Self-Supervised Deep Monocular Depth Estimation with Ambiguity Boosting

Juan Luis Gonzalez Bello, *Student Member, IEEE,* and Munchurl Kim, *Senior Member, IEEE,*

Abstract—We propose a novel two-stage training strategy with ambiguity boosting for the self-supervised learning of single view depths from stereo images. Our proposed two-stage learning strategy firstly aims to obtain a coarse depth prior by training an auto-encoder network for a stereoscopic view synthesis task. This prior knowledge is then boosted and used to self-supervise the model in the second stage of training in our novel ambiguity boosting loss. Our ambiguity boosting loss is a confidence-guided type of data augmentation loss that improves the accuracy and consistency of generated depth maps under several transformations of the single-image input. To show the benefits of the proposed two-stage training strategy with boosting, our two previous depth estimation (DE) networks, one with t-shaped adaptive kernels and the other with exponential disparity volumes, are extended with our new learning strategy, referred to as DBoosterNet-t and DBoosterNet-e, respectively. Our self-supervised DBoosterNets are competitive, and in some cases even better, compared to the most recent supervised SOTA methods, and are remarkably superior to the previous self-supervised methods for monocular DE on the challenging KITTI dataset. We present intensive experimental results, showing the efficacy of our method for the self-supervised monocular DE task.

Index Terms—Monocular depth estimation, self-supervised learning, ambiguity boosting.

1 Introduction

Depth estimation (DE) refers to the task of predicting the distance between the camera and objects in a scene for each pixel in a target frame given single or multiple input images. Estimating the depth is an essential building block in computer vision and computational imaging. Depth from single or multiple images provides information about the underlying 3D scene that can be used for many applications ranging from the computer vision used in navigation systems, robotics, 3D reconstruction, novel view synthesis, salient object detection, and semantic segmentation, to the computational imaging of artificial de-focus blur, bokeh effects, and hazing/de-hazing processes, to cite a few.

Depth estimation has been extensively studied in classical multi-view geometry approaches [5], [6], [7], [8], [9], [10], [11]. However, non-learning-based classical techniques are extremely limited when estimating the depth from a single image, as they are based on rigid assumptions, handcrafted features, or require guidance from human users [12], [13], [14], [15], [16]. On the other hand, deep-learning-based methods outperform the classical techniques in multi-view input cases and have shown state-of-the-art (SOTA) performance for the single-image depth estimation tasks. However, there is still much room for improvement. There are several means of estimating depth from single or multiple images via deep learning. These include direct DE and indirect DE methods: (i) Direct DE methods directly produce the final depth maps and can further be divided into the supervised methods [17], [18], [19], [20], [21], [22], [23], [24], [25], [26] with hard-to-obtain depth ground-truth (GT) data and the self-supervised approaches where the networks are trained without GT, but with additional views such as stereo images [27], [28], [29], [30], [31], [32], [33], [34], monocular video [35], [36], [37], [38], [39], or stereo video [40], [41]; (ii) Indirect DE methods estimate adaptive kernels designed to synthesize a new view seen from a different camera position [1], [2], [42], [43], [44], from which a pseudo-disparity map can be extracted.

Both direct and indirect DE methods have advantages and disadvantages when training deep convolutional neural networks (DCNNs) in a self-supervised manner. Indirect DE methods are superior in terms of structural coherence and thin object detection in the estimated depths but tend to be ineffective for highly homogeneous and reflective areas, leaving "holes" in the generated depth maps. This occurs because the reflective and homogeneous regions can be better reconstructed in the synthetic images when they are left "untouched" by the predicted kernels. Additionally, indirect DE methods are associated with discretization artifacts in the estimated depth maps when their kernel sizes are small. Increasing the kernel sizes is not always an option, as they require more memory and higher computation complexity.

In this paper, we propose a new two-stage training strategy with ambiguity boosting which we apply to extend our previous depth estimation networks with t-shaped adaptive kernels [1] and exponentially quantized disparity volumes [2], yielding DBoosterNet-t and DBoosterNet-e, respectively. Our DBoosterNets (DBoosterNet-t and DBoosterNet-e) take an indirect DE approach in both training stages, but contrary to the previous indirect DE methods [1], [2], [42], [43], [44], they do not produce "holes" or discretization artifacts in the predicted depth maps (even with relatively small kernel sizes) owing to our ambiguity boosting loss strategy. With this novel ambiguity boosting loss, we generate different depth map estimates for the target input images under several transformations and selectively fuse them using our predicted confidence or "ambiguity" masks. The distilled or "boosted" depth estimates are used as self-supervision

- *http://viclab.kaist.ac.kr*

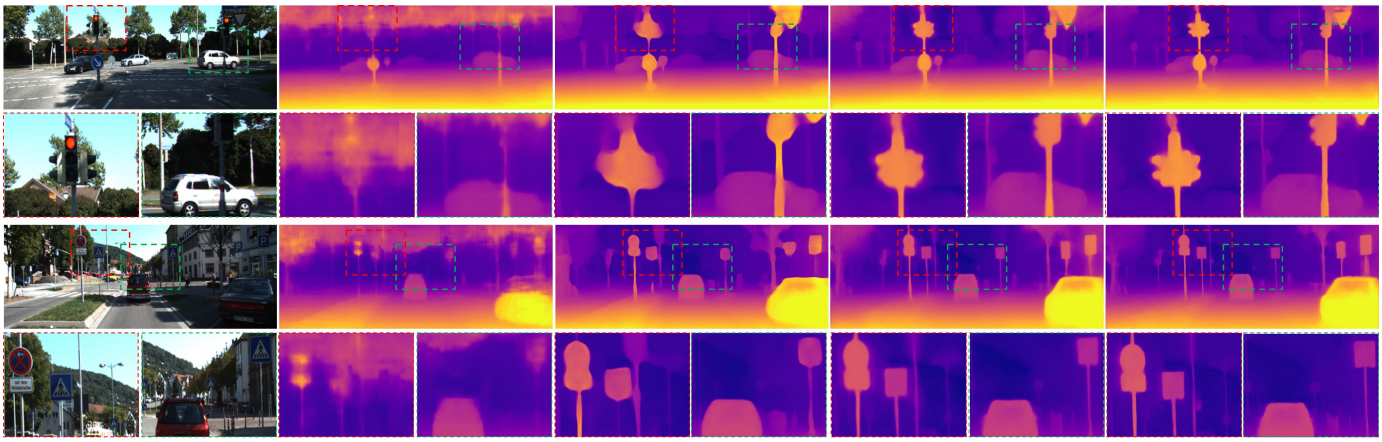

Fig. 1. Our DBoosterNet-t and DBoosterNet-e, built on our previous works [1] and [2], yield more consistent depth estimates and better preserve structural details compared to the previous self- and fully supervised methods. Our DBoosterNet-e produces the most accurate depth estimate.

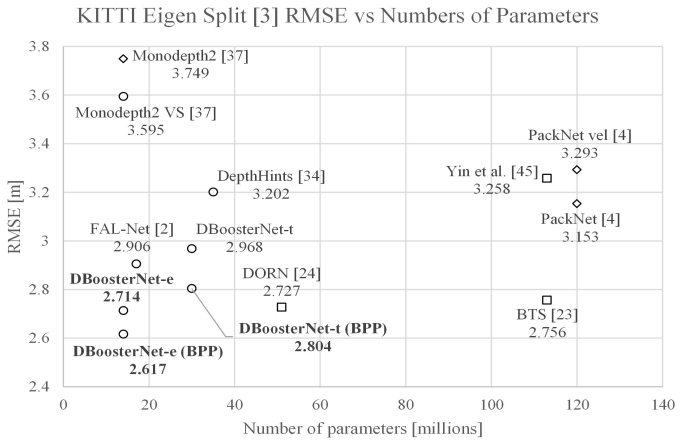

Fig. 2. Performance of self- and fully supervised single-image DE methods in terms of the root mean squared error (RMSE) on the KITTI Eigen test split with improved ground-truths [3]. ○: self-supervised from stereo. ◊: Self-supervised from videos. □: Fully-supervised. PackNet vel [4] is supervised with velocity ground-truth data for scale-aware SFM. Our self-supervised DBoosterNet-e with boosting post-processing (BPP) achieves state-of-the-art result. Our method without any BPP still outperforms previous works in terms of RMSE while keeping a lower network complexity in terms of number of parameters.

signals to improve the initial depth estimates of the input images. We summarize our main contributions as follows:

1) A novel two-stage learning (or training) strategy is proposed for DCNN-based single-image depth estimators, called DBoosterNets, that (i) leverage the well-posedness of depth priors from a first-stage DCNN trained for new view synthesis and (ii) extracts the "hidden" capabilities of the model via ambiguity boosting self-supervision in the second stage of training.
2) Ambiguity boosting, presented first, can be used during the training process as a loss function or at inference time to refine the depth estimates further, when the computational budget allows its use. Our ambiguity boosting loss strategy can also be understood as a type of confidence-guided data augmentation loss.
3) Our DBoosterNets are competitive and, in some cases, even better than previous supervised DE methods [24], [25], [45] and remarkably outperform the self-supervised [2], [4], [34] DE methods in most if not all metrics to the best of our knowledge.
4) Additionally, we propose an aggressive data augmentation strategy for training with random resizing, which is often avoided in most previous methods. We show that our DBoosterNets particularly benefit from the aggressive data augmentation due to our novel two-stage training strategy with ambiguity boosting.
5) Finally, our two-stage training strategy with ambiguity boosting is not limited to our DBoosterNets but can also be applied to any self-supervised indirect DE network with single or multiple view inputs that incorporates the estimation of a confidence mask in its output layers.

This paper is organized as follows: In Section 2, we discuss the related works and compare them to our method. This is followed by a detailed explanation of the proposed pipeline in Section 3. In Section 4, we show our experimental results for single-image DE, where our self-supervised method with ambiguity boosting *significantly* outperforms the previous SOTA methods by large margins from both a qualitative perspective (see Fig. 1) and a quantitative perspective (see Fig. 2) while keeping a considerably lower network complexity. Ablation studies are provided to support the effectiveness of each of our contributions in Section 4. Finally, we conclude our work in Section 5.

2 Related Works

We can divide self-supervised single image depth estimators into direct DE methods [4], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [46], [47], which directly estimate a single-channel depth map in their output layers, and indirect DE methods [1], [43], [44], [48], [49], that generate adaptive kernels or similar image representations in their output layers for the view synthesis task. In the indirect DE methods, a pseudo-depth map can be extracted from the predicted kernels.

2.1 Regression-based DE (Direct Estimation of Depth Values)

Self-supervised approaches for single-image direct DE often rely on backward-warping-based photometric reconstruc-

tion cost functions from stereo pairs or monocular video.

For the stereo cases, the early work of Garg *et al.* [27] proposed to learn single-image depths from binocular images by estimating a left depth map and using it to re-project the right-view into the left camera, where the reconstruction loss can be computed. Godard *et al.* [28] proposed to estimate both the left and right disparities from a given left input view and achieved considerable performance improvements by introducing the left-right (LR) consistency loss between the estimated left and right disparities.

Even when photometric reconstruction and consistency losses can be used to train DCNNs for DE in a self-supervised fashion, they impose ambiguities in the predicted depth maps, as every pixel is not visible in both the left and right views. Moreover, if a region is visible in both views, the non-Lambertian characteristics of natural scenes impose inconsistencies for stereo matching. To overcome these problems, Wong *et al.* [31] proposed to re-weight the LR consistency and smoothness terms to handle the mismatches between the left and right disparities. They used the local and global pixel-wise reconstruction residuals to estimate these weights or masks during training. In contrast, our previous work (Gonzalez and Kim [29]) proposed to learn to predict such masks, which can be interpreted as letting the network learn to predict the loss functions' errors, which we refer to as "ambiguity learning".

Poggi *et al.* [30] proposed "3Net" for learning from a trinocular camera setup with left, center, and right views. For stereo datasets (left and right cameras only), 3Net is trained with an alternating protocol for the learning of left-center and center-right disparities. During the inference step, the final center disparity is obtained by combining the lateral disparities following the post-processing step defined in earlier work [28].

Extending the learning of the single-image depth from binocular images, Pilzer *et al.* proposed what was termed "refine and distill" [32], which utilizes of three auto-encoder sub-networks. Given a right-view image as input, the first network estimates the left-view disparity to warp the right-view into the left-view camera. In a similar way, the second network re-warps the warped image back to the right-view camera. This creates a cycle during which inconsistencies are measured and fed to the third network, which generates a refined left-view disparity. In addition to photometric reconstruction losses, distillation losses between the refined disparity (or "teacher") and the first network output disparity (or "student") were proposed by Pilzer *et al.* [32].

Tosi *et al.* [33] and Watson *et al.* [34] independently proposed the use of traditional stereo matching techniques, such as semi-global matching (SGM), to provide additional supervision signals (or proxy labels) for their single-image based models, which are trained following a training strategy in the literature [28]. While Tosi *et al.* used an LR consistency check to distill the proxy labels, Watson *et al.* analyzed the photometric pixel reconstruction error such that the proxy label is only used when the DCNN-based depth estimator generates a worse warped image than the SGM depth.

For monocular video cases, the pioneering work of Zhou *et al.* [35] proposed two networks to learn single-image depth maps and relative poses in a self-supervised manner. Their networks were trained to minimize the photometric reconstruction errors between the target view (or center frame) and the projection of several reference views (past and future frames) into the target view camera, guided by the predicted depth and camera motions between the center and reference frames. Additionally, the photometric errors were weighted by a 'visibility mask', which roughly ignored the dynamic objects in the scene.

Godard *et al.* in their Monodepth2 [37], proposed to handle moving objects and occlusions in videos not by learning a mask, but by analyzing the loss functions during training. In their work, Godard *et al.* proposed to minimize the lowest pixel-wise photometric reconstruction errors among all reference frames and to only measure such errors for non-static pixels. Static pixels include objects moving at the same speed of the camera, or pixel regions with high homogeneity and/or infinite depth. Zhou *et al.* [38] noted that most monocular DE methods tend to perform inference on sub-sampled input images and argued that this degrades the performance. Zhou *et al.* proposed a dual-network type of architecture for low- and full-resolution feature extraction and trained it on full-resolution images with the training strategy devised by Godard *et al.* [37].

Other recent works propose to combine multiple low-level vision tasks to reinforce the ill-posed learning of depth from monocular videos. In their "Competitive Collaboration" [36], Ranjan *et al.* proposed to learn depth, camera motion, optical flow, and motion segmentation by enforcing spatio-temporal constraints and treating each pixel in the target views either as an independently moving pixel or a rigid pixel. Gordon *et al.* [39] proposed to learn not only camera poses and depths in a manner similar to [35] but also relative object motion, camera intrinsics and lens distortion. Gordon *et al.* used a pre-trained semantic segmentation network to produce a "likely moving object" mask to detect and potentially ignore pixels belonging to the classes that usually include moving objects (e.g., cars, trucks, bikers).

In their recent work, Guizilini *et al.* proposed a large auto-encoder network with approximately 120M parameters with symmetric 3D packing and unpacking modules, called PackNet [4]. Their 3D-packing/unpacking modules incorporate sub-pixel convolutions and deconvolutions [50] instead of simple striding, pooling, or interpolation, allowing for fine detail preservation. Guizilini *et al.* followed an earlier training strategy [37] for learning from videos but proposed an optional velocity supervision loss that allows for the learning of scale-aware structure-from-motion (SFM) for metrically accurate camera poses and depths. In their later work, Guizilini *et al.* [47] proposed to support the depth estimation task with semantic features by incorporating a pre-trained (and fully-supervised) semantic segmentation network, amounting to a total of 140M parameters from PackNet. The semantic features are injected into the decoder side of their PackNet [4] in a multi-scale manner, improving the structural quality of the predicted depths.

2.2 Kernel-based DE (Indirect Estimation of Depth Values)

DCNNs can be effectively utilized for single-image based novel view synthesis when adaptive kernels are adopted

in the networks' output layers. Such networks appear to learn strong 3D representations of scenes, as can be observed in their kernel activations [1], [43], [48], [49]. The early work of Xie *et al.* introduced "Deep3D" [43], a non-fully convolutional neural net for fixed baseline stereoscopic view synthesis. Their network generates a "disparity probability volume", which can be interpreted as a 1D convolutional kernel [1] that samples and blends pixels from horizontally shifted versions of the input image. The weighted sum of the probability volume's planes reveals a disparity map [1], [43], [48]. The work of Niklaus *et al.* [44] proposed an approximation of an N×N adaptive kernel by 1×N and N×1 adaptive separable kernels. Despite the fact that it was originally designed for video frame interpolation, it has been used for stereoscopic view synthesis [1], [48].

The later work of Gonzalez and Kim [1] proposed a "t-shaped" kernel with adaptive dilations, which is well-posed for stereoscopic view synthesis. Their t-shaped kernel assigns more sampling positions to the kernel "wing" that matches the motion of the camera for the synthesis of a right or left-view image. The adaptive dilation property of their kernel allows the network to synthesize remote objects in a novel view with small dilation kernels and nearby objects with large dilation kernels. Gonzalez and Kim [1] proposed to extract the disparity information embedded in the long "wing" of their dilation-adaptive "t-shaped" kernel as depth prior knowledge and further refined it with a secondary network and achieved high performance for self-supervised monocular DE tasks on the KITTI2015 dataset [51]. However, we note that the use of conventional photometric and consistency losses in their shallow refinement network limit the performance of their method, as these mainly smooth out the disparity prior but do little to improve the predicted depth quality. Additionally, Gonzalez and Kim proposed the use of ambiguity masks to guide their post-processing step, which is reasonable as this strategy uses the predicted loss or "ambiguity" roughly to remove erroneous depth values. In contrast, in this paper, we propose ambiguity boosting, which enforces consistency among multiple input transformations during *training*, not as a post-processing step only. It should be noted that ambiguity learning as described in our previous work [1] was incorporated to aid directly in the learning of depth estimates with fewer artifacts compared to an earlier method [29], while ambiguity boosting here aims to generate boosted depth predictions that act as proxy-labels in our ambiguity boosting loss, with not only flipped versions of the input image but also with several transformations of the input (e.g., upscaled, downscaled, and flipped-downscaled versions). Additionally, in this paper, the ambiguity masks are estimated in a separate output branch, and their only objective is to learn the warping loss, not to attempt to reduce occlusion artifacts in the predicted depths. Ambiguity boosting allows our models to achieve superior performance by learning from themselves.

In our recent work we proposed "FAL-net: Forget About the LiDAR" [2] for the self-supervised training of depth estimators with a two-stage training strategy for learning monocular depth from stereo pairs that exploits mirrored exponentially quantized disparity representations (called MED) for accurate single image DE. Such MED representations are manipulated to obtain realistic occlusion masks,

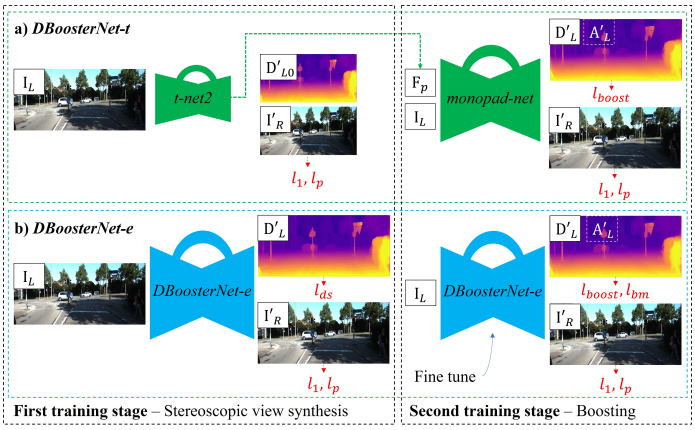

Fig. 3. Overview of the proposed two-stage training strategy with boosting for two different network designs: (a) the DBoosterNet-t and (b) the DBoosterNet-e.

which are used to ignore the occluded regions due to parallax in the loss functions. In this previous work, we showed that SOTA performance could be achieved by light and straightforward networks trained under our earlier method [2]. However, that method [2] is limited, as the computed occlusion masks are not perfect, which also affects the "mirror loss" in that study [2]. The mirror loss provides self-supervision signals to the left-occluded regions, which are not visible in the right view, via a mirror disparity map estimated from a horizontally flipped input view. These regions are ignored in the photometric reconstruction losses, commonly leading to depth artifacts on the left-sides of scene objects. It is important to note that the mirror disparity map is obtained from a fixed copy of the network trained on the first training stage only, which limits the quality of the self-supervisory signal. In this paper, we propose the use of "ambiguity boosting" to enhance this type of supportive disparity estimate during the training phase, and to provide boosted occlusion masks for better guidance when learning accurate depth estimations.

3 METHOD

We define a new two-stage training procedure for learning single view depth from stereo pairs (left and right views) with ambiguity boosting. A high-level overview of our two-stage training strategy with boosting is described in Algorithm 1 and depicted in Fig. 3. In the first stage, we learn to estimate a highly well-posed depth prior via a stereoscopic view synthesis task by training our DBoosterNets with a synthesis loss l_{syn} (which is a combination of l_1 and perceptual losses [52]) to generate a synthetic right or left view from a center view input image. Once trained, we keep a copy of our DBoosterNets with their parameters fixed during the second training stage. In the second stage, our DBoosterNets are also trained for right view synthesis with l_1 and perceptual losses, but more importantly, with our novel ambiguity boosting loss. With our ambiguity boosting loss, a boosted depth $\boldsymbol{D}^*$ is generated (fully or partially) from the fixed DBoosterNets by blending multiple depth estimates from various transformations of the input image, weighted by their corresponding estimated or computed ambiguity masks.

To show that the proposed two-stage training strategy with boosting generalizes to a variety of network architectures, we incorporate it into two different network designs, DBoosterNet-t and DBoosterNet-e, as shown correspondingly in Figures 3-(a) and -(b), where their architectures are extended from our previous works ([1] and [2], respectively). In particular, DBoosterNet-t in Fig. 3-(a) is constructed from two sub-networks: (i) t-net2 as an extension to t-net [1], which performs stereoscopic view synthesis with "t-shaped" dilation adaptive kernels in the first training stage, and (ii) a monocular prior-aware depth network, called monopad-net, which refines t-net2's depth prior estimate in the second training stage. On the other hand, our DBoosterNet-e in Fig. 3-(b), as an extension of FAL-net [2], is a single network with exponentially quantized disparity representations with its weights fine-tuned in the second stage of training.

Algorithm 1 High-level overview of our two-stage training strategy with ambiguity boosting. $\boldsymbol{D}^*$: boosted depth map. l_{syn}: Image synthesis loss. l_{boost}: ambiguity boosting loss. l_{amb}: ambiguity learning loss. f: resize and crop function

procedure DBOOSTERNET TRAINING
  Prepare training dataset
  Initialize DBoosterNet
  $epoch \leftarrow 0$
  while $epoch < 50$ **do** ▷ Training stage 1
    for $\boldsymbol{I}_L, \boldsymbol{I}_R$ in dataset **do**
      $r_f \leftarrow$ uniform(0.5, 2.5)
      $\boldsymbol{I}_L \leftarrow f(\boldsymbol{I}_L, r_f)$
      $\boldsymbol{I}_R \leftarrow f(\boldsymbol{I}_R, r_f)$
      Train DBoosterNet with l_{syn}
    $epoch \leftarrow epoch + 1$
  fix-DBoosterNet $\leftarrow$ DBoosterNet
  $epoch \leftarrow 0$
  while $epoch < 20$ **do** ▷ Training stage 2
    for $\boldsymbol{I}_L, \boldsymbol{I}_R$ in dataset **do**
      $r_f \leftarrow$ uniform(0.5, 2.5)
      $\boldsymbol{I}_L^d \leftarrow f(\boldsymbol{I}_L, \frac{2}{3} r_f)$
      $\boldsymbol{I}_L^{hd} \leftarrow flip(\boldsymbol{I}_L^d)$
      $\boldsymbol{I}_L^u \leftarrow f(\boldsymbol{I}_L, \frac{3}{2} r_f)$
      $\boldsymbol{I}_L \leftarrow f(\boldsymbol{I}_L, r_f)$
      $\boldsymbol{I}_L^h \leftarrow flip(\boldsymbol{I}_L)$
      $\boldsymbol{I}_R \leftarrow f(\boldsymbol{I}_R, r_f)$
      if $epoch = 0$ and DBoosterNet is 't' **then**
        Train DBoosterNet with l_{syn}
      else
        Get $\boldsymbol{D}^*(\boldsymbol{I}_L^h, \boldsymbol{I}_L^u, \boldsymbol{I}_L^d, \boldsymbol{I}_L^{hd})$ from fix-DBoosterNet
        Train DBoosterNet with $l_{syn} + l_{boost} + l_{amb}$
    $epoch \leftarrow epoch + 1$

Our ambiguity boosting loss function can also be considered as a type of confidence-guided data augmentation loss. To obtain these confidence or ambiguity masks, we explored the following: (i) having the network estimate the ambiguity masks in our DBoosterNet-t, as in earlier work [1], [29], and (ii) computing the ambiguity masks directly from the disparity probability volume representations in our DBoosterNet-e, as also described in the literature [2]. For our DBoosterNet-t, the monopad-net is trained for ambiguity learning [29]. However, in contrast to the aforementioned studies [29], [35], which also provide their networks with means by which to estimate confidence masks, we train our monopad-net to make itself directly learn to predict

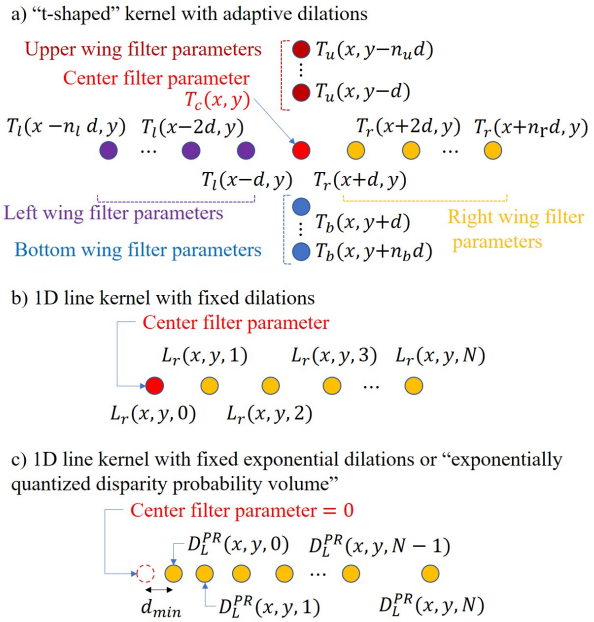

Fig. 4. Definition of different kernels for stereoscopic view synthesis (and indirect depth estimation) in DBoosterNets.

the warping errors between the input left-view and the left-camera-warped right-view. Such warped right-view is guided by the estimated left-view disparity, on which gradients are disabled when performing the warping operation to partially decouple disparity from ambiguity learning. Also note that for our DBoosterNet-t, we train it with l_{syn}-only in the first epoch of the second training stage to initialize the monopad-net, as shown in Algorithm 1.

Once fully trained, our DBoosterNets can produce very accurate depth estimates for single-view input images. Moreover, their performance capabilities can be improved further by adopting ambiguity boosting as a post-processing step, which we refer to as "boosting post-processing." This is possible as the ambiguity boosting process only requires a single image as input.

In the following subsections, we review the stereoscopic view synthesis models used in DBoosterNets, followed by the corresponding detailed network architectures, the details of the boosting process, loss functions, and the aggressive data augmentations proposed in this work.

3.1 Stereoscopic view synthesis via t-shaped kernels with adaptive dilations

The stereoscopic view synthesis in t-net2 is based on our previous work [1], where t-shaped kernels with adaptive dilations were presented. The t-net in study [1] showed that highly well-posed depth priors can be implicitly learnt using t-shaped kernels with adaptive dilations. The t-shaped kernels depicted in Fig. 4-(a) have a cross shape "+" with one wing longer than the other three. The longer wing points to the camera movement (leftward or rightward) and encodes the disparity information. The shorter wings encode occlusion information, as they try to fill-in the gaps left by synthetic pixels that cannot be generated by sampling the contents along the longer wing.

In this paper, t-net [1] is extended to t-net2 by further considering camera grid information, an improved network

structure, and an extended disparity range (from 152 to 300 pixels). Following the original work [1], the synthetic right view $\boldsymbol{I}'_R$ at pixel location $\boldsymbol{p}$ can be generated from the left view input $\boldsymbol{I}_L$ by

$$\boldsymbol{I}'_R(\boldsymbol{p}) = \sum_{i=1}^{N} \boldsymbol{W}_i(\boldsymbol{p}) \left[\boldsymbol{I}_L * \boldsymbol{T}_i\right](\boldsymbol{p}), \tag{1}$$

where $\{\boldsymbol{T}_i\}$ is a set of N t-shaped kernels with N different dilations, $*$ is the convolution operator, and $\{\boldsymbol{W}_i\}$ is the set of N blending weights where $\boldsymbol{W}_i$ is the i-th weight map applied to the convolution of $\boldsymbol{I}_L$ with the i-th t-shaped kernel $\boldsymbol{T}_i$. In other words, $\{\boldsymbol{W}_i\}$ assigns a proper dilation to each output pixel. Note that $\{\boldsymbol{T}_i\}$ and $\{\boldsymbol{W}_i\}$ effectively approximate a t-shaped kernel with adaptive dilations; these are estimated by t-net2, as described by

$$[\{\boldsymbol{T}_i\}, \{\boldsymbol{W}_i\}, \boldsymbol{F}_p] = \text{t-net2}(\boldsymbol{I}_L, d_{max}; \Theta_t), \tag{2}$$

where d_{max} is the maximum disparity hyper-parameter, Θ_t denotes the learnable network parameters of t-net2, and $\boldsymbol{F}_p$ represents the depth prior features to be fed later into the monopad-net. After convergence, t-net2 learns to assign small dilations to distant objects and large dilations to nearby objects, as such assignments better minimize the reconstruction losses, as shown in the aforementioned study [1]. Following that study [1], a disparity prior $\boldsymbol{D}'_{L0}$ can be obtained from the longer wings of the t-shaped kernels by a weighted sum of the corresponding kernel element values, as given by

$$\boldsymbol{D}'_{L0}(\boldsymbol{p}) = d_{max} \sum_{i=0}^{N} \boldsymbol{W}_i(\boldsymbol{p}) \frac{i}{N} \sum_{k=0}^{n_r-1} \frac{k}{n_r - 1} \boldsymbol{T}_i^k(\boldsymbol{p}), \tag{3}$$

where $\boldsymbol{T}_i^k$ is the k-th kernel element of the longer wing of the i-th t-shaped kernel $\boldsymbol{T}_i$ and n_r is the number of kernel elements in the longer wing of the t-shaped kernel.

3.2 Stereoscopic view synthesis in monopad-net

To obtain a *full*-resolution depth map, the *simple-but-effective* monopad-net is incorporated with a plain 1D-line adaptive kernel in its output layer, which can be interpreted as the longer wing of a single t-shaped kernel, as depicted in Fig. 4-(b). We remove the short wings of the kernel, as these will guide the network to approximate occluded regions, leaving holes in the predicted depths. In contrast to t-net2, only one kernel with a fixed dilation level is used in monopad-net to retain a manageable level of computation complexity at a full-resolution. Furthermore, we found that the t-shaped kernels with adaptive dilations are not suitable for depth refinement, as each i-th kernel contributes to the final depth in different and overlapping depth ranges, which appears to make the back-propagation of the gradients ambiguous.

View synthesis as performed by monopad-net is not designed to handle occlusions or to generate high-quality synthetic right views. Instead, it yields accurate depth estimates when trained in a self-supervised manner with ambiguity boosting. The synthetic right view $\boldsymbol{I}'_R$ in monopad-net, which is generated and used only for training, can then be simply described as

$$\boldsymbol{I}'_R(\boldsymbol{p}) = \left[\boldsymbol{I}_L * \boldsymbol{L}\right](\boldsymbol{p}), \tag{4}$$

where $\boldsymbol{L}$ is the estimated 1D-line kernel, which is generated by monopad-net and is given by

$$[\boldsymbol{L}, \boldsymbol{A}_L] = \text{monopad-net}(\boldsymbol{I}_L, \boldsymbol{F}_p, d_{max}; \Theta_{mp}), \tag{5}$$

where Θ_{mp} denotes the learnable parameters of the monopad-net. $\boldsymbol{A}_L$ is the estimated ambiguity mask, which is also part of the output of monopad-net and is further described in Sections 3.4 and 3.5.3. Note that monopad-net is fed with the depth prior features $\boldsymbol{F}_p$ from t-net2. The final disparity estimate $\boldsymbol{D}'_L$ can be extracted from the line kernel $\boldsymbol{L}$ by

$$\boldsymbol{D}'_L(\boldsymbol{p}) = d_{max} \sum_{k=0}^{N-1} \frac{k}{N-1} \boldsymbol{L}^k(\boldsymbol{p}), \tag{6}$$

where $\boldsymbol{L}^k$ and N are the k-th kernel element and the total number of kernel elements of the 1D-line kernel $\boldsymbol{L}$, respectively.

3.3 Stereoscopic view synthesis in DBoosterNet-e

For DBoosterNet-e, we adopt the image formation model [2], which can be understood as convolving the input image with a 1D line kernel with fixed exponential dilations, as shown in Fig. 4-(c), or as an exponentially quantized disparity probability volume [2]. In the aforementioned work [2], given a left input view $\boldsymbol{I}_L$, the CNN outputs the disparity probability logit volume $\boldsymbol{D}_L^L$. $\boldsymbol{D}_L^L$ can be soft-maxed to generate the left disparity probability volume $\boldsymbol{D}_L^{PL}$ or projected to the right-view (progressively plane by plane) and soft-maxed along the channel axis to form the right-from-left disparity probability volume $\boldsymbol{D}_L^{PR}$. $\boldsymbol{D}_L^{PR}$ can be employed for stereoscopic view synthesis with

$$\boldsymbol{I}'_R = \textstyle\sum_{n=0}^{N} g\left(\boldsymbol{I}_L, d_n\right) \odot \boldsymbol{D}_{L_n}^{PR}, \tag{7}$$

where $\odot$ indicates the element-wise multiplication, $g(\cdot)$ denotes a sliding of the input image to the left by d_n pixels, and N is the number of channels in $\boldsymbol{D}_L^{PR}$. On the other hand, $\boldsymbol{D}_L^{PL}$ can be used to extract the final disparity map $\boldsymbol{D}'_L$, which is implicitly learnt in the view synthesis task, as given by

$$\boldsymbol{D}'_L = \textstyle\sum_{n=0}^{N} d_n \boldsymbol{D}_{L_n}^{PL}. \tag{8}$$

Exponential quantization of disparity controlled by d_n) assigns distant and nearby depth quantization levels more uniformly than the linear scheme due to the inverse relationship between the depth and the disparity. The quantization levels or bins are then given by

$$d_n = d_{max} e^{\ln d_{max}/d_{min}(n/N-1)}, \tag{9}$$

where d_{min} is the minimum disparity hyper-parameter (set to zero in our t-net2 and monopad-net). For a fair comparison with [2], we set $d_{max} = 300$ and $d_{min} = 2$ in our DBoosterNet-e for all our experiments.

In contrast to monopad-net and in a similar manner to that of t-net2, the confidence or ambiguity masks in DBoosterNet-e can be computed directly from the probability volumes by

$$\boldsymbol{A}_L = \max\left(\textstyle\sum_{n=0}^{N} g\left(\boldsymbol{D}_{L_n}^{PR}, -d_n\right), 1\right) \tag{10}$$

Note that due to the shifting operation $g(\cdot)$, the result of the summation at a pixel location can be greater than 1; therefore, the *max* operation is applied to cap $\boldsymbol{A}_L$ to $[0, 1]$.

TABLE 1
Detailed Convolutional Backbone Used in DBoosterNets

In	Layer description	Out	Ch	Size
Conv0, $\boldsymbol{C}$, d_{mm}	Conv(s2), elu, ResBlock	Conv1	128	$\lfloor$H×W$\rfloor/2$
Conv1, $\boldsymbol{C}$, d_{mm}	Conv(s2), elu, ResBlock	Conv2	256	$\lfloor$H×W$\rfloor/4$
Conv2, $\boldsymbol{C}$, d_{mm}	Conv(s2), elu, ResBlock	Conv3	256	$\lfloor$H×W$\rfloor/8$
Conv3, $\boldsymbol{C}$, d_{mm}	Conv(s2), elu, ResBlock	Conv4	256	$\lfloor$H×W$\rfloor/16$
Conv4, $\boldsymbol{C}$, d_{mm}	Conv(s2), elu, ResBlock	Conv5	256	$\lfloor$H×W$\rfloor/32$
Conv5, $\boldsymbol{C}$, d_{mm}	Conv(s2), elu, ResBlock	Conv6	256	$\lfloor$H×W$\rfloor/64$
Conv6	Nearest, Conv, elu	Dec6	128	Conv5
Conv5, Dec6	Concat, Conv, elu	iConv6	256	Conv5
iConv5	Nearest, Conv, elu	Dec5	128	Conv4
Conv4, Dec5	Concat, Conv, elu	iConv5	256	Conv4
iConv4	Nearest, Conv, elu	Dec4	128	Conv3
Conv2, Dec4	Concat, Conv, elu	iConv4	256	Conv3
iConv3	Nearest, Conv, elu	Dec3	128	Conv2
Conv2, Dec3	Concat, Conv, elu	iConv3	256	Conv2
iConv2	Nearest, Conv, elu	Dec2	128	Conv1
Conv1, Dec2	Concat, Conv, elu	iConv1	128	Conv1
iConv1	Nearest, Conv, elu	Dec1	64	Conv0
Conv0, Dec1	Concat, Conv, elu	iConv0	64	Conv0
ResBlock				
$\boldsymbol{X}$	Conv(elu(Conv($\boldsymbol{X}$)))	$\boldsymbol{Y}_0$	$\boldsymbol{X}$	$\boldsymbol{X}$
$\boldsymbol{Y}_0$	elu($\boldsymbol{Y}_0$ + $\boldsymbol{X}$)	$\boldsymbol{Y}$	$\boldsymbol{X}$	$\boldsymbol{X}$

Conv: 3×3 convolutional layer with a stride of 1, otherwise specified. S2: stride of 2. Elu: exponential linear unit. Conv0: input features. $\boldsymbol{C}$: Camera grid. d_{mm}: two-channel tensor with min-max disparity hyper-parameters, respectively. $\lfloor H \times W \rfloor$ denotes the dimensions of the Conv0 features. Nearest: nearest interpolation to the size of the skip connection.

TABLE 2
Detailed DBoosterNet-e Network Architecture

In	Layer description	Out	Ch	Size
$\boldsymbol{I}_L$	Conv, elu, ResBlock	Conv0	32	H×W
Conv0, $\boldsymbol{C}$, d_{mm}	Backbone	iConv0	64	H×W
iConv0	Conv	$\boldsymbol{D}_L^L$	N	H×W
$\boldsymbol{D}_L^L$, d_n	$\sigma(\{g(\boldsymbol{D}_{L_n}^L, d_n)\}_0^N)$	$\boldsymbol{D}_L^{PR}$	N	H×W
$\boldsymbol{I}_L$, $\boldsymbol{D}_L^{PR}$, d_n	$\sum_{n=0}^N g\left(\boldsymbol{I}_L, d_n\right) \odot \boldsymbol{D}_{L_n}^{PR}$	$\boldsymbol{I}_R'$	3	H×W
$\boldsymbol{D}_L^L$	$\sigma(\boldsymbol{D}_L^L)$	$\boldsymbol{D}_L^{PL}$	N	H×W
$\boldsymbol{D}_L^{PL}$, d_n	$\sum_{n=0}^N d_n \boldsymbol{D}_{L_n}^{PL}$	$\boldsymbol{D}_L'$	1	H×W
$\boldsymbol{D}_R^{PL}$, d_n	$\sum_{n=0}^N g\left(\boldsymbol{D}_{L_n}^{PR}, -d_n\right)$	$\boldsymbol{A}_L$	1	H×W
n: Ch. number	$d_n = d_{max} e^{\ln \frac{d_{max}}{d_{min}} (\frac{n}{N}-1)}$	d_n	1	1

3.4 Network architectures

3.4.1 Convolutional Backbone

DBoosterNets adopt a fully-convolutional auto-encoder backbone with residual blocks in the encoder side and skip connections on its decoder side, as described in Table 1. Additionally, our backbone incorporates one feature containing the minimum and maximum disparities (d_{min} and d_{max}) as hyper-parameters and a two-channel camera grid $\boldsymbol{C}$ which contains the original relative pixel coordinates (x, y) of the input image before random cropping is applied during the training phase. These supportive features are incorporated in a multi-scale fashion at each strided convolution on the encoder side, as shown in Table 1. Feeding the camera grid to the networks is particularly useful for learning to predict depths of objects near the image borders, where objects tend to stretch due to their projection on the camera plane. On the other hand, feeding d_{min} and d_{max} is useful when concurrently training with multiple datasets [1] with different baselines (distance between cameras), where the ambiguities and occlusions are increased as the baseline becomes wider. While DBoosterNets adopt the same convolutional backbone, they differ in terms of their input and output layers, as explained in the following subsections.

TABLE 3
Detailed t-net2 Network Architecture

In	Layer description	Out	Ch	Size
$\boldsymbol{I}_L$	Conv, elu, ResBlock	Conv0	64	H×W
Conv0, $\boldsymbol{C}$, d_{mm}	Backbone (iConv1)	$\boldsymbol{F}_p$	128	$\lfloor$H×W$\rfloor/2$
$\boldsymbol{F}_p$	Conv, ch-wise softmax	$\boldsymbol{T}_1$	81	$\lfloor$H×W$\rfloor/2$
$\boldsymbol{F}_p$	Conv, ch-wise softmax	$\boldsymbol{T}_2$	81	$\lfloor$H×W$\rfloor/2$
$\boldsymbol{F}_p$	Conv, ch-wise softmax	$\boldsymbol{T}_3$	81	$\lfloor$H×W$\rfloor/2$
$\boldsymbol{F}_p$, $\boldsymbol{T}_1$, $\boldsymbol{T}_2$, $\boldsymbol{T}_3$	Conv, ch-wise softmax	$\boldsymbol{W}_i$	3	$\lfloor$H×W$\rfloor/2$

Channel(ch)-wise softmax denotes a softmax operation along the channel axis. ResBlock: See Table 1 for more details.

TABLE 4
Detailed monopad-net Architecture

In	Layer description	Out	Ch	Size
$\boldsymbol{I}_L$	Conv, elu, ResBlock	Conv0	32	H×W
$\boldsymbol{F}_p$	Conv, elu, Nearest	$\boldsymbol{F}_{pu}$	32	H×W
Conv0, $\boldsymbol{F}_{pu}$, $\boldsymbol{C}$, d_{mm}	Backbone	iConv0	64	H×W
iConv0	Conv	$\boldsymbol{L}_L$	N	H×W
iConv0, $\boldsymbol{D}_L$, $\boldsymbol{I}_L$	Conv, elu	$\boldsymbol{A}_0$	32	H×W
$\boldsymbol{A}_0$	Conv, sigmoid	$\boldsymbol{A}_L$	1	H×W

3.4.2 DBoosterNet-e

DBoostetNet-e takes as input the left view $\boldsymbol{I}_L$ and yields as output an exponentially quantized disparity logit volume $\boldsymbol{D}_L^L$, from which a synthetic right view $\boldsymbol{I}_R'$, a disparity map $\boldsymbol{D}_L'$, and a pseudo-occlusion (or "ambiguity") mask $\boldsymbol{A}_L$ can be computed following earlier work [2], as shown in Tab. 2.

3.4.3 DBoosterNet-t

DBoosterNet-t consists of a cascade of (i) t-net2 (described in Table 3) that generates a depth prior in the form of the half-resolution features $\boldsymbol{F}_p$ and (ii) monopad-net (depicted in Table 4) that generates a full-resolution depth map $\boldsymbol{D}_L'$ and ambiguity mask $\boldsymbol{A}_L$.

The t-net2. During training, t-net2 outputs a synthetic right view $\boldsymbol{I}_R'$ at half-resolution, which is bilinearly upscaled and compared to the ground-truth right view $\boldsymbol{I}_R$ in the image synthesis loss. It should be noted that during the second-stage of training and inference, t-net2 outputs the features $\boldsymbol{F}_p$ (shown in Table 3) to be used as our depth prior in the monopad-net to avoid the expensive computation of the synthetic right view $\boldsymbol{I}_R'$ and the disparity prior $\boldsymbol{D}_{L0}'$ in Eqs. 1 and 3, respectively.

Monopad-net. This sub-netork takes as input the left-view image $\boldsymbol{I}_L$ and the lower resolution depth prior features $\boldsymbol{F}_p$ from t-net2. The depth prior $\boldsymbol{F}_p$ is passed through a convolutional layer that reduces its number of channels, after which it is upscaled via nearest interpolation to the input image dimensions. The resulting prior features are concatenated with the early extracted features denoted by Conv0 and fed to the convolutional Backbone, as shown in Table 4. The backbone output (denoted by iConv0) is further divided into two branches, one for the estimation of the line kernel elements $\boldsymbol{L}_L$, which can be used for coarse

stereoscopic view synthesis and depth estimation, and the other for the ambiguity learning of $\boldsymbol{A}_L$.

3.5 Ambiguity Boosting

We propose to obtain an enhanced depth estimate by boosting the predicted depth with multiple forward passes of DBoosterNet (-t or -e) with various versions of the input image. As already pointed out in several previous works [1], [28], [30], [46], running the networks with flipped inputs generates slightly different depths, with occlusion artifacts due to parallax on opposite positions relative to the scene objects. Godard *et al.* [28] simply attempted to alleviate such depth artifacts by averaging the normal and flipped estimates, thus obtaining a slightly enhanced depth map. However, we note that such a simple element-wise averaging results in blur or overly-smooth depth estimates.

In this paper, we initially analyse the effects of multiple input transformations on the depth estimation networks and particularly observe that running a DBoosterNet on up-scaled inputs can generate better (more detailed) depths for the distant objects, while running it on down-scaled inputs can produce better (more consistent) depths for nearby objects. We observed that running the DBoosterNet on vertically flipped inputs makes learning slow, severely affecting the performance, at least for a training schedule that is comparable to those in previous works.

Contrary to previous works [1], [2], [28], [30], [34], we propose the generation of our boosted depths during training and use them as self-supervision signals, or depth proxy-labels, in our ambiguity boosting loss function. Furthermore, instead of simple element-wise averaging, we propose to guide the blending of the multiple depths for different input transformations with confidence or "ambiguity" masks for their respective depth estimates. The ambiguity masks $\boldsymbol{A}$ predicted by DBoosterNets approximate the warping errors (in our DBoosterNet-t) or represent the occluded regions relative to the predicted depth map (in our DBoosterNet-e); they can be used to blend two (or more) depth estimates of a single input image in a "guided" manner. When blending, lower weights are adaptively assigned to those pixels with higher predicted errors or occlusions, thus generating sharper depth maps as compared to plain element-wise averaging.

Fig. 5 depicts the proposed ambiguity boosting process with the identity, flip, upscale and flip-downscale input transformations. It can be observed that the down-scaling transformation produces more consistent depths for close-to-camera objects (red boxes), while up-scaling produces more detailed depths for distant objects (green boxes), where the green boxes are all magnified for better visualization in the zoomed regions on the bottom-right side of Fig. 5. Our ambiguity-boosted depth $\boldsymbol{D}^*$, which can be regarded as a proxy label in our ambiguity boosting loss during training or as a final output depth map during test time, is given by

$$\boldsymbol{D}^* = \sum_{i}^{N_{tr}} \frac{e^{\beta \boldsymbol{A}'_i}}{\sum_j^{N_{tr}} e^{\beta \boldsymbol{A}'_j}} \odot \frac{1}{s_i} \boldsymbol{D}'_i, \tag{11}$$

where N_{tr} denotes the number of transformations applied to the input image $\boldsymbol{I}_L$, and $\boldsymbol{D}'_i$ and $\boldsymbol{A}'_i$ are the correspondingly the predicted depth and ambiguity mask for the i-th transformed input image. The left-hand side of the element-wise multiplication operator $\odot$ indicates the element-wise softmax of the N_{tr} ambiguity masks, where $\beta = 2$ is empirically set to concentrate the distribution more on the most confident estimates. $1/s_i$ handles the possible scale change in transformation i. We utilized $N_{tr} = 5$ input transformations, in this case $\boldsymbol{I}_L$ (identity or no-transform), $\boldsymbol{I}_L^h$ (horizontal flip), $\boldsymbol{I}_L^d$ (down-scaling by a factor of 2/3), $\boldsymbol{I}_L^{hd}$ (horizontal flip followed by 2/3 down-scaling), and $\boldsymbol{I}_L^u$ (upscaling with a factor of 3/2). This down-scaling factor was selected to keep the details fine enough, while such an up-scaling factor was selected to maintain a reasonable level of computational complexity. It should also be noted that $\boldsymbol{I}_L^{hu}$ (horizontal flip followed by 3/2 upscaling) was excluded for computation simplicity, which does not significantly affect the fidelity of the estimated depths.

3.6 Loss functions

DBoosterNets are supervised with image synthesis loss l_{syn} in the first stage of training and with a compound loss which includes the synthesis and ambiguity boosting losses in the second training stage. Additionally, DBoosterNet-t is trained with "ambiguity learning" loss l_{amb} in the second training stage, yielding the full loss function l_t

$$l_t = l_{syn} + l_{boost} + l_{amb} \tag{12}$$

On the other hand, DBoosterNet-e is additionally trained with a boosted "mirror" loss l_{bmr}, resulting in the full loss l_e as given by

$$l_e = l_{syn} + l_{boost} + l_{bmr} \tag{13}$$

3.6.1 Image synthesis loss (l_{syn})

In conjunction with the l_1 loss, the quality of the generated synthetic right-view $\boldsymbol{I}'_R$ can be enhanced by minimizing its distance against the ground-truth $\boldsymbol{I}_R$ in the deep feature space of a pre-trained classification network, known as the 'perceptual loss' [52]. We use the output of the first three pooling layers ($l = 1, 2$ and 3) of the pre-trained VGG19 [53], denoted by $\phi^l(\boldsymbol{I})$ in our synthesis loss l_{syn}, as given by

$$l_{syn} = ||\boldsymbol{I}_R - \boldsymbol{I}'_R||_1 + \alpha_p \sum_{l=1}^{3} ||\phi^l(\boldsymbol{I}_R) - \phi^l(\boldsymbol{I}'_R)||_2^2 \tag{14}$$

where α_p weights the contribution of the perceptual loss and was set to 0.01 to balance the contribution between l_1 and the perceptual terms empirically.

3.6.2 Ambiguity boosting loss (l_{boost})

The ambiguity boosting loss guides the predicted disparity $\boldsymbol{D}'_L$ so that it is similar to $\boldsymbol{D}^*_L$ and is given by

$$l_{boost} = \frac{\alpha_{l_1}}{max(\boldsymbol{D}^*_L)} ||\boldsymbol{D}^*_L - \boldsymbol{D}'_L||_1 + \alpha_p \sum_{l=1}^{3} ||\phi^l(\boldsymbol{D}^*_L) - \phi^l(\boldsymbol{D}'_L)||_2^2 \tag{15}$$

where $max(\boldsymbol{D}^*_L)$ is the maximum value of the boosted disparity and keeps the loss in a range between 0 and 1. In a manner similar to that of our image synthesis loss, $\boldsymbol{D}'_L$ and $\boldsymbol{D}^*_L$ are compared in the VGG19 deep feature space. We set α_{l_1} to 1 and 0 for DBoosterNet-t and -e, respectively.

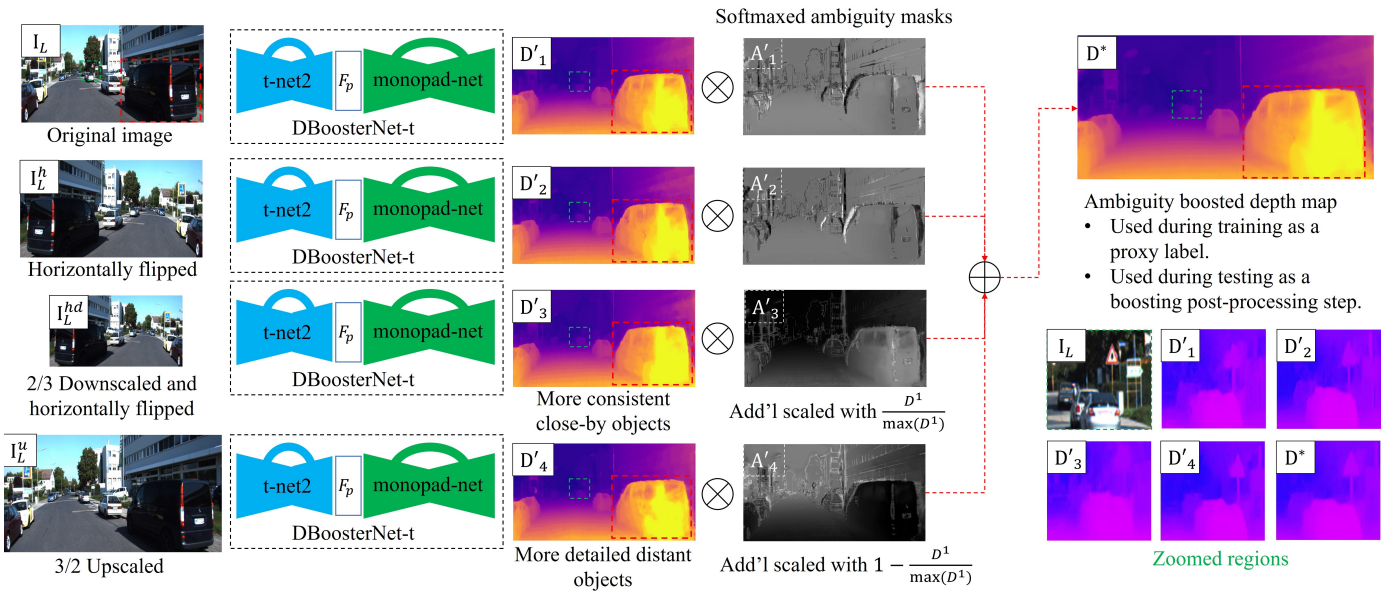

Fig. 5. Ambiguity boosting process. DBoosterNet is run with multiple versions of the input image to generate multiple depth estimates. These estimates are combined into the boosted depth map $\boldsymbol{D}^*$, weighted by the predicted ambiguity masks. Running DBoosterNet with horizontally flipped inputs helps to alleviate depth artifacts due to parallax. Running with downscaled inputs generates more consistent depth estimates for the nearby objects as shown by the red boxes. Running with upscaled inputs generates considerably more detailed predictions for the distant objects, as depicted by the green boxes.

3.6.3 *Ambiguity learning loss (l_{amb})*

In DBoosterNet-t, instead of computing a pseudo-occlusion mask from the line kernel activations, we explore the estimation of the ambiguity mask $\boldsymbol{A}_L$ and propose an ambiguity learning loss functions as a combination of the warping loss l_w and binary cross-entropy loss l_{bce}, as given by

$$l_{amb} = \alpha_w l_w + \alpha_{bce} l_{bce} \tag{16}$$

where α_w and α_{bce} are empirically set to 0.01 and 0.05, respectively. The ambiguity learning loss encourages the network to learn the warping error between the input left view and the right view warped into the left camera $\boldsymbol{I}'_{R\to L}$, which is guided by the predicted disparity $\boldsymbol{D}'_L$ and can be used as an indicator of the quality of $\boldsymbol{D}'_L$. The warping loss (l_w) and the warped image ($\boldsymbol{I}'_{R\to L}$) are given by

$$l_w = \alpha_p \sum_{l=1}^{3} ||\phi^l(\boldsymbol{I}_L) - \phi^l(\boldsymbol{I}'_{R\to L})||_2^2 \tag{17}$$

$$\boldsymbol{I}'_{R\to L} = (1 - \boldsymbol{A}'_L) \odot \boldsymbol{I}_L + \boldsymbol{A}'_L \odot (g_{ng}(\boldsymbol{I}_R, \boldsymbol{D}'_L)), \tag{18}$$

where $g_{ng}(\cdot)$ warps the right view into the left camera with no-gradients propagated through $\boldsymbol{D}'_L$, as we prevent this warping loss from supervising our depth estimation branch. To measure the warping error, we empirically find that the perceptual loss in the feature domain is more robust than the plain l_1 or l_2 loss in pixel domain.

Similar to earlier work [29], [35], the cross-entropy loss l_{bce} is adopted to guide monopad-net to generate ambiguity masks with values close to 1 for regions with low photometric and consistency errors. This is expressed as follows:

$$l_{bce} = ||\log \boldsymbol{A}'_L||_1 \tag{19}$$

Both l_w and l_{bce} help monopad-net learn the ambiguity masks $\boldsymbol{A}_L$ in terms of the probabilities as the pixel-wise confidences for the estimated depth values.

3.6.4 *Boosted Mirror loss (l_{bmr})*

Following an earlier method [2], DBoosterNet-e incorporates the mirror loss in its second stage of training. The mirror loss provides self-supervisory signals to the left-occluded regions that are ignored in the photometric reconstructions. However, instead of using the plain mirror loss [2], we propose a boosted mirror loss. In our boosted mirror loss, such self-supervision comes from our boosted disparity $\boldsymbol{D}^*$, and the left-occluded regions are denoted by the boosted ambiguity mask $\boldsymbol{A}^*_L$, which is given by

$$\boldsymbol{A}^*_L = \boldsymbol{A}_L \odot \boldsymbol{A}^d_L \odot \boldsymbol{A}^u_L, \tag{20}$$

where $\boldsymbol{A}^d_L$ and $\boldsymbol{A}^u_L$ are the ambiguity masks of the down- and upscaling transformations. The boosted mirror loss is then expressed as shown below.

$$l_{bmr} = \frac{1}{max(\boldsymbol{D}^*_L)} ||(1 - \boldsymbol{A}^*_L) \odot (\boldsymbol{D}'_L - \boldsymbol{D}^*_L)||_1 \tag{21}$$

3.7 Aggressive data augmentations

Concerns have recently been raised that CNNs are not actually learning from data but are instead finding shortcuts in the datasets [54]. For the task of monocular depth estimation, finding shortcuts instead of learning could well be the case, as the networks could be learning road plane detection as a shortcut for learning depth as suggested by Dijk and Croon [55]. In their work, Dijk and Croon showed that deep networks mainly rely on the heights of the objects relative to the road plane to infer depth, which can be seen as a reasonable cue for depth estimation but would also be limited to driving car perspectives.

To challenge this, we propose the use of not only random horizontal flips and luminance shifts as in most previous works that are self-supervised but also aggressive random resizing followed by a random crop during training. When

training on the KITTI [56] dataset, we apply random bicubic resizing to the input images between a factor of 0.5 and a factor of 2.5. When training on the CityScapes dataset [57], due to its much higher resolution than KITTI [56], we use a resize factor between 0.5 and 1.1. Large upscale factors followed by random crops can generate augmented samples that do not contain roads, forcing the networks to learn more general depth cues instead. It is reasonable to assume that most objects will be located not too far and not too close to the cameras in natural images. This implies that most disparity estimates will fall into a limited range that is likely to exclude the upper and lower disparity levels in indirect DE methods. Then, aggressive random resizing should allow the indirect DE methods to expose their lower and upper disparity levels during training since downscaled images will make use of lower disparity levels, and upscaled images will make use of higher disparity levels. This is not the case of direct DE methods [4], [28], [33], [34], [37] that might not be able to gain anything from random resizing unless they are fine-tuned with our proposed ambiguity boosting method that requires the network to be familiar with varying scales of input images.

4 Experiments and Results

To quantify the impact of each of our contributions and support the effectiveness of our two-stage training strategy with boosting, we performed extensive ablation studies on the challenging KITTI [3] dataset (see Table 5). Additionally, to show that the effectiveness of our method is not tied to a specific dataset, we train our DBoosterNets with the CityScapes [57] and KITTI datasets concurrently. We compare our method against recent SOTA methods on the challenging KITTI Eigen test split [58] (see Table 7) and show that DBoosterNets achieve the best performance among all previous methods. To demonstrate the generalization capabilities of DBoosterNets, we performed qualitative and quantitative comparisons on the Make3D [59] dataset (see Fig. 11 and Table 10), for which the proposed method obtained the best performance among competing methods.

4.1 Datasets

4.1.1 The KITTI dataset

The KITTI dataset [56] consists of several 384×1280 stereo frames captured from a driving perspective. We train all of our models on the widely used KITTI Eigen training split defined in earlier work [58], consisting of 22.6k image pairs across several scenes where static-car captures were avoided to prevent repeated frames. We evaluate our models on the original [58] and improved [3] KITTI Eigen test splits, which consist of 697 and 652 images with projected LiDAR ground-truths, respectively, from scenes excluded from the KITTI Eigen training split. The improved test split [3] is made of denser GT's that result from accumulating LiDAR points from 5 consecutive frames that are filtered with stereo SGM depths to remove moving objects. When evaluating on the KITTI Eigen test split, we adopt the standard practices of using the central crop [27], [58] and capping the predicted depths from 1m to 80m during the evaluation.

4.1.2 The CityScapes dataset

Compared to KITTI [56], CityScapes [57] is a driving dataset with higher-resolution. We use the train, train_extra and test directories of the CityScapes dataset for training with 24.5k additional stereo pairs. After removing image borders and car hoods, the final resolution of a CityScapes image has a size of 799×1948. Inspired by earlier work [1], we perform multi-dataset training, which is possible as all of our network architectures adopt the max disparity d_{max} as an additional input channel, which allows us to take into account the baseline difference between the KITTI and CityScapes datasets.

4.1.3 The Make3D dataset

We present additional quantitative and qualitative comparisons for the results on the Make3D [59] Test134 dataset, which consists of 134 high-resolution RGB outdoor images with very low-resolutions and not perfectly aligned depth GT depths. Note that our networks and the relevant self-supervised methods compared here were not trained on the Make3D [59] training dataset.

4.2 Implementation details

We define a two-stage training procedure for DBoostetNets. For both training stages we use the Adam [60] optimizer with common βs for the regression task (0.5 and 0.999) with an initial learning rate of 0.0001 and the following data augmentations on-the-fly: random resize with a factor between 0.5 and 2.5 conditioned by a subsequent 192×640 (patch size) random crop; random horizontal flip; random gamma; and random brightness and individual RGB color brightness. For all our experiments, the max disparity d^K_{max} was set to 300 pixels when feeding KITTI images, which have a baseline (distance between cameras) of 54cm. When feeding the CityScapes images, with a baseline of 22cm, the maximum disparity was set to $d^{CS}_{max} = (22/54)d^K_{max}$, similar to the method used earlier [1]. Because the images are rectified, feeding the max disparity allows the network to treat images from different focal-length cameras approximately as if they were all from the same camera but simply up-scaled or down-scaled. Particularly, in the first training stage, DBoosterNets are trained for 50 epochs with a batch size of 8, halving the learning rate at epochs 30 and 40. On the other hand, in the second stage of training we adopt a batch size of 6 and train for 20 epochs, halving the learning rate at epoch numbers 10 and 20.

The dilation adaptive t-shaped kernel in our t-net2 was set to have 16 elements on its short wings and 32 elements on its longer wing. The number of t-shaped kernels used to generate a synthetic right-view was set to 3. The line-shaped kernel in monopad-net was set to a size of 32 elements. For our DBoosterNet-e, we used 49 levels (as in [2]) in its exponentially quantized probability volumes. We implemented our method in PyTorch on a TITAN XP GPU with an Intel core i7 CPU and 64GB RAM. Under these conditions, our best model, the DBooterNet-e, trains in approximately 3.5 days for the first stage of training, and in 1.5 days for the second stage of training.

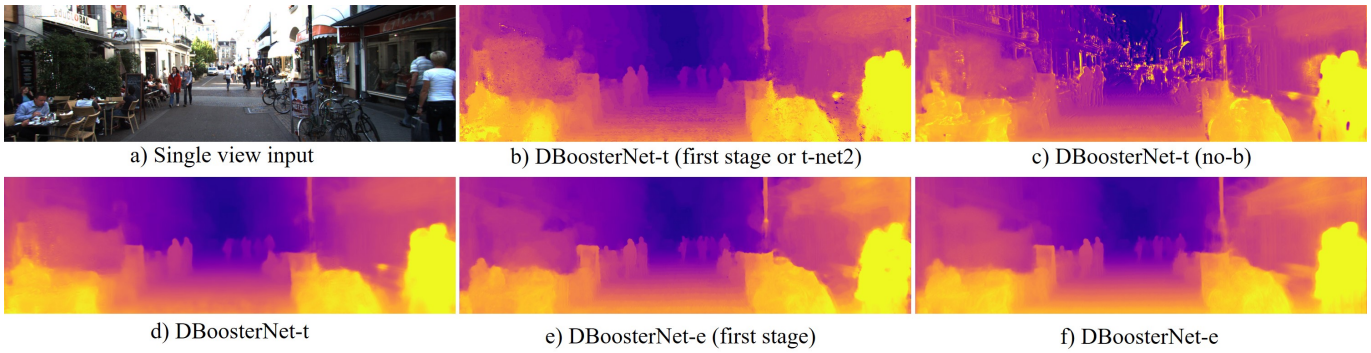

Fig. 6. Qualitative comparison between depth estimates from DBoosterNets at different training stages.

TABLE 5
Ablation Study on the KITTI Eigen Split [58]

DBoosterNet	Data	rel	sq rel	rmse	$\log_{rmse}$	a^1	a^2	a^3
Ablation studies on the first stage of training								
-t (t-net)	K	0.083	0.408	3.302	0.123	0.928	**0.987**	**0.997**
-t (t-net, ext)	K	**0.080**	**0.398**	3.271	**0.121**	0.930	**0.987**	**0.997**
-t (t-net)	K+CS	0.086	0.405	**3.269**	0.122	0.930	**0.987**	**0.997**
-t (t-net, ext)	K+CS	0.083	0.411	3.292	**0.121**	**0.931**	**0.987**	**0.997**
-e	K	0.072	0.307	3.014	0.113	0.935	0.989	**0.998**
-e (ext)	K	0.073	0.307	2.977	0.112	0.937	0.990	**0.998**
-e (no xy, mm)	K+CS	0.076	0.302	2.853	0.109	0.942	0.989	0.997
-e (no xy)	K+CS	0.075	0.300	2.819	0.108	0.944	0.989	0.997
-e	K+CS	0.070	0.271	2.792	0.105	0.944	**0.992**	**0.998**
-e (ext)	K+CS	**0.068**	**0.264**	**2.755**	**0.103**	**0.948**	**0.992**	**0.998**
Ablation studies on the second stage of training								
-t	K	0.079	0.348	3.087	0.117	0.932	0.989	0.997
-t (PP)	K	0.078	0.330	3.015	0.115	0.934	0.990	**0.998**
-t (BPP)	K	0.076	0.310	2.928	0.112	0.936	0.991	**0.998**
-t (ext)	K	0.074	0.334	3.044	0.114	0.936	0.989	0.997
-t (ext, BPP)	K	0.070	0.301	2.908	0.109	0.940	0.990	**0.998**
-t (no-b)	K+CS	0.109	1.001	6.228	0.252	0.860	0.946	0.972
-t	K+CS	0.073	0.319	2.968	0.110	0.939	0.990	**0.998**
-t (PP)	K+CS	0.072	0.302	2.889	0.108	0.941	0.991	**0.998**
-t (BPP)	K+CS	0.069	0.283	**2.804**	0.105	0.944	0.991	**0.998**
-t (ext)	K+CS	0.072	0.320	3.010	0.109	0.942	0.990	**0.998**
-t (ext, BPP)	K+CS	**0.068**	**0.282**	2.824	**0.103**	**0.947**	**0.992**	**0.998**
-e	K	0.070	0.287	2.878	0.109	0.938	0.991	**0.998**
-e (PP)	K	0.069	0.279	2.844	0.107	0.940	0.991	**0.998**
-e (BPP)	K	0.068	0.271	2.791	0.105	0.943	0.992	**0.998**
-e (ext)	K	0.070	0.298	2.916	0.109	0.940	0.991	**0.998**
-e (ext, BPP)	K	0.068	0.282	2.825	0.106	0.943	0.992	**0.998**
-e (as [2])	K+CS	0.069	0.267	2.793	0.105	0.945	0.992	**0.998**
-e (no-b)	K+CS	0.066	0.274	2.838	0.103	0.947	0.992	**0.998**
-e	K+CS	0.065	0.255	2.688	0.099	0.950	0.993	**0.998**
-e (PP)	K+CS	0.064	0.247	2.643	0.099	0.950	0.993	**0.998**
-e (BPP)	K+CS	**0.062**	**0.234**	**2.577**	**0.096**	0.953	**0.994**	0.998
-e (ext)	K+CS	0.064	0.258	2.714	0.099	0.952	0.993	**0.998**
-e (ext, BPP)	K+CS	**0.062**	0.242	2.617	**0.096**	**0.955**	**0.994**	0.998

*Evaluations performed on the KITTI Eigen test split with improved GT [3]. K: Training on the KITTI [56] dataset. K+CS: simultaneous training on KITTI [56] and CityScapes [57]. BPP: boosting post-processing. PP: post-processing step as defined earlier [28]. no-b: no ambiguity boosting loss. ext: aggressive data augmentation. Metrics are the lower the better and the higher the better. **Best** and second best metrics.*

4.3 Ablation studies

We present extensive ablation studies in Table 5 for our DBoosterNets with one and two stages of training, with and without boosting, under normal and aggressive data augmentations, and with single or multiple datasets. As confirmed in Table 5, training with the proposed two-stage strategy with ambiguity boosting significantly improves the performance of our DBoosterNets in comparison with that of the first training stage.

For DBoosterNet-t, an a^1 accuracy gain of 1.1% is observed when training with ambiguity boosting. The performance of DBoosterNet-t in the first training stage (or t-net2) is considerably lower than that of the full DBoosterNet-t, but still competitive against recent methods, although it is only an intermediate result. A depth estimate from t-net2 is depicted in Fig. 6-(b). Note that severe depth artifacts are present at object borders and in highly homogeneous regions. Training DBoosterNet-t (monopad-net) with the image synthesis loss only in the second learning stage, denoted in Table 5 as (no-b), does not help to improve the performance against the depth prior estimate from t-net2. This occurs because the 32-element line kernel in monopad-net is not able to learn the correct view-synthesis mapping, as the 300-pixel disparity is distributed to only 32 samples, leading to heavy occlusion and discretization artifacts, as depicted in Fig. 6-(c). When we add our ambiguity boosting loss, substantial performance improvements are achieved consistently in all metrics, as shown in Table 5. Our ambiguity boosting loss not only improves the quantitative metrics but also qualitatively helps to generate more consistent depths as depicted in Fig. 6-(d).

DBoosterNet-e, even when it already shows good performance in the first training stage, presents an accuracy gain of 0.7% in the a^1 metric when training with ambiguity boosting. To compare with our previous work [2], we trained our DBoosterNet-e following that strategy [2], as indicated by (as [2]) in Table 5, and observed considerably worse error metrics (rel, sq rel, rms and $\log_{rms}$). Additionally, we trained DBoosterNet-e without our boosting loss but still with the boosted mirror loss, as denoted by Net-e (no-b). By partially supervising our DBoosterNet-e with the mirror loss, we still observe good accuracy metrics but also slightly higher error metrics. DBoosterNet-e yields the best performance in all metrics in comparison with DBoosterNet-t, showing that our two-stage training strategy with boosting benefits any network under training, even when it already shows good performance in the first stage of training. As can be observed in Figures 6-(e) and (f), training with ambiguity boosting not only sharpens the depth predictions at the object contours but also removes occlusion and discretization artifacts.

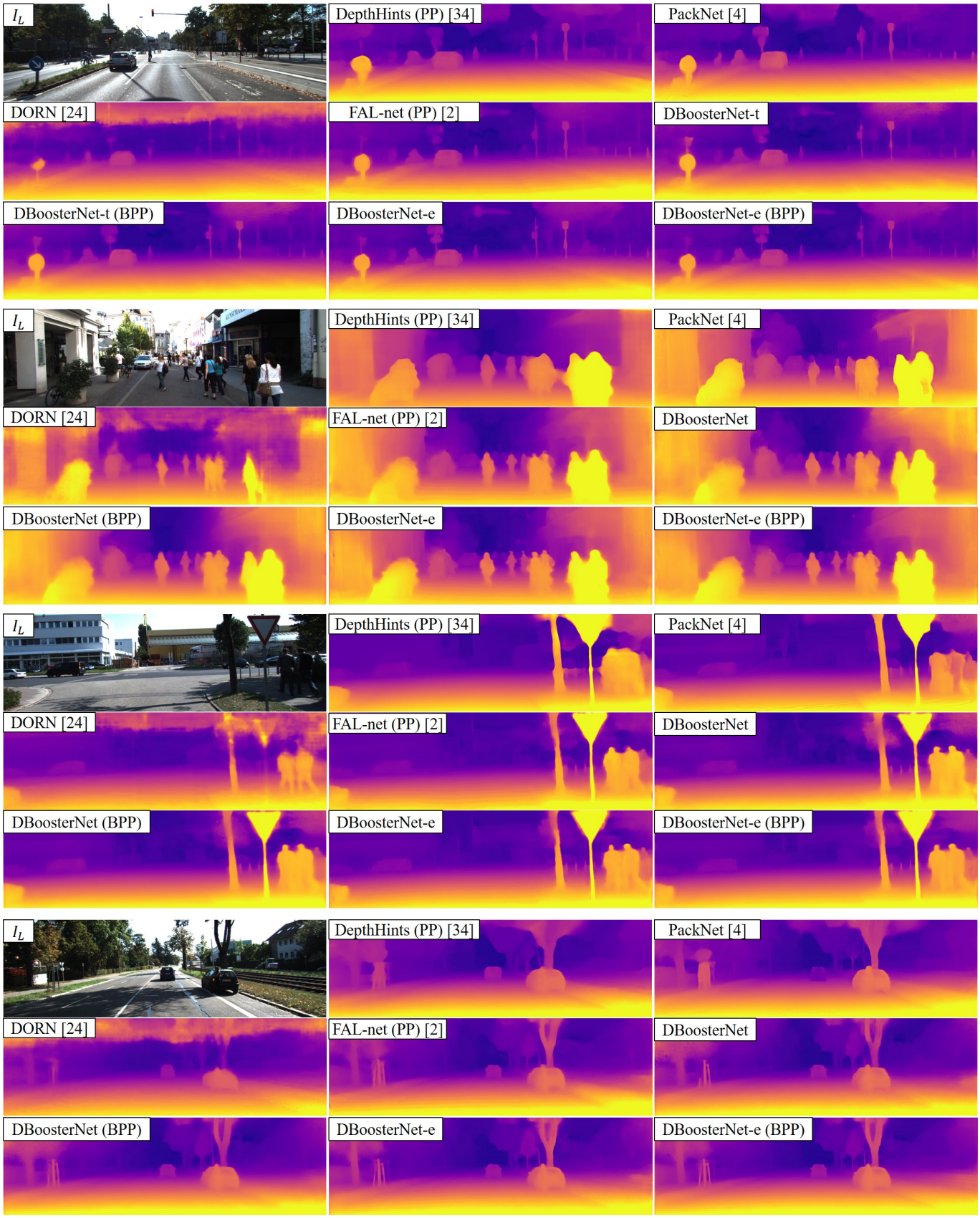

Fig. 7. Qualitative results on the KITTI Eigen test split [58] and comparison between the supervised SOTA DORN [24], the self-supervised SOTAs DepthHints [34] and PackNet [4] and our self-supervised DBoosterNet. Our DBoosterNet even without boosting post-processing (BPP) generates very detailed depths. Our method generates superior depth for very close objects, such as the pedestrians in the second and first samples, and for the distant objects, such as the traffic signs and trees in the first and last samples.

TABLE 6
Effects of aggressive data augmentations on different depth ranges

DBoosterNet	Data	rel	sq rel	rmse	$\log_{rmse}$	a^1	a^2	a^3
First training stage (Depths capped from 1 to 80m)								
-e	K+CS	0.070	0.271	2.792	0.105	0.944	0.992	0.998
-e (ext)	K+CS	**0.068**	**0.265**	**2.755**	**0.103**	**0.948**	**0.992**	0.998
First training stage (Depths capped from 1 to 20m)								
-e	K+CS	0.059	0.114	1.083	0.085	0.965	0.995	**0.999**
-e (ext)	K+CS	**0.057**	**0.107**	1.045	**0.082**	**0.968**	**0.996**	0.999
First training stage (Depths capped from 1 to 40m)								
-e	K+CS	0.066	0.190	**1.837**	0.096	0.954	0.993	0.998
-e (ext)	K+CS	**0.064**	**0.188**	1.853	**0.095**	**0.957**	**0.994**	0.998
First training stage (Depths capped from 40 to 80m)								
-e	K+CS	0.145	2.024	10.267	0.198	0.767	0.960	**0.991**
-e (ext)	K+CS	**0.141**	**1.974**	**10.042**	**0.193**	**0.782**	0.960	0.990

Evaluations performed on the KITTI Eigen test split with improved GT [3]. Metrics are the lower the better and the higher the better

The effects of additional training data (denoted as K+CS) and aggressive data augmentations (denoted by (ext)) were also ablated. When training with KITTI [56] and CityScapes [57] with aggressive data augmentations, we obtain substantial performance improvements of **1.4%** and **1.3%** in terms of the a^1 accuracy for our DBoosterNet-t and -e respectively, versus training with KITTI only and plain augmentations.

The benefits of aggressive data augmentations might not be clear in Table 5, as depth estimates are averaged over the standard measuring range (1-80m). To clarify this, we show the quantitative performance of our best network, the DBoosterNet-e, with and without aggressive random resizing over different depth ranges in Table 6. As can be noted, using aggressive data augmentations generates more consistent depth estimates for the closer and further-away objects and mostly yields the best metrics. The quantitative differences between the model trained with aggressive resizing and the model without it become larger when measured over more constrained depth ranges (1-20m, 1-40m, and 40-80m) in Table 6. The vanilla DBoosterNet-e yields marginally (0.038m) less RMSE performance in the closest range than the network trained with aggressive resizing and tends to yield more depth outliers as indicated by the lower a^1 and a^2 metrics. On the other hand, the DBoosterNet-e (ext) exhibits **0.225** less RMSE in the 40-80m range and fewer outliers as indicated by the higher a^1 metric.

The effects of not incorporating a camera grid and min-max disparity values as network inputs is also ablated in Table 5 denoted by (no-xy) and (no-mm), respectively. As observed in Table 5, gradually adding the min-max disparities and camera grid progressively improves the DE performance of the network.

Additionally, we observed that small quantitative improvements in the already well-performing DBoosterNet-e make large differences in the predicted depth maps' perceived visual quality. We show this in Fig. 8 for the DBoosterNet-e in its first training stage without and with aggressive data augmentations (denoted by DBoosterNet-e and DBoosterNet-e (ext), respectively) and also for the fine-tuned DBoosterNet-e with our ambiguity boosting training strategy (denoted by DBoosterNet-e (ext-boosting)). By examining the zoomed regions (in red and green boxes), it is evident that our network (ext-boosting) with resizing augmentations and ambiguity boosting fine-tuning can generate much sharper and consistent depth estimates, even when its RMSE is only 0.078m and 0.041m lower than those of the DBoosterNet-e and DBoosterNet-e (ext), respectively. The green boxes in Fig. 8 show the depth estimates of a relatively close-by object (car) where the DBoosterNet-e (ext) generates more consistent and detailed predictions than the model without aggressive resizing. The DBoosterNet-e (ext-boosting) generates even more consistent and sharper depth boundaries. The red boxes depict objects that are far away from the camera. DBoosterNet-e generates less detailed and noisier depths, while DBoosterNet-e (ext) generates sharper depths to the point that we can start identifying the depths of the distant car and trees. Finally, the DBoosterNet-e (ext-boosting) generates very clean depths where the distant car and other thin structures are clearly distinguishable.

Adding a post-processing (PP) step as in [28] (simple element-wise averaging between original and flipped estimates) marginally improves DBoosterNets performances in Table 5. On the other hand, our boosting post-processing, denoted as "BPP" in Table 5, remarkably improves the performance of our models. Our DBoosterNet-t benefits more from the proposed boosting post-processing step, with an additional performance gain of **0.5%** in a^1 for DBoosterNet-t (ext, BPP) in Table 5. For our DBoosterNet-e, a substantial performance gain in terms of the RMSE, from 2.714 to **2.617** meters, is observed when applying BPP. However, using BPP considerably increases the inference time for a full-resolution KITTI image of 384×1280 from 60ms to 320ms.

4.4 Results on KITTI

We quantitatively compare our method versus existing works on the KITTI Eigen test split in Table 7 and provide qualitative comparisons in Fig. 7. It is shown that DBoosterNets deliver the best performance, as measured with the KITTI metrics [58], among all of the self-supervised methods compared here. Our method even outperforms the SOTA supervised single-image DE methods [18], [20], [24], [45], [61] in terms of nearly all metrics in Table 7. Qualitatively, our DBoosterNets show sharper and more detailed depth estimates than those in previous works [2], [4], [24], [34] as shown in Fig. 7.

The top section of Table 7 shows that our DBoosterNets outperform the previous methods on the original KITTI Eigen split [58] in most metrics, achieving even lower RMSE values than the semi- and fully-supervised methods [18], [20], [61]. In particular, our DBoosterNet-e, even without a post-processing step, already outperforms all competing methods in terms of the RMSE. Adding boosting post-processing further improves our results.

The bottom section of Table 7 provides a quantitative comparison on the improved KITTI Eigen Test Split [3], which contains denser ground-truths from multiple LiDAR scans. Again, our DBoosterNets with boosting post-processing (BPP) deliver the best performance in all metrics, and even without BPP, our method achieves the lowest RMSE among the self-supervised methods under comparison here. The superior quality of our predicted depths is depicted in Fig. 7, where it is shown that DBoosterNets

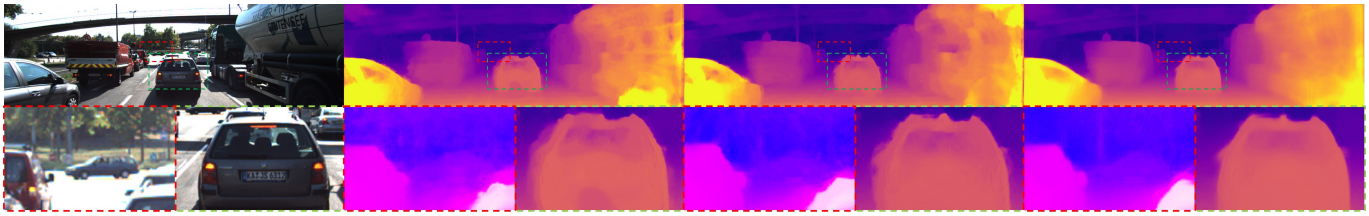

Fig. 8. Qualitative comparison between our DBoosterNet-e trained without and with aggressive data augmentations and with our ambiguity boosting training strategy. Note that the depth images in the red boxes are adjusted in brightness for better visualization. Results are without any post-processing.

TABLE 7
Comparison to Existing Self-, Semi- and Fully-Supervised Monocular DE Methods on the KITTI Eigen Split [58]

Ref	Methods	PP	Sup	Data	#Par	abs rel	sq rel	rmse	$\log_{rmse}$	a^1	a^2	a^3
Original Eigen Test Split [56]												
[61]	Gur *et al.*		DoF	K	-	0.110	0.666	4.186	**0.168**	0.880	0.966	**0.988**
[18]	Guo *et al.*		Syn	K+CS+SF	-	0.095	0.703	4.316	0.177	0.892	0.966	0.984
[20]	Luo *et al.*		D+S	K	-	0.094	0.626	4.252	0.177	0.891	0.965	0.984
[39]	Gordon *et al.*		V	K	-	0.128	0.959	5.230	0.212	0.845	0.947	0.976
[38]	Zhou *et al.*		V	K	34	0.121	0.837	4.945	0.197	0.853	0.955	0.982
[37]	Monodepth2		V	K	14	0.115	0.882	4.701	0.190	0.879	0.961	0.982
[4]	PackNet		V	K	120	0.107	0.802	4.538	0.186	0.889	0.962	0.981
[39]	Gordon *et al.*		V	K+CS	-	0.124	0.930	5.120	0.206	0.851	0.950	0.978
[4]	PackNet		V	CS→K	120	0.104	0.758	4.386	0.182	0.895	0.964	0.982
[4]	PackNet		V+v	CS→K	120	0.103	0.796	4.404	0.189	0.881	0.959	0.980
[47]	Guizilini *et al.*		V+Se	CS→K	140	0.100	0.761	4.270	0.175	0.902	0.965	0.982
[30]	3Net	✔	S	K	48	0.126	0.961	5.205	0.220	0.835	0.941	0.974
[46]	SuperDepth		S	K	-	0.112	0.875	4.958	0.207	0.852	0.947	0.977
[33]	Tosi *et al.*	✔	S_{SGM}	K	42	0.111	0.867	4.714	0.199	0.864	0.954	0.979
[32]	Refine&Distill		S	K	-	0.098	0.831	4.656	0.202	0.882	0.948	0.973
[34]	DepthHints	✔	S_{SGM}	K	35	0.096	0.710	4.393	0.185	0.890	0.962	0.981
[28]	Monodepth	✔	S	CS→K	32	0.114	0.898	4.935	0.206	0.861	0.949	0.976
[30]	3Net	✔	S	CS→K	48	0.111	0.849	4.822	0.202	0.865	0.952	0.978
[33]	Tosi *et al.*	✔	S_{SGM}	CS→K	42	0.096	0.673	4.351	0.184	0.890	0.961	0.981
[2]	FAL-net		S	K+CS	17	0.091	0.562	4.016	0.178	0.894	0.964	0.983
our	DBoosterNet-t		S	K	30	0.097	0.634	4.144	0.181	0.886	0.963	0.983
our	DBoosterNet-t		S	K+CS	30	0.094	0.599	4.074	0.177	0.893	0.965	0.984
our	DBoosterNet-t	BPP	S	K+CS	30	0.091	0.565	3.976	0.174	0.895	0.966	0.985
our	DBoosterNet-e		S	K	14	0.095	0.636	4.105	0.178	0.890	0.963	0.984
our	DBoosterNet-e		S	K+CS	14	0.087	0.553	3.891	0.169	**0.905**	0.968	0.985
our	DBoosterNet-e	BPP	S	K+CS	14	**0.086**	**0.538**	**3.852**	**0.168**	**0.905**	**0.969**	0.985
Improved Eigen Test Split [3]												
[24]	DORN		D	K	51	0.072	0.307	2.727	0.120	0.932	0.984	0.995
[45]	Yin *et al.*		D	K	113	0.072	-	3.258	0.117	0.938	0.990	0.998
[25]	BTS		D	K	113	0.059	0.245	2.756	0.096	0.956	0.993	0.998
[26]	SAN		D	K	120	**0.057**	**0.229**	**2.704**	**0.092**	**0.961**	**0.994**	**0.999**
[37]	Monodepth2		V	K	14	0.092	0.536	3.749	0.135	0.916	0.984	0.995
[4]	PackNet		V	CS→K	120	0.071	0.359	3.153	0.109	0.944	0.990	0.997
[4]	PackNet		V+v	CS→K	120	0.075	0.384	3.293	0.114	0.938	0.984	0.995
[37]	Monodepth2		V+S	K	14	0.087	0.479	3.595	0.131	0.916	0.984	0.996
[34]	DepthHints	✔	S_{SGM}	K	35	0.074	0.364	3.202	0.114	0.936	0.989	0.997
[2]	FAL-net	✔	S	K	17	0.071	0.281	2.912	0.108	0.943	0.991	**0.998**
[2]	FAL-net		S	K+CS	17	0.071	0.287	2.905	0.109	0.941	0.990	**0.998**
[2]	FAL-net	✔	S	K+CS	17	0.068	0.276	2.906	0.106	0.944	0.991	**0.998**
our	DBoosterNet-t		S	K	30	0.074	0.334	3.044	0.114	0.936	0.989	0.998
our	DBoosterNet-t		S	K+CS	30	0.072	0.320	3.010	0.109	0.942	0.990	**0.998**
our	DBoosterNet-t	BPP	S	K+CS	30	0.068	0.282	2.824	0.103	0.947	0.992	**0.998**
our	DBoosterNet-e		S	K	14	0.070	0.298	2.916	0.109	0.940	0.991	**0.998**
our	DBoosterNet-e		S	K+CS	14	0.064	0.258	2.714	0.099	0.952	0.993	**0.998**
our	DBoosterNet-e	BPP	S	K+CS	14	**0.062**	**0.242**	**2.617**	**0.096**	**0.955**	**0.994**	**0.998**

DoF, Syn, and D: fully supervised with depth of field images, synthetic datasets, or depth GT, respectively. V, V+v, V+se: Self-supervised from videos, videos + velocity supervision, and from videos + pre-trained semantics, respectively. S and S_{SGM}: Self-supervised from stereo and from stereo + semi-global matching, respectively. Models are trained with the Eigen [58] train-split (K), CityScapes [57] (CS), or the Sceneflow [17] (SF) datasets. PP: post-processing following [28], otherwise specified. BPP: our boosting post-processing. #Par: numbers of parameters in millions. ***Best*** *and second best metrics. Results capped at 80m.*

with and without BPP generate more consistent and detailed depth estimates than DORN [24], PackNet [4] and DepthHints [34]. In particular, the newly proposed DBoosterNets yield consistent depth estimates without the unpleasant depth discretization artifacts clearly visible in our previous work of FAL-net [2], as shown in Fig. 7. The predictions from DORN [24], PackNet [4] and DepthHints [34] were obtained from their respective open source repositories.

It is worth noting that the prior works [33], [34] use a classical stereo matcher to generate supervision signals for monocular depth estimation. Also, the work [18] utilized a semi-supervised deep learning-based stereo matching model to get supervisory signals for training. Our proposed DBoosterNets outperform the previous methods [18], [33], [34] with significant margins even without any post-processing, and instead relies on ambiguity boosting only.

4.5 Ambiguity Boosting Versus Infusing Deep-Stereo Information

It is reasonable to raise the question: "If we have stereo images during training, why not learning stereo disparity and use it for guiding the monocular network?". To thoroughly compare our ambiguity boosting training strategy, we have intensively studied the effects of infusing deep-stereo disparity information into single-view DE networks. Firstly, we naïvely extend our DBoosterNet-e for stereo inputs (referred to as DBoosterNet-es) by independently processing the left and right views via the same encoder and then fusing the extracted features at 1/32 resolution. The DBoosterNet-es (trained with L1, perceptual, and disparity smoothness losses) unsurprisingly outperforms our monocular variant in all metrics as shown in Table 8. We employ the disparity estimates from DBoosterNet-es to train our single view DE network in many ways:

1. Naïve stereo disparity supervision (denoted as naïve);
2. Stereo disparity supervision accounting for occluded regions. Occlusion masks are extracted from the stereo disparity probability volume as in [2] (denoted as occ);
3. Stereo disparity supervision with photometric distillation (similar to how DepthHints [34] treated the SGM disparities) and occlusion treatment (denoted as dstl-occ);
4. Stereo disparity supervision with photometric distillation and occlusion treatment in cooperation with a mirror loss (which uses a flipped or "mirrored" augmentation) as in [2] (denoted as dstl-occ-m);
5. Instead of distilling the predicted depth as in DepthHints [34], we just incorporate the stereo disparity as if it was another forward pass of the monocular model into our ambiguity boosting process (denoted as ours).

As can be observed in Table 8, training our DBoosterNet-e with a supportive stereo matcher does not produce as good results as our ambiguity boosting strategy. Indeed, naïvely attempting to supervise the monocular disparity with the stereo disparity results in no performance improvements. Other more elaborated ways of infusing the deep stereo disparity knowledge ('occ', 'dstl-occ', 'dstl-occ-m') yielded some performance improvements but are still inferior to our ambiguity boosting method which at this point could be considered more straightforward. In general, the stereo matchers tend to overfit the photometric losses, thus producing depth maps that are not very well correlated to the monocular input and that are hence difficult for the monocular network to learn. Stereo matchers can get easily confused on reflective, highly homogeneous, and occluded regions as seen in the DBoosterNet-es disparity estimate in the second column of Fig. 9.

TABLE 8
Effects of Infusing Deep Stereo Disparity Into Single View DE

DBoosterNet	Data	rel	sq rel	rmse	$\log_{rmse}$	a^1	a^2	a^3
First training stage								
-e	K	0.073	0.307	2.977	0.112	0.937	0.990	0.998
-es	K	0.042	0.175	2.501	0.076	0.977	0.996	0.999
Second training stage								
-e	K	0.070	0.298	2.916	0.109	0.940	0.991	0.998
Second training stage with deep stereo supervision								
-e (naïve)	K	0.073	0.302	3.012	0.112	0.936	0.991	0.998
-e (occ)	K	0.071	0.295	2.999	0.110	0.938	0.991	0.998
-e (dstl-occ)	K	0.071	0.294	2.946	0.110	0.939	0.991	0.998
-e (dstl-occ-m)	K	0.072	0.299	2.952	0.112	0.936	0.990	0.998
-e (ours)	K	0.069	0.284	2.851	0.106	0.943	0.992	0.998

Evaluations performed on the KITTI Eigen test split with improved GT [3].

The results in Table 8 might seem counter-intuitive at first glance. Still, it should be noted that they actually make sense if we consider that the stereo disparities from DBoosterNet-es provide the same kind of supervisory signals as the photometric reconstruction losses. On the other hand, our ambiguity boosting loss provides novel supervisory information by analyzing the depths of the input image under several input transformations (such as re-scaling and horizontal flips). Additionally, in Table 8, we can see that our ambiguity boosting process can effectively incorporate the stereo disparity information. The DBoosterNet-e (ours) in Table 8 achieves the highest performance levels at the cost of pre-training an additional stereo network.

4.6 Generalization of Ambiguity Boosting in Direct DE Networks

To demonstrate the effectiveness of our method, we have performed additional extensive experiments with a direct DE method that is trained following Monodepth [28] in its first training stage, and with our ambiguity boosting in the second stage. This network, referred to as DBoosterNet-m, incorporates ambiguity learning in its output layers and predicts left and right disparities from the single input view as in [28]. For a fair comparison, the DBoosterNet-m adopts the same auto-encoder backbone in Table 1).

The experiments with the DBoosterNet-m were carried out on the KITTI Eigen Split [58] and the results are shown in Table 9. Interestingly, the improvements in our DBoosterNet-m are more significant than those of our DBoosterNet-t and DBoosterNet-e, which is reasonable because the DBoosterNet-m yields lower performance in the first training stage. With our ambiguity boosting-based fine-tuning, the DBoosterNet-m achieves performance levels even superior to those of DepthHints [34] which exploits stereo disparity (via SGM) supervision and is a relatively much more recent work than Monodepth [28]. Qualitatively, there is also a considerable difference between the plain

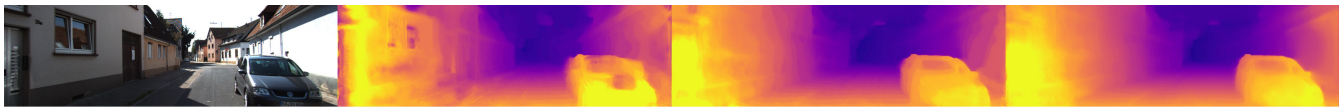

Fig. 9. Effects of deep stereo supervision. DBoosterNet-es and DBoosterNet-e are trained on a single stage. DBoosterNet-e (ours) is trained with our ambiguity boosting technique and deep stereo information.

TABLE 9
Performance improvements of DBoosterNet-m

DBoosterNet	Data	rel	sq rel	rmse	$\log_{rmse}$	a^1	a^2	a^3
First training stage								
-m	K	0.078	0.426	3.329	0.121	0.930	0.985	0.996
Second training stage								
-m	K	0.072	0.341	3.033	0.112	0.938	0.989	0.997
-m (PP)	K	0.071	0.327	2.973	0.112	0.937	0.989	0.997
-m (BPP)	K	**0.069**	**0.311**	**2.887**	**0.109**	**0.940**	**0.990**	**0.998**

Evaluations performed on the KITTI Eigen test split with improved GT [3].

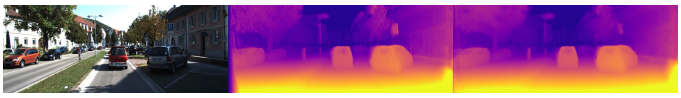

Fig. 10. Effects of our proposed ambiguity boosting on DBoosterNet-m. From left to right: input image, disparity estimate by training following [28], and disparity estimate by fine tuning with our ambiguity boosting.

DBoosterNet-m and its ambiguity-boosted version. As can be observed in Fig. 10, the depth estimates from the network fine-tuned with ambiguity boosting are more consistent and contain much fewer occlusion artifacts. The DBoosterNet-m's significant performance gains when trained with our ambiguity boosting comes from the fact that our method can also boost up the learning process for direct DE methods.

4.7 Results on Make3D

We tested our method on the Make3D [59] dataset following an earlier protocol from [28], [37], that is, evaluating on a center crop of 852×1704 only where the GT depth is less than 70 meters [62]. As shown in Table 10, our DBoosterNets outperform all previous *self-supervised* methods (bottom section of the table) and achieves results very close to those by the *fully-supervised* methods trained with the depth GT from the Make3D training dataset (top section). Our method was not fine-tuned nor trained on the Make3D data. Therefore, its superior performance indicates that the proposed method generalizes well. The depths predicted by our DBoosterNets are depicted in Fig. 11. Our method generates more precise depths that are closer to the ground-truth in both cases with and without the BPP.

5 Conclusion

We presented a new learning strategy to boost up the performance of DCNNs in an unsupervised fashion for the depth estimation regression task. Our key contribution is our novel two-stage training strategy with ambiguity boosting, which here boosted the performance of simple auto-encoder networks beyond those of the SOTA methods. We showed that the proposed training strategy is effective and can be applied to any direct (as our DBoosterNet-m) or indirect (as in our DBoosterNet-t and -e) DE network that incorporates an ambiguity (or confidence) mask. We also showed that networks that incorporate single (as in our DBoosterNet-e) or multiple sub-nets (as in our DBoosterNet-t) could benefit from the proposed ambiguity boosting training strategy. Our DBoosterNets have low complexity, with far fewer filter parameters than recent fully-supervised SOTA methods for DE, which supports the effectiveness of our two-stage training strategy with ambiguity boosting. Our work improves the SOTA self-supervised methods that learn from stereo images, even outperforming recent fully supervised SOTA methods in terms of RMSE for the single-image depth estimation task. Moreover, we believe that our boosting approach should benefit DE methods that learn from sparse GT. In this case, our technique should help improve the supervisory signals in the regions without depth labels. However, it might be hard to precisely measure the quantitative improvements for this case as most of the improvements will be potentially reflected in regions with less GT available. For these reasons, we believe that learning supervised DE with ambiguity boosting and a study of fair metrics are worthwhile exploring in future works. Following our "self-boosting" philosophy, presuming proper capture conditions, will allow us to eliminate the need for depth sensors in the future.

TABLE 10
C1 Metrics [62] for Methods Evaluated on Make3D [59] Test134

Method	abs rel	sq rel	rmse	Avg log_{10}
*Liu *et al.* [62]	0.475	6.562	10.05	0.165
*Laina *et al.* [63]	**0.204**	**1.840**	**5.683**	**0.084**
Godard *et al.* [28]	0.544	10.94	11.76	0.193
Zhou *et al.* [35]	0.383	5.321	10.47	0.478
Wang *et al.* [64]	0.387	4.720	8.09	0.204
Gonzalez and Kim [29]	0.323	4.021	7.507	0.121
Godard *et al.* v2 [37]	0.322	3.589	7.417	0.163
Zhou *et al.* [38]	0.318	2.288	6.669	-
FAL-net [2]	0.256	2.179	6.201	0.106
FAL-net (PP) [2]	**0.254**	**2.132**	6.139	0.105
DBoosterNet-t	0.294	2.593	6.442	0.122
DBoosterNet-t (BPP)	0.289	2.487	6.334	0.119
DBoosterNet-e	0.269	2.369	6.165	0.106
DBoosterNet-e (PP)	0.263	2.289	6.069	**0.104**
DBoosterNet-e (BPP)	0.261	2.244	**6.021**	**0.104**

*The first two methods are *supervised with depth GT from the Make3D training dataset. **Best** metrics in bold font. Our method achieves the best result among the unsupervised (last 7) methods.*

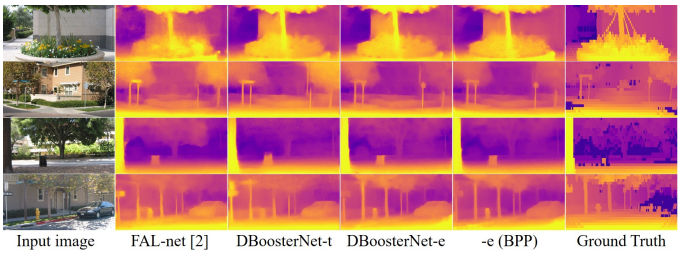

Fig. 11. Qualitative results on the Make3D [59] dataset.

ACKNOWLEDGMENTS

This work was supported by a grant from the Institute for Information & Communications Technology Promotion (IITP) funded by the Korean government (MSIT) (No. 2017-0-00419, Intelligent High Realistic Visual Processing for Smart Broadcasting Media).

REFERENCES

[1] Juan Luis GonzalezBello and Munchurl Kim, "Deep 3d pan via local adaptive "t-shaped" convolutions with global and local adaptive dilations," in *International Conference on Learning Representations*, 2020.

[2] Juan Luis GonzalezBello and Munchurl Kim, "Forget about the lidar: Self-supervised depth estimators with med probability volumes," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds. 2020, vol. 33, pp. 12626–12637, Curran Associates, Inc.

[3] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger, "Sparsity invariant cnns," in *2017 International Conference on 3D Vision (3DV)*. IEEE, 2017, pp. 11–20.

[4] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, and Adrien Gaidon, "Packnet-sfm: 3d packing for self-supervised monocular depth estimation," *CoRR*, vol. abs/1905.02693, 2019.

[5] Yasutaka Furukawa, Carlos Hernández, et al., "Multi-view stereo: A tutorial," *Foundations and Trends® in Computer Graphics and Vision*, vol. 9, no. 1-2, pp. 1–148, 2015.

[6] Daniel Scharstein and Richard Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1-3, pp. 7–42, 2002.

[7] Robert J. Woodham, "Photometric Method For Determining Surface Orientation From Multiple Images," *Optical Engineering*, vol. 19, no. 1, pp. 139 – 144, 1980.

[8] Gaurav Chaurasia, Sylvain Duchêne, Olga Sorkine-Hornung, and George Drettakis, "Depth synthesis and local warps for plausible image-based navigation," *ACM Transactions on Graphics*, vol. 32, 2013, to be presented at SIGGRAPH 2013.

[9] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala, "Content-preserving warps for 3d video stabilization," in *ACM SIGGRAPH 2009 Papers*, New York, NY, USA, 2009, SIGGRAPH '09, pp. 44:1–44:9, ACM.

[10] O. J. Woodford, I. D. Reid, P. H. S. Torr, and A. W. Fitzgibbon, "On new view synthesis using multiview stereo," in *In Proc. BMVC*, 2007, pp. 1120–1129.

[11] Rene Ranftl, Vibhav Vineet, Qifeng Chen, and Vladlen Koltun, "Dense monocular depth estimation in complex dynamic scenes," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 4058–4066.

[12] Derek Hoiem, Alexei A. Efros, and Martial Hebert, "Automatic photo pop-up," in *ACM SIGGRAPH 2005 Papers*, New York, NY, USA, 2005, SIGGRAPH '05, pp. 577–584, ACM.

[13] Austin Abrams, Christopher Hawley, and Robert Pless, "Heliometric stereo: Shape from sun position," in *Computer Vision – ECCV 2012*, Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid, Eds., Berlin, Heidelberg, 2012, pp. 357–370, Springer Berlin Heidelberg.

[14] Aamir Saeed Malik and Tae-Sun Choi, "A novel algorithm for estimation of depth map using image focus for 3d shape recovery in the presence of noise," *Pattern Recognition*, vol. 41, pp. 2200–2225, 2008.

[15] Ashutosh Saxena, Sung H Chung, and Andrew Y Ng, "Learning depth from single monocular images," in *Advances in Neural Information Processing Systems*, 2006, pp. 1161–1168.

[16] Youichi Horry, Ken Anjyo, and Kiyoshi Arai, "Tour into the picture: Using spidery mesh interface to make animation from a single image"," 01 1997, pp. 225–232.

[17] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040–4048.

[18] Xiaoyang Guo, Hongsheng Li, Shuai Yi, Jimmy Ren, and Xiaogang Wang, "Learning monocular depth by distilling cross-domain stereo networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 484–500.

[19] Rui Wang, Stephen M Pizer, and Jan-Michael Frahm, "Recurrent neural network for (un-) supervised learning of monocular video visual odometry and depth," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5555–5564.

[20] Yue Luo, Jimmy Ren, Mude Lin, Jiahao Pang, Wenxiu Sun, Hongsheng Li, and Liang Lin, "Single view stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 155–163.

[21] Jose M. Facil, Benjamin Ummenhofer, Huizhong Zhou, Luis Montesano, Thomas Brox, and Javier Civera, "Cam-convs: Camera-aware multi-scale convolutions for single-view depth," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[22] Rahul Garg, Neal Wadhwa, Sameer Ansari, and Jonathan T. Barron, "Learning single camera depth estimation using dual-pixels," in *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[23] Sunghoon Im, Hae-Gon Jeon, Stephen Lin, and In So Kweon, "DPSNet: End-to-end deep plane sweep stereo," in *International Conference on Learning Representations*, 2019.

[24] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao, "Deep ordinal regression network for monocular depth estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2002–2011.

[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," *CoRR*, vol. abs/1907.10326, 2019.

[26] Vitor Guizilini, Rares Ambrus, Wolfram Burgard, and Adrien Gaidon, "Sparse auxiliary networks for unified monocular depth prediction and completion," *CoRR*, vol. abs/2103.16690, 2021.

[27] 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*. Springer, 2016, pp. 740–756.

[28] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 270–279.

[29] Juan Luis GonzalezBello and Munchurl Kim, "A novel monocular disparity estimation network with domain transformation and ambiguity learning," in *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, pp. 474–478.

[30] Matteo Poggi, Fabio Tosi, and Stefano Mattoccia, "Learning monocular depth estimation with unsupervised trinocular assumptions," in *2018 International Conference on 3D Vision (3DV)*. IEEE, 2018, pp. 324–333.

[31] Alex Wong and Stefano Soatto, "Bilateral cyclic constraint and adaptive regularization for unsupervised monocular depth prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5644–5653.

[32] Andrea Pilzer, Stephane Lathuiliere, Nicu Sebe, and Elisa Ricci, "Refine and distill: Exploiting cycle-inconsistency and knowledge distillation for unsupervised monocular depth estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9768–9777.

[33] Fabio Tosi, Filippo Aleotti, Matteo Poggi, and Stefano Mattoccia, "Learning monocular depth estimation infusing traditional stereo knowledge," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9799–9809.

[34] Jamie Watson, Michael Firman, Gabriel J Brostow, and Daniyar Turmukhambetov, "Self-supervised monocular depth hints," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 2162–2171.

[35] 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*, 2017, pp. 1851–1858.

[36] Anurag Ranjan, Varun Jampani, Lukas Balles, Kihwan Kim, Deqing Sun, Jonas Wulff, and Michael J. Black, "Competitive collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[37] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow, "Digging into self-supervised monocular depth estimation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3828–3838.

[38] Junsheng Zhou, Yuwang Wang, Kaihuai Qin, and Wenjun Zeng, "Unsupervised high-resolution depth learning from videos with dual networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6872–6881.

[39] 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 International Conference on Computer Vision*, 2019, pp. 8977–8986.

[40] Hsueh-Ying Lai, Yi-Hsuan Tsai, and Wei-Chen Chiu, "Bridging stereo matching and optical flow via spatiotemporal correspondence," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1890–1899.

[41] Yang Wang, Peng Wang, Zhenheng Yang, Chenxu Luo, Yi Yang, and Wei Xu, "Unos: Unified unsupervised optical-flow and stereo-depth estimation by watching videos," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[42] John Flynn, Ivan Neulander, James Philbin, and Noah Snavely, "Deepstereo: Learning to predict new views from the world's imagery," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[43] Junyuan Xie, Ross Girshick, and Ali Farhadi, "Deep3d: Fully automatic 2d-to-3d video conversion with deep convolutional neural networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 842–857.

[44] Simon Niklaus, Long Mai, and Feng Liu, "Video frame interpolation via adaptive separable convolution," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 261–270.

[45] 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.

[46] Sudeep Pillai, Rareş Ambruş, and Adrien Gaidon, "Superdepth: Self-supervised, super-resolved monocular depth estimation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9250–9256.

[47] Vitor Guizilini, Rui Hou, Jie Li, Rares Ambrus, and Adrien Gaidon, "Semantically-guided representation learning for self-supervised monocular depth," in *International Conference on Learning Representations*, 2020.

[48] Yu Zhang, Dongqing Zou, Jimmy S. Ren, Zhe Jiang, and Xiaohao Chen, "Structure-preserving stereoscopic view synthesis with multi-scale adversarial correlation matching," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[49] Richard Tucker and Noah Snavely, "Single-view view synthesis with multiplane images," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[50] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 1874–1883.

[51] Moritz Menze and Andreas Geiger, "Object scene flow for autonomous vehicles," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[52] Justin Johnson, Alexandre Alahi, and Li Fei-Fei, "Perceptual losses for real-time style transfer and super-resolution," in *European conference on computer vision*. Springer, 2016, pp. 694–711.

[53] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[54] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann, "Shortcut learning in deep neural networks," *arXiv preprint arXiv:2004.07780*, 2020.

[55] Tom van Dijk and Guido de Croon, "How do neural networks see depth in single images?," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[56] 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*. IEEE, 2012, pp. 3354–3361.

[57] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016, pp. 3213–3223.

[58] David Eigen, Christian Puhrsch, and Rob Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in Neural Information Processing Systems*, 2014, pp. 2366–2374.

[59] Ashutosh Saxena, Min Sun, and Andrew Y Ng, "Make3d: Learning 3d scene structure from a single still image," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 824–840, 2008.

[60] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[61] Shir Gur and Lior Wolf, "Single image depth estimation trained via depth from defocus cues," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7683–7692.

[62] Miaomiao Liu, Mathieu Salzmann, and Xuming He, "Discrete-continuous depth estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 716–723.

[63] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab, "Deeper depth prediction with fully convolutional residual networks," in *2016 Fourth international conference on 3D vision (3DV)*. IEEE, 2016, pp. 239–248.

[64] Chaoyang Wang, José Miguel Buenaposada, Rui Zhu, and Simon Lucey, "Learning depth from monocular videos using direct methods," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2022–2030.

Juan Luis Gonzalez Bello Juan Luis Gonzalez Bello received a B.S degree in electro-mechanical engineering, specialized in digital systems, from the Mexican Autonomous National University (UNAM) in 2013. He worked in Procter and Gamble for three years in the engineering and regional engineering departments, managing projects and assisting in designing new advanced manufacturing machinery. Since 2017 he has joined the school of electrical engineering at Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, where he is currently pursuing a Ph.D. degree. His research interests include novel-view synthesis, monocular/stereo depth estimation, and deep learning, with related papers published in top conference venues such as CVPR2021, NeurIPS2020, ICLR2020, and ICIP2019-2020 as the first author.

Munchurl Kim (M'99–SM'13) received the B.E. degree in electronics from Kyungpook National University, Daegu, South Korea, in 1989, and the M.E. and Ph.D. degrees in electrical and computer engineering from the University of Florida, Gainesville, in 1992 and 1996, respectively. He joined the Electronics and Telecommunications Research Institute, Daejeon, South Korea, as a Senior Research Staff Member, where he led the Realistic Broadcasting Media Research Team. In 2001, he joined the School of Engineering, Information and Communications University (ICU), Daejeon, as an Assistant Professor. Since 2009, he has been with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, where he is currently a Full Professor. He has published 180 (165) international (domestic) journal and conrerence papers. He holds several dozens of essential HEVE patents and about 200 registered domestic and internation patents in the areas of image restoration and video coding. He received a commendation from the Korea President in the 54th National Innovation Day 2019, and was awarded the Grand Prize for Research Excellence in Commemoration of the 50th Anniversary of Founding for KAIST. He had an invited keynote speech on evolution of conventional and deep video compression technologies in 2020 Multimedia Modeling conference. His team was awarded the runner-up in Challenge on the Video Temporal Super-Resolution track in AIM (Advances in Image Manipulation workshop and challenges on image and video manipulation) in ECCV 2020, and received the Winner award on "PIRM Challenge on Perceptual Image Enhancement – Track A: Image Super Resolution" in ECCV 2018. His research interest includes image restoration with deep learning, video coding and image understanding.