Estimating Spatially Varying Defocus Blur from A Single Image

Xiang Zhu, *Student Member, IEEE,* Scott Cohen, *Member, IEEE,* Stephen Schiller, *Member, IEEE,* and Peyman Milanfar, *Fellow, IEEE*

***Abstract*—Estimating the amount of blur in a given image is important for computer vision applications. More specifically, the spatially varying defocus point-spread-functions (PSFs) over an image reveal geometric information of the scene, and their estimate can also be used to recover an all-in-focus image. A PSF for a defocus blur can be specified by a single parameter indicating its scale. Most existing algorithms can only select an optimal blur from a finite set of candidate PSFs for each pixel. Some of those methods require a coded aperture filter inserted in the camera. In this paper we present an algorithm estimating a defocus scale map from a single image, which is applicable to conventional cameras. This method is capable of measuring the probability of local defocus scale in the *continuous* domain. It also takes smoothness and color edge information into consideration to generate a coherent blur map indicating the amount of blur at each pixel. Simulated and real data experiments illustrate excellent performance, and its successful applications in foreground/background segmentation.**

***Index Terms*—spatially varying blur estimation, defocus blur.**

I. Introduction

OPTICAL IMAGING systems have a limited depth of field, which may lead to defocus blur. Most blind deconvolution algorithms focus on estimating shift-invariant point-spread-functions (PSFs), or shift-varying PSFs that can be treated as projections of a globally constant blur descriptor caused by camera shake [1][2][3][4][5]. However, estimating defocus blur is a challenging task mainly because the corresponding PSFs are spatially varying and cannot be represented by any global descriptor. Indeed, spatially varying defocus PSFs for a given camera can be pre-calibrated and described typically through a simple model (e.g. disc, Gaussian) that is characterized by a single parameter indicating its scale (radius, standard deviation, etc.) For an image, we call the 2D map of the scale parameter the *defocus blur map*, which indicates the level of local blur at each pixel (see an example in Fig. 1). The main purpose of this paper is to provide an automatic way of estimating a defocus blur map from a single input image.

Defocus blur map estimation has several potential applications. For example, it can be employed to detect and segment in-focus subjects from the out-of-focus background, helping a photo editor to edit the subject of interest or the background, separately. Besides that, since defocus blur level is intimately related with depth of the scene, a blur map also provides important information for depth estimation. The computation of depth information typically requires two photos of the same scene taken at the same time but from slightly different vantage points, i.e. a stereo pair [6]. However, in most cases only one image is available. A blur map allows one to reconstruct a 3D scene from a single photograph as long as the camera settings (focal length, aperture settings, etc.) are known. For image restoration applications, if both the defocus PSF calibration and blur map estimation are made, we can reconstruct an all-in-focus image through a non-blind spatially varying deblurring process.

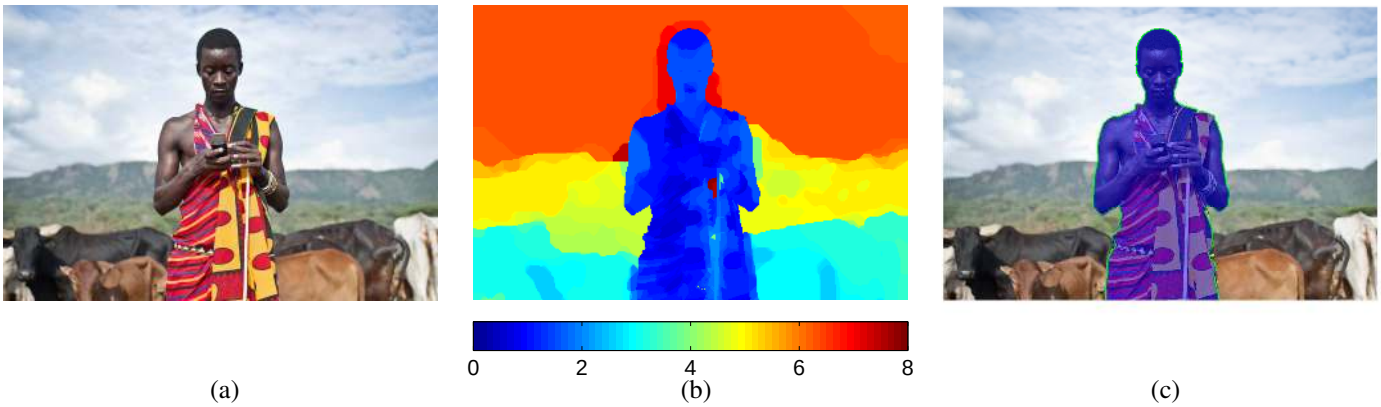
In [7] Levin et al. proposed an algorithm that simultaneously restores a sharp image and a depth map from a single input. This method locally selects the best PSF by evaluating its deconvolution errors. It requires a specially designed aperture filter for the camera, which strongly limits its domain of application. Instead of estimating the optimal blur scale in the continuous domain, it can only identify the most likely candidate from a finite number of calibrated PSFs with somewhat limited accuracy. Chakrabarti et al. suggested a method estimating the likelihood function of a given candidate PSF based on local frequency component analysis without deconvolution [8]. In their paper the method is applied to detect simple motion blur, but it can also be employed for defocus blur identification. Again it can only detect optimal PSFs from a finite number of candidates.

In this paper we propose a new method for estimating PSF scale at each pixel. The estimation is based on local frequency component analysis similar to [8], but is significantly more general since it is carried out in the continuous domain. Smoothness constraints and image color edge information are also taken into consideration to generate a map that is smooth and meanwhile allows discontinuity in the boundary regions between objects (such as boundaries between sharp foreground subject and blurry background). This algorithm does not rely on any specific functional model of the PSFs and is therefore very generally applicable. It can be implemented using any PSF model that is a function of a single parameter. As we illustrate in Section IV, even without accurate PSF calibration and modeling the method can still roughly tell local blur level for real images by employing the disc function as an approximate model.

Bae and Durand [9] perform blur estimation to magnify focus differences, but the blur estimation is done only at edges. Their blur map is essentially interpolated elsewhere. Their first step is an explicit edge detection step, which may not be very robust to either strong blur or noise. Since the goal in [9] is magnifying focus differences, the case of a background that is

X. Zhu and P. Milanfar are with the Department of Electrical Engineering, UC Santa Cruz, Santa Cruz, CA, 95064. {xzhu,milanfar}@soe.ucsc.edu.

S. Cohen and S. Schiller are with Adobe Research, Adobe Systems Incorporated, San Jose, CA 95110. {schiller,scohen}@adobe.com.



(a) (b) (c)

Fig. 1. Defocus blur map estimation experiment using a real image. (a) Test image. (b) Estimated defocus blur map. (c) Automatic foreground/background segmentation.

too blurry for reliable edge detection is not mentioned. On the other hand, our statistical models are applicable everywhere there is some image contrast, even where there is not a single clear edge that can be localized. Thus, we produce a dense set of probability distributions versus blur radii over the image. Our method models changes in energy at all frequencies with blur and not just very high frequencies (edges). The method of [9] models only step edges with a Gaussian blur PSF.

Our continuous blur radius modeling discussed in section III-A leads to a very accurate estimate of local blur, which in turn provides for better discrimination than [8] in separating the effects of defocus blur over noise and image content. A second important improvement over [8] is that we find and enhance 2nd and 3rd local maxima in the blur radius probability distribution at each pixel. This is discussed in section III-B. When the global maximum does not give the correct blur radius, the 2nd or 3rd highest local maximum almost always does (see Fig. 5). Our smoothness constraint then allows our method to choose the proper radius, thereby signficantly reducing errors in the blur radius map.

The use of a sharp-edged window in the local frequency analysis in [8] causes more mixing of values in the frequency domain thus reducing the signal to noise ratio. Our formulation uses a Gaussian windowing function to avoid this problem. In [8], a single horizontal or vertical motion blur kernel is chosen from a set of candidate motions and then a final binary labeling problem is solved to segment the moving object from the assumed static background. This final step can only distinguish between part of the image blurred with one blur kernel and those parts of the image that are not blurred with this one kernel. Our technique is more flexible in that we can distinguish between areas of the image that are blurred with multiple (but different) blur kernels.

The rest of this paper is organized as follows. Section II gives an analysis on local image statistics to motivate the basic estimation idea. The proposed algorithm is described in Section III. Simulated and real data experiments are given in Section IV to show the algorithm performance. We also provide application examples in this section, focusing mainly on automatic foreground/background segmentation. Known shortcomings are discussed in Section V. Finally, we summarize and discuss directions of future research in Section VI.

II. Local Image Statistics Analysis

An imaging process suffering from spatially changing blur and random noise can be generally modeled as:

$$g[\mathbf{x}] = (h_{\mathbf{x}} \otimes f)[\mathbf{x}] + n[\mathbf{x}] \tag{1}$$

where $\otimes$ denotes a 2-D convolution operator. f and g represent the ideal all-in-focus image and the observed blurry image (in gray level), respectively. $h_{\mathbf{x}}$ is the spatially varying blur kernel at position $\mathbf{x}$, and n denotes random noise that is assumed to be i.i.d. Gaussian: $n[\mathbf{x}] \sim \mathcal{N}(0, \sigma_n^2)$.

Because both f and $h_{\mathbf{x}}$ are unknown, the blur estimation is highly ill-posed, and thus prior knowledge about the latent image content f is required. Although the distribution of f is difficult to describe, we assume that its gradient field can be locally modeled as white Gaussian. Specifically, in a small analysis window η of size $N \times N$ we have

$$f^{\nabla}[\mathbf{x}] = (\nabla \otimes f)[\mathbf{x}] \sim \mathcal{N}(0, s_{\mathbf{x}}), \quad \forall \mathbf{x} \in \eta \tag{2}$$

where ∇ denotes a derivative operator in a particular direction (horizontal or vertical). $s_{\mathbf{x}}$ represents local variance in the window η around $\mathbf{x}$. We assume that blur kernel $h_{\mathbf{x}}$ is constant inside η. For simplicity, in the rest part of this paper we use h and s to replace $h_{\mathbf{x}}$ and $s_{\mathbf{x}}$, respectively.

It is known that information about blur can be conveniently analyzed by the means of a frequency spectrum given the observed g. We first define a *localized* 2-D Fourier filter basis $\{t_i\}_i$, which is a set of functions over the same spatial extent as the analysis window η. Each such function represents a different spatial frequency, or a group of related spatial frequencies. Specifically, a Gabor filter is employed, which is the product of a pure sinusoid with a 2-D Gaussian function. For example, for the i-th frequency $(\omega_1^{(i)}, \omega_2^{(i)})$, the function value at position $\mathbf{x} = (x_1, x_2)^T$ is

$$t_i[\mathbf{x}] = w[\mathbf{x}] \exp\left(-2\pi j \left(x_1\omega_1^{(i)} + x_2\omega_2^{(i)}\right)\right). \tag{3}$$

Here the 2-D Gaussian function $w[\mathbf{x}]$ is centered in the analysis window η and its standard deviation is $1/4$ of the diameter N of the window size. This has the advantage of tapering values down to 0 as they approach the edges of the window. Otherwise the window edges will appear to be sharp in the image and mask the true frequency response.

The choice of the set $\{(\omega_1^{(i)}, \omega_2^{(i)})\}_i$ depends on the window size. We use frequencies $(\omega_1, \omega_2) = (a_1/N, a_2/N)$ for even values of a_1 in the interval $[-\frac{N-1}{4}, \frac{N-1}{4}]$ and even values of a_2 in the interval $[0, \frac{N-1}{4}]$, except that when $a_2 = 0$ then a_1 is in $[-\frac{N-1}{4}, -2]$ to avoid redundancy and the DC filter $a_1 = a_2 = 0$. For $N = 41$, we have 60 complex filters t_i. A subset of the real (cosine) filters in our Gabor filter bank for $N = 41$ is shown in Figure 2.

If we impose such localized Fourier analysis onto image g within the window centered at $\mathbf{x}$:

$$g_i^{\nabla}[\mathbf{x}] = \left(g^{\nabla} \otimes t_i\right)[\mathbf{x}], \tag{4}$$

then, using [8] as a starting point we can derive the likelihood function of the modulus squared of these coefficients as:

$$p\left(\{|g_i^{\nabla}[\mathbf{x}]|^2\}_i | h, s\right) = \prod_i \mathbf{Exp}\left(|g_i^{\nabla}[\mathbf{x}]|^2; 1/\left(s\sigma_{hi}^2 + \sigma_{ni}^2\right)\right) \tag{5}$$

where $\mathbf{Exp}$ is the exponential distribution, and where $\{\sigma_{hi}^2\}_i$ is called the *blur spectrum* for blur kernel h defined by:

$$\sigma_{hi}^2 = \sum_{\mathbf{x}} |(h \otimes t_i)[\mathbf{x}]|^2 \tag{6}$$

and $\{\sigma_{ni}^2\}_i$ is the noise spectrum:

$$\sigma_{ni}^2 = \sigma_n^2 \sigma_{\nabla i}^2 \quad \text{with} \quad \sigma_{\nabla i}^2 \triangleq \textstyle\sum_{\mathbf{x}\in\eta} |\nabla \otimes t_i[\mathbf{x}]|^2. \tag{7}$$

The real and imaginary parts $\mathrm{Re}(g_i^{\nabla})$ and $\mathrm{Im}(g_i^{\nabla})$ are independent normal distributions with equal variances $\frac{1}{2}(s\sigma_{hi}^2 + \sigma_{ni}^2)$ when our window function w is not used. When w is used, then this statement is approximate but very accurate for the higher frequencies $(\omega_1^{(i)}, \omega_2^{(i)})$ in our filter bank. We are modeling $|g_i^{\nabla}[\mathbf{x}]|^2 = (\mathrm{Re}(g_i^{\nabla}[\mathbf{x}]))^2 + (\mathrm{Im}(g_i^{\nabla}[\mathbf{x}]))^2$. It is well known that the sum of the squares of two standard normal $N(0,1)$ random variables is $\chi_2^2 \equiv \mathbf{Exp}(\frac{1}{2})$. It is then easy to derive (5) for two independent $N(0, \frac{1}{2}(s\sigma_{hi}^2 + \sigma_{ni}^2))$ variables.

Note that in [8] no Gaussian window is used. Instead, a hard rectangular window is implicitly imposed on the image data. One advantage of using a hard window is that the local power spectra can be better localized in space. However, as we know from the convolution theorem, multiplication in one domain (spatial or frequency) corresponds to a convolution in the other domain. Thus, by using a hard window in the spatial domain the frequency data is convolved with the Fourier transform of the hard window: a 2D tensor product of sinc functions. In contrast, multiplication by a Gaussian in the spatial domain corresponds to convolution with a Gaussian in the frequency domain. Since the power of $\mathrm{sinc}(f)$ function falls off as $1/|f|$ its power is much more spread out than that of a Gaussian, and thus there is more mixing of components of the spectrum from the hard window.

The effects from mixing are ameliorated to some extent by computing the power spectrum of the blur kernels using the same window (hard or soft), thus introducing the same mixing into those spectra as the spectra obtianed from the image. Still, in real experiments with both hard and soft windows we have found that using the Gaussian window gives more accurate results in distinguishing between various blur radii. However in other situations, such as estimating motion blur, the superiority of the hard window in localizing frequency information may be a more important factor and would thus warrant the choice of a hard window.

Assume that the defocus PSF model h is given, and that it can be indexed by its scale value r: $h = h(r)$. Theoretically the optimal $\hat{r}$ could be selected by maximizing the likelihood function (5) if both s and σ_n^2 are given:

$$\hat{r} = \arg\max_r p\left(\{|g_i^{\nabla}[\mathbf{x}]|^2\}_i | h(r), s\right) \tag{8}$$

Although the image noise variance σ_n^2 may be compressed at the light or dark ends of the camera response, for our purposes it has been sufficient to model the noise as spatially constant. It can be estimated by many approaches, for example [10]. However, the variance of the latent image gradients s is unknown and is difficult to estimate directly.

From (5) we estimate the conditional likelihood function as

$$p\left(\{|g_i^{\nabla}[\mathbf{x}]|^2\}_i | h\right) \propto \max_s \prod_i \mathbf{Exp}\left(|g_i^{\nabla}[\mathbf{x}]|^2; 1/\left(s\sigma_{hi}^2 + \sigma_{ni}^2\right)\right) \tag{9}$$

where the optimal $\hat{s}$ that maximizes the conditional likelihood is selected for each given h. In other words, maximizing the likelihood function (9) is equivalent to optimizing it over both s and r simultaneously:

$$< \hat{r}, \hat{s} >= \arg\max_{r,s} p\left(\{|g_i^{\nabla}[\mathbf{x}]|^2\}_i | h(r), s\right) \tag{10}$$

However, it is still not quite clear why optimizing (5) with respect to both r and s is a reasonable way to select the scale r, since we do not have any prior knowledge about r or s.

To further analyze the behavior of the likelihood function (5) over r and s, a simulated experiment is carried out and shown in Fig. 3, where (a) shows the latent test image patch of size 41×41. We use a disc function to simulate the defocus PSF and its radius to define the scale value r. The radius of the true h that convolves the image patch is set as: $r^* = 5$. White Gaussian noise with $\sigma_n^2 = 10^{-7}$ is also added according to (1).[1] Then, we decompose the simulated patch (b) through equation (4) (where horizontal derivative filter $\nabla = [1, -1]$ is used) and calculate the likelihood value p with different s and r based on equation (5). The results are plotted in (d), where a global maximum is located in the point with the true radius value. In this case, maximizing function (10) in the continuous domain can generate the correct r.

However, it is not guaranteed that the global maximum always indicates the true radius. Fig. 4 illustrates another example where we implement the same simulation as Fig. 3 but with a different patch (see Fig. 4 (a)). At this time there still exists a local maximum around the true radius value, but it is no longer the global maximum.

[1] The pixel intensity range here is $[0, 1]$.

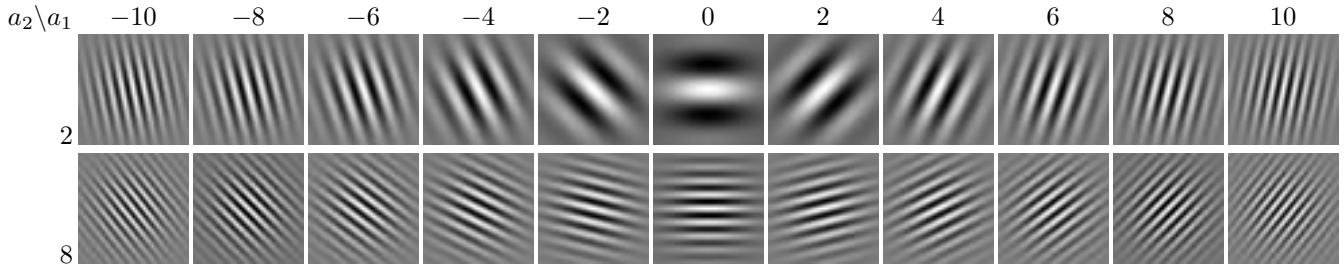


Fig. 2. A subset of the cosine filters $w[\mathbf{x}]\cos\left(-2\pi\left(x_1(a_1/N)+x_2(a_2/N)\right)\right)$ in our Gabor filter bank for $N\times N$ windows of size $N=41$.

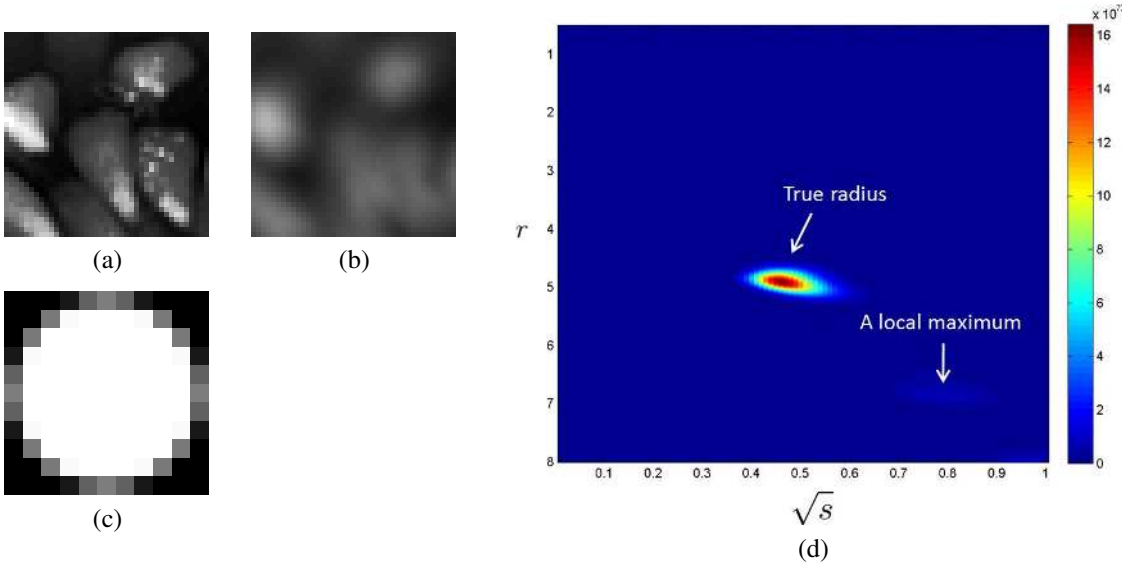


Fig. 3. Simulated experiment based on a local patch. (a) Latent test patch. (b) Simulated blurry patch. (c) True PSF using the disc model. (d) Plot of the conditional likelihood values of (5) with different r and s.

We repeat this experiment on overlapping patches centered at every pixel of Fig. 5 (b), which is uniformly convolved by the disc function with $r^*=5$. For each patch, the radius $\hat{r}_1$ corresponding to the global maximum, and the radius $\hat{r}_2$ corresponding to the second highest local maximum are detected and illustrated in Fig. 5 (c) and (d) respectively. From (c) we can see that for most pixels the latent radii are correctly captured, but meanwhile there exist some "holes" where the maximum likelihood estimation failed (see circled regions for example). At the same time, for most of these holes the correct radii values are captured by the second highest maxima (see (d)).

This phenomenon is further illustrated in Fig. 6 we see plots of $\log(p(r))$, assuming optimal s as in eq. (9), at three different points in the image in Fig. 5 blured with the disc of radius 5. The blue plot is at a point where the maximum likelihood estimation gives the correct radius. The red and green plots are at points where the maximum likelihood estimation fails, but where there is a clear local maxima at $r=5$.

At this point we should make a few comments as to why the maximum likelihood estimation fails in some cases. One factor is that the power spectra of the disc kernels we are using have periodic lobes and zeros that scale along the frequency axis with the radius of the disc. Thus, if r_0 is estimated to have a high probability, and another r shares some of the same power spectrum zeros as r_0, then r will also tend to be assigned a high probability. This results in the various local maxima in the plot of $\log(p)$ versus r. Then, one local maximum may be elevated over another for reaons of noise, or the latent image not being sharp, or the actual power spectrum of the latent image being far from the modeled one.

From the above simulations, we can conclude that:

1. Function (5) is non-convex over r and s.
2. In many cases, the global maximum point of (5) corresponds to the latent r^*, but this is not guaranteed.
3. For most cases, the true radius value r is located in a local/global maximum with a relatively high probability.

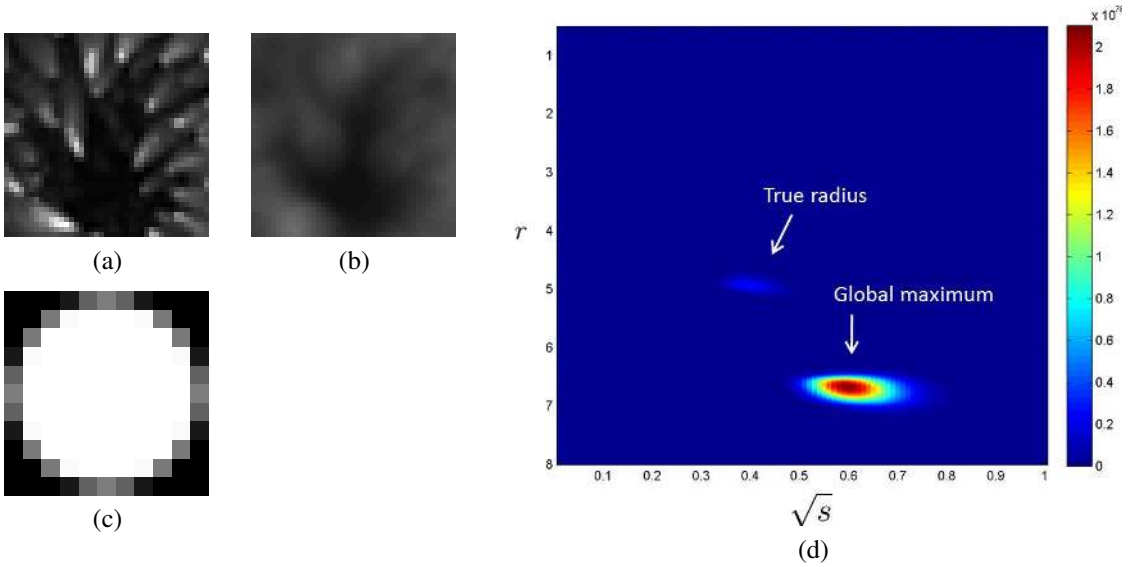
For the maximum likelihood estimation in (5), because we don't have any prior on either r or s, its accuracy is limited. However, function (5) still provides candidate r for most patches. If priors or constraints about r can be taken into account, then it is possible to improve the quality of blur map estimation further.

III. Proposed Method

Our blur map estimation approach includes two main steps:

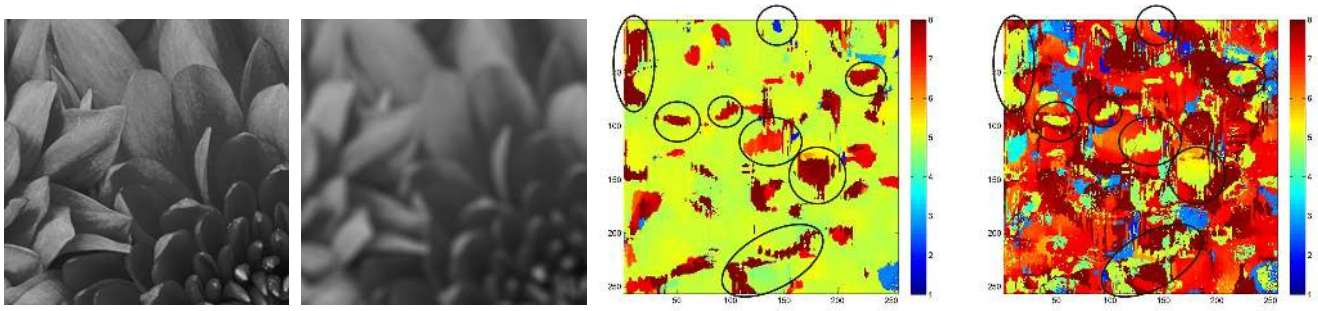
1. Local probability estimation;
2. Coherent map labeling.

Given an input color image, the first step estimates up to 3 candidate scale r values for every pixel in its luminance



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

Fig. 4. Simulated experiment based on a local patch. (a) Latent test patch. (b) Simulated blurry patch. (c) True PSF using the disc model. (d) Plot of the conditional likelihood values of (9) with different r and s.



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

Fig. 5. Simulated experiment based on an image. (a) Latent in-focus image. (b) Simulated blurry image convolved by a disc function with radius $r^* = 5$. (c) Estimated radii map corresponding to the global maxima. (d) Estimated radii map corresponding to the second highest local maxima. In the circled regions, the true radii values are missed by the global maxima, but captured by the second highest local maxima.

channel: the candidate r values correspond to the global/local maxima of function (5) with the highest likelihood values, and they are calculated in the continuous domain. The second step creates a coherent blur map based on the estimate of the first step, image derivative information, and a smoothness prior.

A. Local Probability Estimation

To find the most important local maxima of (5) we use a fixed point iteration, namely, calculating the optimal r or s iteratively with the other variable fixed.

In the defocus blur situation the blur spectrum $\{\sigma_{hi}^2\}_i$ is solely determined by r, i.e. $h = h(r)$. Thus, in this section we use the notation $\sigma_i^2(r)$ to describe σ_{hi}^2. It has been deduced in [8] that given a fixed set of blur spectra $\{\sigma_i^2(r)\}_i$, the optimal $\hat{s}$ maximizing (5) can be found through the following fixed point iteration:

$$\hat{s} = \left(\sum_i \rho_i(\hat{s})\right)^{-1} \sum_i \rho_i(\hat{s}) \frac{|g_i^{\nabla}[\mathbf{x}]|^2 - \sigma_{ni}^2}{\sigma_i^2(r)}, \qquad (11)$$

where

$$\rho_i(\hat{s}) = \left(1 + \frac{\sigma_{ni}^2}{\hat{s}\sigma_i^2(r)}\right)^{-2}.$$

Although it is possible to analytically compute the blur spectrum for a disc kernel, we want to keep our system general enough to handle any blur kernel model (such as the somewhat polygonal blur kernels arising from the leaf shutters of some cameras). We therefore *fit* the function $\sigma_i^2(r)$ under a reasonably limited domain.

Consider a given domain of r (e.g. $r \in [0, 8]$). We select samples equally spaced over the domain with a relatively small interval, say $\Delta r = 0.1$. Then a set of sample PSFs can be generated according to the PSF model, and their blur spectrum $\{\sigma_i^2(r)\}$ can be calculated by equation (6) for each basis function t_i. We note that these discrete samples are only used to generate the continuous fitting functions $\{\sigma_i^2(r)\}_i$.

Then, for each frequency i we fit the following function of r to the samples

$$\sigma_i^2(r) = \exp\left(\alpha_{i,p} r^p + \alpha_{i,p+1} r^{p+1} + \cdots + \alpha_{i,0} r^0 + \cdots + \alpha_{i,q} r^q\right) \qquad (12)$$

For defocus PSFs, blur spectrum are likely to be close to zero in some domain locations, in which case a mild fitting error may be exaggerated when calculating the likelihood (5). So an exponential function is used here to promote the fitting

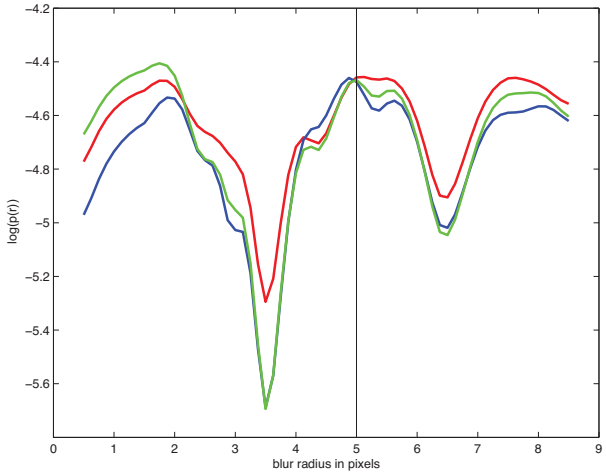


Fig. 6. $\log(p)$ versus r at three different points in the image of Fig. 5 in blurred with disc of radius $r = 5$. The black verical line shows the location of the true blur radius.

accuracy for the small values. A least squares criterion is used to get the best fitting function $\tilde{\sigma}_i^2(r)$ for each frequency basis function t_i.

Once the function set $\{\tilde{\sigma}_i^2(r)\}_i$ is available, given a fixed s, the optimal $\hat{r}$ can be generated by maximizing function (5), or equivalently by minimizing the following:

$$\hat{r} = \arg\min_r \sum_i \left(\frac{|g_i^{\nabla}[\mathbf{x}]|^2}{s\tilde{\sigma}_{hi}^2(r) + \sigma_{ni}^2} + \log\left(s\tilde{\sigma}_{hi}^2(r) + \sigma_{ni}^2\right) \right) \tag{13}$$

Because $\tilde{\sigma}_{hi}^2(r)$ is differentiable, (13) can be optimized through a gradient descent algorithm. Here a steepest descent procedure is employed.

However, in a gradient descent optimization process calculating the spectrum values and their derivatives directly from function (12) is computationally expensive, since this process needs to be carried out for every frequency basis function at every pixel. To reduce the cost, look-up tables, which store the spectrum values, their first and second order derivatives, are employed to replace the runtime computation. Experiments show that using look-up tables takes only one-tenth the time of the runtime computation.

We can also generate these tables directly from the analytic description of the blur model, if it is available. However, in practice we may not have a parametric model for the PSFs of a given lens. In the calibration step it is impractical to get a huge amount of PSF samples to generate the dense look-up tables. It is easier to collect fewer PSF samples through calibration, fit the spectrum curves from the sparse samples using function (12), and finally get the dense look-up tables through the fitted curves. Hence the benefit of the fitting and re-sampling. Since the fitting function does not depend on any specific function of h, our system can be implemented given any PSF model indexed by a single scalar r as long as the blur spectrum $\{\sigma_{hi}^2\}_i$ are smooth over r. For example, it can be a model based on data collected from a particular lens used on a particular camera. A new model can be easily implemented in our system by simply replacing the fitting function set $\{\tilde{\sigma}_i^2(r)\}_i$.

The above fixed point iteration process optimizing (5) at each pixel $\mathbf{x}$ is summarized as follows:

1. Set $l = 0$, and initialize r^l.
2. Compute $s^{l+1} = \arg\max_s p\left(\{g_i^{\nabla}[\mathbf{x}]\}_i | h(r^l), s\right)$ by (11).
3. Compute $r^{l+1} = \arg\max_r p\left(\{g_i^{\nabla}[\mathbf{x}]\}_i | h(r), s^{l+1}\right)$ by (13).
4. $l \leftarrow l + 1$.
5. End if stopping criterion is met, otherwise go to Step 2.

This optimization is sensitive to the initial guess r^0 since (5) is non-convex. To cover most local maxima, we make a set of initial values. For example, we choose the integers $1, 2, \cdots, 8$ as the initial guess and run the optimization procedure for all these values, so that most local maxima over the domain $[0, 8]$ could be captured. After such searching step, only the top 3 optimal scales $\{\hat{r}_1, \hat{r}_2, \hat{r}_3\}$ and their corresponding likelihood values $\{\hat{p}_1, \hat{p}_2, \hat{p}_3\}$ are stored for each pixel $\mathbf{x}$. These data will be sent to the following stage.

B. Coherent Map Labeling

This section discusses how to make a coherent blur map based on the previous probability estimation and other constraints (e.g. smoothness). This goal can be achieved by minimizing the following energy function:

$$E(\mathbf{R}) = \sum_{\mathbf{x}} D_{\mathbf{x}}(r_{\mathbf{x}}) + \sum_{(\mathbf{x},\mathbf{v})\in\nu} \lambda_{\mathbf{x},\mathbf{v}} V(r_{\mathbf{x}}, r_{\mathbf{v}}), \tag{14}$$

which includes two major terms: a data term $D_{\mathbf{x}}(r_{\mathbf{x}})$ reflecting fidelity to the previous probability estimation at position $\mathbf{x}$, and a smoothness term $V(r_{\mathbf{x}}, r_{\mathbf{v}})$ regularizing the output. The smoothness parameter $\lambda_{\mathbf{x},\mathbf{v}}$ controls the strength of this constraint, and is adaptive to local image content. ν is the set of pairs of neighboring pixels. In our system, given pixel $\mathbf{x}$, only the 8 surrounding pixels are considered for the smoothness term. $\mathbf{R} = \{r_{\mathbf{x}}\}_{\mathbf{x}}$ denotes a solution over all positions.

Because the data term is highly non-convex, estimating the optimal solution in the continuous domain is not trivial. To use existing optimization techniques, without introducing too much error, a discrete labeling procedure is carried out. In the blur map labeling problem, labels are discrete r from a finite set φ of possible values. Note that as long as the possible labels within the required range are sufficiently dense, we can still get a good approximation to the continuous solution.

It may seem strange that we went though considerable effort in the preceding local probability estimation to obtain the exact r for the top three local maxima in the continuous domain and now switch to a discrete domain for r in this phase. However, the effort to estimate r in the continuous domain is not wasted. The values attained at various local maxima in the $p(r)$ function can be very close, as can be seen in Fig. 6. A discrete sampling could miss a local maxima or return a lower $p(r)$ that is actually attained. It is the detection of these local maxima and the values attained at them that are most important; the exact value of r at which the maximum is attained does not require pinpoint accuracy. Thus, the information gained in the preceding probability estimation step will not be lost if we round r, but not $p(r)$, to a discrete value.

Theoretically, the data term should give the fidelity cost of $r_{\mathbf{x}}$ assigning to $\mathbf{x}$ with respect to the likelihood values

from equation (5). However, using the values directly from (5), such as $D_{\mathbf{x}}(r) = -\log p(h(r))$, does not perform well. It is computationally expensive, and it does not give sufficient prominence to the top of local maxima. So in our system for pixel $\mathbf{x}$, given the estimated candidates $\{\hat{r}_1, \hat{r}_2, \hat{r}_3\}$ and their corresponding likelihood values $\{\hat{p}_1, \hat{p}_2, \hat{p}_3\}$ from the first estimation step, an *artificial* discrete likelihood array $\tilde{p}_{\mathbf{x}}(r)$ are made through the following scheme (see Fig. 7):

1. Create an empty array $p_{\mathbf{x}}(r) = 0$, where $r \in \varphi$.
2. Set $p_{\mathbf{x}}(\hat{r}_l) = \hat{p}_l$, $l = 1, 2, 3$.
3. Convolve $p(r)$ with a symmetric 1D kernel κ. Then, normalize $p_{\mathbf{x}}(r) \otimes \kappa$ to sum to 1 to get an array $\tilde{p}_{\mathbf{x}}(r)$.

We set $\kappa = [10^{-20}, 10^{-12}, 10^{-7}, 10^{-3}, 10^{-1}, 1, 10^{-1}, 10^{-3}, 10^{-7}, 10^{-12}, 10^{-20}]$.

This convolution array is just wide enough so that similar, but not exactly equal, r values that are at adjacent pixels do not incur a large penalty in the smoothness, V, term of the energy function.

Finally, we let $D_{\mathbf{x}}(r) = -\log \tilde{p}_{\mathbf{x}}(r)$. Since in the labeling problem only a finite set of labels need to be considered, such an array can sufficiently describe the data function $D_{\mathbf{x}}(r)$.

A simple and efficient V function for creating a coherent blur map is

$$V(r_{\mathbf{x}}, r_{\mathbf{v}}) = |r_{\mathbf{x}} - r_{\mathbf{v}}| \tag{15}$$

The bigger the difference between the scales, the larger the penalty becomes. There is zero cost to setting adjacent pixels with the same scale value. This smoothness term can reduce the noise effect in the data term, correcting the errors caused in the first probability estimation stage.

However, such smoothness constraint may also blur the boundaries between different focus planes. To encourage the discontinuity of the blur map to fall along object edges, we define the smoothness parameter as:

$$\lambda_{\mathbf{x},\mathbf{v}} = \lambda_0 \exp\left(-\frac{\|\mathbf{I}_{\mathbf{x}} - \mathbf{I}_{\mathbf{v}}\|^2}{2\sigma_\lambda^2}\right) \tag{16}$$

Here λ_0 is a global parameter controlling the overall strength of the smoothness term. $\mathbf{I}_{\mathbf{x}}$ is a 3×1 vector containing the RGB values of pixel $\mathbf{x}$ of the input color image. Color is an important and effective feature for object distinguishing, because different objects tend to have different colors. $\|\mathbf{I}_{\mathbf{x}} - \mathbf{I}_{\mathbf{v}}\|^2$ measures the color difference between $\mathbf{x}$ and $\mathbf{v}$. σ_λ is another tuning parameter. In general, the value of $\lambda_{\mathbf{x},\mathbf{v}}$ decreases if the color distance between pixel $\mathbf{x}$ and $\mathbf{v}$ is large, protecting the boundaries between the objects in different focus planes.

In our system, α-expansion is used to minimize the energy function (14) [11].

1) Foreground/background segmentation:
Besides blur map labeling, another interesting application of the proposed coherent map estimation method is foreground/background segmentation, which labels the infocus foreground subject from the rest of the input image. However, in this case only a binary labeling map is required. This goal can be easily achieved using the same labeling form as (14):

$$E(\phi) = \sum_{\mathbf{x}} D_{\mathbf{x}}(\phi_{\mathbf{x}}) + \sum_{(\mathbf{x},\mathbf{v}) \in \nu} \lambda_{\mathbf{x},\mathbf{v}} V(\phi_{\mathbf{x}}, \phi_{\mathbf{v}}), \tag{17}$$

where $\mathbf{\Phi} = \{\phi_{\mathbf{x}}\}$ denotes a binary labeling solution. $\phi = 0$ is the blurry label, and $\phi = 1$ is the in-focus label. The data term in this case can be simplified as:

$$D_{\mathbf{x}}(0) = -\log \max_{r > \tau} \tilde{p}_{\mathbf{x}}(r),\ D_{\mathbf{x}}(1) = -\log \max_{r \leq \tau} \tilde{p}_{\mathbf{x}}(r) \tag{18}$$

where τ represents the in-focus threshold. So if there exists a large blur ($r > \tau$) with high probability, then there is a low cost $D_{\mathbf{x}}(0)$ of labeling pixel $\mathbf{x}$ as blurry. Similarly, if there exists a small blur $r \leq \tau$ with high probability, then there is a low cost $D_{\mathbf{x}}(1)$ of labeling pixel $\mathbf{x}$ as sharp.

The smoothness term here is defined the same as (15). Again, we only use the 8 surrounding pixels for ν.

IV. Experiments

Both simulated and real data experiments are carried out to test the performance of the proposed defocus blur estimation framework. In the local probability estimation step, we use square windows with side length $N = 41$. Our default noise setting is $\sigma_n^2 = 10^{-4}$. The coherent blur maps choose the blur radius r from the set $\{0, 0.1, 0.2, \ldots, 7.9, 8\}$. Our default parameter settings for the coherent blur labeling are $\lambda_0 = 20$ and $\sigma_\lambda = 0.1$ (for intensities in the range $[0, 1]$). The settings for the binary foreground/background segmentation problem are $\tau = 2$, $\lambda_0 = 1000$ and $\sigma_\lambda = 0.04$.

Unless otherwise noted, the default parameter values are used. As can be seen in the results, the default settings work well for nearly all the test images shown in this section. In fact, the only parameter we varied in these experiments is the noise variance σ_n^2. In a few of the examples presented here, we found it useful to set $\sigma_n^2 = 10^{-6}$ (very low noise).

A. Simulated Experiments

A simulated experiment is illustrated in Fig. 8, which allows us to quantitatively test the performance of the proposed method. The input image is generated according to the model in (1). Similar to the test in Fig. 3 disc functions are employed to simulate defocus PSFs. Variance of the additive white Gaussian noise is $\sigma_n^2 = 1 \times 10^{-6}$. The latent blur map is given in (c), where the blur radius continuously changes over the image space. This actually violates the assumption that local blur $h_{\mathbf{x}}$ is constant within local analysis window η. However, the proposed output seems to be robust to the violation of this assumption (see Fig. 8 (d)): trend of the blur change is successfully captured by our method. This is probably because we use overlapping windows. The mean-squared-error (MSE) of (d) with respect to the latent map (c) is 0.022.

Based on the blur map estimation $\{\hat{r}_{\mathbf{x}}\}$, and further the PSFs $\{\hat{h}_{\mathbf{x}}\}$ generated by the disc function, a spatially varying deconvolution procedure is carried out through the following optimization:

$$\hat{f} = \arg\min_f \sum_{\mathbf{x}} \left| \left(\hat{h}_{\mathbf{x}} \otimes f\right)[\mathbf{x}] - g[\mathbf{x}] \right|^2 + \lambda \sum_{\mathbf{x}} \|\nabla f[\mathbf{x}]\|_1 \tag{19}$$

The deblurred output $\hat{f}$ is given in Fig. 8 (b), where we can observe that spatially varying blurs have been successfully removed (see zoomed parts (e)-(h)). The peak signal-to-noise

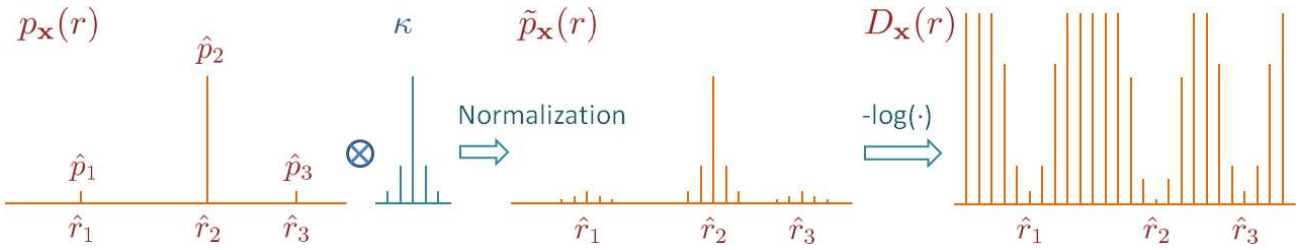


Fig. 7. Making the artificial likelihood array.

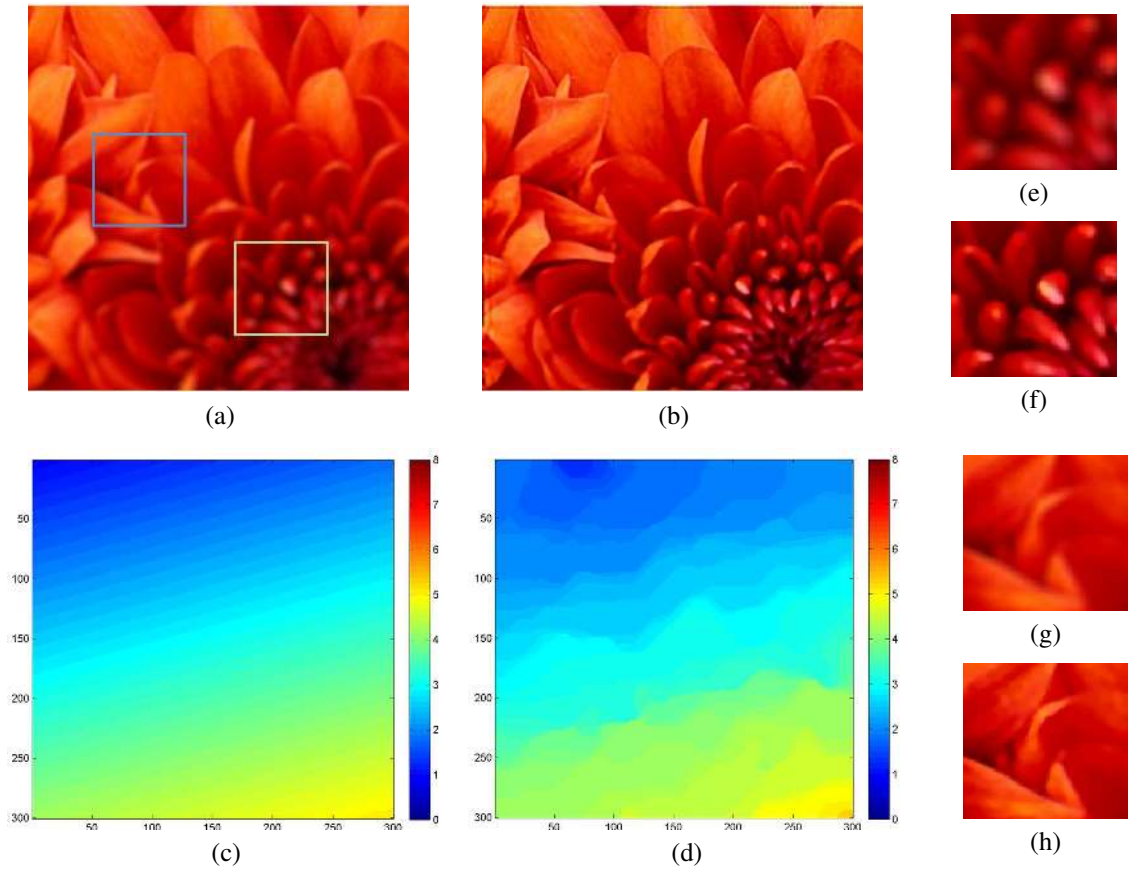


Fig. 8. Simulated experiment for blur map estimation and spatially varying blind deblurring. (a): Simulated input image with spatially varying blur, whose PSNR is 27.3dB. (b): Deblurred image based on the proposed estimate in (d). Its PSNR is 32.3dB. (c): Latent blur map. (d): Estimated blur map. (e), (g): zoomed part of (a). (f), (h): zoomed part of (b).

ratio (PSNR) of the original input (a) is 27.3dB, whereas the PSNR of (b) is 32.3dB with 5dB improved, which means in this experiment the accuracy of our estimation method is good enough for *blind* deblurring.

B. Real Data Experiments

Real image experiments are given in Fig. 1 and Fig. 9. Because these data are collected from outside sources, the corresponding calibrated defocus PSFs are not available. However, it is known that blur from an ideal lens with a circular aperture could be modeled by the disc function in the absence of diffraction effects [12]. Since diffraction effects are almost always negligible once the blur is of visible size, we use the disc function to approximate the real PSFs in our experiments. Even though the actual blur PSF for cameras used for the test images are unknown, the disc approximation seems to be quite adequate. Our method still captures the amount of local defocus blur for all these test images, depicts 3-D geometric information for each scene, and does a good job in identifying in-focus subjects.

For example, Fig. 1 (a) contains four focal layers: the in-focus herdsman, the slightly defocused cattle, the background mountain and the highly blurry sky. These layers are all reflected in the output blur map (b), and the in-focus herdsman is also correctly labeled in (c). In Fig. 9 (a) the lizard and part of the rock are in-focus, which are correctly identified and labeled by Fig. 9 (b)-(c). Note that here we are not doing pure object segmentation, and that the segmentation only depends on local sharpness level (which means we are not trying to segment the lizard only from the rest of the image). Fig. 9 (j) illustrates another example with the blur smoothly varies over the space. Again, our blur map captures the progression of out-of-focus to in-focus to out-of-focus along the correct angle (lower left to upper right of the image) on the wood

(see Fig. 9 (k)).

Next we show many additional real examples further demonstrating that our method works well on a broad range of inputs. Fig. 10 shows more examples of defocus blur maps and in-focus segmentations computed using our method. Fig. 11 shows even more results of our automatic binary segmentation algorithm into in-focus and out-of-focus regions.

We obtain high quality results on outdoor scenes with natural objects such as animals and flowers in Fig. 10(a),(b) and Fig. 11(a),(b),(c). The llama example in Fig. 11(e) is one for which segmentation based on color would be difficult since the foreground and background have similar colors. Another such example is shown in Fig. 10(c) where the sprinkler and the ground are the same color.

In Fig. 11(d), note how our automatic in-focus segmentation correctly captures the depth of field for this shot. The result in Fig. 11(f) correctly segments the sharp background from the blurry face. The very jagged boundary near the edge of the glasses is due to the graph cut segmentation following the details of the background texture to place the segmentation boundary along strong image edges.

Finally, in Fig. 12 we show some comparisons with the blur maps produced by Defocus Magnification [9] (DM) on some examples in [9]. In the DM blur maps in the middle column of Fig. 12, the whiter the pixel the larger the standard deviation in their fitted Gaussian blur model and the higher the predicted blurriness. The DM approach estimates the blur only at image edges and then propagates the sparse blur estimates to the rest of the image by assuming pixels of similar intensity and color have similar blurriness.

In general, the DM blur estimation method tends to show the underlying image edges in places where the blur measure is actually smooth. Examples of this in the center column of Fig. 12(a) include the nose of the dog, and the boundary of the legs of the stuffed animals. Our blur estimates are (correctly) much smoother in these areas. In Fig. 12(b), the DM result on the grass to the left of the subject has the same level of blur as the subject, while ours captures the distinct blur levels of the subject, the grass closer to the subject, and the patch of white flowers further back. Both methods do well on the *cup* example in Fig. 12(c), but our result has crisper blur discontinuities.

Our result in Fig. 12(d) is much better than the DM result. DM has blur discontinuities within the hands (e.g. between the pinky and the other hand on the left of the image) and between the sidewalk and grass in the upper right where the blur should be smooth. Indeed our blur estimates are smoother in these areas, and we correctly identify the entirety of both hands as in-focus.

For all of these examples, our boundaries between in-focus foreground and out-of-focus background are much more sharply delinated. Also, the DM approach results in splotchy estimates which lack the smoothness of the true blur. The DM blur maps show the limitations of making a binary decision as to where to compute the blur estimate, using only high frequencies in the blur model, and subsequently interpolating/propagating to obtain a dense answer.

V. Problems

One problem with our method for estimating blur maps is that we do not explicitly model the case where a window overlaps areas with different blur scales. In this case we have found that the sub-area of a window with the smallest blur size tends to dominate the probability analysis, as this sub-area contributes more power per pixel to the power spectrum. Thus, sharp areas would be enlarged by the radius of the analysis window in a blur map produced without the coherence labeling step. We rely on the coherence labeling step to snap the boundary back to the closest color boundary in the underlying image, which is usually where the actually depth discontinuity lies. But in cases of gradually changing blur, or in cases where there is not a good color boundary at the depth discontinuity, this may fail.

For the coherence labeling step to have the above correcting influence we have to set the λ parameter to a significant value. This can cause a slight over smoothing of the blur values and manifests as the blur values taking discrete steps in areas where the blur changes continuously as seen in Fig.8. The jaggedness of the contours in the same figure is partially a result of the inherent uncertainty present in the statistical computation.

The same comments apply to the foreground/background segmentation case, in that we are relying on coherence to snap the foreground mask to the nearest color boundaries in the image. Generally the results are quite good, but we occasionally see problems. Because there is increased energy to follow a serpentine contour sometimes long, thin parts of the subject of interest are cut off. See for example the feathers of the wings in Fig. 10(a) and Fig. 11(c).

VI. Conclusion

In this paper we proposed a method estimating a defocus blur map from a single image. It is capable of measuring the probability of local blur scale in the continuous domain by analyzing the *localized Fourier* (Gabor filtering) spectrum. For each analysis window, not only the global optimum maximizing the likelihood function but also a few local optima are detected as candidate scales. Finally, color edge information and smoothness constraints are incorporated into the system to select the best candidates and generate a coherent map of the blur scale at each pixel. Experiments show that this method can be used to approximate geometric information of the input, help remove spatially varying blur, and segment infocus subjects from defocused background.

Currently our MATLAB implementation takes around 10 minutes to process a 500×500 image using a PC with a 2.70GHz CPU. Efficiency could be improved through C++ implementation. The runtime can be reduced further by using parallel computation. For example, the local optima searching process starting from the 8 different initial r values described in Section III-A could be done in parallel.

Future research will also focus on improving the statistical model of the latent sharp image. Currently latent image gradients are assumed to be Gaussian distributed. However, it has been discovered that for natural images "heavy-tailed"

distribution models are more proper. Such distributions can be approximated using a Gaussian mixture model. Estimation accuracy may be enhanced by incorporating this model.

REFERENCES

[1] R. Fergus, B. Singh, A. Hertsmann, S. T. Roweis, and W. T. Freeman, "Removing camera shake from a single image," *SIGGRAPH*, 2006.

[2] Q. Shan, J. Jia, and A. Agarwala, "High-quality motion deblurring from a single image," *ACM Transactions on Graphics (SIGGRAPH)*, 2008.

[3] A. Levin, Y. Weiss, F. Durand, and W. T. Freeman, "Understanding and evaluating blind deconvolution algorithms," *CVPR*, 2009.

[4] L. Xu and J. Jia, "Two-phase kernel estimation for robust motion deblurring," *ECCV*, 2010.

[5] O. Whyte, J. Sivic, A. Zisserman, and J. Ponce, "Non-uniform deblurring for shaken images," *CVPR*, 2010.

[6] A. N. Rajagopalan, S. Chaudhuri, and U. Mudenagudi, "Depth estimation and image restoration using defocused stereo pairs," *PAMI*, vol. 26, no. 11, pp. 1521–1525, Nov. 2004.

[7] A. Levin, R. Fergus, F. Durand, and W. T. Freeman, "Image and depth from a conventional camera with a coded aperture," *SIGGRAPH*, 2007.

[8] A. Chakrabarti, T. Zickler, and W. T. Freeman, "Analyzing spatially-varying blur," *CVPR*, 2010.

[9] S. Bae and F. Durand, "Defocus magnification," *Computer Graphics Forum*, vol. 26, no. 3, pp. 571–579, 2007.

[10] D. Zoran and Y. Weiss, "Scale invariance and noise in natural images," *IEEE International Conference on Computer Vision*, Sep. 2009.

[11] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *ICCV*, 1999.

[12] M. Potmesil and I. Chakravarty, "Synthetic image generation with a lens and aperture camera model," *ACM Transactions on Graphics*, vol. 1, no. 2, April 1982.

[13] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott, "A perceptually motivated online benchmark for image matting," in *CVPR*, June 2009. [Online]. Available: http://www.alphamatting.com/

Xiang Zhu (S'08) received the B.S. and M.S. degrees in electrical engineering from Nanjing University, Nanjing, China, in 2005 and 2008, respectively, and is currently pursuing the Ph.D. degree in electrical engineering at the University of California, Santa Cruz.

His research interests are in the domain of image processing (denoising, deblurring, super-resolution, and image quality assessment).

Scott Cohen received the B.S. degree in mathematics from Stanford University, Stanford, CA, in 1993, and the B.S., M.S., and Ph.D. degrees in computer science from Stanford University in 1993, 1996, and 1999, respectively.

He is currently a Principal Scientist at Adobe Reseach of Adobe Systems Incorporated, San Jose, CA. His research interests include interactive image and video segmentation, image and video matting, stereo, upsampling, and deblurring.

Stephen Schiller received the B.S. degree in mathematics from University of California, Santa Barbara, in 1974, and the M.S. degree in computer science from University of California, Berkeley in 1979.

He is currently a Principal Scientist at Adobe Reseach of Adobe Systems Incorporated, San Jose, CA. His research interests include image segmentation, image de-blurring, synthesis and analysis of textures, and image-to-vector conversion.

Peyman Milanfar Peyman Milanfar is a Professor of Electrical Engineering at UC Santa Cruz, and was Associate Dean for research from 2010 to 2012. He is currently on leave at Google-[x]. He received the B.S. in EE/Mathematics from Berkeley, and his Ph.D. in EECS from MIT. Prior to UCSC, he was at SRI, and a Consulting Professor of CS at Stanford. He founded MotionDSP, which has brought state of the art video enhancement to market. His technical expertise are in statistical signal, image and video processing, computational photography and machine vision. He is a fellow of IEEE.

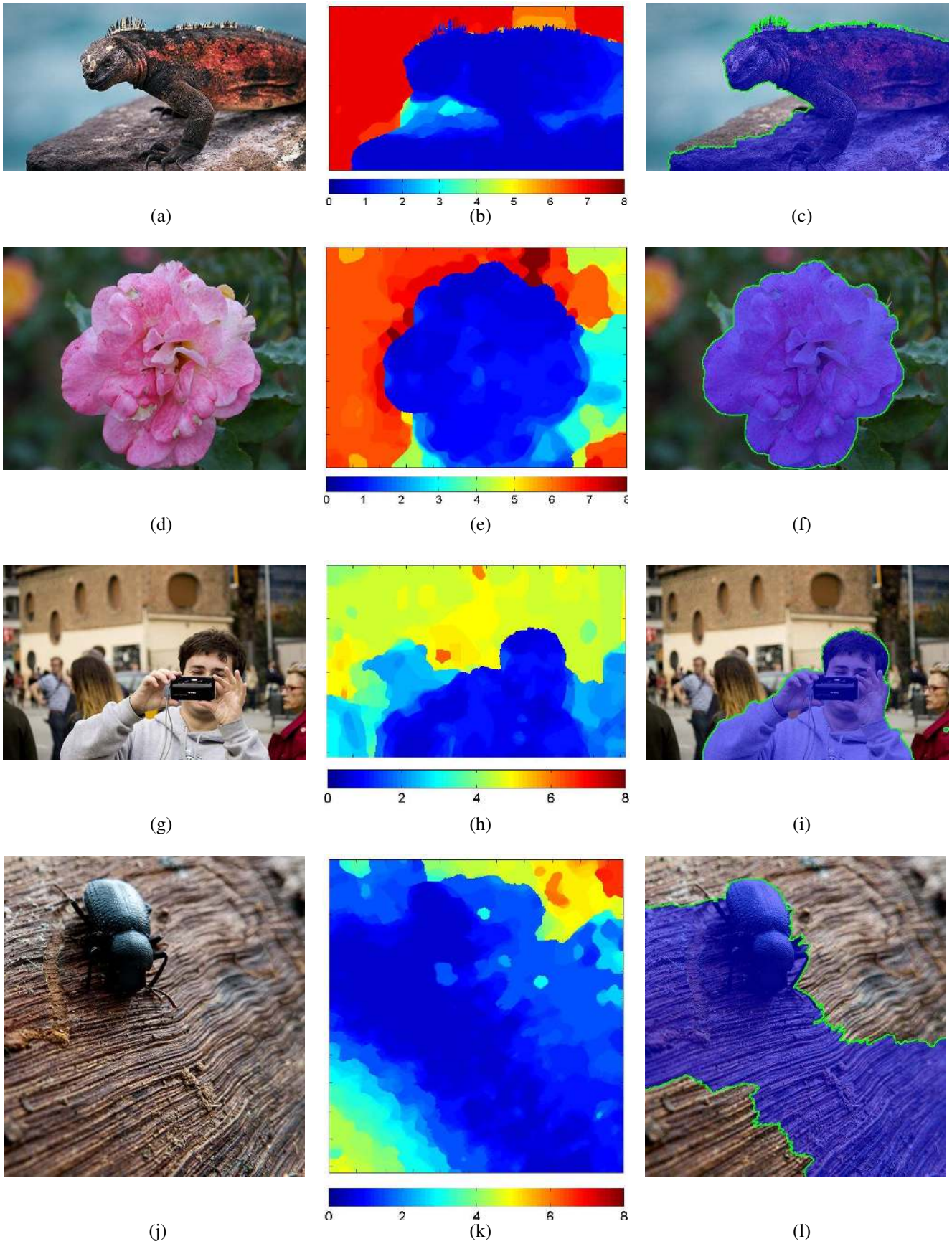


Fig. 9. Defocus blur map estimation experiments using real images. Left column: input images. Middle column: estimated defocus blur maps. Right column: automatic foreground/background segmentation results.

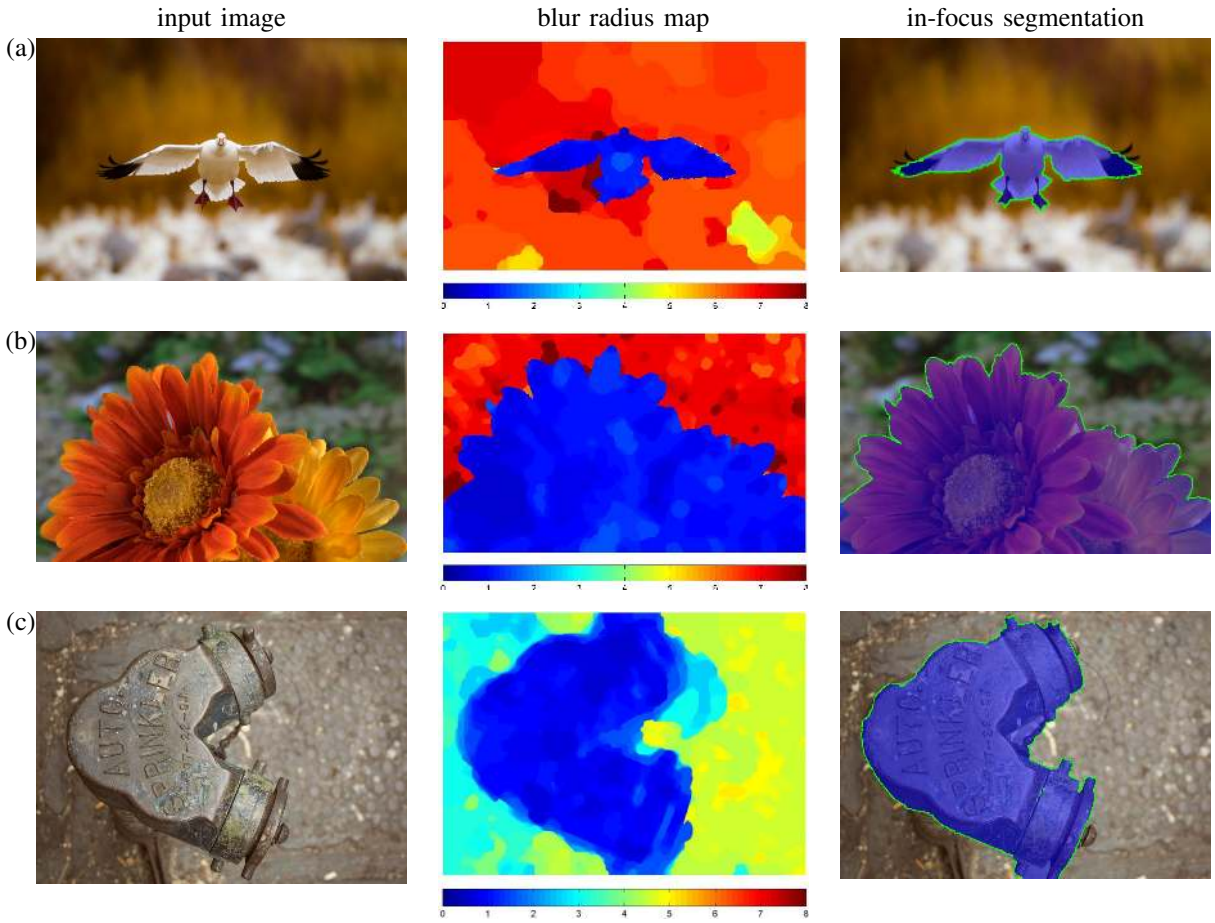


Fig. 10. More blur map results on real images. (a) *bosque-Edit_small*. ©Aravind Krishnaswamy. Used with permission. (b) *input_training_lowres/GT01* [13]. $\sigma_n^2 = 10^{-6}$. (c) *autoSprinkler*. Image courtesy of Katrin Eismann. Segmentation based on color would be difficult in this example.

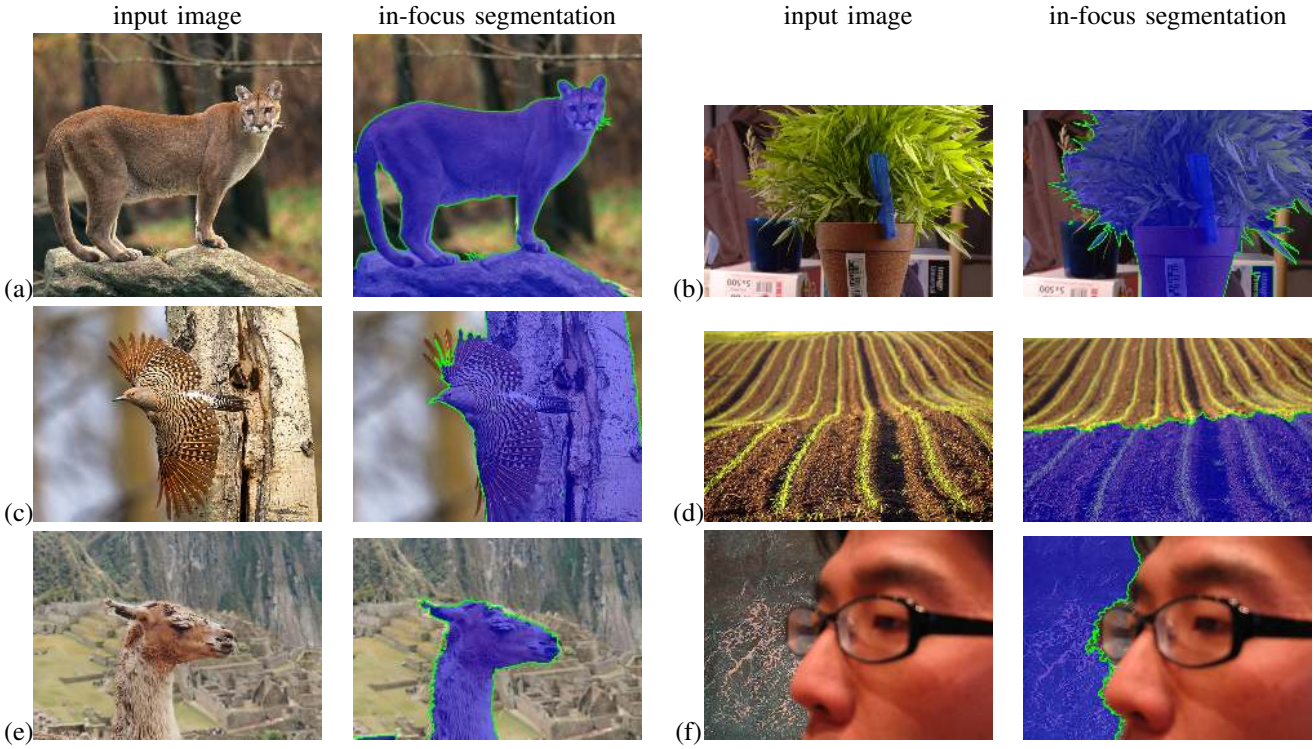


Fig. 11. Automatic In-focus Segmentation Results. (a) *03Meow*. (b) *input_lowres/plant* [13]. (c) *morning-Edit_small*. ©Aravind Krishnaswamy. Used with permission. (d) *field* [9]. (e) *llama*. $\sigma_n^2 = 10^{-6}$. Segmentation based on color would be difficult in this example. (f) *ref_in* [9].

Fig. 12. Comparison with Defocus Magnification [9] blur map results. (a) *IMG_0419*. (b) *man*. (c) *cup*. (d) *hands*.