# A Unified Approach of Multi-scale Deep and Hand-crafted Features for Defocus Estimation

Jinsun Park† zzangjinsun@gmail.com
Yu-Wing Tai‡ yuwing@gmail.com
Donghyeon Cho† cdh12242@gmail.com
In So Kweon† iskweon@kaist.ac.kr

†Robotics and Computer Vision Lab., KAIST, Korea, Republic of
‡Tencent YouTu Lab., China

## Abstract

*In this paper, we introduce robust and synergetic hand-crafted features and a simple but efficient deep feature from a convolutional neural network (CNN) architecture for defocus estimation. This paper systematically analyzes the effectiveness of different features, and shows how each feature can compensate for the weaknesses of other features when they are concatenated. For a full defocus map estimation, we extract image patches on strong edges sparsely, after which we use them for deep and hand-crafted feature extraction. In order to reduce the degree of patch-scale dependency, we also propose a multi-scale patch extraction strategy. A sparse defocus map is generated using a neural network classifier followed by a probability-joint bilateral filter. The final defocus map is obtained from the sparse defocus map with guidance from an edge-preserving filtered input image. Experimental results show that our algorithm is superior to state-of-the-art algorithms in terms of defocus estimation. Our work can be used for applications such as segmentation, blur magnification, all-in-focus image generation, and 3-D estimation.*

## 1. Introduction

The amount of defocus represents priceless information can be obtained from a single image. If we know the amount of defocus at each pixel in an image, higher level information can be inferred based on defocus values such as depth [43], salient region [13] and foreground and background of a scene [26] and so on. Defocus estimation, however, is a highly challenging task, not only because the estimated defocus values vary spatially, but also because the estimated solution contains ambiguities [17], where the appearances of two regions with different amounts of defocus can be very similar. Conventional methods [2, 35, 42] rely on strong edges to estimate the amount of defocus. Determining the amount of defocus only based on the strength of strong edges, however, may lead to overconfidence and misestimations. Thus, we need a more reliable and robust defocus descriptor for defocus estimations.

(a) Image (b) Defocus Map (c) Refocused

Figure 1: Our defocus estimation result and a digital refocusing application example.

In this paper, we present hand-crafted and deep features which assess various aspects of an image for defocus estimation, and a method to obtain a reliable full defocus map of a scene. Our hand-crafted features focus on three components of an image: the frequency domain power distribution, the gradient distribution and the singular values of an image. We also utilize a convolutional neural network (CNN) to extract high-dimensional deep features directly learnt from millions of in-focus and blurred image patches. All of the features are concatenated to construct our defocus feature vector and are fed into a fully connected neural network classifier to determine the amount of defocus.

One of the challenges associated with the defocus estimation is the vagueness of the amount of defocus in homogeneous regions, as such regions show almost no difference in appearance when they are in-focus or blurred. To avoid this problem, we first estimate the amount of defocus using multi-scale image patches from only strong edges, and then propagate the estimated values into homogeneous regions with the guidance of edge-preserving filtered input image. The full defocus map is obtained after the propagation step. We use the defocus map for various applications, such as segmentation, blur magnification, all-in-focus image generation, and 3-D estimation. Figure 1 shows an example of our defocus estimation result and a refocusing application.

## 2. Related Work

**Defocus Estimation** Defocus estimation plays an important role in many applications in the computer vision

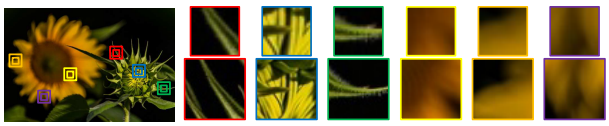
community. It is used in digital refocusing [2,5], depth from defocus [19,30,43], salient region detection [13] and image matting [26], to name just a few.

Elder and Zucker [10] estimate the minimum reliable scale for edge detection and defocus estimation. They utilize second derivative Gaussian filter responses, but this method is not robust due to errors which arise during the localization of edges. Bae and Durand [2] also utilize second derivative Gaussian filter responses to magnify the amount of defocus on the background region. However, their strategy is time-consuming owing to its use of a brute-force scheme. Tai and Brown [35] employ a measure called *local contrast prior*, which considers the relationships between local image gradients and local image contrasts, but the local contrast prior is not robust to noise. Zhuo and Sim [42] use the ratio between the gradients of input and re-blurred images with a known Gaussian blur kernel. However, it easily fails with noise and edge mislocalization. Liu *et al.* [22] inspect the power spectrum slope, gradient histogram, maximum saturation and autocorrelation congruency. Their segmentation result, however, cannot precisely localize blurry regions. Shi *et al.* [31] use not only statistical measures such as peakedness and heavy-tailedness but also learnt filters from image data with labels. Homogeneous regions in the image are weak points in their algorithm. Shi *et al.* [32] construct sparse dictionaries containing sharp and blurry bases and determine which dictionary can reconstruct an input image sparsely, but their algorithm is not robust to large blur, as it is tailored for just noticeable blur estimation.

**Neural Networks** Neural networks have proved their worth as algorithms superior to their conventional counterparts in many computer vision tasks, such as object and video classification [14, 29], image restoration [9], image matting [7], image deconvolution [38], motion blur estimation [34], blur classification [1,40], super-resolution [8], salient region detection [16] and edge-aware filtering [39].

Sun *et al.* [34] focus on motion blur kernel estimation. They use a CNN to estimate pre-defined discretized motion blur kernels. However, their approach requires rotational input augmentation and takes a considerable amount of time during the MRF propagation step. Aizenberg *et al.* [1] use a multilayer neural network based on multivalued neurons (MVN) for blur identification. The MVN learning step is computationally efficient. However, their neural network structure is quite simple. Yan and Shao [40] adopt two-stage deep belief networks (DBN) to classify blur types and to identify blur parameters, but they only rely on features from the frequency domain.

**Our Work** Compared with the previous works, instead of using only hand-crafted features, we demonstrate how we can apply deep features to the defocus estimation problem. The deep feature is learnt directly from training data with different amounts of defocus blur. Because each extracted deep feature is still a local feature, our hand-crafted features, which capture both local and global information of an image patch, demonstrate the synergetic effect of boosting the performance of our algorithm. Our work significantly outperforms previous works on defocus estimation in terms of both quality and accuracy.



(a) Image (b) From strong edges (c) From weak edges

Figure 2: Multi-scale patches from strong and weak edges.

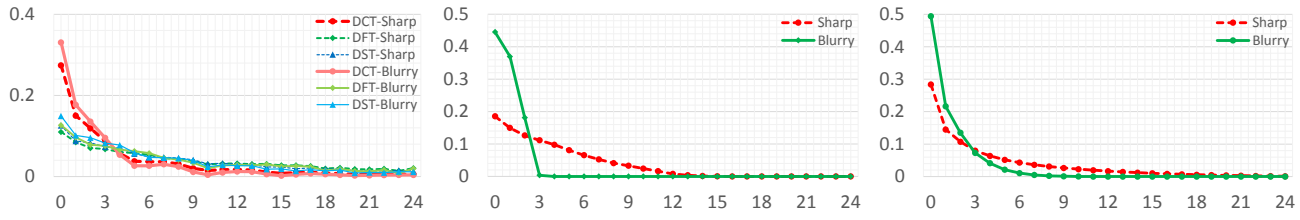
## 3. Feature Extraction

We extract multi-scale image patches from an input image for feature extraction. In addition, we extract image patches on edges only because homogeneous regions are ambiguous in defocus estimation. For edge extraction, we first transform the input image from the RGB to the HSV color space and then use a V channel to extract image edges.

There have been numerous hand-crafted features for sharpness measurements [3, 22, 25, 35, 37]. In this work, three hand-crafted features related to the frequency domain power distribution, the gradient distribution, and the singular values of a grayscale image patch are proposed. The deep feature is extracted from a CNN which directly processes color image patches in the RGB space for feature extraction. All of the extracted features are then concatenated to form our final defocus feature.

### 3.1. Multi-scale Patch Extraction

Because we extract hand-crafted and deep features based on image patches, it is important to determine a suitable patch size for each pixel in an image. Although there have been many works related to scale-space theories [20,23,24], there is still some ambiguity with regard to the relationship between the patch scale and the hand-crafted features, as they utilize global information in an image patch. In other words, a sharp image patch can be regarded as blurry mistakenly depending on the size of the patch, and vice versa. In order to avoid patch scale dependency, we extract multi-scale patches depending on the strength of the edges. In natural images, strong edges are more likely to be in-focus than blurry ones ordinarily. Therefore, we assume that image patches from strong edges are in-focus and that weak edges are blurry during the patch extraction step. For sharp patches, we can determine their sharpness accurately with a small patch size, whereas blurry patches can be ambiguous with a small patch size because of their little change in the appearance when they are blurred or not. Figure 2 shows multi-scale patches from strong and weak edges. Figure 2 (b) shows that small patches from strong




(a) (b) (c)

Figure 3: Sharp and blurry hand-crafted features. Average (a) DCT, DFT, DST, (b) gradient and (c) SVD features from sharp (dotted) and blurry (solid) patches. The number of samples exceeds more than 11K. The absolute difference between sharp and blurry features can be a measure of discriminative power.

edges still have abundant information for defocus estimation, while Figure 2 (c) shows that small patches from weak edges severely lack defocus information and contain high degrees of ambiguity. Based on this observation, we extract small patches from strong edges and large patches from weak edges. Edges are simply extracted using the Canny edge detector [4] with multi-threshold values. In general, the majority of the strong edges can be extracted in an in-focus area. Weak edges can be extracted in both sharp and blurry areas because they can come from in-focus weak textures or out-of-focus sharp textures. Our multi-scale patch extraction scheme boosts the performance of a defocus estimation algorithm drastically (Section 3.6).

### 3.2. DCT Feature

We transform a grayscale image patch $P_I$ to the frequency domain to analyze its power distribution. We utilize the discrete cosine transform (DCT) because the DCT offers strong energy compaction [28]; i.e., most of the information pertaining to a typical signal tends to be concentrated in a few low-frequency bands. Hence, when an image is more detailed, more non-zero DCT coefficients are needed to preserve the information. Accordingly, we can examine high-frequency bands at a higher resolution with the DCT than with the discrete Fourier transform (DFT) or the discrete sine transform (DST). Because an in-focus image has more high-frequency components than an out-of-focus image, the ratio of high-frequency components in an image patch can be a good measure of the blurriness of an image. Our DCT feature $f_D$ is constructed using the power distribution ratio of frequency components as follows:

$$f_D(k) = \frac{1}{W_D} \log \left( 1 + \sum_{\theta} \sum_{\rho=\rho_k}^{\rho_{k+1}} \frac{|\mathcal{P}(\rho, \theta)|}{S_k} \right), \; k \in [1, n_D], \tag{1}$$

where $|\cdot|$, $\mathcal{P}(\rho, \theta)$, $\rho_k$, $S_k$, $W_D$ and $n_D$ denote the absolute operator, the discrete cosine transformed image patch with polar coordinates, the $k$-th boundary of the radial coordinate, the area enclosed by $\rho_k$ and $\rho_{k+1}$, a normalization factor to make sum of the feature unity, and the dimensions of the feature, respectively. Figure 3 (a) shows features from sharp and blurry patches after different transformations. The absolute difference between sharp and blurry features can be a measure of discriminative power. In this case, the DCT feature has the best discriminative power because its absolute difference between sharp and blurry features is greater than those of the other transformations.

(a) Original (b) $n = 60$ (c) $n = 30$ (d) $n = 15$

Figure 4: Low-rank matrix approximation of an image. (a) Original and (b)-(d) approximated images with the number of preserved singular values. The more number of singular values are preserved, the more details are also preserved.

### 3.3. Gradient Feature

We calculate the gradients of $P_I$ using Sobel filtering to obtain a gradient patch $P_G$. Typically, there are more strong gradients in a sharp image patch than in a blurry image. Therefore, the ratio of the strong gradient components in an image patch can be another measure of the sharpness of the image. We use the normalized histogram of $P_G$ as a second component of our defocus feature. Our gradient feature $f_G$ is defined as follows:

$$f_G(k) = \frac{1}{W_G} \log \left( 1 + \mathcal{H}_G(k) \right), \; k \in [1, n_G], \tag{2}$$

where $\mathcal{H}_G$, $W_G$ and $n_G$ denote the histogram of $P_G$, the normalization factor and the dimensions of the feature, respectively. Figure 3 (b) shows a comparison of sharp and blurry gradient features. Despite its simplicity, the gradient feature shows quite effective discriminative power.

### 3.4. SVD Feature

Singular value decomposition (SVD) has many useful applications in signal processing. One such application of SVD is the low-rank matrix approximation [27] of an image. The factorization of an $m \times n$ real matrix $\mathbf{A}$ can be

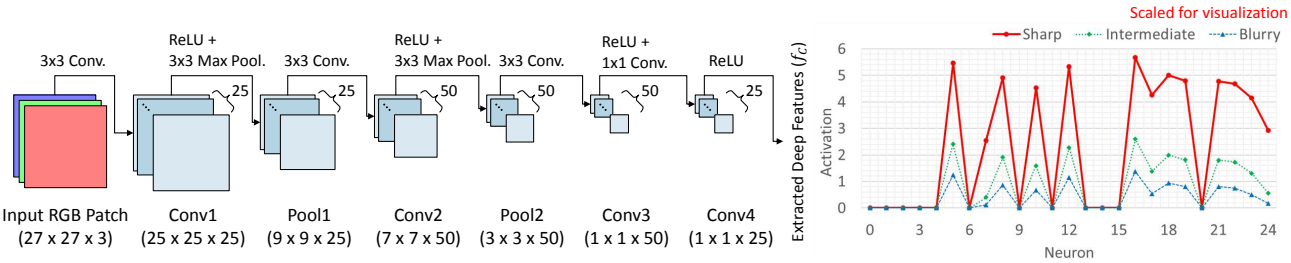



Figure 5: Our deep feature extraction network and average activations with sharp, intermediate and blurry patches. The output dimensions of each layer are shown together. The stride is set to 1 for convolution and to 3 for max pooling.

written as follows:

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T = \sum_{k=1}^{N} A_k = \sum_{k=1}^{N} \lambda_k u_k v_k^T, \qquad (3)$$

where $\mathbf{\Lambda}$, $N$, $\lambda_k$, $u_k$ and $v_k$ denote the $m \times n$ diagonal matrix, the number of non-zero singular values of $\mathbf{A}$, the $k$-th singular value, and the $k$-th column of the real unitary matrices $\mathbf{U}$ and $\mathbf{V}$, respectively. If we construct a matrix $\tilde{\mathbf{A}} = \sum_{k=1}^{n} A_k$, where $n \in [1, N]$, we can approximate the given matrix $\mathbf{A}$ with $\tilde{\mathbf{A}}$. In the case of image reconstruction, low-rank matrix approximation will discard small details in the image, and the amount of the loss of details is inversely proportional to $n$. Figure 4 shows an example of the low-rank matrix approximation of an image. Note that the amount of preserved image detail is proportional to the number of preserved singular values.

We extract a SVD feature based on low-rank matrix approximation. Because more non-zero singular values are needed to preserve the details in an image, a sharp image patch tends to have more non-zero singular values than a blurry image patch; i.e., a non-zero $\lambda_k$ with large $k$ is a clue to measure the amount of detail. The scaled singular values define our last hand-crafted feature as follows:

$$f_S(k) = \frac{1}{W_S} \log\left(1 + \lambda_k\right), \; k \in [1, n_S], \qquad (4)$$

where $W_S$ denotes the normalization factor and $n_S$ denotes the dimensions of the feature. Figure 3 (c) shows a comparison of sharp and blurry SVD features. The long tail of the sharp feature implies that more details are preserved in an image patch.

### 3.5. Deep Feature

We extract the deep feature $f_C$ from a color image patch using a CNN. To deal with multi-scale patches, small-scale patches are zero-padded before they are fed into the CNN. Our feature extraction network consists of convolutional, ReLU and max pooling layers, as illustrated in Figure 5. Successive convolutional, ReLU and max pooling layers are suitable to obtain highly non-linear features. Because the deep feature is learnt from a massive amount of training data (Section 5.1), it can accurately distinguish between sharp and blurry features. In addition, it compensates for the lack of color and cross-channel information in the hand-crafted features, which are important and valuable for our task. Figure 5 also shows the average outputs from our feature extraction network with sharp, intermediate and blurry image patches. The activations are proportional to the sharpness of the input image.

| Feature | Accuracy(%) | Feature | Accuracy(%) |
|---|---|---|---|
| $f_D$ (DCT) | 38.15 | $f_H$ (Hand-crafted) | 71.49 |
| $f_G$ (Gradient) | 68.36 | $f_C$ (Deep) | 89.38 |
| $f_S$ (SVD) | 61.76 | $f_B$ (Concatenated) | 94.16 |

Table 1: Classification accuracies. Note that the accuracy of a random guess is 9.09 %.

### 3.6. Our Defocus Feature

We concatenate all of the extracted features to construct our final defocus feature $f_B$ as follows:

$$f_B = [f_H,\, f_C] = [[f_D,\, f_G,\, f_S],\, f_C], \qquad (5)$$

where $[\cdot]$ denotes the concatenation. Table 1 shows the classification accuracy of each feature. We use a neural network classifier for the accuracy comparison. The classification tolerance is set to an absolute difference of 0.15 compared to the standard deviation value $\sigma$ of the ground truth blur kernel. We train neural networks with the same architecture using those features individually and test on 576,000 features of 11 classes. The details of the classifier and the training process will be presented in Sections 4.1 and 5.1. Our deep feature, $f_C$, has the most discriminative power with regard to blurry and sharp features as compared to other individual hand-crafted features, $\{f_D, f_G, f_S\}$, and their concatenation, $f_H$. When all hand-crafted and deep features are concatenated ($f_B$), the performance is even more enhanced. Removing one of the hand-crafted features drops the performance by approximately 1-3%. For example, the classification accuracies of $[f_D, f_S, f_C]$ and $[f_G, f_S, f_C]$ are 93.25% and 91.10%, respectively. In addition, the performance of $f_B$ with only single-scale patch extraction also decreases to 91.00%.

## 4. Defocus Map Estimation

The defocus features are classified using a neural network classifier to determine the amount of defocus at the center point of each patch. We obtain an initial sparse defocus map after the classification step and then filter out a number of outliers from the sparse defocus map using a sparse joint bilateral filter [42] adjusted with the classification confidence values. A full defocus map is estimated from the filtered sparse defocus map using a matting Laplacian algorithm [18]. The edge-preserving smoothing filtered [41] color image is used as a guidance of propagation.

### 4.1. Neural Network Classifier

We adopt a neural network classifier for classification because it can capture the highly non-linear relationship between our defocus feature components and the amount of defocus. Moreover, its outstanding performance has been demonstrated in various works [14, 29, 33]. Our classifier network consists of three fully connected layers (300-150-11 neurons each) with ReLU and dropout layers. The softmax classifier is used for the last layer. Details about the classifier training process will be presented in Sections 5.1 and 5.2. Using this classification network, we obtain the labels and probabilities of features, after which the labels are converted to the corresponding $\sigma$ values of the Gaussian kernel, which describe the amount of defocus. Subsequently, we construct the sparse defocus map $I_S$ using the $\sigma$ values and the confidence map $I_C$ using the probabilities.

### 4.2. Sparse Defocus Map

The sparse defocus map is filtered by the probability-joint bilateral filter to reject certain outliers and noise. In addition, we filter the input image with an edge-preserving smoothing filter to create a guidance image $I_G$ for probability-joint bilateral filtering and sparse defocus map propagation. A rolling guidance filter [41] is chosen because it can effectively remove image noise together with the distracting small-scale image features and prevent erroneous guidances on some textured and noisy regions. Our probability-joint bilateral filter $B(\cdot)$ is defined as follows:

$$B(\mathbf{x}) = \frac{1}{W(\mathbf{x})} \sum_{\mathbf{p}\in\mathcal{N}_\mathbf{x}} G_{\sigma_s}(\mathbf{x},\mathbf{p}) \times G_{\sigma_r}(I_G(\mathbf{x}), I_G(\mathbf{p})) G_{\sigma_c}(1, I_C(\mathbf{p})) I_S(\mathbf{p}), \quad (6)$$

where $G_\sigma(\mathbf{u},\mathbf{v}) = \exp(-\|\mathbf{u}-\mathbf{v}\|_F^2/2\sigma^2)$ and $\|\cdot\|_F$, $W$, $\mathcal{N}_\mathbf{x}$, $\sigma_s$, $\sigma_r$ and $\sigma_c$ denote the Frobenius norm, the normalization factor, the non-zero neighborhoods of $\mathbf{x}$, the standard deviation of spatial, range and probability weight, respectively. The probability weight $G_{\sigma_c}$ is added to the conventional joint bilateral filter in order to reduce the unwanted effects of the outliers within neighborhood with low probability values. Probability-joint bilateral filtering removes outliers and regularizes the sparse defocus map effectively as shown in Figure 6.

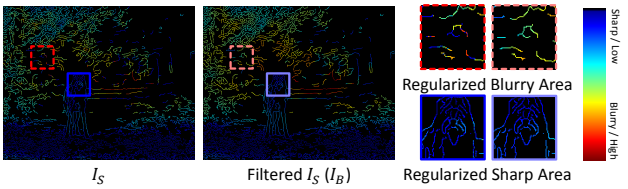



Figure 6: Sparse defocus maps before ($I_S$) and after ($I_B$) probability-joint bilateral filtering. Solid boxes denote sharp areas (small value) and dashed boxes denote blurry areas (large value). Some outliers are filtered out effectively.

### 4.3. Full Defocus Map

The full defocus map is obtained from the sparse defocus map using the matting Laplacian algorithm [18] with the help of the guidance image $I_G$. The matting Laplacian is defined as follows:

$$L(i,j) = \sum_{k|(i,j)\in \mathrm{w}_k} \Big(\delta_{ij} - \frac{1}{|\mathrm{w}_k|}\big(1 + (I_G(i)-\mu_k) \times (\Sigma_k + \frac{\epsilon}{|\mathrm{w}_k|} I_3)^{-1}(I_G(j)-\mu_k)\big)\Big), \quad (7)$$

where $i$, $j$, $k$, $\mathrm{w}_k$, $|\mathrm{w}_k|$, $\delta_{ij}$, $\mu_k$, $\Sigma_k$, $I_3$ and $\epsilon$ denote the linear indices of pixels, a small window centered at $k$, the size of $\mathrm{w}_k$, the Kronecker delta, the mean and variance of the $\mathrm{w}_k$, a 3×3 identity matrix and a regularization parameter, respectively. The full defocus map $I_F$ is obtained by solving the following least-squares problem:

$$\hat{I}_F = \gamma\,(L + D_\gamma)^{-1}\,\hat{I}_B, \quad (8)$$

where $\hat{I}$, $\gamma$ and $D_\gamma$ denote the vector form of matrix $I$, a user parameter and a diagonal matrix whose element $D(i,i)$ is $\gamma$ when $I_B(i)$ is not zero, respectively. An example of a full defocus map is shown in Figure 1 (b).

## 5. Experiments

We first describe our method to generate the training data and train the classifier, after which we compare the performance of our algorithm with the performances of state-of-the-art algorithms using a blur detection dataset [31]. Various applications such as blur magnification, all-in-focus image generation and 3-D estimation are also presented. Our codes and dataset are available on our project page.[1]

### 5.1. Classifier Training

For the classification network training, 300 sharp images are randomly selected from the ILSVRC [29] training data.

[1] https://github.com/zzangjinsun/DHDE_CVPR17

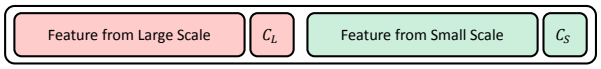



Figure 7: Our feature scale encoding scheme. If a feature is from large-scale, we fill the small-scale positions with zeros, and vice versa.

Similar to the feature extraction step, we extract approximately 1M multi-scale image patches on strong edges and regard these patches as sharp patches. After that, each sharp patch $P_I^S$ is convolved with synthetic blur kernels to generate blurry patches $P_I^B$ as follows:

$$P_I^{B_l} = P_I^S * h_{\sigma_l}\ ,\ l \in [1, L], \tag{9}$$

where $h_\sigma$, $*$ and $L$ denote the Gaussian blur kernel with a zero mean and variance $\sigma^2$, the convolution operator and the number of labels, respectively. We set $L = 11$ and the $\sigma$ values for each blur kernel are then calculated as follows:

$$\sigma_l = \sigma_{min} + (l-1)\sigma_{inter}, \tag{10}$$

where we set $\sigma_{min} = 0.5$ and $\sigma_{inter} = 0.15$.

For the training of the deep feature, $f_C$, we directly connect the feature extraction network and the classifier network to train the deep feature and classifier simultaneously. The same method is applied when we use the concatenated feature, $f_H$, for training. For the training of $f_B$, we initially train the classifier connected to the feature extraction network only (i.e., with $f_C$ only), after which we fine-tune the classifier with the hand-crafted features, $f_H$. We use the Caffe [12] library for our network implementation.

### 5.2. Scale Encoding

While we trained the classifier with features from small and large patches together, we found that the network does not converge. This occurs because some features from different scales with different labels can be similar. If we train classifiers for each scale individually, it is inefficient and risky, as the parameters to be trained are doubled and there is no assurance that individual networks are consistent.

In order to encode scale information to a feature itself, we assign specific positions for features from each scale, as shown in Figure 7. An entire feature consists of positions and constants for each scale. Additional constants are assigned to deal with biases for each scale.

In neural networks, the update rules for weights and the bias in a fully connected layer are as follows:

$$w_{jk}^t \rightarrow w_{jk}^t - \eta(a_k^{t-1}\delta_j^t), \tag{11}$$

$$b_j^t \rightarrow b_j^t - \eta\delta_j^t, \tag{12}$$

where $w_{jk}^t$, $b_j^t$, $\eta$, $a$ and $\delta$ denote the $k$-th weight in the $j$-th neuron of the layer $t$, the bias of the $j$-th neuron of the layer $t$, the learning rate, the activation, and the error, respectively. For the input layer ($t = 1$), if we set the $k$-th dimension of the input feature to a constant $C$, Equation (11) becomes $w_{jk}^1 \rightarrow w_{jk}^1 - \eta(C\delta_j^1)$, and surprisingly, if we let $C = 1$, the update rule takes the same form of the bias update rule. Therefore, by introducing additional constants into each scale, we are able not only effectively to separate the biases for each scale but also to apply different learning rates for each bias with different constants. After encoding scale information to a feature itself, the classifier network converges. In our experiments, we simply set $C_L = C_S = 1$.

Owing to this encoding scheme, we set the number of neurons in the first layer of the classifier network such that it is roughly two times greater than the dimensions of our descriptor to decode small- and large-scale information from encoded features.

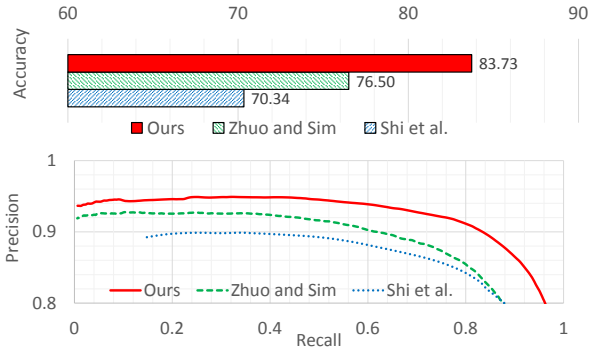



Figure 8: Segmentation accuracies (top) and Precision-Recall comparison (bottom) of Shi *et al.* [31], Zhuo and Sim [42] and our algorithm.

### 5.3. Blur Detection Dataset

We verify the reliability and robustness of our algorithm using a blur detection dataset [31]. The dataset used contains 704 natural images containing blurry and sharp regions and corresponding binary blurry region masks manually segmented by a human. We extract $15 \times 15$ patches on strong edges and $27 \times 27$ patches on weak edges. We set $n_D = n_G = n_S = 25$ for large patches, $n_D = n_G = n_S = 13$ for small patches, $\sigma_s = \sigma_r = 100.0$, $\sigma_c = 1.0$, $\epsilon = 1e^{-5}$ and $\gamma = 0.005$ for all experiments. We compare our algorithm to the results of Shi *et al.* [31], Shi *et al.* [32] and Zhuo and Sim [42]. Because the blur detection dataset contains only binary masks, quantitative results are obtained using binary blurry region masks from each algorithm. For binary segmentation, we apply a simple thresholding method to the full defocus map. The threshold value $\tau$ is determined as follows:

$$\tau = \alpha v_{max} + (1-\alpha)v_{min}, \tag{13}$$

where $v_{max}$ and $v_{min}$ denote the maximum and the minimum values of the full defocus map, and $\alpha$ is a user parameter. We set $\alpha = 0.3$ for the experiments empirically and

(a) (b) (c) (d) (e) (f) (g) (h)

Figure 9: Defocus map estimation and binary blurry region segmentation results. (a) Input images. (b) Results of [30]. (c) Results of [31]. (d) Results of [32] (Inverted for visualization). (e) Results of [42]. (f) Our defocus maps and (g) corresponding binary masks. (h) Ground truth binary masks.

(a) Images (b) $f_D$ (c) $f_G$ (d) $f_S$ (e) $f_C$ (f) $f_H$ (g) $f_B$

Figure 10: Defocus maps from each feature. Features used for the estimation of each defocus map are annotated. The blue solid boxes and red dashed boxes in (e), (f) and (g) show the complementary roles of the hand-crafted and deep features.

this value works reasonably well. Figure 8 shows the segmentation accuracies and the precision-recall curves, and Figure 9 shows the results of the different algorithms. The segmentation accuracies are obtained from the ratio of the number of pixels correctly classified to the total number of pixels. Precision-Recall curves can be calculated by adjusting $\tau$ from $\sigma_1$ to $\sigma_L$.

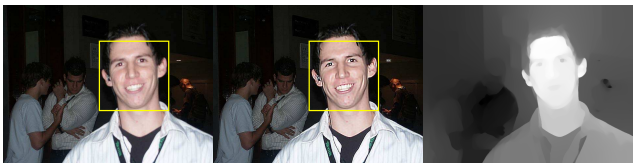
Our algorithm shows better results than the state-of-the-art methods quantitatively and qualitatively. In the homogeneous regions of an image, sufficient textures for defocus estimation do not exist. The results of [31] and [32], therefore, show some erroneous classification results in such regions. Our algorithm and [42] can avoid this problem with the help of sparse patch extraction on strong edges. In contrast, sparse map propagation can also lead to uncertain results in textured regions because the prior used for the propagation is based on the color line model [18], as shown in Figure 9 (e). An edge-preserving smoothed color image is adopted as a propagation prior in our algorithm, and it gives better results, as shown in Figures 9 (f) and (g). In addition, we compare our algorithm with that of Shi *et al.* [30]. We conducted a *RelOrder* [30] evaluation and obtained a result of 0.1572 on their dataset, which is much better than their result of 0.1393[2] of [30]. Moreover, the segmentation accuracy of [30] (53.30%) is much lower than the accuracy of our algorithm because their method easily fails with a large amount of blur (Figure 9 (b)).

We also examined defocus maps from single and concatenated features qualitatively. As shown in Figures 10 (b), (c) and (d), single hand-crafted features give unsatisfactory results compared to deep or concatenated features. Surprisingly, a deep feature alone (Figure 10 (e)) works quite well but gives a slightly moderate result compared to the concatenated features (See the solid blue boxes). The hand-crafted feature alone (Figure 10 (f)) also works nicely but there are several misclassifications (See the dashed red boxes). These features show complementary roles when they are concatenated. Certain misclassifications due to the hand-crafted feature are well handled by the deep feature, and the discriminative power of the deep feature was strengthened with the aid of the hand-crafted features, as shown in Figure 10 (g).

[2]Different from the reported value in [30] because we utilize our own implementation. The evaluation codes of [30] have not been released.

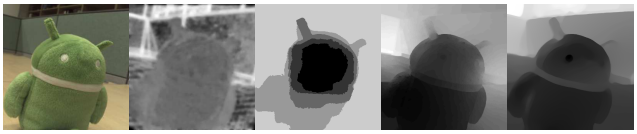
(a) Input (b) Magnified (c) Defocus map

Figure 11: Defocus blur magnification. The background blurriness is amplified to direct more attention to the foreground.



(a) Input (b) All-in-focus (c) Defocus map

Figure 12: All-in-focus image generation. Blurry regions (yellow boxes) in the original image look clearer in the all-in-focus image.



(a) (b) (c) (d) (e)

Figure 13: Depth estimation. (a) Input image. (b) Tao *et al.* [36]. (c) Jeon *et al.* [11]. (d) Liu *et al.* [21] (e) Ours. Our depth map looks reasonable compared to those in the other works.

### 5.4. Applications

The estimated defocus maps can be used for various applications. We apply our defocus maps to the following applications and the results are quite pleasing.

**Defocus Blur Magnification** Our algorithm can be used for defocus blur magnification tasks. We can highlight the foreground by amplifying the blurriness of the background. Figure 11 shows an example of the defocus blur magnification. The foreground objects appear more prominent in the blur magnified image.

**All-in-focus Image Generation** Contrary to defocus blur magnification, we can deblur blurry regions in an image to obtain an all-in-focus image. Using the $\sigma$ values of each pixel in the defocus map, we generate corresponding Gaussian blur kernels and use them to deblur the image. We use the hyper-laplacian prior [15] for non-blind deconvolution. Figure 12 shows an example of the all-in-focus image generation. In accordance with the defocus map, we deblur the original image in a pixel-by-pixel manner. The blurry regions in the original image are restored considerably in the all-in-focus image (Figure 12 (b)).

**3-D Estimation from a Single Image** The amount of defocus is closely related to the depth of the corresponding point because the scaled defocus values can be regarded as pseudo-depth values if all of the objects are located on the same side of the focal plane. Because we need both defocused images and depth maps, we utilize light-field images, as there are numerous depth estimation algorithms and because digital refocusing can easily be done. We decode light-field images using [6], and then generate a refocused image using [36]. Our algorithm is compared to algorithms for light-field depth estimation [11,36] and an algorithm for single image depth estimation [21]. Figure 13 shows an input image and depth map from each algorithm. Our depth map from a single image appears reasonable compared to those in the other works, which utilize correspondences between multiple images.

(a) (b) (c) (d) (e)

Figure 14: Effects of additional random seed points. (a) Input image. (b) $I_S$. (c) $I_S$ + Seeds. (d) $I_F$ from (b). (e) $I_F$ from (c). Additional random seed points can effectively enhance the defocus accuracy in large homogeneous regions.

## 6. Conclusion

We have introduced a unified approach to combine handcrafted and deep features and demonstrated their complementary effects for defocus estimation. A neural network classifier is shown to be able to capture highly non-linear relationships between each feature, resulting in high discriminative power. In order to reduce the patch scale dependency, multi-scale patches are extracted depending on the strength of the edges. The homogeneous regions in an image are well handled by a sparse defocus map, and the propagation process is guided by an edge-preserving smoothed image. The performance of our algorithm is compared to those of the state-of-the-art algorithms. In addition, the potential for use in various applications is demonstrated.

One limitation of our algorithm is that we occasionally obtain incorrect defocus values in a large homogeneous area. This is due to the fact that there is no strong edge within such regions for defocus estimation. A simple remedy to address this problem involves the random addition of classification seed points in large homogeneous regions. Figure 14 shows the effect of additional seed points. Additional random seed points effectively guide the sparse defocus map, causing it to be propagated correctly into large homogeneous areas.

For future works, we expect to develop fully convolutional network architectures for our task.

## Acknowledgements

This work was supported by the Technology Innovation Program (No. 2017-10069072) funded by the Ministry of Trade, Industry & Energy (MOTIE, Korea).

## References


[1] I. Aizenberg, D. V. Paliy, J. M. Zurada, and J. T. Astola. Blur identification by multilayer neural network based on multivalued neurons. *IEEE Transactions on Neural Networks*, 19(5):883–898, 2008.

[2] S. Bae and F. Durand. Defocus magnification. In *Computer Graphics Forum*, volume 26, pages 571–579. Wiley Online Library, 2007.

[3] C. F. Batten. *Autofocusing and astigmatism correction in the scanning electron microscope*. PhD thesis, Citeseer, 2000.

[4] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (6):679–698, 1986.

[5] Y. Cao, S. Fang, and Z. Wang. Digital multi-focusing from a single photograph taken with an uncalibrated conventional camera. *IEEE Transactions on Image Processing*, 22(9):3703–3714, 2013.

[6] D. Cho, M. Lee, S. Kim, and Y.-W. Tai. Modeling the calibration pipeline of the lytro camera for high quality light-field image reconstruction. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3280–3287. IEEE, 2013.

[7] D. Cho, Y.-W. Tai, and I. Kweon. Natural image matting using deep convolutional neural networks. In *European Conference on Computer Vision (ECCV)*, pages 626–643. Springer, 2016.

[8] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a deep convolutional network for image super-resolution. In *European Conference on Computer Vision (ECCV)*, pages 184–199. Springer, 2014.

[9] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 633–640, 2013.

[10] J. H. Elder and S. W. Zucker. Local scale control for edge detection and blur estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):699–716, 1998.

[11] H.-G. Jeon, J. Park, G. Choe, J. Park, Y. Bok, Y.-W. Tai, and I. S. Kweon. Accurate depth map estimation from a lenslet light field camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1547–1555, 2015.

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[13] P. Jiang, H. Ling, J. Yu, and J. Peng. Salient region detection by ufo: Uniqueness, focusness and objectness. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1976–1983. IEEE, 2013.

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014.

[15] D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1033–1041, 2009.

[16] G. Lee, Y.-W. Tai, and J. Kim. Deep saliency with encoded low level distance map and high level features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 660–668, 2016.

[17] A. Levin, R. Fergus, F. Durand, and W. T. Freeman. Image and depth from a conventional camera with a coded aperture. *ACM Transactions on Graphics*, 26(3), July 2007.

[18] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):228–242, 2008.

[19] J. Lin, X. Ji, W. Xu, and Q. Dai. Absolute depth estimation from a single defocused image. *IEEE Transactions on Image Processing*, 22(11):4545–4550, 2013.

[20] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998.

[21] F. Liu, C. Shen, and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5162–5170, 2015.

[22] R. Liu, Z. Li, and J. Jia. Image partial blur detection and classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE, 2008.

[23] X.-M. Liu, C. Wang, H. Yao, and L. Zhang. The scale of edges. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 462–469. IEEE, 2012.

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[25] X. Marichal, W.-Y. Ma, and H. Zhang. Blur determination in the compressed domain using dct information. In *IEEE International Conference on Image Processing (ICIP)*, volume 2, pages 386–390. IEEE, 1999.

[26] M. McGuire, W. Matusik, H. Pfister, J. F. Hughes, and F. Durand. Defocus video matting. In *ACM Transactions on Graphics*, volume 24, pages 567–576. ACM, 2005.

[27] W. H. Press. *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press, 2007.

[28] K. R. Rao and P. Yip. *Discrete cosine transform: algorithms, advantages, applications*. Academic press, 2014.

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, pages 1–42, 2015.

[30] J. Shi, X. Tao, L. Xu, and J. Jia. Break ames room illusion: depth from general single images. *ACM Transactions on Graphics*, 34(6):225, 2015.

[31] J. Shi, L. Xu, and J. Jia. Discriminative blur detection features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2965–2972, 2014.

[32] J. Shi, L. Xu, and J. Jia. Just noticeable defocus blur detection and estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 657–665, 2015.

[33] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[34] J. Sun, W. Cao, Z. Xu, and J. Ponce. Learning a convolutional neural network for non-uniform motion blur removal. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 769–777. IEEE, 2015.

[35] Y.-W. Tai and M. S. Brown. Single image defocus map estimation using local contrast prior. In *IEEE International Conference on Image Processing (ICIP)*, pages 1797–1800. IEEE, 2009.

[36] M. W. Tao, S. Hadap, J. Malik, and R. Ramamoorthi. Depth from combining defocus and correspondence using light-field cameras. In *IEEE International Conference on Computer Vision (ICCV)*, pages 673–680. IEEE, 2013.

[37] C.-Y. Wee and R. Paramesran. Image sharpness measure using eigenvalues. In *International Conference on Signal Processing (ICSP)*, pages 840–843. IEEE, 2008.

[38] L. Xu, J. S. Ren, C. Liu, and J. Jia. Deep convolutional neural network for image deconvolution. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1790–1798, 2014.

[39] L. Xu, J. S. Ren, Q. Yan, R. Liao, and J. Jia. Deep edge-aware filters. In *International Conference on Machine Learning (ICML)*, pages 1669–1678, 2015.

[40] R. Yan and L. Shao. Image blur classification and parameter identification using two-stage deep belief networks. In *British Machine Vision Conference (BMVC)*. Citeseer, 2013.

[41] Q. Zhang, X. Shen, L. Xu, and J. Jia. Rolling guidance filter. In *European Conference on Computer Vision (ECCV)*, pages 815–830. Springer, 2014.

[42] S. Zhuo and T. Sim. Defocus map estimation from a single image. *Pattern Recognition*, 44(9):1852–1858, 2011.

[43] D. Ziou and F. Deschênes. Depth from defocus estimation in spatial domain. *Computer Vision and Image Understanding*, 81(2):143–165, 2001.