AdaBins: Depth Estimation Using Adaptive Bins

Shariq Farooq Bhat
KAUST
shariq.bhat@kaust.edu.sa

Ibraheem Alhashim
KAUST
ibraheem.alhashim@kaust.edu.sa

Peter Wonka
KAUST
pwonka@gmail.com

Abstract

We address the problem of estimating a high quality dense depth map from a single RGB input image. We start out with a baseline encoder-decoder convolutional neural network architecture and pose the question of how the global processing of information can help improve overall depth estimation. To this end, we propose a transformer-based architecture block that divides the depth range into bins whose center value is estimated adaptively per image. The final depth values are estimated as linear combinations of the bin centers. We call our new building block AdaBins. *Our results show a decisive improvement over the state-of-the-art on several popular depth datasets across all metrics. We also validate the effectiveness of the proposed block with an ablation study and provide the code and corresponding pre-trained weights of the new state-of-the-art model.*

1. Introduction

This paper tackles the problem of estimating a high quality dense depth map from a single RGB input image. This is a classical problem in computer vision that is essential for many applications [26, 30, 16, 6]. In this work, we propose a new architecture building block, called *AdaBins* that leads to a new state-of-the-art architecture for depth estimation on the two most popular indoor and outdoor datasets, NYU [36] and KITTI [13].

The motivation for our work is the conjecture that current architectures do not perform enough global analysis of the output values. A drawback of convolutional layers is that they only process global information once the tensors reach a very low spatial resolution at or near the bottleneck. However, we believe that global processing is much more powerful when done at high resolution. Our general idea is to perform a global statistical analysis of the output of a traditional encoder-decoder architecture and to refine the output with a learned post-processing building block that operates at the highest resolution. As a particular realization of this idea, we propose to analyze and modify the distribution of the depth values.

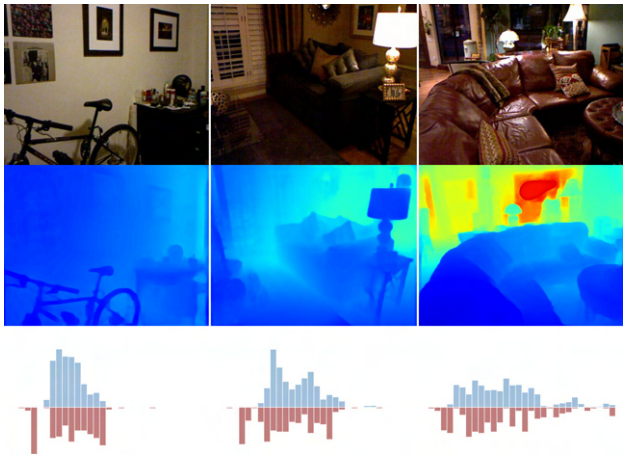


Figure 1: Illustration of AdaBins: **Top**: input RGB images. **Middle**: depth predicted by our model. **Bottom**: histogram of depth values of the ground truth (blue) and predicted bin density (red) with depth values increasing from left to right. Note that the predicted bin-centers are focused near smaller depth values for closeup images but are widely distributed for images with a wider range of depth values.

Depth distribution corresponding to different RGB inputs can vary to a large extent (see Fig. 1). Some images have most of the objects located over a very small range of depth values. Closeup images of furniture will, for example, contain pixels most of which are close to the camera while other images may have depth values distributed over a much broader range, e.g. a corridor, where depth values range from a small value to the maximum depth supported by the network. Along with the ill-posed nature of the problem, such a variation in depth distribution makes depth regression in an end-to-end manner an even more difficult task. Recent works have proposed to exploit assumptions about indoor environments such as planarity constraints [25, 21] to guide the network, which may or may not hold for a real-world environment, especially for outdoors scenes.

Instead of imposing such assumptions, we investigate an approach where the network learns to adaptively *focus* on regions of the depth range which are more probable to occur

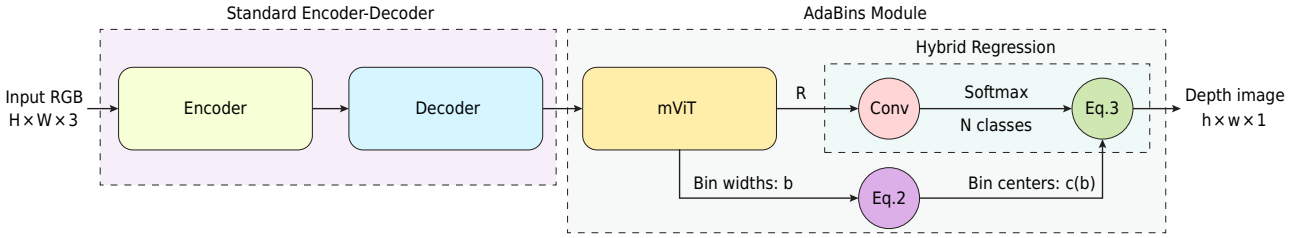


Figure 2: Overview of our proposed network architecture. Our architecture consists of two major components: an encoder-decoder block and our proposed adaptive bin-width estimator block called AdaBins. The input to our network is an RGB image of spatial dimensions H and W, and the output is a single channel $h \times w$ depth image (e.g., half the spatial resolution).

in the scene of the input image.

Our main contributions are the following:

- We propose an architecture building block that performs global processing of the scene's information. We propose to divide the predicted depth range into bins where the bin widths change per image. The final depth estimation is a linear combination of the bin center values.
- We show a decisive improvement for supervised single image depth estimation across all metrics for the two most popular datasets, NYU [36] and KITTI [13].
- We analyze our findings and investigate different modifications on the proposed AdaBins block and study their effect on the accuracy of the depth estimation.

2. Related Work

The problem of 3D scene reconstruction from RGB images is an ill-posed problem. Issues such as lack of scene coverage, scale ambiguities, translucent or reflective materials all contribute to ambiguous cases where geometry cannot be derived from appearance. Recently, methods that rely on convolutional neural networks (CNNs) are able to produce reasonable depth maps from a single RGB input image at real-time speeds.

Monocular depth estimation has been considered by many CNN methods as a regression of a dense depth map from a single RGB image [7, 24, 44, 15, 45, 10, 18, 1, 25, 21]. As the two most important competitors, we consider BTS [25] and DAV [21]. BTS uses local planar guidance layers to guide the features to full resolution instead of standard upsampling layers during the decoding phase. DAV uses a standard encoder-decoder scheme and proposes to exploit co-planarity of objects in the scene via attention at the bottleneck. Our results section compares to these (and many other) methods.

Encoder-decoder networks have made significant contributions in many vision related problems such as image segmentation [34], optical flow estimation [9], and image restoration [27]. In recent years, the use of such architectures have shown great success both in the supervised and the unsupervised setting of the depth estimation problem [14, 40, 20, 47, 1]. Such methods typically use one or more encoder-decoder networks as a sub part of their larger network. In this paper we adapted the baseline encoder-decoder network architecture used by [1]. This allows us to more explicitly study the performance attribution of our proposed extension on the pipeline which is typically a difficult task.

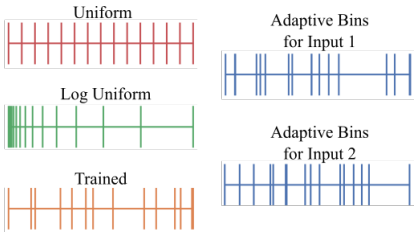


Figure 3: Choices for bin widths. Uniform and Log-uniform bins are pre-determined. 'Trained bins' vary from one dataset to another. Adaptive bins vary for each input image.

Transformer networks are gaining greater attention as a viable building block outside of their traditional use in NLP tasks and into computer vision tasks [31, 42, 2, 5]. Following the success of recent trends that combine CNNs with Transformers [2], we propose to leverage a Transformer encoder as a building block for non-local processing on the output of a CNN.

3. Methodology

In this section, we present the motivation for this work, provide details of the AdaBins architecture, and describe the corresponding loss functions used.

3.1. Motivation

Our idea could be seen as a generalization of depth estimation via an ordinal regression network as proposed by

Fu et al. [10]. Fu et al. observed that a performance improvement could be achieved if the depth regression task is transformed into a classification task. They proposed to divide the depth range into a fixed number of bins of predetermined width. Our generalization solves multiple limitations of the initial approach. First, we propose to compute adaptive bins that dynamically change depending on the features of the input scene. Second, a classification approach leads to a discretization of depth values which results in poor visual quality with obvious sharp depth discontinuities. This might still lead to good results with regard to the standard evaluation metrics, but it can present a challenge for downstream applications, e.g. computational photography or 3D reconstruction. Therefore, we propose to predict the final depth values as a linear combination of bin centers. This allows us to combine the advantages of classification with the advantages of depth-map regression. Finally, compared to other architectures, e.g. DAV [21], we compute information globally at a high resolution and not primarily in the bottleneck part at a low resolution.

3.2. AdaBins design

Here, we discuss four design choices of our proposed architecture that are most important for the obtained results.

First, we employ an adaptive binning strategy to discretize the depth interval $D = (d_{min}, d_{max})$ into N bins. This interval is fixed for a given dataset and is determined by dataset specification or manually set to a reasonable range. To illustrate our idea of dividing a depth interval into bins, we would like to contrast our final solution with three other possible design choices we evaluated:

- Fixed bins with a uniform bin width: the depth interval D is divided into N bins of equal size.
- Fixed bins with a log scale bin width: the depth interval D is divided into bins of equal size in log scale.
- Trained bin widths: the bin widths are adaptive and can be learned for a particular dataset. While the bin widths are general, all images finally share the same bin subdivision of the depth interval D.
- AdaBins: the bin widths **b** are adaptively computed for each image.

We recommend the strategy of AdaBins as the best option and our ablation study validates this choice by showing the superiority of this design over its alternatives. An illustration of the four design choices for bin widths can be seen in Fig. 3.

Second, discretizing the depth interval D into bins and assigning each pixel to a single bin leads to depth discretization artifacts. We therefore predict the final depth as a linear combination of bin centers enabling the model to estimate smoothly varying depth values.

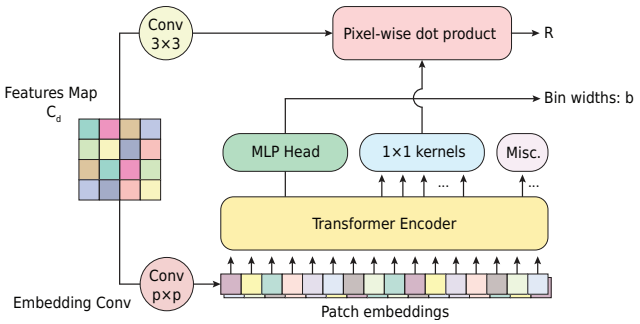


Figure 4: An overview of the mini-ViT block. The input to the block is a multi-channel feature map of the input image. The block includes a Transformer encoder that is applied on patch embeddings of the input for the purpose of learning to estimate bin widths b and a set of convolutional kernels needed to compute our Range-Attention-Maps R.

Third, several previous architectures propose performing global processing using attention blocks to process information after an encoder block in the architecture (e.g., image captioning [4, 17] or object detection [2]). Also, the current state-of-the-art in depth estimation uses this strategy [21]. Such an architecture consists of three blocks ordered as such: encoder, attention, followed by a decoder. We initially followed this approach but noticed that better results can be achieved when using attention at the spatially higher resolution tensors. We therefore propose an architecture that also has these three blocks, but ordered as follows: encoder, decoder, and finally attention.

Fourth, we would like to build on the simplest possible architecture to isolate the effects of our newly proposed *AdaBins* concept. We therefore build on a modern encoder-decoder [1] using EfficientNet B5 [39] as the backbone for the encoder.

In the next subsection, we provide a description of the entire architecture.

3.3. Architecture description

Fig. 2 shows an overview of our proposed depth estimating architecture. Our architecture consists of two major components: 1) an encoder-decoder block built on a pretrained EfficientNet B5 [39] encoder and a standard feature upsampling decoder; 2) our proposed adaptive bin-width estimator block called AdaBins. The first component is primarily based on the simple depth regression network of Alhashim and Wonka [1] with some modifications. The two basic modifications are switching the encoder from DenseNet [19] to EfficientNet B5 and using a different appropriate loss function for the new architecture. In addition, the output of the decoder is a tensor $\mathbf{x}_d \in \mathbb{R}^{h \times w \times C_d}$,

Patch size (p)	E	Layers	num heads	C	MLP Size	Params
16	128	4	4	128	1024	5.8 M

Table 1: Mini-ViT architecture details.

not a single channel image representing the final depth values. We refer to this tensor as the "*decoded features*". The second component is a key contribution in this paper, the AdaBins module. The input to the AdaBins module are *decoded features* of size $h \times w \times C_d$ and the output tensor is of size $h \times w \times 1$. Due to memory limitations of current GPU hardware, we use $h = H/2$ and $w = W/2$ to facilitate better learning with larger batch sizes. The final depth map is computed by simply bilinearly upsampling to $H \times W \times 1$.

The first block in the AdaBins module is called mini-ViT. An overview of this block is shown in Fig. 4. It is a simplified version of a recently proposed technique of using transformers for image recognition [5] with minor modifications. The details of mini-ViT are explained in the next paragraph. There are two outputs of mini-ViT: 1) a vector $\mathbf{b}$ of bin-widths, which defines how the depth interval D is to be divided for the input image, and 2) Range-Attention-Maps $\mathcal{R}$ of size $h \times w \times C$, that contain useful information for pixel-level depth computation.

Mini-ViT. Estimating sub-intervals within the depth range D which are more probable to occur for a given image would require a combination of local structural information and global distributional information at the same time. We propose to use global attention in order to calculate a bin-widths vector $\mathbf{b}$ for each input image. Global attention is expensive both in terms of memory and computational complexity, especially at higher resolutions. However, recent rapid advances in transformers provide some efficient alternatives. We take inspiration from the Vision Transformer ViT [5] in designing our AdaBins module with transformers. We also use a much smaller version of the transformer proposed as our dataset is smaller and refer to this transformer as mini-ViT or mViT in the following description.

Bin-widths. We first describe how the bin-widths vector $\mathbf{b}$ is obtained using mViT. The input to the mViT block is a tensor of *decoded features* $\mathbf{x_d} \in \mathbb{R}^{h \times w \times C_d}$. However, a transformer takes a sequence of fixed size vectors as input. We first pass the *decoded features* through a convolutional block, named as *Embedding Conv* (see Fig. 4), with kernel size $p \times p$, stride p and number of output channels E. Thus, the result of this convolution is a tensor of size $h/p \times w/p \times E$ (assuming both h and w are divisible by p). The result is reshaped into a spatially flattened tensor $\mathbf{x_p} \in \mathbb{R}^{S \times E}$, where $S = \frac{hw}{p^2}$ serves as the effective sequence length for the transformer. We refer to this sequence of E-dimensional vectors as *patch embeddings*.

Following common practice [2, 5], we add learned positional encodings to the patch embeddings before feeding them to the transformer. Our transformer is a small transformer encoder (see Table. 1 for details) and outputs a sequence of *output embeddings* $\mathbf{x_o} \in \mathbb{R}^{S \times E}$. We use an MLP head over the first output embedding (we also experimented with a version that has an additional special token as first input, but did not see an improvement). The MLP head uses a ReLU activation and outputs an N-dimensional vector $\mathbf{b}'$. Finally, we normalize the vector $\mathbf{b}'$ such that it sums up to 1, to obtain the bin-widths vector $\mathbf{b}$ as follows:

$$b_i = \frac{b'_i + \epsilon}{\sum_{j=1}^{N}(b'_j + \epsilon)}, \tag{1}$$

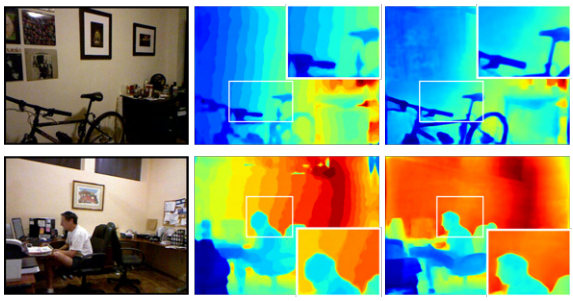
where $\epsilon = 10^{-3}$. The small positive ϵ ensures each bin-width is strictly positive. The normalization introduces a competition among the bin-widths and conceptually forces the network to *focus* on sub-intervals within D by predicting smaller bin-widths at interesting regions of D.

In the next subsection, we describe how the Range-Attention-Maps $\mathcal{R}$ are obtained from the *decoded features* and the transformer output embeddings.

Range attention maps. At this point, the *decoded features* represent a high-resolution and local pixel-level information while the transformer output embeddings effectively contain more global information. As shown in Fig. 4, output embeddings 2 through $C + 1$ from the transformer are used as a set of 1×1 convolutional kernels and are convolved with the *decoded features* (following a 3×3 convolutional layer) to obtain the Range-Attention Maps $\mathcal{R}$. This is equivalent to calculating the Dot-Product attention weights between pixel-wise features treated as 'keys' and transformer output embeddings as 'queries'. This simple design of using output embeddings as convolutional kernels lets the network integrate adaptive global information from the transformer into the local information of the *decoded features*. $\mathcal{R}$ and $\mathbf{b}$ are used together to obtain the final depth map.

Hybrid regression. Range-Attention Maps $\mathcal{R}$ are passed through a 1×1 convolutional layer to obtain N-channels which is followed by a Softmax activation. We interpret the N Softmax scores p_k, $k = 1, ..., N$, at each pixel as probabilities over N depth-bin-centers $c(\mathbf{b}) := \{c(b_1), c(b_2), ..., c(b_N)\}$ calculated from bin-widths vector $\mathbf{b}$ as follows:

$$c(b_i) = d_{min} + (d_{max} - d_{min})(b_i/2 + \sum_{j=1}^{i-1} b_j) \tag{2}$$



RGB Fu *et al*. [10] Ours

Figure 5: Demonstration of artifacts introduced by the discretization of the depth interval. Our hybrid regression results in smoother depth maps.

Finally, at each pixel, the final depth value $\tilde{d}$ is calculated from the linear combination of Softmax scores at that pixel and the depth-bin-centers $c(\mathbf{b})$ as follows:

$$\tilde{d} = \sum_{k=1}^{N} c(b_k) p_k \tag{3}$$

Compared to Fu et al. [10] we do not predict the depth as the bin center of the most likely bin. This enables us to predict smooth depth maps without the discretization artifacts as can bee seen in Fig. 5.

3.4. Loss function

Pixel-wise depth loss. Inspired by [25], we use a scaled version of the Scale-Invariant loss (SI) introduced by Eigen *et al*. [7]:

$$\mathcal{L}_{pixel} = \alpha \sqrt{\frac{1}{T}\sum_i g_i^2 - \frac{\lambda}{T^2}(\sum_i g_i)^2} \tag{4}$$

where $g_i = \log \tilde{d}_i - \log d_i$ and the ground truth depth d_i and T denotes the number of pixels having valid ground truth values. We use $\lambda = 0.85$ and $\alpha = 10$ for all our experiments.

Bin-center density loss. This loss term encourages the distribution of bin centers to follow the distribution of depth values in the ground truth. We would like to encourage the bin centers to be close to the actual ground truth depth values and the other way around. We denote the set of bin centers as $c(\mathbf{b})$ and the set of all depth values in the ground truth image as X and use the bi-directional Chamfer Loss [8] as a regularizer:

$$\mathcal{L}_{bins} = \sum_{x \in X} \min_{y \in c(\mathbf{b})} \|x - y\|^2 + \sum_{y \in c(\mathbf{b})} \min_{x \in X} \|x - y\|^2 \tag{5}$$

Finally, we define the total loss as:

$$\mathcal{L}_{total} = \mathcal{L}_{pixel} + \beta \mathcal{L}_{bins} \tag{6}$$

We set $\beta = 0.1$ for all our experiments. We experimented with different loss functions including the RMSE loss, and the combined SSIM [41] plus L_1 loss suggested by [1]. However, we were able to achieve the best results with our proposed loss. We offer a comparison of the different loss functions and their performance in our ablation study.

4. Experiments

We conducted an extensive set of experiments on the standard depth estimation from a single image datasets for both indoor and outdoor scenes. In the following, we first briefly describe the datasets and the evaluation metrics, and then present quantitative comparisons to the state-of-the-art in supervised monocular depth estimation.

4.1. Datasets and evaluation metrics

NYU Depth v2 is a dataset that provides images and depth maps for different indoor scenes captured at a pixel resolution of 640×480 [36]. The dataset contains 120K training samples and 654 testing samples [7]. We train our network on a 50K subset. The depth maps have an upper bound of 10 meters. Our network outputs depth prediction having a resolution of 320×240 which we then upsample by $2\times$ to match the ground truth resolution during both training and testing. We evaluate on the pre-defined center cropping by Eigen et al. [7]. At test time, we compute the final output by taking the average of an image's prediction and the prediction of its mirror image which is commonly used in previous work.

KITTI is a dataset that provides stereo images and corresponding 3D laser scans of outdoor scenes captured using equipment mounted on a moving vehicle [13]. The RGB images have a resolution of around 1241×376 while the corresponding depth maps are of very low density with lots of missing data. We train our network on a subset of around 26K images, from the left view, corresponding to scenes not included in the 697 test set specified by [7]. The depth maps have an upper bound of 80 meters. We train our network on a random crop of size 704×352. For evaluation, we use the crop as defined by Garg *et al*. [12] and bilinearly upsample the prediction to match the ground truth resolution. The final output is computed by taking the average of an image's prediction and the prediction of its mirror image.

SUN RGB-D is an indoor dataset consisting of around $10K$ images with high scene diversity collected with four different sensors [38, 43, 22]. We use this dataset only for cross-evaluating pre-trained models on the official test set of 5050 images. We do not use it for training.

Method	Encoder	*#params* (M)	$\delta_1\uparrow$	$\delta_2\uparrow$	$\delta_3\uparrow$	REL $\downarrow$	RMS $\downarrow$	$log_{10}\downarrow$
Eigen *et al*. [7]	-	141	0.769	0.950	0.988	0.158	0.641	–
Laina *et al*. [24]	ResNet-50	64	0.811	0.953	0.988	0.127	0.573	0.055
Hao *et al*. [15]	ResNet-101	60	0.841	0.966	0.991	0.127	0.555	0.053
Lee *et al*. [26]	-	119	0.837	0.971	0.994	0.131	0.538	–
Fu *et al*. [10]	ResNet-101	110	0.828	0.965	0.992	0.115	0.509	0.051
SharpNet [33]	-	-	0.836	0.966	0.993	0.139	0.502	0.047
Hu *et al*. [18]	SENet-154	157	0.866	0.975	0.993	0.115	0.530	0.050
Chen *et al*. [3]	SENet	210	0.878	0.977	0.994	0.111	0.514	0.048
Yin *et al*. [46]	ResNeXt-101	114	0.875	0.976	0.994	0.108	0.416	0.048
BTS [25]	DenseNet-161	47	0.885	0.978	0.994	0.110	0.392	0.047
DAV [21]	-	25	0.882	0.980	0.996	0.108	0.412	–
AdaBins (Ours)	EfficientNet-B5	78	**0.903**	**0.984**	**0.997**	**0.103**	**0.364**	**0.044**

Table 2: Comparison of performances on the NYU-Depth-v2 dataset. The reported numbers are from the corresponding original papers. Best results are in bold, second best are underlined.

Method	Encoder	*#params* (M)	$\delta_1\uparrow$	$\delta_2\uparrow$	$\delta_3\uparrow$	REL $\downarrow$	Sq Rel $\downarrow$	RMS $\downarrow$	RMS log $\downarrow$
Saxena *et al*. [35]	-	-	0.601	0.820	0.926	0.280	3.012	8.734	0.361
Eigen *et al*. [7]	-	-	0.702	0.898	0.967	0.203	1.548	6.307	0.282
Liu *et al*. [28]	-	40	0.680	0.898	0.967	0.201	1.584	6.471	0.273
Godard *et al*. [14]	ResNet-50	31	0.861	0.949	0.976	0.114	0.898	4.935	0.206
Kuznietsov *et al*. [23]	ResNet-50	-	0.862	0.960	0.986	0.113	0.741	4.621	0.189
Gan *et al*. [11]	ResNet-50	-	0.890	0.964	0.985	0.098	0.666	3.933	0.173
Fu *et al*. [10]	ResNet-101	110	0.932	0.984	0.994	0.072	0.307	2.727	0.120
Yin *et al*. [46]	ResNeXt-101	114	0.938	0.990	0.998	0.072	–	3.258	0.117
BTS[25]	ResNeXt-101	113	0.956	0.993	0.998	0.059	0.245	2.756	0.096
AdaBins (Ours)	EfficientNet-B5	78	**0.964**	**0.995**	**0.999**	**0.058**	**0.190**	**2.360**	**0.088**

Table 3: Comparison of performances on the KITTI dataset. We compare our network against the state-of-the-art on this dataset. The reported numbers are from the corresponding original papers. Measurements are made for the depth range from $0m$ to $80m$. Best results are in bold, second best are underlined.

Loss	$\delta_1\uparrow$	$\delta_2\uparrow$	$\delta_3\uparrow$	REL$\downarrow$	RMS$\downarrow$	$log_{10}\downarrow$
L_1/SSIM	0.888	0.980	0.995	0.107	0.384	0.046
SI	0.897	**0.984**	**0.997**	0.106	0.368	**0.044**
SI+Bins	**0.903**	**0.984**	**0.997**	**0.103**	**0.364**	**0.044**

Table 4: Comparison of performance with respect to the choice of loss function.

Evaluation metrics. We use the standard six metrics used in prior work [7] to compare our method against state-of-the-art. These error metrics are defined as: average relative error (REL): $\frac{1}{n}\sum_p^n \frac{|y_p-\hat{y}_p|}{y}$; root mean squared error (RMS): $\sqrt{\frac{1}{n}\sum_p^n (y_p-\hat{y}_p)^2}$; average ($\log_{10}$) error: $\frac{1}{n}\sum_p^n |\log_{10}(y_p) - \log_{10}(\hat{y}_p)|$; threshold accuracy ($\delta_i$): % of y_p s.t. $\max(\frac{y_p}{\hat{y}_p}, \frac{\hat{y}_p}{y_p}) = \delta < thr$ for $thr = 1.25, 1.25^2, 1.25^3$; where y_p is a pixel in depth image y, $\hat{y}_p$ is a pixel in the predicted depth image $\hat{y}$, and n is the total number of pixels for each depth image. Additionally for KITTI, we use the two standard metrics: Squared Relative Difference (Sq. Rel): $\frac{1}{n}\sum_p^n \frac{\|y_p-\hat{y}_p\|^2}{y}$; and RMSE log: $\sqrt{\frac{1}{n}\sum_p^n \|\log y_p - \log \hat{y}_p\|^2}$.

4.2. Implementation details

We implement the proposed network in PyTorch [32]. For training, we use the AdamW optimizer [29] with weight-decay 10^{-2}. We use the 1-cycle policy [37] for the learning rate with $max\_lr = 3.5 \times 10^{-4}$, linear warm-up from $max\_lr/25$ to max_lr for the first 30% of iterations followed by cosine annealing to $max\_lr/75$. Total number of epochs is set to 25 with batch size 16. Training our model takes 20 min per epoch on a single node with four NVIDIA

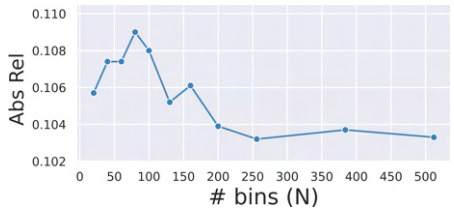


Figure 6: Effect of number of bins (N) on performance as measured by Absolute Relative Error metric. we can observe interesting behaviour for lower values of N. As N increases, performance starts to saturate.

V100 32GB GPUs. For all results presented we train for 25 epochs. The code and trained models will be made publicly available upon acceptance. Our main model has about 78M parameters: 28M for the CNN encoder, 44M for the CNN decoder, and 5.8M for the new AdaBins module.

4.3. Comparison to the state-of-the-art

We consider the following two methods to be our main competitors: BTS [25] and DAV [21]. For completeness, we also include selected previous related methods in the comparison tables. For BTS and DAV we report the corresponding evaluation numbers from their papers. For BTS we also verified these numbers by retraining their network using the authors code. DAV did not have code available by the deadline, but the authors sent us the resulting depth images used in our figures. In our tables we report the numbers given by the authors in their paper. [1]

NYU-Depth-v2. See Table 2 for the comparison of the performance on the official NYU-Depth-v2 test set. While the state-of-the-art performance on NYU has been saturated for quite some time, we were able to significantly outperform the state-of-the-art in all metrics. The large gap to the previous state-of-the-art emphasises that our proposed architecture addition makes an important contribution to improving the results.

KITTI. Table 3 lists the performance metrics on the KITTI dataset. Our proposed architecture significantly outperforms previous state-of-the-art across all metrics. In particular, our method improves the RMS score by about 13.5% and Squared Relative Difference by 22.4% over the previous state-of-the-art.

SUN RGB-D. To compare the generalisation performance, we perform a cross-dataset evaluation by training our network on the NYU-Depth-v2 dataset and evaluate it on the test set of the SUN RGB-D dataset without any fine-tuning. For comparison, we also used the same strategy for competing methods for which pretrained models are available [25, 46, 3] and report results in Table. 5.

[1] DAV computes the depth maps at 1/4th the resolution and then downsamples the GT for evaluation. However, we believe that all other methods, including ours, evaluate at the full resolution.

Method	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL$\downarrow$	RMS$\downarrow$	$log_{10} \downarrow$
Chen [3]	0.757	0.943	**0.984**	0.166	0.494	0.071
Yin [46]	0.696	0.912	0.973	0.183	0.541	0.082
BTS [25]	0.740	0.933	0.980	0.172	0.515	0.075
Ours	**0.771**	**0.944**	0.983	**0.159**	**0.476**	**0.068**

Table 5: Results of models trained on the NYU-Depth-v2 dataset and tested on the SUN RGB-D dataset [38] without fine-tuning.

Variant	$\delta_1 \uparrow$	$\delta_2 \uparrow$	$\delta_3 \uparrow$	REL $\downarrow$	RMS $\downarrow$
Base + R	0.881	0.980	0.996	0.111	0.419
Base + Uniform-Fix-HR	0.892	0.981	0.995	0.107	0.383
Base + Log-Fix-HR	0.896	0.981	0.995	0.108	0.379
Base + Train-Fix-HR	0.893	0.981	0.995	0.109	0.381
Base + AdaBins-HR	**0.903**	**0.984**	**0.997**	**0.103**	**0.364**

Table 6: Comparison of different design choices for bin-widths and regression. AdaBins module results in a significant boost in performance. Base: encoder-decoder with an EfficientNet B5 encoder. R: standard regression. HR: Hybrid Regression. (Log)Uniform-Fix: Fixed (log) uniform bin-widths. Train-Fix: Trained bin-widths but Fixed for each dataset.

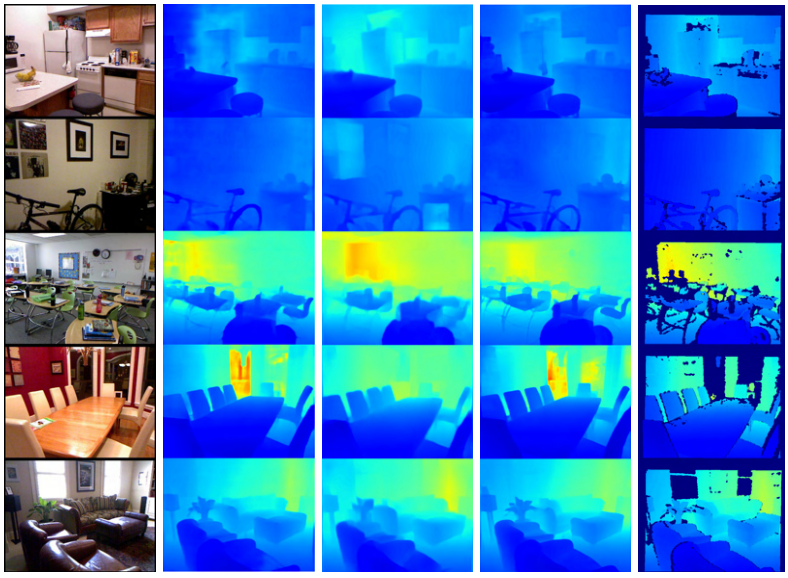
4.4. Ablation study

For our ablation study, we evaluate the influence of the following design choices on our results:

AdaBins. We first evaluate the importance of our AdaBins module. We remove the AdaBins block from the architecture and use the encoder-decoder to directly predict the depth map by setting $C_d = 1$. We then use the loss given by Eq. 4 to train the network. We call this design *standard regression* and compare it against variants of our AdaBins module. Table. 6 shows that the architecture without AdaBins (Row 1) performs worse than all other variants (Rows 2-5).

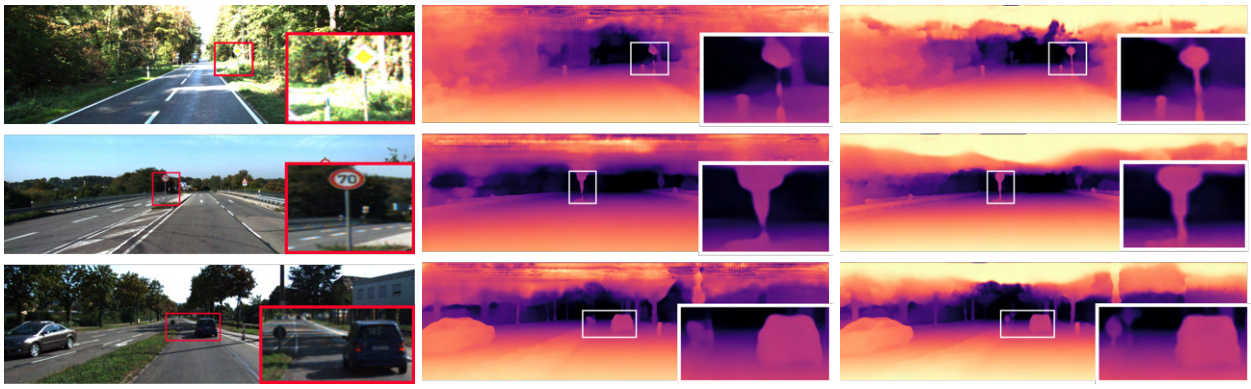
Bin types. In this set of experiments we examine the performance of adaptive bins over other choices as stated in Sec. 3.2. Table. 6 lists results for all the discussed variants. The Trained-but-Fixed variant performs worst among all choices and our final choice employing adaptive bins significantly improves the performance and outperforms all other variants.

Number of bins (N). To study the influence of the number of bins, we train our network for various values of N and measure the performance in terms of Absolute Relative Error metric. Results are plotted in Fig. 6. Interestingly, starting from $N = 20$, the error first increases with increasing N and then decreases significantly. As we keep



(a) RGB (b) BTS [25] (c) DAV [3] (d) Ours (e) GT

Figure 7: Qualitative comparison with the state-of-the-art on the NYU-Depth-v2 dataset.



(a) RGB (b) BTS [25] (c) Ours

Figure 8: Qualitative comparison with the state-of-the-art on the KITTI dataset.

increasing N above 256, and with higher values the gain in performance starts to diminish. We use $N = 256$ for our final model.

Loss function. Table. 4 lists performance corresponding to the three choices of loss function. Firstly, the L_1/SSIM combination does not lead to the state-of-the-art performance in our case. Secondly, we trained our network with and without the proposed Chamfer loss (Eq. 5). Introducing the Chamfer loss clearly gives a boost to the performance. For example, introducing the Chamfer loss reduces the Absolute Relative Error from 10.6% to 10.3%.

5. Conclusion

We introduced a new architecture block, called *AdaBins* for depth estimation from a single RGB image. AdaBins leads to a decisive improvement in the state-of-the-art for the two most popular datasets, NYU and KITTI. In future work, we would like to investigate if global processing of information at a high resolution can also improve performance on other tasks, such as segmentation, normal estimation, and 3D reconstruction from multiple images.

Acknowledgements This work was supported by the KAUST Office of Sponsored Research (OSR) under Award No. OSR-CRG2018-3730.

References

[1] Ibraheem Alhashim and Peter Wonka. High quality monocular depth estimation via transfer learning. *CoRR*, abs/1812.11941, 2018.

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 213–229, Cham, 2020. Springer International Publishing.

[3] Xiaotian Chen, Xuejin Chen, and Zheng-Jun Zha. Structure-aware residual pyramid network for monocular depth estimation. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 694–700. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[4] Marcella Cornia, Matteo Stefanini, Lorenzo Baraldi, and Rita Cucchiara. Meshed-memory transformer for image captioning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[6] Ruofei Du, Eric Lee Turner, Maksym Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Dourgarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Moura e Silva Cruces, Shahram Izadi, Adarsh Kowdle, Konstantine Nicholas John Tsotsos, and David Kim. Depthlab: Real-time 3d interaction with depth maps for mobile augmented reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*, page 15, 2020.

[7] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.

[8] H. Fan, H. Su, and L. Guibas. A point set generation network for 3d object reconstruction from a single image. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2463–2471, 2017.

[9] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2758–2766, 2015.

[10] Huan Fu, Mingming Gong, Chaohui Wang, Nematollah Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2002–2011, 2018.

[11] Yukang Gan, Xiangyu Xu, Wenxiu Sun, and Liang Lin. Monocular depth estimation with affinity, vertical pooling, and label enhancement. In Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, editors, *Computer Vision – ECCV 2018*, pages 232–247, Cham, 2018. Springer International Publishing.

[12] Ravi Garg, Vijay Kumar B.G., Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 740–756, Cham, 2016. Springer International Publishing.

[13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *I. J. Robotics Res.*, 32:1231–1237, 2013.

[14] Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6602–6611, 2017.

[15] Zhixiang Hao, Yu Li, Shaodi You, and Feng Lu. Detail preserving depth estimation from a single image using attention guided networks. *2018 International Conference on 3D Vision (3DV)*, pages 304–313, 2018.

[16] Caner Hazirbas, Lingni Ma, Csaba Domokos, and Daniel Cremers. Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *ACCV*, 2016.

[17] Simao Herdade, Armin Kappeler, Kofi Boakye, and Joao Soares. Image captioning: Transforming objects into words. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 11137–11147. Curran Associates, Inc., 2019.

[18] Junjie Hu, Mete Ozay, Yan Zhang, and Takayuki Okatani. Revisiting single image depth estimation: Toward higher resolution maps with accurate object boundaries. *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1043–1051, 2018.

[19] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269, 2017.

[20] Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. Deepmvs: Learning multi-view stereopsis. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2821–2830, 2018.

[21] Lam Huynh, Phong Nguyen-Ha, Jiri Matas, Esa Rahtu, and Janne Heikkila. Guiding monocular depth estimation using depth-attention volume. *arXiv preprint arXiv:2004.02760*, 2020.

[22] Allison Janoch, Sergey Karayev, Yangqing Jia, Jonathan T Barron, Mario Fritz, Kate Saenko, and Trevor Darrell. A category-level 3d object dataset: Putting the kinect to work. In *Consumer depth cameras for computer vision*, pages 141–165. Springer, 2013.

[23] Yevhen Kuznietsov, Jörg Stückler, and Bastian Leibe. Semi-supervised deep learning for monocular depth map prediction. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2215–2223, 2017.

[24] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction

with fully convolutional residual networks. *2016 Fourth International Conference on 3D Vision (3DV)*, pages 239–248, 2016.

[25] 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.

[26] Wonwoo Lee, Nohyoung Park, and Woontack Woo. Depth-assisted real-time 3d object detection for augmented reality. *ICAT'11*, 2:126–132, 2011.

[27] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2Noise: Learning image restoration without clean data. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2965–2974, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[28] Fayao Liu, Chunhua Shen, Guosheng Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38:2024–2039, 2016.

[29] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[30] Francesc Moreno-Noguer, Peter N. Belhumeur, and Shree K. Nayar. Active refocusing of images and videos. *ACM Trans. Graph.*, 26(3), July 2007.

[31] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In Jennifer Dy and Andreas Krause, editors, *Proceedings of Machine Learning Research*, volume 80 of *Proceedings of Machine Learning Research*, pages 4055–4064, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037. Curran Associates, Inc., 2019.

[33] 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 (ICCV) Workshops*, Oct 2019.

[34] 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*, pages 234–241, Cham, 2015. Springer International Publishing.

[35] Ashutosh Saxena, Sung H. Chung, and Andrew Y. Ng. Learning depth from single monocular images. In *Proceedings of the 18th International Conference on Neural Information Processing Systems*, NIPS'05, page 1161–1168, Cambridge, MA, USA, 2005. MIT Press.

[36] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision – ECCV 2012*, pages 746–760, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[37] Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of residual networks using large learning rates. *CoRR*, abs/1708.07120, 2017.

[38] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 567–576, 2015.

[39] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 2019.

[40] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5622–5631, 2017.

[41] Jiheng Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13:600–612, 2004.

[42] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[43] J. Xiao, A. Owens, and A. Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *2013 IEEE International Conference on Computer Vision*, pages 1625–1632, 2013.

[44] Dan Xu, Elisa Ricci, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Multi-scale continuous crfs as sequential deep networks for monocular depth estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5354–5362, 2017.

[45] Dong Xu, Wei Wang, Hao Tang, Hong W. Liu, Nicu Sebe, and Elisa Ricci. Structured attention guided convolutional neural fields for monocular depth estimation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3917–3925, 2018.

[46] 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 (ICCV)*, October 2019.

[47] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. Deeptam: Deep tracking and mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 822–838, 2018.