HITNet: Hierarchical Iterative Tile Refinement Network for Real-time Stereo Matching

Vladimir Tankovich   Christian Häne   Yinda Zhang   Adarsh Kowdle   Sean Fanello

Sofien Bouaziz

Google

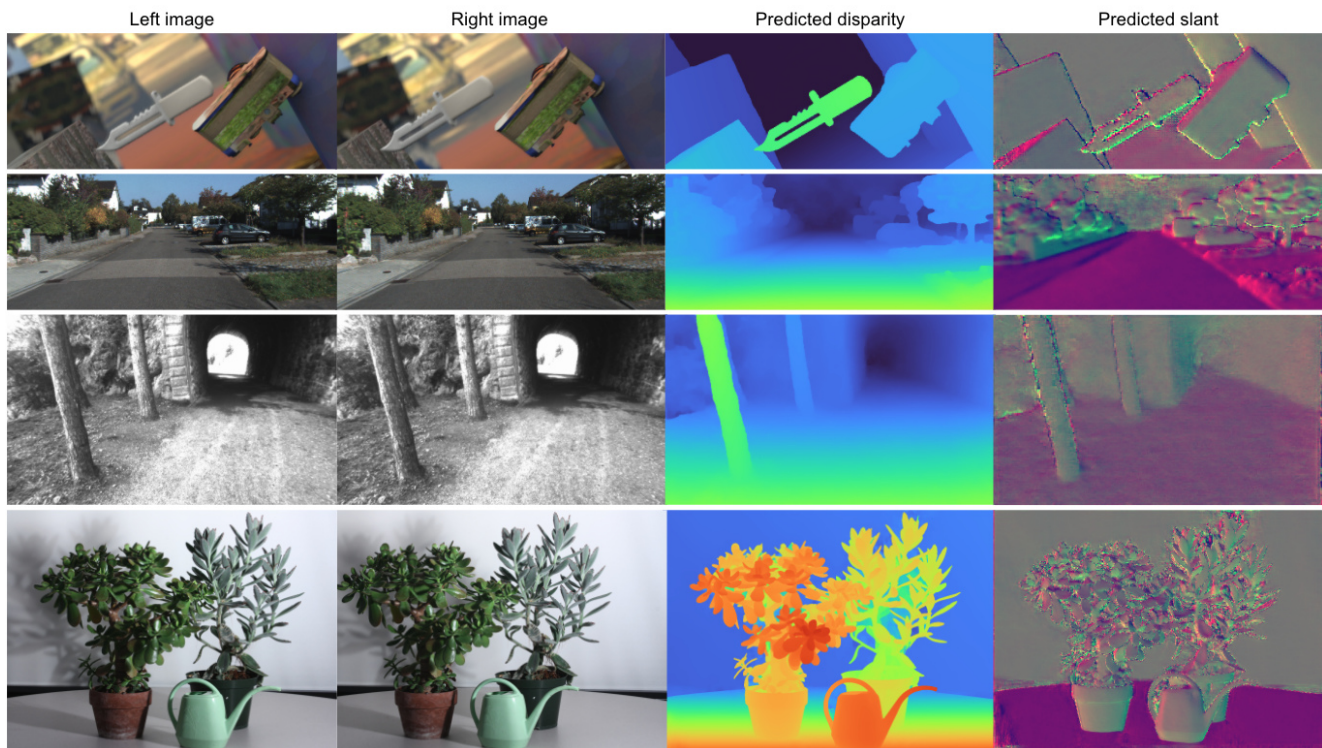{vtankovich, chaene, yindaz, adarshkowdle, seanfa, sofien}@google.com

Figure 1: Example result of HITNet on the SceneFlow, KITTI, ETH3D and Middlebury datasets. Our approach predicts accurate depth with crisp edges. HITNet obtains state-of-art results on KITTI, ETH3D and Middlebury-v3 benchmarks.

Abstract

This paper presents HITNet, a novel neural network architecture for real-time stereo matching. Contrary to many recent neural network approaches that operate on a full cost volume and rely on 3D convolutions, our approach does not explicitly build a volume and instead relies on a fast multi-resolution initialization step, differentiable 2D geometric propagation and warping mechanisms to infer disparity hypotheses. To achieve a high level of accuracy, our network not only geometrically reasons about disparities but also infers slanted plane hypotheses allowing to more accu-rately perform geometric warping and upsampling opera-tions. Our architecture is inherently multi-resolution allow-ing the propagation of information across different levels. Multiple experiments prove the effectiveness of the proposed approach at a fraction of the computation required by state-of-the-art methods. At the time of writing, HITNet ranks 1st-3rd on all the metrics published on the ETH3D website for two view stereo, ranks 1st on most of the metrics amongst all the end-to-end learning approaches on Middlebury-v3, ranks 1st on the popular KITTI 2012 and 2015 benchmarks among the published methods faster than 100ms.

1. Introduction

Recent research on depth from stereo matching has largely focused on developing accurate but computationally expensive deep learning approaches. Large convolutional neural networks (CNNs) can often use up to a second or even more to process an image pair and infer a disparity map. For active agents such as mobile robots or self driving cars such a high latency is undesirable and methods which are able to process an image pair in a matter of milliseconds are required instead. Despite this, only 4 out of the top 100 methods on the KITTI 2012 leaderboard are published approaches that take less than 100ms[1].

A common pattern in end-to-end learning based approaches to computational stereo is utilizing a CNN which is largely unaware of the geometric properties of the stereo matching problem. In fact, initial end-to-end networks were based on a generic U-Net architecture [37]. Subsequent works have pointed out that incorporating explicit matching cost volumes encoding the cost of assigning a disparity to a pixel, in conjunction with 3D convolutions provides a notable improvement in terms of accuracy but at the cost of significantly increasing the amount of computation [24]. Follow up work [25] showed that a downsampled cost volume could provide a reasonable trade-off between speed and accuracy. However, the downsampling of the cost volume comes at the price of sacrificing accuracy.

Multiple recent stereo matching methods [51, 12, 28] have increased the efficiency of disparity estimation for active stereo while maintaining a high level of accuracy. These methods are mainly built on three intuitions: Firstly, the use of compact/sparse features for fast high resolution matching cost computation; Secondly, very efficient disparity optimization schemes that do not rely on the full cost volume; Thirdly, iterative image warps using slanted planes to achieve high accuracy by minimizing image dissimilarity. All these design choices are used without explicitly operating on a full 3D cost volume. By doing so these approaches achieve very fast and accurate results for active stereo but they do not directly generalize to passive stereo due to the lack of using a powerful machine learning system. This therefore raises the question if such mechanisms can be integrated into neural network based stereo-matching systems to achieve efficient and accurate results opening up the possibility of using passive stereo based depth sensing in latency critical applications.

We propose HITNet, a framework for neural network based depth estimation which overcomes the computational disadvantages of operating on a 3D volume by integrating image warping, spatial propagation and a fast *high resolution initialization step* into the network architecture, while

keeping the flexibility of a learned representation by allowing features to flow through the network. The main idea of our approach is to represent image tiles as planar patches which have a learned compact feature descriptor attached to them. The basic principle of our approach is to fuse information from the high resolution initialization and the current hypotheses using spatial propagation. The propagation is implemented via a convolutional neural network module that updates the estimate of the planar patches and their attached features. In order for the network to iteratively increase the accuracy of the disparity predictions, we provide the network a local cost volume in a narrow band (± 1 disparity) around the planar patch using in-network image warping allowing the network to minimize image dissimilarity. To reconstruct fine details while also capturing large texture-less areas we start at low resolution and hierarchically upsample predictions to higher resolution. A critical feature of our architecture is that at each resolution, matches from the initialization module are provided to facilitate recovery of thin structures that cannot be represented at low resolution. An example output of our method shows how our network recovers very accurate boundaries, fine detail and thin structures in Fig. 1.

To summarize, our main contributions are:

- A fast multi-resolution initialization step that computes *high resolution matches* using learned features.

- An efficient 2D disparity propagation that makes use of slanted support windows with learned descriptors.

- State-of-art results in popular benchmarks using a fraction of the computation compared to other methods.

2. Related Work

Stereo matching has been an active field of research for decades [36, 43, 19]. *Traditional methods* utilize handcrafted schemes to find local correspondences [55, 22, 4, 21] and global optimization to exploit spatial context [14, 26, 27]. The run-time efficiency of most of these approaches are correlated with the size of the disparity space, which prevents real-time applications. *Efficient algorithms* [31, 34, 5, 3] avoid searching the full disparity space by using patchmatch [1] and super-pixel [31] techniques. A family of machine learning based approaches, using random forest and decision trees, are able to establish correspondences quickly [10, 11, 12, 13]. However, these methods require either camera specific learning or post processing. Recently, *deep learning* brought big improvements to stereo matching. Early works trained siamese networks to extract patch-wise features or predict matching costs [35, 58, 56, 57]. End-to-end networks have been proposed to learn all steps jointly, yielding more accurate results [46, 37, 40, 47].

---

[1]Additional approaches faster than 100ms are on the leaderboard but the algorithms are unpublished and hence it is unknown how the results were achieved.

A key component in modern architectures is a cost volume layer [24] (or correlation layer [23]), allowing the network to run per-pixel feature matching. To speed up computation, cascaded models [39, 7, 32, 16, 53, 59] have been proposed to search in disparity space in a coarse-to-fine fashion. In particular, [39] uses multiple residual blocks to improve the current disparity estimate. The recent work [53] relies on a hierarchical cost volume, allowing the method to be trained on high resolution images and generate different resolutions *on demand*. All these methods rely on expensive cost-volume filtering operations using 3D convolutions [53] or multiple refinement layers [39], preventing real-time performance. Fast approaches [25, 60] *downsample* the cost volume in spatial and disparity space and attempt to recover fine details by edge-aware upsampling layers. These methods show real-time performance but sacrifice accuracy especially for thin structures and edges since they are missing in the low-res initialization.

Our method is inspired by classical stereo matching methods, which aim at propagating good sparse matches [12, 13, 51]. In particular, Tankovich et al. [51] proposed a hierarchical algorithm that makes use of slanted support windows to amortize the matching cost computation in *tiles*. Inspired by this work, we propose an end-to-end learning approach that overcomes the issues of hand-crafted algorithms, while maintaining computational efficiency.

PWC-Net [49], although designed for optical flow estimation, is related to our approach. The method uses a *low resolution* cost volume with multiple refinement stages via image warps and local matching cost computations. Thereby following the classical pyramidal matching approach where a low resolution result gets hierarchically upsampled and refined by initializing the current level with the previous level's solution. In contrast we propose a fast, multi-scale, *high resolution* initialization which is able to recover fine details that cannot be represented at low resolution. Finally, our refinement steps produce local slanted plane approximations, which are used to predict the final disparities, as opposed to standard bilinear warping and interpolation employed in [49].

3. Method

The design of HITNet, follows the principles of traditional stereo matching methods [43]. In particular, we observe that recent efficient methods rely on the three following steps: ① compact feature representations are extracted [12, 13]; ② a high resolution disparity initialization step utilizes these features to retrieve feasible hypotheses; ③ an efficient propagation step refines the estimates using slanted support windows [51]. Motivated by these observations, we represent the disparity map as planar tiles at various resolutions and attach a learnable feature vector to each tile hypothesis (Sec. 3.1). This allows our network to learn which

information about a small part of the disparity map is relevant to further improving the result. This can be interpreted as an efficient and sparse version of the learnable 3D cost volumes that have shown to be beneficial [24].

The overall method is depicted in Fig. 2. Our feature extraction module relies on a very small U-Net [41], where the multi-resolution features of the decoder are used by the rest of the pipelines. These features encode multi-scale details of the image, similar to [7] (Sec. 3.2). Once the features are extracted, we initialize disparity maps as fronto parallel tiles at multiple resolutions. To do so, a matcher evaluates multiple hypotheses and selects the one with the lowest ℓ_1 distance between left and right view feature. Additionally, a compact per-tile descriptor is computed using a small network (Sec. 3.3). The output of the initialization is then passed to a propagation stage, which acts similarly to the approximated Conditional Random Field solution used in [12, 51]. This stage hierarchically refines the tile hypotheses in an iterative fashion (Sec. 3.4).

3.1. Tile Hypothesis

We define a tile hypothesis as a planar patch with a learnable feature attached to it. Concretely, it consists of a geometric part describing a slanted plane with the disparity d and the gradient of disparity in x and y directions (d_x, d_y), and a learnable part $\mathbf{p}$ which we call tile feature descriptor. The hypothesis is therefore described as a vector which encodes a slanted 3D plane,

$$\mathbf{h} = [\underbrace{d, d_x, d_y}_{\text{plane}}, \underbrace{\mathbf{p}}_{\text{descriptor}}] \tag{1}$$

The tile feature descriptor is a learned representation of the tile which allows the network to attach additional information to the tile. This could for example be matching quality or local surface properties such as how planar the geometry actually is. We do not constrain this information and learned it end-to-end from the data instead.

3.2. Feature Extractor

The feature extractor provides a set of multi-scale feature maps $\mathcal{E} = \{\mathbf{e}_0, \dots \mathbf{e}_M\}$ that are used for initial matching and for warping in the propagation stage. We denote a feature map as $\mathbf{e}_l$ and an embedding vector $\mathbf{e}_{l,x,y}$ for locations x, y at resolution $l \in 0, \dots, M$, 0 being the original image resolution and M denoting a $2^M \times 2^M$ downsampled resolution. A single embedding vector $\mathbf{e}_{l,x,y}$ is composed of multiple feature channels. We implement the feature extractor $\mathcal{E} = \mathcal{F}(\mathbf{I}; \boldsymbol{\theta}_\mathcal{F})$ as a U-Net like architecture [41, 33], i.e. an encoder-decoder with skip connections, with learnable parameters $\boldsymbol{\theta}_\mathcal{F}$. The network is composed of strided convolutions and transposed convolutions with leaky ReLUs as non-linearities. The set of feature maps $\mathcal{E}$ that we use in
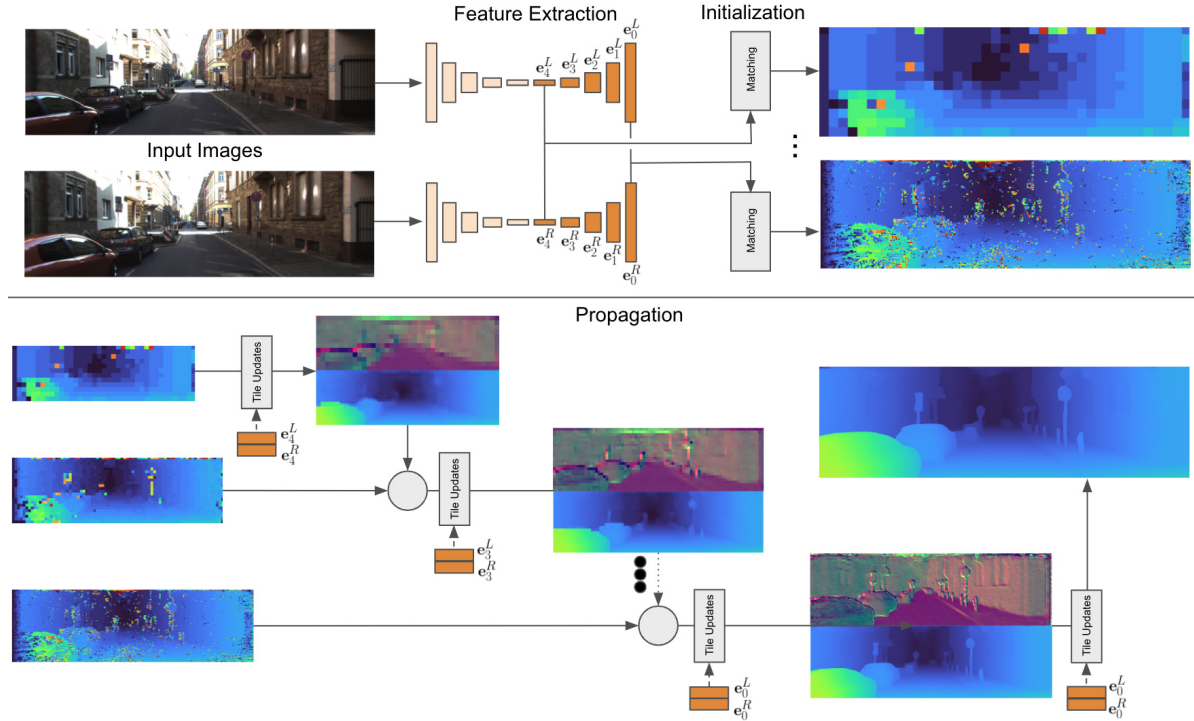
Figure 2: Overview of the proposed framework. (Top) A U-Net is used to extract features at multiple scales from left and right images. The initialization step is run on each scale of the extracted features. This step operates on *tiles* of 4×4 feature regions and evaluates multiple disparity hypotheses. The disparity with the minimum cost is selected. (Bottom) The output of the initialization is then used at propagation stage to refine the predicted disparity hypotheses using slanted support windows.

the remainder of the network are the outputs of the upsampling part of the U-Net at all resolutions. This means that even the high resolution features do contain some amount of spatial context. In more details, one down-sampling block of the U-Net has a single 3×3 convolution followed by a 2×2 convolution with stride 2. One up-sampling block applies 2×2 stride 2 transpose convolutions to up-sample results of coarser U-Net resolution. Features are concatenated with skip-connection, and a 1×1 convolution followed by a 3×3 convolution are applied to merge the skipped and upsampled feature for the current resolution. Each upsampling block generates a feature map $\mathbf{e}_l$, which is then used for downstream tasks and also further upsampled in the U-Net to generate a higher resolution feature map. We run the feature extractor on the left and the right image and obtain two multi-scale representations $\mathcal{E}^L$ and $\mathcal{E}^R$.

3.3. Initialization

The goal of the initialization is to extract an initial disparity d^{init} and a feature vector $\mathbf{p}^{\text{init}}$ for each tile at various resolutions. The output of the initialization is fronto-parallel tile hypotheses of the form $\mathbf{h}^{\text{init}} = [d^{\text{init}}, 0, 0, \mathbf{p}^{\text{init}}]$.

Tile Disparity. In order to keep the initial disparity resolution high we use overlapping tiles along the x direction,

i.e. the width, in the right (secondary) image but we still use non-overlapping tiles in the left (reference) image for efficient matching. To extract the tile features we run a 4×4 convolution on each extracted feature map $\mathbf{e}_l$. The strides for the left (reference) image and the right (secondary) image are different to facilitate the aforementioned overlapping tiles. For the left image we use strides of 4×4 and for the right image we use strides of 4×1, which is crucial to maintain the full disparity resolution to maximize accuracy. This convolution is followed by a leaky ReLU and an MLP.

The output of this step will be a new set of feature maps $\tilde{\mathcal{E}} = \{\tilde{\mathbf{e}}_0, \ldots, \tilde{\mathbf{e}}_M\}$ with per tile features $\tilde{\mathbf{e}}_{l,x,y}$. Note that the width of the feature maps in $\tilde{\mathcal{E}}^L$ and $\tilde{\mathcal{E}}^R$ are now different. The per-tile features are explicitly matched along the scan lines. We define the matching cost ϱ at location (x, y) and resolution l with disparity d as:

$$\varrho(l, x, y, d) = \|\tilde{\mathbf{e}}_{l,x,y}^L - \tilde{\mathbf{e}}_{l,4x-d,y}^R\|_1 \qquad (2)$$

The initial disparities are then computed as:

$$d_{l,x,y}^{\text{init}} = \text{argmin}_{d \in [0,D]} \varrho(l, x, y, d) \qquad (3)$$

for each (x, y) location and resolution l, where D is the maximal disparity that is considered. Note that despite the fact that the initialization stage exhaustively computes

matches for all disparities there is no need to ever store the whole cost volume. At test time only the location of the best match needs to be extracted, which can be done very efficiently utilizing fast memory, e.g. shared memory on GPUs and a fused implementation in a single Op. Hence, there is no need to store and process a 3D cost volume.

Tile Feature Descriptor. The initialization stage also predicts a feature description $\mathbf{p}_{l,x,y}^{\text{init}}$ for each (x,y) location and resolution l:

$$\mathbf{p}_{l,x,y}^{\text{init}} = \mathcal{D}(\varrho(d_{l,x,y}^{\text{init}}), \tilde{\mathbf{e}}_{l,x,y}^{L}; \boldsymbol{\theta}_{\mathcal{D}_l}). \tag{4}$$

The features are based on the embedding vector of the reference image $\tilde{\mathbf{e}}_{l,x,y}^{L}$ and the costs ϱ of the best matching disparity d_{init}. We utilize a perceptron $\mathcal{D}$, with learnable weights $\boldsymbol{\theta}_{\mathcal{D}}$, which is implemented with a 1×1 convolution followed by a leaky ReLU. The input to the tile feature descriptor includes the matching costs $\varrho(\cdot)$, which allows the network to get a sense of the confidence of the match.

3.4. Propagation

The propagation step takes tile hypotheses as input and outputs refined tile hypotheses based on spatial propagation of information and fusion of information. It internally warps the features from the feature extraction stage from the right image (secondary) to the left image (reference) in order to predict highly accurate offsets to the input tiles. An additional confidence is predicted which allows for effective fusion between hypotheses coming from earlier propagation layers and from the initialization stage.

Warping. The warping step computes the matching costs between the feature maps $\mathbf{e}_l^L$ and $\mathbf{e}_l^R$ at the feature resolution l associated to the tiles. This step is used to build a local cost volume around the current hypothesis. Each tile hypothesis is converted into a planar patch of size 4×4 that it originally covered in the feature map. We denote the corresponding 4×4 local disparity map as $\mathbf{d}'$ with

$$\mathbf{d}'_{i,j} = d + (i - 1.5)d_x + (j - 1.5)d_y, \tag{5}$$

for patch coordinates $i, j \in \{0, \cdots, 3\}$. The local disparities are then used to warp the features $\mathbf{e}_l^R$ from the right (secondary) image to the left (reference) image using linear interpolation along the scan lines. This results in a warped feature representation $\mathbf{e}_l^{R'}$ which should be very similar to the corresponding features of the left (reference) image $\mathbf{e}^L$ if the local disparity maps $\mathbf{d}'$ are accurate. Comparing the features of the reference (x,y) tile with the warped secondary tile we define the cost vector $\boldsymbol{\phi}(\mathbf{e}, \mathbf{d}') \in \mathbb{R}^{16}$ as:

$$\boldsymbol{\phi}(\mathbf{e}_l, \mathbf{d}') = [c_{0,0}, c_{0,1}, \ldots, c_{0,3}, c_{1,0} \ldots c_{3,3}], \tag{6}$$

where $c_{i,j} = \|\mathbf{e}_{l,4x+i,4y+j}^{L} - \mathbf{e}_{l,4x+i-\mathbf{d}'_{i,j},4y+j}^{R}\|_1$.

Tile Update Prediction. This step takes n tile hypotheses as input and predicts deltas for the tile hypotheses plus a scalar value w for each tile indicating how likely this tile is to be correct, i.e. a confidence measure. This mechanism is implemented as a CNN module $\mathcal{U}$, the convolutional architecture allows the network to see the tile hypotheses in a spatial neighborhood and hence is able to spatially propagate information. A key part of this step is that we augment the tile hypothesis with the matching costs $\boldsymbol{\phi}$ from the warping step. By doing this for a small neighborhood in disparity space we build up a local cost volume which allows the network to refine the tile hypotheses effectively. Concretely, we displace all the disparities in a tile by a constant offset of one disparity $\mathbf{1}$ in the positive and negative directions and compute the cost three times. Using this let $\mathbf{a}$ be the augmented tile hypothesis map for input tile map $\mathbf{h}$:

$$\mathbf{a}_{l,x,y} = [\mathbf{h}_{l,x,y}, \underbrace{\boldsymbol{\phi}(\mathbf{e}_l, \mathbf{d}' - \mathbf{1}), \boldsymbol{\phi}(\mathbf{e}_l, \mathbf{d}'), \boldsymbol{\phi}(\mathbf{e}_l, \mathbf{d}' + \mathbf{1})}_{\text{local cost volume}}], \tag{7}$$

for a location (x,y) and resolution l, The CNN module $\mathcal{U}_l$ then predicts updates for each of the n tile hypothesis maps and additionally $w^i \in \mathbb{R}$ which represent the confidence of the tile hypotheses:

$$(\underbrace{\Delta\mathbf{h}_l^1, w^1, \ldots, \Delta\mathbf{h}_l^n, w^n}_{\text{hypotheses updates}}) = \mathcal{U}_l(\mathbf{a}_l^1, \ldots, \mathbf{a}_l^n; \boldsymbol{\theta}_{\mathcal{U}_l}). \tag{8}$$

The architecture of $\mathcal{U}$ is implemented with residual blocks [20] but without batch normalization. Following [25] we use dilated convolutions to increase the receptive field. Before running a sequence of residual blocks with varying dilation factors we run a 1×1 convolution followed by a leaky ReLU to decrease the number of feature channels. The update module is applied in a hierarchical iterative fashion (see Fig. 2). At the lowest resolution $l = M$ we only have 1 tile hypothesis per location from the initialization stage, hence $n = 1$. We apply the tile updates by summing the input tile hypotheses and the deltas and upsample the tiles by a factor of 2 in each direction. Thereby, the disparity d is upsampled using the plane equation of the tile and the remaining parts of the tile hypothesis d_x, d_y and $\mathbf{p}$ are upsampled using nearest neighbor sampling. At the next resolution $M-1$ we now have two hypotheses: the one from the initialization stage and the upsampled hypotheses from the lower resolution, hence $n = 2$. We utilize the w^i to select the updated tile hypothesis with highest confidence for each location. We iterate this procedure until we reach the resolution 0, which corresponds to tile size 4×4 and full disparity resolution in all our experiments. To further refine the disparity map we use the winning hypothesis for the 4×4 tiles and apply propagation module 3 times: for 4×4, 2×2, 1×1 resolutions, using $n = 1$. The output at tile size 1×1 is our final prediction. More details about the network architecture are provided in the supplementary material.

4. Loss Functions

Our network is trained end-to-end with ground truth disparities d^{gt} utilizing the losses described in the remainder of this section. The final loss is a sum of the losses over all the scales and pixels: $\sum_{l,x,y} L_l^{\mathrm{init}} + L_l^{\mathrm{prop}} + L_l^{\mathrm{slant}} + L_l^{\mathrm{w}}$.

4.1. Initialization Loss

Ground truth disparities are given with subpixel precision, however matching in initialization happens with integer disparities. Therefore we compute the matching cost for subpixel disparities using linear interpolation. The cost for subpixel disparities is then given as

$$\psi(d) = (d - \lfloor d \rfloor)\varrho(\lfloor d \rfloor + 1) + (\lfloor d \rfloor + 1 - d)\varrho(\lfloor d \rfloor), \quad (9)$$

where we dropped the l, x, y subscripts for clarity. To compute them at multiple resolutions we maxpool the ground truth disparity maps to downsample them to the required resolution. We aim at training the features $\mathcal{E}$ to be such that the matching cost ψ is smallest at the ground truth disparity and larger everywhere else. To achieve this, we impose an ℓ_1 contrastive loss [18]

$$L^{\mathrm{init}}(d^{\mathrm{gt}}, d^{\mathrm{nm}}) = \psi(d^{\mathrm{gt}}) + \max(\beta - \psi(d^{\mathrm{nm}}), 0), \quad (10)$$

where $\beta > 0$ is a margin, d^{gt} the ground truth disparity for a specific location and

$$d^{\mathrm{nm}} = \mathrm{argmin}_{d \in [0,D]/\{d: d \in [d^{\mathrm{gt}} - 1.5, d^{\mathrm{gt}} + 1.5]\}} \varrho(d) \quad (11)$$

the disparity of the lowest cost non match for the same location. This cost pushes the ground truth cost toward 0 as well as the lowest cost non match toward a certain margin. In all our experiments we set the margin to $\beta = 1$. Similar contrastive losses have been used to learn the matching score in earlier deep learning based approaches to stereo matching [58, 35]. However, they either used a random non-matching location as negative sample or used all the non matching locations as negative samples, respectively.

4.2. Propagation Loss

During propagation we impose a loss on the tile geometry d, d_x, d_y and the tile confidence w. We use the ground truth disparity d^{gt} and ground truth disparity gradients d_x^{gt} and d_y^{gt}, which we compute by robustly fitting a plane to d^{gt} in a 9×9 window centered at the pixel. In order to apply the loss on the tile geometry we first expand the tiles to a full resolution disparities $\hat{d}$ using the plane equation (d, d_x, d_y) analogously to Eq. 5. The slant portion is also up-sampled to full resolution using nearest neighbor approach before slant loss is applied. We use the general robust loss function $\rho(\cdot)$ from [2] which resembles a smooth ℓ_1 loss, i.e., Huber loss. Additionally, we apply a truncation to the loss with threshold A

$$L^{\mathrm{prop}}(d, dx, dy) = \rho(\min(|d^{\mathrm{diff}}|, A), \alpha, c), \quad (12)$$

where $d^{\mathrm{diff}} = d^{\mathrm{gt}} - \hat{d}$. Further we impose a loss on the surface slant, as

$$L^{\mathrm{slant}}(d_x, d_y) = \left\| \begin{matrix} d_x^{\mathrm{gt}} - d_x \\ d_y^{\mathrm{gt}} - d_y \end{matrix} \right\|_1 \chi_{|d^{\mathrm{diff}}| < B}, \quad (13)$$

where χ is an indicator function which evaluates to 1 when the condition is satisfied and 0 otherwise. To supervise the confidence w we impose a loss which increases the confidence if the predicted hypothesis is closer than a threshold C_1 from the ground truth and decrease the confidence if the predicted hypothesis is further than a threshold C_2 away from the ground truth.

$$L^{\mathrm{w}}(w) = \max(1 - w, 0)\chi_{|d^{\mathrm{diff}}| < C_1} + \max(w, 0)\chi_{|d^{\mathrm{diff}}| > C_2} \quad (14)$$

For all our experiments $A = B = C_1 = 1$; $C_2 = 1.5$. For the last several levels, when only a single hypotheses is available, loss is applied to all pixels ($A = \infty$).

5. Experiments

We evaluate the proposed approach on popular benchmarks showing competitive results at a fraction of the computational time compared to other methods. We consider the following datasets: SceneFlow [37], KITTI 2012 [15], KITTI 2015 [38], ETH3D [44], Middlebury dataset V3 [42]. Following the standard evaluation settings we consider the two popular metrics: the End-Point-Error (EPE), which is the absolute distance in disparity space between the predicted output and the groundtruth; the x-pixels error, which is the percentage of pixels with disparity error greater than x. For the EPE computation on SceneFlow we adopt the same methodology of PSMNet [7], which excludes all the pixel with ground truth disparity bigger than 192 from the evaluation. Unless stated otherwise we use a HITNet with 5 levels, i.e. $M = 4$.

In this section we focus on comparisons with state-of-art on popular benchmarks, detailed ablation studies, run-time breakdown, cross-domain generalization and additional evaluations, are provided in the supplementary material. The trained models used for submission to benchmarks and evaluation scripts can be found at https://github.com/google-research/google-research/tree/master/hitnet

5.1. Comparisons with State-of-the-art

SceneFlow. On the synthetic dataset SceneFlow "final-pass" we achieve the remarkable End-Point-Error (EPE) of 0.36, which is 2X better than state-of-art at time of writing (see supplementary materials for details of L and XL versions). Representative competitors are reported in Tab. 1. The PSMNet algorithm [7] performs multi-scale feature extraction similarly to our method, but in contrast they use a more sophisticated pooling layer. Here we show that
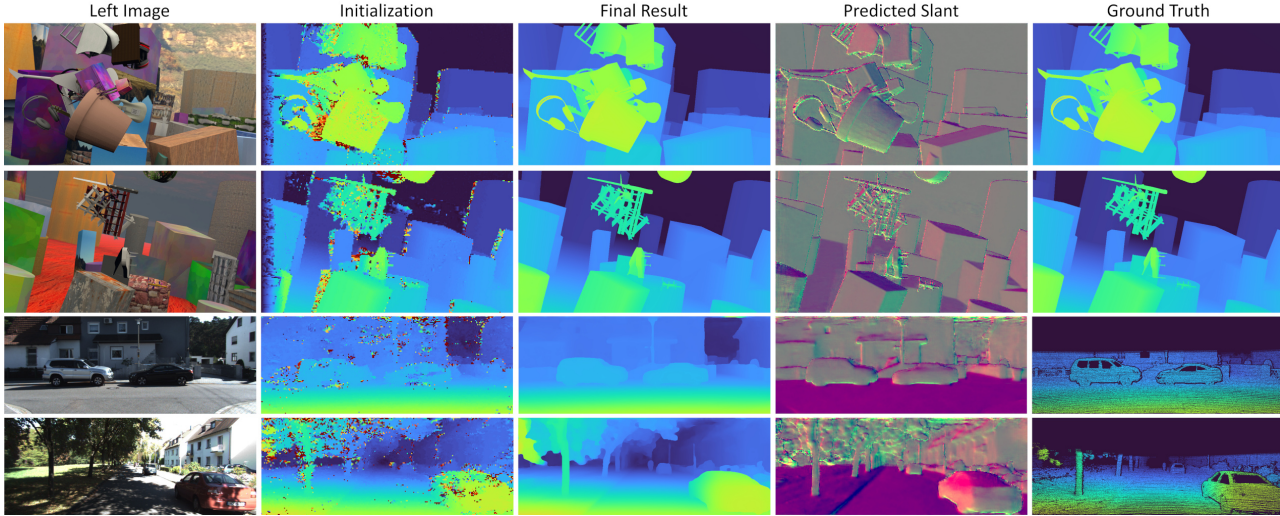
| Left Image | Initialization | Final Result | Predicted Slant | Ground Truth |

Figure 3: Qualitative results on SceneFlow and KITTI 2012. Note how the model is able to recover fine details, textureless regions and crisp edges.

Method	EPE px	Runtime s
HITNet XL	0.36	0.114
HITNet L	0.43	0.054
EdgeStereo [48]	0.74	0.32
LEAStereo [8]	0.78	0.3
GA-Net [59]	0.84	1.6
PSMNet [7]	1.09	0.41
StereoNet [25]	1.1	0.015

Table 1: Comparisons with state-of-the-art methods on Scene Flow "finalpass" dataset, lower is better.

Method	EPE px	Bad 1	Bad 2	Runtime
HITNet (ours)	0.20	2.79	0.80	0.02 s
R-Stereo	0.18	2.44	0.44	0.81 s
DN-CSS	0.22	2.69	0.77	0.31 s
AdaStereo [47]	0.26	3.41	0.74	0.40 s
Deep-Pruner [9]	0.26	3.52	0.86	0.16 s
iResNet [32]	0.24	3.68	1.00	0.20 s
Stereo-DRNet [6]	0.27	4.46	0.83	0.33 s
PSMNet [7]	0.33	5.02	1.09	0.54 s

Table 2: Comparisons with state-of-the-art methods on ETH3D stereo dataset. For all metrics lower is better.

our architecture is more effective. Compared to GA-Net [59], we do not need complex message passing steps such as SGM. The results we obtain show that our strategy is also achieving a very similar inference. Finally, a representative fast method, StereoNet [25] is considered, which we consistently outperform. As result our method achieves the lowest EPE while still maintaining real-time performance. See Figure 3 for qualitative results.

Middlebury Stereo Dataset v3. We evaluated our method with multiple state-of-art approaches on the Middlebury stereo dataset v3, see Table 4 and the official benchmark website.[2]. As we can observe *we outperform all the other end-to-end learning based approaches on most of the metrics*, we rank among the top 10 when considering also hand-crafted approaches and in particular we rank *first* for bad 0.5 and A50, *second* for bad 1 and *avgerr*. In addition, we note that our average error is impacted by specifically one image, *DjembL*, which is due to the fact that we do not explicitly handle harsh lighting variations between input pairs. For visual results on the Middlebury datasets and details regarding the training procedure we refer the reader to the supplementary material.

ETH3D two view stereo. We evaluated our method with multiple state-of-art approaches on the ETH3D dataset, see Tab. 2. At time of submission to benchmark, HITNet ranks 1st-4rd on all the metrics published on the website. In particular, our method ranks 1nd on the following metrics: bad 0.5, bad 4, average error, rms error, 50% quantile, 90% quantile: this shows that HITNet is resilient to the particular measurement chosen, whereas competitive approaches exhibits substantial differences when different metrics are selected. See the submission website for details.[3].

KITTI 2012 and 2015. At time of writing, among the published methods faster than 100ms, HITNet ranks #1 on KITTI 2012 and 2015 benchmarks. Compared to other state-of-the-art stereo matchers (see Tab. 3), our approach

---

[2]See "HITNet" entry on the official dataset website.
[3]See the ETH3D Website at https://www.eth3d.net/low_res_two_view for the complete metrics.

Method	KITTI 2012 [15]						KITTI 2015 [38]			
	2-noc	2-all	3-noc	3-all	EPE noc	EPE all	D1-bg	D1-fg	D1-all	Run-time
HITNet (ours)	2.00	2.65	1.41	1.89	0.4	0.5	1.74	3.20	1.98	0.02s
LEAStereo [8]	1.90	2.39	1.13	1.45	0.4	0.5	1.40	2.91	1.65	0.3s
GANet-deep [59]	1.89	2.50	1.19	1.6	0.4	0.5	1.48	3.46	1.81	1.8s
EdgeStereo-V2 [48]	2.32	2.88	1.46	1.83	0.4	0.5	1.84	3.30	2.08	0.32s
GC-Net [24]	2.71	3.46	1.77	2.30	0.6	0.7	2.21	6.16	2.87	0.9s
SGM-Net [45]	3.60	5.15	2.29	3.50	0.7	0.9	2.66	8.64	3.66	67s
ESMNet [17]	3.65	4.30	2.08	2.53	0.6	0.7	2.57	4.86	2.95	0.06s
MC-CNN-acrt [58]	3.90	5.45	2.09	3.22	0.6	0.7	2.89	8.88	3.89	67s
RTSNet [29]	3.98	4.61	2.43	2.90	0.7	0.7	2.86	6.19	3.41	0.02s
Fast DS-CS [54]	4.54	5.34	2.61	3.20	0.7	0.8	2.83	4.31	3.08	0.02s
StereoNet [25]	4.91	6.02	-	-	0.8	0.9	4.30	7.45	4.83	0.015s

Table 3: Quantitative evaluation on KITTI 2012 and KITTI 2015. For KITTI 2012 we report the percentage of pixels with error bigger than x disparities in both non-occluded (x-noc) and all regions (x-all), as well as the overall EPE in both non occluded (EPE-noc) and all the pixels (EPE-all). For KITTI 2015 We report the percentage of pixels with error bigger than 1 disparity in background regions (bg), foreground areas (fg), and all.

Method	RMS	AvgErr	Bad 0.5	Bad 1.0	Bad 2.0	Bad 4.0	A50	Run-time
HITNet (ours)	9.97	1.71	34.2	13.3	6.46	3.81	0.40	0.14 s
LEAStereo [8]	8.11	1.43	49.5	20.8	7.15	2.75	0.53	2.9 s
NOSS-ROB [30]	12.2	2.08	38.2	13.2	5.01	3.46	0.42	662s (CPU)
LocalExp [50]	13.4	2.24	38.7	13.9	5.43	3.69	0.43	881s (CPU)
CRLE [52]	13.6	2.25	38.1	13.4	5.75	3.90	0.42	1589s (CPU)
HSM [53]	10.3	2.07	50.7	24.6	10.2	4.83	0.56	0.51 s
MC-CNN [58]	21.3	3.82	40.7	17.1	8.08	4.91	0.45	150 s
EdgeStereo [48]	9.84	2.67	55.6	32.4	18.7	10.8	0.72	0.35 s

Table 4: Comparisons with state-of-the-art methods on Middlebury V3 dataset. For all metrics lower is better.

compares favorably to GC-Net [24], [39] and many others. Recent methods such as GA-Net [59] and HSM [53] are obtaining slightly better metrics, although they require 1.8 and 0.15 seconds respectively. Note also that HSM [53] has been trained with additional external high resolution data. Similarly, GA-Net [59] is pre-trained on SceneFlow and fine-tuned on KITTI benchmarks, whereas our approach is fully trained on the small data available on KITTI. Compared to fast methods such as StereoNet [25] and RTSNet [29], our method consistently outperforms them by a considerable margin, showing that it can be employed in latency critical scenarios without sacrificing accuracy.[4]

6. Conclusion

We presented HITNet, a real-time end-to-end architecture for accurate stereo matching. We presented a fast initialization step that is able to compute high resolution matches using learned features very efficiently. These tile initializations are then fused using propagation and fusion steps. The use of slanted support windows with learned descriptors provides additional accuracy. We presented state-of-the art accuracy on multiple commonly used benchmarks. A limitation of our algorithm is that it needs to be trained on a dataset with ground truth depth. To address this in the future we are planning to investigate self-supervised methods and self-distillation methods to further increase the accuracy and decrease the amount of training data that is required. A limitation of our experiments is that different datasets are trained on separately and use slightly different model architectures. To address this in the future, a single experiment is required that aligns with Robust Vision Challenge requirements.

References

[1] Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (ToG)*, 2009. 2

---

[4]See the KITTI Website at http://www.cvlibs.net/datasets/kitti/eval_stereo.php for the complete metrics.

[2] Jonathan T Barron. A general and adaptive robust loss function. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 6

[3] Frederic Besse, Carsten Rother, Andrew Fitzgibbon, and Jan Kautz. Pmbp: Patchmatch belief propagation for correspondence field estimation. *International Journal of Computer Vision (IJCV)*, 2014. 2

[4] Michael Bleyer and Margrit Gelautz. Simple but effective tree structures for dynamic programming-based stereo matching. In *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2008. 2

[5] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo-stereo matching with slanted support windows. In *British Machine Vision Conference (BMVC)*, 2011. 2

[6] Rohan Chabra, Julian Straub, Christopher Sweeney, Richard Newcombe, and Henry Fuchs. StereoDRNet: Dilated residual stereonet. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 7

[7] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3, 6, 7

[8] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Tom Drummond, Hongdong Li, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. In *NIPS*, 2020. 7, 8

[9] Shivam Duggal, Shenlong Wang, Wei-Chiu Ma, Rui Hu, and Raquel Urtasun. Deeppruner: Learning efficient stereo matching via differentiable patchmatch. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4383–4392, 2019. 7

[10] S. R. Fanello, C. Keskin, S. Izadi, P. Kohli, D. Kim, D. Sweeney, A. Criminisi, J. Shotton, S.B. Kang, and T. Paek. Learning to be a depth camera for close-range human capture and interaction. *Transaction On Graphics (TOG)*, 2014. 2

[11] S. R. Fanello, C. Rhemann, V. Tankovich, A. Kowdle, S. Orts Escolano, D. Kim, and S. Izadi. Hyperdepth: Learning depth from structured light without matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[12] Sean Ryan Fanello, Julien Valentin, Adarsh Kowdle, Christoph Rhemann, Vladimir Tankovich, Carlo Ciliberto, Philip Davidson, and Shahram Izadi. Low compute and fully parallel computer vision with hashmatch. *Inernational Conference on Computer Vision (ICCV)*, 2017. 2, 3

[13] Sean Ryan Fanello, Julien Valentin, Christoph Rhemann, Adarsh Kowdle, Vladimir Tankovich, Philip Davidson, and Shahram Izadi. Ultrastereo: Efficient learning-based matching for active stereo systems. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2, 3

[14] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient belief propagation for early vision. *International journal of computer vision (IJCV)*, 2006. 2

[15] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012. 6, 8

[16] Spyros Gidaris and Nikos Komodakis. Detect, replace, refine: Deep structured prediction for pixel wise labeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[17] Chenggang Guo, Dongyi Chen, and Zhiqi Huang. Learning efficient stereo matching network with depth discontinuity aware super-resolution. *IEEE Access*, 2019. 8

[18] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006. 6

[19] Rostam Affendi Hamzah and Haidi Ibrahim. Literature survey on stereo vision disparity map algorithms. *Journal of Sensors*, 2016. 2

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2016. 5

[21] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence (TPAMI)*, 2008. 2

[22] Asmaa Hosni, Christoph Rhemann, Michael Bleyer, Carsten Rother, and Margrit Gelautz. Fast cost-volume filtering for visual correspondence and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2013. 2

[23] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[24] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *IEEE International Conference on Computer Vision (ICCV)*, 2017. 2, 3, 8

[25] Sameh Khamis, Sean Fanello, Christoph Rhemann, Adarsh Kowdle, Julien Valentin, and Shahram Izadi. StereoNet: Guided hierarchical refinement for edge-aware depth prediction. In *European Conference on Computer Vision (ECCV)*, 2018. 2, 3, 5, 7, 8

[26] Andreas Klaus, Mario Sormann, and Konrad Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *International Conference on Pattern Recognition (ICPR)*, 2006. 2

[27] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *International Conference on Computer Vision (ICCV)*, 2001. 2

[28] Adarsh Kowdle, Christoph Rhemann, Sean Fanello, Andrea Tagliasacchi, Jon Taylor, Philip Davidson, Mingsong Dou, Kaiwen Guo, Cem Keskin, Sameh Khamis, David Kim, Danhang Tang, Vladimir Tankovich, Julien Valentin, and Shahram Izadi. The need 4 speed in real-time dense visual tracking. *Transaction On Graphics (TOG)*, 2018. 2

[29] H. Lee and Y. Shin. Real-time stereo matching network with high accuracy. In *IEEE International Conference on Image Processing (ICIP)*, 2019. 8

[30] Jie Li, Penglei Ji, and Xinguo Liu. Superpixel alpha-expansion and normal adjustment for stereo matching. In *CAD/Graphics 2019*, 2019. 8

[31] Yu Li, Dongbo Min, Michael S Brown, Minh N Do, and Jiangbo Lu. SPM-BP: Sped-up patchmatch belief propagation for continuous mrfs. In *IEEE International Conference on Computer Vision (ICCV)*, 2015. 2

[32] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Linbo Qiao, Wei Chen, Li Zhou, and Jianfeng Zhang. Learning for disparity estimation through feature constancy. 2017. 3, 7

[33] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2015. 3

[34] Jiangbo Lu, Hongsheng Yang, Dongbo Min, and Minh N Do. Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2013. 2

[35] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 6

[36] David Marr and Tomaso Poggio. Cooperative computation of stereo disparity. *Science*, 1976. 2

[37] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 6

[38] Moritz Menze, Christian Heipke, and Andreas Geiger. Joint 3d estimation of vehicles and scene flow. In *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015. 6, 8

[39] Jiahao Pang, Wenxiu Sun, JS Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *International Conference on Computer Vision-Workshop on Geometry Meets Deep Learning (ICCVW 2017)*, 2017. 3, 8

[40] Anurag Ranjan and Michael J Black. Optical flow estimation using a spatial pyramid network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[41] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention (MICCAI)*, 2015. 3

[42] Daniel Scharstein, Heiko Hirschmuller, York Kitajima, Greg Krathwohl, Nera Nesic, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition (GCPR)*, 2014. 6

[43] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 2002. 2, 3

[44] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6

[45] A. Seki and M. Pollefeys. SGM-Nets: Semi-global matching with neural networks. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 8

[46] Amit Shaked and Lior Wolf. Improved stereo matching with constant highway networks and reflective confidence learning. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2017. 2

[47] Xiao Song, Guorun Yang, Xinge Zhu, Hui Zhou, Zhe Wang, and Jianping Shi. Adastereo: A simple and efficient approach for adaptive stereo matching. Technical report, arXiv preprint arXiv:2004.04627, 2020. 2, 7

[48] Xiao Song, Xu Zhao, Liangji Fang, Hanwen Hu, and Yizhou Yu. Edgestereo: An effective multi-task learning network for stereo matching and edge detection. *International Journal of Computer Vision (IJCV)*, 2020. 7, 8

[49] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: Cnns for optical flow using pyramid, warping, and cost volume. *IEEE Conference on Computer Vision and Pattern Recognition*, 2018. 3

[50] Tatsunori Taniai, Yasuyuki Matsushita, Yoichi Sato, and Takeshi Naemura. Continuous 3D Label Stereo Matching using Local Expansion Moves. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(11):2725–2739, 2018. 8

[51] Vladimir Tankovich, Michael Schoenberg, Sean Ryan Fanello, Adarsh Kowdle, Christoph Rhemann, Max Dzitsiuk, Mirko Schmidt, Julien Valentin, and Shahram Izadi. Sos: Stereo matching in o(1) with slanted support windows. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018. 2, 3

[52] H. Xu, X. Chen, H. Liang, S. Ren, Y. Wang, and H. Cai. Crosspatch-based rolling label expansion for dense stereo matching. *IEEE Access*, 8:63470–63481, 2020. 8

[53] Gengshan Yang, Joshua Manela, Michael Happold, and Deva Ramanan. Hierarchical deep stereo matching on high-resolution images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 8

[54] Kyle Yee and Ayan Chakrabarti. Fast deep stereo with 2d convolutional processing of cost signatures. In *Winter Conference on Applications of Computer Vision (WACV)*, 2020. 8

[55] Kuk-Jin Yoon and In-So Kweon. Locally adaptive support-weight approach for visual correspondence search. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. 2

[56] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2

[57] Jure Zbontar and Yann LeCun. Computing the stereo matching cost with a convolutional neural network. In *IEEE conference on computer vision and pattern recognition (CVPR)*, 2015. 2

[58] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research (JMLR)*, 2016. 2, 6, 8

[59] Feihu Zhang, Victor Prisacariu, Ruigang Yang, and Philip HS Torr. Ga-net: Guided aggregation net for end-to-end stereo matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 7, 8

[60] Yinda Zhang, Sameh Khamis, Christoph Rhemann, Julien Valentin, Adarsh Kowdle, Vladimir Tankovich, Michael Schoenberg, Shahram Izadi, Thomas Funkhouser, and Sean Fanello. ActiveStereoNet: End-to-end self-supervised learning for active stereo systems. *European Conference on Computer Vision (ECCV)*, 2018. 3