Anytime Stereo Image Depth Estimation on Mobile Devices

Yan Wang\textsuperscript{1}, Zihang Lai\textsuperscript{2}, Gao Huang\textsuperscript{1}, Brian H. Wang\textsuperscript{1}, Laurens van der Maaten\textsuperscript{3}, Mark Campbell\textsuperscript{1}, and Kilian Q. Weinberger\textsuperscript{1}

Abstract—Many real-world 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 can either generate accurate but slow mappings, or fast but inaccurate ones, and typically require far too many parameters for power- or memory-constrained devices. Motivated by this shortcoming, 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. The 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. Our code is available as open source on 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 [28], [49], [37], [40] to augmented reality [51], [1], [33]. 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 cheaper and consume less energy [27]. 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 [26], in recent years the adoption of deep convolutional neural networks (CNN) [50], [30], [19], [23], [34] 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 flip side, 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, only obtains a rate below 0.3FPS on the NVIDIA Jetson TX2 GPU — far too slow for timely obstacle avoidance by drones or other autonomous robots.

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, for example, 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.

Depth estimation with convolutional networks typically scales cubically with the image resolution, and linearly with the maximum disparity that is considered [19]. We leverage these facts and 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 (√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. The correction is obtained by predicting...
the residual error of the up-sampled disparity map from the input images with a CNN. Despite the higher resolution 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 usual range.

Different from most existing multi-scale network structures [38], [14], [18], the successive updates avoid full-range disparity computation at all but the initial low resolution setting and ensure that all computation is re-used. Further, our algorithm can be polled at any time in order to retrieve the current best estimate of the depth map. The outputs span a wide range of possible frame rates (10-35FPS on a TX2 module), while still maintaining 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) [24] 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], [39]. They typically comprise the following four steps: (1) computing the costs of matching image patches over a range of disparities, (2) smoothing of the resulting cost tensor via aggregation methods, (3) estimation of the disparity by finding a low-cost match between the patches in the left image and those in the right image, and (4) refinement of these disparity estimates by introducing global smoothness priors on the disparity map [39], [12], [11], [13]. Several recent papers have studied the use of convolutional networks in step (1). In particular, Zbontar & LeCun [50] 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 [27] and deeper highway networks trained to minimize a multilevel loss [41].

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 [25], optical flow [7], [16], and depth estimation from a single frame [5], several recent studies have explored end-to-end disparity estimation models [30], [19], [23], [34]. For example, in [30], 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 of DispNet include a cascaded refinement procedure [34]. 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 [23]. 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 [52].

Our work is partially inspired by the Geometry and Context Network (GCNet) proposed in [19]. 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. The model we investigate in this study 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 from GCNet in that our model can perform anytime prediction: it produces an initial disparity map prediction very rapidly, and then progressively predict the residual to correct this prediction. LiteFlowNet [15] also tries to predict the residual for optical flow problem. But the main difference is that LiteFlowNet uses the residual to facilitate large-displacement flow inference instead of computation speedup.

c) Anytime prediction: There exists a substantial body of work on machine learned models with computational budget constraints at inference time [44], [10], [17], [47], [48], [45], [14]. Most of these approaches are based on ensembles of decision-tree classifiers [44], [47], [10], [48], which allow for tree-by-tree evaluation: this facilitates the progressive prediction updates that are the hallmark of anytime prediction. Several recent studies have explored anytime prediction with CNN: in particular, they explore image classification networks that dynamically evaluate parts of the network to progressively refine its predictions [22], [14], [43], [46], [31], [6]. Our work differs from these prior anytime CNN models in that we focus on the structured prediction problem of disparity-map estimation, rather than on standard 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 our 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. 3) 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 turned
The original input images are down-sampled through max-pooling or strided convolution and then processed with convolutional filters. Lower resolution feature maps capture local details. At scale 1/8 and 1/4, the final convolutional layer incorporates the previously computed features from the lower scale.

b) Disparity Network: The Disparity Network (Fig. 4) takes as input feature maps from the left and right stereo image and computes a disparity map. We use this architecture in two contexts: to compute the initial disparity map (stage 1) and to compute the residual maps for subsequent corrections (stages 2 & 3). The first step of the disparity network is to compute 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 how well pixel $(i, j)$ of the left image matches the pixel $(i, j-k)$ in the right image and $M$ denotes the maximum disparity under consideration. We can represent each pixel $(i, j)$ in the left image as a vector $p^L_{i,j}$, where dimension $\alpha$ corresponds to the $(i, j)$ entry in the $\alpha^{th}$ input feature map associated with the left image. Similarly we can define $p^R_{i,j}$. The entry $(i, j, k)$ in the cost volume is then defined as the $L_1$ distance between the two vectors $p^L_{i,j}$ and $p^R_{i,j+k}$, i.e., $C_{ijk} = \|p^L_{i,j} - p^R_{i,j+k}\|_1$.

The cost volume constructed in the aforementioned way 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 [19] 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. [19] and compute a weighted average (Disparity regression in Fig. 4)

$$\hat{D}_{ij} = \frac{\sum_{k=0}^{M} k \times \exp\left(-C_{ijk}\right)}{\sum_{k'=0}^{M} \exp\left(-C_{ijk'}\right)}.$$ (1)
Fig. 4: Disparity network. See text for details.

If one disparity $k$ has clearly the lowest cost (i.e. 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 a very low resolution in Stage 1. In Stages 2 & 3 we predict residuals [15]. 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 on KITTI [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 vast 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. Typically (due to the coarses of the low resolution) they are still off by several pixels, which is what we correct by computing residual disparity maps. Prediction of the residual disparity is accomplished in a way similar to what is described above. The only difference is that the cost volume is computed as $C_{ijk} = \|p_{ij} - p_{i(j-k+2)}\|_1$ 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 Stage 4 in which we use a Spatial Propagation Network (SPNet) [24] to refine our disparity predictions. The SPNet sharpens the disparity map by applying a local filter whose weights are themselves predicted with a small CNN from 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 particular, we benchmark the efficiency of our approach on a NVIDIA Jetson TX2 computing module.

a) Implementation Details: We implement AnyNet in PyTorch [35]. 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 $\pm 2$, corresponding to $\pm 16$ pixels in Stages 2 and $\pm 8$ 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 [20], and two orders of magnitude fewer than PSMNet [4]. Our model is trained using Adam [21] with initial learning rate $5e^{-4}$ and batch size 6. On the Scene Flow dataset [30], 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 took 3.5 hours on Scene Flow and 30 minutes on KITTI. All results are averaged over five randomized 80/20 train/validation splits.

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.

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.
b) **Datasets:** Our model is trained on the synthetic Scene Flow [30] dataset and evaluated on two real-world datasets, KITTI-2012 [8] and KITTI-2015 [32]. The Scene Flow dataset contains 22,000 stereo image pairs for training, and 4,370 image pairs for testing. Each image has a resolution of $960 \times 540$ pixels. As in [19], we train our model on $512 \times 256$ 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 $1242 \times 375$.

c) **Baselines:** Although state-of-the-art CNN based stereo estimation methods have been reported to reach 60FPS on a TITAN X GPU [20], 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 [20], DispNet [30], and StereoDNN [42]. 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 and Semi-Global Block Matching, supported by OpenCV [3].

In order to collect meaningful results for these baseline methods on the TX2, we use down-sampled input images for faster inference times. The baseline methods are re-implemented, and trained on down-sampled stereo images - this is to allow 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.

![Fig. 5: (a)-(d) Disparity prediction from 4 stages of AnyNet in KITTI-2015. The prediction gets refined and become more accurate as larger computational budget is consumed. (e) is the ground truth LiDAR image. (f) in the left input image.](image)

Table II contains numerical results for AnyNet on the KITTI-2012 and KITTI-2015 datasets. Lower values are better.

| Dataset   | Stage 1 | Stage 2 | Stage 3 | Stage 4 |
|-----------|---------|---------|---------|---------|
| 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 |

**A. Evaluation Results**

Table II contains numerical results for AnyNet on the KITTI-2012 and KITTI-2015 datasets. Additionally, Figures 6a and 6b demonstrate the evaluation error and inference time of our model as compared to baseline methods. Baseline algorithm results originally reported in [3], [20], [4], [29], [42] 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 ≥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 larger 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) is much shorter than for any of the baselines, reducing the amount of wasted computation if a query is issued during a disparity map computation.
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.

B. Ablation Study

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

As far as we know AnyNet is the first algorithm for any-time depth estimation from stereo images. As (low-power) GPUs become cheaper and are increasingly incorporated into mobile computing devices we hope that accurate and reliable real-time depth estimation could become readily available for a large variety of robotic applications. Our code is available as open source on https://github.com/mileyan/AnyNet.

ACKNOWLEDGMENT

This research was supported in part by the III-1618134, III-1526012, IIS-1149882, IIS-1724282, and TRIPODS-
[50] 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.

[51] 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.

[52] 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.