Wavelet Synthesis Net for Disparity Estimation to Synthesize DSLR Calibre Bokeh Effect on Smartphones

Chenchi Luo*, Yingmao Li*, Kaimo Lin*, George Chen*,
Seok-Jun Lee*, Jihwan Choi+, Youngjun Francis Yoo*, Michael O. Polley*
*Samsung Research America
+Samsung Electronics
{eric.luo, timothy.li, kaimo.lin, george.chen}@samsung.com

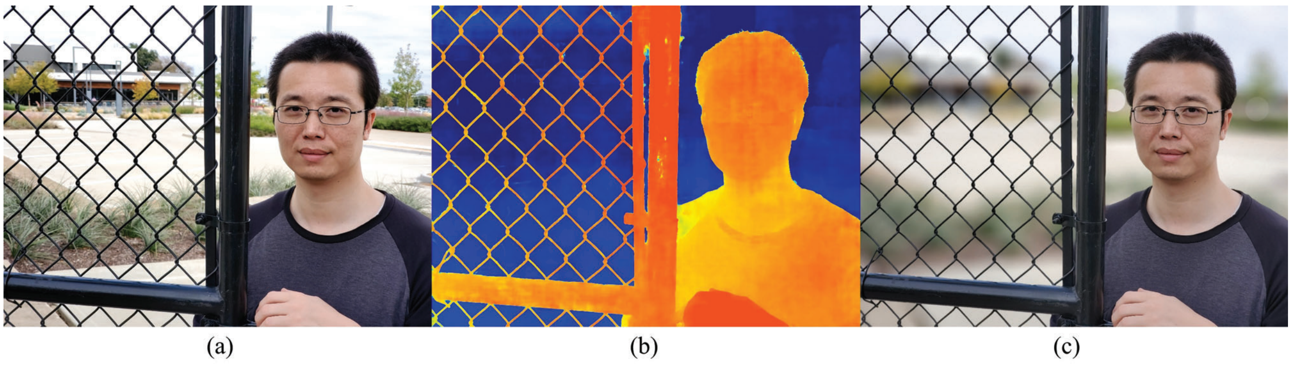

Figure 1. Example of our disparity map and rendered bokeh: (a) All-in-focus image. (b) Disparity map. (c) Rendered bokeh.

Abstract

Modern smartphone cameras can match traditional digital single lens reflex (DSLR) cameras in many areas thanks to the introduction of camera arrays and multi-frame processing. Among all types of DSLR effects, the narrow depth of field (DoF) or so called bokeh probably arouses most interest. Today's smartphones try to overcome the physical lens and sensor limitations by introducing computational methods that utilize a depth map to synthesize the narrow DoF effect from all-in-focus images. However, a high quality depth map remains to be the key differentiator between computational bokeh and DSLR optical bokeh. Empowered by a novel wavelet synthesis network architecture, we have narrowed the gap between DSLR and smartphone camera in terms of bokeh more than ever before. We describe three key enablers of our bokeh solution: a synthetic graphics engine to generate training data with precisely prescribed characteristics that match the real smartphone captures, a novel wavelet synthesis neural network (WSN) architecture to produce unprecedented high definition disparity map promptly on smartphones, and a new evaluation metric to quantify the quality of the disparity map for real images from the bokeh rendering perspective. Experimental results show that the disparity map produced from our neural network achieves much better accuracy than the other state-of-the-art CNN based algorithms. Combining the high resolution disparity map with our rendering algorithm, we demonstrate visually superior bokeh pictures compared with existing top rated flagship smartphones listed on the DXOMARK mobiles.

1. Introduction

One of the major differences between a smartphone camera and a DSLR camera is the shallow depth of field effect, which forms the aesthetic blur in the out-of-focus part of an image. Such an effect can be physically achieved by using a lens with a large focal length and/or a large aperture size (e.g., an 85mm f1.8) [1]. Due to the limitations on size and weight, a smartphone camera cannot generate the same bokeh as DSLR can do. Most smartphone makers break this limitation using computational methods to simulate bokeh using the depth information obtained from either dual pixel camera or stereo cameras [2, 3, 4]. These kinds of photo capture modes are typically referred as “Portrait”, “Aperture” or “Live Focus” mode in smartphones. Despite all the efforts, a high resolution and high quality depth map remains the key bottleneck to synthesize DSLR calibre bokeh images.

Although time of flight (ToF) or structured light depth sensors become increasingly more ubiquitous in smartphones, they are suffering from either low resolution or high susceptibility to ambient lighting interferences. On the other hand, mono camera depth algorithms [5, 6, 7, 8] only work for specific scenes and are not general enough to handle all scenarios. Stereo depth [9, 10, 11, 12] remains the most mature solution for smartphones. Despite the hardware and algorithm advances with AI, today’s top ranked smartphones in the DXOMARK mobiles are still left behind by DSLRs by a large margin due to the inferior depth quality. The purpose of this paper is to elevate the stereo disparity estimation to a new level so that smartphones can be a legitimate challenger to DSLRs in terms of bokeh effect.

The flowchart of our computational bokeh pipeline can be seen in Fig. 2. It contains three key blocks shown in dark blue. The calibration module is responsible for rectifying the pair of heterogeneous cameras with different field of view (FOV), focus, etc. The WSN produces the disparity map given the calibrated stereo pair. The renderer produces the final bokeh image given the focus point location and the disparity map. The input image pair comes from the main camera which produces the reference frame for bokeh and the sub camera which has a larger FOV. Typically, tele/wide lens or wide/ultra wide lens combination on smartphones constitutes the main/sub camera combination. Fig. 1 shows an example of our bokeh output.

The scope of this paper is focused on the WSN disparity estimation module as it is the most critical component to produce DSLR calibre bokeh image. The calibration and bokeh rendering modules are standard procedures, which are not covered in this paper. Due to the smartphone cameras’ characteristics such as optical imaging stabilizer (OIS), auto focus, and lens distortion, it is extremely difficult to achieve perfect calibration. For this reason, we formulate the disparity estimation problem as a constrained optical flow problem to better accommodate the smartphone camera characteristics. We ignore the orthogonal baseline direction flow and treat the baseline direction flow as the disparity output.

However, general high performing CNN based optical flow algorithms [13, 14, 15, 16] represented by the FlowNet framework are too complex to fit into smartphones. A commonality of the family of these networks is that they are trained on the same existing public datasets such as KITTI, cityscape, flying chairs/things, and mpiSintel [17, 13, 18]. As a result, none of them is able to produce high quality disparity map on real images captured by smartphones in the presence of small baseline and heterogeneous lens/ISP settings.

Our first contribution is that we create our own synthetic graphics data engine and the corresponding data generation methodology to produce high quality training data that can be tailored for any target mobile device. We propose a novel 3 stage training recipe to bridge the gap between synthetic image training and real image training with learnable image augmentation.

Our second contribution is the proposal of an efficient wavelet synthesis network architecture for disparity estimation. Even though the light weight LiteFlowNet [16] runs for 35.83ms on nvidia GTX 1080 for a resolution of 1024×436, it is still very difficult to port it onto the smartphone at a even higher resolution of 2048×1536, which is required to render DSLR calibre bokeh. By comparison, we have benchmarked the WSN for 1.8s on Qualcomm Snapdragon 855’s mobile GPU at the 2048×1536 input/output resolution.

To further reduce the complexity and improve the training convergence speed, we introduce two new types of layers in WSN, i.e., the invertible wavelet layer and the normalized correlation layer. The invertible wavelet layer is applied to iteratively decompose and synthesize the feature maps. The normalized correlation layer is introduced for robust dense feature map matching and is highly coupled with the smartphone specifications including baseline distance and calibration accuracy. With the novel network architecture and our hybrid synthetic data training methodology, we obtain the state-of-the-art disparity map in terms of quality, resolution, and runtime on smartphone.

Our third contribution is the introduction of a new evaluation metric called max IoU (mxIoU) to quantify the capability of the disparity map to segment the foreground object from the background for real images. We compare our neural network with existing state-of-the-art methods on smartphone captured images. The quantitative experiment results show that our method produces the best disparity map for smartphone cameras. We also compare our rendered bokeh images with the those produced by the top ranked flagship smartphones on the DXOMARK mobiles and our results are visually superior to all existing solutions.

The rest of this paper is organized as follows. In section 2, we disclose the details of WSN for disparity estimation, including the two new types of layers. We show details of our synthetic graphic engine as well as the experimental results in Section 3 and Section 4. We draw conclusions in Section 5.

2. Wavelet Synthesis Net

The network topology and layer details can be seen in Fig. 3. We have a pair of calibrated stereo image inputs L

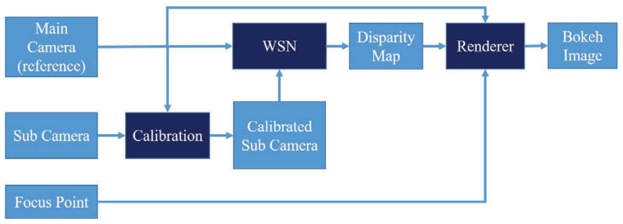

Figure 2. Bokeh algorithm flow chart. The sub camera is calibrated to the match the FOV of the main camera. WSN calculates the disparity with regard to the main camera. The renderer generates the bokeh image given the chosen focus point.

and R, where L is the reference image from the main camera where the bokeh effect is applied upon and R is the calibrated image captured from the sub camera. The output of the network is the optical flow from L to R. In the end, we ignore the flow in the orthogonal baseline direction and treat the baseline direction flow as the disparity output.

The network is composed of three major components sequentially: the feature encoder that extracts high level features from the input image pair, the normalized correlation layer that calculates the cross-correlation between the left and right feature maps, and the feature decoder that gradually complements the details in the output. Compared with prior arts in the FlowNet framework, the proposed WSN has two unique layers, i.e., the invertible wavelet layer and the normalized correlation layer.

All the convolution modules in the network have the same spatial resolution for their input and output feature maps. One convolution module contains one or more convolutional blocks. Each convolution block follows the micro-architecture in MobileNet [19, 20].

Before reaching the normalized correlation layer, the spatial resolution of the feature maps is reduced by a factor of eight by means of the wavelet layer. The detail feature maps in early stages are preserved to be synthesized by means of the inverse wavelet layer with the feature maps in later stages to restore the spatial resolution of the output. A detailed layer level topology of the network is available in the supplementary materials.

2.1. Invertible Wavelet Layer

In convolutional neural networks, pooling layers are needed to increase the receptive field of the feature extractor multiplicatively. However, the drawback of the pooling layer is that it introduces information loss. For example, for a 2×2 pooling layer, 75% of the detail information is discarded. However, for pixel-to-pixel applications such as semantic segmentation, disparity or optical flow estimation, the output resolution is typically the same as the input resolution. In these applications, we need more detail information passing through the network. For this reason, UNet [21, 22, 23, 24] like architectures are widely used to feed forward low level feature maps through the skip branches.

In this paper, we propose a more elegant way to achieve both spatial resolution reduction and information preservation with discrete wavelet and inverse wavelet transforms. As we know, wavelet transforms are perfectly invertible and they achieve the same spatial resolution reduction effect as the pooling layer without any information loss. Fig. 4 (a) demonstrates the idea of the proposed invertible wavelet layers. We apply a 2D wavelet transform to decompose a 2D image into four quadrants: LL, HL, LH, HH, where LL represents the low frequency component or the average information of the 2D image while, HL, LH, HH represent the high frequency components or the details in the 2D image. We stack the HL, LH, HH in the channel dimension to form a new feature map. This basic idea can be extended to 3D feature maps of dimension (H,W,C), the wavelet layer produces two feature maps of dimension (H/2,W/2,C) and (H/2,W/2,3C) that represent the average and detail information respectively. These two types of feature maps should be treated differently in the neural network. The network's main branch should iteratively process the average feature maps in order to have a global context understanding of the image without the interference of local details. At the same time, the detail feature maps are responsible for restoring the spatial resolution of the output. Naturally, the way we restore the spatial resolution of the network is by means of the inverse wavelet transform, which is also a lossless process. The invertible discrete wavelet layers have two main advantages. Firstly, they are linear transformations with $O(NlogN)$ complexity and are differentiable for end-to-end training. For discrete Haar wavelet, only addition and subtraction operations are needed. Secondly, the transforms are invertible so that no detail information is lost in this layer.

2.2. Normalized Correlation Layer

The correlation layer is introduced in the FlowNet-C architecture in [13]. This paper offers several improvements as shown in Fig. 4 (b) to make it work most effectively for our application. The original correlation layer is introduced to solve optical flow problem. Therefore, the search window size is large and symmetric in all directions. However, for the calibrated image pair on the smartphone, we understand the error margin of our calibration algorithm, the focused object range (0.5m-2.5m), and the baseline distance (1cm). Therefore we can precisely prescribe the asymmetric search window size to reduce the possibility of underfitting or over-fitting. Due to the limitation of the smartphone camera settings, our search window also needs to cover a small range in the dx+,dy+,dy- direction, where dx- is the baseline direction, instead of only doing a linear search along dx- when the calibration is done perfectly. For input image size of 2048×1536, our search window

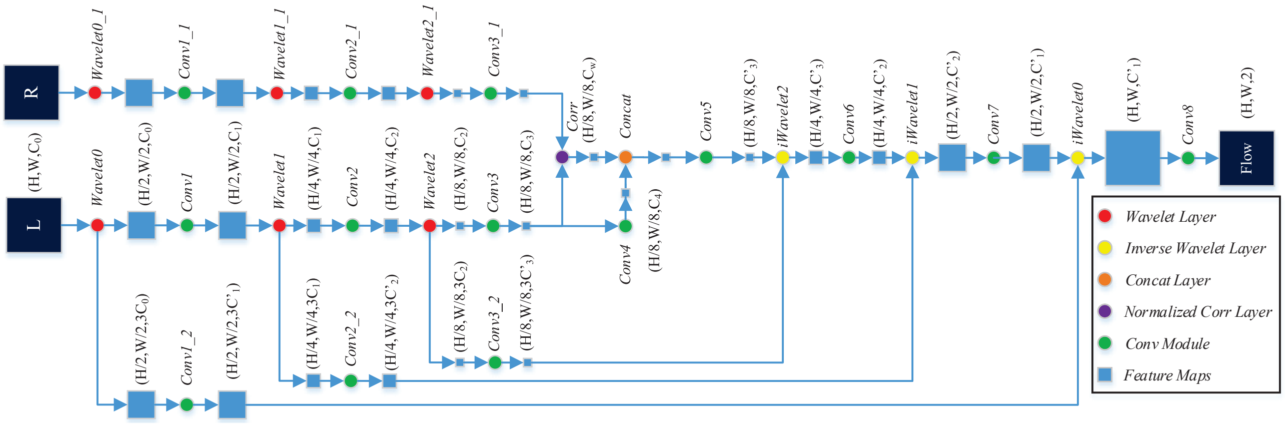

Figure 3. WSN network topology. L stands for the main camera image. R stands for the calibrated sub camera image.

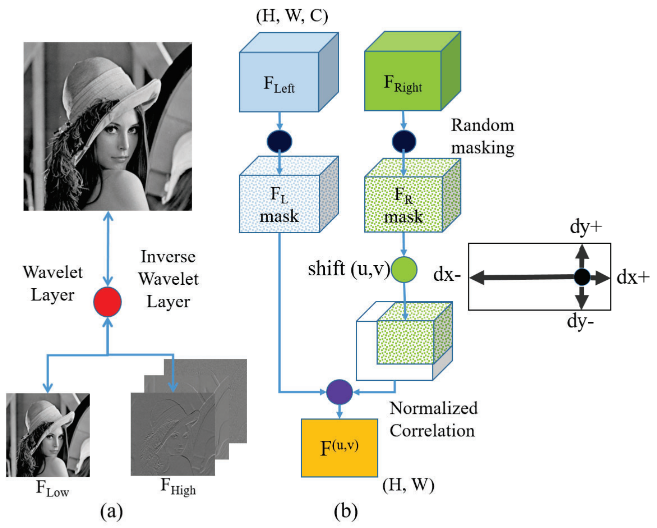

Figure 4. (a) Invertible wavelet layers. The wavelet layer decomposes feature map of dimension (H,W,C) into the low frequency feature map F_{Low} (H/2, W/2, C) and the high frequency feature map F_{High} (H/2, W/2, 3C). The inverse wavelet layer synthesizes the original feature map from the low and high frequency feature maps. (b) Normalized correlation layer. F_{Left} and F_{Right} stand for the feature map in the main and sub camera image branches respectively. Each channel in the output feature map corresponds to the normalized correlation between F_{Left} and a shifted F_{Right} with direction (u,v) in the search window.

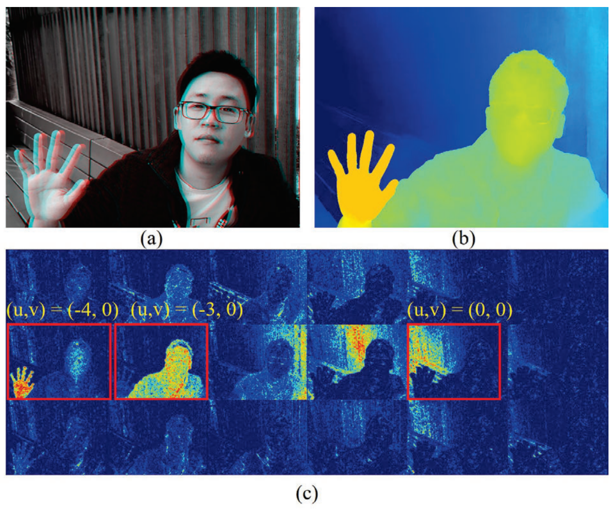

Figure 5. Normalized correlation layer visualization (a) Calibrated and overlaid stereo input image pair (b) Final disparity estimation (c) Stacked view of the normalized correlation layer feature maps output

is $u \in [-20, 4]$ in baseline direction and $v \in [-4, 4]$ in the orthogonal baseline direction. The second improvement is the feature maps preprocessing. We apply independent and random masking or dropout operations to the left and right feature maps before applying the correlation operation to simulate the situation that some features in one feature map are not visible in the other. This enforces the network to infer the matching based on the context. In our experiments, we find that better result is achieved when starting with a high dropout rate to avoid overfitting and reducing the dropout rate gradually to improve disparity detail refinement. The third improvement is the normalization step to make sure the output feature maps are constrained in [0,1] to improve training convergence and stability. For a search direction (u, v) in the search window, we have

$$F_o^{(u,v)} = \frac{< F_L, F_R^{(u,v)} >_c}{\epsilon + \text{var}_c(F_L)\text{var}_c(F_R^{(u,v)})} \tag{1}$$

$$< F_L, F_R^{(u,v)} >_c (i,j) = \sum_{k=0}^{C-1}[F_L(i,j,k) - \overline{F_L(i,j,:)}]$$
$$[F_R(i-u, j-v, k) - \overline{F_R(i-u, j-v, :)}]$$

where $F_o^{(u,v)}$ is the 2D output feature map, F_L and $F_R^{(u,v)}$ are the left and shifted right 3D input feature maps, $\text{var}_c(\cdot)$

and $\overline{(\cdot)}$ stands for the variance and the mean of the feature map over the channel dimension and $\epsilon = 10^{-6}$ is to avoid the divide by zero operation. We keep the above process for all the directions (u, v) inside the search window and stack the 2D feature maps $F_o^{(u,v)}$ along the channel dimension to formulate the 3D feature maps output of the proposed normalized correlation layer.

According to our experiments, the normalized correlation layer significantly improves the numerical stability and convergence speed of the network during the training process. Fig. 5 shows the visualization of the feature maps produced by the proposed normalized correlation layer. The 3D feature maps are re-stacked in Fig.5 (c) to show their responses towards different shift (u, v). As the figure shows, for $(u, v) = (0, 0)$, the network responds to the farthest background (disparity = 0). As the horizontal shift u increases gradually, the network responds to the background fence, body and hand orderly. However, the spatial resolution of the feature maps after the normalized correlation layer is only 1/8 of the input resolution. We still need to rely on the detail feature maps in early stages to restore the spatial resolution of the output.

3. Data and Training

3.1. Synthetic Training Data

Getting a pixel-to-pixel ground truth for disparity map is considered as a hard problem in real world scenario. Existing depth sensors such as ToF cameras , LiDAR, are not able to produce a perfect pixel level depth map. The performance of these depth sensors is limited due to ambient light, occlusion, sensor noise, and reflective materials. Another difficulty is that we still need to align the depth ground truth with the camera images. This adds even more uncertainty to the ground truth.

In this paper we carefully design our synthetic training data to match the statistics of the smartphone camera working scenarios. Prior arts [13] use computer graphics software and game engines to generate the synthetic training data for the virtual stereo cameras. Our literature search suggests that almost all of the previous works [13, 14, 15, 16] are trained on similar public datasets such as flying things, flying chairs, MPISintels, KITTI, etc. However, when testing on smartphone captured images taken from daily life using these published works, the performance is not as good as what is reported. Our investigation shows that the root cause is that the disaprity histogram of real portrait images in our bokeh application is distributed in [0, 60] pixel range with a probability density function (PDF) similar to the exponential distribution while the histogram of the FlyingThings dataset is distributed in the [0, 150] pixel range with a very different shaped PDF.

To accommodate this issue, we build our own training

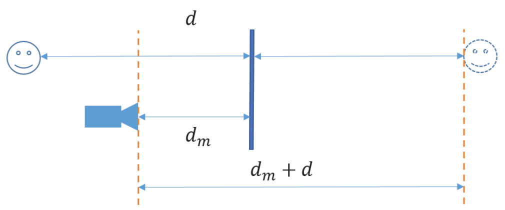

Figure 6. Illustration of the difference between the depth based disparity and the true optical disparity.

data with the Unreal Engine [25]. We setup a virtual 3D space with simulated depth and cameras with known intrinsic and extrinsic parameters. The virtual objects in the 3D space are randomly placed in certain depth ranges and rendered with randomly generated textures. We generate the disparity ground truth directly from the depth. Our data is produced with the guidance of the following methodologies.

Photo-Realism. Intuitively, people may believe that a photo realistic synthetic training data should result in a better network performance. However, we surprisingly find out that the network converges much slower and results in worse performance using photo-realistic training data instead of using the training data with the minimum rendering quality. It turns out the reflection option in the rendering engine is the culprit. As illustrated in Fig. 6, the physical distance between the reflective surface (e.g., a mirror) and the camera is d_m. Thus the disparity of the object's image in the mirror is derived from d_m in the rendering engine. However, in the real world, the disparity of the object's image in the mirror corresponds to the depth of the object's image, which is $d_m + d$. With the reflection option turned on, we are actually teaching the network the wrong disparity to learn in that case.

Another interesting finding is that the network performance is not related to the semantics of the training data, which is aligned with the observation reported by [13]. The virtual object in the training data can be anything, even if it does not have any semantic meanings. For example, a photo realistic human model is not required for the network to perform well for human. What the network learns is just matching using a global context. Therefore, our training data is rendered without any photo-realistic effect turned on in the rendering engine to avoid any possible confusion.

Pattern Ambiguity. The capability to customize our training data enables us to control the response of the network to produce desired results without depth artifacts. Traditional stereo algorithms struggle at homogeneous regions such as feature-less planar or repetitive patterns because they do not have a global context understanding of the image. With the advent of CNNs, we can potentially achieve a very large receptive field so that they can better match the dense image features. To take advantage of CNNs' potentials from this perspective, we introduce a lot of ambiguous

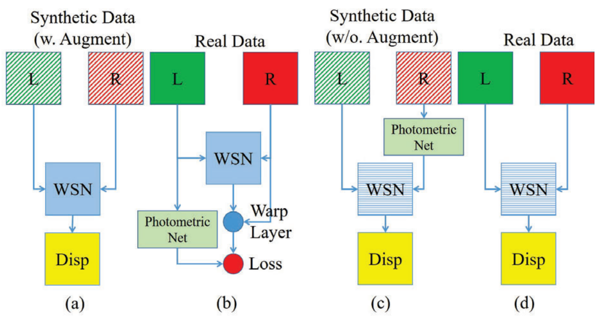

Figure 7. Iterative training recipe of WSN. (a) Stage 1: synthetic image training with shared feature encoder. (b) Stage 2: real image training to learn the photometric transform between the stereo image pair. (c) Stage 3: synthetic image training with independent feature encoders. (d) The inference network structure.

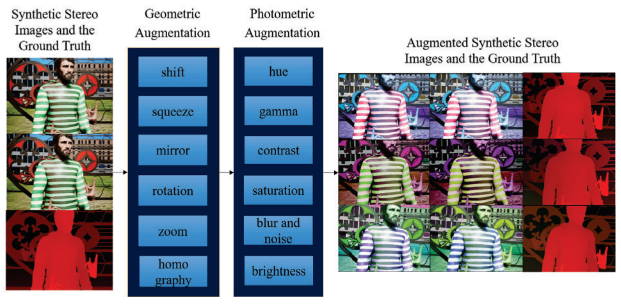

Figure 8. Data augmentation pipeline and example effects.

patterns and textures randomly applied to the 3D objects.

3.2. Training Details

Our network is trained on both the synthetic data with data augmentation and real smartphone captured images. The data augmentation module needs to reflect both the geometric calibration imperfection and the photometric mapping between the stereo cameras on smartphones. To achieve this, we use a three stage hybrid training methodology as shown in Fig. 7.

The First Stage. In the first stage of training, the WSN is only trained with our synthetic data. The weights between the two stereo image encoder branches are shared during this process. We have generated 100K pairs of synthesized data with 1024×768 resolution, and use 95K pairs of the images for training, and the rest of them for validation. We pass the training data through our augmentation pipeline, as shown in Fig. 8. The augmentation pipeline contains two modules, i.e., photometric augmentation and geometric augmentation. The augmentation applies random perturbation on left and right images independently. The photometric augmentation module applies random blurriness, chroma, illumination, gamma and noise to the stereo pair, such that the network is robust against photometric discrepancies of the training images. The geometric augmentation applies random zoom, rotate, homogeneous distortion, skew, and crop to the training images.

The training minimizes the l_2 end point error (EPE) loss. The learning rate starts from 1×10^{-3} with the ADAM solver. We apply dropout [26] layer right before the normalized correlation layer. The initial drop out rate is 0.25 and we reduce the drop out rate by 0.05 every 20 epochs until 0. During the training, we reduce the learning rate by a factor of two for every 40 epochs, and the entire training takes about 400 epochs to achieve a satisfactory result.

The Second Stage. The purpose of this stage is to learn the photometric mapping between the two cameras. The source of the photometric discrepancies originates from the fact that the smartphone stereo cameras usually have different lenses and the ISPs have different settings and tuning for the two cameras. Our goal is to train a small network such that the photometric difference between the two heterogeneous camera captured images is minimized. As a result, the performance of WSN can be further optimized for the specific camera platform. We build a light-weight three layer fully convolutional network which is called the "PhotometricNet" as seen in Fig. 7 (b). We use 800 calibrated stereo image pairs from our target smartphone as our training data. We denote the stereo images from left (main) and right (sub) camera as $\mathbf{I}_L$ and $\mathbf{I}_R$, respectively. During the training process, the weights of WSN remain locked. We pass the training data through WSN and warp the image from the right camera $\mathbf{I}_R$ using the predicted flow as $\hat{\mathbf{I}}_L$, then we use $\mathbf{I}_L$ as the input to the PhotometricNet and $\hat{\mathbf{I}}_L$ as the label during the training. The training process starts with the learning rate 1×10^{-5} and minimize the l_2 loss between the network output and $\hat{\mathbf{I}}_L$ with the ADAM optimizer. The training takes about 30 epochs to converge, and we lower the learning rate by half every 10 epochs. As a result, the PhotometricNet learns the photometric mapping from the main camera $\mathbf{I}_L$ to the sub camera $\mathbf{I}_R$.

The third Stage. As seen in Fig. 7 (c), in the third training stage, we apply the PhotometricNet to sub camera $\mathbf{I}_R$ before passing it to WSN. We disable the photometric augmentation and only use geometric augmentation in our pipeline. In this way, we make sure that the main and augmented sub camera synthetic images fed into WSN have the same relative photometric characteristics as the real main/sub camera images. In this stage, we also disable the weights sharing between the left and right encoders to fine tune WSN. As a result, WSN learns to handle the photometric differences in the real world images in the optimal way. We train the WSN with our synthetic data, starting with a learning rate of 1×10^{-5} for 200 epochs, and then we lower the learning rate by half after every 50 epochs.

Ideally, we should repeat the second and third stages for multiple iterations to make sure that both the WSN and

Table 1. Quantitative algorithmic comparison on both real and synthetic images.

Algorithm	EPE	mean mxIoU	median mxIoU
LiteFlowNet	3.398	0.86837	0.90511
PWCNet	3.603	0.87884	0.89758
FlowNet2	2.878	0.91155	0.93914
FlowNetCSS	3.329	0.88048	0.92008
FlowNetCS	3.364	0.88648	0.91847
FlowNetC	4.056	0.84481	0.86377
FlowNetS	4.408	0.80402	0.81482
WSN	**0.586**	**0.95221**	**0.98133**

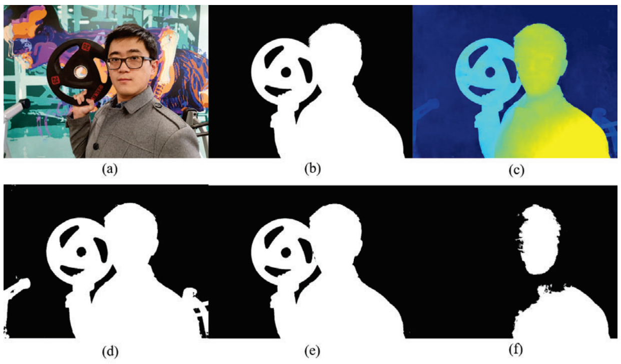

Figure 9. mxIoU calculation example. (a) Reference image. (b) Ground truth foreground mask. (c) Disparity map from WSN. (d) The foreground mask that corresponds to the disparity threshold of $d = 7.0$. (e) The foreground mask that corresponds to the disparity threshold of $d = 22.0$. (f) The foreground mask that corresponds to the disparity threshold of $d = 30.0$.

the PhotometricNet are optimal. However, our experiment shows that one iteration is usually sufficient to reach satisfactory result. In the inference stage as shown in Fig. 7 (d), the real world stereo images are passed directly to WSN to predict the disparity.

4. Experiment Results

To test the performance of WSN, we conduct three types of evaluation experiments.

Quantitative Evaluation on Synthetic Images. To better simulate the disparity statistics from the images captured with a real smartphone, we conduct a quantitative disparity evaluation on our own synthetic dataset by calculating the standard EPE scores. We compare our WSN against the other state-of-the-art CNN based optical flow estimation algorithms such as LiteFlowNet, PWCNet , FlowNet2, and FlowNet2-CSS [14, 15, 16], etc. To make a fair comparison, we fine-tune the other state-of-the-arts networks using our synthetic data by following the methodologies stated in the original paper. We make a dedicated validation synthetic dataset that is not trained by any algorithm to evaluate the EPE scores for all the algorithms and the result is summarized in Table. 1. Our WSN produces a significantly lower EPE against all other methods.

Quantitative Evaluation on Real Images. A lower EPE on synthetic images does not necessarily indicate a better performance of the network on real world images. Generally speaking, it is very difficult to obtain the ground truth disparity for each pixel for any given general real world captured picture. To solve this problem, we introduce a new metric called max IoU (mxIoU) to quantitatively evaluate the disparity quality in the sense of the capability to separate a given subject from the background.

The metric of intersection of union (IoU) [27] is commonly used to evaluate the accuracy of a segmentation mask. The key step in bokeh image rendering is also to segment the focused object from the background.

In our work, we borrow the similar idea. For a given portrait picture, we choose the subject that stands out from its background, and manually label the subject to create the ground truth mask $\mathbf{M}_f$ as seen in Fig. 9 (b). To evaluate the disparity map $\mathbf{D}$, we enumerate all the disparity values within the disparity map, as illustrated in Fig. 9 (d-f). For each selected disparity d, we place threshold on $\mathbf{D}$ to obtain a foreground mask $\mathbf{M}_d$ such that all the pixels covered by the mask have disparities larger than or equivalent to d. At the same time, we compute the IoU score between $\mathbf{M}_d$ and $\mathbf{M}_f$. The mxIoU score is the maximum IoU score we obtain for a given image. In our experiment, we take 200 portrait pictures using a Samsung Galaxy Note 10+, and hire photoshop professionals to manually label the subjects that clearly stand out from their background as the ground truth masks. The mean and median mxIoU scores from different algorithms can be seen in Table. 1. We visualize the disparity map from each algorithm as seen in Fig. 10. Compared with the existing state-of-the-arts, the disparity map generated from our algorithm shows visually superior quality in addition to a higher mxIoU score. WSN shows significant advantage in terms of details and ambiguous pattern handling over the other algorithms for real images.

Qualitative Evaluation of Bokeh Artifacts. Furthermore, we apply the WSN in our bokeh pipeline shown in Fig. 2 and compare our bokeh quality against four top ranked flagship smartphones listed on the DXOMARK mobiles [1], i.e., Huawei Mate30 Pro, Samsung Galaxy Note 10+, iPhone 11 Pro Max, and Google Pixel 4. Due to the page limit, the sample testing images can be seen in the supplementary materials. We have labeled the depth artifacts in red in these sample images. Our bokeh images have the best detail and the least amount of depth artifacts compared with the existing bokeh solutions in the market.

5. Conclusion

We have introduced a new network architecture, WSN, for disparity estimation and we have described the detailed methodologies for synthetic training data prepara-

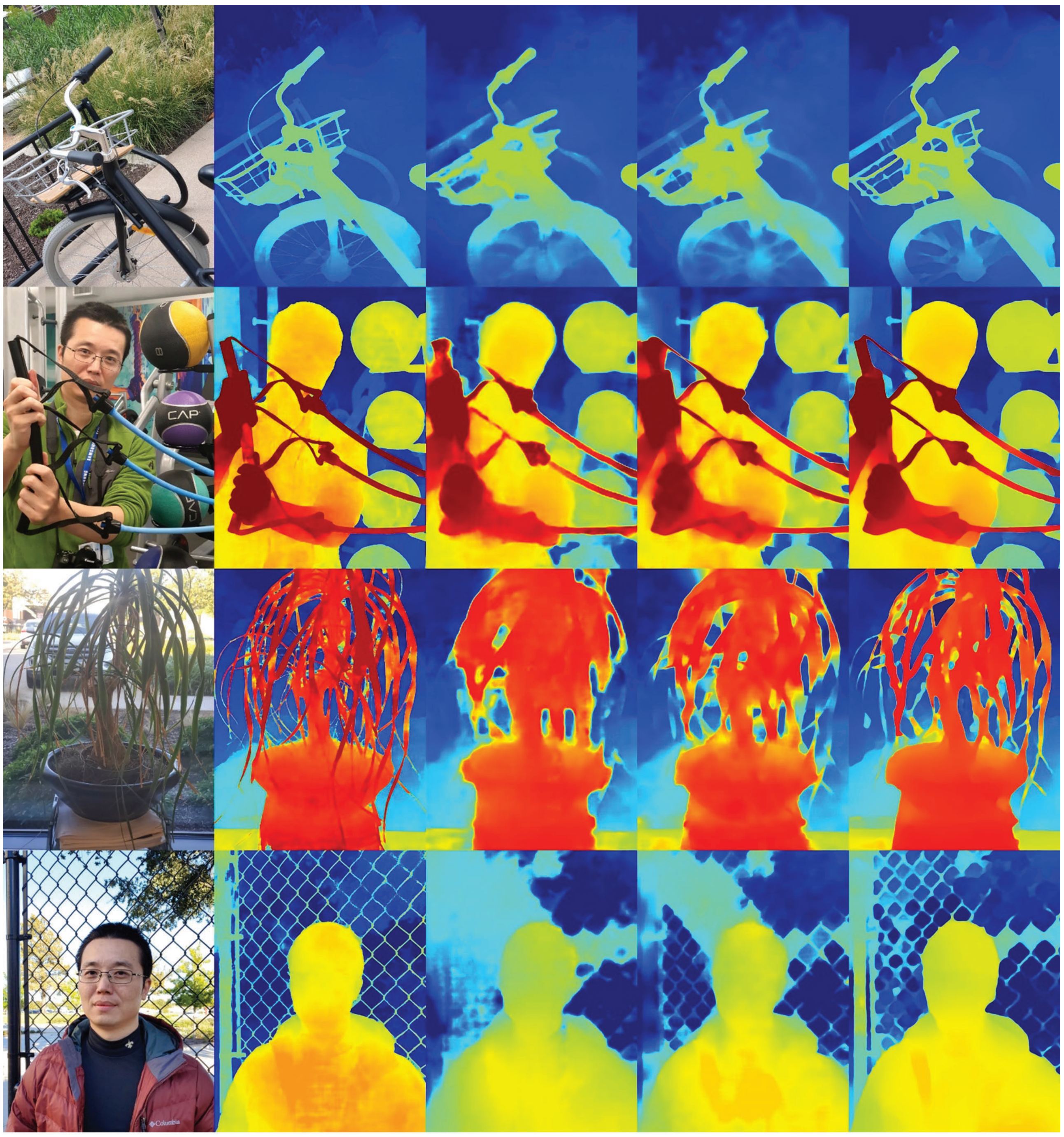

Figure 10. Disparity map comparison examples. (a) Reference image. (b) WSN. (c) PWCNet. (d) LiteFlowNet. (e) FlowNet2.

tion and the multi-stage network training for the heterogeneous smartphone stereo camera setups. We have demonstrated that our method outperforms all existing state-of-the-art methods by a large margin using both synthetic data and real world data, quantitatively and visually. With the superior disparity quality, we have shown that our rendered bokeh images are much better than top ranked flagship smartphones in terms of depth artifacts and details so that smartphones can truely produce DSLR calibre bokeh.

References

[1] W. Hauser, B. Neveu, J.-B. Jourdain, C. Viard, and F. Guichard, "Image quality benchmark of computational bokeh," *Electronic Imaging*, vol. 2018, no. 12, pp. 340–1, 2018.

[2] H. Lin, C. Chen, S. Bing Kang, and J. Yu, "Depth recovery from light field using focal stack symmetry," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3451–3459, 2015.

[3] N. Wadhwa, R. Garg, D. E. Jacobs, B. E. Feldman, N. Kanazawa, R. Carroll, Y. Movshovitz-Attias, J. T. Barron, Y. Pritch, and M. Levoy, "Synthetic depth-of-field with a single-camera mobile phone," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 64, 2018.

[4] D. Liu, R. Nicolescu, and R. Klette, "Bokeh effects based on stereo vision," in *International Conference on Computer Analysis of Images and Patterns*, pp. 198–210, Springer, 2015.

[5] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, pp. 2366–2374, 2014.

[6] N. Wadhwa, R. Garg, D. E. Jacobs, B. E. Feldman, N. Kanazawa, R. Carroll, Y. Movshovitz-Attias, J. T. Barron, Y. Pritch, and M. Levoy, "Synthetic depth-of-field with a single-camera mobile phone," *ACM Transactions on Graphics (TOG)*, vol. 37, no. 4, p. 64, 2018.

[7] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *European Conference on Computer Vision*, pp. 740–756, Springer, 2016.

[8] B. Liu, S. Gould, and D. Koller, "Single image depth estimation from predicted semantic labels," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1253–1260, IEEE, 2010.

[9] J. Zbontar, Y. LeCun, *et al.*, "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.

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

[11] M. Poggi, D. Pallotti, F. Tosi, and S. Mattoccia, "Guided stereo matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 979–988, 2019.

[12] Z. Liang, Y. Feng, Y. Guo, H. Liu, W. Chen, L. Qiao, L. Zhou, and J. Zhang, "Learning for disparity estimation through feature constancy," 2018.

[13] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. Van Der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766, 2015.

[14] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, "Flownet 2.0: Evolution of optical flow estimation with deep networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2462–2470, 2017.

[15] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2881–2890, 2017.

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

[17] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[18] D. Butler, J. Wulff, G. Stanley, and M. Black, "Mpi-sintel optical flow benchmark: Supplemental material," in *MPI-IS-TR-006, MPI for Intelligent Systems (2012*, Citeseer, 2012.

[19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 2018.

[21] 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*, pp. 234–241, Springer, 2015.

[22] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3d u-net: learning dense volumetric segmentation from sparse annotation," in *International conference on medical image computing and computer-assisted intervention*, pp. 424–432, Springer, 2016.

[23] Z. Zhang, Q. Liu, and Y. Wang, "Road extraction by deep residual u-net," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 5, pp. 749–753, 2018.

[24] A. Jansson, E. Humphrey, N. Montecchio, R. Bittner, A. Kumar, and T. Weyde, "Singing voice separation with deep u-net convolutional networks," 2017.

[25] W. Qiu and A. Yuille, "Unrealcv: Connecting computer vision to unreal engine," in *European Conference on Computer Vision*, pp. 909–916, Springer, 2016.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[27] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.