as low-level image similarity and feature affinity in weakly supervised segmentation. Chaitanya et al. [6] use contrastive learning at the level of local and global features. All these methods assume that pixels belonging to the same object share the same representation and label since the target task is semantic segmentation. However, this assumption does not hold when facing depth prediction tasks, since even pixels from the same object category may have different representations and depth labels. Indeed, distinguishing the positive and negative pairs in the depth map is challenging, and some of the main issues about deploying a contrastive loss in depth prediction tasks are:

- Discriminating between positive and negative examples for all pixels is challenging. For example, in Fig. 4, we can easily define the set (A, B) as a positive pair since the two are close in distance and representation and the set (A, D) as a negative pair being far in distance and representation. However, for the set (B, C), it is hard to define whether they are positive or negative pairs since they are close in distance but belong to different objects.

The similarity map $T \in \mathbb{R}^{N^2 \times N^2}$ between each element...
Our module segments one $H \times W \times C$ feature map into multiple tiles of the same size, each containing $N \times N$ elements. Then, it computes a similarity map and constructs positive and negative pairs within the window. By enlarging the gap between positive and negative pairs, the features representation becomes more meaningful of the depth distribution in the scene.

In the window, the similarity is computed by means of dot product and normalization as

$$T = \frac{Q^T K}{\|Q\| \|K\|}$$  \hfill (2)

We employ the exponential function $\exp()$ to make the similarity map non-negative. Then, we sort $T$ in descending order:

$$T' = \text{sort}(T)$$  \hfill (3)

and use contrastive learning to enlarge the gap between positive and negative pairs in the representation space. We sample the first $N_1$ of $T'$ as positive pairs $P_T$, and the last $N_1$ negative pairs as $N_T$ to compute the contrastive loss $L_{\text{loss}}$ as

$$L_{\text{loss}} = \frac{1}{N_w} \sum_{i=1}^{N_w} - \log \frac{\sum_{j=1}^{N_1} P_T/N_1}{\sum_{j=1}^{N_1} P_N/N_1} + a$$  \hfill (4)

with $a$ being a constant set to 1; $N_1$ being set to the 20% of the total $N^2$; $N_w$ being the number of windows. Accordingly, the total loss function of a depth prediction network employing window-based contrastive learning is defined as:

$$L_{\text{total}} = L_{\text{depth}} + w \ast L_{\text{loss}}$$  \hfill (5)

with $L_{\text{depth}}$ the original depth loss from the depth prediction network, and $w$ a weighting term for the contrastive loss.

Our WCL module is specifically designed for depth prediction tasks and can be seamlessly integrated into many networks. As shown in Fig. 6, the WCL block works on feature maps between two layers. This way, the module enables the contrast between the positive and negative pairs in the representation space.

4. Experiments

In this section, we evaluate our method on three main depth prediction tasks, i.e., depth completion, monocular depth estimation, and self-supervised monocular depth estimation. We start from existing networks, assumed as baselines over which we want to improve. We retrain them both with and without our module, allowing for a fair comparison under the same experimental setting (i.e., data, hardware support). This comes with little effort since our WCL module is a plug-and-play component easily embeddable in any depth prediction architecture. All the experiments are conducted using the PyTorch framework, on a single NVIDIA RTX 3090 GPU.

4.1. Dataset

KITTI dataset. KITTI [20, 62] is a popular outdoor dataset providing sparse depth maps captured by Velodyne LiDAR HDL-64e, color images and corresponding semi-dense ground truth. The sparse depth maps provide 5.9% valid depth values on all pixels, while the ground truth maps contain 16% valid depth values over the whole image. The dataset contains 85 895 training frames, with 1000 more selected validation frames, as well as 1000 and 500 test samples for which ground truth is withheld, respectively, for depth completion and depth prediction tasks.

The Eigen split [17] is the standard portion of KITTI used for evaluating self-supervised monocular depth prediction, after being filtered [21, 82] of static frames, which are unsuited for training from videos [82]. In this split, 39 910 images and 4 424 images are used for training and valida-
### Table 1. Quantitative results on KITTI [62] – Depth Completion. Comparison between MSG-CHN and its WCL counterpart.

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE (mm)</th>
<th>MAE (mm)</th>
<th>iRMSE (1/km)</th>
<th>iMAE (1/km)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSG-CHN</td>
<td>820.145</td>
<td>223.987</td>
<td>2.425</td>
<td>0.979</td>
</tr>
<tr>
<td>MSG-CHN+WCL</td>
<td>810.273</td>
<td>222.719</td>
<td>2.428</td>
<td>0.973</td>
</tr>
</tbody>
</table>

### Table 2. Quantitative results on NYUv2 [61] – Depth Completion. Comparison between Sparse-to-Dense and its WCL counterpart.

<table>
<thead>
<tr>
<th>Method</th>
<th>RMSE (mm)</th>
<th>REL</th>
<th>$\delta_{1.25^1}$</th>
<th>$\delta_{1.25^2}$</th>
<th>$\delta_{1.25^3}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sparse-to-Dense [45]</td>
<td>0.1097</td>
<td>0.0185</td>
<td>99.39</td>
<td>99.91</td>
<td>99.98</td>
</tr>
<tr>
<td>Sparse-to-Dense+WCL</td>
<td>0.1038</td>
<td>0.0149</td>
<td>99.41</td>
<td>99.92</td>
<td>99.99</td>
</tr>
</tbody>
</table>

### 4.2. Evaluation Metrics

Following [21,36,38,48], we use common metrics in the field: Mean absolute error (MAE, mm), Root Mean Squared Error (RMSE, mm), Mean Absolute Error of the inverse depth (iMAE, 1/km), Root Mean Squared Error of the inverse depth (iRMSE, 1/km), Mean Absolute Relative Error (REL), Absolute relative difference (Abs Rel), Square relative difference (Sq Rel) and percentage of predicted pixels where the relative error is within a threshold ($\delta_i$).

### 4.3. Experimental Results

We select a set of models representative of the three specific tasks, to which we apply our method to improve their accuracy consistently. Any model and its WCL variant are trained using the standard hyper-parameters reported in the original papers.

#### 4.3.1 Depth completion

We consider two depth completion methods [38, 45], respectively MSG-CHN and Sparse-to-Dense. MSG-CHN [38] is a multi-branch guided cascade hourglass network for depth completion. Sparse-to-Dense [45] is built with an early-fusion network for multi-modal data. We insert our module between the last two layers of each branch in MSG-CHN and train it on the KITTI dataset [62]. In Sparse-to-Dense, we use ResNet18 [25] as the backbone to extract features, we insert our module between the last two layers of the decoder and train it on NYU Depth v2 [61]. We use the same training parameters, except for the batch size, for both networks. The batch size is set to 8 and 12, respectively. For the two networks, we set $w = 0.2$, $N = 7$ and $w = 0.1$, $N = 7$ respectively. Tab. 1 and Tab. 2 show that both MSG-CHN and Sparse-to-Dense can benefit from our WCL module. Fig. 7 shows a qualitative comparison between MSG-CHN and its counterpart using WCL on a sample from the KITTI dataset. We can notice how our module allows for more precise boundaries at depth discontinuities. Fig. 8 instead compares Sparse-to-Dense models on the NYUv2 dataset, highlighting the same behavior observed for MSG-CHN.

#### 4.3.2 Monocular Depth estimation

For this task, we select BTS [36] as a baseline. It is a state-of-the-art method using local planar guidance layers as geometric constraints to guide the features to depth upsampling in the decoding phase. We use BTS variants with different backbones, i.e., ResNet50 [25] and DenseNet-121 [28], on NYUv2 for the monocular depth estimation task. We use
Figure 8. **Qualitative comparison on the NYUv2 depth completion dataset** [61]. From left to right, top to bottom, RGB image, ground truth, results of Sparse-to-Dense [45] and Sparse-to-Dense [45]+WCL. On the right of each image, we zoom into the red rectangle. With WCL, predicted depth maps expose more precise structures and boundaries.

<table>
<thead>
<tr>
<th>Methods</th>
<th>( \delta_{1.25} )</th>
<th>( \delta_{1.25^2} )</th>
<th>( \delta_{1.25^3} )</th>
<th>AbsRel</th>
<th>Sq Rel</th>
<th>RMSE</th>
<th>RMSE log</th>
<th>log10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BTS-ResNet50</td>
<td>0.865</td>
<td>0.975</td>
<td>0.993</td>
<td>0.119</td>
<td>0.075</td>
<td>0.419</td>
<td>0.152</td>
<td>0.051</td>
</tr>
<tr>
<td>BTS-ResNet50+WCL</td>
<td>0.871</td>
<td>0.977</td>
<td>0.994</td>
<td>0.117</td>
<td>0.072</td>
<td>0.409</td>
<td>0.149</td>
<td>0.050</td>
</tr>
<tr>
<td>BTS-DenseNet-121</td>
<td>0.865</td>
<td>0.976</td>
<td>\textbf{0.995}</td>
<td>0.120</td>
<td>0.075</td>
<td>0.421</td>
<td>0.152</td>
<td>0.051</td>
</tr>
<tr>
<td>BTS-DenseNet-121+WCL</td>
<td>0.869</td>
<td>0.977</td>
<td>0.994</td>
<td>0.117</td>
<td>0.072</td>
<td>\textbf{0.413}</td>
<td>\textbf{0.149}</td>
<td>\textbf{0.050}</td>
</tr>
</tbody>
</table>

Table 3. **Quantitative results on NYUv2** [61] – **Monocular Depth estimation.** Comparison between BTS variants and their WCL counterparts.

the same training parameters suggested by the authors [36], except for the batch size that is set to 10. For this task, we set \( w = 0.1, N = 7 \) for both the backbones. Tab. 3 shows that our WCL module allows consistent improvements over both BTS variants.

### 4.3.3 Self-Supervised Monocular Depth Estimation

Self-supervised monocular depth estimation eliminates the need for ground truth depth labels, which are usually hard to source. Supervision can be obtained in the form of monocular videos or stereo images. We select MonoDepth2 [21] as a state-of-the-art baseline for this task, simultaneously learning for depth and relative poses between consecutive frames in a video to implement the aforementioned self-supervised training scheme. We test it on the KITTI dataset [20] using the Eigen split [17]. Its training is realized by minimizing the photometric re-projection errors, either between temporally adjacent frames or stereo images. We use ResNet18 [25] as the backbone, process images resized to \( 192 \times 640 \), and keep the same training parameters detailed by the authors [21]. In both cases, we evaluate our module setting \( w = 0.01, N = 7 \). Tab. 4 confirms that our method can also boost the accuracy of self-supervised depth estimation frameworks.

### 4.4. Ablation Study

In this section, we conduct an ablation study to verify the impact of the different hyper-parameters of our WCL module. For the experiments in this section, we focus on the depth completion task; we use a subset of 10000 samples from the KITTI depth completion dataset for training and evaluate the performance on the validation split. Images are center cropped to \( 1216 \times 256 \), to focus on regions with available LiDAR points. We use Sparse-to-Dense [45] as the baseline network and adopt ResNet-18 as the backbone. All the parameters are optimized using Adam (\( \beta_1 = 0.9, \beta_2 = 0.99 \)) and the weight decay factor is set to 0.0001. The learning rate is initialized to 0.001, decayed by \( \{0.5, 0.2, 0.1, 0.01\} \) at epoch \( \{10, 15, 20, 25\} \). The network is trained for 30 epochs using a batch size of 10 samples. RMSE, MAE, and iMAE are used as the evaluation metrics.

**Window size.** We measure how the window size over which the contrastive loss is applied impacts the final results. Tab. 5 collects the outcome of this experiment. We evaluate our module with window sizes, \( 3, 5, 7, 9, 11, 13, 15 \). \( w \) is fixed instead, to 0.1. We can observe that window with \( N = 7 \) outperforms others according to the main evaluation metric, RMSE. In contrast, bigger window sizes cannot improve further while increasing the computational requirements. A small window size (\( N = 3 \)) yields negli-
gible improvements, probably because most $3 \times 3$ regions are not significant enough for applying contrastive learning effectively.

**Shifted windows** When using WCL, all windows are non-overlapped. Thus, distribution optimization occurs locally. Previous works exploiting windows processing as well [44,75] use effective shifted window partitioning to introduce connections between neighboring non-overlapping windows. We ran an ablation study about whether shifted window partitioning can bring improvement in our method or not. Following [44,75], we shift the windows by $(\frac{N}{T}, \frac{N}{T})$ pixels in the feature map and calculate the contrastive loss after computing the loss of the previous windows. We set $w = 0.1$ and $N = 7$. We shift the windows $1, 2, 3, 4$ times. From the results in Tab. 6, we can conclude that shifting the windows does not bring improvement while increasing computational cost.

**Embedded module Location** Our WCL module can be easily embedded in between network layers, allowing to contrast pixels in the representation space. The baseline network has six layers in the decoder stage. We define them as $layer5$, $layer4$, $layer3$, $layer2$, $layer1$, and $layer0$ from bottleneck to final layer. We performed an ablation study to determine how much improvement our module can bring when placed at different locations within the network. We set $w = 0.1$ and $N = 7$ in all the experiments except the last one. In the last one, we set $w_1 = 0.1$ for the contrastive loss between $layer1$ and $layer0$ and $w_2 = 0.0001$ between $layer2$ and $layer1$. From the results in Tab. 7, we can find that the closer to the final layer, the better the results. The outputs from the layers near the final layer present features strongly related to final depth maps, while the outputs near the bottleneck show higher-level, semantical information. Partitioning and contrasting the outputs near the bottleneck break semantic information and cause degradation.

## 5. Conclusion

We presented a Window-based Contrastive Learning (WCL) module for depth prediction. Our approach partitions the image into windows, and the contrastive loss is implemented within each. Accordingly, it constructs and sorts positive and negative pairs, then enlarges the gap between the two in feature space, which makes depth distribution more meaningful of the real depth in the scene. We evaluate our method on multiple depth prediction tasks, such as depth completion, depth estimation, and self-supervised depth estimation, reporting consistent improvements.

### Table 4. Quantitative results on KITTI Eigen split [17] – Self-Supervised Monocular depth estimation. All methods are trained and tested with $192 \times 640$ images. The best results in each category are in **bold**; M: Monocular supervision; S: Stereo supervision.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Train</th>
<th>$\delta_{1.25}$</th>
<th>$\delta_{1.25}^2$</th>
<th>$\delta_{1.25}^3$</th>
<th>AbsRel</th>
<th>Sq Rel</th>
<th>RMSE</th>
<th>RMSE log</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monodepth2</td>
<td>M</td>
<td>0.871</td>
<td>0.957</td>
<td>0.980</td>
<td>0.118</td>
<td>0.912</td>
<td>4.911</td>
<td>0.196</td>
</tr>
<tr>
<td>Monodepth2 + WCL</td>
<td>M</td>
<td><strong>0.873</strong></td>
<td><strong>0.959</strong></td>
<td><strong>0.981</strong></td>
<td><strong>0.116</strong></td>
<td><strong>0.852</strong></td>
<td><strong>4.837</strong></td>
<td><strong>0.194</strong></td>
</tr>
<tr>
<td>Monodepth2</td>
<td>S</td>
<td>0.865</td>
<td>0.949</td>
<td>0.975</td>
<td><strong>0.109</strong></td>
<td>0.909</td>
<td>5.015</td>
<td>0.208</td>
</tr>
<tr>
<td>Monodepth2 + WCL</td>
<td>S</td>
<td><strong>0.867</strong></td>
<td><strong>0.951</strong></td>
<td><strong>0.975</strong></td>
<td><strong>0.109</strong></td>
<td><strong>0.892</strong></td>
<td><strong>4.961</strong></td>
<td><strong>0.207</strong></td>
</tr>
</tbody>
</table>

### Table 5. Ablation results on the different window sizes on KITTI depth completion validation set.

<table>
<thead>
<tr>
<th>Window size (N)</th>
<th>RMSE (M)</th>
<th>MAE (mm)</th>
<th>iMAE (1/km)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>930.326</td>
<td>269.866</td>
<td>2.575</td>
</tr>
<tr>
<td>3</td>
<td>927.818</td>
<td>266.248</td>
<td>2.536</td>
</tr>
<tr>
<td>5</td>
<td>925.169</td>
<td>267.290</td>
<td>2.813</td>
</tr>
<tr>
<td>7</td>
<td><strong>922.512</strong></td>
<td>264.044</td>
<td>2.030</td>
</tr>
<tr>
<td>9</td>
<td>925.358</td>
<td>267.614</td>
<td>2.045</td>
</tr>
<tr>
<td>11</td>
<td>926.511</td>
<td><strong>263.053</strong></td>
<td>2.639</td>
</tr>
<tr>
<td>13</td>
<td>925.072</td>
<td>265.628</td>
<td>2.429</td>
</tr>
<tr>
<td>15</td>
<td>925.074</td>
<td>264.810</td>
<td><strong>2.022</strong></td>
</tr>
</tbody>
</table>

### Table 6. Ablation results on the shift number on KITTI depth completion validation set.

<table>
<thead>
<tr>
<th>Shift number (N)</th>
<th>RMSE (M)</th>
<th>MAE (mm)</th>
<th>iMAE (1/km)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>930.326</td>
<td>269.866</td>
<td>2.575</td>
</tr>
<tr>
<td>0</td>
<td><strong>922.512</strong></td>
<td><strong>264.044</strong></td>
<td><strong>2.030</strong></td>
</tr>
<tr>
<td>1</td>
<td>923.461</td>
<td>266.447</td>
<td>2.692</td>
</tr>
<tr>
<td>2</td>
<td>926.473</td>
<td>265.348</td>
<td>2.228</td>
</tr>
<tr>
<td>3</td>
<td>929.037</td>
<td>266.340</td>
<td>2.083</td>
</tr>
<tr>
<td>4</td>
<td>927.404</td>
<td>264.493</td>
<td>2.324</td>
</tr>
</tbody>
</table>

### Table 7. Ablation results on the different locations on KITTI depth completion validation set. -1 denotes between $layer1$ and $layer0$; -2 denotes between $layer2$ and $layer1$; -3 denotes between $layer3$ and $layer2$; -4 denotes between $layer4$ and $layer3$; -5 denotes between $layer5$ and $layer4$.

<table>
<thead>
<tr>
<th>Location</th>
<th>RMSE (M)</th>
<th>MAE (mm)</th>
<th>iMAE (1/km)</th>
</tr>
</thead>
<tbody>
<tr>
<td>baseline</td>
<td>930.326</td>
<td>269.866</td>
<td>2.575</td>
</tr>
<tr>
<td>-1</td>
<td>922.512</td>
<td>264.044</td>
<td>2.030</td>
</tr>
<tr>
<td>-2</td>
<td>928.279</td>
<td>267.821</td>
<td>2.150</td>
</tr>
<tr>
<td>-3</td>
<td>929.924</td>
<td>264.688</td>
<td>2.594</td>
</tr>
<tr>
<td>-4</td>
<td>932.127</td>
<td>270.720</td>
<td><strong>1.992</strong></td>
</tr>
<tr>
<td>-5</td>
<td>940.022</td>
<td>270.118</td>
<td>2.120</td>
</tr>
<tr>
<td>-1 &amp; -2</td>
<td><strong>912.286</strong></td>
<td><strong>263.886</strong></td>
<td><strong>6.604</strong></td>
</tr>
</tbody>
</table>
References


