Depth of Field Rendering Using Multilayer-Neighborhood Optimization

Benxuan Zhang, Bin Sheng, Ping Li, and Tong-Yee Lee, *Senior Member, IEEE*

Abstract—Depth of field (DOF) is utilized widely to deliver artistic effects in photography. However, existing post-processing techniques for rendering DOF effects introduce visual artifacts such as color leakage, blurring discontinuity, and the partial occlusion problems which limit the application of DOF. Traditionally, occluded pixels are ignored or not well estimated although they might make key contributions to images. In this paper, we propose a new filtering approach which takes approximated occluded pixels into account to synthesize the DOF effects for images. In our approach, images are separated into different layers based on depth. Besides, we utilize adaptive PatchMatch method to estimate the intensities of occluded pixels, especially in the background region. We again propose a new multilayer-neighborhood optimization to estimate occluded pixels contributions and render the images. Finally, we apply gathering filter to achieve the rendered images with elite DOF effects. Multiple experiments have shown that our approach can handle color leakage, blurring discontinuity and partial occlusion problem while providing high-quality DOF rendering effects.

Index Terms—Depth of field, multilayer, neighborhood optimization, rendering

1 Introduction

Depth of field (DOF) describes the range of objects that appears to be sharp in an image, which is used extensively in photography to produce artistic effects. The scene points outside of the range of the DOF map to circular regions in the image are called the circle of confusion (CoC), while points within the depth of field appear to be focused. The DOF effects enhance the perception of depth and makes images natural. However, we could not obtain an image with DOF effects without optical camera except via post-processing. Thus, it will significantly enhance the performance of camera if realistic DOF effects can be achieved via post-processing methods. Therefore, DOF rendering has become a significant field for obtaining realistic images.

Numerous methods have been dedicated to rendering DOF effects [1], [2], [3], [4]. These methods can be mainly classified into three groups: multipass approaches [5], [6], light field approaches [7], [8], [9], and post-processing approaches [10], [11], [12], [13], [14], [15]. Multipass approaches operate on 3D scene representations, and light field approaches rely on computational photography [16], [17] and special equipment support [18]. Consequently, they are not applicable to image-based DOF rendering due to lack of 3D scene data information. Post-processing approaches work in the image space with corresponding depth information. They calculate each pixel's blurring degree (CoC) and each pixel of the image is blurred based on CoC. Post-processing methods can further be categorized into three types of approaches on the basis of how the blurring is performed, namely, scattering, gathering, and layered. Scattering methods scatter the color of a pixel to all the pixels lie in its CoC. Gathering methods gather all the color from a pixel's CoC to synthesize the pixel's intensity. Both of them suffer from color leakage problems when foreground color gets into focused region. Layered methods decompose the image into layers and can reduce color leakage problem and partial occlusion. However, the methods might cause blur discontinuity problems.

In this paper, we propose a spatially variant DOF synthesis from a single input image with depth information. The occlusion problem is discussed and illustrated by Barsky et al. [19]. The colors from parts of the scene behind objects are missing in image due to pinhole camera model. Thus, methods that approximate the intensity of occluded pixels are proposed [20]. They apply a GPU-based pyramidal interpolation method to estimate occluded pixels. Lee et al. [21] proposed a layered image-based method to approximate different views in order to obtain good estimation of occluded pixels. Lu et al. [22] extended the bilateral filtering into motion bilateral filtering in order to reduce temporal flickering. Unlike their method, our method uses patch-based processing to estimate the intensity of blocked pixels. Besides, we propose a recursive multilayer-neighborhood optimization to estimate the contributions of blocked pixels, which offers us more accurate results. Fig. 1 shows the DOF rendering effects of our method focused at different depths. Note that our approach provides realistic

- B. Zhang and B. Sheng are with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China. E-mail: 150576806@qq.com, shengbin@sjtu.edu.cn.
- P. Li is with the Faculty of Information Technology, Macau University of Science and Technology, Macau 999078, China. E-mail: pli@must.edu.mo.
- T.-Y. Lee is with the Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan 70101, Taiwan. E-mail: tonylee@mail.ncku.edu.tw.

Manuscript received 8 July 2018; revised 6 Jan. 2019; accepted 18 Jan. 2019. Date of publication 23 Jan. 2019; date of current version 6 July 2020. (Corresponding author: Bin Sheng.) Recommended for acceptance by K. Zhou. Digital Object Identifier no. 10.1109/TVCG.2019.2894627

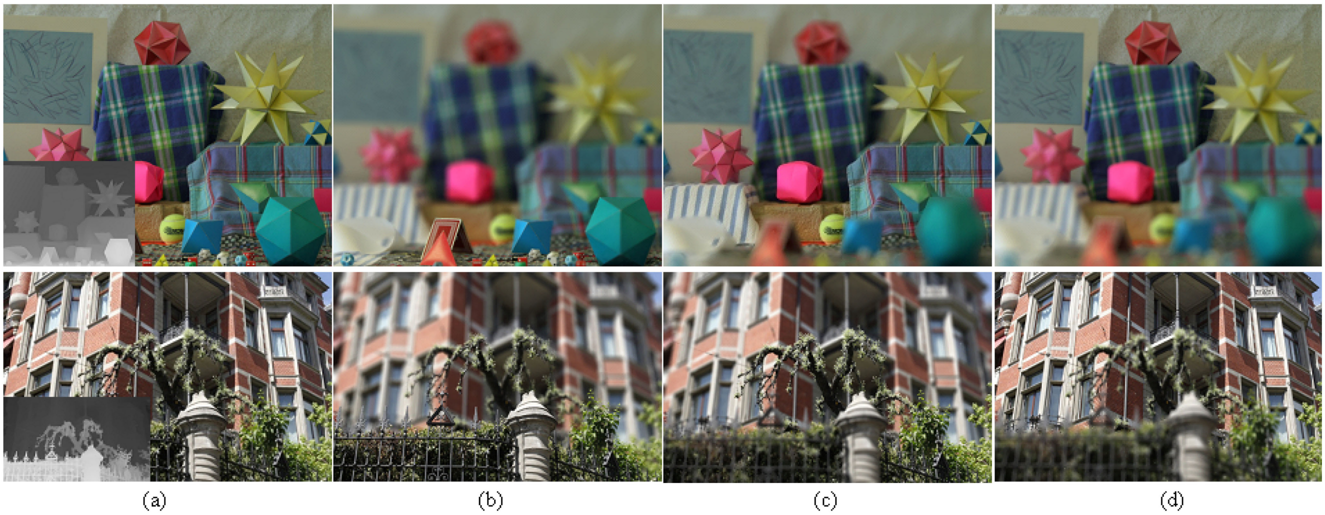

Fig. 1. DOF effects generated by our DOF rendering approach focused at various depths. (a) Original all in-focus image and depth map, (b) focus on the foreground, (c) focus on the middle depth, and (d) focus on the background. Our approach generates realistic DOF effects without color leakage, blurring discontinuity, and partial occlusion.

DOF rendering, and is able to handle well the color leakage, blurring discontinuity, and partial occlusion. Our approach makes an assumption that the pixel depth information is available through existing techniques such as stereo camera and depth camera, and in one-to-one pixel correspondence with original images. Although the depth images acquired from real world are often in low-quality with holes. We demonstrate that depth images with holes inpainted can be used for producing realistic DOF rendering effect. Therefore, our approach can be effectively incorporated into existing RGB-D image/video applications/hardware (such as Kinect and RGB-D cameras), which provide both color image and depth information. This offers a compelling alternative to the current state-of-the-art techniques which include a filter-based ad hoc method to approximate DOF effects. The main contributions of our work are as following:

1) A new patch-based blocked pixels estimation method that preserves spatial coherence, which is later used in DOF rendering by applying recursive multilayer-neighborhood optimization.
2) A new recursive bilateral filtering approach for estimating the contributions of blocked pixels.
3) We achieved realistic DOF rendering in real-time via GPU and demonstrated that it can further be used in mobile devices with various applications potential.

Our paper is organized as: Section 2 introduces the previous work about depth of field rendering and related methods involved in our approach. Section 3 illustrates our proposed approach for DOF rendering in detail. Section 4 shows the experimental results of our methods and makes comparison with previous work. Finally, we give out the conclusion and future directions of our work in Section 5.

2 Related Work

Several DOF rendering methods exist in the image processing and graphics fields. They can be categorized into three groups: multipass approaches, light field approaches, and post-processing approaches.

2.1 Multipass Approaches

Multipass approaches are utilized in 3D scenes and render the DOF effects in the 3D graphics rendering pipeline. The accumulation buffer method [6] and the distributed ray tracing method [5] use such approaches. They use individual cameras to render the scene multiple times and accumulate all sample results. Consequently, those methods can achieve simulating most accurate DOF effects. However, it is computationally expensive to calculate for a large number of repeated rendering. Therefore, these strategies are generally considered for offline rendering. In recent times, some advanced strategies were proposed to generate real-time DOF effects based on ray tracing. McGuire et al. [23] proposed a stochastic rasterization algorithm and can produce DOF effects. It can render impressive effects for complex 3D scenes. However, the accuracy of blur depends on the increasing of sampling, which reduces the time for rendering. Lee et al. [24] built the scene into layers using depth peeling, which can accelerate ray tracing process for DOF effects. These multipass approaches are much easier to be incorporated in VR applications. Nonetheless, they are not applicable to DOF rendering based on images because previous 3D scene information and the accurate rays for accumulation is hard to obtain in image-based methods.

2.2 Light Field Approaches

These strategies are widely used in computational photography that can change/broaden DOF using extra devices or by changing the camera system to obtain additional data. The coded aperture method [7] embeds a designed occluder into the aperture of the camera lens, the focus sweep method [8] uses a special lens with strong chromatic aberration, and the lattice-focal lens method [9] organizes lens patches with different focal powers. These strategies can generate arbitrary images of different focus point after it is shot, but it relies on special hardware and large memory space which can be computing expensive. Yu et al. [25] created the light field through warping and then rendered DOF effects in real-time using GPU-based parallel processing. However, it results in accumulated visual artifacts when rendered with large CoC due to occlusion.

2.3 Post-Processing Approaches

Post-processing approaches work on image space rather than 3D scene. It first takes a clear image and corresponding depth image as input and calculates the size of CoC using thin lens camera model. Finally, it blurs the image based on CoC of every pixel. This kind of method can be further classified into three categories: scattering-based methods, gathering-based techniques, and layered methods [26].

Scattering-based methods scatter the intensity of every pixel to its neighbor pixels according to size of CoC. Next, the intensities of each pixel are blended based on varying depths. Potmesil and Chakravarty [10] proposed the first scatter method for DOF rendering using a direct linear filter. However, there is a color leakage problem with their method which causes the color of background blur blends with the sharp focused region; Shinya [11] proposed a technique that uses a ray distribution buffer to process the blend order to reduce color leakage issue. However, the technique is both time-consuming and space-consuming due to heavy sorting of the pixel depth. Yan et al. [12] built up an interactive framework that develops depth maps and uses the scattering method to blur the background. Lee et al. [27] proposed a per-pixel layered splatting based method for rendering a depth-of-field result which can enable real-time post-processing. We combine the advantage of this scattering method without considering ray tracing which decreases computational time.

Gathering-based methods, on the other hand, use spatial filtering of the neighboring pixels in the CoC of pixel to achieve blurring effects [13]. These methods are efficient than scattering methods in general due to fast execution of filtering. Further, such techniques can exploit the texture look up features of modern GPUs. Zhou et al. [15] proposed a gathering-based method using separable kernel filtering. A technique by Xue et al. [28] utilizes saliency as a depth cue and a non-local means filter to blur the background and foreground regions. However, most gathering-based techniques suffer from color leakage problem. Lee et al. [14] proposed one that uses the anisotropic mipmap interpolation method. We also apply the anisotropic filter for DOF rendering with uniform blurs rather than Gaussian blurs, which is more approximate to actual cameras.

Layered methods decompose the image into layers and then blur and blend them to get a final result [19], [29], [30]. These methods reduce color leakage and alleviate partial occlusion by extrapolating layer boundaries. However, blending of layers can cause new discontinuity artifacts. Kraus and Strengert [20] proposed a pyramidal processing technique, and the rendering results are comparable to distribution ray tracing techniques. However, their layered technique is too sophisticated. Our proposed method combines the scattering and layered methods, and incorporates the concept underlying the layered method to rectify the common artifacts such as color leak and give a better rendering effect in foreground blurring. In addition, our proposed method has excellent parallelism capabilities and can render DOF effects in real-time.

3 Approach

This section describes in detail the methodologies utilized as a part of our proposed DOF rendering. An overview of the framework is given first, followed by background information about depth of field. Finally, every component of our approach is explained in detail.

3.1 Overview

Our proposed approach comprises the following steps: Given a color image and the corresponding depth map, we first calculate an R_{CoC} map containing the range of each pixel's blurriness based on the acquired depth map. Then we apply Simple Linear Iterative Clustering (SLIC) segmentation algorithm [31] on the depth map so that we divide the pixels in the similar depth into several layers. We process each pixel based on its layer and adjacent information. We use PatchMatch method to guess the intensity of occluded pixels near the boundary and apply multilayer-neighborhood optimization to estimate the contributions of each pixel. The framework of our proposed approach is shown in Fig. 2, and also described in Algorithm 1.

Algorithm 1. General Framework of our Approach

```
1: Calculate R_CoC for each pixel;
2: Apply SLIC algorithm to divide image into layers;
3: Use adaptive generalized PatchMatch to guess occluded
   intensity of pixel q' as I(q') via EM algorithm;
4: for each pixel p in the image do
5:   for each nearby pixel q do
6:     if q lies in closer layer then
7:       Calculate the maximal weight of blocked pixels
         whose CoC contains pixel q;
8:     end if
9:   end for
10: end for
11: for each pixel p in the image do
12:   Calculate weighted sum of contributions of visible and
      invisible pixels as the intensity value of pixel p;
13: end for
```

3.2 DOF Rendering

We adopt the classic thin lens formula [32] of the radius of the CoC with blur control parameter α as follows:

$$R_{CoC}(p) = \alpha \cdot \frac{|d_f - d(p)|}{d(p)}, \tag{1}$$

where R_{CoC} is the radius of CoC, d_f is the focused depth, and $d(p)$ is the depth of the pixel p. All parameters in Eq. (1) are lessened to only two parameters in the above equation: d_f and α. d_f represents the focused depth and α represents the settings parameters of the camera and aperture. It controls the blurriness of the defocused region. Since we are dealing with raster images, R_{CoC} needs to be measured in pixels. Thus, the unit for alpha is pixel. Further, since depth value d is normalized in the range 0-255, d_f also needs to be set to a normalized value between zero and 255. The pixels that have R_{CoC} less than one pixel are considered to be acceptably sharp, and the depth range of the acceptably sharp pixels is from d_{fmin} to d_{fmax}.

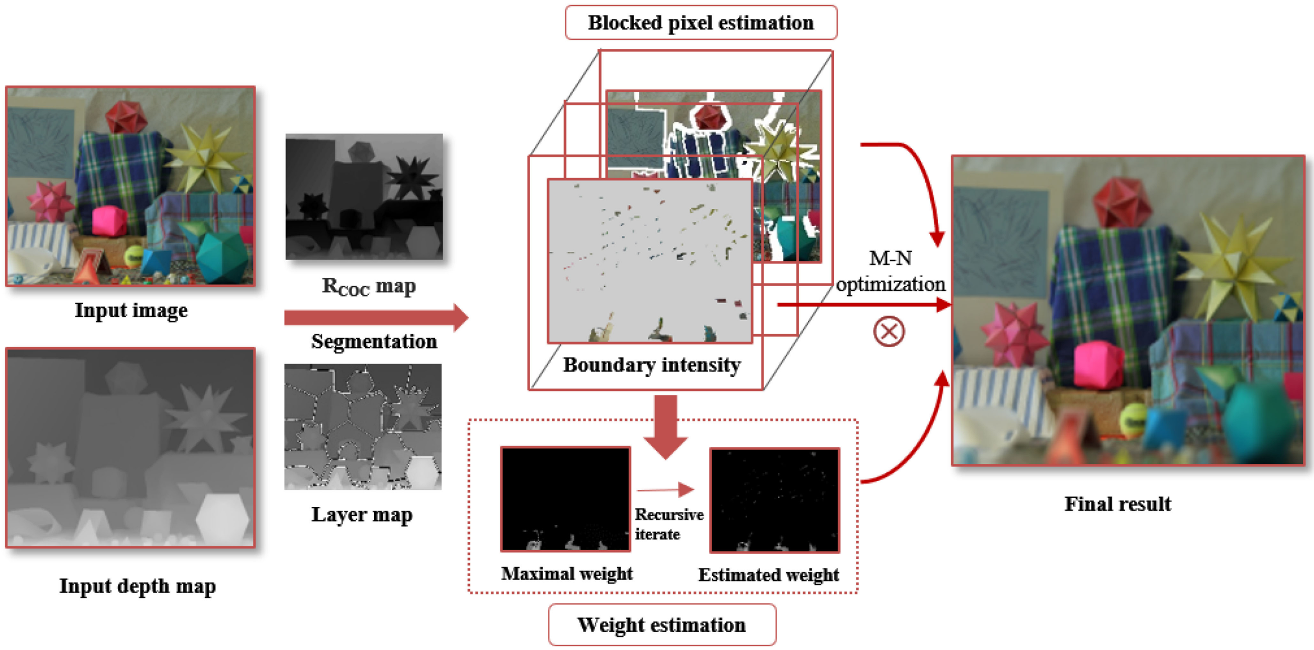

Fig. 2. Framework of our proposed DOF rendering approach. We first calculate the R_{CoC} map and the layer map based on the depth map. Then we use adaptive PatchMatch method to fill the blocked pixels near the boundary. By applying multilayer-neighborhood optimization (M-N optimization in figure) over the maximal weight map using weight as guidance, we obtain weight estimation map and use gathering filter to get the final result.

The major component of our proposed DOF rendering method is a new patch-based blocked pixels estimation method especially for those pixels lie in the neighbor of blurred boundary. Thus, we divide the image region into three regions: foreground region $\Omega_{\text{foreground}}$, in-focus region $\Omega_{\text{in-focus}}$ and background region $\Omega_{\text{background}}$.

$$p \in \begin{cases} \Omega_{\text{foreground}}, & d(p) < d_{fmin} \\ \Omega_{\text{in-focus}}, & d_{fmin} < d(p) < d_{fmax} \\ \Omega_{\text{background}}, & d(p) > d_{fmax}. \end{cases} \quad (2)$$

The artifacts such as color leakages problems are often visible in the region between background and in-focus region and partial occlusion problems are often visible in between in-focus and foreground region. In order to address these problems, we apply SLIC segmentation algorithm to divide it into finer layers and apply patch-based blocked pixels estimation method to tackle such artifacts.

3.2.1 *The Influence of Blocked Pixels*

First of all, we clarify the overlapped influence of pixels in different regions and show our intuition on how to solve these problems. As discussed above, artifacts are generated near the boundary. From perspectives of the optic of thin lens camera, the boundary between foreground region and background region should be blurred continuously. Thus, gathering filter or scattering filter can be applied directly. But the boundary between background and in-focus region should be clear while using scattering filter will result in color leakage problem. Finally, the boundary between foreground and in-focus region is continuous in blur. Besides, occluded pixels will also influence the rendered DOF results. In Fig. 3, there are three different layers overlapping in the blue rectangle region. Considering the rendering of yellow regions, yellow regions block green and red regions. What we do here is not to simplify the blocked regions into one hidden pixel, we need to extend the green and red regions into yellow region as shown in black arrows in the right figure. Thus, there are more than one layer involved in contributing final pixel. For real scenes, if there are multiple layers overlapped, we extend the further layer to estimate the influence of hidden pixels in different layers.

Then, we discuss the potential influence of blocked pixels. As show in Fig. 4, p is a blocked pixel in the background plane while q is a boundary point between in-focus plane and background plane. q' is the focused point in the image plane. Most of rays from p are blocked by in-focus plane except that between two orange rays. Since these rays will not affect the pixels below q' in the image plane, the blocked pixels make no contributions to pixels in the in-focus plane. Furthermore, when q lies in a closer background layer or foreground plane, q projects into a circle which overlap with the CoC of p. Thus, blocked pixels p will make

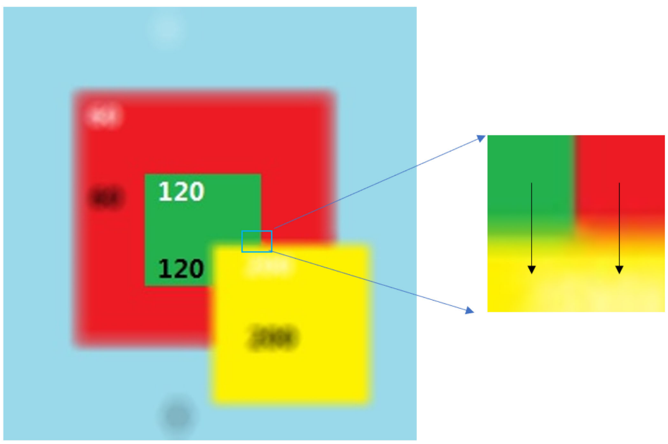

Fig. 3. This picture illustrates overlapping case in blur rectangle region. We need to estimate the hidden pixels behind front layer in yellow.

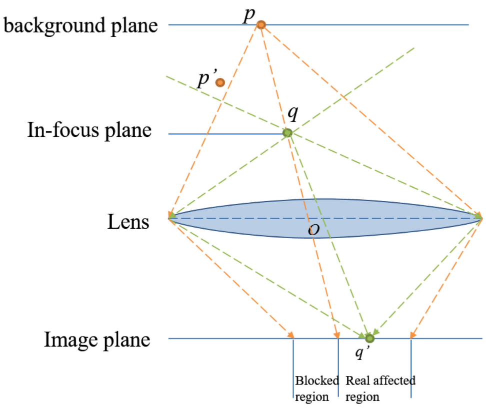

Fig. 4. This is a picture illustrating how the blocked points interacts with other points. p' is blocked by front plane while q is boundary point. q' is projected point in image plane. Rays between orange lines makes contributions to blur.

Fig. 5. The first column is the result of applying SLIC segmentation on the depth image. We group similar depth into same layers. The white pixels are holes for the image in the middle column. And the last column is the estimated intensity of holes.

contributions to those pixels in a closer layer. Besides, pixel hidden behind in-focus plane like p' might make contributions to image, too. Traditional methods ignore or approximate the colors of such pixels while we propose a new patch-based approximating approach for estimating the intensity of blocked pixels which will further be used in weight estimation.

3.2.2 Intensity Estimation

As in Algorithm 1, we start our approach pixel by pixel. Since the depth value d_f has been normalized between 0-255, there are at most 256 layers. To preserve the continuity of the surface, we first use SLIC superpixels segmentation method [31] to partition the region into several small regions. Original SLIC algorithm divides each input image into superpixels by considering image texture. We adopt SLIC method which groups pixels using five-dimensional color and image intensity space to generate uniform superpixels. Individual pixel is represented by a vector (l, a, b, x, y, d) where (l, a, b) is the color vector and $[x, y]$ is the space vector and d is the depth value. Then, it clusters the pixels base on the similarity between such six-dimensional vector.

In Fig. 5, we can see the segmented result. Although it is not accurate enough, we proceed our approach pixel by pixel based on its depth where segmentation result matters little. After we get the different layers of the depth, we can consider the influence of pixels near the boundary of two adjacent layers. As we have analyzed for Fig. 4, we should estimate the potential contributions of blocked pixels p'. The intuition behind this estimation is that blocked pixels are most likely continuous both in intensity and depth value. PatchMatch [33] is a fast searching for find most similar patch in the source image in color space. Inspired by this, we apply patch-based estimation method inspired by Image Melding [34]. PatchMatch and its generalized algorithm are efficient in finding similar patches for image completion, retargeting and reshuffling [35], [36]. We want to estimate both the intensity and depth value of blocked pixels which are unknown pixels near the boundary defined by above SLIC method. We start with the input image in which there are holes near the boundary. We want to optimize energy function as follows:

$$E(T, S) = \Sigma_{q \in T} \min_{p \in S}(s(Q, P) + \lambda(\nabla Q_{depth}, \nabla P_{depth})), \quad (3)$$

where S denotes source input image with holes, T denotes target reconstructed image. $Q = N(q)$ is a patch of size $w \times w$ in source image S and pixel q lies in the top left corner of patch. Similarly, $N(p)$ is a patch of size $w \times w$ in target image T. $P = f(N(p))$ where f is a function including translation, scaling and rotation on a patch. s is a function that calculates sum of square of differences between two patches. P and Q only considers three color channels in each image. ∇Q_{depth} and ∇P_{depth} represents the relative depth which is used to preserve depth coherence. And λ is used to control the ratio between intensity and depth differential. Furthermore, we consider depth value only for $(\nabla Depth_x, \nabla Depth_y)$, where

$$\nabla Depth_x(i, j) = Depth(i, j+1) - Depth(i, j) \quad (4)$$

$$\nabla Depth_y(i, j) = Depth(i+1, j) - Depth(i, j) \quad (5)$$

is the relative depth information. Thus, T is the reconstructed image which is the most similar to original source image S both in color and depth space. If we can solve this energy function. We are able to estimate the color intensities and relative depth information for blocked pixels.

Eq. (3) describes the difference between source image S and destination image T in color space and relative depth. We want to find an output image such that S and T both are similar in color and relative depth value. Here, we choose to use relative depth instead of direct depth value as a constraint. Because the similar structures are translation invariant. Besides, we calculate all the difference between most similar patch in each image as a basic unit. In order to solve this energy equation, we apply expectation-maximization algorithm (EM algorithm) under multi-resolution to find local optimal solution. First, we randomly initialize the intensity of holes. Then, we iterate through two steps. We use adaptive PatchMatch to find most similar patch to those patches containing holes (Expectation Step). Then, for every pixel in holes, we calculate weighted average of all the similar patches that contains holes as the initial value for next iteration (Maximal Step). Here are the detailed discussions.

In Expectation step, we propose an adaptive generalized PatchMatch approach for searching patches that are similar both in intensity and color. We adopt following constraint:

$$D(P,Q) = \sum_{x,y} \sqrt{\sum_{c\in\{r,g,b\}} (c_x - c_y)^2 + (d_x - d_y)^2}, \tag{6}$$

where P is the patch that contains pixel p and the patch Q contains pixel q of the same size $w * w$. x and y are the corresponding pixels in two patches. c_x, c_y, d_x, d_y are the color intensity and depth value for pixel x and y. We use this function to evaluate the similarity between two patches both in color and relative depth values and use this patch to estimate the invisible pixels.

In Maximal step, for every pixel q in the destination image, there are $w \times w$ patches containing this pixel. Thus, optimal destination image T should satisfy following constraint:

$$T = arg \min_{I} D(I, \overline{T}) + \lambda D(\nabla I_{depth}, \overline{\nabla T}_{depth}), \tag{7}$$

where $\overline{T}$ and $\overline{\nabla T}_{depth}$ are the same size as image I. Besides, the intensity of point(i,j) is:

$$\overline{T}(i,j) = \Sigma_{k,l=0\ldots w-1} \frac{NN(Q_{i-k}, j-l)(k,l)}{w^2} \tag{8}$$

$$\overline{T}_{depth}(i,j) = \Sigma_{k,l=0\ldots w-1} \frac{\nabla NN(Q_{i-k}, j-l)(k,l)}{w^2}, \tag{9}$$

where $NN(Q_{i,j})$ is the most similar patch found in Expectation step. $NN(Q_{i,j})(k,j)$ is the pixel at (k,l) of this patch. Therefore, $\overline{T}(i,j)$ calculates weighted sum of all the patches that contains pixel q. It can be shown that solving this equation is equivalent to solving Discrete Screened Poisson Equation. The pseudocode for one iteration is shown in Algorithm 2. We iterate through the two steps until convergence, and the result is shown in Fig. 5. Because the boundary is narrow in the image, our method normally converges in 10 to 15 iterations.

Algorithm 2. Single-Step for EM Algorithm

Input: the source image S and the target image T
Output: target image $\overline{T}$ after one iteration
Initialize $\overline{T} = 0$;
for each pixel $q = (i,j) \subset T$ **do**
  for $i^{'} = i - w + 1$ to i **do**
    for $j^{'} = j - w + 1$ to j **do**
      $P \longleftarrow$ GeneralizedPatchMatch(S, $Q_{i^{'},j^{'}}$);
      for each channel c **do**
        $\overline{T}(i,j,c) = \overline{T}(i,j,c) + \frac{P(i-i^{'}, j-j^{'}, c)}{w^2}$;
      end for
    end for
  end for
end for

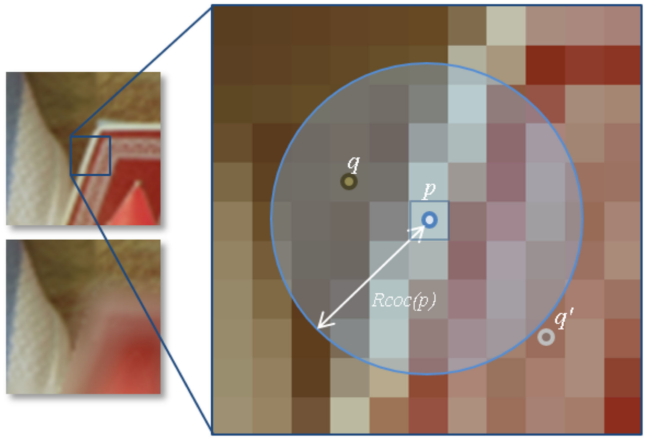

Fig. 6. Illustration of the gathering concept. For every pixel p, we gather all the colors from neighboring pixels in the CoC of p. q is in the CoC of p, therefore the intensity of q is added to p. q' is not in the CoC of p, therefore q' makes no contribution to p.

3.2.3 Weight Map Estimation

The reconstructed target image T contains our inference about the intensity and depth value of blocked pixels. As shown in Fig. 3, we estimate the blocked pixels behind yellow region. We can render the DOF result using T directly. But the pixel value calculated form solving Eq. (3) is not accurate and it is the maximal estimation that we can get. Thus, it need to be further fine-tuned to make it approach real values. After we obtain the maximal estimation of the hidden pixels, we propose a new multilayer-neighborhood optimization to smooth the pixel intensity using acquired maximal estimation as guidance which is similar to method used in [38]. First of all, we propose bilateral filter based method to iteratively update the estimated contributions of hidden pixels. Then we apply a scattering filter to get rid of the pixels in the foreground thus avoids color leakage problem. As shown in Fig. 6, for each pixel p, we gather all the colors scattered from neighboring pixels as in Eq. (10) using the following maximal estimation:

$$\begin{aligned} I_{max}(p) &= \frac{\sum_{q\in S_1} I(q)w(q) + \sum_{q\in S_2} I(q)w(q)}{\sum_{q\in S_1 \bigcup S_2} w(q)} \\ &= (1 - W_{max}(p))\overline{f}(p) + W_{max}(p)\overline{f}_{blocked}(p), \end{aligned} \tag{10}$$

where, we divide maximal intensity obtained for pixel p into two parts: those influenced by existing pixels and those influenced by estimated pixels. $S_1(p)$ as set of points away from the boundary whose CoC contains pixel p. $S_2(p)$ as set of points that within the boundary and whose CoC contains pixel p. $W_{max}(p)$ is the maximal contribution of blocked pixel p. $\overline{f}(p)$ denotes the average color intensity of exist pixels and $\overline{f}_{blocked}$ denotes average color intensity of blocked pixels Since the actual contributions of blocked pixels are unknown. But with the estimated blocked pixel intensity above, we can start with the maximal contributions of occluded pixels and then apply edge-preserving multilayer-neighborhood optimization on it to give a reasonable guess of its contributions. Thus, the maximal contribution of blocked pixels can be given in following form:

$$W_{max}(p) = \frac{card(S_2)}{card(S_1) + card(S_2)}, \tag{11}$$

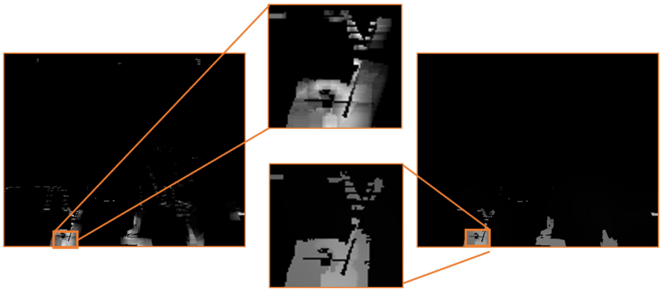

Fig. 7. The left picture is the result of maximal estimated weight while the right picture is the result of applying multilayer-neighborhood optimization for five times. The filtered weight lies between maximal weight and minimal weight.

where $card(S)$ calculates number of elements in the set S. Here, $card(S_1)$ and $card(S_2)$ compute the number of original pixels and number of blocked pixels near the boundary, respectively. We calculate the ratio between the number of pixels within the boundary and number of pixels whose CoC contains pixel p as the maximal weight of blocked pixel.

But the real contribution of p should be not greater than I_{max}. Therefore, an estimate of real contribution is needed. We apply multilayer-neighborhood optimization to obtain the final blur pixel $W'(p)$ as following formula:

$$W'(p) = \frac{\sum_{q\in S_1 \bigcup S_2} \mathbf{F}(p,q)\mathbf{G}(W_{max}(p), W_{max}(q))W_{max}(q)}{\sum_{q\in S_1 \bigcup S_2} \mathbf{F}(p,q)\mathbf{G}(W_{max}(p), W_{max}(q))}, \quad (12)$$

where $\mathbf{F}$ and $\mathbf{G}$ are Gaussian. $\mathbf{F}$ denotes the distance between p and q while $\mathbf{G}$ denotes the intensity distance. $S_1 \bigcup S_2$ is the set of pixels where CoC contains pixel p including estimated blocked pixels. $W'(p)$ is the estimated weight of blocked pixels using $W_{max}(p)$ as guidance. From this equation, we start from the estimated maximal weight map W_{max} and propagate the value from W_{max} to potential weight map W'. Besides, it is an edge-aware filtering function which will preserve the boundary of the blurred image. Thus, it can handle color leakage. Finally, we recursively apply Eq. (12) on the output map to get final estimation result. Normally speaking, we stopped the algorithm at fifth iterations. The results are shown in Fig. 7. The right column is the enlarged filtered weight map. From this picture, we can see that our proposed method gives an estimation between maximal weight and minimal weight while preserving edge during iteration.

After applying same procedure for five times, we can see that the estimated weights are smooth in the center region while preserving edges. Then we simply calculate weighted sum of the blocked pixels intensity as $I_{estimated}(p)$. As shown in Fig. 6, for each pixel p, we gather all the colors scattered from neighboring pixels. If p is in the CoC of its neighboring pixel q, i.e., the distance $d(p,q)$ between p and q is smaller than R_{CoC}, we add a portion of q's intensity to p. Otherwise we do nothing. The intensity of the color scattered from a pixel is inversely proportional to its CoC area $S_{CoC(q)}$ in accordance with the intensity conservation condition. Thus, we use this weight to estimate the final pixel intensity using following equation.

$$I'(p) = \frac{\sum_{q\in S(p)} I(q)\cdot w(q) + W'(p)\cdot I_{estimated}(p)}{\sum_{q\in S(p)} w(q) + W'(p)} \quad (13)$$

$$w(q) = (1 - W'(p))/S_{CoC(q)}, \quad (14)$$

where $S(p) = q|d(p,q) < R_{CoC}(q)$ denotes the set of pixels whose CoC contains p. $I(q)$ is the intensity of pixel q while $w(q)$ is the weight of pixel q. For pixels away from the boundary, we assume uniform distribution of intensity which means that every pixel shares the same weight which is calculated as Eq. (14). We iterate throughout each pixel using this equation to render the final depth of field effect.

4 Experimental Results

We conducted experiments on an Intel Core 4.00GHz CPU, nVidia Geforce GT 970 platform. Because our proposed approach is depth-variant, it might be very slow. Fortunately, PathMatch method and bilateral filtering have fast parallel implementation [38], [39]. Therefore, it could be accelerated via CUDA using parallel computing power of GPUs.

4.1 Results and Discussions

We tested our method using various data. Test image *Squares* is a computer-generated image with a manually-assigned depth map. Test images *Moebius* and *Plant* are from Middlebury stereo database [40] with depth map acquired via stereo vision. The original depth maps are incomplete and have small holes with no depth information. Thus, we used inpainting [41] to obtain complete depth maps. Test images *StillLife* are from Heidelberg datasets [42], whereas tested images *Statue* and *Mansion* are from [43]. We set $\lambda = 0.2$ in Eq. (3) for all our experimental results following the setting in [34] for hole filling. Our experiments show that it is a proper choice for adjusting the weight between color difference and depth difference in patches. Our approach uses five parameters in order to present a satisfying result: α, the blurriness, controls the degree of DOF effects. d_f, the focused depth, can change the focus point of the rendered image, σ_s and σ_w, controls the degree of estimated weight of block pixels, w, patch size parameter controls the degree of blocked pixels. In practice, α and d_f are determined by user-based on various requirements. There are plenty of methods to obtain one-to-one corresponding depth map such as stereo camera and image painting algorithms. If there're lots of holes in depth map acquired from original device, it may impose inaccuracy and noises in our experiment results. On the contrary, we demonstrate that it can achieve promising results using depth map that is inpainted with image inpainting algorithm. We set $\sigma_s = 0.03$, $\sigma_w = 0.08$ and $w = 3$ for our experiments. Fig. 8 compares our results with those obtained by state-of-the-art DOF rendering methods: the mipmap interpolation method proposed in [14], the gathering-based method proposed in [12] and the Adobe Photoshop CC lens blur filter [37]. (The methods proposed in [14] and [12] were implemented by us and may therefore differ from their original results.)

For the image *Squares* (first row of Fig. 8), the focused depth is at the green square in the middle; the red square is behind focus range, and the yellow square is before the focus range. The artifacts primarily occur in the edges between regions of different depths. The gathering-based method suffers from color leakage in edges between the red

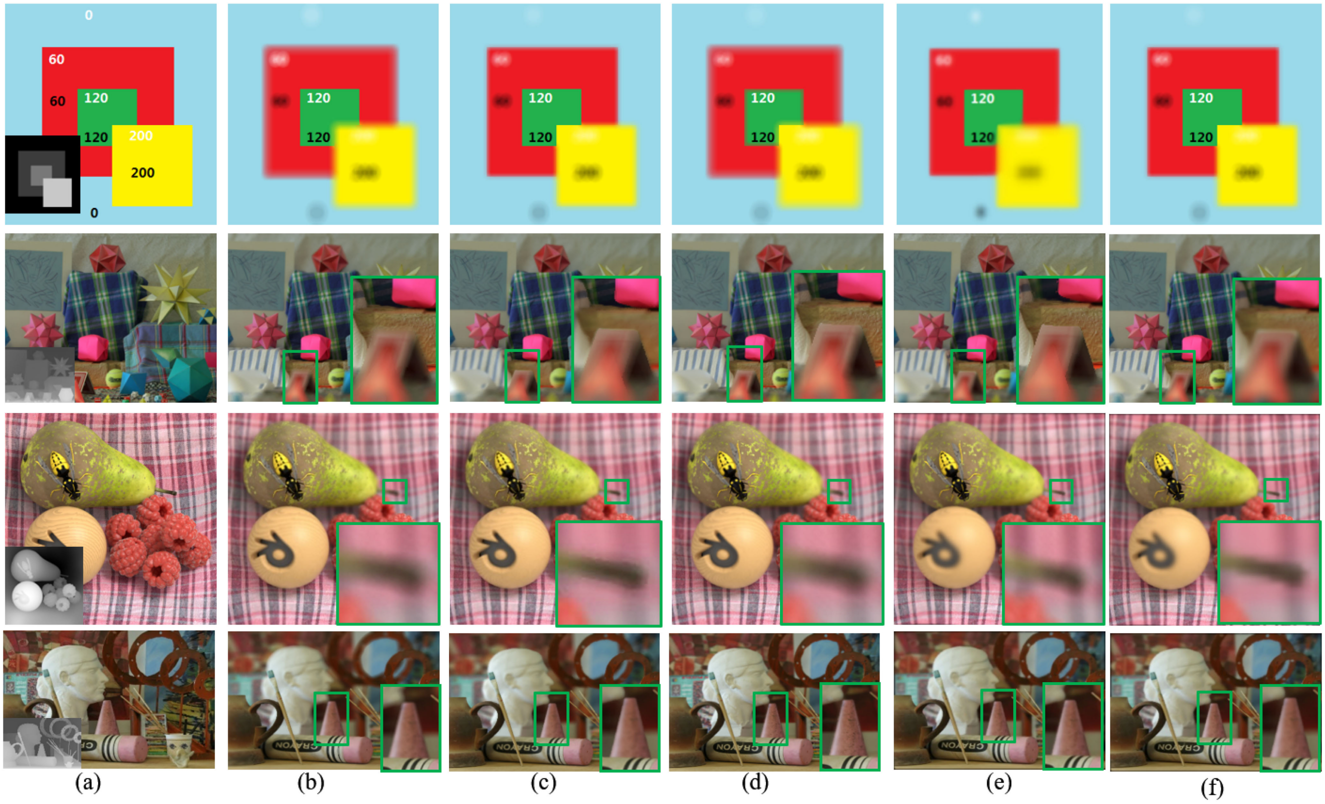

Fig. 8. Comparison of the experimental results for various methods. (a) Original image and corresponding depth map, (b) our results, (c) Photoshop lens blur filter [37], (d) gathering based method [12], (e) mipmap interpolation method [14], and (f) pyramidal method from [20]. From top to bottom, the images are: *Squares*, *Moebius*, *StillLife*, and *Statue*.

and green squares. Photoshop lens blur and the gathering-based method suffer from blurring discontinuity, which is obvious at the corner of the blurred yellow square on top of the green square. Our method produces a semitransparent soft edge for the blurred foreground object, which is more realistic. In the details of image *Moebius* (second row of Fig. 8), it is clear that our method produces a better result than other methods. Because our method uses a uniformly-distributed PSF that is closer to that of real cameras than Gaussian distribution used in mipmap interpolation method, the blurred region is closer to an out-of-focus blur, than to an artificial Gaussian blur. For image *StillLife* (third row of Fig. 8), the blur of the pear's stalk is not smooth in columns (c) and (e). Further, for image *Statue* (last row of Fig. 8), it can be seen that color leakage occurs in column (d).

As shown in Fig. 9, the image on the left shows the color leakage artifact which is a common artifact in scatter method [10], whereas the image on the right shows our result without causing color leakage. Fig. 10 shows the blurring in the foreground region. The image on the left has an opaque blurring foreground region, and the partial occlusion has not been dealt with due to lack of occluded pixel information. In contrast, our result on the left, has a semitransparent edge in the blurring foreground region because we carefully handle the contributions of occluded pixel for partial occlusion. Thus, our proposed method could handle intensity leakage and blur discontinuity problem. Besides, the focused region and the degree of blur can be easily controlled by parameters used in our method as shown in Fig. 11. Fig. 12 shows the structural similarity (SSIM) index [44] values and peak signal-to-noise ratio (PSNR) of our results and the results of other methods using the same parameters against the ground truth results. The data shows the similarity of DOF rendering results and the ground truth. The ground truth image was obtained by DOF synthesizing a densely sampled 4D light field [42].

Fig. 9. Color leakage problems. The result of [10] on the left suffers from color leakage in theadjacent area of focused region. In our result, on the right, the problem has been rectified using scattering-based filter.

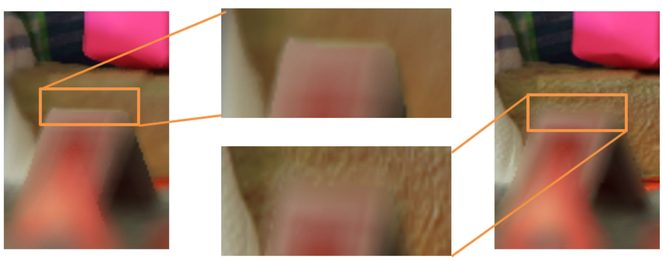

Fig. 10. The foreground blurring artifacts. The Photoshop lens blur result on the left suffers from blurring discontinuity and lacks partial occlusion. In our result on the right, the problem has been rectified by gathering filter in foreground region and an in-focus filter in the in-focus region.

Fig. 11. Experimental results for image *Statue*, *Moebius* and *Plant* with various blurriness α. (a) Original all in-focus image and depth map, (b) $\alpha = 5$, (c) $\alpha = 10$, and (d) $\alpha = 20$.

As shown in Fig. 12, our method outperforms other methods, and is much closer to ground truth.

Our proposed DOF rendering can be used in realistic image rendering or be used as a hint to transfer the focus of viewers. Besides, it can further be used in image editing and information visualization which makes important data clear and blur the unimportant data. Kosara et al. [45] proposed Semantic Depth of Field for focus-and-context display of information. Besides, DOF rendering can be used in practical medical area such as eye surgery. Fig. 13 shows the different rendered result using inpainted depth map and original imperfect depth map. As we can see in the red rectangle region, the quality of image has great improvement

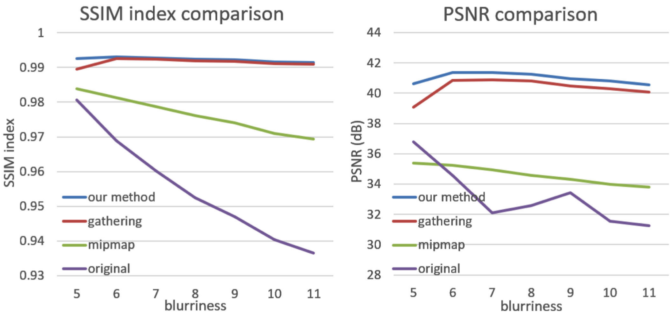

Fig. 12. Comparison of SSIM index and PSNR for various methods with different blurriness parameters. The results of our proposed method, gathering based method [12], mipmap interpolation method [14] and the original unprocessed image are compared with the ground truth result from light field DOF synthesis.

using inpainted depth map. Thus, we can use original depth map with holes to generate a relatively good depth of field rendering result using simple inpainted method. In Fig. 14, since iPhone X provides pixel aligned depth map which can be easily used in our proposed method, we can see that our method can be incorporated in real life application.

4.2 Advantages over Single-Image Approaches

Our method uses a depth map and take the blocked pixels into account, with the information obtained therefrom resulting in an advantage for our method over other single-image approaches. First, our approach is more accurate. Traditional methods ignore the influence of blocked pixels, therefore not realistic. Second, our method can handle complex scene, whereas other single-image approaches cannot segment scenes properly. Single-image approaches approximate the scene and depth information and can only produce simple and inaccurate depth map. As a result, when the image represents a complex scene, visual artifacts will arise. Third, our method can properly render the DOF effects of the image of different situations. We can focus at any depth, background or foreground. We can also adjust the degree of blurriness. Other single-image approaches do not consider the different cases and thus can only be applied over a limited scope. Compared with multiple image approaches, especially light filed approaches [25], which can resolve visibility problem in most cases due to more information on the scene, our method relies on the accuracy of estimation of intensity. We plan to combine the advantages of multiple

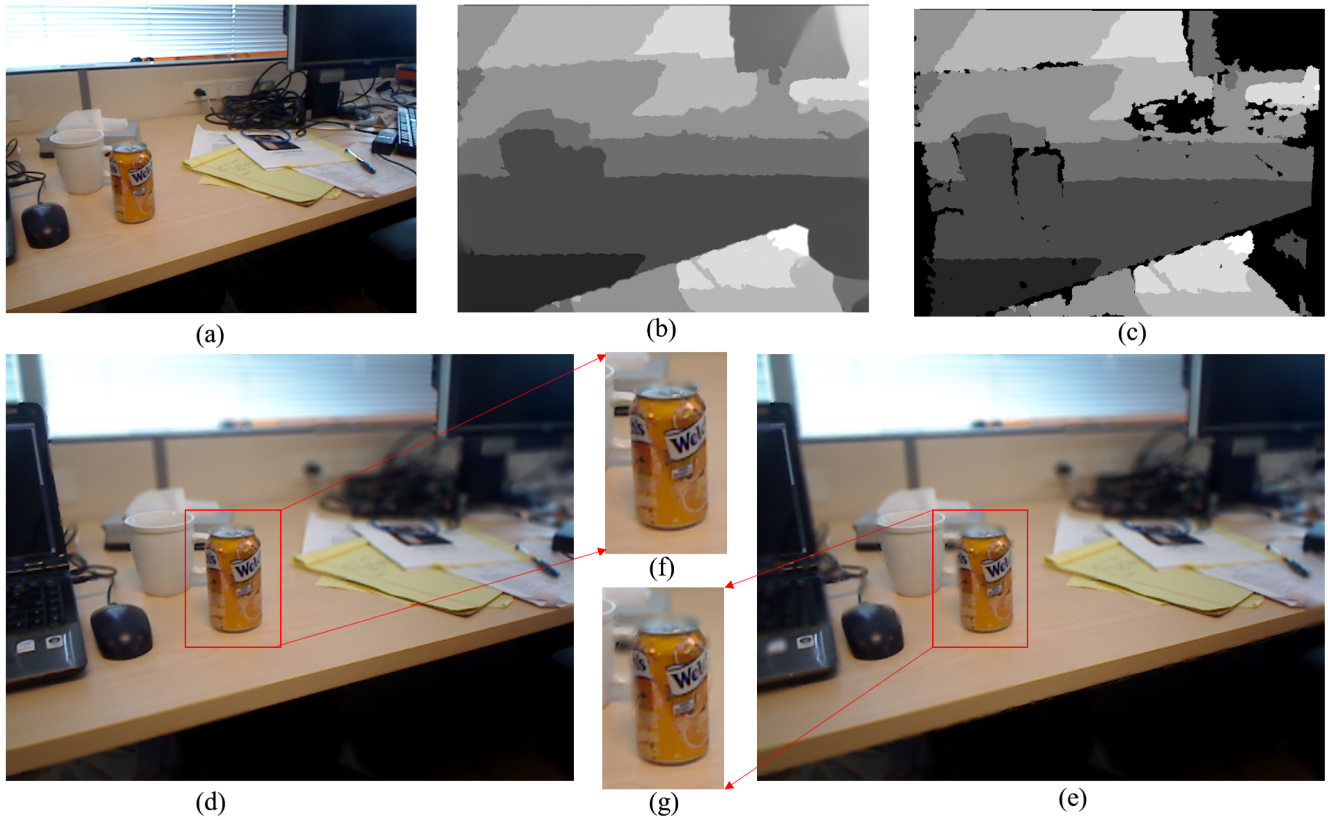

Fig. 13. Difference between rendered results using inpainted depth map and imperfect depth map. (a) Input image, (b) inpainted depth map, (c) depth map with holes, (d) rendered result using (b), (e) rendered result using (c), (f) amplified red region on (d), and (g) amplified red region on (e).

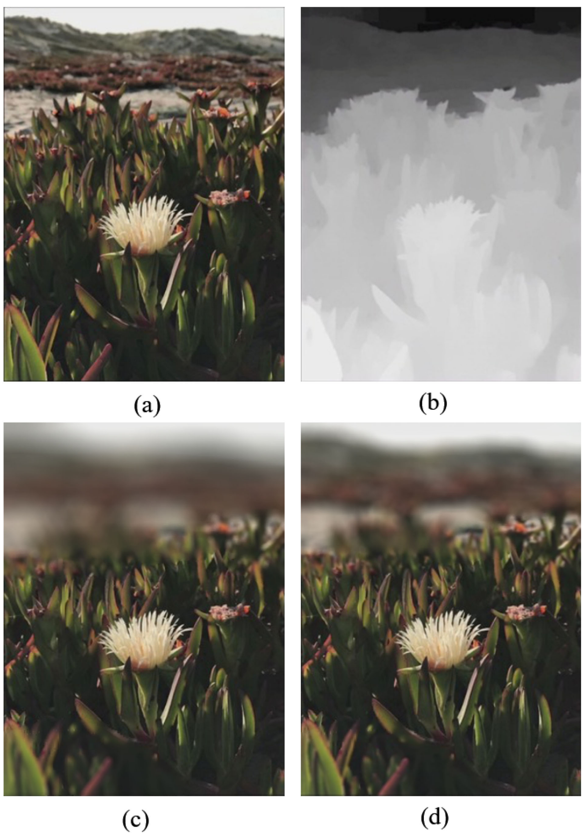

Fig. 14. A simple rendered depth of field result in iPhone X using default depth camera and our proposed method. (a) Original focused image, (b) depth map captured by iPhone X, (c) $\alpha = 5$, and (d) $\alpha = 10$.

images and proposed framework in order to better solve occlusion problem and reduce visual artifacts.

4.3 Limitation

Although our method can be applied to videos. But, video processing is still the limitation of our method. We tried to apply our framework on video in limited case. Fig. 15 shows three frames resulting from DOF rendering of a video sequence by our method. The dataset of the video sequence with depth maps is from [43]. But it requires the scene to move slowly so that the focus range will not change too much. When we apply our method to other videos, we need to adjust the focus range of each frame according to the depth of the objects, which may result in flickering in the video sequence. Therefore, we plan to improve our method in the future by incorporating coherence of objects in videos so that it can further be applied in other video sequences.

4.4 Discussions

SLIC segmentation can be used to divide each image into superpixels by considering the similarity of color intensities between pixels which is a good choice for segmenting depth image into different layers. PatchMatch algorithm is used widely in finding similar patches. Due to its potential use in image completion, we tried applying PatchMatch to fill in the holes which are further used for initial estimation of occluded pixels' influence. EM algorithm is great algorithm for optimizing function iteratively. Due to the efficiency of PatchMatch algorithm, it made it possible to solve this energy function quickly. We tested the performance of our

Fig. 15. Our DOF rendering approach applied to video of *Mansion*. Top row: Frames from original video and corresponding depth maps; Bottom row: Frames from video rendered by our method.

TABLE 1
Performance of our Approach for Various Images and Resolutions, Focus Depth d_f, and Blurriness α

image	resolution	focus depth d_f	blurriness α	CPU time (s)	GPU time (s)	intensity estimation time (s)	weight calculation time (s)
Moebius	463 × 370	120	10	0.467	0.0202	0.219	0.248
Room	640 × 480	120	10	0.923	0.0376	0.692	0.231
Plant	1000 × 800	120	10	2.152	0.1232	1.597	0.5550
Moebius	463 × 370	80	10	0.3668	0.0173	0.1876	0.1792
Moebius	463 × 370	120	10	0.475	0.0241	0.2046	0.2704
Moebius	463 × 370	180	10	0.531	0.0267	0.245	0.286
Moebius	463 × 370	120	5	0.3173	0.0164	0.1433	0.1740
Moebius	463 × 370	120	10	0.3456	0.0174	0.2046	1.410
Moebius	463 × 370	120	20	0.4623	0.0241	0.3219	0.1404

method with various images, resolutions, and parameters. We performed each test ten times and obtained an average time. The running time is related to the image's depth map complexity, resolution, and focus depth. As can be seen in Table 1, the first section compares our method on images with different resolution where focus depth and blurriness are same. The resolution of the image has a major effect on the execution time. Besides, weight calculation time increases not rapidly as intensity estimation time as the resolution increase. Because we apply a fast multilayer-neighborhood optimization method on it. For the second section, we test our method for same image with different focus depth. It can be seen that the parameters of the focus depth do not significantly affect the performance. Finally, we exam our method on the same image with different blurriness value. The experiment has shown that the greater the blurriness, the greater is the computation time required.

The effects of our method rely on five parameters for control. Besides, the layer boundary generated by SLIC method might influence the result because we only consider the occluded pixels near such boundary. Thus, it burdens user to adjust the result. In our experiments, we fix three parameters used for estimation of blocked pixels and provide two parameters for user to adjust. It is a normal scenario when user needs to adjust focus depth and degree of blur for various scenes. First of all, we use GPU to parallel the computing of PatchMatch algorithm. The bottleneck of accelerating PatchMatch is the scan line process of propagation stage. We can use Jump Flood Propagation method to replace original scan line propagation method to make it easier for parallelism. Since the computation in the final scattering filter is independent among every pixel. We can use GPU to accelerate these two stages. Here, we define nearest-neighbor field (NNF) as shown in [33] as a function $f : A \mapsto \mathcal{R}^2$ which means that for each patch a in image A, $f(a)$ denotes the offset between patch a and its nearest neighbor patch b in spatial coordinates. Patch matching not only estimates color intensity but also depth of pixels. If we use simple linear interpolation method, we can only estimate the intensity of hidden pixels. But, with PatchMatch, we can also estimate depth. Besides, simple diffusion interpolation method does not consider the texture of surface and will result in flat interpolation surface. While PatchMatch can fill the missing intensities of holes more consistent than diffusion interpolation. More implementation details can be found in the supplementary material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TVCG.2019.2894627.

As shown in Fig. 16, our method fails in image shown in the last row. A visible artifact is rendered near the ear of toy due to similar intensity between background and in-focus region. A higher weight is estimated for pixels near the boundary which causes this artifact. In the first row, our

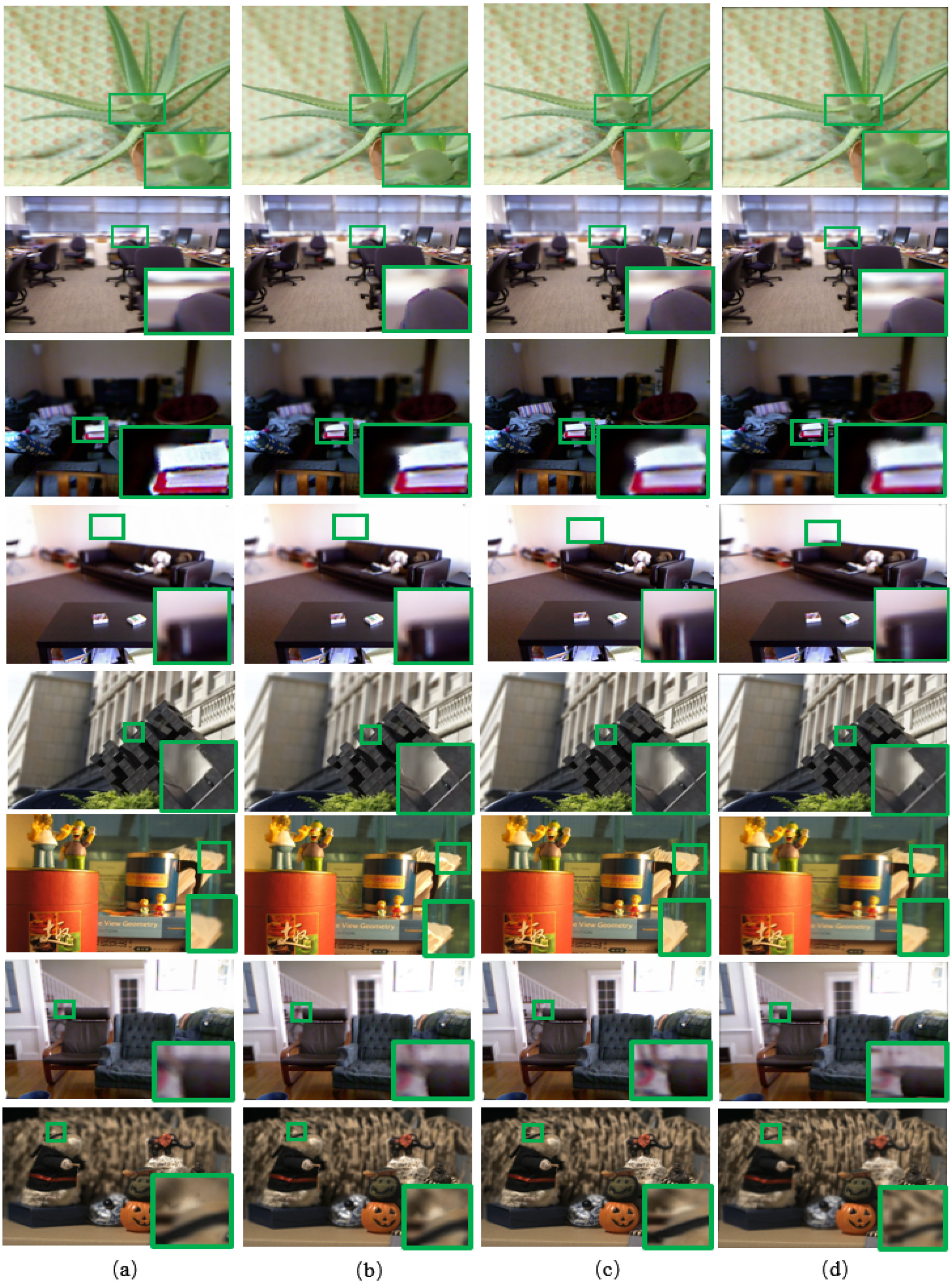

Fig. 16. Comparison of different methods. (a) Our method, (b) pyramid method from [20], (c) light field method [25], and (d) mipmap interpolation method [14]. The final row shows a bad example using our method.

method achieve continuous boundary near the leaf while there are jagged edges in other methods. In the second row, you can see that the black pixels spread to background regions leading color leakage problems while our method keeps the intensity values and presents a clear edge near the boundary. In the third row, our method can also achieve visual-pleasing result in the edge of the book while other methods fail to preserve the structure. Our method achieves visual pleasing result with better image quality measured with SSIM and PSNR. It might be solved by taking the intensity difference between different layers into account which is our future direction.

5 Conclusion and Future work

In this paper, we proposed a new post-processing depth of field rendering approach. Unlike previous DOF rendering methods, which do not consider the influence of blocked pixels directly, our work proposes a new patch-based approach to estimate the depth and intensity of blocked pixels near the boundary. After that, we obtain the maximal weight of blocked pixels. We further propose a multilayer-neighborhood optimization, which takes the weight map as input and recursively get an estimation of potential contributions of blocked pixels. With the estimated weight map, we calculate our blurring results which resolve both color leakage and blur discontinuity problems as shown in various experiments. In the future, we will further speed up DOF rendering. Besides, we will also extend our estimation methods for better estimating of the blocked pixels. Finally, we will improve our method by incorporating temporal relation into our framework and avoid adjusting focus range for each frame for better results on various video processing, e.g., avoiding flickering problem of video example in Fig. 15.

Acknowledgments

We thank the reviewers for their insightful comments and suggestions. This work was supported in part by the National Natural Science Foundation of China (61872241, 61572316), the National Key Research and Development Program of China (2017YFE0104000, 2016YFC1300302), the Macau Science and Technology Development Fund (0027/2018/A1), the Science and Technology Commission of Shanghai Municipality (18410750700, 17411952600, 16DZ0501100), and the Ministry of Science and Technology (107-2221-E-006-196-MY3), Taiwan.

References

[1] D. Iwai, S. Mihara, and K. Sato, "Extended depth-of-field projector by fast focal sweep projection," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 4, pp. 462–470, Apr. 2015.

[2] Y. Itoh, T. Amano, D. Iwai, and G. Klinker, "Gaussian light field: Estimation of viewpoint-dependent blur for optical see-through head-mounted displays," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 11, pp. 2368–2376, Nov. 2016.

[3] H. Qin, M. Chai, Q. Hou, Z. Ren, and K. Zhou, "Cone tracing for furry object rendering," *IEEE Trans. Vis. Comput. Graph.*, vol. 20, no. 8, pp. 1178–1188, Aug. 2014.

[4] D. C. Schedl, C. Birklbauer, J. Gschnaller, and O. Bimber, "Generalized depth-of-field light-field rendering," in *Proc. Int. Conf. Comput. Vis. Graph.*, 2016, pp. 95–105.

[5] R. L. Cook, T. Porter, and L. Carpenter, "Distributed ray tracing," *ACM SIGGRAPH Comput. Graph.*, vol. 18, no. 3, pp. 137–145, 1984.

[6] P. Haeberli and K. Akeley, "The accumulation buffer: Hardware support for high-quality rendering," *ACM SIGGRAPH Comput. Graph.*, vol. 24, no. 4, pp. 309–318, 1990.

[7] A. Veeraraghavan, R. Raskar, A. Agrawal, A. Mohan, and J. Tumblin, "Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing," *ACM Trans. Graph.*, vol. 26, no. 3, pp. 69:1–69:12, 2007.

[8] S. Kuthirummal, H. Nagahara, C. Zhou, and S. K. Nayar, "Flexible depth of field photography," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 58–71, Jan. 2011.

[9] A. Levin, S. W. Hasinoff, P. Green, F. Durand, and W. T. Freeman, "4D frequency analysis of computational cameras for depth of field extension," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–14, 2009.

[10] M. Potmesil and I. Chakravarty, "A lens and aperture camera model for synthetic image generation," *ACM SIGGRAPH Comput. Graph.*, vol. 15, no. 3, pp. 297–305, 1981.

[11] M. Shinya, "Post-filtering for depth of field simulation with ray distribution buffer," in *Proc. Graph. Interface*, 1994, pp. 59–66.

[12] C.-Y. Yan, M.-C. Tien, and J.-L. Wu, "Interactive background blurring," in *Proc. 17th ACM Int. Conf. Multimedia*, 2009, pp. 817–820.

[13] P. Rokita, "Generating depth-of-field effects in virtual reality applications," *IEEE Comput. Graph. Appl.*, vol. 16, no. 2, pp. 18–21, Mar. 1996.

[14] S. Lee, G. J. Kim, and S. Choi, "Real-time depth-of-field rendering using anisotropically filtered mipmap interpolation," *IEEE Trans. Vis. Comput. Graph.*, vol. 15, no. 3, pp. 453–464, Jun. 2009.

[15] T. Zhou, J. X. Chen, and M. Pullen, "Accurate depth of field simulation in real time," *Comput. Graph. Forum*, vol. 26, no. 1, pp. 15–23, 2007.

[16] F. L. Zhang, J. Wang, E. Shechtman, Z. Y. Zhou, J. X. Shi, and S. M. Hu, "PlenoPatch: Patch-based plenoptic image manipulation," *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 5, pp. 1561–1573, May 2017.

[17] Z. Zhu, H. Z. Huang, Z. P. Tan, K. Xu, and S. M. Hu, "Faithful completion of images of scenic landmarks using internet images," *IEEE Trans. Vis. Comput. Graph.*, vol. 22, no. 8, pp. 1945–1958, Aug. 2016.

[18] T. Rhee, L. Petikam, B. Allen, and A. Chalmers, "MR360: Mixed reality rendering for 360° panoramic videos," *IEEE Trans. Vis. Comput. Graph.*, vol. 23, no. 4, pp. 1379–1388, Apr. 2017.

[19] B. A. Barsky, M. J. Tobias, D. P. Chu, and D. R. Horn, "Elimination of artifacts due to occlusion and discretization problems in image space blurring techniques," *Graphical Models*, vol. 67, no. 6, pp. 584–599, Nov. 2005.

[20] M. Kraus and M. Strengert, "Depth-of-field rendering by pyramidal image processing," *Comput. Graph. Forum*, vol. 26, no. 3, pp. 645–654, 2007.

[21] S. Lee, E. Eisemann, and H.-P. Seidel, "Depth-of-field rendering with multiview synthesis," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 134:1–134:6, 2009.

[22] C. Lu, Y. Xiao, and C.-K. Tang, "Real-time video stylization using object flows," *IEEE Trans. Vis. Comput. Graph.*, vol. 24, no. 6, pp. 2051–2063, Jun. 2018.

[23] M. McGuire, E. Enderton, P. Shirley, and D. Luebke, "Real-time stochastic rasterization on conventional GPU architectures," in *Proc. Conf. High Perform. Graph.*, 2010, pp. 173–182.

[24] S. Lee, E. Eisemann, and H.-P. Seidel, "Real-time lens blur effects and focus control," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 65:1–65:7, 2010.

[25] X. Yu, R. Wang, and J. Yu, "Real-time depth of field rendering via dynamic light field generation and filtering," *Comput. Graph. Forum*, vol. 29, no. 7, pp. 2099–2107, 2010.

[26] B. A. Barsky and T. J. Kosloff, "Algorithms for rendering depth of field effects in computer graphics," in *Proc. WSEAS Int. Conf. Comput.*, 2008, pp. 999–1010.

[27] S. Lee, G. J. Kim, and S. Choi, "Real-time depth-of-field rendering using point splatting on per-pixel layers," *Comput. Graph. Forum*, vol. 27, no. 7, pp. 1955–1962, 2008.

[28] W. Xue, X. Zhang, B. Sheng, and L. Ma, "Image-based depth-of-field rendering with non-local means filtering," in *Proc. IEEE Int. Conf. Multimedia Expo Workshops*, 2013, pp. 1–6.

[29] T. J. Kosloff and B. A. Barsky, "An algorithm for rendering generalized depth of field effects based on simulated heat diffusion," in *Proc. Int. Conf. Comput. Sci. Appl. - Vol. Part III*, 2007, pp. 1124–1140.

[30] Z. Xiao, H. Chen, C. Tu, and R. Klette, "An effective graph and depth layer based RGB-D image foreground object extraction method," *Comput. Visual Media*, vol. 3, no. 4, pp. 387–393, 2017.

[31] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2274–2282, Nov. 2012.

[32] B. A. Barsky, D. R. Horn, S. A. Klein, J. A. Pang, and M. Yu, "Camera models and optical systems used in computer graphics: Part I, object-based techniques," in *Proc. Int. Conf. Comput. Sci. Appl.: PartIII*, 2003, pp. 246–255.

[33] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "PatchMatch: A randomized correspondence algorithm for structural image editing," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 24:1–24:11, 2009.

[34] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, "Image melding: Combining inconsistent images using patch-based synthesis," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 82:1–82:10, 2012.
[35] C. Barnes and F.-L. Zhang, "A survey of the state-of-the-art in patch-based synthesis," *Comput. Visual Media*, vol. 3, no. 1, pp. 3–20, 2017.
[36] R. Giraud, V.-T. Ta, A. Bugeau, P. Coupé, and N. Papadakis, "SuperPatchMatch: An algorithm for robust correspondences using superpixel patches," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 4068–4078, Aug. 2017.
[37] Adobe, "Adobe Photoshop CC," 2017. [Online]. Available: http://www.adobe.com/products/photoshop.html
[38] Q. Yang, S. Wang, and N. Ahuja, "Real-time specular highlight removal using bilateral filtering," in *Proc. 11th Eur. Conf. Comput. Vis.: Part IV*, 2010, pp. 87–100.
[39] P. Yu, X. Yang, and L. Chen, "Parallel-friendly patch match based on jump flooding," in *Proc. Adv. Digit. Television Wireless Multimedia Commun.*, 2012, pp. 15–21.
[40] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Proc. IEEE Comput. Society Conf. Comput. Vis. Pattern Recognit.*, 2003, pp. 195–202.
[41] A. Telea, "An image inpainting technique based on the fast marching method," *J. Graph. Tools*, vol. 9, no. 1, pp. 23–34, 2004.
[42] S. Wanner, S. Meister, and B. Goldluecke, "Datasets and benchmarks for densely sampled 4D light fields," in *Proc. Int. Workshop Vis. Modeling Vis.*, 2013, pp. 1–8.
[43] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, and M. Gross, "Scene reconstruction from high spatio-angular resolution light fields," *ACM Trans. Graph.*, vol. 32, no. 4, pp. 73:1–73:12, 2013.
[44] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.
[45] R. Kosara, S. Miksch, and H. Hauser, "Semantic depth of field," in *Proc. IEEE Symp. Inf. Vis.*, 2001, pp. 97–104.

Benxuan Zhang received the BEng degree in computer science from Shanghai Jiao Tong University, Shanghai, China. He is currently working toward the MEng degree in computer science in the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include image/video processing, depth of field rendering, deep learning, and computer graphics.

Bin Sheng received the PhD degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, China. He is currently an associate professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China. His current research interests include image-based rendering, machine learning, virtual reality, and computer graphics.

Ping Li received the PhD degree from The Chinese University of Hong Kong, Hong Kong, China. He is currently an assistant professor with the Macau University of Science and Technology, Macau, China. His current research interests include image/video stylization, big data visualization, GPU acceleration, and creative media. He has one image/video processing national invention patent, and has excellent research project reported worldwide by ACM TechNews.

Tong-Yee Lee received the PhD degree in computer engineering from Washington State University, Pullman, in May 1995. He is currently a chair professor with the Department of Computer Science and Information Engineering, National Cheng-Kung University, Tainan, Taiwan, ROC. He leads the Computer Graphics Group, Visual System Laboratory, National Cheng-Kung University (http://graphics.csie.ncku.edu.tw). His current research interests include computer graphics, non-photorealistic rendering, medical visualization, virtual reality, and media resizing. He is a senior member of the IEEE and a member of the ACM.