Hierarchical Deep Stereo Matching on High-resolution Images

Gengshan Yang[1,*], Joshua Manela[2], Michael Happold[2], Deva Ramanan[1,2]
[1]Carnegie Mellon University, [2]Argo AI
gengshany@cmu.edu {jmanela, mhappold, dramanan}@argo.ai

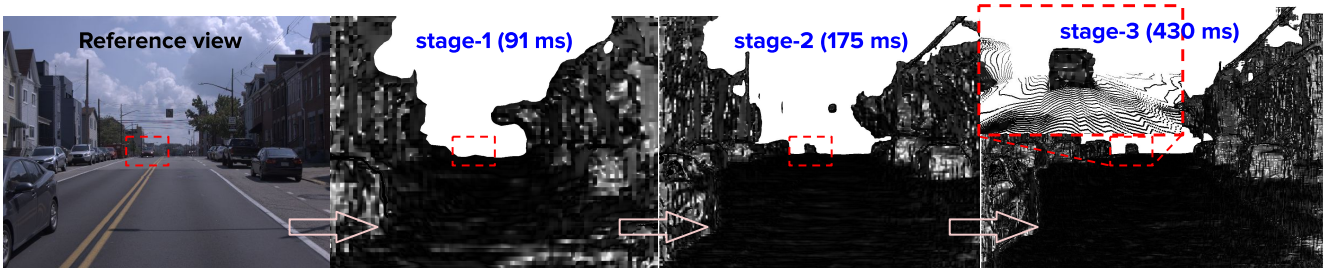

Figure 1: Illustration of on-demand depth sensing with a coarse-to-fine hierarchy on the proposed dataset. Our method (HSM) captures the coarse layout of the scene in 91 milliseconds, finds the far-away car (shown in the red box) in 175 ms, and recovers the details of the car given extra 255 ms. HSM方法估计三个尺度时间91、175和255ms

Abstract

We explore the problem of real-time stereo matching on high-res imagery. Many state-of-the-art (SOTA) methods struggle to process high-res imagery because of memory constraints or speed limitations. To address this issue, we propose an end-to-end framework that searches for correspondences incrementally over a coarse-to-fine hierarchy. Because high-res stereo datasets are relatively rare, we introduce a dataset with high-res stereo pairs for both training and evaluation. Our approach achieved SOTA performance on Middlebury-v3 and KITTI-15 while running significantly faster than its competitors. The hierarchical design also naturally allows for anytime on-demand reports of disparity by capping intermediate coarse results, allowing us to accurately predict disparity for near-range structures with low latency (30ms). We demonstrate that the performance-vs-speed tradeoff afforded by on-demand hierarchies may address sensing needs for time-critical applications such as autonomous driving.

1. Introduction

In safety-critical applications such as autonomous driving, it is important to accurately perceive the depth of obstacles with low latency. Towards that end, we explore the problem of real-time stereo matching on high-resolution imagery.

LiDAR vs Stereo: LiDAR is the common choice for outdoor depth sensing [32]. However, LiDARs are fundamentally limited in spatial density, particularly for long-range sensing. Only so many beams and detectors can be packed together before cross-talk occurs. In principle, one can increase density by scanning slowly, but this introduces latency and rolling shutter effects that can be devastating for dynamic scenes. Density of sensor measurements is crucial for recognizing and segmenting objects at range. High-res, global shutter stereo has the potential to address these limitations.

Why high-res? It is widely recognized that stereo is unreliable for long-range depth sensing [24]: the metric error in estimated depth ΔZ of a triangulation-based stereo system with baseline b, focal length f, and pixel matching error Δd can be written as $\Delta Z = Z^2 \frac{\Delta d}{b \cdot f}$. Hence depth error increases quadratically with depth, implying that stereo will provide unstable far-field depth estimates. But this is important for navigation at even moderate speeds - (see the supplement for stopping distance profiles). Though one can attempt to tweak the other factors to reduce the error, higher-resolution (larger f) appear to be the most promising avenue: innovations in CMOS and CCD sensor technology have allowed for mass-market, low-cost solutions for high-resolution cameras.

*Work done during an internship at Argo AI. Data and code will be available here.

Challenges: Although high-res stereo matching is desirable, there are several practical challenges: optimization-based stereo methods are accurate but cannot scale to high-resolution with regard to both running time and memory overhead. When applied to down-scaled images, these methods run faster, but gives blurry results and inaccurate disparity estimates for the far-field. Recent "deep" stereo methods perform well on low-resolution benchmarks [5, 11, 16, 21, 38], while failing to produce SOTA results on high-res benchmarks [26]. This is likely due to: 1) Their architectures are not efficiently designed to operate on high-resolution images. 2) They do not have enough high-resolution training data.

Approach: We propose an end-to-end framework that efficiently searches for correspondences through hierarchies. Our model reasons in a coarse-to-fine-manner, inspired by classic work on correspondence estimation in stereo and optical flow [1, 18, 39]. Coarse resolution images are used to estimate large disparities, which are then used to bias/pre-warp fine-scale disparity estimates. Though quite efficient, coarse-to-fine methods struggle to match thin structures that "disappear" at coarse resolutions [3]. Instead, our model computes a *high-res* encoder feature that is processed with coarse-to-fine (decoder) feature volumes that gradually increase in resolution. Crucially, the initial coarse volume can generate rough estimates of large-disparity objects *before* the full pipeline is finished. This allows our network to generate reports of closeby objects on-demand, which can be crucial for real-time navigation at speed.

Data: High-res stereo efforts suffer from the lack of benchmark data, for both training and evaluation. We have collected two datasets of high-res rectified stereo pairs, consisting of real-data from an autonomous vehicle and synthetic data from an urban simulator. Interestingly, we show that synthetic data is a valuable tool for training deep stereo networks, particularly for high-res disparity estimation. Real-world calibration and rectification become challenging at high-res, and introducing realistic calibration errors through data-augmentation is important during training.

Our main contributions are as follows:

1. We propose a hierarchical stereo matching architecture that scales to high-resolution images while being able to perform on-demand computation in real-time.

2. We collect two high-resolution stereo datasets for training and testing.

3. We introduce a set of stereo augmentation techniques to improve model's robustness to calibration errors, exposure changes, and camera occlusions.

4. We achieve SOTA accuracy on Middlebury and KITTI while running significantly faster than the prior art.

2. Related Work

Stereo matching is a classic task in computer vision [27]. Traditional methods take rectified image pairs as input (though extensions to multiview stereo exist [32]), extract local descriptors at candidate patches [35, 36] and then build up 3-dimensional cost volumes across corresponding epipolar scanlines [22]. To ensure global ordering and consistency constraints, global optimization techniques are applied with a considerable cost of time and memory, which also poses limitations on scale [10, 14].

Efficient high-resolution stereo: To mitigate this problem, SGM [8] and ELAS [7] describe efficient matching algorithms that allow for 3FPS on 1.5-megapixel images with a strong performance. However, both methods struggle to scale to 6-megapixel images: for example, SGM requires 16 GB to process a disparity search range of 700px, and surprisingly, performance *drops* when compared to a variant that processes lower-res inputs [15]. While other work has also explored efficient high-res processing [15, 30], they do not appear to meet the accuracy and speed requirements for real-time sensing in autonomous driving.

Deep stereo matching: Deep networks tuned for stereo estimation can take advantage of large-scale annotated data. They now produce SOTA performance on several stereo benchmarks, though with considerable use of memory and time. Zbontar et al. and Luo et al. [19, 40] use siamese networks to extract patch-wise features, which are then processed in a traditional cost volume by classic post-processing. Some recent methods [5, 12, 13, 16, 20, 23] replace post-processing with 2D/3D convolutions applied to the cost volume, producing SOTA performance on the KITTI benchmark. However surprisingly, none of them outperform traditional methods on Middlebury, possibly due to 1) memory constraints and 2) lack of training data. Although one may run low-res models on crops of high-res images and stitch together the predictions, however, challenges are that: 1) crop borders can be challenging to match; 2) contextual information may not be well-utilized; and 3) most importantly, this dramatically increases run-time latency.

To our knowledge, we are the first to successfully address these issues and apply deep networks on high-res stereo matching: We propose an efficient hierarchical stereo matching architecture to address the efficiency issue, and leverage high-res synthetic data as well as novel augmentation techniques to overcome data scarcity.

Coarse-to-fine CNNs: Coarse-to-fine design in CNNs dates back to FCN and U-Net [17, 25], which leverages multi-scale features and aggregate coarse-to-fine predic-

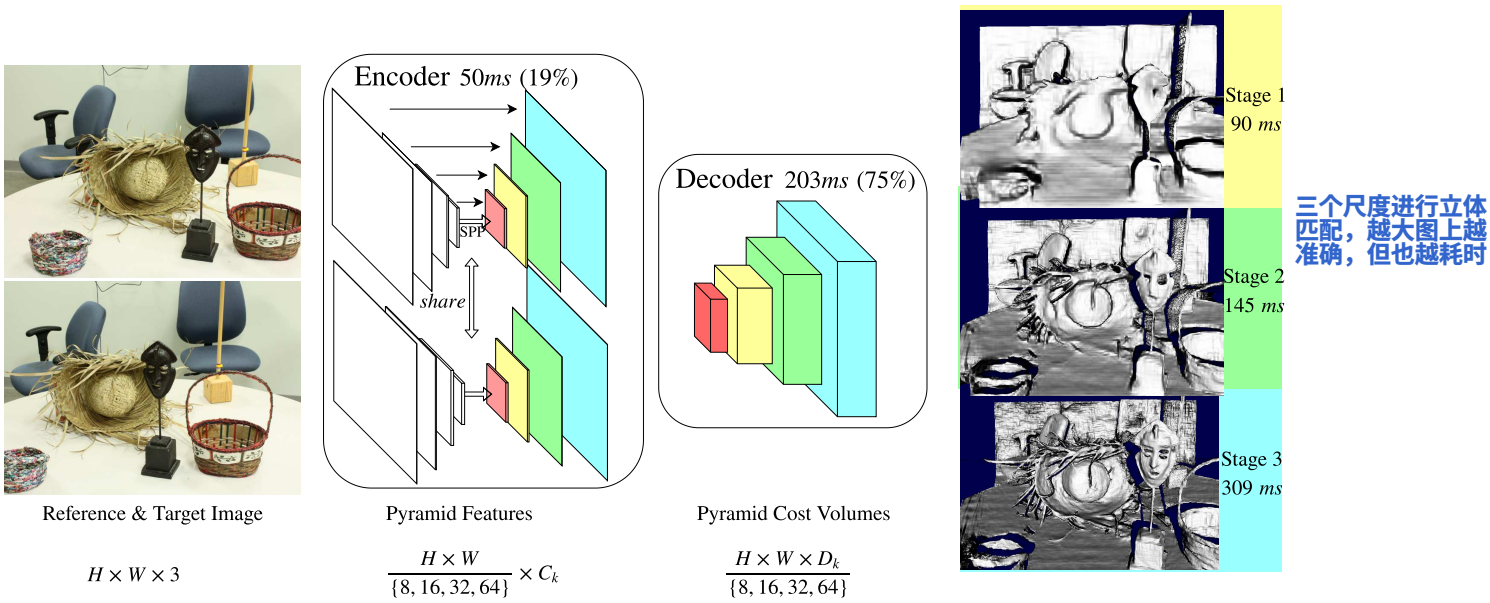

Figure 2: Our on-demand, low-memory architecture for high-res stereo. Given a rectified pair of high-res images, we compute multiscale descriptors for each with a custom resnet "butterfly" encoder-decoder network (which we refer to as our pyramid *encoder*). These descriptors are used to construct 4D feature volumes at each scale ($\frac{C_k \times H \times W \times D}{\{8,16,32,64\}}$, where scale $k \in \{1, 2, 3, 4\}$ and $C_k \in \{16, 16, 16, 32\}$), by taking the difference of potentially matching features extracted from epipolar scanlines. Each feature volume is *decoded* or filtered with 3D convolutions, making use of striding along the disparity dimensions to minimize memory. The decoded output is (a) used to predict 3D cost volumes that generates on-demand disparity estimates for the given scale and (b) upsampled so that it can be combined with the next feature volume in the pyramid. and $D_k \in \{\frac{D_{max}}{4}, \frac{D_{max}}{2}, D_{max}, D_{max}\}$ represent the number of feature channel and the number of disparity bins in the kth scale, and the time is measured for 6-megapixel input with a disparity search range of 256px on Titan X Pascal.

tions to improve semantic segmentation. DispNet-based architectures [20, 23] adopts an encoder-decoder scheme with skip-connections, which compute multi-scale cost volumes by correlation and apply 2D convolution/up-convolutions to refine predictions from coarse-to-fine. Recently, PWC-Net [33] uses a coarse-to-fine architecture to warp features and achieves SOTA results in optical flow estimation. Closest to our approach, GCNet [12] constructs hierarchical 4D feature volumes and processes them from coarse to fine using 3D convolutions, but we differ in our successful application of coarse-to-fine principles to high-resolution inputs and anytime, on-demand processing.

3. Method

In this section, we describe the key ingredients of our approach: 1) an efficient hierarchical stereo matching architecture, 2) a set of novel asymmetric augmentation techniques, and 3) a high-resolution synthetic dataset for training. We also introduce a high-resolution stereo benchmark for real-world autonomous driving.

3.1. Hierarchical Stereo Matching (HSM) network

Our core idea of designing the hierarchical coarse-to-fine network is to first aggressively downsample high res images through the network while extracting multi-scale features, and then use potential correspondences to gradually build up a pyramid of cost volumes that increase in resolution. We provide precise layer and filter dimensions in the supplement.

Design principles: We find that coarse-to-fine design principles are crucial, specifically making use of 1) spatial pyramid pooling (SPP) [41], which allows features to dramatically increase in receptive field. Without this, features tend to have too small a receptive field compared to the rest of the high-res image. The original implementation in SPP [41] upsampled pyramid features back to the original resolution. To reduce memory, we keep pooled features in their native coarse resolution; 2) 3D convolutions that are strided in the disparity dimension, allowing us to process high-res cost volumes efficiently; 3) multi-scale loss functions. The network architecture is shown in Figure 2.

Feature pyramid encoder: We use a feature pyramid encoder to extract descriptors for coarse-to-fine matching.

To efficiently extract features with different levels of details while maintaining the coarse-scale information, we adopt an encoder-decoder architecture with skip connections. Our feature encoder consists of custom resnet backbone with 4 residual blocks, followed by 4 SPP layers (again, to increase receptive fields with limited computation and memory).

Feature volumes: We obtain such features for both a left and right image, and then construct a 4D **feature volume** [5, 12, 13] by considering differences between pairs of potentially-matching descriptors along horizontal scanlines. We construct a pyramid of 4 volumes, each with increasing spatial resolution and increasing disparity resolution. While cost volumes are traditionally 3D (height H by width W by disparity D), our feature volume includes a 4th dimension representing the number of feature channels C, which increases for later layers in the encoder.

Feature volume decoder: Figure 3 visualizes the *decoding*, or filtering, of each feature volume. Let us first define a conv3D "block" as two 3D convolutions with a residual connection. 1) The feature volume is filtered by 6 conv3D blocks. 2) As was the case for feature extraction, we then apply Volumetric Pyramid Pooling (our extension of SPP to feature volumes) to generate features that capture sufficient global context for high-res inputs. 3a) The output is trilinearly-upsampled to a higher spatial (and disparity) resolution so that it can be fused with the next 4D feature volume in the pyramid. 3b) To report on-demand disparities computed from the current scale, the output is processed with another conv3D block to generate a 3D output cost volume. This cost volume can directly report disparities **before** subsequent feature volumes downstream in the pyramid are ever computed.

Multi-scale loss: We train the network to make predictions at different scales in the training phase, which allows for *on-demand* disparity output at any pyramid level, and also serves to regularize the overall network:

$$L = L_1 + \frac{1}{2^2}L_2 + \frac{1}{2^4}L_3 + \frac{1}{2^6}L_4$$

where losses are scaled to account for the increased disparity resolution at each pyramid level. L_1 represents the loss on the finest level, and L_4 is the loss on the most coarse level. A natural loss is a softmax distribution over candidate disparities at the current pyramid level. We found that expected disparity, as in GCNet [12], worked better.

3.2. Stereo data augmentation

In order to train our network, we find it crucial to make use of high-res training data and specific strategies for data augmentation. We discuss both below. Most conventional stereo systems make several assumptions on the target and reference view image pairs, including 1) both images are under the same imaging condition, 2) cameras are perfectly calibrated and 3) there is no occlusion and each pixel can find a match. However, these assumptions do not always hold in real-world scenarios. We propose 3 asymmetric augmentation techniques to tackle these issues accordingly for our learning-based approach.

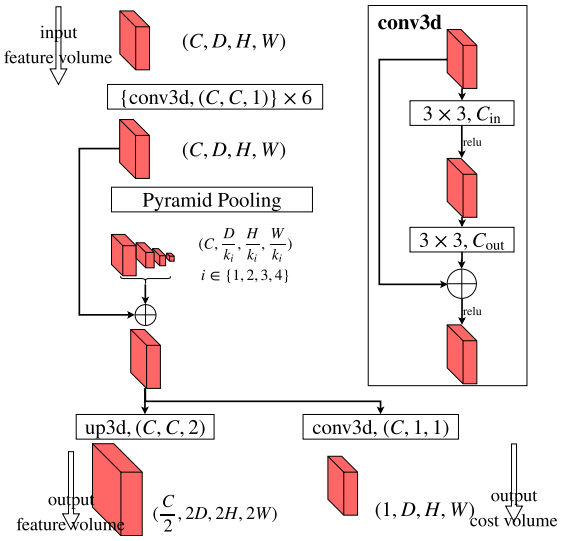

Figure 3: Hierarchical feature volume decoder. 3D Convolutions are defined by (in_channels,out_channels,stride) and feature volumes are defined by (channels, disparity_channels, height, width). To reduce memory constraints, we make use of strided disparity channels for the last and second-to-last volume in the pyramid.

y-disparity augmentation: Most stereo systems assume that cameras are perfectly calibrated and correspondences lie on the same horizontal scan-line. However, it is difficult to perfectly calibrate a high-res image pair, particularly during large temperature changes and vibrations [9]. Such errors result in ground-truth disparity matches that have a y component (e.g., match to a different horizontal scanline). Hierarchical matching partially mitigates this issue by biasing matching at the coarse scale, where calibration error is not as severe. Another approach is to force the network to *learn* robustness to such errors in training time. Notice that errors in camera calibration can be represented by a homography $H \in \mathbb{R}^{3\times3}$, resulting in $I_{imperfect}(\mathbf{x}) = I_{perfect}(\omega(\mathbf{x}, H\mathbf{x}))$ where $\mathbf{x}$ is the image coordinate. To mimic the real-world calibration error, we warp the target view image according to the calibration error matrix. To further constrain the space of the warped images, we limit H to be a rigid 2D transformation.

Asymmetric chromatic augmentation: It is inevitable that stereo cameras may under different lighting and exposure conditions, for example, one camera is under the shadow. Therefore making algorithms robust to such imaging asymmetry is critical out of safety concern. We achieve

Name	Res	Size	Scene	Real
Sintel [4]	0.45	1064	In/Outdoor	N
ETH3D [28]	0.46	27	Campus	Y
KITTI-15 [21]	0.47	200	Driving	Y
Sceneflow [20]	0.52	30k	All	N
Middlebury [26]	6.00	23	Indoor	Y
HR-VS (Ours)	5.07	780	Driving	N
HR-RS (Ours)	4.65	33	Driving	Y

Table 1: Summary of datasets for stereo matching, where the first group contains low-resolution datasets and the second group contains high-resolution datasets. We can see: 1) There is a lack of large-scale high-resolution stereo dataset, especially for outdoor scenes. 2) There is also a lack of high-resolution stereo matching benchmark for driving scenario depth sensing. The datasets we proposed bridge these gaps. Resolution (Res.) is shown in megapixel.

this goal by applying different chromatic augmentation to both reference and target images, in the hope that our feature encoding network may learn a representation robust to such imaging variants.

Asymmetric masking: Most stereo matching algorithms assume that correspondences always exist in the target view. However, this assumption does not hold when occlusion occurs or correspondences are difficult to find. On the other hand, monocular cues, such as shape and continuity, as well as contextual information are found helpful in estimating the disparity. To force the model to rely more on contextual cues, we apply asymmetric masking, which randomly replaces a rectangular region in the target view with mean RGB values of the whole image [29].

3.3. High-resolution datasets

End-to-end high-resolution stereo matching requires high-resolution datasets to make it effective. However, as shown in Table 1, there exist very few such datasets for both training and testing purpose. Middlebury-v3 [26] is the only publicly available dataset for high-resolution stereo matching. However, it contains very few samples and there are no outdoor/driving scenes. To bridge this gap, we aggregate two datasets for high-resolution stereo matching on driving scenes. Synthetic HR-VS is collected for training high-resolution stereo models, while the high-res real stereo (HR-RS) dataset is collected to benchmark high-resolution stereo matching methods under real-world driving scenes.

High-res virtual stereo (HR-VS) dataset: HR-VS is collected using open-sourced Carla simulator [6]. Under 4 weather conditions while maneuvering in Town01, we collected 780 pairs of training data at 2056×2464 resolution. Camera baseline and focal length are set as 0.54m and

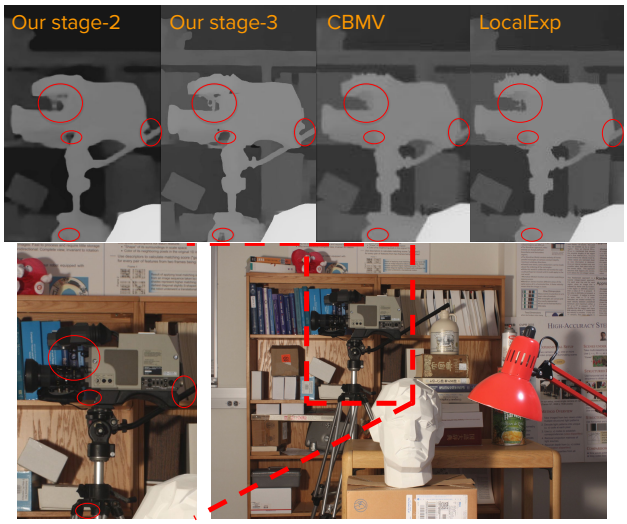

Figure 4: Effectiveness of high-res on Middlebury test image "Newkuba". LocalExp [34] and CBMV_ROB [2] take half-res inputs and rank 1st and 2nd on the bad-1.0 metric over all published methods. As shown in the circled regions, our method gives better details on thin structures.

3578px respectively. Pixels with a depth greater than 200m or a disparity greater than 768px are removed to mimic the disparity distribution of real-world driving scenarios, resulting in a disparity range of $[9.66, 768]$px and depth range of $[2.52, 200]$m. Sample images and ground-truth can be found in the supplements.

High-res real stereo (HR-RS) benchmark: HR-RS includes 33 pairs of images and disparity ground-truths collected using high-resolution stereo cameras and LiDAR while driving in the urban scenes. Images are rectified and cropped to 1918×2424. LiDAR point clouds are projected to image planes and converted to disparities, resulting in a range of $[5.4, 182.3]$px. We also manually removed point clouds on dynamic objects to reduce camera-LiDAR registration error.

Benchmark protocol: Since we are motivated by autonomous depth sensing, it is crucial to know the ideal sensing range under different driving speeds. Under the assumption of dry road condition and maximum deceleration of a car as shown in the supplements, we calculate the safe stopping distance for speed $v \in \{25, 40, 55\}$mph as $d \in \{25, 60, 115\}$m respectively. To ensure the safety of the driver and passengers, we are interested in making correct predictions within these stopping distances, i.e., when an object enters the stopping distance, we have to correctly perceive it. This gives us three sets of metrics with regard to driving speeds, which we refer to as short-range (0-25m), middle-range (25-60m) and long-range (60-115m) metrics. For each safe distance range, we use the same set of metrics in pixel space following [26].

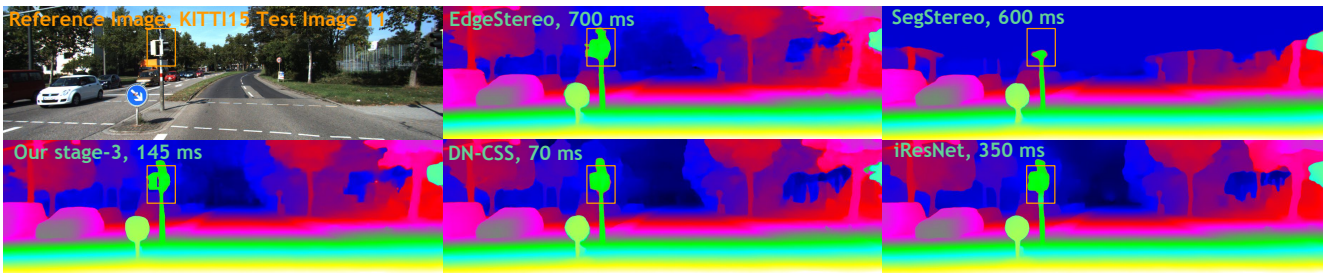

Figure 5: Qualitative results on KITTI-15 benchmark. As shown in the brown rectangle, our hierarchical stereo matching algorithm correctly finds the skinny structures and depth discontinuities while running faster than most SOTA (at 6.9 FPS).

4. Experiments

In this section, we evaluate our hierarchical stereo matching (HSM) network on public benchmarks including high-res Middlebury-v3 and low-res KITTI, as well as our proposed high-resolution benchmark, i.e., HR-RS.

4.1. Setup

Datasets: We used 4 publicly available datasets, including Middlebury-v3, KITTI-15, ETH3D and Sceneflow [20, 21, 26, 28], as well as HR-VS dataset for training. Middlebury-v3 contains 10 high-resolution training image pairs where each has variants with imperfect calibration, different exposures, and different lighting conditions, resulting in 60 pairs in total. KITTI-15 contains 200 low-resolution pairs and ETH3D contains 27 low-resolution pairs with sparsely labeled ground-truth. Sceneflow contains around 30k synthetic image pairs with dense disparity labels. And HR-VS contains 780 training pairs.

Implementation: We implemented the HSM network using Pytorch. We train the model using Adam optimizer with a batch size of 24 on a machine with 4 Titan X Pascal GPUs, while setting the initial learning rate to 0.001 and betas to $(0.9, 0.999)$. We train for 9 epochs and then shrink the learning rate by 10.

During training, we augment Middlebury, KITTI-15, ETH3D [20, 21, 26] and HR-VS to the same size as Sceneflow, resulting in around 170k training samples. We perform both symmetric and asymmetric augmentations on the fly. Asymmetric chromatic augmentations include randomly applying different brightness ($[0.5, 2]$), gamma ($[0.8, 1.2]$) and contrast ($[0.8, 1.2]$) to target and inference images. We apply y-disparity augmentation by uniformly sample a delta y-translation ($[0, 2]$px) and rotation($[0, 0.1]$ deg) at a chance of 0.5. We also apply asymmetric masking at a chance of 0.5 by uniformly sample the width ($[50, 150]$px) and height ($[50, 150]$px) of the mask and randomly placing it on the image. Symmetric augmentations include scaling ($[0.9, 2.4]$ for low-res images and $[0.225, 1.2]$ for high-res images) and randomly cropping to fix-sized (576×768) patches. We set the number of disparities for searching as 768px.

In test time, we make predictions at the last 3 scales from coarse to fine. With full-res input, these predictions are referred to as "Our-F1", "Our-F2" and "Our-F3" respectively, where "F" indicates the input resolution and the following number indicates the stage. Similarly, with half resolution input we have "Our-H2" and "Our-H3"; for quarter resolution inputs, we have "Our-Q3". When testing on Middlebury-v3 images, we set the disparity search range according to maximum disparities in the calibration files, while for 1.8-times upscaled KITTI-15 test images, we set disparity search range as 384. For HR-RS images, we set disparity search range as 512.

Metrics: Different metrics are used for different benchmarks. On Middlebury-v3 [26], we divide official metrics into 3 groups: 1) bad-4.0 (percentage of "bad" pixels whose error is great than 4.0), bad-2.0 and bad-1.0, which tolerates the small deviations such as quantization error. 2) avgerr (average absolute error in pixels) and rms (root-mean-square disparity error in pixels), which also takes into account the subpixel accuracy. 3) A99 (99% error quantile in pixels), A95 and A90, which ignores large deviations while measuring the accuracy. On KITTI-15 [21], we use official metrics D1-all, D1-bg and D1-fg, which measure the percentage of outliers for all pixels, background pixels and foreground pixels respectively. While on HR-RS, we separate pixels to different depth ranges and use the same set of metrics as Middlebury-v3.

4.2. Benchmark performance

High-res Middlebury-v3: Because we could not submit multiple on-demand outputs to the online test server for Middlebury-v3, we evaluate only our full-res, full-pipeline model HSM-F3. We compare against two groups of published methods. The first group of methods includes those running under **1s/image**, which we refer to as the "fast" group, and the second group of methods includes published SOTA that run slower. To compare with iResNet [16], we also add the 13 additional images as they did in training time, and in test time, we take the full resolution input images as input.

Method	time (s)	avgerr	rms	bad-4.0	bad-2.0	bad-1.0	A99	A95	A90
HSM-F3 (Ours)	0.51 (0.61)	**3.44**$_1$	**13.4**$_1$	**9.68**$_7$	**16.5**$_{16}$	**31.2**$_{28}$	**63.8**$_1$	**17.6**$_1$	**4.26**$_3$
SGM_ROB [8]	0.32	14.2	47.5	19.1	26.4	38.6	231	97.5	31.1
iResNet_ROB [16]	0.34 (0.42)	6.56	18.1	22.1	31.7	45.9	87.5	36.2	15.1
ELAS_ROB [7]	0.48	13.4	34.9	26.0	34.6	51.7	152	79.8	38.8
PSMNet_ROB [5]	0.64	8.78	23.3	29.2	47.2	67.3	106	43.4	22.8
DN-CSS_ROB [11]	0.66	5.48	16.8	19.6	28.3	41.3	82.0	25.6	13.3
LPS-H [30]	9.52	19.7	44.7	23.3	27.6	38.8	169	108	58.3
SPS [15]	25.5	24.8	65.8	23.6	28.1	36.9	284	172	89.8
LPS-F [30]	25.8	22.3	54.1	24.7	28.8	36.0	219	134	70.2
MC-CNN-acrt [40]	150	17.9	55.0	15.8	19.1	27.3	261	140	56.6
CBMV_ROB [2]	3946	6.65	27.7	10.3	13.3	21.6	134	36.7	8.41
LocalExp [34]	881	5.13	21.1	8.83	11.7	21.0	109	31.6	5.27

Table 2: Results on Middlebury-v3 benchmark where all pixels are evaluated. The subscript number shows the absolute rank among the benchmark, best results over the "fast" group are bolded, and best results overall are underlined. While running at 510ms/image on 6-megapixel images, our method out-performed all algorithms running faster under 1s/image by a large margin. Over the whole benchmark, we achieved 1st place on avgerr, rms, and A99, and A95.

We first present qualitative results as shown in Figure 4. Then the quantitative comparison to SOTA are shown in Table 2 for all pixels (for non-occluded pixels please refer to the supplements). Compared to MC-CNN-acrt on all-pixels, we reduced avgerr by 80.8%, rms by 75.6% and bad-4.0 by 38.7% while running 294.1 times faster. Compared to CBMV_ROB, we reduced avgerr by 48.3%, rms by 51.6% and bad-4.0 by 6.0% while running 7737.3 times faster. We run 0.31 times slower than iResNet_ROB but reduced avgerr by 47.6%, rms by 26.0% and bad-4.0 by 56.2%. We reached similar conclusion when evaluated on non-occluded pixels as shown in the supplements. Notice that in submission time, we used Tesla V100 on Amazon AWS, which gives us 510ms/image on average while iResNet used Titan Xp. To be fair, we measured running time for HSM-F3 and "iResNet" on the same Titan X Pascal GPU, as shown in parenthesis. We are 31.4% slower, but much more accurate.

Low-res KITTI: Though our focus is not on low-res, we still evaluate on KITTI-15 and partition the SOTA algorithms into two groups: those that run under 200ms and fits real-time needs[1], while the other group is slower but more accurate. During training time, we excluded the ETH3D and Sceneflow datasets and finetuned with Middlebury-v3, KITTI-12 and KITTI-15. At test time, we operate on image pairs up-sampled by a factor of 1.8. We first show qualitative results in Figure 5. Then we refer to Table 3 for quantitative results. We rank 1st among all published methods while running 3.8 times faster than "EdgeStereo", which ranks 2nd among the published methods.

[1]Since many autonomous robots employ sensors synced at 10fps, the deployment-ready version of a 200ms model is likely to fit real-time needs.

Method	D1-all	D1-bg	D1-fg	time (ms)
HSM-stage-3 (Ours)	**2.14**	**1.80**	**3.85**	150
DN-CSS [11]	2.94	2.39	5.71	70
DispNetC [20]	4.34	4.32	4.41	60
StereoNet [13]	4.83	4.30	7.45	20
EdgeStereo [31]	2.16	1.87	3.61	700
SegStereo [38]	2.25	1.88	4.07	600
PSMNet [5]	2.32	1.86	4.62	410
PDSNet [37]	2.58	2.29	4.05	500
CRL [23]	2.67	2.48	3.59	470
iResNet [16]	2.71	2.27	4.89	350
GCNet [12]	2.87	2.21	6.16	900

Table 3: Results on KITTI-15 benchmark where all pixels are evaluated and error metrics are shown in (%). Best results over the "real-time" group are bolded, and best results overall are underlined.

4.2.1 Results on HR-RS

Then we evaluate on HR-RS and compare against a subset of the previous arts under the protocol discussed in method section. ELAS [7] is taken from the Robust Vision Challenge official package, iResNet [16] is taken from their Github repository, and we implemented two-pass SGBM2 [8] using OpenCV (with SAD window size = 3, truncation value for pre-filter = 63, p1 = 216, p2 = 864, uniqueness ratio = 10, speckle window size = 100, speckle range = 32). The results from SGBM2 is also post-processed using weighted least square filter with default parameters. "HSM-F2" means the model operates on full

Method	time (ms)	bad-4.0 (%)			
		S	M	L	All
HSM-F1	91	42.3	43.5	33.7	40.9
HSM-F2	175	16.5	18.8	17.1	17.1
HSM-F3	430	**15.7**	**16.7**	**14.9**	**15.5**
HSM-H2	42	25.9	27.6	34.4	26.9
HSM-H3	74	18.9	18.9	19.7	18.2
HSM-Q3	29	30.3	32.7	33.4	31.2
ELAS-H [7]	464	49.4	32.2	23.9	36.1
SGBM2-Q [8]	1321	50.8	27.3	19.5	32.8
iResNet-H [16]	410	32.1	24.4	22.8	25.8

Table 4: Results on HR-RS dataset. All pixels are evaluated at 3 ranges. S: 0-25m, M: 25-60m, L: 60-115m. HSM achieves significant improvement in all metrics compared to the baselines.

resolution images and make predictions at the second stage (2nd last scale). Results are shown in Table 4.

Anytime on-demand: Cutting off HSM-F at the second stage (HSM-F2), bad-4.0 on all pixels increases by 10.3% while being 1.46x faster, which is still more accurate than ELAS, SGBM and iResNet. Same as iResNet-H, HSM-H3 uses half resolution image as input, but runs 4.54x faster and produces 29.5% lower bad-4.0. Halting earlier (HSM-H2) increases error by 47.8% but is 0.76x faster, which is still more accurate than ELAS and SGM.

Long-range sensing: We analyze our methods for different distance ranges. Halting HSM-F earlier at the second stage (HSM-F2) only increase bad-4.0 on short-range pixels by 5.1%, which suggests that high-res inputs might not help. However, on long-range pixels it increases bad-4.0 by 14.8%. Interestingly, halting HSM-F earlier (HSM-F2) still produces more accurate long-range predictions than HSM-H3 (17.1% versus 19.7%). This is possibly because features pyramid already encodes detailed information from high-res inputs, which helps to predict accurate long-range disparities.

4.3. Diagnostics

We perform ablation study to reveal the strength of individual components of our method. We follow the same training protocol as described in the experiment setup, but train different models with the same pre-trained encoder weights. We train on a single Tesla V100 GPU with a batch size of 8 for 60k iterations. The learning rate is down-scaled by 10 for the last 10k iterations. Quantitative results are shown in Table 5. For the qualitative effects, please check out our supplements.

Feature volume fusion: When aggregating information across pyramid scales, we made the design choice to fuse coarse-scale 4D **feature** volumes instead of 3D **cost** volume. Our intuition was that "feature-volume fusion" tolerates incorrect coarse predictions since the final output does not directly depend on initial predictions. Also as shown in Table 5, we found that replacing "feature-volume fusion" with "cost-volume fusion" results in 13.1% more bad pixels with an error greater than 1px.

Method	avgerr	bad-1.0	bad-2.0*	time (ms)
Full-method	4.01	**46.93**	**26.88**	97
Cost-aggre.	4.02	53.07	31.03	98
- HR-VS	4.02	51.01	30.24	97
- ydisp	4.28	48.75	28.98	98
- multi-scale	4.20	48.53	28.83	97
- A-Chrom.	**3.91**	46.96	27.02	97

Table 5: Diagnostic table for removing individual components. "Cost-aggre." means replace feature volume fusion by cost volume aggregation. Results are ranked by bad-2.0.

High-res synthetic data: After removing HR-VS dataset from training, the bad-1.0 metric increases by 8.7%, and the bad-2.0 metric increases by 11.1%, which shows synthetic high-res data is the key to train high-resolution stereo networks.

y-disparity augmentation: y-disparity augmentation is an effective way of forcing the network to learn features robust to camera calibration error. As shown in Table 5, removing y-disparity augmentation increase bad-1.0 by 7.2%.

Multi-scale loss: Multi-scale training loss regularizes the network by forcing it to learn multiple prediction tasks, while also helps gradients flow through multiple scales. We found that removing Multi-scale loss increase bad-2.0 by 6.8%.

Asymmetric chromatic augmentation: We found removing asymmetric chromatic augmentation does not harm performance on Middlebury additional images. This is probably because we introduce extra noise into training pairs, which harms our predictions on normal images (with the same exposure/lighting). However, we found this technique is helpful in making the network robust to asymmetric imaging conditions.

Conclusion

With the help of hierarchical designs, high-resolution synthetic dataset and asymmetric augmentation techniques, our model achieves SOTA performance on Middlebury-v3 and KITTI-15 while running significantly faster than prior arts. We are also able to perform on-demand disparity estimation at different scales, making possible accurate depth prediction of close-by objects in real-time.

References

[1] Padmanabhan Anandan. A computational framework and an algorithm for the measurement of visual motion. *International Journal of Computer Vision*, 2(3):283–310, 1989. 2

[2] Konstantinos Batsos, Changjiang Cai, and Philipos Mordohai. CBMV: A coalesced bidirectional matching volume for disparity estimation. In *CVPR*, 2018. 5, 7

[3] Thomas Brox and Jitendra Malik. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):500–513, 2011. 2

[4] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. A naturalistic open source movie for optical flow evaluation. In *ECCV*, 2012. 5

[5] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *CVPR*, 2018. 2, 4, 7

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *CoRL*, 2017. 5

[7] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient large-scale stereo matching. In *ACCV*, 2010. 2, 7, 8

[8] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2008. 2, 7, 8

[9] Heiko Hirschmuller and Stefan Gehrig. Stereo matching in the presence of sub-pixel calibration errors. In *CVPR*, 2009. 4

[10] 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*, 35(2):504–511, 2013. 2

[11] Eddy Ilg, Tonmoy Saikia, Margret Keuper, and Thomas Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *ECCV*, 2018. 2, 7

[12] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, and Peter Henry. End-to-end learning of geometry and context for deep stereo regression. In *ICCV*, 2017. 2, 3, 4, 7

[13] Sameh Khamis, Sean Fanello, Christoph Rhemann, Adarsh Kowdle, Julien Valentin, and Shahram Izadi. Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction. In *ECCV*, 2018. 2, 4, 7

[14] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *ICCV*, 2001. 2

[15] Chloe LeGendre, Konstantinos Batsos, and Philippos Mordohai. High-resolution stereo matching based on sampled photoconsistency computation. In *BMVC*, 2017. 2, 7

[16] Zhengfa Liang, Yiliu Feng, Yulan Guo, Hengzhu Liu, Wei Chen, Linbo Qiao, Li Zhou, and Jianfeng Zhang. Learning for disparity estimation through feature constancy. In *CVPR*, 2018. 2, 6, 7, 8

[17] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2

[18] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981. 2

[19] Wenjie Luo, Alexander G Schwing, and Raquel Urtasun. Efficient deep learning for stereo matching. In *CVPR*, 2016. 2

[20] N. Mayer, E. Ilg, P. Häusser, 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 *CVPR*, 2016. 2, 3, 5, 6, 7

[21] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *CVPR*, 2015. 2, 5, 6

[22] Yuichi Ohta and Takeo Kanade. Stereo by intra-and inter-scanline search using dynamic programming. *IEEE Transactions on pattern analysis and machine intelligence*, (2):139–154, 1985. 2

[23] Jiahao Pang, Wenxiu Sun, Jimmy SJ Ren, Chengxi Yang, and Qiong Yan. Cascade residual learning: A two-stage convolutional neural network for stereo matching. In *ICCV*, 2017. 2, 3, 7

[24] Peter Pinggera, David Pfeiffer, Uwe Franke, and Rudolf Mester. Know your limits: Accuracy of long range stereoscopic object measurements in practice. In *ECCV*, 2014. 1

[25] 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*, pages 234–241. Springer, 2015. 2

[26] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. High-resolution stereo datasets with subpixel-accurate ground truth. In *German Conference on Pattern Recognition*, 2014. 2, 5, 6

[27] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002. 2

[28] 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 *CVPR*, 2017. 5, 6

[29] Krishna Kumar Singh and Yong Jae Lee. Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization. In *ICCV*, 2017. 5

[30] Sudipta N Sinha, Daniel Scharstein, and Richard Szeliski. Efficient high-resolution stereo matching using local plane sweeps. In *CVPR*, 2014. 2, 7

[31] Xiao Song, Xu Zhao, Hanwen Hu, and Liangji Fang. Edgestereo: A context integrated residual pyramid network for stereo matching. In *ACCV*, 2018. 7

[32] Christoph Strecha, Wolfgang Von Hansen, Luc Van Gool, Pascal Fua, and Ulrich Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *CVPR*, 2008. 1, 2

[33] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *CVPR*, 2018. 3

[34] 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*, 40(11):2725–2739, 2018. 5, 7

[35] Engin Tola, Vincent Lepetit, and Pascal Fua. A fast local descriptor for dense matching. In *CVPR*, 2008. 2

[36] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An efficient dense descriptor applied to wide-baseline stereo. *IEEE transactions on pattern analysis and machine intelligence*, 32(5):815–830, 2010. 2

[37] Stepan Tulyakov, Anton Ivanov, and Francois Fleuret. Practical deep stereo (pds): Toward applications-friendly deep stereo matching. In *NeurIPS*, 2018. 7

[38] Guorun Yang, Hengshuang Zhao, Jianping Shi, Zhidong Deng, and Jiaya Jia. Segstereo: Exploiting semantic information for disparity estimation. In *ECCV*, 2018. 2, 7

[39] Qingxiong Yang, Liang Wang, Ruigang Yang, Shengnan Wang, Miao Liao, and David Nister. Real-time global stereo matching using hierarchical belief propagation. In *BMVC*, 2006. 2

[40] Jure Zbontar and Yann LeCun. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research*, 17(1-32):2, 2016. 2, 7

[41] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. 3