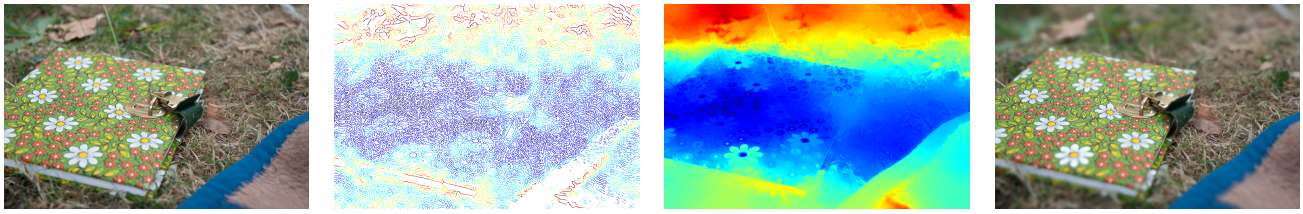
Accelerating defocus blur magnification

Florian Kriener, Thomas Binder and Manuel Wille

Google Inc.



(a) Input image $\boldsymbol{I}$ (b) Sparse blur map $\boldsymbol{\beta}$ (c) Full blur map $\boldsymbol{\alpha}$ (d) Output image $\boldsymbol{J}$

Figure 1: Real world example of the steps of our algorithm. We estimate the blur at edge locations in the image (b), then we interpolate the values to close the gaps (c). This blur map can be used to magnify the defocus blur (d).

ABSTRACT

A shallow depth-of-field is often used as a creative element in photographs. This, however, comes at the cost of expensive and heavy camera equipment, such as large sensor DSLR bodies and fast lenses. In contrast, cheap small-sensor cameras with fixed lenses usually exhibit a larger depth-of-field than desirable. In this case a computational solution is suggesting, since a shallow depth-of-field cannot be achieved by optical means. One possibility is to algorithmically increase the defocus blur already present in the image. Yet, existing algorithmic solutions tackling this problem suffer from poor performance due to the ill-posedness of the problem: The amount of defocus blur can be estimated at edges only; homogeneous areas do not contain such information. However, to magnify the defocus blur we need to know the amount of blur at every pixel position. Estimating it requires solving an optimization problem with many unknowns.

We propose a faster way to propagate the amount of blur from the edges to the entire image by solving the optimization problem on a small scale, followed by edge-aware upsampling using the original image as guide. The resulting approximate defocus map can be used to synthesize images with shallow depth-of-field with quality comparable to the original approach. This is demonstrated by experimental results.

Keywords: defocus blur magnification, image processing, optimization, depth-of-field, depth map blur map, computational photography

1. INTRODUCTION

Creating photographic images with a shallow depth-of-field requires two features: a camera with a large sensor and a lens with a large aperture. Both are required, since even a fast lens in combination with a small sensor is known to exhibit an unpleasant bokeh. Both features are usually limited to high-end equipment whereas cheap point-and-shoot cameras or cameras embedded in cell phones have neither. Furthermore, a photographer has to decide on the depth-of-field at the time of shooting or even well before when he or she decides what lenses

Version of March 18, 2013. Cite as: F. Kriener, T. Binder, M. Wille, "Accelerating defocus blur magnification," Proceedings SPIE Vol. 8667 (Multimedia Content and Mobile Devices), 86671Q (2013). DOI: http://dx.doi.org/10.1117/12.2004118

to bring. This calls for a solution of simulating a shallow depth-of-field computationally and with little user intervention. Existing methods[1,2] roughly follow the same principle, comprising the following three steps.

First, the amount of defocus blur is estimated at edge locations of the original photograph. This yields a sparse defocus map. Second, the sparse data is propagated using the original image as guidance to obtain a full defocus map. The difficulty of this step is to preserve blur discontinuities at edges, while at the same time closing the gaps smoothly. We can express this problem as an optimization problem whose complexity is proportional to the number of pixels. For this reason, solving the optimization problem is time-consuming; it is the performance bottleneck of the present approach. Third, the full defocus map is used by a blurring algorithm to apply the desired bokeh effect to the image. Of course, the choice of the blurring algorithm is highly subjective and therefore outside the scope of this paper.

We present a way to speed up the second step of the approach just outlined: the computation of a dense defocus map from sparse data. The main idea is to reduce the complexity of the optimization problem by solving it on a smaller scale, and thereby significantly reducing the amount of unknowns, followed by edge-aware upsampling that uses the original photograph as guidance. Our experiments show that this yields results similar to the non-downsampled results even though we lose information in the downsampling phase. Ordinary upsampling would blur edges, which in turn would produce artifacts in the final image. Therefore, we employ edge-aware upsampling to ensure that edges remain sharp. Our experimental results show that such an approximate defocus map is sufficient to create high quality shallow depth-of-field images from photographs with a large depth-of-field.

1.1 Related Work

The defocus blur of an image contains information about the depth of the scene. Humans instinctively relate the blur to the depth and machines can be taught to recover the depth in various ways:

Shape from focus[3–5] is a technique to recover the depth of a scene using a stack of images captured with different focal distances and a small depth-of-field, possibly by changing the aperture as well. The depth can be recovered by estimating the sharpness for each pixel of each input image and relating it to the focal distance where the sharpness measurement is maximal. Like other techniques that compare images with different focus and/or aperture settings shape from focus requires a calibrated camera.

Shape from defocus[6–10] uses the blur of the image instead of the sharpness to estimate the depth. However, the blur increases with the distance to the focal plane in both directions. Therefore, the blur is not directly related to depth, as it can be strong in front and behind the focal plane. Although this sign ambiguity could be overcome by incorporating chromatic aberration[11], present shape from defocus methods usually require a second image with a different focal distance and therefore a calibrated camera as well. Zhou et al.[12] even modify the optical system by using coded apertures to increase accuracy.

Focal stack compositing[13,14] uses a stack of shallow depth-of-field images as well and combines them to create arbitrary depth-of-fields effects. It is also possible to use it in conjunction with shape from focus or shape from defocus to synthesize realistic depth-of-field images. One advantage of this approach is that, in theory, an image with a large depth-of-field can be captured faster using a stack of small depth-of-field captures[13].

In contrast, we are only interested in the amount of blur in the image not its depth and can ignore the sign ambiguity shape from defocus has to address. By using only one image we do not need to compare images, therefore our camera does not need to be calibrated. Also, we are not interested to speed up the image capturing process or to create a larger depth-of-field. Our goal is to increase the existing defocus blur in a post-production step as a creative tool for photographers. Furthermore, we want to apply the effect without planning it in advance and therefore restrict ourself to using one single image and an off-the-shelf camera.

Defocus magnification without user interaction was first proposed by Bae and Durand[1], who modified a blur estimator by Elder and Zucker[15] for robustness to find blur estimates near edges and propagate these by solving an optimization problem that is based on a colorization scheme[16]. Zhuo and Sim[2] use a simple but robust method for blur estimation and propagate these using an α-matting method[17]. In this paper we will follow the method proposed by Zhuo and Sim and modify it for better performance. However, our modification could also be applied to the method proposed by Bae and Durand.

The heart of our acceleration scheme (section 2 below) is the guided upsampling of a lower resolution blur map. This problem is similar to the upsampling of a low resolution depth map: Park et al.[18] interpret this problem as a super-resolution problem and employ an optimization scheme that is similar to the propagation used by Bae and Durand[1] for upsampling the depth map. This however incurs exactly the cost that we are trying to avoid by downsampling the image and solving the propagation problem on that smaller scale. Faster edge-aware upsampling can be achieved using bilateral filtering (e.g. Yang et al.[19]) or joint bilateral upsampling[20] as proposed by Chan et al.[21] We will use the guided filter[22] for this task as it connects in a natural way with the propagation method.

2. DEFOCUS BLUR MAGNIFICATION

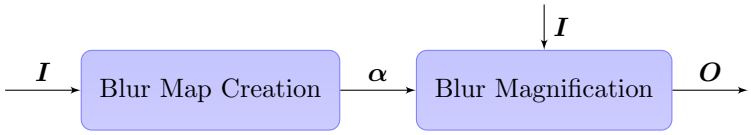


Figure 2: Overview of the general algorithm. $\boldsymbol{I}$ is the input image, $\boldsymbol{\alpha}$ is the full blur map, and $\boldsymbol{O}$ is the output image.

The main difficulty in defocus blur magnification, as shown in figure 2, is the creation of a map containing the spatially-varying blur for every pixel, we call this the *blur map*. Generating the blur map is difficult because the direct estimation of blur is only possible near edges. A patch without edges does not change much after being blurred, because a blur is a filter that suppresses high frequencies but allows lower frequencies to pass. Therefore, the blur map creation process is split into two steps, as shown in figure 3. Afterwards we present our acceleration method.

First, we estimate the blur near edge locations using the algorithm proposed by Zhuo and Sim[2]. We call the result the *sparse blur map*. Second, we propagate the known values from the sparse blur map to every pixel in the image to obtain the *full blur map*. For this propagation we employ an optimization algorithm based on an α-matting algorithm devised by Levin et al.[17] We call this the *direct method* (see figure 3).

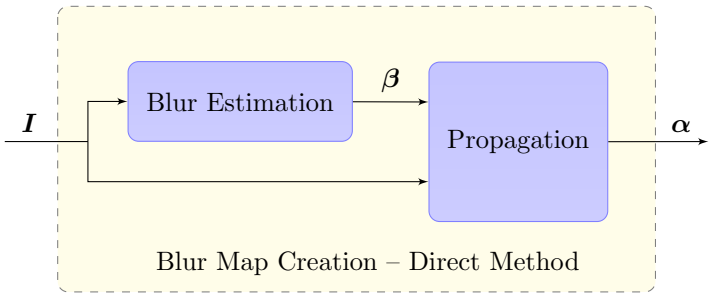


Figure 3: Overview of the direct blur map creation algorithm. $\boldsymbol{I}$ is the input image, $\boldsymbol{\beta}$ the sparse blur map, and $\boldsymbol{\alpha}$ is the full blur map. Everything is done at full resolution.

The propagation step is the bottleneck of the direct method because it needs to solve an optimization problem with many unknowns whereas the blur estimation step consists solely of local operations. We accelerate it by downsampling the estimated sparse blur map $\boldsymbol{\beta}$ by a factor of two and solve the optimization problem on that smaller scale. Figure 4 provides a block diagram illustrating our method. To obtain a full resolution blur map from this low resolution blur map we apply edge-aware upsampling. The edge-aware upsampling algorithm is based on the guided filter proposed by He et al.[22] and is closely related to the α-matting algorithm used in the propagation step. Experiments show that the error introduced by this acceleration technique compared with the direct method is quite small and does not corrupt the end result in a noticeable way.

For the blur magnification step in figure 2 one can use any algorithm that emulates lens blur based on a depth map. Such algorithms are readily available[23–25]; we only replace the depth map input with the blur map. For the examples in this paper we use the *Lens Blur* feature of *Adobe Photoshop® CS6*.

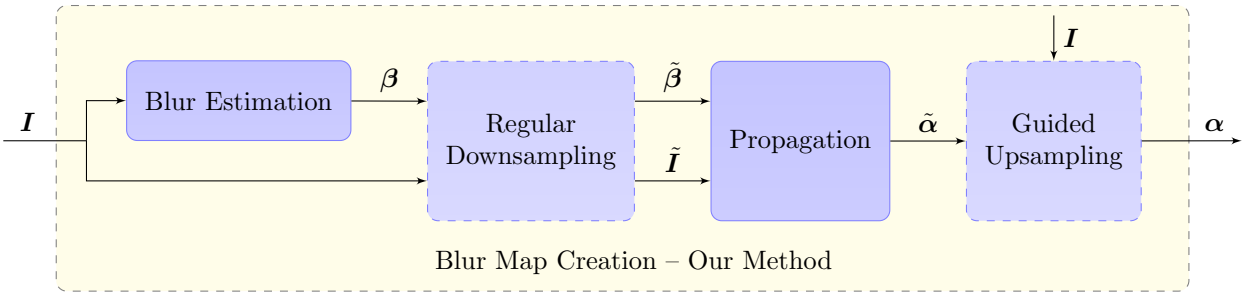


Figure 4: Overview of our blur map creation algorithm. Here $\tilde{\boldsymbol{I}}$ and $\tilde{\boldsymbol{\beta}}$ are the downsampled input image and the downsampled sparse blur map, respectively $\tilde{\boldsymbol{\alpha}}$ is the full blur map at low resolution. The dashed blue blocks highlight our acceleration scheme.

2.1 Blur estimation

As said, we base our blur estimation on the method by Zhuo and Sim[2]. Assuming a Gaussian defocus blur we can estimate its parameters by blurring the original image via convolution with a Gaussian kernel and comparing the gradients of the original to the blurred version. Of course, the defocus blur of a lens is not a Gaussian blur, but that hardly matters because our objective is not to estimate the exact σ of a Gaussian but to represent the amount of perceived blurriness.

The blur estimation algorithm is best described in the continuous domain (which we will employ here and only here). Let $u : \mathbb{R}^2 \to \mathbb{R}$ be a differentiable 2D grayscale image and $g_\sigma : \mathbb{R}^2 \to \mathbb{R}$ a Gaussian with $\sigma > 0$. Let $u_0 = u * g_{\sigma_0}$ be the blurred image with $\sigma_0 > 0$ and let $\mathcal{E} \subset \mathbb{R}^2$ be the set of edge locations of the image u. Here we pragmatically define edge as having sufficiently large gradient magnitude. At $\boldsymbol{x} \in \mathcal{E}$ we can assume $\|\nabla u_0(\boldsymbol{x})\| < \|\nabla u(\boldsymbol{x})\|$ and use the following formula to estimate the blur $\hat{\sigma} : \mathcal{E} \to \mathbb{R}$ at edge locations (see Zhuo and Sim for the derivation)

$$\hat{\sigma}(\boldsymbol{x}) = \sigma_0 \sqrt{\frac{\|\nabla u_0(\boldsymbol{x})\|^2}{\|\nabla u(\boldsymbol{x})\|^2 - \|\nabla u_0(\boldsymbol{x})\|^2}}. \tag{1}$$

It is straightforward how this formula carries over to a discrete domain. We implemented it using Canny's algorithm for edge detection and the Scharr operator* for gradient estimation. The result was blurred with the guided filter[22] to make the method more robust against noise. The result is a discrete image denoted by $\boldsymbol{\beta} \in \mathbb{R}^N$ with undefined values set to 0, here N is the number of pixels.

2.2 Propagation

The previous step estimates the amount of blur at edge locations, but we need to know the amount of blur at every pixel position. The gaps between the edges need to be closed by propagating the blur information from the edges across the image. This is done by incorporating information from the source image into the propagation algorithm to direct the propagation of information from the edges into the gaps but not across edges.

Following Zhuo and Sim[2] once more, we base the propagation on an α-matting method proposed by Levin et al.[17] Let $\mathcal{I} = \{1, 2, \dots N\}$ be the set of pixel indices of the image $\boldsymbol{I} = (\boldsymbol{I}_k)_{k \in \mathcal{I}} \in \mathbb{R}^{N \times 3}$ with $\boldsymbol{I}_k = (I_k^R, I_k^G, I_k^B)^T$. We assume that in a small window $w \subseteq \mathcal{I}$ around any pixel the blur map $\boldsymbol{\alpha} = (\alpha_i)_{i \in \mathcal{I}} \in \mathbb{R}^N$ can be approximated in w by an affine function of the image $\boldsymbol{I}$;

$$\alpha_j \approx \boldsymbol{a}^T \boldsymbol{I}_j + b \qquad \text{for all } j \in w, \tag{2}$$

where $\boldsymbol{a} \in \mathbb{R}^3$ and $b \in \mathbb{R}$ are constant in the window w. This assumption is reasonable since an edge in the image $\boldsymbol{I}$ does not necessarily create an edge in the blur map $\boldsymbol{\alpha}$, e.g. in a flat but differently colored region. Also, a difference in color does not imply a difference in depth or defocus blur, e.g. with similar colored objects in the front and the back of a scene that overlap in a photograph. The downside is that some unwanted information

*The Scharr operator is similar to the Sobel operator but numerically optimized for rotational symmetry.

from the original image might seep into the blur map. We found ,however, that this does not lead to a noticeable corruption in the application of defocus magnification.

Using the assumption above we find the blur map by minimizing the functional

$$\tilde{J}_Z(\boldsymbol{\alpha}, \boldsymbol{A}, \boldsymbol{b}) = \sum_{i \in \mathcal{I}} \left(\sum_{j \in w_i} \left(\alpha_j - \boldsymbol{a}_i^T \boldsymbol{I}_j - b_i \right)^2 + \varepsilon \, \boldsymbol{a}_i^T \boldsymbol{a}_i \right) + \lambda \sum_{i \in \mathcal{I}} d_{ii} \left(\alpha_i - \beta_i \right)^2 \tag{3}$$

with respect to $\boldsymbol{\alpha} = (\alpha_i)_{i \in \mathcal{I}} \in \mathbb{R}^N$, $\boldsymbol{A} = (\boldsymbol{a}_i)_{i \in \mathcal{I}} \in \mathbb{R}^{N \times 3}$ and $\boldsymbol{b} = (b_i)_{i \in \mathcal{I}} \in \mathbb{R}^N$, where $w_i \subset \mathcal{I}$ is a small window around the i-th pixel, $\boldsymbol{\beta} = (\beta_i)_{i \in \mathcal{I}} \in \mathbb{R}^N$ is the sparse blur map, and $\boldsymbol{D} = (d_{ij})_{i,j \in \mathcal{I}} \in \mathbb{R}^{N \times N}$ is a diagonal matrix with $d_{ii} = 1$ if i is an edge location and $d_{ii} = 0$ otherwise. The parameter $\varepsilon > 0$ in the above functional controls its regularization and biases the solution towards a smoother $\boldsymbol{\alpha}$ (cf. Levin et al.[17]) and the weighting parameter $\lambda > 0$ balances both cost functions, allowing to deviate from the input data to achieve a better overall fit.

It can be shown[17] that $\boldsymbol{A}$ and $\boldsymbol{b}$ can be eliminated from the equation, reducing the amount of unknowns from $5N$ to N. This yields the functional

$$J_Z(\boldsymbol{\alpha}) = \boldsymbol{\alpha}^T \boldsymbol{L} \, \boldsymbol{\alpha} + \lambda \, (\boldsymbol{\alpha} - \boldsymbol{\beta})^T \boldsymbol{D} \, (\boldsymbol{\alpha} - \boldsymbol{\beta}), \tag{4}$$

where $\boldsymbol{L} = (l_{ij})_{i,j \in \mathcal{I}} \in \mathbb{R}^{N \times N}$ is called the *matting Laplacian*. It is defined as

$$l_{ij} = \sum_{\{k \in \mathcal{I} | i,j \in w_k\}} \left(\delta_{ij} - \frac{1}{|w_k|} \left(1 + (\mathbf{I}_i - \boldsymbol{\mu}_k)^T \left(\boldsymbol{\Sigma}_k + \frac{\varepsilon}{|w_k|} \boldsymbol{U}_3 \right)^{-1} (\mathbf{I}_j - \boldsymbol{\mu}_k) \right) \right), \tag{5}$$

where δ_{ij} designates the Kronecker delta, $|w_k| \in \mathbb{N}$ is the window size (i.e. the number of pixels in the window), $\boldsymbol{\mu}_k \in \mathbb{R}^3$ is the mean and $\boldsymbol{\Sigma}_k \in \mathbb{R}^{3 \times 3}$ is the covariance matrix of the pixels values in the window w_k, and $\boldsymbol{U}_3 \in \mathbb{R}^{3 \times 3}$ is the 3×3 identity matrix.

Since the matting Laplacian is symmetric and positive definite (we just need to choose an ε that is big enough) we can employ the Conjugate Gradient (CG) method (see Golub and Van Loan[26] for details) to solve the resulting linear system of equations

$$(\boldsymbol{L} - \lambda \boldsymbol{D}) \, \boldsymbol{\alpha} = \lambda \boldsymbol{D} \boldsymbol{\beta}. \tag{6}$$

The CG method is an iterated method and the computational cost for each iteration of a naive implementation of the CG algorithm is dominated by the matrix-vector product $\boldsymbol{L}\boldsymbol{p}$, for some $\boldsymbol{p} \in \mathbb{R}^N$. However, this cost can be mitigated by a technique found by He et al.[27] that allows us to compute the product $\boldsymbol{L}\boldsymbol{p}$ using a series of box filter operations, which can be implemented effectively using the integral image technique[28] or cumulative row and column sums.

2.3 Acceleration scheme

It is clear that doing the propagation step on a smaller scale results in a reduced time consumption. The question is if a high-quality full size blur map can be created from a far smaller blur map. Our experiments show that this is possible indeed. Therefore we propose the following scheme.

First, we estimate the blur using the original, full resolution image. This is necessary because downsampling an image is a low-pass filtering operation (unless it creates artifacts), but we need the the high frequency content for the blur estimation. This is not a performance issue, because the estimation is way faster than the propagation. Second, we downsample the sparse blur map using bicubic interpolation by a factor of two and solve equation (6) using the Conjugate Gradient method. For this we employ He's functional form of the multiplication with the matting Laplacian. We found that this way we obtain a robust blur map for the downsampled image. Third, to obtain a full size blur map we use a joint upsampling technique proposed by He et al.[22] that is closely related to the matting Laplacian.

Let $\tilde{\boldsymbol{\alpha}} \in \mathbb{R}^n$ be the solution of (6) for a downsampled sparse blur map $\boldsymbol{\beta}$, where n is the number of pixels for an image on that scale ($4n \approx N$), and let $\uparrow_2 \colon \mathbb{R}^n \to \mathbb{R}^N$ be the zero-upsampling operator[†]. We need to find

[†]The zero-upsampling operator upsamples the input image by inserting zeros in-between every row and column.

$\boldsymbol{\alpha} \in \mathbb{R}^N$ similar to $\tilde{\boldsymbol{\alpha}}$ satisfying assumption (2). We do that by minimizing the difference between $\boldsymbol{\alpha}$ as defined by (2) and $\uparrow_2 \tilde{\boldsymbol{\alpha}}$ at pixels with defined blur values (i.e. for all $i \in \mathcal{I}$ with $(\uparrow_2 \mathbf{1})_i \neq 0$, where $\mathbf{1} \in \mathbb{R}^n$ is an image with pixel value 1 for all pixels) by minimizing

$$J_U(\boldsymbol{a}_k, b_k) = \sum_{i \in \overline{w}_k} \left(\left(\boldsymbol{a}_k^T \boldsymbol{I}_i + b_k - (\uparrow_2 \tilde{\boldsymbol{\alpha}})_i \right)^2 + \varepsilon \, \boldsymbol{a}_k^T \boldsymbol{a}_k \right) \tag{7}$$

with respect to $\boldsymbol{a}_k$ and b_k for all $k \in \mathcal{I}$, where $\varepsilon > 0$ is a regularization parameter as in (3) and $\overline{w}_k = \{ i \in w_k \mid (\uparrow_2 \mathbf{1})_i \neq 0 \}$ is a window around k including only defined blur values of $\uparrow_2 \tilde{\boldsymbol{\alpha}}$. We then define

$$\alpha_j := \frac{1}{|w_k|} \sum_{k \in w_j} \left(\boldsymbol{a}_k^T I_j + b_k \right). \tag{8}$$

Again, this optimization problem can be solved directly and effectively with a series of box blur operations[22].

2.4 Experimental Results

We expect that our method produces results very similar to the direct method when compared by a human observer. A visual comparison of the output of both algorithms confirms this, see figure 5. The question is how accurate our approximation is when expressed numerically. To compare the methods we ran both with the same parameters for a varying number of iterations and measured time and mean squared error (MSE) compared to the result of the direct method at 500 iterations of the CG algorithm. We choose that number because 500 iterations take longer than any reasonable time constraint (more than half a minute for a 0.5 million pixel image on the machine that we used for the tests) and we do not have a ground truth that we could use to estimate the error. We repeated each measurement $M = 10$ times to estimate sample mean $\overline{T}$ and sample variance s^2:

$$\overline{T} = \frac{1}{M} \sum_{k=1}^{M} T_k, \quad s^2 = \frac{1}{M-1} \sum_{k=1}^{M} (T_k - \overline{T})^2, \tag{9}$$

where T_k is the measured time of the k-th repeat. The sample variance was less than the timer resolution of our system (≈ 590 ns) in all our measurements; therefore we will ignore it henceforth.

We used a custom, single-threaded, unoptimized C++ implementation to run the test. All tests ran on a Mid 2012 Mac Pro with 32GB of Memory and a 6-Core Xeon Processor running OS X 10.8. For all experiments we used the following parameters. The blur estimation uses $\sigma_0 = 3$ for re-blurring. Propagation is done using a 7×7 window with the parameters $\varepsilon = 0.001$, $\lambda = 0.1$. Upsampling uses a 21×21 window with $\varepsilon = 0.01$. We found that these parameters create good results by experimentation. Note that the 21×21 window used for upsampling is empty for the most part, since only every second pixel in each direction has a defined value.

We ran both algorithms for a fixed amount of iterations which we increased after every $M = 10$ runs and measured MSE and time as described above. In figure 6 a typical MSE vs. time plot is shown. Here we use the image that was used for the examples above but we tested with different images and all those plots show the same characteristics. The size of that image is 800×536 pixels. Our method converges faster than the direct method until our method stagnates at the exact solution on the smaller scale (norm of the residual approximately zero). The direct method takes longer but eventually converges to a numerically better result. However, we do not strive to find a numerically more exact solution but to find a good approximation for the application of defocus blur magnification. For this the solution is good if the end result is visually pleasant and free of artifacts, as seen in figure 5.

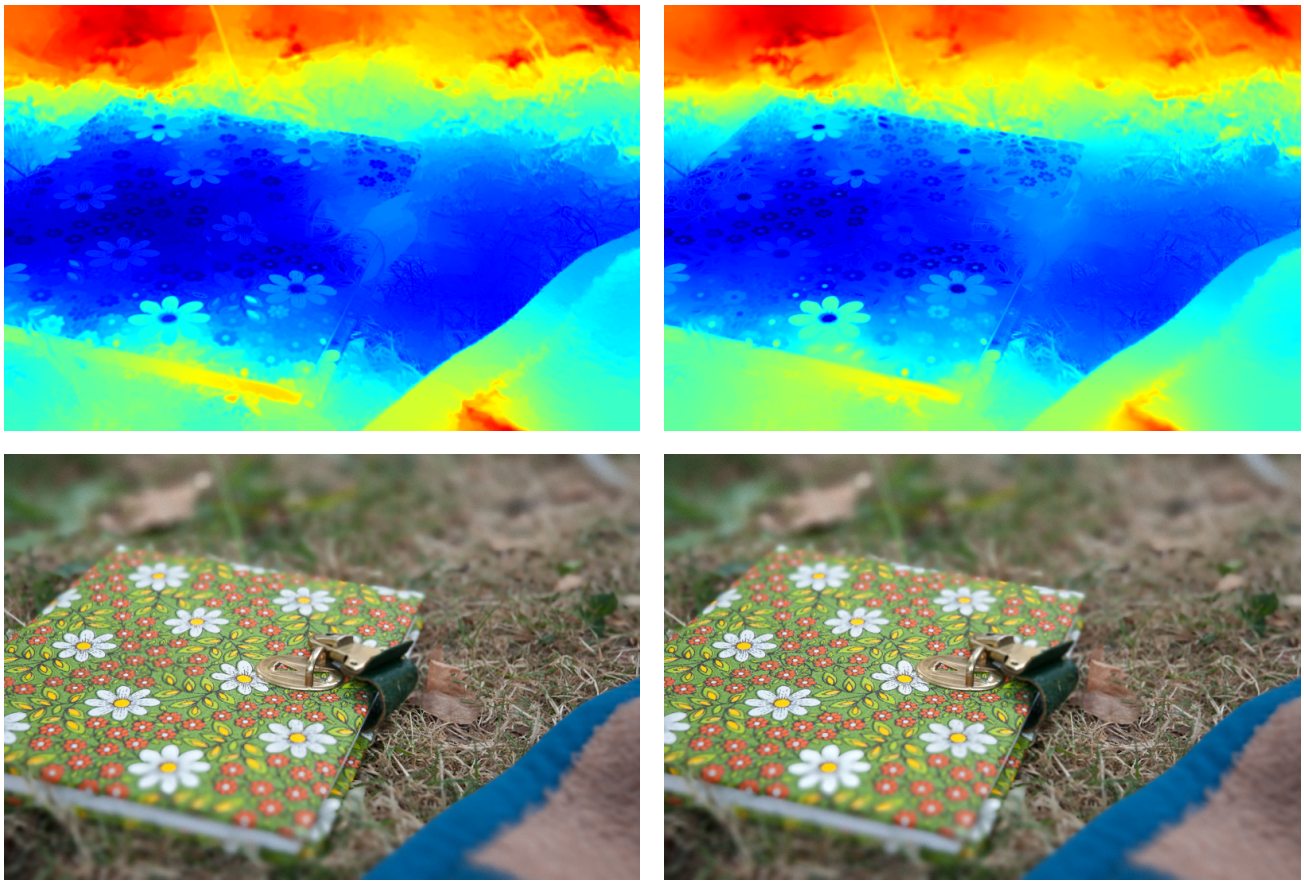


Figure 5: Side-by-side comparison of the direct method (left column) and our method (right column) both at 500 iterations. The top row shows the estimated full resolution blur maps and the bottom row the synthesized images. The differences in the blur maps are subtle but visible, compare e.g. the flower in the middle of the notebook. However, the differences in the result images are almost unnoticeable. It takes around 5 seconds to calculate the lower left image and around 34 seconds to calculate the lower right one.

3. OTHER APPLICATIONS

The described acceleration scheme could be used to accelerate a range of different techniques apart from defocus blur magnification. Hsu et al.[29] use the closed form α-matting algorithm by Levin to estimate how two different light sources mix across the scene. Using this mixture information they apply spatially-varying white balance to a photograph to remove color changes that are created by varying lighting colors or amplify them. He et al.[30] use the same α-matting algorithm to estimate the influence of haze in an image and remove it. The haze information could even be used to estimate depth if we assume a relationship. This again could be used to create a bokeh effect.

4. DISCUSSION

We successfully applied the method of Zhuo and Sim[2] which we modified for better performance by solving the computation intensive propagation step on a downsampled image followed by edge-aware upsampling.

We could achieve a significant speedup for the propagation step in defocus blur magnification. Although this comes at the cost of numerical accuracy the resulting images with applied defocus magnification are visually pleasing, which is what we were aiming for. The runtime for small images is acceptable, i.e. images around 0.5 million pixels take around 2 seconds to process. However, the runtime for state-of-the-art image sizes is still not fast enough: Modern DSLR sensors have 24 million pixels and more and even cameras embedded in

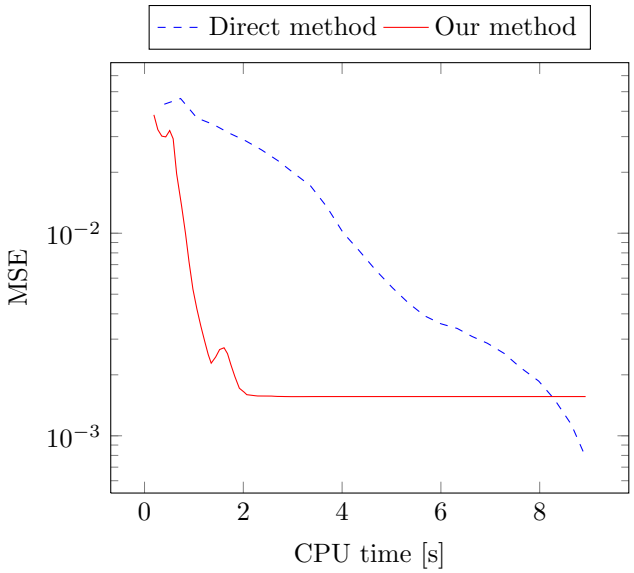


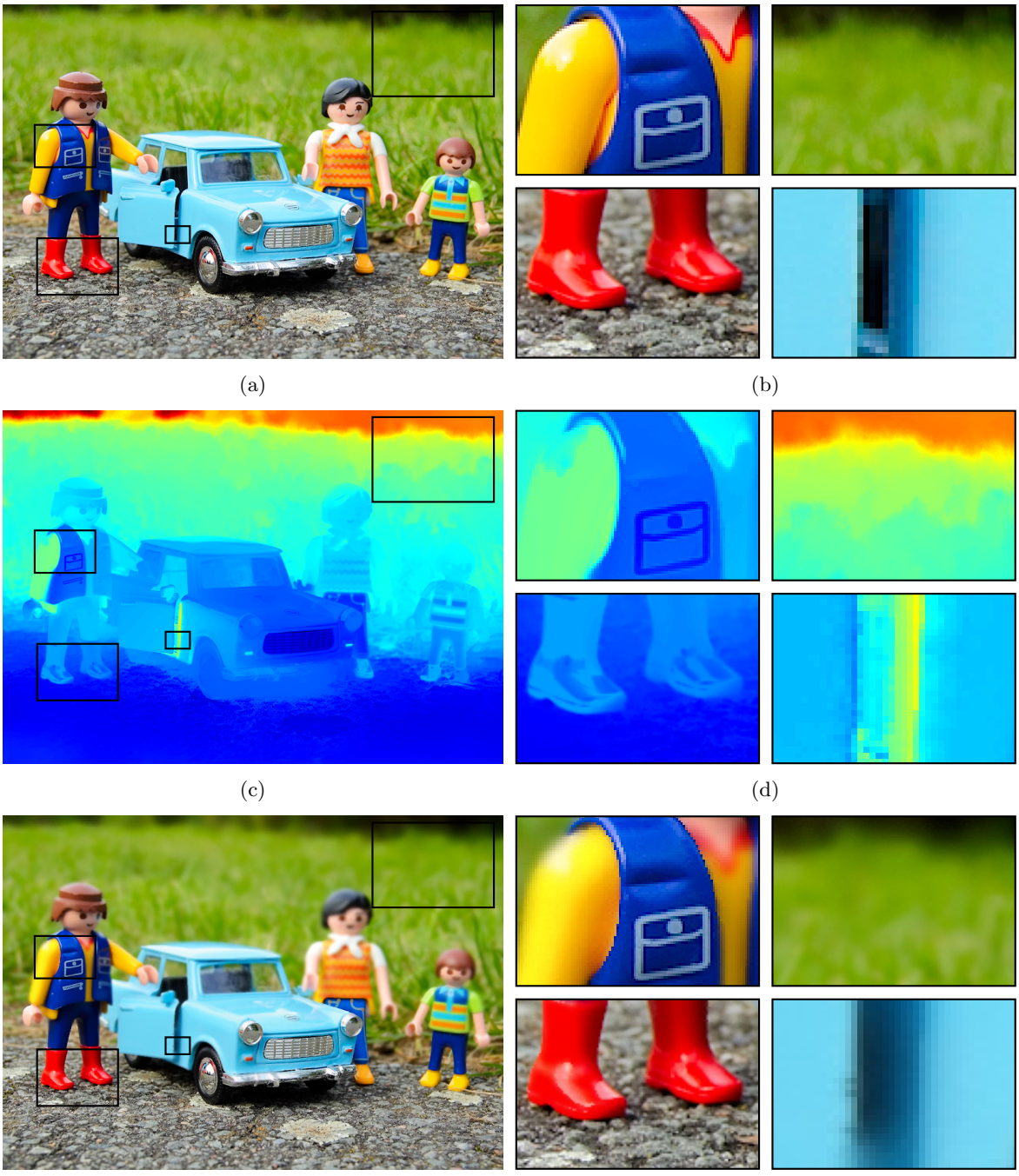
Figure 6: Plot of MSE vs. time for the direct and our method. The abscissa shows the CPU time that the propagation algorithm ran and the ordinate the MSE of the result compared to the result of the direct method at 500 iterations. Note that the MSE axis is logarithmic.

mobile devices produce images with more pixels. Therefore, we experimented with downsampling by a factor of 4 instead of 2. This, however, creates artifacts. If the blur information $\tilde{\boldsymbol{\alpha}}$ is too small, the optimization problem (7) will produce a result that orients more strongly on the guidance image $\boldsymbol{I}$. The result will look more like the original image and not like the blur map. Because the upsampling algorithm limits us to 2^k steps in upsampling we did not test values in-between 2 and 4. If we could test those values we would find that with decreasing scale size the influence of the original image increases. Therefore, it is questionable if we could improve our result that way.

A general problem of the approach, found in all variants of defocus blur magnification, is smooth surfaces in the image such as human skin or plastic toys. The blur estimation algorithm cannot distinguish between a blurred edge and a smooth rounded edge, e.g. a round plastic cylinder, because both have the same characteristics in a 2D image. This can lead to artifacts in such areas, see figure 7. Sometimes these artifacts are not visible (cf. figure 7 lower left in (b), (d) and (f)), because blurring a smooth rounded edge does not necessarily mean a corruption in the end result. However, this is not always true and sometimes results in artifacts in the end result (cf. figure 7 upper left in in (b), (d) and (f)). Some examples of this can be seen in the details right of the images in figure 7.

We could mitigate artifacts like those described above by providing a scribble based user interface that allows the user to mark regions as sharp or blurred. However, because of the high computational cost feedback cannot be given at once which could frustrate the user. Therefore, we are looking for ways to find better blur estimates in the first place that would allow a truly automated workflow for the blur map creation step.

Finally, a defocus blur must be present in the original image for the algorithm to work. This can easily be achieved with a large camera sensor and gets harder the smaller the sensor is. For cameras embedded in mobile phones, which have very small sensors (usually $1/2.5''$), a fixed aperture, and a wide angle lens (usually around 35 mm in 35 mm equivalent), the hyperfocal distance is at around 150 cm. This means that magnifing the defocus blur is only possible when the focus is set to an object near the camera. Thus for the application of defocus blur magnification on mobile devices a dedicated camera app that guides the user in shooting a photograph for defocus blur magnification could be a solution.



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

Figure 7: The top row shows the original image (a) with magnified details to the right (b), the row in the middle shows the blur map (c) with the same regions magnified (d), and the bottom row shows the end result (e) and details (f). From top left to bottom right in (b), (d) and (f): Rounded plastic surface cannot be distinguished from blurred surface leading to artifacts in the end result. Sudden change in blur on change of background texture does not effect the end result. Different surface textures create differences in estimated blur even though it should be the same with negligible artifact in the result. Smooth shadow on a smooth surfaces is interpreted as blur, no artifact in end result.

ACKNOWLEDGMENTS

We like to thank Torsten Beckmann for kindly providing the image shown in figure 7 and our colleagues Björn Beuthien, Daniel Fenner, and Falk Sticken for their helpful comments and suggestions.

REFERENCES

[1] Bae, S. and Durand, F., "Defocus magnification.," *Comput. Graph. Forum* **26**(3), 571–579 (2007).

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

[3] Nayar, S. and Nakagawa, Y., "Shape from focus: An effective approach for rough surfaces," *Proc. ICRA* **1**, 218–225 (1990).

[4] Hasinoff, S. W. and Kutulakos, K. N., "Confocal stereo," *Int. J. Comput. Vision* **81**(1), 82–104 (2009).

[5] Gaganov, V. and Ignatenko, A., "Robust shape from focus via markov random fields," *Proc. Graphicon Conference*, 74–80 (2009).

[6] Pentland, A., "A new sense for depth of field," *Pattern Analysis and Machine Intelligence* **9**(4), 523–531 (1987).

[7] Watanabe, M. and Nayar, S., "Rational filters for passive depth from defocus," *International Journal of Computer Vision* **27**(3), 203–225 (1998).

[8] Favaro, P. and Soatto, S., "A geometric approach to shape from defocus," *Pattern Analysis and Machine Intelligence* **27**(3), 406–417 (2005).

[9] Levin, A., Fergus, R., Durand, F., and Freeman, W. T., "Image and depth from a conventional camera with a coded aperture," *ACM Trans. Graph.* **26**(3) (2007).

[10] Favaro, P., Soatto, S., Burger, M., and Osher, S., "Shape from defocus via diffusion," *Pattern Analysis and Machine Intelligence* **30**(3), 518–531 (2008).

[11] Burge, J. and Geisler, W., "Optimal defocus estimation in individual natural images," *Proceedings of the National Academy of Sciences* **108**(40), 16849–16854 (2011).

[12] Zhou, C., Lin, S., and Nayar, S., "Coded aperture pairs for depth from defocus and defocus deblurring," *International Journal of Computer Vision* **93**(1), 53–72 (2011).

[13] Hasinoff, S. and Kutulakos, K., "Light-efficient photography," *Pattern Analysis and Machine Intelligence* **33**(11), 2203–2214 (2011).

[14] Jacobs, D., Baek, J., and Levoy, M., "Focal stack compositing for depth of field control," (2012).

[15] Elder, J. and Zucker, S., "Local scale control for edge detection and blur estimation," *Pattern Analysis and Machine Intelligence* **20**(7), 699–716 (1998).

[16] Levin, A., Lischinski, D., and Weiss, Y., "Colorization using optimization," *ACM Trans. Graph.* **23**(3), 689–694 (2004).

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

[18] Park, J., Kim, H., Tai, Y., Brown, M., and Kweon, I., "High quality depth map upsampling for 3D-TOF cameras," *Proc. ICCV*, 1623–1630 (2011).

[19] Yang, Q., Yang, R., Davis, J., and Nistér, D., "Spatial-depth super resolution for range images," *Proc. CVPR*, 1–8 (2007).

[20] Kopf, J., Cohen, M. F., Lischinski, D., and Uyttendaele, M., "Joint bilateral upsampling," *ACM Trans. Graph.* **26**(3) (2007).

[21] Chan, D., Buisman, H., Theobalt, C., Thrun, S., et al., "A noise-aware filter for real-time depth upsampling," *Proc. ECCV Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, 1–12 (2008).

[22] He, K., Sun, J., and Tang, X., "Guided image filtering," *Computer Vision – ECCV* **1**, 1–14 (2010).

[23] Potmesil, M. and Chakravarty, I., "A lens and aperture camera model for synthetic image generation," *SIGGRAPH Comput. Graph.* **15**(3), 297–305 (1981).

[24] Huhle, B., Schairer, T., Jenke, P., and Straer, W., "Realistic depth blur for images with range data," in [*Dynamic 3D Imaging*], Kolb, A. and Koch, R., eds., *Lecture Notes in Computer Science* **5742**, 84–95 (2009).

[25] Wu, J., Zheng, C., Hu, X., and Xu, F., "Rendering realistic spectral bokeh due to lens stops and aberrations," *The Visual Computer* **29**, 41–52 (2013).

[26] Golub, G. and Van Loan, C., [*Matrix computations*], vol. 3, Johns Hopkins University Press (1996).

[27] He, K., Sun, J., and Tang, X., "Fast matting using large kernel matting laplacian matrices," *Proc. CVPR*, 2165–2172 (2010).

[28] Crow, F., "Summed-area tables for texture mapping," *Computer Graphics* **18**(3), 207–212 (1984).

[29] Hsu, E., Mertens, T., Paris, S., Avidan, S., and Durand, F., "Light mixture estimation for spatially varying white balance," *ACM Trans. Graph.* **27**(3), 70:1–70:7 (2008).

[30] He, K., Sun, J., and Tang, X., "Single image haze removal using dark channel prior," *Proc. CVPR*, 1956–1963 (2009).