Depth Estimation and Blur Removal from a Single Out-of-focus Image

Saeed Anwar*
Saeed.Anwar@anu.edu.au
Zeeshan Hayder*
Zeeshan.Hayder@anu.edu.au
Fatih Porikli
Fatih.Porikli@anu.edu.au

The Australian National University,
Canberra, Australia

Abstract

This paper presents a depth estimation method that leverages rich representations learned from cascaded convolutional and fully connected neural networks operating on a patch-pooled set of feature maps. Our method is very fast and it substantially improves depth accuracy over the state-of-the-art alternatives, and from this, we computationally reconstruct an all-focus image and achieve synthetic re-focusing, all from a single image. Our experiments on benchmark datasets such as Make3D and NYU-v2 demonstrate superior performance in comparison to other available depth estimation methods by reducing the root-mean-squared error by **57%** & **46%**, and blur removal methods by **0.36 dB** & **0.72 dB** in PSNR, respectively. This improvement is also demonstrated by the superior performance using real defocus images.

1 Introduction

Recovering original image from its defocus version has attracted much attention and recently numerous techniques have been put forward. Due to the inherent information loss, this reconstruction task requires strong prior knowledge or multiple observations to produce effective results. For example, deconvolution with natural image priors [5, 8, 15, 20, 35], hybrid cameras [22, 30, 40] and blurred/noisy image pairs [46] are among the notable solutions.

Reclaiming depth from 2D images is analogous to estimating the third physical dimension lost during the imaging process. For this purpose, several existing approaches incorporate additional information to regularize this inherently ill-posed inverse problem. Here, we briefly discuss common techniques for depth estimation and their capacity for generating a clean image when the input image is defocused.

Levin *et al.* [18] proposed first technique to recover depth from a single image. An aperture mask was designed based on a prior derived from the probability distribution of the gradients of natural gray-scale images. Veeraraghavan *et al.* [41] proposed a coded aperture technique optimizing the aperture patterns based on the shape of power spectra. Moerno-Noguer *et al.* [27] projected a dotted pattern over the scene.These single-shot coded aperture approaches do not explicitly take into account of image structure and noise [47]. Some may require manual intervention to generate reliable depth maps [18]. Most importantly, spectral

© 2017. This work was supported under the Australian Research Councils Discovery Projects funding scheme (project DP150104645) and an Australian Government RTP Scholarship.
*Both authors contributed equally to this work.

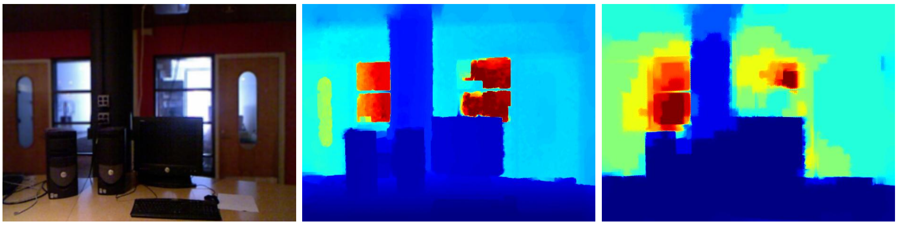

Figure 1: Example of depth prediction using our proposed model on NYU-v2 [29].

distortion introduced by the aperture mask hinders the ability to remove blur since spatial frequencies are systematically attenuated in the captured image [16, 48].

As an alternative to single-shot coded aperture techniques, some methods apply a focus measure for individual pixels across multiple images taken at different focal lengths [32]. The depth map is computed by assigning each pixel the position in the focal stack for which the focus measure of that pixel is maximal. This means the depth resolution is directly proportional to the number of images available. To augment the resolution, filters are applied to focus measure [26, 36] or smooth surfaces are fitted to the estimated depths [39]. Depth-from-defocus with a single image is also targeted by numerous methods [1, 2, 3, 23, 24, 28, 49] that used focus measures and filters.

Many algorithms utilized multiple images [31, 42, 45] for recovering depth of a scene. In [42], the authors used texture invariant rational operators to predict a precise dense depth map. However, accurate customization of those filters is an open question. Xu *et al.* [45] uses two blur observations of the same scene to estimate depth and remove the blur. Li *et al.* [21] measured shading in a scene to refine depth from defocus, iteratively.

Success of deep neural networks in image classification [14, 37], segmentation [25], and recognition [33], inspired single image depth estimation [23]. Recent works of [38], [11], [7] and [23] are relevant to our method. [38] and [11] use a single sharp image to estimate depth map. However, both of these works focus on 3D reconstruction of already known segmented objects. More recently, Liu *et al.* [23] and Eigen *et al.* [7] proposed CNN based approaches for depth estimation. Our algorithm differs from both of these works; Liu *et al.* [23] learns the unary and pairwise potentials from sharp images while Eigen *et al.* [7] use CNN as a black box to estimate depth map using sharp images. Furthermore, we employ out-of-focus images and apply different blur measures to steer our CNN.

Recent deblurring works have imposed constraints on the sparsity of image gradients *e.g.* Levin *et al.* [20] used the hyper-Laplacian prior, cho & Lee applied the L2 prior, Krishnan *et al.* [13] employed the L1/L2 prior. Sinilarly, Xu *et al.* [44] introduced two stage optimization with dominant edges in the image, whereas, Whyte *et al.* [43] used auxiliary variables in Richardson-Lucy deblurring algorithm. Our deblurring method incorporates pixelwise depth map to deblur the images.

In this paper, we aim for depth estimation and blur removal by leveraging on rich representations learned from cascaded convolutional and fully connected neural networks operating on patch pooled feature maps. Current techniques estimate depth from sharp images by relying on manually tuned statistical models. Their depth accuracy is limited due to the variance in the visual world, and usually, human intervention is required. In contrast, our method benefits from the correspondence between the blurred image and depth map. Learning the filters to capture these inherent associations through a deep network acts as a prior for estimating better depth.We also exploit the depth of field to sharpen the out-of-focus image. To the best of our knowledge, predicting depth from a single out-of-focus image using deep

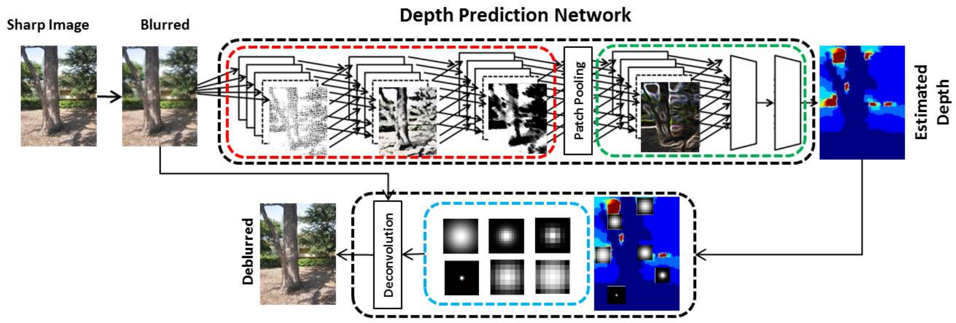

Figure 2: An illustration of the overall method. The sharp image is defocused using circular kernels which simulates capture with a regular aperture. This out-of-focus image is passed to the network ***A*** (shown in red) to compute fully convolutional feature map. A patch pooling extract respective feature map at keypoint locations in the image, which are then propagated through network ***B*** (shown in green) to estimate the depth. Lastly, kernels are computed from the depth map, which are applied to blurred image, that results in an all-focus image.

neural networks has not been investigated before.

Contributions: We claim the following contributions in this paper. 1) Predicting depth from a single out-of-focus image using deep neural networks by exploiting dense overlapping patches, 2) Aligning depth discontinuities between the patches of interest using bilateral filtering, and 3) Incorporating depth map to estimate per pixel blur kernels for non uniform image deblurring.

2 Depth and Deblur Estimation

To estimate depth, we introduce a learning procedure using a modified deep neural network [37] by incorporating image level global context and local evidence. The global context is captured using a fully convolutional network and the local evidence is absorbed using a fully connected network through patch pooling. In the following section, we discuss individual components of our system in more detail.

2.1 Our Network Architecture

The architecture of our network is inspired by VGG 16-layer very deep network [37]. The input to our proposed network at both training and testing stage is a fixed-size RGB image. The only preprocessing we apply is mean-normalization, *i.e.* subtracting the mean RGB value from each pixel (computed separately for Make3D [34] and the NYU-v2 [29] training dataset). The image is passed through a stack of convolutional layers each consist of traditional 3×3 receptive field filters. In contrast to the [37], we didn't utilize 1×1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to one pixel and the spatial padding of convolutional layer is such that the spatial resolution is preserved after convolution, *i.e.* padding one pixel for 3×3 convolutional layers. Spatial pooling is carried out by five max-pooling layers, which is followed by some of the convolutional layers. Max-pooling is performed over a 2×2 pixel window, with a stride of two. Convolutional layers are followed by the patch pooling layer, which takes RGB image as input and generates a dense grid of patches over the entire image. It also pools the feature map for each corresponding patch and returns a fixed size output. Patch pooling layer is followed by three Fully-Connected (FC) layers;

the first two have 4096 channels each while the third performs 21-way depth estimation and thus contains 21 channels (one for each sampled depth class). The final loss layer is the softmax layer.

This objective function essentially minimizes the multinomial logistic loss and it maps the output scores of last fully-connected layer to a probability distribution over classes using the softmax function.

$$\hat{p}_i = \exp(y_i) / \left[\sum_{c=1}^{C} \exp(y_{i^c}) \right] \tag{1}$$

The computed multinomial logistic loss is then computed for the softmax output class probabilities as

$$E = \frac{-1}{N} \sum_{i=1}^{N} \log(\hat{p}_{i,L_i}). \tag{2}$$

where L_i is the quantized depth label for each pixel in the image.

2.2 Depth Prediction

Our network is a cascade of two smaller networks as shown in Fig. 2. The convolutional deep network $\boldsymbol{A}$ (shown in red) is designed specifically to enforce the global image level information in depth estimation. It is followed by a shallow fully connected network $\boldsymbol{B}$ (shown in green) that processes small local evidence for further refinement.

Unlike typical networks, here images are neither cropped nor warped to prevent them from unintended blur artifacts. The network $\boldsymbol{A}$ operates on full scale out of focus images and comprises of 13 convolutional and four max-pooling layers. The output of the network $\boldsymbol{A}$ is a pixel-level feature map and we argue, this is essential in modeling depth dynamic range. Futhermore, each layer of data is a four-dimensional array of size $N \times C \times H \times W$, where N is the number of images in a batch, H is the height of image, W is the width of image and C is the feature (or channel) dimension. The first layer receives the N number of out of focus images Y and in the subsequent layers, input location correspond to the receptive field regions in the image. The convolution, pooling, and activation functions are the basic components and since these operators are translation invariant, they apply to local input regions and depend only on relative spatial positions. In any n-th layer, the feature value f_{ij} for the data vector y_{ij} at location $(i,\ j)$ is computed by

$$f_{ij}^{(n)} = \Psi_{ks}(f_{(i+\delta i, j+\delta j)}^{(n-1)}, 0 < \delta_i, \delta j < k) \tag{3}$$

where k denotes the kernel size of the layer, s is subsampling (by a factor of four in both spatial axes) and Ψ_{ks} is the layer type.

Patch-pooling: Pixel depth prediction requires multiple deconvolutional layers to access an original size image feature map and a pixel-level regression to obtain a full scale depth map. In practice, pixel-level regression with deep and large network architectures require a comparably large number of iterations for convergence in back-propagation, which makes the training memory intensive and slow. To overcome this issue, we introduce a small set of keypoint locations on a regular grid to perform patch pooling. This novel patch pooling layer uses max-pooling to convert the computed network response inside a region of interest into a feature map with a fixed spatial extent of $H \times W$ (e.g., 64×64), where H and W are layer hyper-parameters that are independent of any particular patch. A patch Φ is a rectangular window into the convolutional feature map. A tuple (r, c, h, w) defines each patch and specifies its top-left corner (r, c) and its height and width (h, w). The spatial pyramid-pooling [10] layer is carried out on the output of the network $\boldsymbol{A}$ feature map. For a pyramid level with $n \times m$ keypoints, the patch Φ_{ij} corresponding to (i, j)-th keypoint is denoted by

$$\Phi_{ij} = [\lfloor \frac{i-1}{n} w \rfloor, \lceil \frac{i}{n} w \rceil] \text{ x } [\lfloor \frac{j-1}{m} h \rfloor, \lceil \frac{j}{m} h \rceil]. \tag{4}$$

Intuitively, on the left and top boundary, the floor operation is performed while on the right and bottom boundary, the ceiling. These patches are densely extracted from the entire image and hence, overlap. We extract the respective feature map region for each image patch corresponding to a keypoint. In the backward direction [9], the function computes the gradient of the loss function (i.e. softmax loss) with respect to each of its input data vector y_{ij} at location (i, j) in n-th layer by following the argmax switches as

$$\frac{\partial L}{\partial y_{ij}^{n}} = \sum_{\Phi} \sum_{k} [ij = ij^{*}(\Phi_{ij}, k)] \frac{\partial L}{\partial y_{ij}^{n+1}}. \tag{5}$$

Each mini-batch contains multiple patches (i.e. $\Phi = [\Phi_{00}, \ldots, \Phi_{nm}]$) with the corresponding patch-pooling output y_{ij}^{n+1}. The input data pixel y_{ij}^{n} is a part of several patches, thus (possibly) assigned many different labels k. The partial derivative $\partial L / \partial y_{ij}^{n+1}$ is accumulated if ij is the argmax switch selected for y_{ij}^{n+1} by max pooling. In back-propagation, the partial derivatives $\partial L / \partial y_{ij}^{n+1}$ are already computed by the backward functions of the next layer (*i.e.* the network $\boldsymbol{B}$) on top of our patch-pooling layer.

The network $\boldsymbol{B}$ operates on the sampled feature map, which is defined as 64×64 spatial neighborhood for each sampled keypoint in the image. This network is shallow and consists of only fully connected layers. It is designed specifically to predict *one* depth value for each keypoint in the image. Network $\boldsymbol{A}$, patch-pooling and network $\boldsymbol{B}$ are trained jointly as outlined in Sections 2.4 and 2.5.

2.3 Depth Estimation with Fast Bilateral Filtering

The depth map $\tilde{Z}$ predicted by the network $\boldsymbol{B}$ is not continuous, however, the spatial dimensions of $\tilde{Z}$ and out of focus image Y are same but $\tilde{Z}$ has regions with missing values. In order to estimate the missing pixels in $\tilde{Z}$, we interpolate using nearby keypoints.

Furthermore, our intuition is that the color intensity discontinuities must be aligned with the depth discontinuities between the patches of interest. The predicted depth values at the nearby keypoint locations are used to interpolate the depth for each pixel of out of focus image. Using a fast bilateral filtering with Y, the smoothness constraint on the boundary pixels and the edge alignment constraint on the image pixels can be simultaneously satisfied. Inpainting of depth map is an ill-posed problem, therefore, an additional prior on the structure is required. The filtered depth map Z is a combination of $\tilde{Z}$ (data term) and Y bilateral features (smoothness term) which is inspired by [17].

2.4 Training

We adopt a pragmatic two step training scheme to learn shared features via alternating optimization. We first train network $\boldsymbol{A}$ based on back-propagation to learn weights. Then, fixing the weights for the network $\boldsymbol{A}$, we train network $\boldsymbol{B}$. In addition, we jointly fine-tune both networks, once network $\boldsymbol{A}$ and $\boldsymbol{B}$ are fully trained individually. The training is carried out by mini-batch gradient descent to optimize the softmax objective.

Make3D Depth	**Error (C1)** (lower is better) **rel**	**log10**	**rms**	**Error (C2)** (lower is better) **rel**	**log10**	**rms**
Saxena *et al.* [34]	-	-	-	0.370	-	-
Depth-Transfer (DT) [12]	1.744	0.407	7.089	1.820	0.415	7.787
DCNF [23]	1.644	0.397	6.725	1.698	0.403	7.310
Depth from Defocus (DFD) [4]	0.733	-	4.446	1.000	-	5.149
Ours	**0.213**	**0.075**	**2.560**	**0.202**	**0.312**	**0.079**

Table 1: Comparison for depth estimation on Make3D [34] dataset. Our method achieves the best in all error evaluation metrics using the training/test partition provided with [34] .

In all the experimental settings, the batch size is a single image and its keypoint locations. The number of keypoints are set to 15K patches for NYU-v2 and 7K for Make3D dataset. The learning rate is initially set to 10^{-2} and then decreased by a factor of ten after 15K iterations. In total, we train our system only for 25K iterations (five epochs), hence decreasing the learning rate only once.

As any gradient-descent framework, the initialization of the network weights is important. Improper initialization can stall the convergence due to the numerical instability of gradients in deep networks. To address this issue, we use the pretrained object recognition network weights for initialization of our network. We train the networks for depth prediction using a 21-bin strategy as described in Section 2.3.

2.5 Testing

After jointly fine-tuning both networks $\boldsymbol{A}$ and $\boldsymbol{B}$, we follow the standard test procedure. Given a color input image of size $H \times W \times C$, we extract patches corresponding to all keypoint locations in the image, forward-propagate them through the network $\boldsymbol{A}$ and compute the full image feature map. Subsequently, we perform patch-pooling to extract the features for each corresponding region and forward-propagate them to the network $\boldsymbol{B}$. The output of the network $\boldsymbol{B}$ along with the input image is post-processed using the fast bilateral filtering to estimate the full resolution continuous valued responses for each pixel in the input image.

2.6 Deblurring and Refocusing

After computing the depth map for the out of focus image, we construct a sharp image that is in-focus at all pixels. For this purpose, each pixel of the image is deconvolved using the kernels for every pixel in the depth map. These kernels are directly set from the estimated depth values. Since the deconvolution is done for each pixel, there are no visible artifacts generated near depth discontinuities.

This pixel based deconvolution approach is more effective in comparison to [4, 45] where regions near depth discontinuities exhibit ringing artifacts. We use a modified version of non-blind deblurring by [18]

$$E(x_{ij}) = \|x_{ij} * k^d_{ij} - y_{ij}\|^2 + \tau \|\nabla x_{ij}\|^{0.8}. \quad (6)$$

where x_{ij} is the in-focus image pixel, y_{ij} is the out of focus image pixel and k^d_{ij} is the kernel at location (i, j). For each pixel in image Y, we first compute the kernel k^d_{ij} from the depth map Z at ij-th pixel position. Next, each pixel of the sharp image X is obtained by deconvolving a patch of 25×25 centered around the same pixel of Y with k^d_{ij} using eq 6. This technique ensures that the deconvolved pixel will not be affected by ringing artifacts. The sharp image X is generated by aggregating all deconvolved pixels x_{ij} into their original positions. Although this process of deblurring patches is more accurate but computationally expensive.

NYU-v2	DFD [4]	Saxena [34]	(DT) [12]	Eigen [7]	DCNF [23]	**Ours**
rel	0.609	0.349	0.350	0.215	0.213	**0.094**
log10	-	-	0.131	-	0.087	**0.039**
rms	2.758	1.214	1.200	0.907	0.759	**0.347**

Table 2: Quantitative comparison of our depth algorithm on NYU-v2 [29] dataset with current state-of-the-art alternatives. Our method achieves the best in all error evaluation metrics. Note that the results of Saxena *et al*. [34] and Depth-Transfer [12] are reproduced from [7].

Deblurring	**Peak Signal to Noise Ratio (PSNR)** (Higher is better)					
	Whyte [43]	Cho & Lee [6]	Xu [44]	Krishnan [13]	Levin [20]	**Ours**
Make3D	19.95	20.46	20.71	20.29	20.67	**21.07**
NYU-v2	28.23	31.72	31.82	33.49	33.02	**34.21**

Table 3: Quantitative comparison of our deblurring method on Make3D [34] and NYU-v2 [29] datasets with state of the art deblurring methods.

3 Experimental Analysis and Discussion

In this section, we present both qualitative and quantitative evaluations and comparisons against state-of-the-art methods such as Make3D [34], DepthTransfer [12], DFD [4], and DCNF-FCSP [23]. Similarly for deblurring, we compare with Whyte *et al*. [43], Cho & Lee [6], Krishnan *et al*. [13]. Levin *et al*. [19], and Xu *et al*. [44].

We use average relative error (rel), root-mean-square error (rms) and average log_{10} error for depth estimation and Peak-Signal-to-Noise Ratio (PSNR) for blur removal. Depth estimation experiments were performed using the Caffe framework for efficient inference at the test time. This platform also allows sharing features during training as well. In step-wise training, stochastic gradient descent mini-batches are sampled randomly from N images. Nevertheless, we use all patches for the sampled images in the current mini-batch. Overlapping patches from the same image share computation and memory in the forward and backward passes. Training is done on a standard desktop with an NVIDIA Tesla K40c GPU with 12GB memory.

3.1 Preparing Synthetic Images

Synthetic out of focus images with spatially varying blur kernels were generated by using the corresponding ground truth depth maps. For this purpose, we selected two image datasets having ground truth depth maps for as described in section 3.2. The depth variation is dependent on the collection methods and type of sensors used, *e.g*. Make3D [34] dataset has a depth variation of 1-80 meters with more than two depth layers. Subsequently, the gaussian blur kernel is generated from each depth layer and applied to the corresponding sharp image.

3.2 Datasets

We performed experimental validation on two datasets: NYU-v2 [29] and Make3D [34]. For depth estimation, we use the standard test images provided with these datasets, while for blur removal we use randomly selected subset of images from each dataset.

NYU-v2: This dataset consists of 1449 color and depth images of indoor scenes. The dataset is split into 795 images for training and 654 images for the test. All images are resized to 420×640 and white borders are removed.

Make3D: This dataset consists of 534 color and depth images of outdoor scenes. It is split into 400 images for training and 134 images for the test. All images are resized to 460×345.

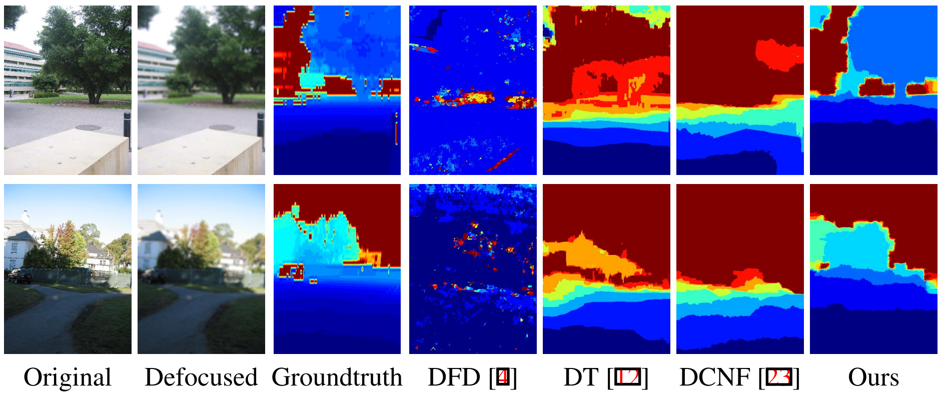

Figure 3: Qualitative comparison of depth estimation on [34] dataset. Our method correctly predicted the depth levels. Red color represents far while blue represents near.

3.3 Depth Estimation

Table 1 shows the results for Make3D dataset. Our proposed method outperforms for all metrics as well as both C1 and C2 errors. In terms of root mean square (rms) our method is leading by a margin of **1.87** (C1 error) and **5.07** (C2 error) from the second best performer.

In [23], the superpixel pooling method extracts useful regions from the convolutional feature maps instead of image crops. However, their superpixel strategy does not take into account the overlapping regions. Besides this, the number of superpixels per image are small and vary in size. In contrast, the patches we select are very dense and have overlapping areas, which helps to predict pixel-wise depth across different patches more accurately. We observe that the keypoint locations and dense patches on a regular grid are more beneficial than non-overlapping superpixel segments. The method in [4] is specifically trained for estimating depth from out of focus images, therefore it outperforms other alternatives that use sharp images only.

The results for NYU-v2 dataset are shown in Table 2. In terms of *rms* error our method is **0.412** higher than the best performing method among the alternatives with similar observations for log_{10} and *rel*. Since the current state-of-the-art methods fail to exploit out of focus images (except [4]), we reproduced their original results for the NYU-v2 dataset. In contrast, our method takes out of focus images to estimate depths and still able to outperform all competing methods by a significant margin. Some qualitative results are shown in Fig. 3. The proposed method has captured the depth accurately for the near as well as far objects in the scene.

3.4 Removing Non-uniform Blur

In this section, we evaluate our blur removal method on test images from NYU-v2 and Make3D. The proposed deblurring method outperforms all competing methods on all test images for non-uniform blur. Figures 4 show the results generated by our and competing methods for different images. Our algorithm delivers higher visual quality than its counterparts. Furthermore, our algorithm is able to restore high frequency texture details with a closer resemblance to the groundtruth than existing methods due to estimating blur kernels from depth layers. In Fig. 4, the highly textured patterns on walls are adeptly reproduced by our algorithm, while these details are clearly missing in the results of the other methods. In this example, most of the other methods tend to smoothen out the variation of the background

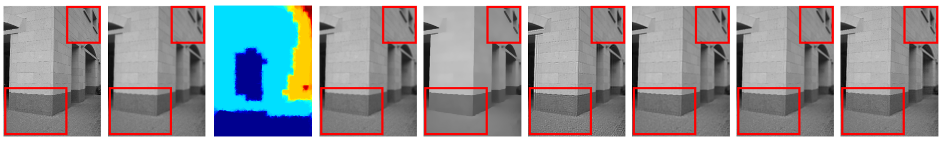

Figure 4: An example from Make3D [34] dataset. Our deblurring method has recovered more details without producing ringing artifacts. Best viewed at higher magnification.

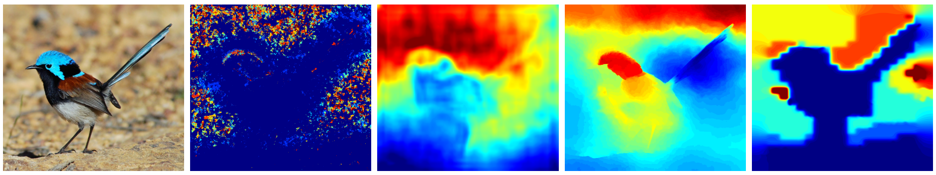

Figure 5: Real defocused image with unknown blur. Our method benefits from the amount of blur in the real images whereas other methods rely on the color and shape of the object which fails to recover the depth.

texture along one of its principal directions.In addition, some methods introduce additional artifacts and artificial textures.

In Table 3, we report the blur removal accuracy of our algorithm, measured by PSNR across all the test images, with the highest PSNR in each comparison is highlighted in bold. The average improvement (in PSNR) by our non-uniform deblurring algorithm over the state-of-the-art methods for NYU-v2 is at least **0.72 dB**, and for Make3D is at least **0.36 dB** on test images as shown in Table 3. This significant improvement demonstrates the advantage of incorporating a deep neural network based depth map for kernel estimation in blur removal.

4 Real Out-of-focus Images

In this experiment, we evaluate the proposed method on a real-world blurred image. Comparisons with state-of-the-art methods [12, 23, 34] are shown in figure 5. In the bird example, the objects close to the camera are in focus while the background is out of focus which is reflected in our results, while other baseline methods fail to capture the relationship between depth and blur and hence, do not perform well in this scenario. As compared to others, our method outperformed, however, our proposed algorithm do not estimate depth on real images as accurately as synthetic images.

5 Conclusion

We have presented a framework that estimates depth map which is utilized to deblur out of focus images. The patch-pooling strategy aims to extract feature map at densely selected keypoint locations which is effective and efficient for depth estimation. The key difference from existing methods is the formulation of the CNN based depth estimation from defocus and incorporating the resulting depth map in deblurring. We have extensively validated our method on benchmark datasets. Our method benefits from out of focus blur, but, it will not be able to estimate depth in presence of camera-shake. Our future work will focus on fixed budget depth estimation from motion blur/camera-shake and joint estimation of depth map and deblur image.

References

[1] Soonmin Bae and FrÃl'do Durand. Defocus magnification. *Computer Graphics Forum*, 2007.

[2] Felipe Calderero and Vicent Caselles. Recovering relative depth from low-level features without explicit t-junction detection and interpretation. *IJCV*, 2013.

[3] Y. Cao, S. Fang, and F. Wang. Single image multi-focusing based on local blur estimation. In *ICIG*, 2011.

[4] Ayan Chakrabarti and Todd Zickler. Depth and deblurring from a spectrally-varying depth-of-field. In *ECCV*. 2012.

[5] Sunghyun Cho and Seungyong Lee. Fast motion deblurring. SIGGRAPH Asia, 2009.

[6] Sunghyun Cho and Seungyong Lee. Fast motion deblurring. In *ACM Transactions on Graphics (TOG)*, 2009.

[7] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.

[8] Rob Fergus, Barun Singh, Aaron Hertzmann, Sam T. Roweis, and William T. Freeman. Removing camera shake from a single photograph. SIGGRAPH, 2006.

[9] Ross Girshick. Fast r-cnn. In *ICCV*, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[11] Abhishek Kar, Shubham Tulsiani, Joao Carreira, and Jitendra Malik. Category-specific object reconstruction from a single image. In *CVPR*, 2015.

[12] Kevin Karsch, Ce Liu, and Sing Bing Kang. Depth transfer: Depth extraction from video using non-parametric sampling. *TPAMI*, 2014.

[13] Dilip Krishnan, Terence Tay, and Rob Fergus. Blind deconvolution using a normalized sparsity measure. In *CVPR*, 2011.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[15] Anat Levin. Blind motion deblurring using image statistics. In *NIPS*, 2006.

[16] Anat Levin. Analyzing depth from coded aperture sets. In *ECCV*. 2010.

[17] Anat Levin, Assaf Zomet, and Yair Weiss. Learning how to inpaint from global image statistics. In *ICCV*. IEEE, 2003.

[18] Anat Levin, Rob Fergus, Frédo Durand, and William T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Trans. Graph.*, 2007.

[19] Anat Levin, Yair Weiss, Fredo Durand, and William T Freeman. Efficient marginal likelihood optimization in blind deconvolution. In *CVPR*, 2011.

[20] Anat Levin, Yair Weiss, Fredo Durand, and William T. Freeman. Understanding blind deconvolution algorithms. *TPAMI*, 2011.

[21] C. Li, S. Su, Y. Matsushita, K. Zhou, and S. Lin. Bayesian depth-from-defocus with shading constraints. In *CVPR*, 2013.

[22] Feng Li, Jingyi Yu, and Jinxiang Chai. A hybrid camera for motion deblurring and depth map super-resolution. In *CVPR*, 2008.

[23] Fayao Liu, Chunhua Shen, and Guosheng Lin. Deep convolutional neural fields for depth estimation from a single image. In *CVPR*, 2015.

[24] M. Liu, M. Salzmann, and X. He. Discrete-continuous depth estimation from a single image. In *CVPR*, 2014.

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

[26] M. Mahmood and T. S. Choi. Nonlinear approach for enhancement of image focus volume in shape from focus. *TIP*, 2012.

[27] Francesc Moreno-Noguer, Peter N. Belhumeur, and Shree K. Nayar. Active refocusing of images and videos. *ACM Trans. Graph.*, 2007.

[28] V. P. Namboodiri and S. Chaudhuri. Recovery of relative depth from a single observation using an uncalibrated (real-aperture) camera. In *CVPR*, 2008.

[29] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[30] S. K. Nayar and M. Ben-Ezra. Motion-based motion deblurring. *TPAMI*, 2004.

[31] C. Paramanand and A. N. Rajagopalan. Non-uniform motion deblurring for bilayer scenes. In *CVPR*, 2013.

[32] Said Pertuz, Domenec Puig, and Miguel Angel Garcia. Analysis of focus measure operators for shape-from-focus. *PR*, 2013.

[33] Ali Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *CVPR Workshops*, 2014.

[34] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Learning 3d scene structure from a single still image. *TPAMI*, 2009.

[35] Qi Shan, Jiaya Jia, and Aseem Agarwala. High-quality motion deblurring from a single image. *ACM Trans. Graph.*, 2008.

[36] S. O. Shim and T. S. Choi. A fast and robust depth estimation method for 3d cameras. In *ICCE*, 2012.

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[38] Hao Su, Qixing Huang, Niloy J. Mitra, Yangyan Li, and Leonidas Guibas. Estimating image depth using shape collections. *ACM Trans. Graph.*, 2014.

[39] M. Subbarao and Tao Choi. Accurate recovery of three-dimensional shape from image focus. *TPAMI*, 1995.

[40] Yu-Wing Tai, Hao Du, M. S. Brown, and S. Lin. Image/video deblurring using a hybrid camera. In *CVPR*, 2008.

[41] Ashok Veeraraghavan, Ramesh Raskar, Amit Agrawal, Ankit Mohan, and Jack Tumblin. Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing. *ACM Trans. Graph.*, 2007.

[42] Masahiro Watanabe and Shree K. Nayar. Rational filters for passive depth from defocus. *IJCV*, 1998.

[43] Oliver Whyte, Josef Sivic, Andrew Zisserman, and Jean Ponce. Non-uniform deblurring for shaken images. *IJCV*, June 2012.

[44] Li Xu and Jiaya Jia. Two-phase kernel estimation for robust motion deblurring. In *ECCV*. 2010.

[45] Li Xu and Jiaya Jia. Depth-aware motion deblurring. In *ICCP*, 2012.

[46] Lu Yuan, Jian Sun, Long Quan, and Heung-Yeung Shum. Image deblurring with blurred/noisy image pairs. SIGGRAPH, 2007.

[47] C. Zhou and S. Nayar. What are good apertures for defocus deblurring? In *ICCP*, 2009.

[48] Changyin Zhou, Stephen Lin, and Shree K Nayar. Coded aperture pairs for depth from defocus and defocus deblurring. *IJCV*, 2011.

[49] Shaojie Zhuo and Terence Sim. Defocus map estimation from a single image. *Pattern Recognition*, 2011.