Journal of Electronic Imaging

SPIEDigitalLibrary.org/jei

Stereo vision–based depth of field rendering on a mobile device

Qiaosong Wang
Zhan Yu
Christopher Rasmussen
Jingyi Yu

Stereo vision–based depth of field rendering on a mobile device

Qiaosong Wang,* Zhan Yu, Christopher Rasmussen, and Jingyi Yu
University of Delaware, Newark, Delaware 19716

Abstract. The depth of field (DoF) effect is a useful tool in photography and cinematography because of its aesthetic value. However, capturing and displaying dynamic DoF effect were until recently a quality unique to expensive and bulky movie cameras. A computational approach to generate realistic DoF effects for mobile devices such as tablets is proposed. We first calibrate the rear-facing stereo cameras and rectify the stereo image pairs through FCam API, then generate a low-res disparity map using graph cuts stereo matching and subsequently upsample it via joint bilateral upsampling. Next, we generate a synthetic light field by warping the raw color image to nearby viewpoints, according to the corresponding values in the upsampled high-resolution disparity map. Finally, we render dynamic DoF effect on the tablet screen with light field rendering. The user can easily capture and generate desired DoF effects with arbitrary aperture sizes or focal depths using the tablet only, with no additional hardware or software required. The system has been examined in a variety of environments with satisfactory results, according to the subjective evaluation tests. *© 2014 SPIE and IS&T* [DOI: 10.1117/1.JEI.23.2.023009]

Keywords: depth of field; programmable cameras; joint bilateral upsampling; light field.

Paper 13493SSP received Sep. 4, 2013; revised manuscript received Jan. 31, 2014; accepted for publication Feb. 26, 2014; published online Mar. 19, 2014.

1 Introduction

Dynamic depth of field (DoF) effect is a useful tool in photography and cinematography because of its aesthetic value. Capturing and displaying dynamic DoF effect were until recently a quality unique to expensive and bulky movie cameras. Problems such as radial distortion may also arise if the lens system is not setup properly.

Recent advances in computational photography enable the user to refocus an image at any desired depth after it has been taken. The hand-held plenoptic camera[1] places a microlens array behind the main lens, so that each microlens image captures the scene from a slightly different viewpoint. By fusing these images together, one can generate photographs focusing at different depths. However, due to the spatial-angular tradeoff[2] of the light field camera, the resolution of the final rendered image is greatly reduced. To overcome this problem, Georgiev and Lumsdaine[3] introduced the focused plenoptic camera and significantly increased spatial resolution near the main lens focal plane. However, angular resolution is reduced and may introduce aliasing effects to the rendered image.

Despite recent advances in computational light field imaging, the costs of plenoptic cameras are still high due to the complicated lens structures. Also, this complicated structure makes it difficult and expensive to integrate light field cameras into small hand-held devices like smartphones or tablets. Moreover, the huge amount of data generated by the plenoptic camera prohibits it from performing light field rendering on video streams.

To address this problem, we develop a light field rendering algorithm on mobile platforms. Because our algorithm works on regular stereo camera systems, it can be directly applied to existing consumer products such as three-dimensional (3-D)-enabled mobile phones, tablets, and portable game consoles. We also consider the current status of mobile computing devices in our software system design and make it less platform dependent by using common libraries such as OpenCV, OpenGL ES, and FCam API. We start by using two cameras provided by the NVIDIA Tegra 3 prototype tablet to capture stereo image pairs. We subsequently recover the high-resolution disparity maps of the scene through graph cuts (GCs)[4] and then generate a synthesized light field for dynamic DoF effect rendering. Once the disparity map is generated, we synthesize a virtual light field by warping the raw color image to nearby viewpoints. Finally, dynamic DoF effects are obtained via light field rendering. The overall pipeline of our system is shown in Fig. 1. We implement our algorithm on the NVIDIA Tegra 3 prototype tablet under the FCam architecture.[5] Experiments show that our system can successfully handle both indoor and outdoor scenes with various depth ranges.

2 Related Work

Light field imaging opens up many new possibilities for computational photography, because it captures full four-dimensional radiance information about the scene. The captured light information can later be used for applications like dynamic DoF rendering and 3-D reconstruction. Since conventional imaging systems are only two-dimensional (2-D), a variety of methods have been developed for capturing and storing light fields in a 2-D form. Lippmann[6] was the first to propose a prototype camera to capture light fields. The Stanford multicamera array[7] is composed of 128 synchronized CMOS firewire cameras and streams, capturing data to four PC hosts for processing. Because of the excessive data

*Address all correspondence to: Qiaosong Wang, E-mail: qiaosong@udel.edu

0091-3286/2014/$25.00 © 2014 SPIE and IS&T

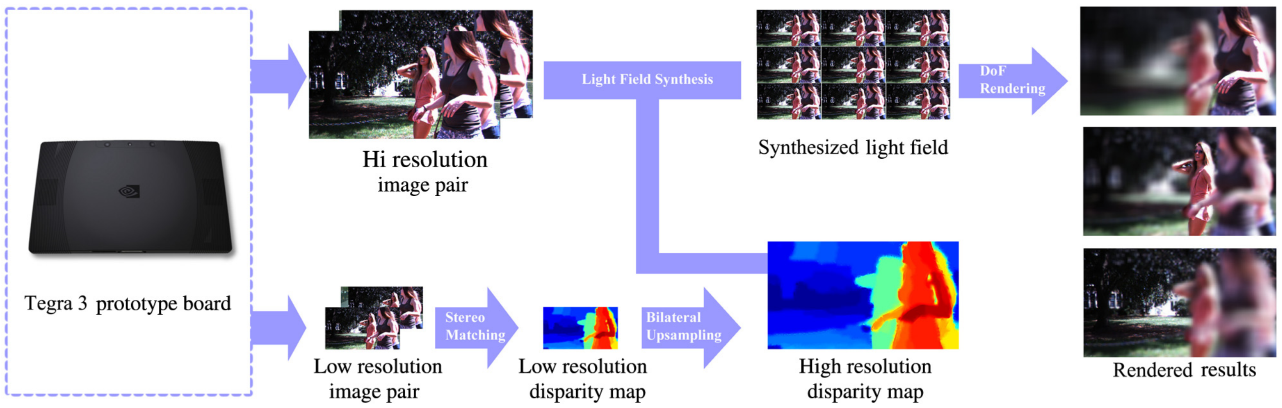

Fig. 1 The NVIDIA Tegra 3 prototype tablet and the processing pipeline of our software system. All modules are implemented on the Android 4.1 operating system.

volume, DoF effects are rendered offline. The Massachusetts Institute of Technology light field camera array[8] uses 64 usb webcams and is capable of performing real-time rendering of DoF effects. However, these camera systems are bulky and hard to build. Recently, Ng et al.[1] have introduced a new camera design by placing a microlens array in front of the sensor with distance equals microlens focal length, wherein each microlens captures a perspective view in the scene from a slightly different position. However, the spatial resolution near the microlens array plane is close to the number of microlenses. To overcome this limitation, Georgiev and Lumsdaine[3] introduced the focused plenoptic camera which trades angular resolution for spatial resolution. An alternative approach is to integrate light-modulating masks to conventional cameras and multiplex the radiance in the frequency domain.[9] This design enables the camera sensor to capture both spatial and angular frequency components, but reduces light efficiency.

As the rapid research and development provide great opportunities, hand-held plenoptic camera has been proven practical and quickly progressed into markets. The Lytro camera[10,11] is the first implementation of a consumer-level plenoptic camera. Recently, Pelican Imaging[12] announced a 16-lens mobile plenoptic camera system and scheduled to implement it to new smartphones in 2014.

Our work is inspired by the algorithms proposed by Yu et al.[13,14] However, the system proposed in these two papers is bulky and expensive, and the algorithm is highly dependent on the GPU performance, making it hard to transfer the proposed method to small hand-held devices such as cellphones and compact size cameras. The system used by Yu et al.[13] is composed of a desktop workstation and a customized stereo camera system. The desktop is equipped with a 3.2 GHz Intel Core i7 970 6-core CPU and a NVIDIA Geforce GTX 480 Graphic Card with 1.5 GB memory. Actually, very few laptops on the market can reach the same level of performance, let alone tablets or cellphones. Also, this system connects to two Point Grey Flea 2 cameras via a Firewire link. The retail price for two Flea cameras is around $1500, and the camera itself requires external power source and professional software for functionalities such as auto exposure, white balancing, and stereo synchronization, which is almost impractical for general users without a computer vision background. In addition, most scenes in this article are indoor scenes with controlled lighting, and the user is required to tune different parameters on a GUI in order to obtain a good-looking disparity map in different scenes. In contrast, our software system works directly on an off-the-shelf tablet, which costs less than $400. Since our algorithm is implemented under the Android operating system using highly optimized CPU-only functions from OpenCV4Android SDK, it can be easily ported to other hand-held Android devices with limited computational power. Besides, we conducted extensive experiments to obtain parameters that generate optimal results. Therefore, it is easy to install and use our software, no hardware setup or parameter adjustment is required. Furthermore, our system uses GCs[15] instead of belief propagation (BP)[16] for stereo matching and is tested working under complex illumination conditions. According to the tests carried out by Tappen and Freeman,[17] GCs generate smoother solutions compared with BP and consistently perform better than BP in all quality metrics for the Middlebury[18] Tsukuba benchmark image pair. To conclude, we made the following contributions:

- We propose light field rendering as a possible solution to generate dynamic DoF effects. We also discussed why our method is good at reducing boundary discontinuity and intensity leakage artifacts compared with depth-based image blurring schemes.
- We implemented the entire system on an off-the-shelf Android tablet using highly optimized CPU-only functions from OpenCV4Android SDK. The system can be easily ported to other mobile photography devices with limited computational power.
- We conducted extensive experiments to obtain the optimal combination of methods and parameters under the Tegra 3 T30 prototype device. As a result, there is no need for parameter adjustment and it is easy for the user to install and use our application.
- We experimented with GCs for disparity map calculation, and the system is capable of working with a variety of scene structures and illumination conditions.

3 Overview

In this article, we demonstrate that the DoF effects can be rendered using low-cost stereo vision sensors on mobile devices. We first capture stereo image pairs by using the FCam API and then apply the GCs stereo-matching algorithm to obtain low-resolution disparity maps. Next, we take raw color images as guide images and upsample the low-resolution disparity maps via joint bilateral upsampling. Once the high-resolution disparity maps are generated, we can synthesize light fields by warping the raw color images from the original viewing position to nearby viewpoints. We then render dynamic DoF effects by using the synthetic light fields and visualize the results on the tablet screen. We evaluate a variety of real-time stereo-matching and edge-preserving upsampling algorithms for the tablet platform. Experimental results show that our approach provides a good tradeoff between expected depth-recovering quality and running time. All aforementioned processing algorithms are implemented to the Android operating system and tested on the Tegra 3 T30 prototype tablet. The user can easily install the software and capture and generate desired DoF effects using the tablet only, with no additional hardware or software required. The system has been tested in a variety of environments with satisfactory results.

4 Programmable Stereo Camera

4.1 *Development Environment*

The Tegra 3 T30 prototype tablet is equipped with a 1.5 GHz quad-core ARM Cortex-A9 CPU and a 520 MHz GPU. It has three sensors. The rear main sensor and secondary sensor are identical with a 6-cm baseline. The third sensor is on the same side of the multitouch screen facing the user. The raw image resolution is 640×360 (16:9).

Our software is running under Android 4.1 (Jelly Bean) operating system. We use the Tegra Android Developer Pack (TADP) for building and debugging the application. This software toolkit integrates Android SDK features to Eclipse IDE by using the Android Development Tools (ADT) Plugin. The ADT extends the capabilities of Eclipse and enables the user to design graphic UI, debug the application using SDK tools, and deploy APK files to physical or virtual devices. Since typical Android applications are written in Java and compiled for the Dalvik Virtual Machine, there is another toolset called Android Native Development Kit (NDT) for the user to implement part of the application in native code languages such as C and C++. However, using the NDT brings certain drawbacks. First, the developer has to use the NDT to compile native code, which hardly integrates with the Java code, so the complexity of the application is increased. Besides, using native code on Android system generally does not result in a noticeable improvement in performance. For our application, since we need to use the FCam API for capturing stereo pairs and OpenCV and OpenGL ES for image processing and visualization, we implemented most of the code in C++ and run the code inside the Android application by using the Java Native Interface (JNI). The JNI is a vendor–neutral interface that permits the Java code to interact with the underlying native code or load dynamic-shared libraries. By using the TADP, our workflow is greatly simplified. We first send commands to the camera using the FCam API, then convert raw stereo image pairs to *cv::Mat* format, and use OpenCV for rectification, stereo matching, joint bilateral upsampling, and DoF rendering. The final results are visualized on the screen using OpenGL ES.

4.2 *FCam API*

Many computational photography applications follow the general pattern of capturing multiple images with changing parameters and combining them into a new picture. However, implementing these algorithms on a consumer-level tablet has been hampered by a number of factors. One fundamental impediment is the lack of open software architecture for controlling the camera parameters. The Frankencamera[5] proposed by Adams et al. is the first architecture to address this problem. Two implementations of this concept are a custom-built F2 camera and a Nokia N900 smartphone running on a modified software stack to include the FCam API. Troccoli et al. extended the implementation of FCam API to support multiple cameras[19] and enabled the NVIDIA Tegra 3 prototype tablet to trigger stereo captures.

4.3 *Calibration, Synchronization, and Autofocus*

Since the two sensors are not perfectly aligned, we calibrated the stereo pair using a planar checker board pattern outlined by Zhang.[20] We computed the calibration parameters and saved them to the tablet hard drive as a configuration file. Once the user starts the application, it automatically loads the calibration parameters to memory for real-time rectification. This reduces distortion caused by the optical lens and improves stereo-matching results. Even though we obtained rectified image pairs, we still need to synchronize the sensors since we cannot rectify over time for dynamic scenes. The main mechanism for synchronizing multiple sensors in FCam API is by extending the basic *sensor* component to a *sensor array*.[19] A new abstract class called *SynchronizationObject* is also derived from the *Device* class with members *release* and *wait* for software synchronization. When the request queue for the camera sensors is being processed, if a *wait* is found and a certain condition is not satisfied, the *sensor* will halt until this condition is satisfied. On the other hand, if a *release* is found and the condition is satisfied, the halted *sensor* will be allowed to proceed. The FCam API also provides classes such as *Fence*, *MultiSensor*, *MultiShot*, *MultiImage*, and *MultiFrame* for the user to control the stereo sensor with desired request parameters.

In our application, we use the rear main camera to continuously detect the best focusing position and to send updates to the other sensor. First, we ask the rear main lens to start sweeping the lens. We then get each frame with its focusing location. Next, we sum up all the values of the sharpness map attached to the frame and send updates to the autofocus function. The autofocus routine will move the lens in a relatively slower speed to refine the best focal depth. Once this process is done, we trigger a stereo capture with identical aperture, exposure, and gain parameters for both sensors. The overall image quality is satisfactory, considering the fact that the size of the sensor is very small and the cost is much lower than research stereo camera systems such as Pointgreys Bumblebee. Figure 2 shows how our software system interacts with the imaging hardware.

5 Disparity Map Generation

Computing depth information from stereo camera systems is one of the core problems in computer vision. Stereo algorithms based on local correspondences[21,22] are usually fast but introduces inaccurate boundaries or even bleeding artifacts. Global stereo estimation methods, such as GCs[15]

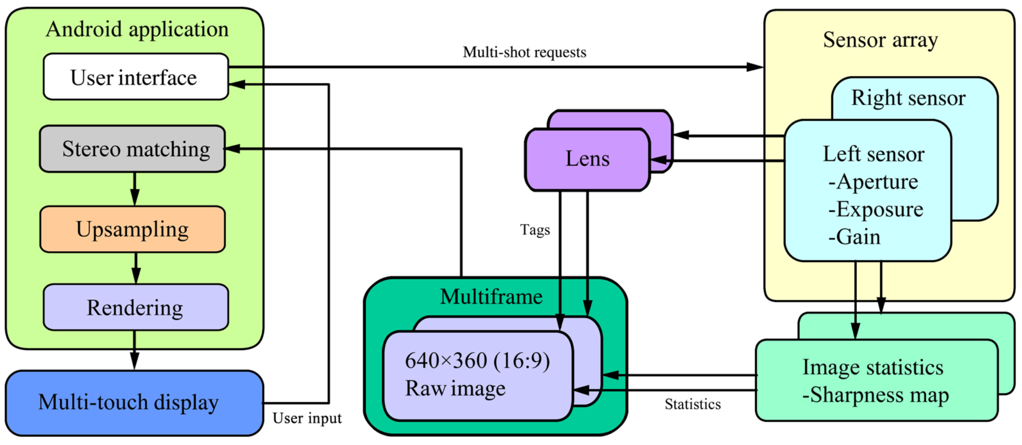

Fig. 2 This diagram shows our system architecture. Our application accepts user input from the multi-touch screen, sends multishot requests to the sensors with desired parameters, and then transfers the raw stereo image pairs to the stereo-matching module. We then upsample the low-resolution disparity map and synthesize a light field image array. Finally, we render DoF effects on the screen of the tablet. We compute the best focal plane by using image statistics information tagged with the raw image frame.

and BP,[16] have shown good results on complex scenes with occlusions, textureless regions, and large depth changes.[18] However, running these algorithms on full-resolution (1 MP) image pairs is still expensive and hence impractical for mobile devices. Therefore, we first downsample the raw input image pair and recover a low-resolution disparity map via GCs. Next, we take each raw color image as the guidance image and upsample the disparity map via joint bilateral upsampling.[23]

5.1 *GCs Stereo Matching*

In order to efficiently generate a high-resolution disparity map with detailed information about the scene, we propose a two-step approach. We first recover a low-resolution disparity map on downsampled image pairs with the size of 160×90. Given the low-resolution image pairs, the goal is to find labeling of pixels indicating their disparities. Suppose $f(p)$ is the label of pixel p; $D_p(x)$ is the data term, reflecting how well pixel p fits its counterpart pixel p shifted by x in the other image; $V_{p,q}(y,z)$ is the smoothness term indicating the penalty of assigning disparity y to pixel p and disparity z to pixel q; and N is the set of neighboring pixels, the correspondence problem can be formulated as minimizing the following energy function:

$$E(f) = \arg\min_f \left\{ \sum_{p \in P} D_p[f(p)] + \sum_{\{p,q\} \in N} V_{p,q}[f(p), f(q)] \right\}. \tag{1}$$

The local minimization of Eq. (1) can be efficiently approximated using the alpha expansion presented in Ref. 15. In our implementation, we set the number of disparities to be 16 and run the algorithm for five iterations. If the algorithm cannot find a valid alpha expansion that decreases the overall energy function value, then it may also terminate in less than five iterations. The performance of GCs on the Tegra 3 tablet platform can be found in Table 1.

To evaluate our scheme, we performed experiments on various stereo image datasets. The stereo-matching methods used here are block matching (BM), semi-global BM (SGBM),[21] efficient large-scale stereo (ELAS),[22] and GCs.[15] Table 1 shows the running time of these algorithms on the Tegra 3 tablet, and Fig. 3 shows the calculated disparity map results. According to our experiments, BM is faster than SGBM and ELAS on any given dataset but requires an adequate choice of the window size. Smaller window sizes may lead to a larger bad pixel percentage, whereas larger window sizes may cause inaccuracy problems on the boundary. Besides, the overall accuracy of disparity values generated by BM is not very high. As can be seen from Fig. 3, we can still identify the curved surface area of the cones from the results generated by SGBM and ELAS, but the same area looks almost flat in BM. SGBM and ELAS are the two very popular stereo-matching algorithms with near real-time performance. According to our experiments on the tablet, they are very similar to each other in terms of running time and accuracy. From Table 1 and Fig. 3, we can see that ELAS generates better boundaries than SGBM on the cones dataset, but takes more processing time and produces more border bleeding artifacts. The GCs gives smooth transitions between regions of different disparity values. According to Table 2, the GCs algorithm outperforms other algorithms in most of the quality metrics on the Middlebury datasets. For our application, since the quality of upsampled result is highly dependent on the precision of boundary values in low-resolution disparity maps, we choose to use GCs rather

Table 1 Comparing running time (ms) of different stereo-matching methods on the Tegra 3 tablet, using the Middlebury Cones dataset. The longer edge is set to 160 pixels, and the number of disparities is set to 16.

Datasets	BM	SGBM	ELAS	GCs
Tsukuba	15	28	51	189
Venus	13	30	97	234
Cones	19	42	124	321

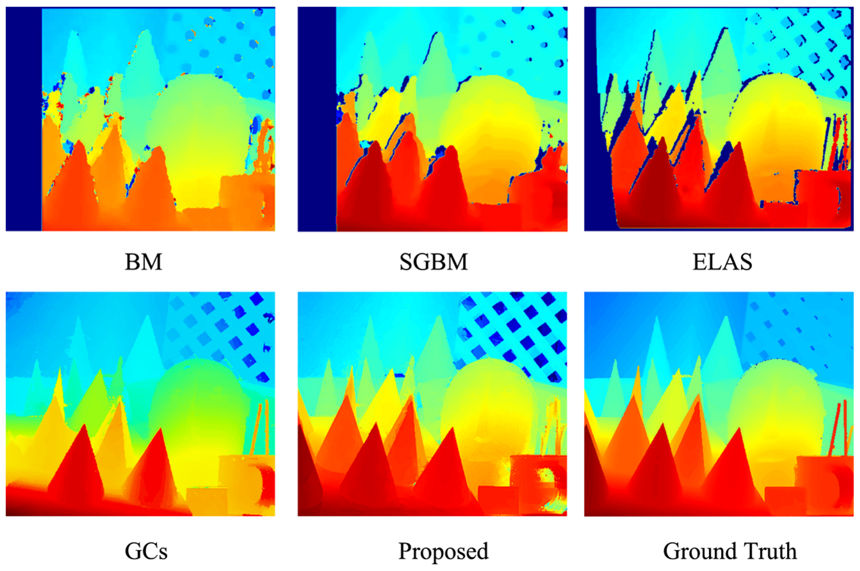

Fig. 3 Comparison of our approach and other popular stereo-matching algorithms.

than other methods which runs faster. Another reason is that we are running the GCs algorithm on low-resolution imagery. According to Table 1, the running time is around 250 ms, which is still acceptable compared with ELAS (around 100 ms). In return, noisy and invalid object boundaries are well optimized and the resulting disparity map is ideal for refinement filters such as joint bilateral upsampling.

5.2 *Joint Bilateral Upsampling*

Because the stereo-matching process is performed on low-resolution stereo image pairs, the resulting disparity map cannot be directly used for DoF synthesis. We need to upsample the disparity map while keeping important edge information.

Bilateral filtering proposed by Tomasi and Manduchi[24] is a simple, noniterative scheme for edge preserving smoothing, which uses both a spatial kernel and a range kernel. However, for low signal-to-noise ratio images, this algorithm cannot keep the edge information very well. A variant called joint bilateral filter introduced by Kopf et al.[23] addresses this problem by adding the original RGB image as a guidance image. More formally, let p and q be two pixels on the full-resolution color image I; $p_\downarrow$ and $q_\downarrow$ denote the corresponding coordinates in the low-resolution disparity map D'; f is the spatial filter kernel, g is the range filter kernel, W is the spatial support of kernel f, and K_p is the normalizing factor. The upsampled solution D_p can be obtained as

$$D_p = \frac{1}{K_p} \sum_{q_\downarrow \in W} D'_{q\downarrow} f(\|p_\downarrow - q_\downarrow\|) g(\|I_p - I_q\|). \tag{2}$$

This method uses RGB values from the color image to create the range filter kernel and combines high-frequency components from the color image and low-frequency components from the disparity map. As a result, color edges are integrated with depth edges in the final upsampled disparity map. Since depth discontinuities typically correspond with color edges, this method can remove small noises. However, it may bring some unwanted effects. First, blurring

Table 2 Evaluation of different stereo-matching methods on the Middlebury stereo datasets cite in bad pixel percentage (%). The method shown in the last row applies five iterations of joint bilateral upsampling to the downsampled results (half of the original size) of GCs, using the full-resolution color image as the guidance image. The resolutions of the four datasets (Tsukuba, Venus, Teddy, and Cones) are 384×288, 434×383, 450×375, and 450×375, respectively. If not specified, raw image size of each individual dataset will be the same for the remainder of this article.

	Tsukuba			Venus			Teddy			Cones		
	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
BM	10.3	11.9	21.5	12.4	13.9	21.6	16.7	23.1	27.3	7.46	17.2	23.8
SGBM	3.26	3.96	12.8	1.00	1.57	11.3	6.02	12.2	16.3	3.06	9.75	8.90
ELAS	3.96	5.42	17.9	1.82	2.78	20.9	7.92	14.5	22.8	6.81	14.9	17.2
GCs	1.94	4.12	9.39	1.79	3.44	8.75	16.5	25.0	24.9	7.70	18.2	15.3
Proposed	1.01	2.83	5.42	0.18	0.59	1.99	6.57	11.2	15.1	3.06	9.70	8.92

Note: nonocc, bad pixel percentage in nonoccluded regions; all, bad pixel percentage in all regions; disc, bad pixel percentage in regions near-depth discontinuities.

and aliasing effects caused by the optical lens are transferred to the disparity map. Besides, the filtering process may change disparity values in occlusion boundaries, according to the high-frequency components in the color image, and thus causing the disparity map to be inaccurate. We address this problem by iteratively refining the disparity map after the upsampling process is done. As a result, the output image of the previous stage becomes the input of the next stage.

Figure 4 shows the results after different numbers of iterations. The initial disparity map [see Fig. 4(a)] is noisy and inaccurate, because it is generated on low-resolution image pairs. However, if too many iterations are applied to the input image [Fig. 4(d)], the boundaries of the cup handle start to bleed into the background, which is a result of over-smoothing. Also, more iterations add to the complexity and processing overhead of the entire application. According to Fig. 5, the quality of the disparity map can be improved during the first five or six iterations. This is because joint bilateral upsampling can preserve edges while removing noises in the disparity map. However, if the refining process contains too many iterations, then the disparities of one side of edges start to bleed into the other side, causing the bad pixel percentage to go up, especially in regions near depth discontinuities (refer to the increase of disc values in Fig. 5). Therefore, a compromise number of iterations must be chosen. In our application, the number is set to 5. Since the Middlebury datasets contain both simple scenes like Venus and complex scenes such as Teddy and Cones, we assume that five iterations should return good results under a variety of scene structures. Generally, it takes around 40 ms to finish the five iteration steps on the tablet. Figure 6 illustrates the detailed view of our result compared with other standard upsampling methods. Because DoF effects are most apparent around the depth edges, it is very important to recover detailed boundaries in the high-resolution disparity map. According to Table 3, our method outperforms other methods in all quality metrics and generates better boundary regions (refer to disc values in Table 3) by using the fine details from the high-resolution color image.

6 DoF Rendering

Once we obtained the high-resolution disparity map, we can proceed to synthesize dynamic DoF effects. Previous studies suggested that the real-time DoF effects can be obtained by applying a spatially varying blur on the color image and using the disparity value to determine the size of the blur kernel.[25,26] However, this method suffers from strong intensity leakage and boundary bleeding artifacts. Other methods such as distributed ray tracing[27] and accumulation buffer[28] give more accurate results. However, these methods are computationally expensive and therefore can only provide a limited frame rate.

6.1 *Synthesized Light Field Generation*

In this article, we use a similar approach to Ref. 29 by generating a synthetic light field on the fly. The main idea is to get the light field image array by warping the raw color image to nearby viewpoints, according to the corresponding values in the upsampled high-resolution disparity map. The light field array can then be used to represent rays in the scene. Each ray in the light field can be indexed by an integer 4-tuple (s,t,u,v), where (s,t) is the image index and (u,v) is the pixel index within an image. Next, we set the rear main camera as the reference camera and use the high-resolution color image and disparity map for reference view R_{00}. We then compute all rays passing through a spatial point X with shifted disparity γ from the reference view. Suppose X is projected to pixel (u_0,v_0) in the reference camera, we can compute its image pixel coordinate in any other light field camera view R_{st} as

$$(u,v) = (u_0,v_0) + (s,t)\cdot\gamma. \tag{3}$$

However, this algorithm may introduce holes in warped views, and this artifact becomes more severe when the synthesized baseline increases. To resolve this issue, we start from the boundary of the holes and iteratively take nearby pixels to fill the holes. Note that this module is only used for generating pleasing individual views for the user to interactively shift the perspective. In the final rendering process, missing rays are simply discarded and the filled pixels are not used. Figure 7 shows the warped views of an indoor scene using the aforementioned warping and hole-filling algorithms.

Since the image formed by a thin lens is proportional to the irradiance at pixel a,[30] if we use $L_{\mathrm{out}}(s,t,u,v)$ to represent the out-of-lens light field and $L_{\mathrm{in}}(s,t,u,v)$ to represent the in-lens light field, the pixels in this image can be obtained as a weighted integral of all incoming radiances through the lens

$$a(x,y) \simeq \sum_{(s,t)} L_{\mathrm{in}}(s,t,u,v)\cdot\cos^4\phi. \tag{4}$$

To compute the out-of-lens light field, we simply remap the pixel $a(x,y)$ to pixel $(u_0,v_0) = (w-x, h-y)$ in the reference view R_{00}. Therefore, we can focus at any scene

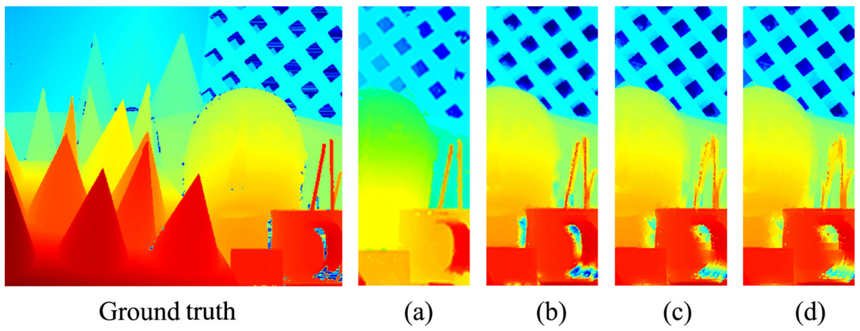

Fig. 4 Comparison of results using different numbers of iterations. Panels (a), (b), (c), (d) are obtained using 0, 5, 10, 20 iterations, respectively.

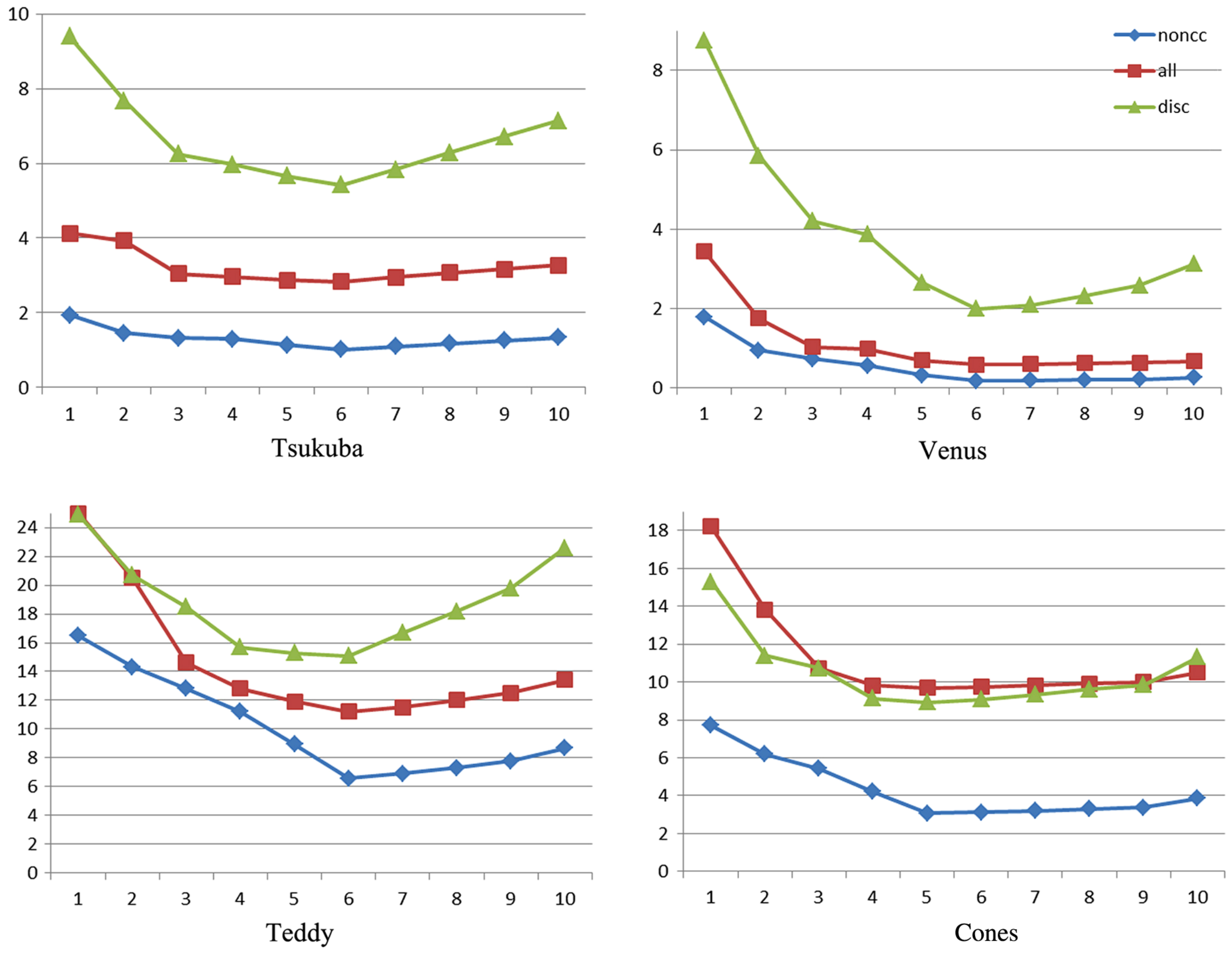

Fig. 5 Evaluation of the disparity maps using different numbers of joint bilateral upsampling iterations on the Middlebury stereo dataset. The horizontal axis shows the number of iterations and the vertical axis shows the bad pixel percentage.

depth with corresponding disparity γ_f by finding the pixel index in camera R_{st} using Eq. (3). Since the direction for each ray is $(s,t,1)$, we can approximate the attenuation term $\cos^4\phi$ as $\frac{1}{(s^2+t^2+1)^2}$, and the irradiance at a can be calculated as

$$a(x,y) \simeq \sum_{(s,t)} \frac{L_{\text{out}}(s,t,u_0+s\cdot\gamma_f,v_0+t\cdot\gamma_f)}{(s^2+t^2+1)^2}. \tag{5}$$

Figure 8 shows the details of the rendered image by using different sizes of the synthesized light field array. Since aliasing artifacts are related to scene depth and sampling frequency,[31] we can reduce aliasing in the rendered image by increasing the size of the synthesized light field array.

6.2 *Comparison of Our Method and Single-Image Blurring*

Reducing boundary artifacts is very important as DoF effects are apparent near the occlusion boundaries. Comparing with single-image blurring methods,[25,26] our light field–based analysis is good at reducing two types of boundary artifacts: the boundary discontinuity and intensity leakage artifacts. We summarize four types of boundary artifacts and analyze them separately. A detailed illustration of the four cases can

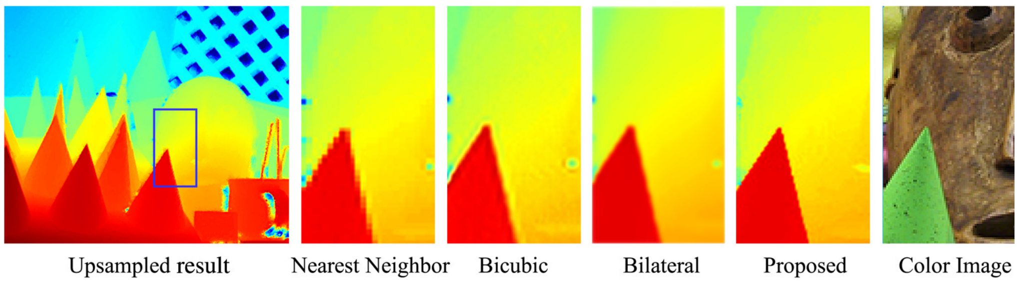

Fig. 6 Comparison of our approach and other upsampling algorithms on the Middlebury cones dataset.

Table 3 Evaluation of various upsampling methods on the Middlebury stereo datasets in bad pixel percentage (%). We run these methods on downsampled ground truth data (half of the original size), and then try to recover the disparity maps at original size and measure the error percentage.

	Tsukuba			Venus			Teddy			Cones		
	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
Nearest neighbor	5.55	6.65	18.3	0.47	1.02	6.56	8.65	9.77	28.2	7.98	9.62	23.7
Bicubic	4.97	5.69	18.7	0.67	0.93	9.32	4.89	5.61	17.8	6.81	7.59	20.6
Bilateral	4.59	5.04	10.8	0.41	0.60	5.75	4.52	5.12	16.3	6.85	8.41	20.5
Proposed	3.08	3.34	7.54	0.25	0.33	3.47	2.41	2.89	8.76	3.45	3.96	10.5

Note: nonocc, bad pixel percentage in nonoccluded regions; all, bad pixel percentage in all regions; disc, bad pixel percentage in regions near-depth discontinuities.

be found at Fig. 9. In practice, the four cases can occur at the same time within a single scene.

Our analysis is based on the real-world scene shown in Fig. 9. Consider a woman in a black dress walking in front of a white building. When we conduct the DoF analysis, the camera is either focused at the foreground (the woman) or at the background (the building). For Figs. 9(a) and 9(b), we assume that the camera to be focused at the background, and for Figs. 9(c) and 9(d), we assume that the camera is focused at the foreground. For each case, a comparison of results using different methods is shown at the right side of the images.

Now consider the first two cases shown in Figs. 9(a) and 9(b). Suppose P_b is a point on the background building and

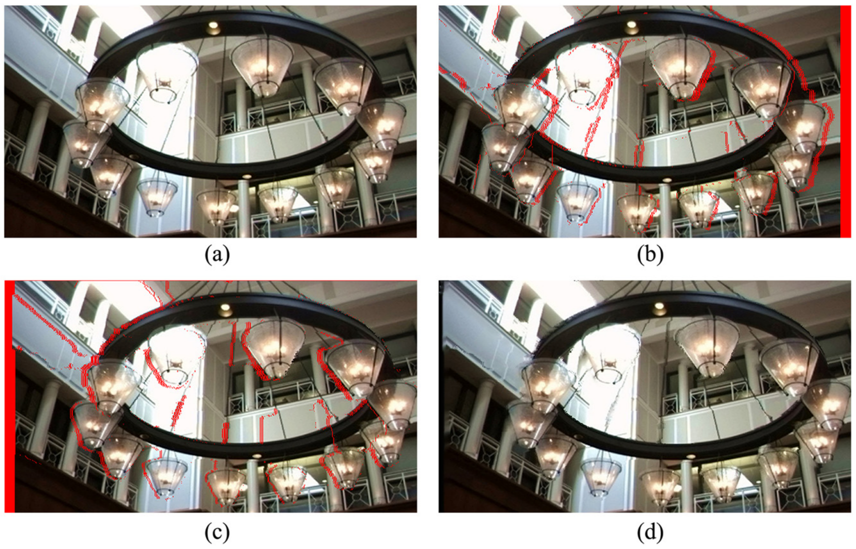

Fig. 7 Synthesized light field view, missing pixels are marked in red. (a) Input image, (b) warped left side view, (c) warped right side view, and (d) resulting image using our hole-filling algorithm, taking (c) as the input.

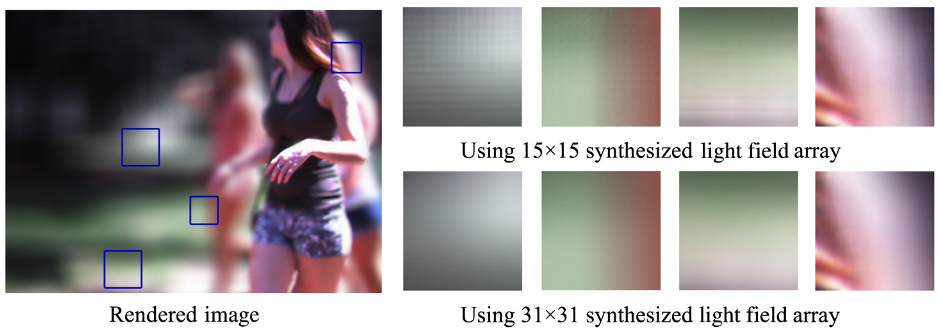
Rendered image
Using 15×15 synthesized light field array
Using 31×31 synthesized light field array

Fig. 8 Comparing rendering results with different sizes of the synthesized light field array.

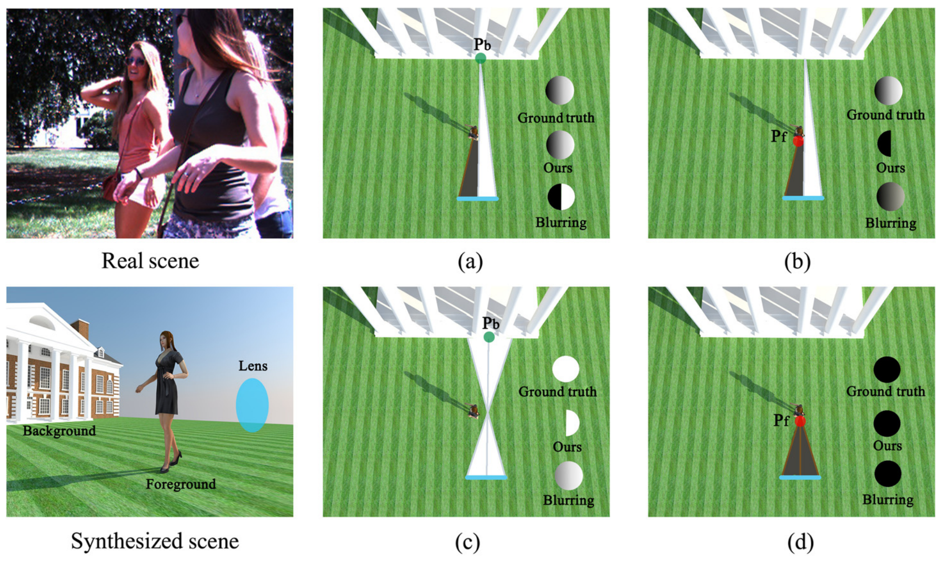

Fig. 9 Causes of different boundary artifacts (see Sec. 6.2 for details). In (a) and (b) the camera is focused at the background. In (c) and (d), the camera is focused at the foreground.

its image I_b in the camera is right next to the foreground as shown in Fig. 9(a). The ground truth result should blend both foreground and background points for calculating I_b to make the transition natural and smooth. However, single-image blurring methods would consider P_b in focus and directly use its color as the value of I_b. This will result in a boundary discontinuity artifact because of the abrupt jump between foreground and background pixel values. Our method, however, takes advantage of the synthesized light field, attempts to use rays originating from both foreground and background to calculate the pixel value of I_b, and hence generates correct results for this scenario. Similarly, for a foreground point P_f shown in Fig. 9(b), the ground truth result should blend its neighboring foreground pixels and a single in-focus background point. The single-image blurring methods will use a large kernel to blend a group of foreground and background pixels, producing the intensity leakage artifact. In contrast, our method only takes rays needed to get the value of P_f and is free of intensity leakage artifacts. However, due to occlusion, some background pixels may be missing. In this case, our method will blend foreground rays and accessible background rays together. Since the missing rays only occupy a small portion of all background rays, our method produces reasonable approximations.

For the other two cases [Figs. 9(c) and 9(d)], assume that the camera is focused at the foreground. As shown in Fig. 9(c), the ground truth result should only blend background pixels. However, because of the blur kernel, the single-image blurring method blends both foreground and background pixels and thus causing intensity leakage problems. Our method, on the other hand, only attempts to blend background rays. Similar to the previous case, some rays are occluded by the foreground. We simply discard these rays and by blending existing rays together, we are able to reach

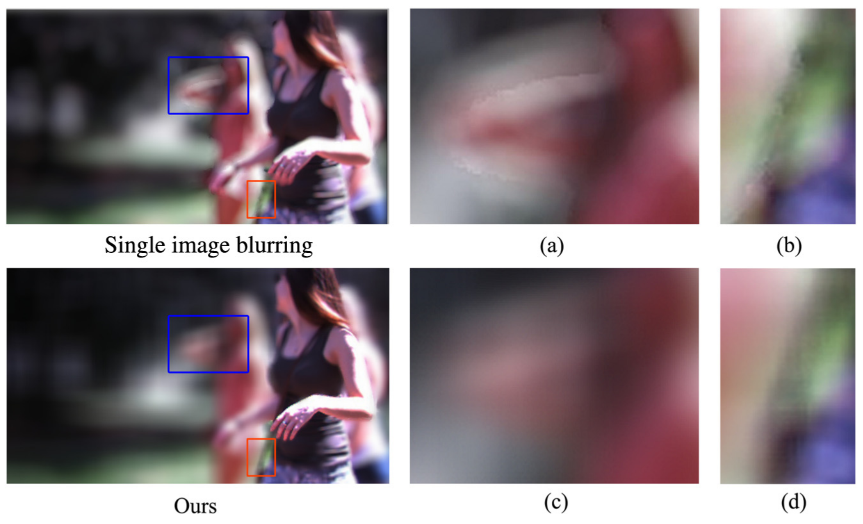

Fig. 10 Comparison between our method and single-image blurring. Single-image blurring methods suffer from intensity leakage (a) and boundary discontinuity (b) artifacts. Our method (c and d) reduces these artifacts.

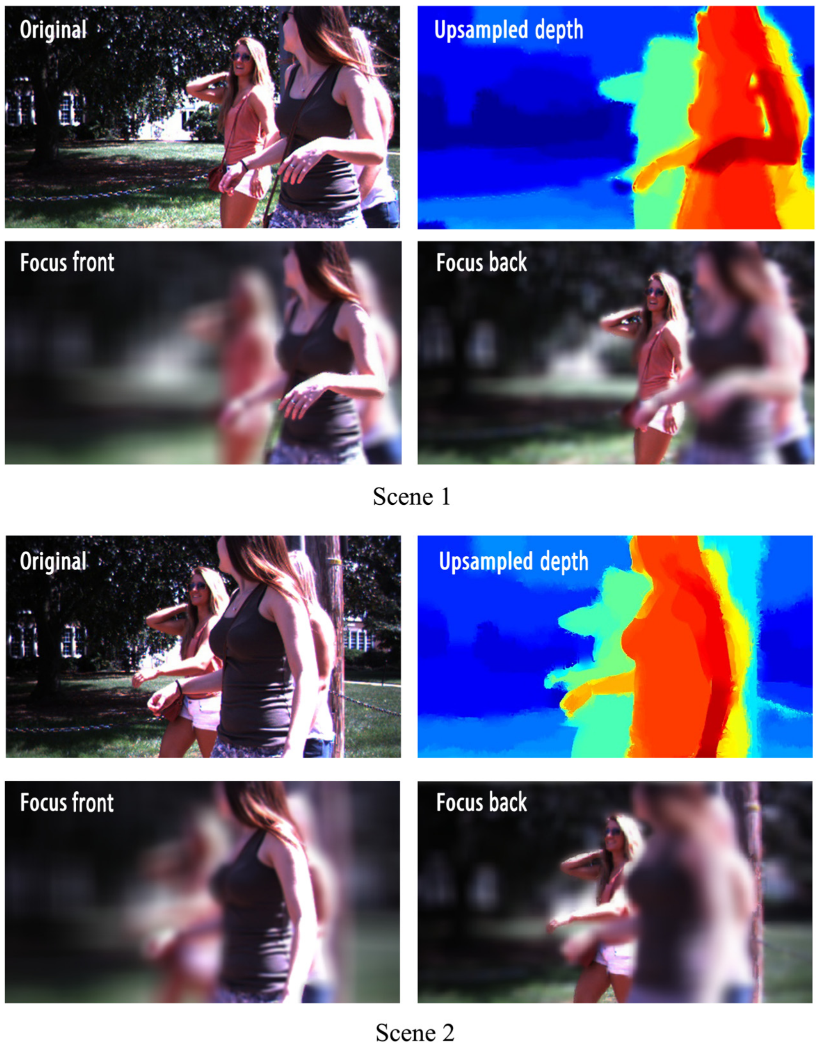

Fig. 11 Input disparity map and rendered images of our system on two frames from the same stereo video sequence.

reasonable approximations of the ground truth. For the last case, consider a point P_f on the foreground, as shown in Fig. 9(d). Since this pixel is considered to be in focus, the single-image blurring method will directly use its color and produces the correct result. Our method collects all rays coming from P_f, and these rays are all accessible. Therefore, our method is also able to get the correct result.

Figure 10 shows the results of our method and single-image blurring on an outdoor scene. As mentioned before, our method reduces artifacts on boundary regions compared with single-image blurring approaches. In fact, our method will not cause any intensity leakage problems. When examining the single-image blurring result [Fig. 10(a)], it is very easy to find intensity leakage artifacts along the boundary, whereas our technique prevents such leakage [Fig. 10(c)]. Also, our method provides smooth transitions from the handbag strips to the background [Fig. 10(d)], whereas single-image blurring method exhibits multiple discontinuous jumps in intensity values.

7 Results and Analysis

We conducted extensive experiments on both indoor and outdoor scenes. Figures 11 and 12 show the results generated by our system under different scene structures and illumination conditions. Scenes 1 and 2 demonstrate our system's ability of handling real-time dynamic scenes; Scene 3 shows the result on an outdoor scene with strong illumination and shadows; Scene 4 displays the result on an indoor scene with transparent and textureless regions.

The processing speed of different frames varies from less than a second to several hundred seconds depending on the parameters such as number of stereo-matching iterations, number of bilateral upsampling iterations, and the size of the synthesized light field array. The user can keep taking pictures while the processing takes place in the background. Considering the performance of current mobile device processors, rendering real-time DoF effects on HD video streams is still not practical. However, this does not prevent users from taking consecutive video frames and rendering them offline, as can be seen in scenes 1 and 2 of Fig. 11. Also, since in general the stereo cameras on mobile devices have a small baseline, the disparity values of pixels in the downsampled images have certain max/min thresholds. We can reduce the number of disparity labels in the GCs algorithm and further improve the processing speed without introducing much performance penalty.

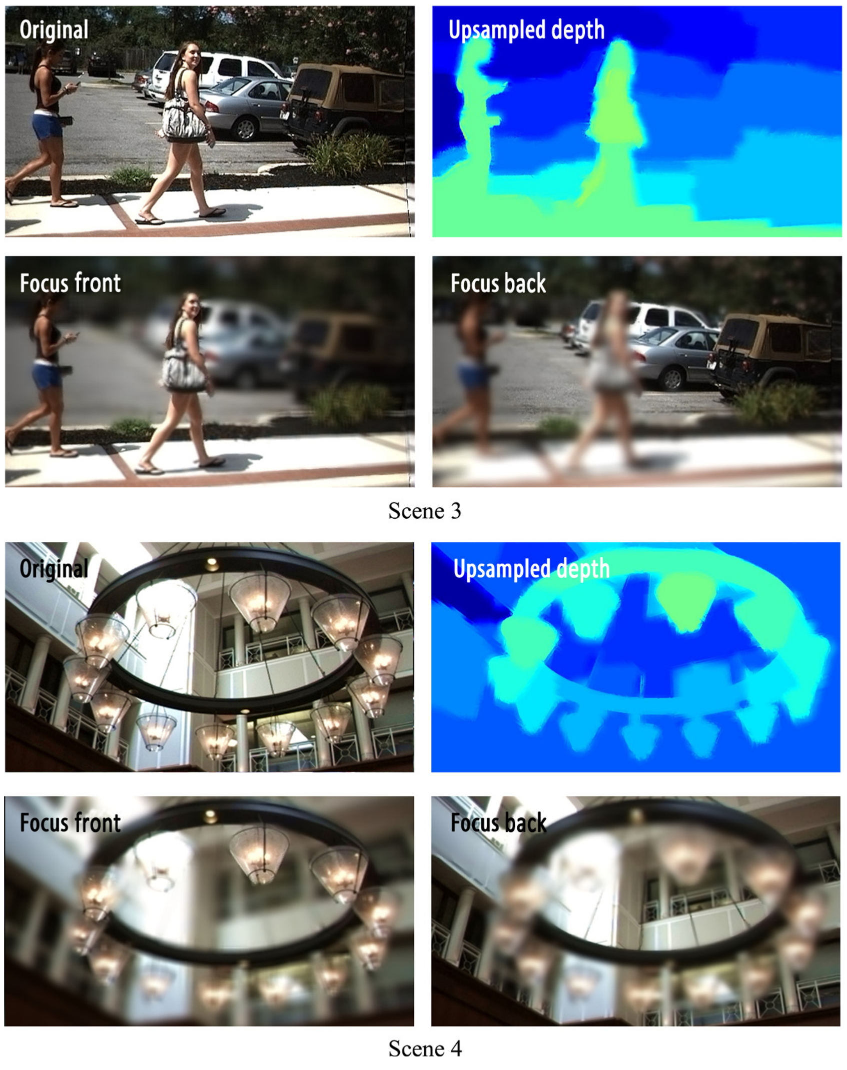

Fig. 12 Input disparity map and rendered images of our system on two real scenes with the same arrangement as in Fig. 11.

We first demonstrate our system in dynamic outdoor scenes. Figure 11 shows the results of two frames from the same video sequence. Since we currently do not have any auto-exposure or high-dynamic range (HDR) modules implemented, some parts of the photo are over-exposed. As shown in the photograph, many texture details are lost in the over-exposed regions, making it challenging for the stereo-matching algorithm to recover accurate disparity values. Moreover, the background lawn contains noticeable shadows and large portions of the building wall are textureless. This adds to the difficulty of finding pixel to pixel correspondences. Notwithstanding, our algorithm generates visually good-looking disparity maps. The edges of the woman's hand and arm are preserved when they are in focus, and objects outside of the focal plane are blurred smoothly.

Scene 3 of Fig. 12 displays a scene of two women walking in front of a parking lot. Typically the working range of the tablet sensor is from half a meter to 5 m. As a result, the cars in the parking lot are already approaching the maximum working distance of the sensor. This, however, does not affect the overall refocusing result as the cars with similar disparity values are either all in focus [Fig. 12, row 2, column 2] or blurred [Fig. 12, row 2, column 1]. The sidewalk in front of the parking lot has a lot of textureless areas, making it difficult to achieve coherent disparity values. As a result, the left and right parts of the sidewalk are blurred slightly differently although they are on the same plane [Fig. 12, row 2, column 2]. Also, because the women in scene 3 are farther away from the camera compared with the women in scenes 1 and 2, the boundaries of women in scene 3 are coarser and fine details on the bodies are lost. Therefore, foregrounds in scene 3 are more uniformly blurred compared with scenes 1 and 2.

Indoor scenes have controllable environments and undoubtedly aid the performance of our system. For example, most structures from an indoor scene are within the working range of our system and typically indoor lighting would not cause problems such as over-exposure or shadows. Scene 4

Table 4 Results of subjective quality rating tests.

User ♯	1	2	3	4	5	6	7	8	9	10	Average
Nonexperts	7	9	9	8	9	8	7	7	9	8	8.1
Experts	5	3	7	6	8	7	1	5	5	6	5.3

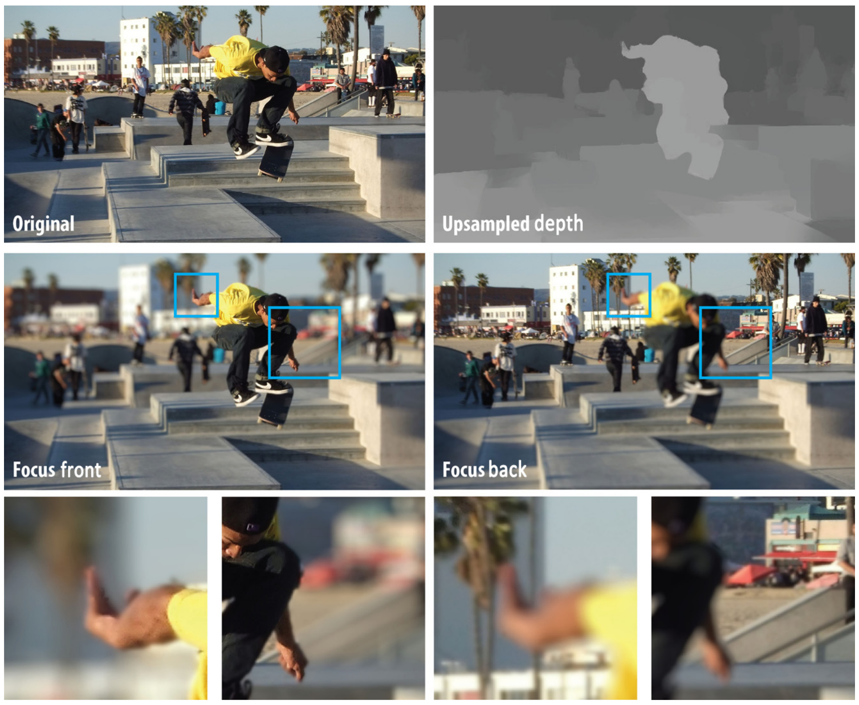

Fig. 13 Our result on a skateboard scene at 6 MP captured by Fujifilm FinePix Real 3-D camera (courtesy of Design-Design).[32]

of Fig. 12 shows the results on an indoor scene with transparent objects and textureless regions. Since our algorithm effectively fills holes and corrects bad pixels on the disparity map by using the guide color image, the resulting disparity map looks clean and disparity edges of the chandelier are well preserved [Fig. 12, row 3, column 2]. The upper left part of the wall surface is over-exposed and the light bulb in the foreground almost merged into the background. However, the disparity map still recovers edges correctly. As can be seen in Fig. 12, row 4, column 2, the defocus blur fades correctly from the out-of-focus light bulb regions into the in-focus wall regions, despite the fact that they are both white and do not have clear boundaries in between.

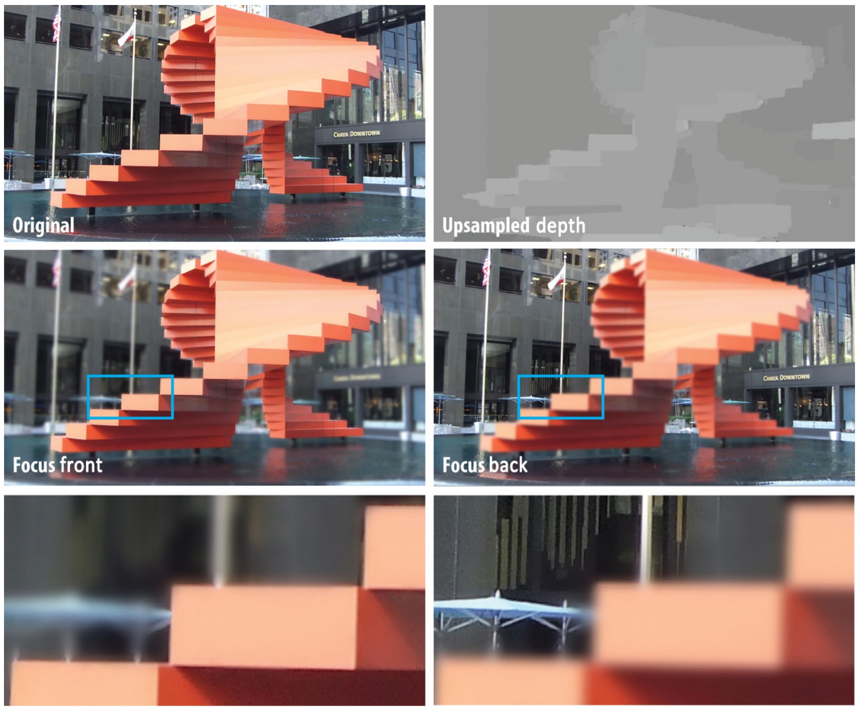

Fig. 14 Our result on a sculpture scene at 6 MP captured by Fujifilm FinePix Real 3-D camera (courtesy of Design-Design).[32]

The discussion here is based on our own captured data, and it is hard to evaluate rendered results because of the lack of ground truth. To address this problem, we conducted subjective rating tests with 20 people. Among these people, 10 have a computer vision or graphics background and the remaining have no expertize in the related field. For convenience and clarity, the rating is done on a 0 to 9 scale for measuring the quality of rendered results. We define the rating as follows: 0 (not acceptable), 1 (acceptable), 3 (good, but needs improvement), 5 (satisfactory), 7 (very good), and 9 (excellent). The test results can be found in Table 4. The average rating of the nonexpert group is 8.1, and the average rating from the experts is 5.3. Therefore, the overall quality of the rendered results can be concluded as satisfactory.

According to Table 2, our method returns the best disparity map results in terms of overall bad pixels percentage. Also, our system correctly handles complex scene structures with real-world illumination conditions. Last but not least, according to the resulting images in Fig. 8, we reduce aliasing artifacts in out-of-focus regions by blending multiple synthesized light field views together.

Finally, to demonstrate that our algorithm is also capable of generating high-quality DoF effects using high-resolution stereo input, we leverage mobile devices Fujifilm FinePix Real 3-D camera to capture a set of stereo images and to generate the shallow DoF images with refocus capabilities at 6-MP resolution, as shown in Figs. 13 and 14. Current light field cameras are not capable of generating such high-resolution images. Figure 13 shows the scene of a person playing with a skateboard. Our algorithm is able to preserve most of the depth discontinuities in the scene such as the edges of the hand, the skateboard, and the leg. Note that the background between the legs is marked as the foreground, leaving artifacts in the final rendering. This is due to the unsuccessful depth estimation of the GCs algorithm, and our current depth upsampling is largely relying on the initial estimation. In the future, we plan to employ the depth error correction into our upsampling scheme. Figure 14 shows a scene of a sculpture in a shopping mall. Despite the complex occlusion conditions in the scene, our algorithm is still able to synthesize shallow DoF effects with little artifacts such as fussy edges on the stairs.

8 Conclusion

We have presented an affordable solution for producing dynamic DoF effects on mobile devices. The whole system runs on an off-the-shelf tablet, which costs less than $400. We compare the performance of popular stereo-matching algorithms and design a hybrid resolution approach, which tries to improve both speed and accuracy. Also, we generate the synthesized light field by using a disparity warping scheme and render the high-quality DoF effects. Finally, we map all processing stages onto the Android system and control the computational imaging device by using the FCam architecture. Our system efficiently renders dynamic DoF effects with arbitrary aperture sizes and focal lengths in a variety of indoor and outdoor scenes.

Our future efforts include adding modules such as auto-exposure or HDR to improve the imaging quality. We would also like to explore the possibility of implementing our approach to sparse camera arrays with limited number of views.

Acknowledgments

We thank NVIDIA Corporation for providing the Tegra prototype devices. This project is supported by the National Science Foundation under grant IIS-1319598.

References

1. R. Ng et al., "Light field photography with a hand-held plenoptic camera," *Comput. Sci. Tech. Rep.* **2**(11), 1–11 (2005).
2. T. Georgiev et al., "Spatio-angular resolution tradeoffs in integral photography," in *Proc. of the 17th Eurographics Conf. on Rendering Techniques, EGSR'06*, Nicosia, Cyprus, pp. 263–272, Eurographics Association, Aire-la-Ville, Switzerland (2006).
3. T. Georgiev and A. Lumsdaine, "Focused plenoptic camera and rendering," *J. Electron. Imaging* **19**(2), 021106 (2010).
4. V. Kolmogorov and R. Zabih, "Computing visual correspondence with occlusions using graph cuts," in *Proc. Eighth IEEE Int. Conf. on Computer Vision, 2001. ICCV 2001*, Vol. 2, pp. 508–515, IEEE (2001).
5. A. Adams et al., "The frankencamera: an experimental platform for computational photography," *ACM Trans. Graph.* **29**(4), 1–12 (2010).
6. G. Lippmann, "La photographie intgrale," *Comp. Rend. Acad. Sci.* **146**, 446–451 (1908).
7. B. Wilburn et al., "High performance imaging using large camera arrays," *ACM Trans. Graph.* **24**(3), 765–776 (2005).
8. J. C. Yang et al., "A real-time distributed light field camera," in *Proc. 13th Eurographics Workshop on Rendering*, pp. 77–86, Eurographics Association, Aire-la-Ville, Switzerland (2002).
9. A. Veeraraghavan et al., "Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing," *ACM Trans. Graph.* **26**(3), 69 (2007).
10. Lytro, "Lytro Camera," https://www.lytro.com (March 2014).
11. T. Georgiev et al., "Lytro camera technology: theory, algorithms, performance analysis," *Proc. SPIE* **8667**, 86671J (2013).
12. PelicanImaging, "Pelican Imaging Array Camera," http://www.pelicanimaging.com (March 2014).
13. Z. Yu et al., "Racking focus and tracking focus on live video streams: a stereo solution," *Visual Comput.* **30**(1), 45–58 (2013).
14. Z. Yu et al., "Dynamic depth of field on live video streams: a stereo solution," in *Proc. of the 2011 Computer Graphics Int. Conf., CGI 2011*, Ottawa, Ontario, Canada, ACM (2011).
15. Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Trans. Pattern Anal. Mach. Intell.* **23**(11), 1222–1239 (2001).
16. J. Sun, N.-N. Zheng, and H.-Y. Shum, "Stereo matching using belief propagation," *IEEE Trans. Pattern Anal. Mach. Intell.* **25**(7), 787–800 (2003).
17. M. F. Tappen and W. T. Freeman, "Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters," in *Proc. Ninth IEEE Int. Conf. Computer Vision*, pp. 900–906, IEEE (2003).
18. D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *Int. J. Comput. Vision* **47**(1–3), 7–42 (2002).
19. A. Troccoli, D. Pajak, and K. Pulli, "Fcam for multiple cameras," *Proc. SPIE* **8304**, 830404 (2012).
20. Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(11), 1330–1334 (2000).
21. H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information," in *IEEE Computer Society Conf. Computer Vision and Pattern Recognition, 2005. CVPR 2005*, Vol. 2, pp. 807–814, IEEE (2005).
22. A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching," in *Proc. of the 10th Asian Conf. on Computer Vision—Volume Part I, ACCV'10*, Queenstown, New Zealand, pp. 25–38, Springer-Verlag, Berlin, Heidelberg (2011).
23. J. Kopf et al., "Joint bilateral upsampling," *ACM Trans. Graph.* **26**(3), 96 (2007).
24. C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Sixth Int. Conf. Computer Vision, 1998*, pp. 839–846, IEEE (1998).
25. S. Lee, E. Eisemann, and H.-P. Seidel, "Depth-of-field rendering with multiview synthesis," *ACM Trans. Graph.* **28**(5), 1–6 (2009).
26. S. Lee, E. Eisemann, and H.-P. Seidel, "Real-time lens blur effects and focus control," *ACM Trans. Graph.* **29**(4), 1–7 (2010).
27. R. L. Cook, T. Porter, and L. Carpenter, "Distributed ray tracing," *ACM SIGGRAPH Comput. Graph.* **18**(3), 137–145 (1984).
28. P. Haeberli and K. Akeley, "The accumulation buffer: hardware support for high-quality rendering," *ACM SIGGRAPH Comput. Graph.* **24**(4), 309–318 (1990).
29. X. Yu, R. Wang, and J. Yu, "Real-time depth of field rendering via dynamic light field generation and filtering," *Comput. Graph. Forum* **29**(7), 2099–2107 (2010).
30. L. Stroebel et al., *Photographic Materials and Processes*, Focal Press, Boston/London (1986).

31. J.-X. Chai et al., "Plenoptic sampling," in *Proc. 27th Annual Conf. Computer Graphics and Interactive Techniques*, pp. 307–318, ACM Press/Addison-Wesley Publishing Co., New York, NY (2000).
32. P. Simcoe, "Design–Design Sample 3D Images," http://www.design-design.co.uk/sample-mpo-3d-images-for-television-display (March 2014).

Qiaosong Wang received his BEng degree from the Department of Automation Science and Technology, Xi'an Jiaotong University, in 2011. He is now a PhD student at the Department of Computer and Information Sciences, University of Delaware. His research interests include computer vision and mobile robotic perception.

Zhan Yu has been a research scientist at Adobe Systems Inc. since December 2013. Before that, he received his PhD degree in computer science from the University of Delaware and a BS degree in software engineering from Xiamen University. His research interests include computational photography, computer graphics, and computer vision.

Christopher Rasmussen is an associate professor in the Department of Computer & Information Sciences at the University of Delaware. He received his PhD degree in computer science from Yale University in 2000 and an AB degree from Harvard University in 1993. His research interests are in mobile robot perception and field robotics. He is the recipient of an NSF CAREER Award.

Jingyi Yu is an associate professor in the Department of Computer & Information Sciences and the Department of Electrical & Computer Engineering at the University of Delaware. He received his BS degree from Caltech in 2000 and a PhD degree from MIT in 2005. His research interests span a range of topics in computer vision and graphics, especially on computational cameras and displays. He is a recipient of both an NSF CAREER Award and the AFOSR YIP Award.