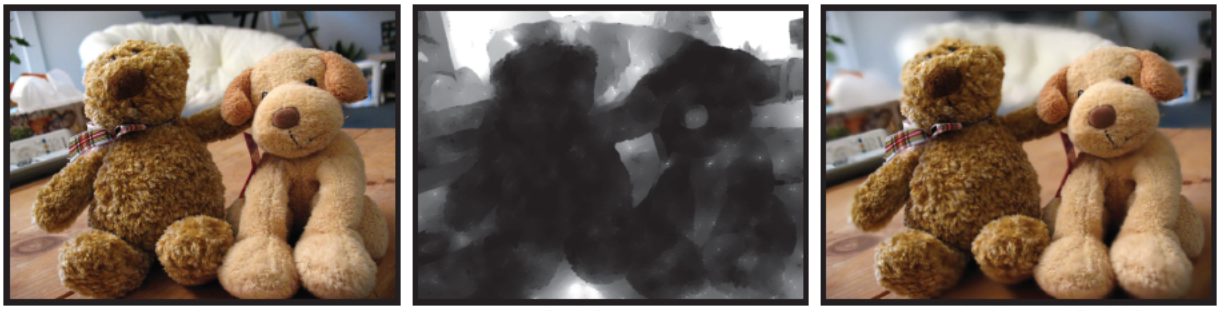
EUROGRAPHICS 2007 / D. Cohen-Or and P. Slavík
(Guest Editors)

*Volume 26 (2007), Number 3*

# Defocus Magnification

Soonmin Bae Frédo Durand

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology



(a) input (b) defocus map (c) our result with magnified defocus

**Figure 1:** *Our technique magnifies defocus given a single image. Our defocus map characterizes blurriness at edges. This enables shallow depth of field effects by magnifying existing defocus. The input photo was taken by a Canon PowerShot A80, a point-and-shoot camera with a sensor size of* $7.18 \times 5.32$ *mm, and a* $7.8$ *mm lens at f/2.8.*

**Abstract**

*A blurry background due to shallow depth of field is often desired for photographs such as portraits, but, unfortunately, small point-and-shoot cameras do not permit enough defocus because of the small diameter of their lenses. We present an image-processing technique that increases the defocus in an image to simulate the shallow depth of field of a lens with a larger aperture.*

*Our technique estimates the spatially-varying amount of blur over the image, and then uses a simple image-based technique to increase defocus. We first estimate the size of the blur kernel at edges and then propagate this defocus measure over the image. Using our defocus map, we magnify the existing blurriness, which means that we blur blurry regions and keep sharp regions sharp. In contrast to more difficult problems such as depth from defocus, we do not require precise depth estimation and do not need to disambiguate textureless regions.*

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Applications

## 1. Introduction

Sharp foreground with blurred background is preferred in many types of photography such as portraits. But point-and-shoot cameras have small lenses and sensors, which fundamentally limits their ability to defocus the background and generate shallow depth of field. We present an image-processing technique that magnifies existing defocus given a single photo.

For a given field of view and subject distance, depth of field is directly related to the physical diameter of the lens aperture. This means that compact cameras that rely on smaller sensors – and therefore on smaller lenses – yield less defocus and cannot blur the background the way a large-aperture single-lens reflex (SLR) lens can (Fig. 2). While a smaller amount of defocus (larger depth of field) can be desirable, for example in landscape or macro photography, it is often a serious limitation for portraits and creative photography. Users of compact cameras often complain that their portraits do not look "artistic" and lack the clarity afforded by defocused backgrounds. In fact, the quality of a blurry

background, called *bokeh*, has a real cult following among some photographers.

(a) large sensor (22.2 x 14.8 mm) (b) small sensor (7.18 x 5.32 mm)

**Figure 2:** *Given the same field of view and the same f-number (f/2.8), a large sensor (a) yields more defocus than a small sensor (b) does.*

Our technique takes a single input image where the depth of field is too large and increases the amount of defocus present in out-of-focus regions. That is, our goal is opposite to that of work that seeks to create images that are sharp everywhere.

Our approach first estimates the spatially-varying amount of blur over the image, and then uses an off-the-shelf image-based technique to increase defocus. We first estimate the size of the blur kernel at edges, building on the method by Elder and Zucker [EZ98], and then propagate this defocus measure over the image with a non-homogeneous optimization. Using our defocus map, we can magnify the existing blurriness, which means that we further blur blurry regions and keep sharp regions sharp.

Note that in contrast to more difficult problems such as depth from defocus, we do not require precise depth estimation and do not need to accurately disambiguate smooth regions of the image, since such regions are not much affected by extra blur due to defocus. The fundamental ambiguity between out-of-focus edges and originally smooth edges is out of the scope of our work. We also do not need to disambiguate between objects in front and behind the plane of focus. We simply compute the amount of blur and increase it. While our method does not produce outputs that perfectly matches images captured with a larger-aperture lens, it qualitatively reproduces the amount of defocus. We refer interested readers to Appendix A where we review thin-lens optics and defocus.

### 1.1. Related work

Defocus effects have been an interest of the Computer Vision community in the context of recovering 3D from 2D. Camera focus and defocus have been used to reconstruct depth or 3D scenes from multiple images: depth from focus [Hor68, DW88, EL93, NN94, HK06] and depth of defocus [Pen87, EL93, WN98, FS02, JF02, FS05]. These methods use multiple images with different focus settings and estimate the corresponding depth for each pixel. They have to know the focus distance and focal length to computer the depth map. In contrast, we do not estimate the depth but the blur kernel. We want to treat this problem without the help of any special camera settings, but only with image post-processing techniques.

Image processing methods have been introduced to modify defocus effects without reconstructing depth. Eltoukhy and Kavusi [EK03] use multiple photos with different focus settings and fuse them to produce an image with extended depth of field. Özkan et al. [OTS94] and Trussell and Fogel [TF92] have developed a system to restore space-varying blurred images and Reeves and Mersereau [RM92] find a blur model to restore blurred images. This is the opposite of what we want to do. They want to restore blurred images, while we want to increase existing blurriness.

Kubota and Aizawa [KA05] use linear filters to reconstruct arbitrarily focused images from two differently focused images. On the contrary, we want to modify defocus effects only with a single image. Lai et al. [LFC92] use a single image to estimate the defocus kernel and corresponding depth. But their method only works on an image composed of straight lines at a spatially fixed depth.

Given an image with a corresponding depth map, depth of field can be approximated using a spatially-varying blur, e.g. [PC81, BHK*03], but note that special attention must be paid to occlusion boundaries [BTCH05]. Similar techniques are now available in commercial software such as Adobe® Photoshop® (lens blur) and Depth of Field Generator Pro (*dofpro.com*). In our work we simply use these features and instead of providing a depth map, we provide a blurriness map estimated from the photo. While the amount of blurriness is only related to depth and is not strictly the same as depth, we have found that the results qualitatively achieve the desired effect and correctly increase defocus where appropriate. Note that a simple remapping of blurriness would yield a map that resembles more closely a depth map.

## 2. Overview of Our Approach

For each pixel, we estimate the spatially-varying amount of blur. We call our blur estimation the *defocus map*. We estimate the defocus map in two steps. First, we estimate the amount of blur at edges. Then, we propagate this blur measure to the rest of the image.

We model an edge as a step function and the blur of this edge as a Gaussian blurring kernel. We adapt the method by Elder and Zucker [EZ98], which uses multiscale filter responses to determine the size of this kernel. We add a cross-bilateral filtering step [ED04, PAH*04] to remove outlier estimates.

We propagate the blur measure using non-homogeneous optimization [LLW04]. Our assumption is that blurriness varies smoothly over the image except where the color is dis-

continuous. We propagate blurriness measure to the neighbors with similar intensity and color.

We can use our defocus map to magnify defocus effects. We blur each pixel according to its estimated blurriness. If we double our defocus map, it doubles defocus effects as if the image is taken with an aperture that is twice as large. In this paper, our results are generated using Adobe® Photoshop® lens blur with our defocus map as a depth map.

## 3. Blur Estimation

The amount of blur can be estimated reliably only in areas of an image that has significant frequency content. This is why we focus on edges. However, we need to extract and analyze edges with various levels of blurriness, which makes the technique by Elder and Zucker [EZ98] particularly appropriate. We refine their technique by introducing the explicit fitting of a blurred edge model that is more robust than the original technique. Also, our refinement step reduce outliers due to blurry features such as soft shadows.

### 3.1. Detect blurred edges

Following Elder and Zucker, we model an edge as a step function in intensity, and the blur of this edge as a Gaussian blurring kernel:

$$g(x,y,\sigma_b) = \frac{1}{2\pi\sigma_b^2}\exp(-(x^2+y^2)/2\sigma_b^2) \qquad (1)$$

where $\sigma_b$ denotes the scale of the blur, and is what we want to estimate.

For each pixel, Elder and Zucker determine the right scale for edge detection using the noise thresholds. More details can be found in the Appendix B. We used $\sigma_1 \in \{64\ 32\ 16\ 8\ 4\ 2\ 1\ 0.5\}$ pixels and $\sigma_2 \in \{32\ 16\ 8\ 4\ 2\ 1\ 0.5\}$ pixels. We apply a strict threshold, $s_n = 2.5$ and $\alpha_I = 0.0001\%$, to achieve very reliable blur estimation.

### 3.2. Estimate blur

In their technique, Elder and Zucker estimate the amount of blur by measuring the distance $d$ between second derivative extrema of opposite sign in the gradient direction. This directly follows from the analytical derivation of a perfect step edge convolved with a Gaussian, as shown in their paper [EZ98].

However, we have found that, for real images, the localization of the second-derivative extrema of the edge using the zero-crossing of the third derivative is not robust, which is acknowledged in their article. This leads to errors in the estimation of the blur amount. Therefore, instead of measuring the distance between actual extrema, we fit the multiscale models of the second derivative Gaussian filter response to the pixel responses and find the distance with a least square fitting error. Given the estimated distance, we compute the size of blur kernel $\sigma_b$ using Equation 2 (d). This provides us with a sparse set of blur measures $BM$ at edge pixels in the image.

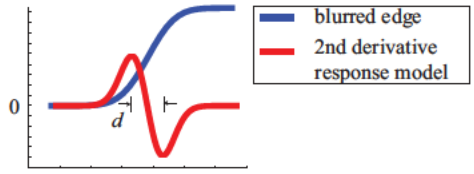



**Figure 3:** *The model for the distance between second-derivative extrema. We numerically fit this response model with various d around the edge pixel and along the gradient direction to find the distance d with a least square fitting error.*

We fit the response model using a brute-force strategy. We fit the response model with a number of values for distance $d$ (Fig. 3) to a window around the edge pixel and along the gradient direction. Elder and Zucker use an edge pixel at the dark side of the edge. But we found that using both bright and dark sides of the edge generates more reliable defocus maps. We use window sizes from $3 \times 3$ to $71 \times 71$. Given a blurred step edge along the y axis of amplitude $A$ and blur parameter $\sigma_b$, the expected response to the second derivative filter is modeled by:

$$r_2^x(x,y,\sigma_2) = Au(x) * g_2^x(x,y,\sigma_b^2+\sigma_2^2) \qquad (2a)$$

$$= \frac{-Ax}{\sqrt{2\pi}(\sigma_b^2+\sigma_2^2)^{3/2}}\exp(-x^2/2(\sigma_b^2+\sigma_2^2)) \qquad (2b)$$

$$= \frac{-Ax}{\sqrt{2\pi}(d/2)^3}\exp(-x^2/2(d/2)^2) \qquad (2c)$$

$$\text{with: } (d/2)^2 = \sigma_b^2+\sigma_2^2 \qquad (2d)$$

where $u(x)$ is a step function. We derive $A$ from the local extrema within each window.

Figure 4 shows that our approach can successfully estimate blur measures while the zero-crossing of the third derivative cannot localize the second derivative extrema.

### 3.3. Refine blur estimation

Depth of field effects are not the only cause of edge blurriness in images and phenomena such as soft shadows and glossy highlights can result in erroneous estimates of defocus.

We suppress the influence of these outliers by smoothing the blur measure with an edge-preserving filter. We apply cross bilateral filtering [ED04, PAH*04] to our sparse set of blur measures, $BM$. The cross-bilateral filtering output is a weighted mean of its neighbors where the weights decrease

(a) input (b) actual blur sigma (c) the zero-crossing of the third derivative (d) blur measure using our approach

**Figure 4:** *The zero-crossing of the third derivative (c) is greatly affected by neighboring edges and cannot localize the second derivative extrema. In contrast, our approach (d) can estimate the blur sigma that is close to the actual blur sigma (b). The input (a) is generated using the blur sigma (b).*

with the distance in space and with the range difference of a reference image.

In addition to the original cross bilateral filtering weights, we use a sharpness bias, $b(BM) = \exp(-BM/2)$. The sharpness bias corrects blur measures in soft shadows and glossy highlights that are higher than they are supposed to be.

With $g_\sigma(x) = \exp(-x^2/2\sigma^2)$, a Gaussian function, we define the biased cross bilateral filtering of a sparse set of blur measures, $BM$ at an edge pixel $\mathbf{p}$ as the following:

$$bCBF(BM)_\mathbf{p} = \frac{1}{k}\sum_{\mathbf{q}\in BM} w_\mathbf{pq}\, b(BM_\mathbf{q})\, BM_\mathbf{q} \tag{3a}$$

$$\text{with:}\quad w_\mathbf{pq} \propto \sum_{\mathbf{i}\in\{R,G,B\}} g_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, g_{\sigma_r}(|C_i(\mathbf{p})-C_i(\mathbf{q})|) \tag{3b}$$

$$\text{and}\quad k = \sum_{\mathbf{q}\in BM} w_\mathbf{pq}\, b(BM_\mathbf{q}) \tag{3c}$$

where $\sigma_s$ controls the spatial neighborhood, and $\sigma_r$ the influence of the intensity difference, and $k$ normalizes the weights. We use the RGB color channels of the original input image as the reference and set $\sigma_r = 10\%$ of the image range and $\sigma_s = 10\%$ of the image size. This refinement process does not generate much change but refines a few outliers as shown in Figure 5. The cross bilateral filtering refines outliers such as yellow and green measures (b) in the focused regions to be blue (c).

## 4. Blur Propagation

Our blur estimation provides blur kernels only at edges and we need to propagate this blur measure. We use non-homogeneous optimization [LLW04] and assume that the amount of defocus is smooth when intensity and color are smooth.

### 4.1. Propagate using optimization

Our propagation is inspired by the *colorization* paper by Levin et al. [LLW04]. We impose the constraint that neighboring pixels $\mathbf{p},\mathbf{q}$ have similar blurriness if they have similar intensities and colors. We minimize the difference between the blurriness $B(\mathbf{p})$ and a weighted average of blurriness of neighboring pixels:

$$E(B) = \sum(B(\mathbf{p}) - \sum_{\mathbf{q}\in N(\mathbf{p})} w_\mathbf{pq} B(\mathbf{q}))^2 \tag{4a}$$

$$+ \sum \alpha_\mathbf{p}\,(B(\mathbf{p}) - BM(\mathbf{p}))^2 \tag{4b}$$

$$\text{with:}\quad w_\mathbf{pq} \propto \sum_{\mathbf{i}\in\{R,G,B\}} exp(\frac{-(C_i(\mathbf{p})-C_i(\mathbf{q}))^2}{2\sigma_{i_\mathbf{p}}^2}) \tag{4c}$$

where $\sigma_\mathbf{p}$ is the standard deviation of the intensities and colors of neiboring pixels in a window around $\mathbf{p}$. The window size used is $7\times 7$. We have experimented both with setting the second term as hard constraints vs. as a quadratic data term, and have found that the latter is more robust to potential remaining errors in the blur measure.

We solve this optimization problem by solving the corresponding sparse linear system. Figure 6 shows the defocus map for various values of $\alpha$. We use $\alpha = 0.5$ for edge pixels.

## 5. Results

We have implemented our blur estimation using Matlab. Our defocus map enables defocus magnification. We rely on Photoshop's lens blur to compute the defocused output. We crop the upper and lower 5% of the defocus map and clamp its minimum value to 0. In addition, we apply Gaussian blur to the defocus map to use it as a depth map. The Gaussian blur radius is set to 0.5% of the image size.

Using our defocus map, we can simulate the effect of doubling the aperture size. Figure 7 compares two input defocus maps of two images with the f-number 8 (a) and 4 (b). As we double the defocus map (c) of the f/8 image, we obtain a result similar to the defocus map (d) of the f/4 image. While the simulated defocused map (e) is not exactly the same as the real map (d), the output image with magnified defocus (f) is visually close to the f/4 photograph (b).

In Figure 11, we show the results of using our defocus map to magnify the existing defocus effects in the original images. The results preserve the sharpness of the focused regions but increase the blurriness of the out-of-focus regions.

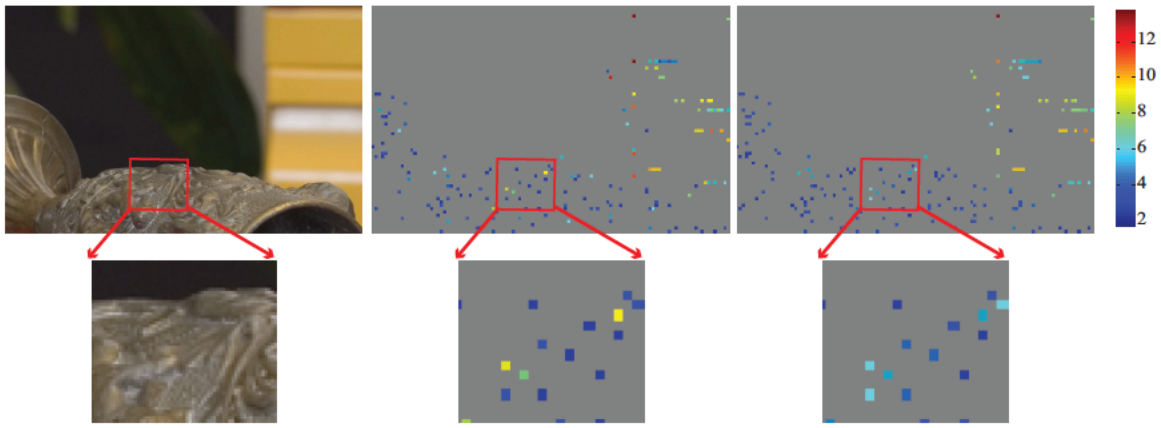



**Figure 5:** *Blur measure before and after the cross bilateral filtering. The cross bilateral filtering refines outliers such as yellow and green measures (b), which mean blurry, in the focused regions to be blue measures (c), which means sharp. The blur measures are downsampled using nearest neighbor for better illustration.*

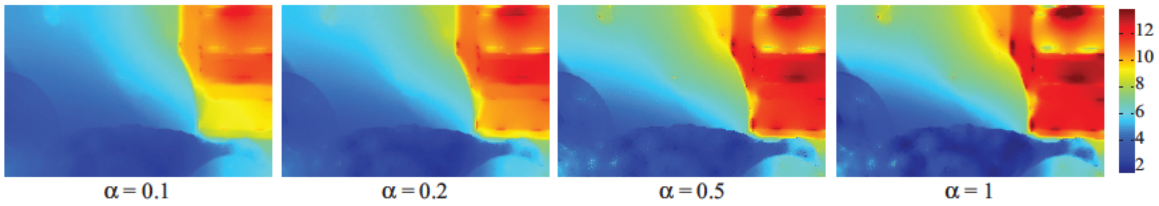



**Figure 6:** *Defocus map with various α. α controls the balance between the smoothness penalty term and data term in Equation 4. We use α = 0.5 for edge pixels and α = 0 for non-edge pixels, which do not have values. In this plot, red means blurry region and blue means sharp regions. The input image is Figure 5.*

In addition, while our defocus map is not really a depth map, it is sometimes possible to use it to refocus a photograph resembling the effect of Ng et al. [NLB*05] and Isaksen et al. [IMG00]. Figure 8 shows a result where our defocus magnification is applied with a virtual focusing distance. Before we apply lens blur, we performed deconvolution using our defocus map. The result looks as if the foreground is focused.

Figure 1 and 8 and the two rows in the middle of Figure 11 were taken by a Canon PowerShot A80, a point-and-shoot camera with a sensor size of $7.18 \times 5.32$ mm, and a 7.8 mm lens at f/2.8. Figure 5 and 7 were taken by a Canon 1D Mark II with a sensor size of $28.7 \times 19.1$ mm and a Canon EF 85mm f/1.2L lens. The first input of Figure 11 was taken by a Nikon D50 with a sensor size of $23.7 \times 15.6$ mm and a 180.0 mm lens at f/4.8. The two rows at the bottom of Figure 11 are from *bigfoto.com*.

### 5.1. Discussion

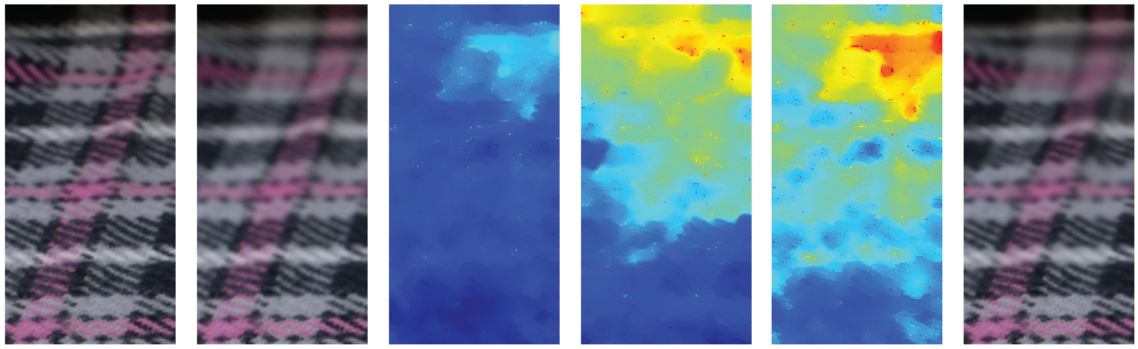
Our defocus maps are different from their actual depth maps mostly in smooth regions of the image that are not much affected by extra blur due to defocus. For example, the gradients in human skin are interpreted as blurry regions. However, such artifacts do not cause visual defects in the results with magnified defocus. You can notice some of these issues in Figure 11.

A limitation of our technique is that occlusion boundaries that separate sharp foreground and blurry background are sometimes erroneously blurred (e.g. the top of the Teddy bear in Fig. 1)

## 6. Conclusions

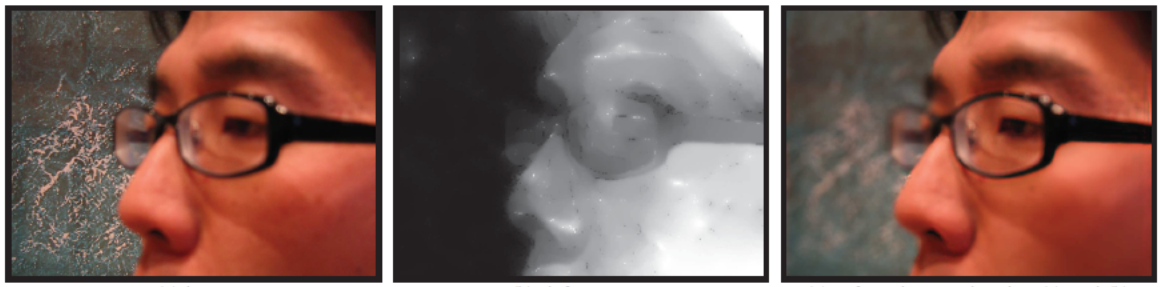
We have presented an image-processing technique to magnify the amount of defocus due to lens aperture. Given a single image, we estimate the size of the blur kernel at edges and propagate the blur measure to the overall image. We use a multiscale edge detector and model fitting to estimate the size of blur kernel. We propagate the blur measure assuming that blurriness is smooth where intensity and color are similar.

Unlike more difficult problems such as depth from defocus, we do not need to generate an accurate depth map and



(a) f/8 input (b) f/4 input (c) f/8 defocus map (d) f/4 defocus map (e) doubling (c) (f) our synthesized result using (a) and (e)

**Figure 7:** *Doubled defocus. Doubling the defocus map generates a effect of doubling the aperture size. As we double the defocus map (c) of the f/8 image, we obtain a result similar to the defocus map (d) of the f/4 image. While the simulated defocused map (e) is not exactly the same as the real map (d), the output image with magnified defocus (f) is visually close to the f/4 photograph (b).*



(a) input (b) defocus map (c) refocusing result using (a) and (b)

**Figure 8:** *Using our defocus map, we can synthesize refocusing effects. We perform deconvolution using our defocus map (b) and apply lens blur. The result (c) looks as if the foreground is focused. The input photo was taken by a Canon PowerShot A80, a point-and-shoot camera with a sensor size of* $7.18 \times 5.32$ *mm, and a* $7.8$ *mm lens at f/2.8.*

do not need to disambiguate textureless regions. Our defocus map focuses on edges and texture regions that are visually affected by defocusing and approximates textureless regions without causing visual defects.

In future work, we want to extend this work to video inputs where the effect of motion blur needs to be distinguished from depth of field. Finally, we also want to further study occlusion boundaries, a traditional issue for depth of field effects.

**Acknowledgement** We thank the MIT Computer Graphics Group and the anonymous reviewers for their comments. This work was supported by a National Science Foundation CAREER award 0447561 "Transient Signal Processing for Realistic Imagery," an NSF Grant No. 0429739 "Parametric Analysis and Transfer of Pictorial Style," and a grant from Royal Dutch/Shell Group. Frédo Durand acknowledges a Microsoft Research New Faculty Fellowship, a Sloan Fellowship, and a Jamieson chair. Soonmin Bae is financially supported by the Samsung Scholarship.

## References

[BHK*03] BARSKY B., HORN D., KLEIN S., PANG J., YU M.: Camera models and optical systems used in computer graphics: Part ii, image based techniques. In *ICCSA* (2003).

[BTCH05] BARSKY B. A., TOBIAS M. J., CHU D. P., HORN D. R.: Elimination of artifacts due to occlusion and discretization problems in image space blurring techniques. *Graph. Models 67*, 6 (2005), 584–599.

[DW88] DARRELL T., WOHN K.: Pyramid based depth from focus. In *IEEE CVPR* (1988).

[ED04] EISEMANN E., DURAND F.: Flash photography enhancement via intrinsic relighting. *ACM Transactions on Graphics 23*, 3 (2004). Proceeding of ACM SIGGRAPH conference.

[EK03] ELTOUKHY H. A., KAVUSI S.: A computationally efficient algorithm for multi-focus image reconsction. In *SPIE Electronic imaging* (June 2003), vol. 3813.

[EL93] ENS J., LAWRENCE P.: An investigation of methods for determining depth from focus. *IEEE Transactions on PAMI 15*, 2 (1993), 97–108.

[EZ98] ELDER J. H., ZUCKER S. W.: Local scale control for edge detection and blur estimation. *IEEE Transactions on PAMI 20*, 7 (1998), 699–716.

[FS02] FAVARO P., SOATTO S.: Learning shape from defocus. In *ECCV* (London, UK, 2002), Springer-Verlag, pp. 735–745.

[FS05] FAVARO P., SOATTO S.: A geometric approach to shape from defocus. *IEEE Transactions on PAMI 27*, 3 (2005), 406–417.

[Hec02] HECHT E.: *Optics*. Addison-Wesley, Reading, MA, 2002.

[HK06] HASINOFF S. W., KUTULAKOS K. N.: Confocal stereo. In *ECCV* (2006), pp. 620–634.

[Hor68] HORN B.: Focusing. *Memo*, 10 (May 1968).

[IMG00] ISAKSEN A., MCMILLAN L., GORTLER S. J.: Dynamically reparameterized light fields. *ACM Transactions on Graphics*, 3 (2000).

[JF02] JIN H., FAVARO P.: A variational approach to shape from defocus. In *ECCV* (May 2002).

[KA05] KUBOTA A., AIZAWA K.: Reconstructing arbitrarily focused images from two differently focused images using linear filters. *IEEE Transactions on IP 14*, 11 (November 2005), 1848–1859.

[LFC92] LAI S.-H., FU C.-W., CHANG S.: A generalized depth estimation algorithm with a single image. *IEEE Transactions on PAMI 14*, 4 (1992), 405–411.

[LLW04] LEVIN A., LISCHINSKI D., WEISS Y.: Colorization using optimization. *ACM Transactions on Graphics 23*, 3 (2004), 689–694. Proceeding of ACM SIGGRAPH conference.

[NLB*05] NG R., LEVOY M., BREDIF M., DUVAL G., HOROWITZ M., HANRAHAN P.: Light field photography with a hand-held plenoptic camera.

[NN94] NAYAR S. K., NAKAGAWA Y.: Shape from focus. *IEEE Transactions on PAMI 16*, 8 (1994), 824–831.

[OTS94] ÖZKAN M. K., TEKALP A. M., SEZAN M. I.: Pocs-based restoration of space-varying blurred images. *IEEE Transactions on IP 3*, 4 (1994), 450–454.

[PAH*04] PETSCHNIGG G., AGRAWALA M., HOPPE H., SZELISKI R., COHEN M. F., TOYAMA K.: Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics 23*, 3 (2004). Proceeding of ACM SIGGRAPH conference.

[PC81] POTMESIL M., CHAKRAVARTY I.: A lens and aperture camera model for synthetic image generation. In *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1981), ACM Press, pp. 297–305.

[Pen87] PENTLAND A. P.: A new sense for depth of field. *IEEE Transactions on PAMI 9*, 4 (1987), 523–531.

[RM92] REEVES S. J., MERSEREAU R. M.: Blur identification by the method of generalized cross-validation. *IEEE Transactions on IP 1*, 3 (1992), 301–311.

[TF92] TRUSSELL H. J., FOGEL S.: Identification and restoration of spatially variant motion blurs in sequential images. *IEEE Transactions on IP 1*, 1 (1992), 123–126.

[WN98] WATANABE M., NAYAR S. K.: Rational filters for passive depth from defocus. *IJCV 27*, 3 (1998), 203–225.

**Appendix A:** Defocus and circle of confusion size

Although most camera lenses use more intricate designs with multiple lens elements, we review the simplified thin lens model, which suffices in our context.

The main role of a lens is to make all the rays coming from a point at the focus distance converge to a point in the image plane. In contrast, the rays originating at a scene point away from the focus distance converge in front of or behind the lens, and that point appears as a blurred spot in the image. The blurred spot is called the circle of confusion. It is not strictly speaking a circle and depends on the aperture shape and diffraction, but it is often modeled as a circle or a Gaussian.

We express the circle of confusion diameter $c$ of a point at distance $S$ (Figure 9). A detailed derivation can be found in optics textbooks such as Hecht's [Hec02]. Given the focal length $f$ of the lens, the thin-lens formula gives us the lens-sensor distance $f_D$ to focus at distance $D$: $\frac{1}{f} = \frac{1}{f_D} + \frac{1}{D}$.

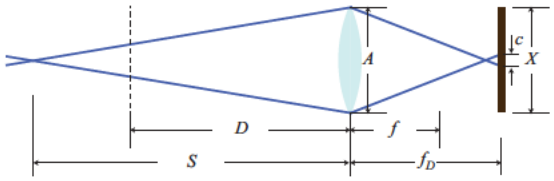



**Figure 9:** *A thin-lens system. The lens' diameter is A and its focal length is f. The image plane is at distance $f_D$ from the lens and the focus distance is D. Rays from a point at distance S generates a circle of confusion diameter c. And the rays generates a virtual blur circle diameter C at the focus distance D.*

The f-number $N$ gives the aperture diameter $A$ as a fraction of the focal length ($A = Nf$). Note that $N$ has no unit. $N$ is the number, such as 2.8, that photographers set to control the diaphragm. The aperture is then denoted by, e.g., f/2.8 to

express that the diameter is the focal length divided by the f-number. The diameter of the circle of confusion is then

$$c = \frac{|S-D|}{S} \cdot \frac{f^2}{N(D-f)} \tag{5}$$

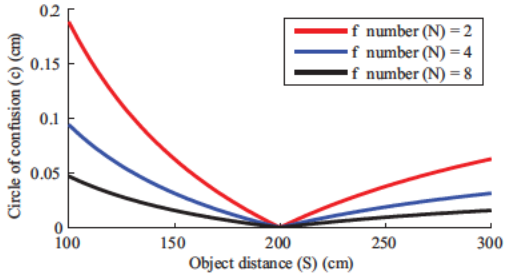



**Figure 10:** *This plot shows how the circle of confusion diameter, c, changes according to the change of object distance S and f-number N. c increases as a point is away from the focus distance D. The focus distance D is* 200*cm, and the focal length f is* 8.5*cm*

Figure 10 shows that the circle of confusion diameter $c$ increases as a point is away from the focus distance $D$. The relationship is not linear (hyperbolic) and is not symmetrical for points in front of and behind the plane in focus.

We now study the effect of the sensor size, $X$. To express the amount of defocus in terms of image-space blur, we use the relative size of the circle of confusion $c' = c/X$. For sensors of different sizes, the same field of view is obtained if the relative focal length $f' = f/X$ is the same. Replacing $c$ and $f$ by their relative version in Eq. 5 we obtain

$$c' = \frac{|S-D|}{S} \cdot \frac{f'^2 X}{N(D-f'X)} \tag{6}$$

To a first-order approximation, the amount of defocus is proportional to the sensor size $X$. This confirms that for a given f-number $N$, a smaller sensor does not yield as much defocus as a larger sensor. More defocus could be achieved by using a smaller f-number (larger lens aperture), but this would require bending rays that reach the periphery of the lens aperture by angles that are physically challenging to achieve. Scaling down the sensor and lens inherently scales down the amount of defocus.

## Appendix B: Elder and Zucker's edge detector

Elder and Zucker [EZ98] detect edges with various levels of blurriness. To determine the right scale for edge detection, they compute the minimum reliable scale for each pixel, based on the noise thresholds . They locate edges by testing nonzero gradient and zero-crossing of second derivative at the minimum reliable scale.

For each pixel, Elder and Zucker compute its multiscale responses to the steerable Gaussian first derivative filters and steerable second derivative of Gaussian filters and compute the gradient using the steerable Gaussian first derivative basis filters:

$$g_1^x(x,y,\sigma_1) = \frac{-x}{2\pi\sigma_1^4}\exp(-(x^2+y^2)/2\sigma_1^2) \tag{7a}$$

$$g_1^y(x,y,\sigma_1) = \frac{-y}{2\pi\sigma_1^4}\exp(-(x^2+y^2)/2\sigma_1^2) \tag{7b}$$

where $\sigma_1$ is the scale of the first derivative Gaussian estimator. A weighted sum of these two filter responses is used to compute the gradient direction $\theta$ that maximizes the gradient magnitude.

They compute the second derivative in the direction $\theta$ using a steerable second derivative of Gaussian operator:

$$g_2^x(x,y,\sigma_2) = \frac{(x/\sigma_2)^2-1}{2\pi\sigma_2^4}\exp(\frac{-(x^2+y^2)}{2\sigma_2^2}) \tag{8a}$$

$$g_2^y(x,y,\sigma_2) = \frac{(y/\sigma_2)^2-1}{2\pi\sigma_2^4}\exp(\frac{-(x^2+y^2)}{2\sigma_2^2}) \tag{8b}$$

$$g_2^{xy}(x,y,\sigma_2) = \frac{xy}{2\pi\sigma_2^6}\exp(\frac{-(x^2+y^2)}{2\sigma_2^2}) \tag{8c}$$

$$g_2^\theta(x,y,\sigma_2) = \cos^2(\theta)g_2^x(x,y,\sigma_2)+\sin^2(\theta)g_2^y(x,y,\sigma_2) \tag{8d}$$

$$-2\cos(\theta)\sin(\theta)g_2^{xy}(x,y,\sigma_2) \tag{8e}$$

where $\sigma_2$ is the scale of the second derivative of Gaussian filter.

They test the reliability of filter responses by setting a threshold for each scale. The thresholds are derived from the sensor noise level $s_n$. In the following equations, $c_1$ denotes the threshold for Gaussian first derivative filter in a function of $\sigma_1$ and $c_2$ denotes the threshold for the second derivative of Gaussian filter in a function of $\sigma_2$.

$$c_1(\sigma_1) = \frac{s_n\sqrt{-2\ln\alpha_p}}{2\sqrt{2\pi}\cdot\sigma_1^2} \tag{9a}$$

$$c_2(\sigma_2) = \frac{s_n\sqrt{2}\cdot erf^{-1}(\alpha_p)}{4\sqrt{\pi/3}\cdot\sigma_2^3} \tag{9b}$$

$$\text{with: } \alpha_p = 1-(1-\alpha_I)^{1/n} \tag{9c}$$

where $n$ is the number of pixels. The thresholds are computed statistically based on the standard deviation of the sensor noise $s_n$ and a false positive tolerance $\alpha_I$. At the minimum reliable scale, pixel filter responses are larger than the threshold of the scale.

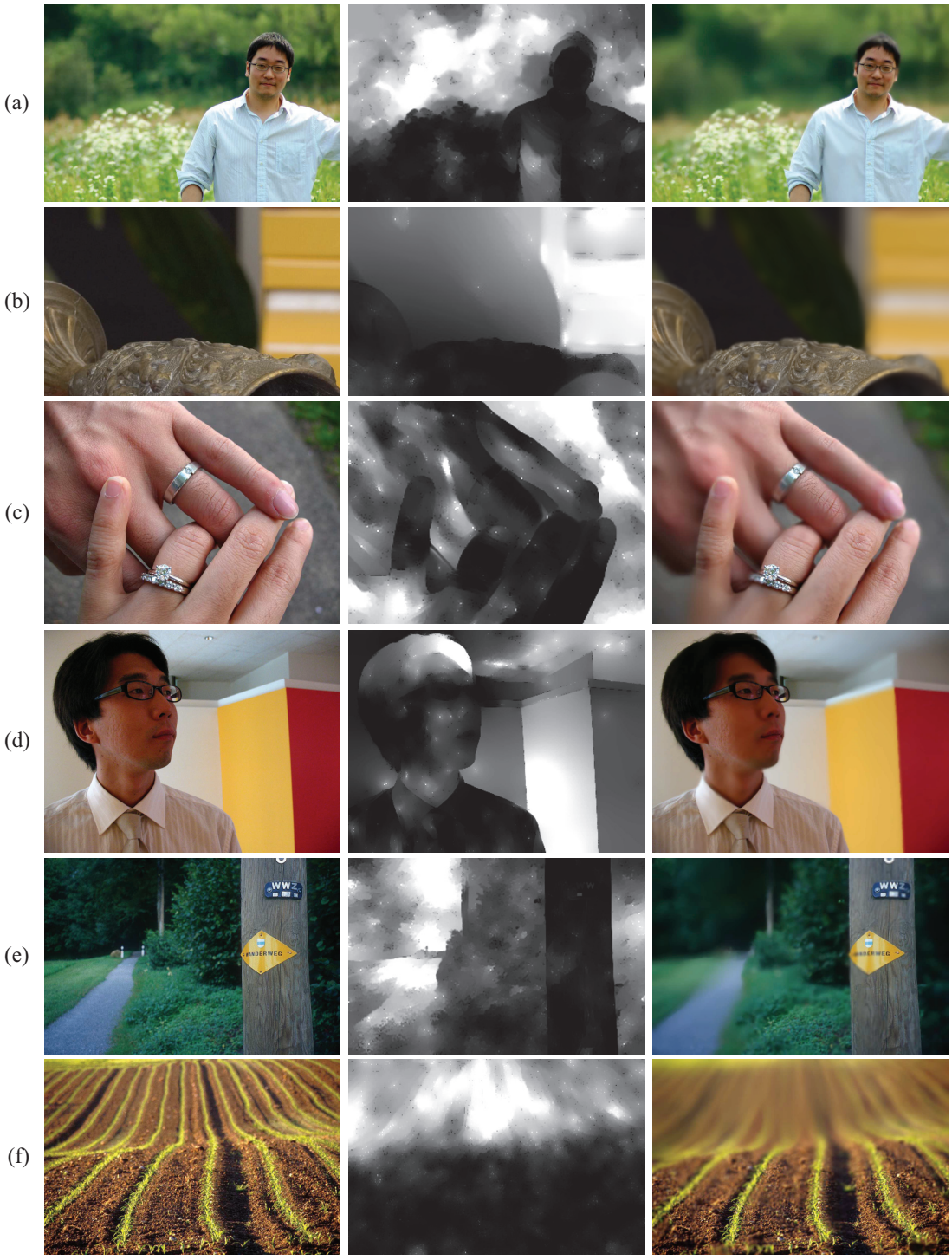



**Figure 11:** *Results. The original images, their defocus maps, and results blurred using our approach. The inputs were taken by (a) a Nikon D50 with a sensor size of* $23.7 \times 15.6$ *mm and a* $180.0$ *mm lens at f/4.8, (b) a Canon 1D Mark II with a sensor size of* $28.7 \times 19.1$ *mm and a Canon EF 85mm f/1.2L lens, and (c, d) a Canon PowerShot A80, a point-and-shoot camera with a sensor size of* $7.18 \times 5.32$ *mm, and a* $7.8$ *mm lens at f/2.8. The two at the bottom are from bigfoto.com.*