Anytime Stereo Image Depth Estimation on Mobile Devices

Yan Wang*[1], Zihang Lai*[2], Gao Huang[1], Brian H. Wang[1], Laurens van der Maaten[3], Mark Campbell[1], and Kilian Q. Weinberger[1]

***Abstract*— Many applications of stereo depth estimation in robotics require the generation of accurate disparity maps in real time under significant computational constraints. Current state-of-the-art algorithms force a choice between either generating accurate mappings at a slow pace, or quickly generating inaccurate ones, and additionally these methods typically require far too many parameters to be usable on power- or memory-constrained devices. Motivated by these shortcomings, we propose a novel approach for *disparity prediction* in the *anytime* setting. In contrast to prior work, our end-to-end learned approach can trade off computation and accuracy at inference time. Depth estimation is performed in stages, during which the model can be queried at any time to output its current best estimate. Our final model can process 1242×375 resolution images within a range of 10-35 FPS on an NVIDIA Jetson TX2 module with only marginal increases in error – using two orders of magnitude fewer parameters than the most competitive baseline. The source code is available at `https://github.com/mileyan/AnyNet`.**

I. INTRODUCTION

Depth estimation from stereo camera images is an important task for 3D scene reconstruction and understanding, with numerous applications ranging from robotics [30], [51], [39], [42] to augmented reality [53], [1], [35]. High-resolution stereo cameras provide a reliable solution for 3D perception - unlike time-of-flight cameras, they work well both indoors and outdoors, and compared to LiDAR they are substantially more affordable and energy-efficient [29]. Given a rectified stereo image pair, the focal length, and the stereo baseline distance between the two cameras, depth estimation can be cast into a stereo matching problem, the goal of which is to find the disparity between corresponding pixels in the two images. Although disparity estimation from stereo images is a long-standing problem in computer vision [28], in recent years the adoption of deep convolutional neural networks (CNN) [52], [32], [20], [25], [36] has led to significant progress in the field. Deep networks can solve the matching problem via supervised learning in an end-to-end fashion, and they have the ability to incorporate local context as well as prior knowledge into the estimation process.

On the other hand, deep neural networks tend to be computationally intensive and suffer from significant latency when processing high-resolution stereo images. For example, PSMNet [4], arguably the current state-of-the-art algorithm for depth estimation, obtains a frame rate below 0.3FPS on the Nvidia Jetson TX2 GPU computing module — far too slow for timely obstacle avoidance by drones or other autonomous robots.

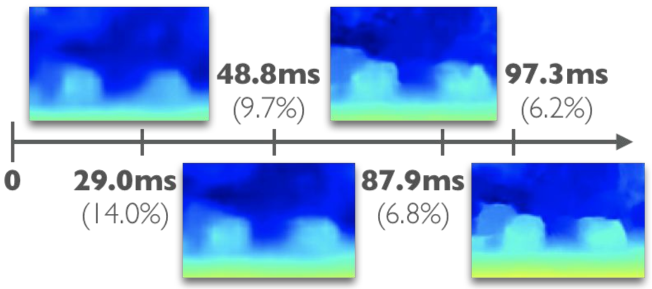


Fig. 1: Example timeline of AnyNet predictions. As time progresses the depth estimation becomes increasingly accurate. The algorithm can be polled at any time to return the current best estimate of the depth map. The initial estimates may be sufficient to trigger an obstacle avoidance maneuver, whereas the later images contain enough detail for more advanced path planning procedures. (3-pixel error rate below time.)

In this paper, we argue for an *anytime* computational approach to disparity estimation, and present a model that trades off between speed and accuracy dynamically (see Figure 1). For example, an autonomous drone flying at high speed can poll our 3D depth estimation method at a high frequency. If an object appears in its flight path, it will be able to perceive it rapidly and react accordingly by lowering its speed or performing an evasive maneuver. When flying at low speed, latency is not as detrimental, and the same drone could compute a higher resolution and more accurate 3D depth map, enabling tasks such as high precision navigation in crowded scenes or detailed mapping of an environment.

The computational complexity of depth estimation with convolutional networks typically scales cubically with the image resolution, and linearly with the maximum disparity that is considered [20]. Keeping these characteristics in mind, we refine the depth map successively, while always ensuring that either the resolution or the maximum disparity range is sufficiently low to ensure minimal computation time. We start with low resolution ($1/16$) estimates of the depth map at the full disparity range. The cubic complexity allows us to compute this initial depth map in a few milliseconds (where the bulk of the time is spent on the initial feature extraction and down-sampling). Starting with this low resolution estimate, we successively increase the resolution of the disparity map by up-sampling and subsequently correcting the errors that are now apparent at the higher resolution. Correction is

* Authors contributed equally
1 Cornell University. {yw763, gh349, bhw45, mc288, kqw4}@cornell.edu
2 University of Oxford. This work performed during Zihang Lai's internship at Cornell University. zihang.lai@cs.ox.ac.uk
3 Facebook AI Research. lvdmaaten@gmail.com

performed by predicting the residual error of the up-sampled disparity map from the input images with a CNN. Despite the higher resolution used, these updates are still fast because the residual disparity can be assumed to be bounded within a few pixels, allowing us to restrict the maximum disparity, and associated computation, to a mere $10-20\%$ of the full range.

These successive updates avoid full-range disparity computation at all but the initial low resolution setting, and ensure that all computation is re-used, setting our method apart from most existing multi-scale network structures [40], [15], [19]. Furthermore, our algorithm can be polled at any time in order to retrieve the current best estimated depth map. A wide range of possible frame rates are attainable (10-35FPS on a TX2 module), while still preserving accurate disparity estimation in the high-latency setting. Our entire network can be trained end-to-end using a joint loss over all scales, and we refer to it as Anytime Stereo Network (AnyNet).

We evaluate AnyNet on multiple benchmark data sets for depth estimation, with various encouraging findings: Firstly, AnyNet obtains competitive accuracy with state of the art approaches, while having orders of magnitude fewer parameters than the baselines. This is especially impactful for resource-constrained embedded devices. Secondly, we find that deep convolutional networks are highly capable at predicting residuals from coarse disparity maps. Finally, including a final spatial propagation model (SPNet) [26] significantly improves the disparity map quality, yielding state-of-the-art results at a fraction of the computational cost (and parameter storage requirements) of existing methods.

II. Related Work

*a) **Disparity estimation**:* Traditional approaches to disparity estimation are based on matching features between the left and right images [2], [41]. These approaches are typically comprised of the following four steps: (1) computing the costs of matching image patches over a range of disparities, (2) smoothing the resulting cost tensor via aggregation methods, (3) estimating the disparity by finding a low-cost match between the patches in the left image and those in the right image, and (4) refining these disparity estimates by introducing global smoothness priors on the disparity map [41], [13], [11], [14]. Several recent papers have studied the use of convolutional networks in step (1). In particular, Zbontar & LeCun [52] use a Siamese convolutional network to predict patch similarities for matching left and right patches. Their method was further improved via the use of more efficient matching networks [29] and deeper highway networks trained to minimize a multilevel loss [43].

*b) **End-to-end disparity prediction**:* Inspired by these initial successes of convolutional networks in disparity estimation, as well as by their successes in semantic segmentation [27], optical flow computation [7], [17], and depth estimation from a single frame [5], several recent studies have explored end-to-end disparity estimation models [32], [20], [25], [36]. For example, in [32], the disparity prediction problem is formulated as a supervised learning problem, and a convolutional network called DispNet is proposed that directly predicts disparities for an image pair. Improvements made by DispNet include a cascaded refinement procedure [36]. Other studies adopt the *correlation layer* introduced in [7] to obtain the initial matching costs; a set of two convolutional networks are trained to predict and further refine the disparity map for the image pair [25]. Several prior studies have also explored moving away from the supervised learning paradigm by performing depth estimation in an unsupervised fashion using stereo images [9] or video streams [54].

Our work is partially inspired by the Geometry and Context Network (GCNet) proposed in [20]. In order to predict the disparity map between two images, GCNet combines a 2D Siamese convolutional network operating on the image pair with a 3D convolutional network that operates on the matching cost tensor. GCNet is trained in an end-to-end fashion, and is presently one of the state-of-the-art methods in terms of accuracy and computational efficiency. Our model is related to GCNet in that it has the same two stages (a 2D Siamese image convolutional network and a 3D cost tensor convolutional network), but it differs in that it can perform *anytime prediction* by rapidly producing an initial disparity map prediction and then progressively predicting the residual to correct this prediction. LiteFlowNet [16] also tries to predict the residual for optical flow computation. However, LiteFlowNet uses the residual to facilitate large-displacement flow inference rather than computational speedup.

*c) **Anytime prediction**:* There exists a substantial body of work on machine learned models with computational budget constraints at inference time [46], [10], [18], [49], [50], [47], [15]. Most of these approaches are based on ensembles of decision-tree classifiers [46], [49], [10], [50] which allow for tree-by-tree evaluation, facilitating the progressive prediction updates that are the hallmark of anytime prediction. Several recent studies have explored anytime prediction with image classification CNNs that dynamically evaluate parts of the network to progressively refine their predictions [24], [15], [45], [48], [33], [6]. Our work differs from these earlier anytime CNN models in that we focus on the structured prediction problem of disparity-map estimation, rather than on image classification tasks. Our models exploit particular properties of the disparity-prediction problem: namely, that progressive estimation of disparities can be achieved by progressively increasing the resolution of the image data within the internal representation of the CNN.

III. AnyNet

Fig. 2 shows a schematic layout of the AnyNet architecture. An input image pair first passes through the U-Net feature extractor, which computes feature maps at several output resolutions (of scale 1/16, 1/8, 1/4). In the first stage, only the lowest-scale features (1/16) are computed and passed through a disparity network (Fig. 4) to produce a low-resolution disparity map (*Disparity Stage 1*). A disparity map estimates the horizontal offset of each pixel in the right input image w.r.t. the left input image, and can be used to compute

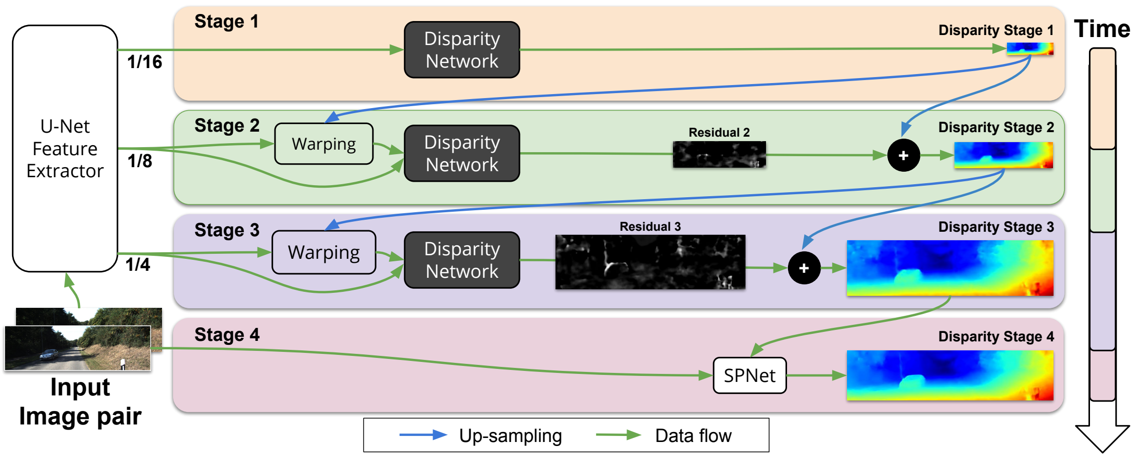


Fig. 2: Network structure of AnyNet.

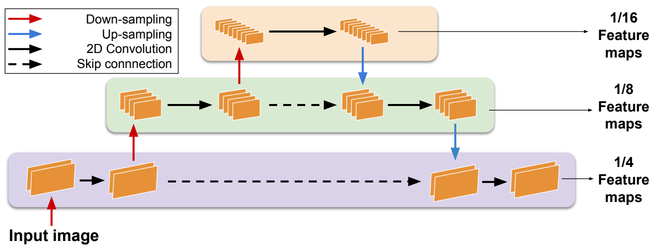


Fig. 3: U-Net Feature Extractor. See text for details.

a depth map. Because of the low input resolution, the entire Stage 1 computation requires only a few milliseconds. If more computation time is permitted, we enter Stage 2 by continuing the computation in the U-Net to obtain larger-scale (1/8) features. Instead of computing a full disparity map at this higher resolution, in Stage 2 we simply correct the already-computed disparity map from Stage 1. First, we up-scale the disparity map to match the resolution of Stage 2. We then compute a residual map, which contains small corrections that specify how much the disparity map should be increased or decreased for each pixel. If time permits, Stage 3 follows a similar process as Stage 2, and doubles the resolution again from a scale of 1/8 to 1/4. Stage 4 refines the disparity map from Stage 3 with an SPNet [26].

In the remainder of this section we describe the individual components of our model in greater detail.

a) U-Net Feature Extractor: Fig 3 illustrates the U-Net [38] Feature Extractor in detail, which is applied to both the left and right image. The U-Net architecture computes feature maps at various resolutions (1/16, 1/8, 1/4), which are used as input at stages 1-3 and only computed when needed. The original input images are down-sampled through max-pooling or strided convolution and then processed with convolutional filters. Lower resolution feature maps capture the global context, whereas higher resolution feature maps capture local details. At scale 1/8 and 1/4, the final convolutional layer incorporates the previously computed lower-scale features.

b) Disparity Network: The disparity network (Fig. 4) takes as input the feature maps from the left and right stereo images in order to compute a disparity map. We use this component to compute the initial disparity map (stage 1) as well as to compute the residual maps for subsequent corrections (stages 2 & 3). The disparity network first computes a disparity cost volume. Here, the cost refers to the similarity between a pixel in the left image and a corresponding pixel in the right image. If the input feature maps are of dimensions $H \times W$, the cost volume has dimensions $H \times W \times M$, where the (i, j, k) entry describes the degree to which pixel (i, j) of the left image matches pixel $(i, j-k)$ in the right image. M denotes the maximum disparity under consideration. We can represent each pixel (i, j) in the left image as a vector $\mathbf{p}^L_{ij}$, where dimension α corresponds to the (i, j) entry in the α^{th} input feature map associated with the left image. Similarly we can define $\mathbf{p}^R_{ij}$. The entry (i, j, k) in the cost volume is then defined as the L_1 distance between the two vectors $\mathbf{p}^L_{ij}$ and $\mathbf{p}^R_{i(j+k)}$, i.e. $C_{ijk} = \|\mathbf{p}^L_{ij} - \mathbf{p}^R_{i(j-k)}\|_1$.

This cost volume may still contain errors due to blurry objects, occlusions, or ambiguous matchings in the input images. As a second step in the disparity network (*3D Conv* in Fig 4), we refine the cost volume with several 3D convolution layers [20] to further improve the obtained cost volume.

The disparity for pixel (i, j) in the left image is k if the pixel $(i, j-k)$ in the right image is most similar. If the cost volume is exact, we could therefore compute the disparity of pixel (i, j) as $\hat{D}_{ij} = \text{argmin}_k C_{i,j,k}$. However, the cost estimates may be too noisy to search for the hard minimum. Instead, we follow the suggestion by Kendall et al. [20] and compute a weighted average (*Disparity regression* in Fig 4)

$$\hat{D}_{ij} = \sum_{k=0}^{M} k \times \frac{\exp(-C_{ijk})}{\sum_{k'=0}^{M} \exp(-C_{ijk'})}. \qquad (1)$$

If one disparity k clearly has the lowest cost (i.e. it is the only good match), it will be recovered by the weighted average. If there is ambiguity, the output will be an average of the viable candidates.

c) Residual Prediction: A crucial aspect of our AnyNet architecture is that we only compute the full disparity map at

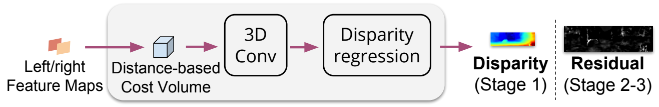


Fig. 4: Disparity network. See text for details.

a very low resolution in Stage 1. In Stages 2 & 3 we predict residuals [16]. The most expensive part of the disparity prediction is the construction and refinement of the cost volume. The cost volume scales $H \times W \times M$, where M is the maximum disparity. In high resolutions, the maximum disparity between two pixels can be very large (typically $M = 192$ pixels in the KITTI dataset [8]). By restricting ourselves to residuals, i.e. corrections of existing disparities, we can limit ourselves to $M = 5$ (corresponding to offsets $-2, -1, 0, 1, 2$) and obtain sizable speedups.

In order to compute residuals in stages 2 & 3, we first up-scale the coarse disparity map and use it to warp the input features at the higher scale (Fig. 2) by applying the disparity estimations pixel-wise. In particular, if the left disparity of pixel (i, j) is estimated to be k, we overwrite the value of pixel (i, j) in each right feature map to the corresponding value of pixel $(i, j + k)$ (using zero if out of bounds). If the current disparity estimate is correct, the updated right feature maps should match the left feature maps. Due to the coarseness of the low resolution inputs, there is typically still a mismatch of several pixels, which we correct by computing residual disparity maps. Prediction of the residual disparity is accomplished similarly to the full disparity map computation. The only difference is that the cost volume is computed as $C_{ijk} = \|\mathbf{p}_{ij} - \mathbf{p}_{i(j-k+2)}\|_1$, and the resulting residual disparity map is added to the up-scaled disparity map from the previous stage.

d) Spatial Propagation Network: To further improve our results, we add a final fourth stage in which we use a Spatial Propagation Network (SPNet) [26] to refine our disparity predictions. The SPNet sharpens the disparity map by applying a local filter whose weights are predicted by applying a small CNN to the left input image. We show that this refinement improves our results significantly at relatively little extra cost.

IV. EXPERIMENTAL RESULTS

In this section, we empirically evaluate our method and compare it with existing stereo algorithms. In addition, we benchmark the efficiency of our approach on an Nvidia Jetson TX2 computing module.

a) ***Implementation Details****:* We implement AnyNet in PyTorch [37]. See Table I for a detailed network description. Our experiments use an AnyNet implementation with four stages, as shown in Figure 2 and described in the previous section. The maximum disparity is set to 192 pixels in the original image, which corresponds to a Stage 1 cost volume depth of $M = 192/16 = 12$. In Stages 2 & 3 the residual range is ± 2, corresponding to ± 16 pixels in Stage 2 and ± 8 pixels in Stage 3. All four stages, including the SPNet in Stage 4, are trained jointly, but the losses are weighted differently, with weights $\lambda_1 = 1/4$, $\lambda_2 = 1/2$, $\lambda_3 = 1$ and $\lambda_4 = 1$, respectively. In total, our model contains 40,000 parameters - this is an order of magnitude fewer parameters than StereoNet [21], and two orders of magnitude fewer than PSMNet [4]. Our model is trained end-to-end using Adam [22] with initial learning rate $5e^{-4}$ and batch size 6. On the Scene Flow dataset [32], the learning rate is kept constant, and the training lasts for 10 epochs in total. For the KITTI dataset we first pre-train the model on Scene Flow, before fine-tuning it for 300 epochs. The learning rate is divided by 10 after epoch 200. All input images are normalized to be zero-mean with unit variance. All experiments were conducted using original image resolutions. Using one GTX 1080Ti GPU, training on the Scene Flow dataset took 3.5 hours, and training on KITTI took 30 minutes. All results are averaged over five randomized 80/20 train/validation splits.

0	Input image
	2-D Unet features
1	3×3 conv with 1 filter
2	3×3 conv with stride 2 and 1 filter
3	2×2 maxpooling with stride 2
4-5	3×3 conv with 2 filters
6	2×2 maxpooling with stride 2
7-8	3×3 conv with 4 filters
9	2×2 maxpooling with stride 2
10-11	3×3 conv with 8 filters
12	Bilinear upsample layer 11 (features) into 2x size
13	Concatenate layer 8 and 12
14-15	3×3 conv with 4 filters
16	Bilinear upsample layer 15 (features) into 2x size
17	Concatenate layer 5 and 16
18-19	3×3 conv with 2 filters
	Cost volume
20	Warp and build cost volume from layer 11
21	Warp and build cost volume from layer 15 and layer 29
22	Warp and build cost volume from layer 19 and layer 36
	Regularization
23-27	$3 \times 3 \times 3$ 3-D conv with 16 filters
28	$3 \times 3 \times 3$ 3-D conv with 1 filter
29	Disparity regression
30	Upsample layer 29 to image size: **stage 1 disparity output**
31-35	$3 \times 3 \times 3$ 3-D conv with 4 filters
36	Disparity regression: residual of stage 2
37	Upsample 36 it into image size
38	Add layer 37 and layer 30
39	Upsample layer 38 to image size: **stage 2 disparity output**
40-44	$3 \times 3 \times 3$ 3-D conv with 4 filters
45	Disparity regression: residual of stage 3
46	Add layer 44 and layer 38
47	Upsample layer 46 to image size: **stage 3 disparity output**
	Spatial propagation network
48-51	3×3 conv with 16 filters (on input image)
52	3×3 conv with 24 filters: affinity matrix
53	3×3 conv with 8 filters (on layer 47)
54	Spatial propagate layer 53 with layer 52 (affinity matrix)
55	3×3 conv with 1 filters: **stage 4 disparity output**

TABLE I: Network configurations. Note that a *conv* stands for a sequence of operations: batch normalization, rectified linear units (ReLU) and convolution. The default stride is 1.

Figure 5 visualizes the disparity maps predicted at the four stages of our model. As more computation time is made available, AnyNet produces increasingly refined disparity maps. The final output from stage 4 is even sharper and more accurate, due to the SPNet post-processing.

b) ***Datasets****:* Our model is trained on the synthetic Scene Flow [32] dataset and evaluated on two real-world datasets, KITTI-2012 [8] and KITTI-2015 [34]. The Scene Flow dataset contains $22,000$ stereo image pairs for train-

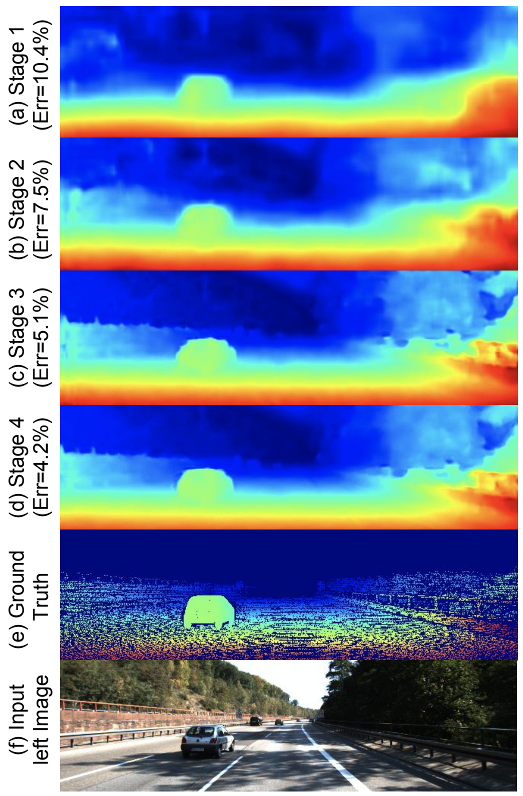


Fig. 5: (a)-(d) Disparity prediction from 4 stages of AnyNet on KITTI-2015. As a larger computational budget is made available, the prediction is refined and becomes more accurate. (e) shows the ground truth LiDAR image, and (f) shows the left input image.

ing, and $4,370$ image pairs for testing. Each image has a resolution of 960×540 pixels. As in [20], we train our model on 512×256 patches randomly cropped from the original images. The KITTI-2012 dataset contains 194 pairs of images for training and 195 for testing, while KITTI-2015 contains 200 image pairs for each. All of the KITTI images are of size 1242×375.

*c) **Baselines**:* Although state-of-the-art CNN based stereo estimation methods have been reported to reach 60FPS on a TITAN X GPU [21], they are far from achieving real-time performance on more resource-constrained computing devices such as the Nvidia Jetson TX2. Here, we present a controlled comparison on a TX2 between our method and four competitive baseline algorithms: PSMNet [4], StereoNet [21], DispNet [32], and StereoDNN [44]. The PSMNet model has two different versions: PSMNet-classic and PSMNet-hourglass. We use the former, as it is much more efficient than PSMNet-hourglass while having comparable accuracy. For StereoNet, we report running times using a Tensorflow implementation, which we found to be twice as fast as a PyTorch implementation.

Finally, we also compare AnyNet to two classical stereo matching approaches: Block Matching [23] and Semi-Global Block Matching [12], supported by OpenCV [3].

In order to collect meaningful results for these baseline methods on the TX2, we use down-sampled input images

Dataset	Stage 1 28.9ms	Stage 2 48.8ms	Stage 3 87.9ms	Stage 4 97.3ms
KITTI2012	15.1 ± 1.1	9.9 ± 0.6	6.7 ± 0.4	6.1 ± 0.3
KITTI2015	14.0 ± 0.7	9.7 ± 0.7	6.8 ± 0.6	6.2 ± 0.6

TABLE II: Three-Pixel error (%) of AnyNet on KITTI-2012 and KITTI-2015 datasets. Lower values are better.

for faster inference times. The baseline methods are re-implemented, and trained on down-sampled stereo images - this allows a fair comparison, since a model trained on full-sized images would be expected to suffer a significant performance decrease when given lower-resolution inputs. After obtaining a low-resolution prediction, we up-sample it to the original size using bilinear interpolation.

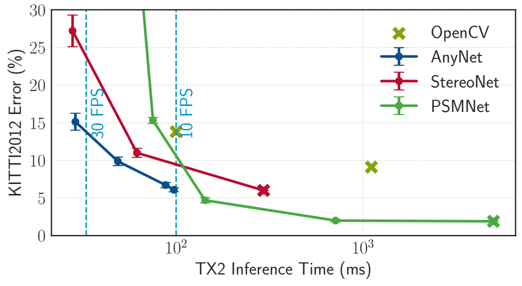
A. Evaluation Results

Table II contains numerical results for AnyNet on the KITTI-2012 and KITTI-2015 datasets. Additionally, Figures 6a and 6a demonstrate the evaluation error and inference time of our model as compared to baseline methods. Baseline algorithm results originally reported in [3], [21], [4], [31], [44] are shown plotted with crosses. For AnyNet as well as the StereoNet and PSMNet baselines, computations are performed across multiple down-sampling input resolutions. Results are generated from inputs at full resolution as well as at $1/4$, $1/8$, and $1/16$ resolution, with lower resolution corresponding to faster inference time as shown on Figs. 6a and 6b. As seen in both plots, only AnyNet and StereoNet are capable of rapid real-time prediction at $\geq$30 FPS, and AnyNet obtains a drastically lower error rate on both data sets. AnyNet is additionally capable of running at over 10 FPS even with full-resolution inputs, and at each possible inference time range, AnyNet clearly dominates all baselines in terms of prediction error. PSMNet is capable of producing the most accurate results overall, however this is only true at computation rates of 1 FPS or slower. We also observe that the only non-CNN based approach, OpenCV, is not competitive in any inference time range.

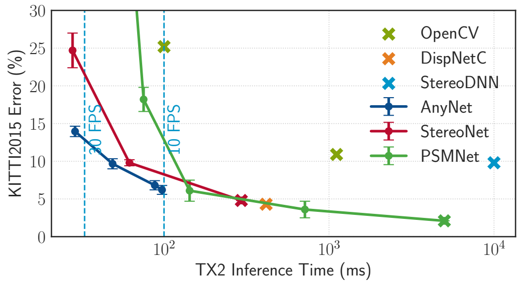
*a) **Anytime setting**:* We also evaluate AnyNet in the *anytime* setting, in which we can poll the model prematurely at any given time t in order to retrieve its most recent prediction. In order to mimic an anytime setting for the baseline OpenCV, StereoNet, and PSMNet models, we make predictions successively at increasingly higher input resolutions and execute them sequentially in ascending order of size. At time t we evaluate the most recently computed disparity map. Figures 6c and 6d show the three-pixel error rates in the anytime setting. Similarly to the non-anytime results, AnyNet obtains significantly more accurate results in the 10-30 FPS range. Furthermore, the times between disparity map completions (visualized as horizontal lines in Figs. 6c and 6d) are much shorter than for any of the baselines, reducing the amount of wasted computation if a query is issued during a disparity map computation.

B. Ablation Study

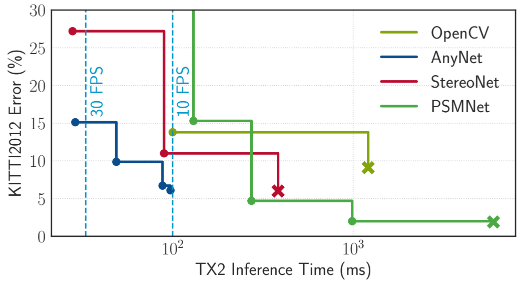
In order to examine the impact of various components of the AnyNet architecture, we conduct an ablation study



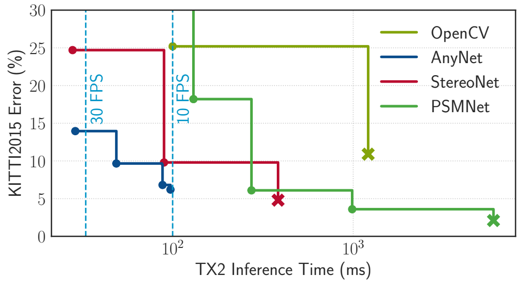
(a) KITTI-2012 results across different down-sampling resolutions



(b) KITTI-2015 results across different down-sampling resolutions



(c) KITTI-2012 results using the anytime setting



(d) KITTI-2015 results using the anytime setting

Fig. 6: Comparisons of the 3-pixel error rate (%) KITTI-2012/2015 datasets. Dots with error bars show accuracies obtained from our implementations. Crosses show values obtained from original publications.

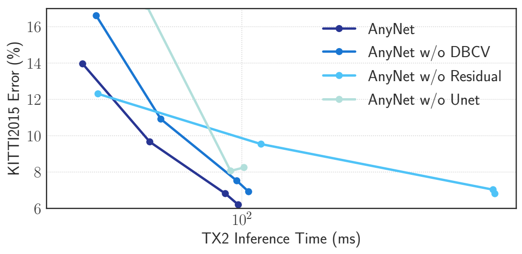


Fig. 7: Ablation results as three pixel error on KITTI-2015.

using three variants of our model. The first replaces the U-Net feature extractor with three separated ConvNets without shared weights; the second computes a full-scale prediction at each resolution level, instead of only predicting the residual disparity; while the third replaces the distance-based cost volume construction method with the method in PSMNet [4] that produces a stack of $2 \times M$ cost volumes. All ablated variants of our method are trained from scratch, and results from evaluating them on KITTI-2015 are shown in Fig. 7.

*a) **Feature extractor**:* We modify the model's feature extractor by replacing the U-Net with three separate 2D convolutional neural networks which are similar to one another in terms of computational cost. As seen in Fig. 7 (line AnyNet w/o UNet), the errors increase drastically in the first two stages (20.4% and 7.3%). We hypothesize that by extracting contextual information from higher resolutions, the U-Net produces high-quality cost volumes even at low resolutions. This makes it a desirable choice for feature extraction.

*b) **Residual Prediction**:* We compare our default network with a variant that refines the disparity estimation by directly predicting disparities, instead of residuals, in the second and third stages. Results are shown in Fig. 7 (line AnyNet w/o Residual). While this variant is capable of attaining similar accuracy to the original model, the evaluation time in the last two stages is increased by a factor of more than six. This increase suggests that the proposed method to predict residuals is highly efficient at refining coarse disparity maps, by avoiding the construction of large cost volumes which need to account for a large range of disparities.

*c) **Distance-based Cost Volume**:* Finally, we evaluate the distance-based method for cost volume construction, by comparing it to the method used in PSMNet [4]. This method builds multiple cost volumes without explicitly calculating the distance between features from the left and right images. The results in Fig. 7 (line AnyNet w/o DBCV) show that our distance-based approach is about 10% faster than this choice, indicating that explicitly considering the feature distance leads to a better trade-off between accuracy and speed.

V. Discussion and Conclusion

To the best of our knowledge, AnyNet is the first algorithm for anytime depth estimation from stereo images. As (low-power) GPUs become more affordable and are increasingly incorporated into mobile computing devices, anytime depth estimation will enable accurate and reliable real-time depth estimation for a large variety of robotic applications.

Acknowledgment

This research is supported in part by grants from the National Science Foundation (III-1618134, III-1526012, IIS1149882, IIS-1724282, and TRIPODS-1740822), the Office of Naval Research DOD (N00014-17-1-2175), and the Bill and Melinda Gates Foundation. We are thankful for the generous support of SAP America Inc. We thank Camillo J. Taylor for helpful discussion.

References

[1] H. A. Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets computer vision: Efficient data generation for urban driving scenes," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 961–972, 2018.

[2] S. T. Barnard and M. A. Fischler, "Computational stereo," *ACM Computing Surveys (CSUR)*, vol. 14, no. 4, pp. 553–572, 1982.

[3] G. Bradski and A. Kaehler, "Opencv," *Dr. Dobbs journal of software tools*, vol. 3, 2000.

[4] J.-R. Chang and Y.-S. Chen, "Pyramid stereo matching network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5410–5418.

[5] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *ICCV*, 2015.

[6] M. Figurnov, M. Collins, Y. Zhu, L. Zhang, J. Huang, D. Vetrov, and R. Salakhutdinov, "Spatially adaptive computation time for residual networks," in *CVPR*, 2017.

[7] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. Van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," *arXiv preprint arXiv:1504.06852*, 2015.

[8] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[9] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," *CVPR*, vol. 2, no. 6, p. 7, 2017.

[10] A. Grubb and D. Bagnell, "Speedboost: Anytime prediction with uniform near-optimality." in *AISTATS*, vol. 15, 2012, pp. 458–466.

[11] H. Hirschmuller and D. Scharstein, "Evaluation of stereo matching costs on images with radiometric differences," *TPAMI*, vol. 31, pp. 1582–1599, 2009.

[12] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

[13] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz, "Fast cost-volume filtering for visual correspondence and beyond," *TPAMI*, vol. 35, pp. 504–511, 2013.

[14] X. Hu and P. Mordohai, "A quantitative evaluation of confidence measures for stereo vision," *TPAMI*, vol. 34, pp. 2121–2133, 2012.

[15] G. Huang, D. Chen, T. Li, F. Wu, L. van der Maaten, and K. Q. Weinberger, "Multi-scale dense convolutional networks for efficient prediction," in *ICLR*, 2017.

[16] T.-W. Hui, X. Tang, and C. C. Loy, "Liteflownet: A lightweight convolutional neural network for optical flow estimation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018, pp. 8981–8989.

[17] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *CVPR*, 2017.

[18] S. Karayev, M. Fritz, and T. Darrell, "Anytime recognition of objects and scenes," in *CVPR*, 2014, pp. 572–579.

[19] T. Ke, M. Maire, and S. X. Yu, "Neural multigrid," *CoRR*, vol. abs/1611.07661, 2016. [Online]. Available: http://arxiv.org/abs/1611.07661

[20] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-end learning of geometry and context for deep stereo regression," in *ICCV*, 2017.

[21] S. Khamis, S. Fanello, C. Rhemann, A. Kowdle, J. Valentin, and S. Izadi, "Stereonet: Guided hierarchical refinement for real-time edge-aware depth prediction," *arXiv preprint arXiv:1807.08865*, 2018.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[23] K. Konolige, "Small vision systems: Hardware and implementation," in *Robotics research*. Springer, 1998, pp. 203–212.

[24] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," in *ICLR*, 2017.

[25] Z. Liang, Y. Feng, Y. Guo, H. Liu, L. Qiao, W. Chen, L. Zhou, and J. Zhang, "Learning deep correspondence through prior and posterior feature constancy," *arXiv preprint arXiv:1712.01039*, 2017.

[26] S. Liu, S. De Mello, J. Gu, G. Zhong, M.-H. Yang, and J. Kautz, "Learning affinity via spatial propagation networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 1520–1530.

[27] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[28] B. D. Lucas, T. Kanade *et al.*, "An iterative image registration technique with an application to stereo vision," *IJCAI*, 1981.

[29] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *CVPR*, 2016, pp. 5695–5703.

[30] M. Mancini, G. Costante, P. Valigi, and T. A. Ciarfuglia, "Fast robust monocular depth estimation for obstacle detection with fully convolutional networks," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4296–4303.

[31] 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 *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, arXiv:1512.02134. [Online]. Available: http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16

[32] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4040–4048.

[33] M. McGill and P. Perona, "Deciding how to decide: Dynamic routing in artificial neural networks," in *arXiv:1703.06217*, 2017.

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

[35] C. Nguyen, S. DiVerdi, A. Hertzmann, and F. Liu, "Depth conflict reduction for stereo vr video interfaces," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 2018, p. 64.

[36] J. Pang and W. Sun, "Cascade residual learning: A two-stage convolutional neural network for stereo matching," in *arXiv:1708.09204*, 2017.

[37] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[38] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[39] A. Saxena, J. Schulte, A. Y. Ng *et al.*, "Depth estimation using monocular and stereo cues." in *IJCAI*, vol. 7, 2007.

[40] S. Saxena and J. Verbeek, "Convolutional neural fabrics," in *NIPS*, 2016, pp. 4053–4061.

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

[42] K. Schmid, T. Tomic, F. Ruess, H. Hirschmüller, and M. Suppa, "Stereo vision based indoor/outdoor navigation for flying robots," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 3955–3962.

[43] A. Shaked and L. Wolf, "Improved stereo matching with constant highway networks and reflective confidence learning," in *CVPR*, 2017.

[44] N. Smolyanskiy, A. Kamenev, and S. Birchfield, "On the importance of stereo for accurate depth estimation: An efficient semi-supervised deep neural network approach," *arXiv preprint arXiv:1803.09719*, 2018.

[45] A. Veit and S. Belongie, "Convolutional networks with adaptive computation graphs," in *arXiv:1711.11503*, 2017.

[46] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 4, no. 34–47, 2001.

[47] J. Wang, K. Trapeznikov, and V. Saligrama, "Efficient learning by directed acyclic graph for resource constrained prediction," in *NIPS*, 2015, pp. 2152–2160.

[48] Z. Wu, T. Nagarajan, A. Kumar, S. Rennie, L. Davis, K. Grauman, and R. Feris, "Blockdrop: Dynamic inference paths in residual networks," in *arXiv:1711.08393*, 2017.

[49] Z. Xu, K. Q. Weinberger, and O. Chapelle, "The greedy miser: Learning under test-time budgets," in *ICML*, 2012, pp. 1175–1182.

[50] Z. Xu, M. Kusner, M. Chen, and K. Q. Weinberger, "Cost-sensitive tree of classifiers," in *ICML*, vol. 28, 2013, pp. 133–141.

[51] M. Ye, E. Johns, A. Handa, L. Zhang, P. Pratt, and G.-Z. Yang, "Self-supervised siamese learning on stereo image pairs for depth estimation in robotic surgery," *arXiv preprint arXiv:1705.08260*, 2017.

[52] J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *Journal of Machine Learning Research*, vol. 17, no. 1-32, p. 2, 2016.

[53] N. Zenati and N. Zerhouni, "Dense stereo matching with application to augmented reality," in *Signal processing and communications, 2007. ICSPC 2007. IEEE international conference on*. IEEE, 2007, pp. 1503–1506.

[54] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," *CVPR*, vol. 2, no. 6, p. 7, 2017.