DEFOCUS MAGNIFICATION WITH CUDA

Chih-Wei Chen (陳智偉), Chun-Ta Lin (林君達), Yu-Lin Sung (宋侑霖)
, and Chiou-Shann Fuh (傅楸善)
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
E-mail:d98922005, r99922017, r99922104, fuh@csie.ntu.edu.tw

ABSTRACT

In photography, the application of depth-of-field can be used to make the main subject more prominent. Photographer can modify the range of depth-of-field by adjusting the aperture size. Unfortunately, due to the limitation caused by the physical diameter of the lens aperture and the area of the photodiode, the compact camera cannot control the depth-of-field as well as single-lens reflex camera. In this paper, we based on the method proposed by S. Bae [1] and propose another algorithm to interpolate the defocus map. Then, we use image processing approaches as post-processing to obtain a bokeh image. The above-mentioned approaches are implemented in CUDA (Compute Unified Device Architecture) to increase the integral throughput. Experimental results show that the modified method can be applied to emphasize the main subject in varied shot environment. In addition, we speed up the algorithm thus explore the possibility for real-time application.

Keywords: Defocus, Bokeh, Reblur, Blur measurement, Blur Interpolation, Depth-of-Field.

1. INTRODUCTION

In photography, the shallow depth field technique is usually applied in practice to emphasize the main subject. Photographer can modify the range of depth-of-field by adjusting the aperture. Unfortunately, due to the physical diameter limited in compact camera, the expected visual effect is generally hard to acquire in such devices. Even though the light-field camera which enables the post focusing ability for image acquisition, it is still difficult for them to be embedded in mobile device, where space is extremely limited.

In this research, our primary objective is to enhance the defocus region that is already in the photos. The term *enhance* here is different from its prevailing meaning. We have no intention to sharpen the images such as super resolution and de-blur techniques [2], instead, we want to further blur this region. Our approach can be mainly split into two stages, blur estimation from a single image stage and re-blur the image stage. We put emphasis on the estimation from single image to distinguish from other approaches.

Moreover, we design our algorithm in a parallelizable way and implement our approach on NVIDIA CUDA (Compute Unified Device Architecture) platform to improve the throughput of our algorithm. The intrinsic needs of edge-preserving lead us to apply cross bilateral filter in different stages. Based on the observation that bilateral filter is just another extended dimension, the algorithm is benefited from the separability of filter thus experiences a factor of performance gain.

The remainder of this article is organized as follows. In Section 2, we will have a short survey on other approach. In Section 3, we introduce the blur estimation algorithm and describe the framework step-by-step. Section 4 presents the blur interpolation approach in detail. In Section 5, some photographs and their magnified defocus results are presented. Finally, the conclusion is drawn in Section 6.

2. RELATED WORK

In computational photography field, there are several approaches to achieve the same effect. One of them is using hardware, where additional hardware

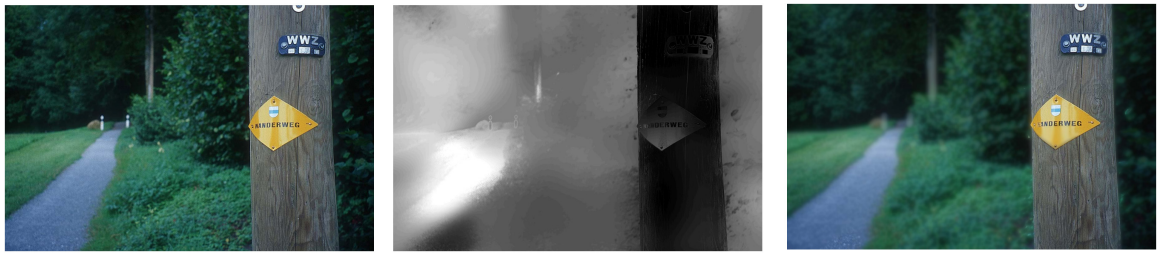


Figure 1: Our framework makes the main subject in photograph more prominent. (a) A given photograph. (b) Generated defocus map. (c) The result of shallow depth-of-field effect. This input photograph is from bigfoto.com, and the image size is 1183×784.

assistance is required. For example, light field [3] and light coding. Light field require an extra lens in front of the CMOS (Complementary Metal-Oxide-Semiconductor) sensor. We can imagine it as another optical lens that equipped in DSLR (Digital Single Lens Reflex Camera) and easily to understand that it might be challenge to scale down to space limited mobile device. Light coding is the technology Microsoft used in Kinect, where requires infra-red emitter and receiver. Despite the additional hardware that light coding demanded, there is a privacy issue raised as well. Users may concern that infra-red equipped mobile device might expose us under peeping risk. The other mainframe of achieving this effect is to use purely software solution. This can be further categorize into two, multiple inputs or single input. For multiple input algorithms, such approaches suffer from the limitation of temporal movement. Without the aid of tripod, optical flow or other image alignment methods should be applied first, where extra inaccuracies are introduced. For defocus magnification from single image, method proposed by S. Bae [1] could acquire satisfied result but suffered from performance issue raised by a large linear solver.

3. BLUR MEASUREMENT

The input image may contains many noises, e.g. salt-and-pepper noise, fixed pattern noise, and so on. Moreover, the compression algorithm in digital still camera may create unwanted block effect. Therefore, we apply a bilateral filter as preprocessing step to suppress the noise effect and preserve the detail information. After noise removal, we extract the gray-level values around the edge pixel from input image to obtain the blurred information. Then, we calculate the second-order derivative of the extracted intensities along the gradient direction. Figure 2 (a) and (c) shown the concept of the intensity extraction along the gradient direction on sharp edge and blur edge respectively. In implement, bilinear interpolation is applied to increase the continuous characteristic of an input digital image.

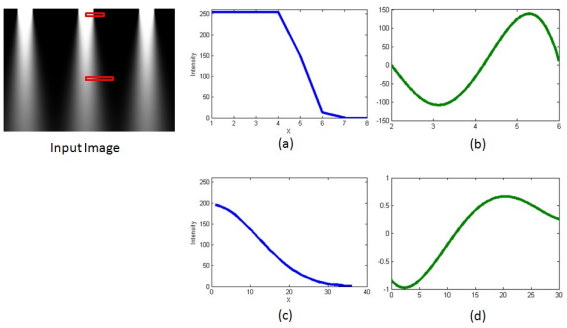


Figure 2: Concept of the intensity variance and the corresponding second-order derivative. (a)Sharp edge (upper rectangle in input image). (b)Second-order derivative of sharp edge. (c)Blur edge (lower rectangle in input image). (d)Second-order derivative of blur edge.

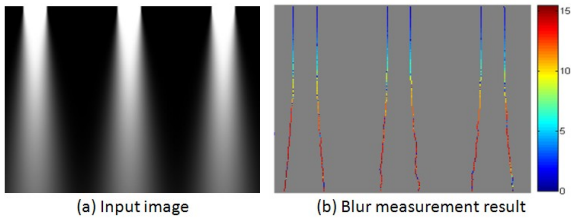
The result of the preliminary 2nd derivative is sensitive to the intensity variance. The inappropriate jitter in the preliminary 2nd derivative will cause the erroneous result of blur measurement. We use polynomial curve fitting approach (the degree of a polynomial is set to 5 in our implement) to resolve this problem. Figure 2 (b) and (d) shown

the modified 2nd derivative results. We fit the modified sigmoid curve by various response models and compare their mean square errors. The above response models are composited by second derivative of Gaussian blur kernel g, step function $u(x)$, and the local maximum A within sampling slice. The response model is defined by [1]:

$$r_x^2(x, y, \sigma) = Au(x) * g_x^2\left(x, y, (d/2)^2\right) \quad (1)$$

$$= \frac{-Ax}{\sqrt{2\pi}(d/2)^3} exp\left(-x^2/2(d/2)^2\right) \quad (2)$$

Since, the response models are steerable, we use brute force method to attempt to obtain the blurred values on edge pixels in photograph. This fitting process needs to try various σ in different slice sizes. In this research, the slice sizes are from 3 to 71. Figure 3 shown the result of the blur measurement.



(a) Input image (b) Blur measurement result

Figure 3: The result of the blur measurement. (a)Input image, this image size is 266×191. (b)Result of the blur measurement. Blue means the σ of the best fitting response model is small. Red means the σ of the best fitting response model is big.

4. BLUR INTERPOLATION

To generate complete blur estimation, several operations should be applied. In this section, we will first explain the fundamental data structure in detail and then demonstrate procedures based on this data structure to reach the final blur estimation.

We use a data structure called bilateral grid. It is an advanced data structure proposed by [4] aimed for performing bilateral like operation in less computation cost. Primary motivation behind bilateral grid is to convert the 2D-image into a 3D grid and perform desired operation on the grid. Finally, an inverse transformation is applied to obtain 2D-image. These three steps are called scattering, processing and slicing.

4.1. Scattering

First step of this data structure is creating the grid, which is called scattering. Scattering is the key concept that convert 2D image into 3D grid. Every pixel in the image will be mapped into a cell in grids, added as a homogeneous vector. It is worth to notice the grid data is extremely sparse, thus there is an incentive to further compress the data structure.

For clarity in explaining algorithm in following text, let us define few notation here, $Grid(x, y, z)$ represent the homogeneous vector in three dimension grid, $I(x, y)$ represent the intensity in 2D image and the compression rate in x-axis, y-axis and z-axis are C_x, C_y and C_z. With these notations, we now formally list the pseudo code for scatter as below:

Algorithm 1 Scatter($Grid, I, C_x, C_y, C_z$)

```
Initialize every cell in Grid to zero
for each pixel (x, y) in I do
    x' = rounding(x/C_x)
    y' = rounding(y/C_y)
    z' = rounding(z/I[x, y])
    Grid[x', y', z']+=(I[x, y], 1)
end for
```

Although it is tempting to directly parallelize the above algorithm, there exists a pitfall inside. It might be misinterpreted that every pixel can be independently handled. However, the $Grid(x, y, z)$ exist dependency. Two different solutions can be used to solve this. One of them is using atomic operation on each cell of the grid. However, this approach suffers from significant performance degrade. Another approach performs the grid computation in an inverse direction, listed below to ensure no simultaneous writing will occur.

4.2. Processing

After scattering, we now obtained a 3D grid. Operation applied in the 2D image can be used as usually on the bilateral grid with extra control on z-axis. How the z-axis ripples is actually the con-

Algorithm 2 Scatter Parallelized version($Grid, I, C_x, C_y, C_z$)

Initialize every cell in $Grid$ to zero
for each (x, y) pair in $Grid$, open a new thread **do**
  //calculate corresponding mapping in I
  $Grid[x/C_x, y/C_y, z/I[x, y]]$+=$(I[x, y], 1)$
  $x_{start} = x \times C_x$
  $y_{start} = y \times C_y$
  $x_{end} = (x + 1) \times C_x$
  $y_{end} = (y + 1) \times C_y$
  for each x_I in the range of $[x_{start}, x_{end})$ **do**
    for each y_I in the range of $[y_{start}, y_{end})$ **do**
      $Grid[x, y, I[x_I, y_I]] \mathrel{+}= (I[x_I, y_I], 1)$
    end for
  end for
end for

trol parameter on edge-preserving. Take bilateral filtering as an example; one can use a three dimension Gaussian blur directly to the grid. An obvious advantage should be further pointed out here. In the traditional bilateral filter, kernel varies in space. However, with the aid of grid, we use a single kernel for the whole processing. This leads to a separable convolution, hence reduce the complexity from $O(N^3)$ to $O(N)$.

4.3. Slicing

Finally, to transform back to image, a slicing procedure is required. Basic concept of slicing is to perform inverse operation in opposite direction of scattering.

Algorithm 3 Slice($Grid, Output, C_x, C_y, C_z$)

for each (x, y) in $Output$ **do**
  //Using bilinear interpolation to get
  //homogeneous vector at
  $Grid[x/C_x, y/C_y, z/I[x, y]]$
  $vec = Grid[x/C_x, y/C_y, z/I[x, y]]$
  //Converting vector to normal representation
  $vec = vec/vec(3)$ //Fourth World element
  $Output[x, y] = vec$
end for

This procedure is safe to parallelize, however, special handling on image border and divisor is required.

4.4. Cross Bilateral Filter (CBF)

The last step of explaining our interpolation algorithm is to modify above mentioned to cross bilateral filter. A cross bilateral filter [5] is taking two images as sources and proceed bilateral filtering on one of the image. One of the image is called *edge* image, used to calculate the kernel to convolve. Another one called *data* image is the one to convolve with the computed kernel. To unify the notation, let $Grid(x, y, z)$ represent the homogeneous vector as before , $I_{edge}(x, y)$ represent the intensity of edge image, I_{data} represent the data image. The modified algorithm is listed as algorithm 4.

Algorithm 4 Scatter Parallelized version for CBF($Grid, I_{edge}, I_{data}, C_x, C_y, C_z$)

Initialize every cell in $Grid$ to zero
for each (x, y) pair in $Grid$, open a new thread **do**
  //calculate corresponding mapping in I
  $x_{start} = x \times C_x$
  $y_{start} = y \times C_y$
  $x_{end} = (x + 1) \times C_x$
  $y_{end} = (y + 1) \times C_y$
  for each x_I in the range of $[x_{start}, x_{end})$ **do**
    for each y_I in the range of $[y_{start}, y_{end})$ **do**
      $Grid[x, y, I_{edge}[x_I, y_I]] \mathrel{+}= (I_{data}[x_I, y_I], 1)$
    end for
  end for
end for

The remaining procedures are mostly the same as before, except using I_{edge} as I in slicing procedure.

4.5. Two-pass interpolation simulation by CBF

With all the algorithms prepared, we now generate the interpolated blur estimation in two-pass. Because estimation errors exist in the blur measurement stage, in order to increase the robustness of this algorithm, we apply a CBF on the estimation before we further process, which is the first pass. Figure 4 shown the difference before and after CBF.

In the second pass, we propagate the known estimation to the dark blue pixels by an abnormal large bilateral. With compression rated $C_x = 8$ and $C_y = 8$, the execution time is reduced to a factor of 64 in spite of benefit brings by parallelism.

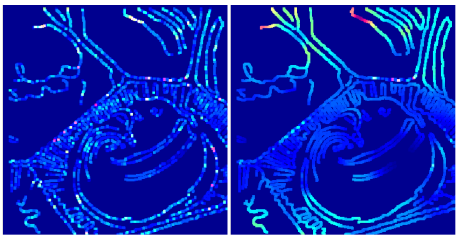


Figure 4: The difference before (left image) and after (right image) cross bilateral filter.

5. EXPERIMENTAL RESULTS

Due to the result of the blur measurement and the image size are interdependent and inseparable, we use varied size of images as input to evaluate the performance. Figure 6 shown our processed results and the corresponding input photographs. The first column is the input, the second column is the generated defocus map, and the last column is the corresponding result of shallow depth-of-field effect. The input photographs (c), (d), (f), and (g) are from bigfoto.com, photograph (b) is from dofpro.com, photograph (a) is from sfl94.over-blog.com, and photograph (e) was token by using Canon PowerShot G7 with 7.4mm lens at f/2.8. Figure 5 gives the comparison between S. Bae and our approach.

6. CONCLUSION

To judge the performance of defocus magnification is quite subjective. In this research, we generate the defocus map by using blur measurement and blur interpolation approach, and re-blur the input image by the generated defocus map. There still are many challenging minutiae, including harmonic defocus magnification in application, defocus magnification for portable device, and so on.

ACKNOWLEDGEMENT

The authors would like to thank Prof. Wei-Chao Chen in Department of Computer Science and Information Engineering, National Taiwan University for kindly helping to give us many useful suggestions and guide us.

REFERENCES

[1] Bae, S., and Durand, F. 2007. Defocus magnification. *Computer Graphics Forum 26*, 3 (Sept.), 571–579.

[2] Fergus, R., Singh, B., Hertzmann, A., Roweis, S. T., and Freeman, W. T. 2006. Removing camera shake from a single photograph. *ACM Trans. Graph. 25*, 3 (July), 787–794. ACM ID: 1141956.

[3] Liang, C.-K., Lin, T.-H., Wong, B.-Y., Liu, C., and Chen, H. 2008. Programmable aperture photography: Multiplexed light field acquisition. *ACM Transactions on Graphics 27*, 3, 55:1–55:10.

[4] Chen, J., Paris, S., and Durand, F. 2007. Real-time edge-aware image processing with the bilateral grid. In *ACM Transactions on Graphics (TOG)*, ACM, New York, NY, USA, SIGGRAPH '07. ACM ID: 1276506.

[5] Paris, S., and Durand, F. 2009. A fast approximation of the bilateral filter using a signal processing approach. *Int. J. Comput. Vision 81*, 1 (Jan.), 24–52. ACM ID: 1487517.

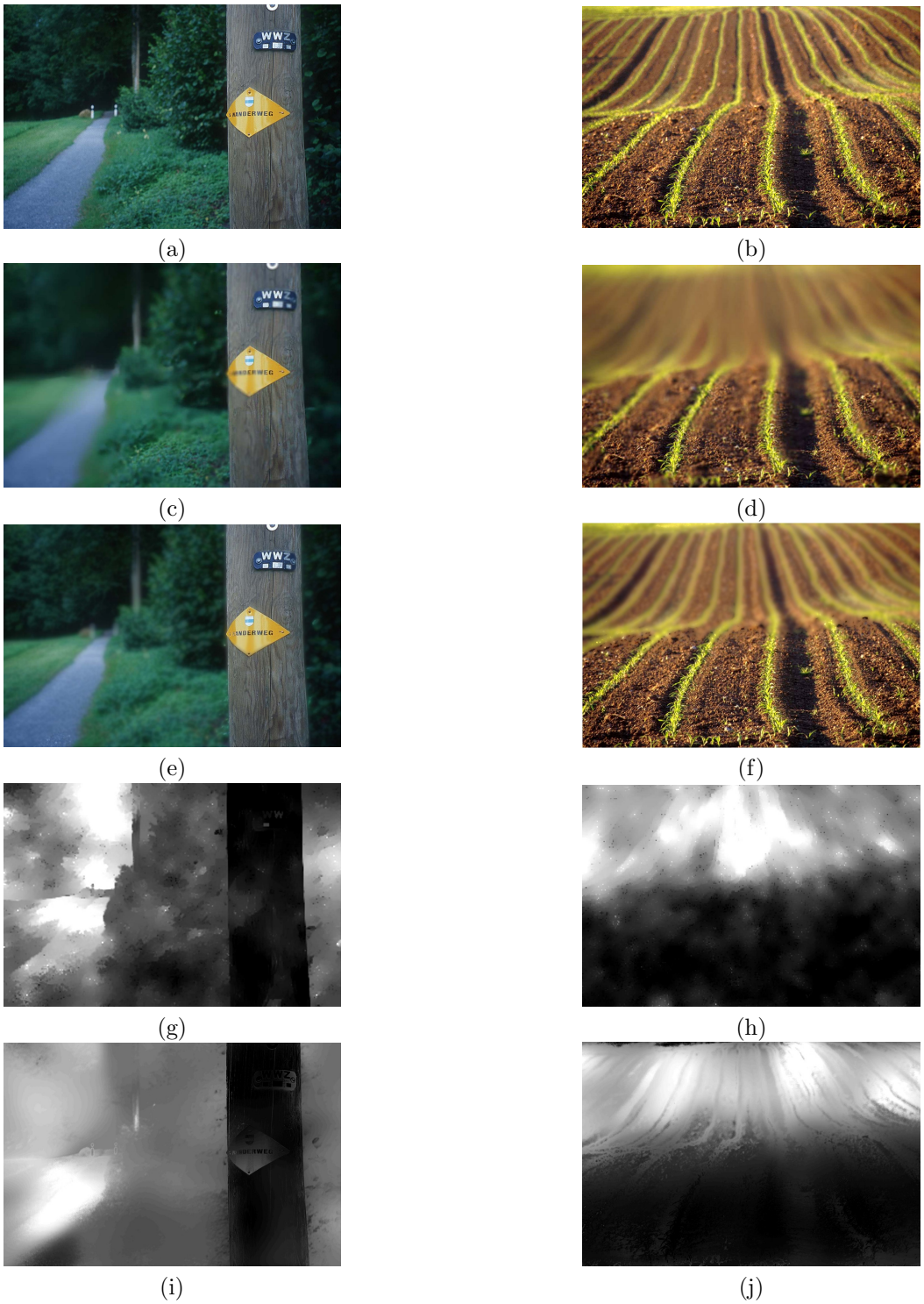


Figure 5: The comparison between S. Bae and our approach. (a) and (b) are the input images. (c) and (d) are Bae's results, (g) and (h) are the corresponding defocus maps respectively. (e) and (f) are our results, (i) and (j) are the corresponding defocus maps respectively.

Figure 6: Comparison the input photographs with the processed results. The first column is the input, the second column is the generated defocus map, and the last column is the corresponding result of shallow depth-of-field effect. The input photographs (c), (d), (f), and (g) are from bigfoto.com, photograph (b) is from dofpro.com, photograph (a) is from sfl94.over-blog.com, and photograph (e) was token by using Canon PowerShot G7 with 7.4mm lens at f/2.8.