Zoom and Learn: Generalizing Deep Stereo Matching to Novel Domains

Jiahao Pang[1] Wenxiu Sun[1] Chengxi Yang[1] Jimmy Ren[1] Ruichao Xiao[1] Jin Zeng[1] Liang Lin[1,2]
[1]SenseTime Research [2]Sun Yat-sen University
{pangjiahao, sunwenxiu, yangchengxi, rensijie, xiaoruichao, zengjin, linliang}@sensetime.com

Abstract

Despite the recent success of stereo matching with convolutional neural networks (CNNs), it remains arduous to generalize a pre-trained deep stereo model to a novel domain. A major difficulty is to collect accurate ground-truth disparities for stereo pairs in the target domain. In this work, we propose a self-adaptation approach for CNN training, utilizing both synthetic training data (with ground-truth disparities) and stereo pairs in the new domain (without ground-truths). Our method is driven by two empirical observations. By feeding real stereo pairs of different domains to stereo models pre-trained with synthetic data, we see that: i) a pre-trained model does not generalize well to the new domain, producing artifacts at boundaries and ill-posed regions; however, ii) feeding an up-sampled stereo pair leads to a disparity map with extra details. To avoid i) while exploiting ii), we formulate an iterative optimization problem with graph Laplacian regularization. At each iteration, the CNN adapts itself better to the new domain: we let the CNN learn its own higher-resolution output; at the meanwhile, a graph Laplacian regularization is imposed to discriminatively keep the desired edges while smoothing out the artifacts. We demonstrate the effectiveness of our method in two domains: daily scenes collected by smartphone cameras, and street views captured in a driving car.

1. Introduction

Stereo matching is a classic yet important problem for many computer vision tasks (*e.g.,* 3D reconstruction [7] and autonomous vehicles [6]). Particularly, given a rectified image pair captured by stereo cameras, one aims at estimating the disparity of each pixel between the two images. Traditionally, a stereo matching pipeline starts from matching cost computation and cost aggregation. Further optimization and refinement lead to the output disparity [13]. Recent advances in deep learning has inspired a lot of end-to-end convolutional neural networks (CNNs) for stereo matching, *e.g.,* [18, 23]. Unlike the traditional wisdom, an end-to-end CNN integrates the stereo matching pipeline into a holistic deep architecture by learning from the training data. Under confined scenarios with proper training data (*e.g.,* the KITTI dataset [6]), the end-to-end deep stereo models achieve unprecedented state-of-the-art performance.

However, it remains difficult to generalize a pre-trained deep stereo model to a novel scenario. Firstly, the contents in the source domain may have very different characteristics from the target domain. Moreover, real stereo pairs collected with different stereo modules suffer from several degenerations—*e.g.*, noise corruption, photometric distortions, imperfections in rectification—to different extents. Directly feeding a stereo pair of the target domain to a CNN pre-trained from another domain deteriorates its performance significantly. Consequently, state-of-the-art approaches, *e.g.*, [18, 28], train their models with synthetic datasets [23], then perform finetuning on a fewer amount of domain-specific data with ground-truths. Unfortunately, besides a few public datasets for research purpose, *e.g.*, the KITTI dataset [6] and the Middlebury dataset [32], it is expensive and troublesome to collect real stereo pairs with accurate ground-truth disparities.

To resolve this dilemma, we propose a self-adaptation approach to generalize deep stereo matching methods to novel domains. We utilize synthetic training data and stereo pairs of the target domain, where only the synthetic data have known disparity maps. Our approach is compatible with end-to-end deep stereo methods, *e.g.,* [23, 28], guiding a pre-trained model to gradually adapt to the target scenario. We start our explorations by feeding real stereo pairs from different domains to models pre-trained with synthetic data, resulting in two empirical observations:

(i) *Generalization glitches*: a pre-trained model does not generalize well on the target domain—the produced disparity maps can be blurry at object edges and erroneous at ill-posed regions;

(ii) *Scale diversity*: feeding a properly up-sampled stereo pair (the same stereo pair at a finer scale) leads to another disparity map with more meaningful details, *e.g.*, sharper object boundaries, more high-frequency contents of the scene.

To avoid the issues of (i) while exploiting the benefits of (ii),

Figure 1. Feeding stereo pairs collected from smartphones to models pre-trained from synthetic data leads to blurry edges and artifacts. In contrast, our self-adaptation approach brings significant improvements to the disparity maps.

we propose an iterative regularization scheme for finetuning deep stereo matching models.

We formulate the CNN training as an iterative optimization problem with graph Laplacian regularization. On one hand, we let the CNN learn its own finer-grain output; on the other hand, a graph Laplacian regularization is imposed to discriminatively retain the useful edges while smoothing out the undesired artifacts. Our formulation, composing of a data term and a smoothness term, is solved iteratively, leading to a model well suited for the novel domain *e.g.*, Figure 1. The proposed self-adaptation approach is called *zoom and learn*, or ZOLE, for short. We demonstrate the effectiveness of our approach to two different domains: daily scenes collected by smartphone cameras, and street views captured from the perspective of a driving car.

This paper is structured as follows. Related works are reviewed in Section 2. We then illustrate our observations about deep stereo models in Section 3. The proposed self-adaptation approach is introduced in Section 4. Section 5 presents the experimental results and Section 6 concludes our work.

2. Related Works

We first review several stereo matching algorithms based on convolutional neural networks (CNNs). We then turn to related works on graph Laplacian regularization and iterative regularization/filtering.

Deep stereo algorithms: Recent breakthroughs in deep learning have reshaped the paradigms of many computer vision tasks, including stereo matching. Early works employing CNNs for stereo matching focuses on learning a robust similarity measure for matching cost computation *e.g.*, [11, 37]. To produce disparity maps, modules in the traditional stereo matching pipeline are indispensable. The remarkable work, *DispNet*, proposed by Mayer *et al* [23], is the first end-to-end CNN approach for stereo matching, where an encoder-decoder architecture is employed for supervised learning. Other recent works with leading performance include CRL [28], GC-NET [18], DRR [8], *etc*. These works explore different CNN architectures tailormade for stereo matching. They achieve superior results on the KITTI 2015 stereo benchmark [6], a benchmark containing driving scenes. Despite the success of these methodologies, to adopt them in a novel domain, it is necessary to fine-tune the models with new domain-specific data. Unfortunately, in practice, it is very difficult to collect accurate disparity maps for training [6, 32].

To mitigate this problem, some recent works proposed semi-/un-supervised approaches to train a CNN model for stereo matching (or its related problem, monocular depth estimation). This category of works is essentially based on left-right consistency/warping, *e.g.*, [10, 19, 38, 39]. For instance, one may synthesize the left (or right) view according to the estimated left (or right) disparity and the right (or left) view for computing a loss function. However, left-right consistency becomes vulnerable when the stereo pairs are imperfect, *e.g.*, when the two views have different photometric distortions. Another line of research by Tonioni *et al.* [34] propose to finetune a pre-trained model to achieve domain adaptation. Their method relies on the results of other stereo methods and confidence measures. Our work also performs finetuning with a pre-trained stereo model. In contrast, we do not rely on external models or setups: our self-supervised domain adaptation method lets the CNN discriminatively learn the useful details from its own finer-grain outputs.

Other related works: According to [21, 33], graph Laplacian regularization is particularly useful for the recovery of piecewise smooth signals, *e.g.*, disparity maps. By having an appropriate graph, edges can be preserved while undesired defects are suppressed [26, 27]. Hence, we propose to apply graph Laplacian regularization to selectively learn and preserve the meaningful details from the higher-resolution disparity outputs.

Iterative regularization/filtering is an important technique in classic image restoration [17, 24, 25]. To restore a corrupted image, it is regularized iteratively through a variational formulation, so that its quality improves at each iteration. To utilize scale diversity while avoiding generalization glitches (as mentioned in Section 1), we embed iterative regularization into the CNN training process, making the model parameters improve gradually. Different from iterative refinement via a stacked neural network architecture, *e.g.*, [15, 35], our iterative process occurs during training.

3. Observations

We first present two phenomena by feeding real-world stereo pairs in different domains to deep stereo models pre-trained with synthetic datasets (*e.g.*, FlyingThings3D [23], MPI Sintel [3], Virtual KITTI [5]). Underlying reasons for these phenomena will also be presented. We choose the off-the-shelf *DispNet* [23] architectures—both the one with explicit correlation (DispNetC) and the one based on convolution only (DispNetS)—for our discussions. Their encoder-decoder architectures are representative and also widely used in the deep learning literature, *e.g.*, [1, 22, 31].

3.1. Generalization Glitches

In general, a stereo model pre-trained with synthetic data does not perform well on real stereo data in a particular domain. Firstly, the contents of the synthetic data may differ from that of the target domain. Moreover, real stereo pairs inevitably suffer from defects arising from the imaging process. For instance, they are likely corrupted by noise. Besides, the two views may have different photometric distortions due to inconformity of the two cameras. In some cases, the stereo pair may not even be well rectified, *e.g.*, two corresponding pixels are not on the same scan-line. All the above factors deteriorate the performance of a model pre-trained with synthetic data.

For illustration, we use smartphones equipped with two rear-facing cameras to collect a few stereo pairs (of size 1024×1024), then perform the following tests. We first adopt the released DispNetC model pre-trained with the FlyingThings3D dataset [23]. Since stereo pairs of smartphones have small disparity values, we also fine-tune a model from the released model, where we remove those FlyingThings3D stereo pairs with maximum disparity larger than 80. Data augmentation is introduced for the two views individually during training, please refer to Section 5 for more details. The resulting model is called DispNetC-80. Both DispNetC and DispNetC-80 perform very well on the FlyingThings3D dataset, but are problematic when applied to real smartphone data. Figure 1 shows a few disparity estimates of DispNetC and DispNetC-80. As can be seen, the results are blurry at object edges. Moreover, at ill-posed regions, *i.e.*, object occlusions, repeated patterns, and textureless regions, the disparity maps are erroneous. In this work we call this *generalization glitches*, meaning the mistakes that a deep stereo model (pre-trained with synthetic data) make when it is applied to real stereo pairs of a certain domain.

3.2. Scale Diversity

In spite of the unpleasant generalization glitches, we find that deep stereo models have an encouraging property. Suppose we have a stereo pair $P = (L, R)$, where L and R are the left and the right views, respectively. We denote a deep stereo model parameterized by $\mathbf{\Theta}$ as $S(\cdot; \mathbf{\Theta})$. By applying it to the stereo pair P leads to a disparity map $D = S(P; \mathbf{\Theta})$. The operation of up-sampling by r times is denoted as $\uparrow_r(\cdot)$ while down-sampling by r times is $\downarrow_r(\cdot)$. By passing an up-sampled stereo pair to S then down-sampling the result, we obtain another disparity map, D', of the same size as D,

$$D' = \frac{1}{r} \cdot \downarrow_r \left(S \left(\uparrow_r (P); \mathbf{\Theta} \right) \right). \tag{1}$$

Note that after downsampling, the factor $1/r$ is necessary for making D' to have the correct scaling. Compared to D, D' usually contains more high-frequency details. To see this, we apply the released DispNetC model to a few stereo pairs captured by smartphones. We make the original size of the stereo pairs as 640×640. For each of them, we estimate three disparity maps based on (1) with $r \in \{1, 1.5, 2\}$. Visual results are shown in Figure 2. We see that as r grows, more fine details are produced on the disparity maps.

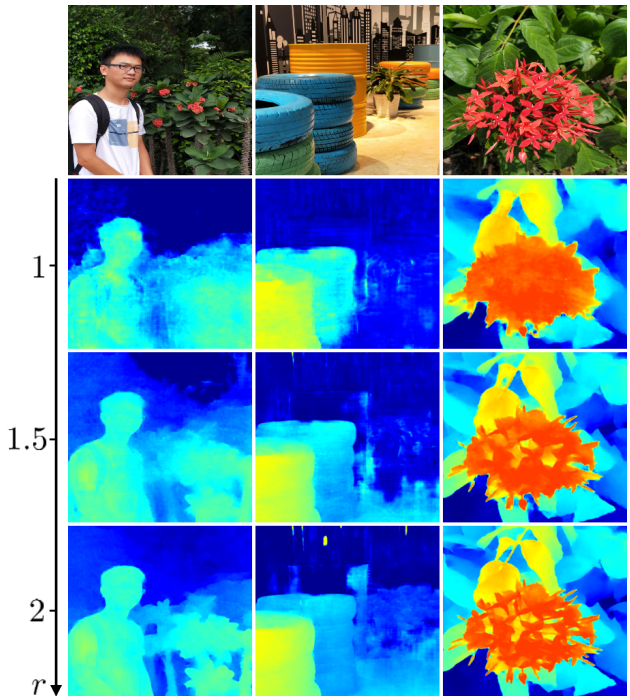


Figure 2. For the same stereo pair, feeding its zoomed-in version to a stereo matching CNN leads to a disparity map with extra details. The four rows are the left image, the disparity maps obtained by (1), with up-sampling ratio $r = 1, 1.5, 2$, respectively.

However, a bigger r does not necessarily mean better results. For further inspection, we adopt the released DispNetC and DispNetS models (trained with the FlyingThings3D dataset) and measure their performance on the training set of KITTI stereo 2015 [6] at different resolutions. The results, in terms of the percentage of pixels with an error greater than 3, or three-pixel error rate (3ER), are listed

Table 1. The average three-pixel error rates of the released DispNetC and DispNetS models on the training set of KITTI stereo 2015. A resolution of N means the stereo pairs are resized to $N \times N$ before passing to the CNNs.

Network	Resolution				
	896	1280	1664	2048	2432
DispNetC	14.26%	9.97%	**8.81%**	9.17%	10.53%
DispNetS	18.95%	11.61%	9.18%	**8.64%**	9.08%

in Table 1. We see that as the input resolution increases, the performance first improves then deteriorates. Because:

(i) Up-sampling the stereo pairs enables the model to perform stereo matching at a localized manner with sub-pixel accuracy. Hence, more details on the stereo pairs are taken into account for computation, leading to disparity estimates with extra high-frequency contents;

(ii) A finer-scale input translates to a smaller effective search range (or receptive field). As a CNN becomes too "short-sighted," it lacks non-local information to estimate a proper disparity map, and its performance start to decline.

This phenomenon—different results can be observed with different input scales—is called *scale diversity*, akin to the concept of transmit diversity in communication [30]. We find that scale diversity also exists in other problems, *e.g.*, optical flow estimation [15, 23] and image segmentation [22], please refer to the supplementary material for more details.

4. Zoom and Learn

To achieve effective self-adaptation, our approach—zoom and learn (ZOLE)—finetunes a model pre-trained with synthetic data. It iteratively suppresses generalization glitches while utilizing the benefits of scale diversity.

4.1. Graph Laplacian Regularization

Graph Laplacian regularization is employed in a wide range of image restoration literature, *e.g.*, [4, 9, 24]. It is also proven to be effective for the recovery of piecewise smooth signals [14, 26, 33]. We adopt graph Laplacian regularization (on a patch-by-patch basis) to guide the learning of CNNs. Graph Laplacian regularization assumes the ground-truth signal $\mathbf{s} \in \mathbb{R}^m$—in our case, a patch on the ground-truth disparity—is smooth with respect to a predefined graph $\mathcal{G}$ with m vertices. Specifically, it imposes that the value of $\mathbf{s}^{\mathrm{T}}\mathbf{L}\mathbf{s}$, *i.e.*, the graph Laplacian regularizer, should be small for the ground-truth patch $\mathbf{s}$, where $\mathbf{L} \in \mathbb{R}^{m \times m}$ is the *graph Laplacian matrix* of graph $\mathcal{G}$. Given a disparity map D produced by a deep stereo model, we compute the values of the graph Laplacian regularizers for the patches on D. The obtained values are summed up as a graph Laplacian regularization loss for CNN training.

For an effective regularization with graph Laplacian, it is critical to constructing a graph $\mathcal{G}$ properly. We employ the graph structure of [12, 26] which works well for disparity map denoising. For illustration, we first introduce the concept of *exemplar patches*. Exemplar patches are a set of K patches, $\mathbf{f}_k \in \mathbb{R}^m$ where $1 \leq k \leq K$, that are statistically related to the ground-truth patch $\mathbf{s}$. For instance, an exemplar patch can be a rough estimate of $\mathbf{s}$, or the co-located patch on the left image, *etc*. Our choices of the exemplar patches will be presented in Section 4.2. With the exemplar patches, the edge weight w_{ij} connecting pixel i and pixel j on patch $\mathbf{s}$ is given by

$$w_{ij} = \begin{cases} \exp\left(-d_{ij}^2\right) & \text{if } |d_{ij}| \leq \epsilon, \\ 0 & \text{otherwise,} \end{cases}$$

where ϵ is a threshold, d_{ij}^2 is a distance measure between pixel i and pixel j. Hence, the resulting graph $\mathcal{G}$ is an ϵ-neighborhood graph, *i.e.*, there is no edge connecting two pixels with a distance greater than ϵ. We choose an individual value of ϵ for each patch, making every vertex of the graph has at least 4 edges. The distance measure d_{ij}^2 is defined as follows:

$$d_{ij}^2 = \sum\nolimits_{k=1}^{K} \left(\mathbf{f}_k(i) - \mathbf{f}_k(j)\right)^2 + \alpha \cdot l_{ij}^2, \qquad (2)$$

where $\mathbf{f}_k(i)$ and $\mathbf{f}_k(j)$ denote the i-th and the j-th entries of $\mathbf{f}_k$, respectively, so the first term of (2) measures the Euclidean distance between pixels i and j in a K-dimensional space defined by the exemplar patches. l_{ij} is simply the spatial distance (length) between pixels i and j, and α is a constant weight, empirically set to be a small value 0.2.

The adjacency matrix of $\mathcal{G}$ is denoted as $\mathbf{A}$, where the (i, j)-th entry of $\mathbf{A}$ is w_{ij}. The degree matrix of $\mathcal{G}$ is a diagonal matrix $\mathbf{D}$, its i-th diagonal entry is $\sum_{j=1}^{m} w_{ij}$. Then the graph Laplacian $\mathbf{L}$ is given by $\mathbf{L} = \mathbf{D} - \mathbf{A}$, leading to the graph Laplacian regularizer $\mathbf{s}^{\mathrm{T}}\mathbf{L}\mathbf{s} \in \mathbb{R}$. From the analysis of [26], graph Laplacian regularizer is an *adaptive metric*. If the same edge (or gradient) pattern appears in the majority of the exemplar patches, minimizing the graph Laplacian regularizer promotes the very edge pattern; if the exemplar patches are inconsistent, graph Laplacian regularization leads to a smoothed patch. We exploit this property to guide a deep stereo model to selectively learn the desired details.

4.2. Training by Iterative Regularization

We borrow the notion of iterative regularization [24] for generalizing deep stereo models to novel domains, giving rise to the proposed zoom and learn approach. Suppose we have a deep stereo model $S(\cdot; \mathbf{\Theta}^{(0)})$ (parameterized by $\mathbf{\Theta}^{(0)}$) pre-trained with synthetic data. We also have a set

of N stereo pairs, $P_i = (L_i, R_i), 1 \leq i \leq N$, where the first N_{dom} of them are real stereo pairs of the target domain while the rest $N_{\text{syn}} = N - N_{\text{dom}}$ pairs are synthetic data, among which only the synthetic data has ground truth disparities D_i ($N_{\text{dom}} + 1 \leq i \leq N$).

We solve for a new set of model parameters $\mathbf{\Theta}^{(k+1)}$ at iteration k. For a constant $r > 1$, we first create a set of "ground-truths" for the N_{dom} real stereo pairs by zooming (up-sampling), *i.e.*,

$$D_i = \frac{1}{r} \cdot \downarrow_r \left(S\left(\uparrow_r (P_i); \mathbf{\Theta}^{(k)} \right) \right), 1 \leq i \leq N_{\text{dom}}. \quad (3)$$

From Section 3.2, D_i contains more details than $S(P_i; \mathbf{\Theta}^{(k)})$. We divide a disparity map D_i into M square patches tiling it where each patch is a vector of length m. The vectorization operator is denoted as $\text{vec}(\cdot)$ so that $\text{vec}(D_i) \in \mathbb{R}^{Mm}$. The m-by-Mm matrix extracting the j-th patch from D_i is denoted as $\mathbf{R}_j$. With these settings, we formulate the following iterative optimization problem,

$$\mathbf{\Theta}^{(k+1)} = \arg\min_{\mathbf{\Theta}} \sum_{i=1}^{N_{\text{dom}}} \sum_{j=1}^{M} \|\mathbf{s}_{ij} - \mathbf{d}_{ij}\|_1 + \lambda \cdot \mathbf{s}_{ij}^{\mathrm{T}} \mathbf{L}_{ij}^{(k)} \mathbf{s}_{ij} + \tau \cdot \sum_{i=N_{\text{dom}}+1}^{N} \|S(P_i; \mathbf{\Theta}) - D_i\|_1, \quad (4)$$

$$\text{s.t. } \mathbf{s}_{ij} = \mathbf{R}_j \cdot \text{vec}\left(S(P_i; \mathbf{\Theta})\right), \ \mathbf{d}_{ij} = \mathbf{R}_j \cdot \text{vec}\left(D_i\right).$$

Here $\mathbf{s}_{ij}$ and $\mathbf{d}_{ij}$ are the j-th patches of $S(P_i; \mathbf{\Theta})$ and D_i, respectively. λ and τ are positive constants. Our optimization problem (4) first minimizes over each patch on the N_{dom} stereo pairs: the first term (data term) drives $\mathbf{s}_{ij}$ to be similar to $\mathbf{d}_{ij}$; and the second term (smoothness term) is a graph Laplacian regularizer induces from the matrix $\mathbf{L}_{ij}^{(k)}$. The third term of (4) lets $\mathbf{\Theta}^{(k+1)}$ be a feasible deep stereo model; it literally means that: a deep stereo model works well for the target domain should also has reasonable performance on the synthetic data.

At iteration k, a graph $\mathcal{G}_{ij}^{(k)}$ ($1 \leq i \leq N_{\text{dom}}$, $1 \leq j \leq M$), and hence the corresponding graph Laplacian, $\mathbf{L}_{ij}^{(k)}$, are pre-computed for calculating a loss $\mathbf{s}_{ij}^{\mathrm{T}} \mathbf{L}_{ij}^{(k)} \mathbf{s}_{ij}$. We choose the following three exemplar patches for building $\mathcal{G}_{ij}^{(k)}$:

$$\begin{aligned} \mathbf{f}_{\text{left}} &= w_{\text{left}} \cdot \mathbf{R}_j \cdot \text{vec}(L_i), \\ \mathbf{f}_{\text{curr}} &= w_{\text{curr}} \cdot \mathbf{R}_j \cdot \text{vec}(S(P_i; \mathbf{\Theta}^{(k)})), \\ \mathbf{f}_{\text{fine}} &= w_{\text{fine}} \cdot \mathbf{R}_j \cdot \text{vec}(D_i) = w_{\text{fine}} \cdot \mathbf{d}_{ij}, \end{aligned}$$

where w_{left}, w_{curr} and w_{fine} are constants. In other words, $\mathbf{f}_{\text{left}}$, $\mathbf{f}_{\text{curr}}$, and $\mathbf{f}_{\text{fine}}$ are the j-th patches of the left image L_i, the current prediction $S(P_i; \mathbf{\Theta}^{(k)})$ and the finer-grain prediction D_i (3), respectively.

Our chosen exemplar patches lead to a graph Laplacian regularizer that discriminatively retain the desired details from $\mathbf{f}_{\text{fine}}$ whilst smoothing out possible artifacts on both $\mathbf{f}_{\text{curr}}$ and $\mathbf{f}_{\text{fine}}$. We analyze how the patches $\mathbf{f}_{\text{left}}$, $\mathbf{f}_{\text{curr}}$ and $\mathbf{f}_{\text{fine}}$ affects the behavior of the graph Laplacian:

(i) Suppose a desired object boundary (denoted by A) does not appear in the current predicted patch $\mathbf{f}_{\text{curr}}$. However, it has appeared in the finer-grain patch $\mathbf{f}_{\text{fine}}$ by virtue of scale diversity (Section 3.2), then A should also appear in $\mathbf{f}_{\text{left}}$; otherwise the CNN cannot generate A on $\mathbf{f}_{\text{fine}}$. In this case, both $\mathbf{f}_{\text{fine}}$ and $\mathbf{f}_{\text{left}}$ have boundary A, resulting in a Laplacian $\mathbf{L}_{ij}^{(k)}$ that promotes A on $\mathbf{s}_{ij}$.

(ii) Suppose due to generalization glitches, an undesired pattern (denoted as B) is produced in one exemplar patch, $\mathbf{f}_{\text{curr}}$ or $\mathbf{f}_{\text{fine}}$. Since B is absence in the other exemplar patches, the corresponding graph Laplacian $\mathbf{L}_{ij}^{(k)}$ penalizes B on $\mathbf{s}_{ij}$.

Hence, our graph Laplacian regularizer guides the CNN to only learn the meaningful details.

4.3. Practical Algorithm

Iteratively solving the optimization problem (4) can be achieved by training the model $S(\cdot; \mathbf{\Theta})$ with standard back-propagation [20]. We hereby present how to use the proposed formulation for finetuning a pre-trained model in practice. Since a disparity map D_i is tiled by M patches, $\mathbf{d}_{ij}$ with $1 \leq j \leq M$, the first term in (4) equals $\sum_{i=1}^{N_{\text{dom}}} \|S(P_i; \mathbf{\Theta}) - D_i\|_1$. Hence, the objective of (4) can be rewritten as:

$$\mathbf{\Theta}^{(k+1)} = \arg\min_{\mathbf{\Theta}} \sum_{i=1}^{N_{\text{dom}}} \|S(P_i; \mathbf{\Theta}) - D_i\|_1 + \tau \cdot \sum_{i=N_{\text{dom}}+1}^{N} \|S(P_i; \mathbf{\Theta}) - D_i\|_1 + \lambda \cdot \sum_{i=1}^{N_{\text{dom}}} \sum_{j=1}^{M} \mathbf{s}_{ij}^{\mathrm{T}} \mathbf{L}_{ij}^{(k)} \mathbf{s}_{ij}, \quad (5)$$

We see that the first two terms of (5) are simply L1 loss with different weightings for the target domain and the synthetic data. The third term is the proposed graph Laplacian regularization loss, we discuss its backpropagation in the supplementary material.

In general, there are a lot of training examples (N is large), yet in practice, every training iteration can only take in a batch of $n \ll N$ training examples and perform stochastic gradient descent. As a result, we shuffle all the N stereo pairs and sequentially take out n of them to form a training batch for the current iteration. For a synthetic stereo pair P_i ($N_{\text{dom}} + 1 \leq i \leq N$) in the batch, we directly use its L1 loss for backpropagation since its ground-truth D_i is

Algorithm 1 Zoom and learn (ZOLE)

Input: Pre-trained deep stereo model $S(\cdot;\mathbf{\Theta}^{(0)})$, training data $\{P_i\}_{i=1}^{N_{\text{dom}}}$ and $\{P_i, D_i\}_{i=N_{\text{dom}}+1}^{N}$
Shuffle the training data to form a list ℓ
for $k = 0$ to $k_{\max} - 1$ **do**
  for $b = 1$ to n **do**
    Draw an index i from list ℓ
    if $i \leq N_{\text{dom}}$ **then**
      Compute D_i, then compute $S(P_i;\mathbf{\Theta}^{(k)})$ and hence the graph Laplacian matrices $\mathbf{L}_{ij}^{(k)}$
    end if
    Insert $\{P_i, D_i\}$ to the current batch
  end for
  Use the formed training batch and the pre-computed Laplacian matrices to perform a step of gradient descent
  if $\text{mod}(k+1, t) = 0$ **then**
    Perform validation, update $\mathbf{\Theta}^{(\text{bst})}$ and $v^{(\text{bst})}$ if needed
  end if
end for
Output: model parameters $\mathbf{\Theta}^{(\text{bst})}$

known. Otherwise, for a stereo pair P_i with $1 \leq i \leq N_{\text{dom}}$ in the batch, we first feed its up-sampled version to the CNN for computing the finer-grain "ground-truth" D_i, we also compute the current estimate $S(P_i;\mathbf{\Theta}^{(k)})$ and hence the graph Laplacian matrices $\mathbf{L}_{ij}^{(k)}$ for each patch. With D_i and the pre-computed $\mathbf{L}_{ij}^{(k)}$'s, $1 \leq j \leq M$, both the L1 loss and the graph Laplacian regularization loss are employed for backpropagation.

For every t training iterations, we perform a validation procedure with left-right consistency, using another set of $N_{\text{dom}}^{(\text{v})}$ stereo pairs in the target domain. We first estimate the disparity maps with the up-to-date model then synthesize $N_{\text{dom}}^{(\text{v})}$ left images with the estimated disparity maps and the right images. Then we compute the peak signal-to-noise ratios (PSNRs) between the synthesized left images and the genuine ones. The average PSNR reflects the performance of the current model. During the training process, we keep track of the best PSNR value $v^{(\text{bst})}$ and its corresponding model $\mathbf{\Theta}^{(\text{bst})}$. After $k_{\max}$ training iterations, we terminate the training and output $\mathbf{\Theta}^{(\text{bst})}$. Algorithm 1 summarizes the key steps of our self-adaptation approach.

5. Experimentation

In this section, we generalize deep stereo matching for two different domains in the real world: daily scenes captured by smartphone cameras, and street views from the perspective of a driving car (the KITTI dataset [6]). We again choose the representative DispNetC [23] architecture for our experiments.

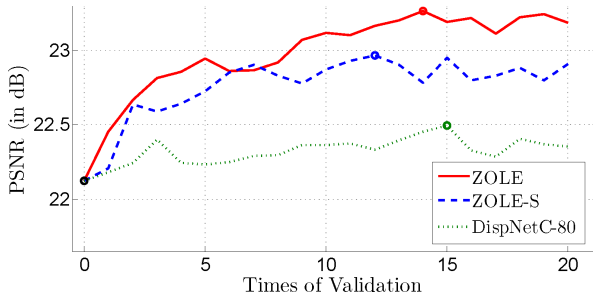


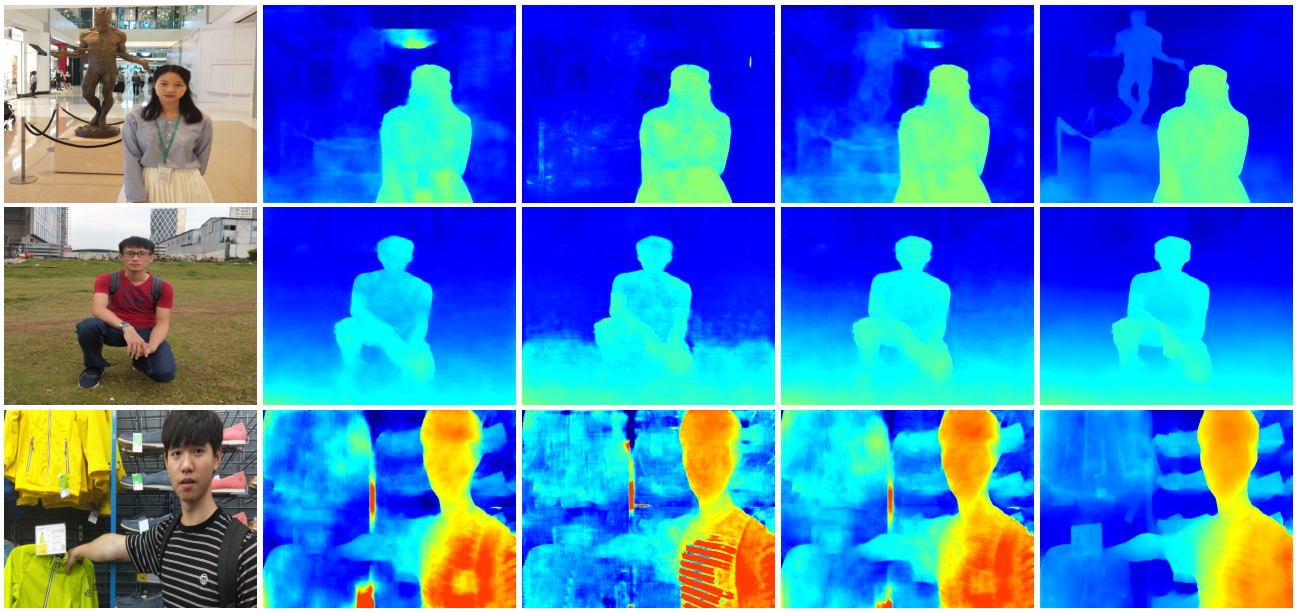
Figure 3. Validation performance of three different models during finetuning. The curves are plotted in terms of the average PSNR between the synthesized left images and the genuine ones.

5.1. Daily Scenes from Smartphones

Recently, many companies (*e.g.*, Apple, Samsung) have equipped their smartphones with two rear-facing cameras. Stereo pairs collected by these cameras have small disparity and possibly contaminated by noise due to the small area of their image sensors. With two views of the same scene, stereo matching is applied to estimate a dense disparity map for subsequent applications, *e.g.*, synthetic bokeh [2] and segmentation [22].

We aim at generalizing the released DispNetC model (pre-trained with the FlyingThings3D dataset [23]) for daily scenes captured by smartphones cameras. For this purpose, we used various models of smartphones to collect $N_{\text{dom}} = 1900$, $N_{\text{dom}}^{(\text{v})} = 320$ and $N_{\text{dom}}^{(\text{t})} = 320$ stereo pairs for training, validation, and testing, respectively. These stereo pairs contain daily scenes like human portraits and objects taken in various indoor and outdoor environments (*e.g.*, library, office, playground, park). All the collected images are rectified and resized to 768×768, their ground-truth disparity maps are unknown. Besides, we use the FlyingThings3D dataset for synthetic training examples in our method, they are also resized to 768×768. Since their original size is 960×960, their disparity maps need to be rescaled by a factor of 0.8. To cater for the small disparity values of the smartphone data, we only keep those synthetic examples with maximum disparity no greater than 80 after rescaling, leading to 9619 available examples. Among them, $N_{\text{syn}} = 8000$ examples are used for training and the rest $N_{\text{syn}}^{(\text{t})} = 1619$ are withheld for testing. We call this set of data FlyingThings3D-80. In our experiments, all stereo pairs have intensity ranges from 0 to 255.

The Caffe framework [16] is employed to implement our method. During training, we randomly crop the images to 640×640 before passing them to a CNN, and let the patch size be 20×20 for building the graphs, resulting in $32 \times 32 = 1024$ graphs for each training example. We modify the L1 loss layer of [23] to capture the first two terms of (5): for a synthetic pair, its L1 loss is weighted by 1.2 times, otherwise the weight is 1. We empirically set $w_{\text{left}} = 0.3$,



Left image | Tonioni *et al.* [34] | DispNetC | ZOLE-S | ZOLE

Figure 4. Visual comparisons of several models on the test set of our smartphone data. This figure shows fragments of left images and the corresponding disparity maps obtained with different models. It is clear that our ZOLE approach produces superior disparity results.

Table 2. Performance comparison of our obtained zoom and learn (ZOLE) model and the other four models.

Dataset	**Metric**		**Model**									
			Tonioni [34]		DispNetC		DispNetC-80		ZOLE-S		ZOLE	
Smartphone	PSNR	SSIM	22.92	0.845	21.99	0.790	22.39	0.817	22.84	0.851	**23.12**	**0.855**
FlyingThings3D-80	EPE	3ER	1.08	6.79%	1.03	5.63%	**0.93**	**5.11%**	1.10	6.88%	1.11	6.54%

$w_{\text{fine}} = 0.8$ and $w_{\text{curr}} = 1$, all the computed $\mathbf{s}_{ij}^{\mathrm{T}}\mathbf{L}_{ij}^{(k)}\mathbf{s}_{ij}$ are averaged then weighted by 1.5 times for a loss (the third term in (5)). We have tried out different up-sampling ratios r's ranging from 1.2 to 2 for computing D_i, and found the the obtained CNNs have similar performance. In our experiments, we let $r = 1.5$. Data augmentation is introduced to the synthetic stereo pairs. For each individual view in a synthetic pair, Gaussian noise ($\sigma \in \{0, 10, 15\}$) are randomly added. The brightness of each image channel are also randomly adjusted (by a factor of $\rho \in \{0.8, 1, 1.2\}$). We let the batch size be 6, the learning rate be 5×10^{-5}, and finetune the model for $k_{\max} = 10^4$ iterations, validation is performed every 500 iterations.

We first study the following models:

(i) ZOLE: Generalize the pre-trained model for smartphone stereo pairs with our method;

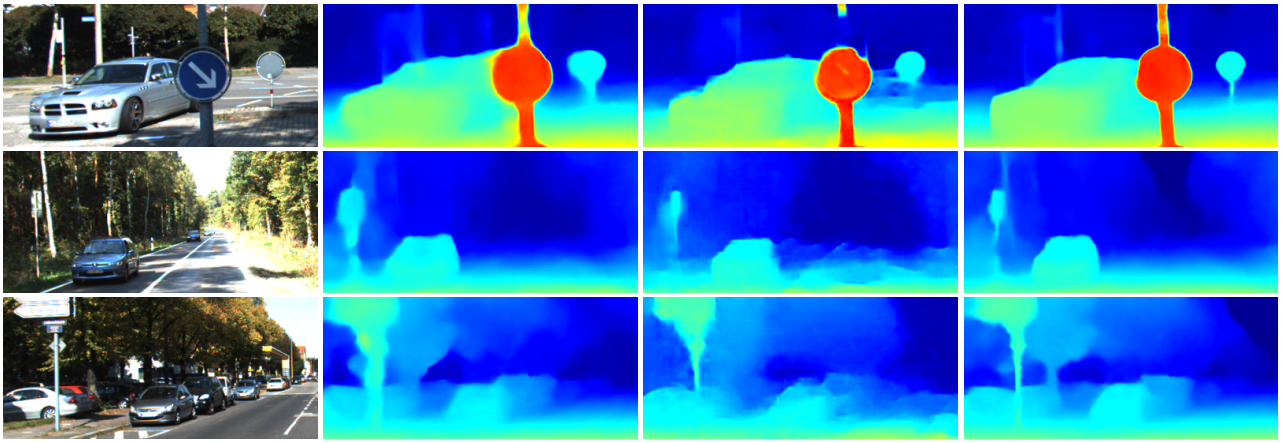
(ii) ZOLE-S: Remove graph regularization and simply let the CNN iteratively learn its own finer-grain outputs;

(iii) DispNetC-80: Finetune the pre-trained model on the FlyingThings3D-80 examples;

(iv) DispNetC: Released model pre-trained with FlyingThings3D [23].

The very recent method [34] by Tonioni *et al.* also finetunes a pre-trained model using stereo pairs from the target domain. They first estimate disparity maps for the target domain with AD-CENCUS [36]. To finetune the model, they treat the obtained disparity maps as "ground-truths" while taking a confidence measure [29] into account. For comparison, we finetune a model with their released code under their recommended settings.

Since the stereo pairs of smartphones do not have ground-truth disparities, we evaluate the performance of a model in a way similar to the validation process presented in Section 4.3. We synthesize the left images with the estimated disparities and the right images, then measure the difference between the synthesized left images and the genuine ones, using both PSNR and SSIM as the difference metrics. For testing or validation, all the stereo pairs are fed to the CNN at a fixed resolution of 1024×1024. Figure 3 plots the performance of ZOLE, ZOLE-S and DispNetC-80 on the validation set of the smartphone data during training (measured in terms of average PSNR of the synthesized left images). Besides, Table 2 presents the performance of all the aforementioned models, on both the test sets of the smartphone data and FlyingThings3D-80. We use end-point-



Left image | Tonioni *et al.* [34] | DispNetC | ZOLE

Figure 5. Visual comparisons of several models on the KITTI stereo 2015 dataset, where our ZOLE method produces accurate fine details.

Table 3. Objective performance on the KITTI stereo 2015 dataset.

Metric	Model			
	Tonioni [34]	DispNetC	ZOLE-S	ZOLE
EPE	1.27	1.64	1.34	**1.25**
3ER	7.06%	11.41%	7.56%	**6.76%**

error (EPE) and three-pixel error rate (3ER) as the evaluation metrics for the FlyingThings3D-80 dataset. Compared to the models trained only with the synthetic data (DispNetC and DispNetC-80), the one obtained with our method (ZOLE) achieves the best PSNR and SSIM performance. Figure 4 shows visual comparisons of four models on the test sets of the smartphone data. One can clearly see that, our approach leads to smooth disparities with very sharp details, while disparity maps produced by other models may be blurry or contain artifacts.

5.2. Driving Scenes of KITTI

Our self-adaptation method is also applied to generalize the pre-trained DispNetC model to the KITTI stereo 2015 dataset [6], which contains dynamic street views from the perspective of a driving car. The KITTI stereo 2015 dataset have 800 stereo pairs. Among them, 200 examples have publicly available (sparse) ground-truth disparity maps. They are employed for testing, while the rest 600 pairs are used for validation. For training, we first gather $N_{\text{syn}} = 9000$ stereo pairs randomly from the FlyingThings3D dataset. Since the KITTI 3D object 2017 dataset [6] have more than 10k stereo pairs of the same characteristics as KITTI stereo 2015, we randomly pick $N_{\text{dom}} = 3000$ stereo pairs from it for training. During training, we adopt similar settings as presented in Section 5.1. However, in this scenario, all images are resized to 1280×400 then randomly cropped to 1024×384 before passing to the CNN for training.

We hereby compare our approach, ZOLE, with models obtained with ZOLE-S and [34]; while the original DispNetC model is adopted as a baseline. For a fair comparison, all the images are resized to 1280×384 before feeding to the network. Table 3 presents the objective metrics of ZOLE, along with those of the competing methods. We see that our method has the best objective performance, while the method of Tonioni *et al.* also provides a reasonable gain. Figure 5 shows several fragments of the resulting disparity images. One can see that our method provides accurate edges even for very fine details.

More results and discussions are provided in the supplementary material. Our method is essentially different from those deep stereo algorithms relying on left-right consistency for backpropagation [38, 39]. Hence, it is possible to combine our rationale—discriminatively learns from the finer-grain outputs—with these methods to achieve further improvements. Moreover, the same rationale can be applied to other pixel-wise regression/classification problems, *e.g.*, optical flow estimation [15, 23] and segmentation [22]. We leave these research directions for future exploration.

6. Conclusion

Due to the deficiency of ground-truth data, it is difficult to generalize a pre-trained deep stereo model to a novel domain. To tackle this problem, we propose a self-adaption approach for CNN training without ground-truth disparity maps of the target domain. We first observe and analyze two phenomena, namely, generalization glitches and scale diversity. To exploit scale diversity while avoiding generalization glitches, we let the model learn from its own finer-grain output, while a graph Laplacian regularization is imposed to selectively keep the desired edges and smoothing out the artifacts. We call our method zoom and learn, or ZOLE for short. It is applied to two domains: daily scenes collected by smartphone cameras and street views captured from the perspective of a driving car.

References

[1] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for scene segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017. 3

[2] J. T. Barron, A. Adams, Y. Shih, and C. Hernández. Fast bilateral-space stereo for synthetic defocus. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4466–4474, 2015. 6

[3] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conference on Computer Vision*, pages 611–625. Springer, 2012. 3

[4] A. Elmoataz, O. Lezoray, and S. Bougleux. Nonlocal discrete regularization on weighted graphs: A framework for image and manifold processing. *IEEE Transactions on Image Processing*, 17(7):1047–1060, 2008. 4

[5] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4340–4349, 2016. 3

[6] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? The KITTI vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361. IEEE, 2012. 1, 2, 3, 6, 8

[7] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 963–968. Ieee, 2011. 1

[8] S. Gidaris and N. Komodakis. Detect, replace, refine: Deep structured prediction for pixel wise labeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5248–5257, 2017. 2

[9] G. Gilboa and S. Osher. Nonlocal linear image regularization and supervised segmentation. *Multiscale Modeling & Simulation*, 6(2):595–630, 2007. 4

[10] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[11] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3279–3286, 2015. 2

[12] M. Hein, J.-Y. Audibert, and U. v. Luxburg. Graph Laplacians and their convergence on random neighborhood graphs. *Journal of Machine Learning Research*, 8(Jun):1325–1368, 2007. 4

[13] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008. 1

[14] W. Hu, G. Cheung, and M. Kazui. Graph-based dequantization of block-compressed piecewise smooth images. *IEEE Signal Processing Letters*, 23(2):242–246, 2016. 4

[15] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2, 4, 8

[16] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 675–678. ACM, 2014. 6

[17] A. K. Katsaggelos. Iterative image restoration algorithms. *Optical engineering*, 28(7):735–748, 1989. 2

[18] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry. End-to-end learning of geometry and context for deep stereo regression. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1, 2

[19] Y. Kuznietsov, J. Stückler, and B. Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[20] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. 5

[21] X. Liu, G. Cheung, X. Wu, and D. Zhao. Random walk graph Laplacian-based smoothness prior for soft decoding of jpeg images. *IEEE Transactions on Image Processing*, 26(2):509–524, 2017. 2

[22] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015. 3, 4, 6, 8

[23] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. 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 (CVPR)*, pages 4040–4048, 2016. 1, 2, 3, 4, 6, 7, 8

[24] P. Milanfar. A tour of modern image filtering: New insights and methods, both practical and theoretical. *IEEE Signal Processing Magazine*, 30(1):106–128, 2013. 2, 4

[25] S. Osher, M. Burger, D. Goldfarb, J. Xu, and W. Yin. An iterative regularization method for total variation-based image restoration. *Multiscale Modeling & Simulation*, 4(2):460–489, 2005. 2

[26] J. Pang and G. Cheung. Graph Laplacian regularization for image denoising: Analysis in the continuous domain. *IEEE Transactions on Image Processing*, 26(4):1770–1785, 2017. 2, 4

[27] J. Pang, G. Cheung, A. Ortega, and O. C. Au. Optimal graph Laplacian regularization for natural image denoising. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2294–2298. IEEE, 2015. 2

[28] J. Pang, W. Sun, J. S. Ren, C. Yang, and Q. Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *ICCV Workshop on Geometry Meets Deep Learning*, Oct 2017. 1, 2

[29] M. Poggi and S. Mattoccia. Learning from scratch a confidence measure. In *BMVC*, 2016. 7

[30] T. S. Rappaport et al. *Wireless communications: Principles and practice*, volume 2. prentice hall PTR New Jersey, 1996. 4

[31] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015. 3

[32] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition*, pages 31–42. Springer, 2014. 1, 2

[33] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013. 2, 4

[34] A. Tonioni, M. Poggi, S. Mattoccia, and L. Di Stefano. Unsupervised adaptation for deep stereo. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 7, 8

[35] B. Ummenhofer, H. Zhou, J. Uhrig, N. Mayer, E. Ilg, A. Dosovitskiy, and T. Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017. 2

[36] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *European Conference on Computer Vision*, pages 151–158. Springer, 1994. 7

[37] J. Zbontar and Y. LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2, 2016. 2

[38] Y. Zhong, Y. Dai, and H. Li. Self-supervised learning for stereo matching with self-improving ability. *arXiv preprint arXiv:1709.00930*, 2017. 2, 8

[39] C. Zhou, H. Zhang, X. Shen, and J. Jia. Unsupervised learning of stereo matching. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 8