Stereo Vision based Depth of Field Rendering on a Mobile Device

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

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 was until recently a quality unique to expensive and bulky movie cameras. In this paper, we propose a computational approach to generate realistic DoF effects for mobile devices such as tablets. We first calibrate the rear-facing stereo cameras and rectify 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 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 subjective evaluation tests.

Keywords: Depth of Field, Programmable Cameras, Joint Bilateral Upsampling, Light Field

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 was 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 set up 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, Lumsdaine and Georgiev[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 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 3D-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 high-resolution disparity maps of the scene through Graph Cuts (GC)[4] and then generate a synthesized light field for dynamic DoF effect rendering.

Further author information: (Send correspondence to Qiaosong Wang)
Qiaosong Wang: E-mail: qiaosong@udel.edu, Telephone: +1(302)220-9707

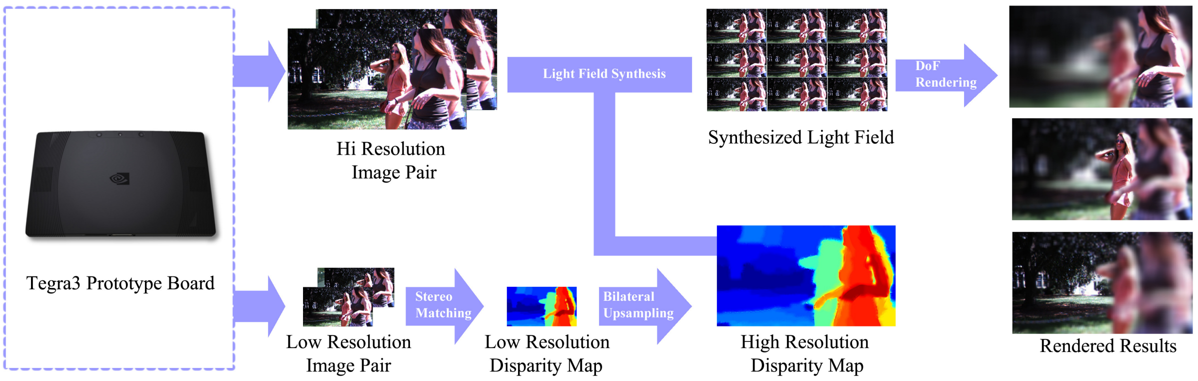


Figure 1. The NVIDIA Tegra3 prototype tablet and the processing pipeline of our software system. All modules are implemented on the Android 4.1 operating system.

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 Figure 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 4D radiance information about the scene. The captured light information can later be used for applications like dynamic DoF rendering and 3D reconstruction. Since conventional imaging systems are only two-dimensional, a variety of methods have been developed for capturing and storing light fields in a 2D form. Ives and Lippmann[6] were the first to propose a prototype camera to capture light fields. The Stanford multi-camera array[7] is composed of 128 synchronized CMOS firewire cameras and streams captured data to 4 PC hosts for processing. Because of the excessive data volume, DoF effects are rendered offline. The MIT 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[1] has 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, Lumsdaine and Georgiev[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 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, expensive and the algorithm is highly dependent on the GPU performance, making it hard to transfer the proposed method to small handheld 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 dollars and the camera itself requires external power source, 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 paper 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 dollars. 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 handheld Android devices with limited computational power. Besides, we conducted extensive experiments to obtain parameters that generate optimal results. Therefore, it's easy to install and use our software, no hardware setup or parameter adjustment is required. Furthermore, our system uses Graph Cuts[15] instead of Belief Propagation (BP)[16] for stereo matching, and is tested working under complex illumination conditions. According to tests carried out by M. Tappen *et al.*,[17] Graph Cuts generates smoother solutions compared to BP and consistently performs 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 generating dynamic DoF effects. We also discussed why our method is good at reducing boundary discontinuity and intensity leakage artifacts compared to 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 Graph Cuts for disparity map calculation and the system is capable of working with a variety of scene structures and illumination conditions.

3. OVERVIEW

In this paper, we demonstrate that 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, then apply the Graph Cuts 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 the 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, 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 centimeter baseline. The third sensor is on the same side of the multi-touch 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. 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's 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. Firstly, 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, 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 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 The FCam API

Many computational photography applications follow the general pattern of capturing multiple images with changing parameters and combine 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 a modified software stack to include the FCam API. Alejandro *et al.* extended the implementation of FCam API to support multiple cameras[19] and enables 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 send updates to the other sensor. Firstly, 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

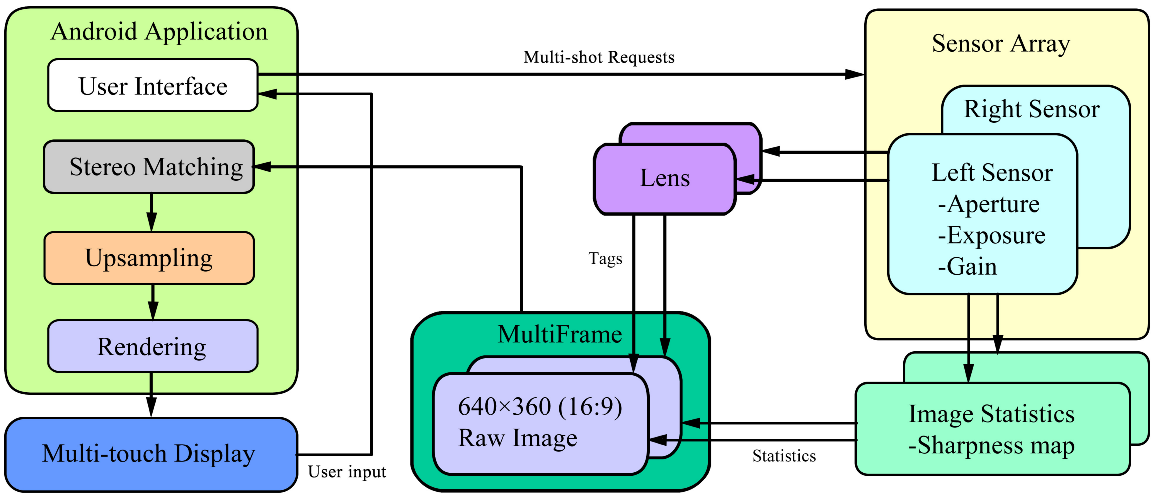


Figure 2. This diagram shows how our application interacts with the camera system. Our application accepts user input from the multi-touch screen, sends multi-shot requests to the sensors with desired parameters and then transfers the raw stereo image pairs to the stereo matching module. We then upsample 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.

bleeding artifacts. Global stereo estimation methods such as Graph Cuts (GC)[15] and Belief Propagation (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 mega pixel) 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 GC. Next, we take each raw color image as the guidance image and upsample the disparity map via joint bilateral upsampling.[23]

5.1 Graph Cuts 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; N is the set of neighboring pixels, the correspondence problem can be formulated as minimizing the following energy function:

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

The local minimization of function (1) can be efficiently approximated using the alpha-expansion presented in.[15] In our implementation, we set the number of disparities to be 16 and run the algorithm for 5 iterations. If the algorithm cannot find a valid alpha expansion that decreases the overall energy function value, it may also terminate in less than 5 iterations. The performance of GC on the Tegra 3 tablet platform can be found at 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 block matching (SGBM),[21] efficient large-scale stereo (ELAS)[22] and Graph Cuts (GC).[15] Table 1 shows the running time of these algorithms on the Tegra 3 tablet and Figure 3 shows the calculated disparity map results. According to our experiments, BM is faster than SGBM

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.

Data sets	BM	SGBM	ELAS	GC
Tsukuba	15	28	51	189
Venus	13	30	97	234
Cones	19	42	124	321

Table 2. Evaluation of different stereo matching methods on the Middlebury stereo datasets in bad pixel percentage (%). The method shown in the last row applies 5 iterations of Joint Bilateral Upsampling to the downsampled results (half of the original size) of GC, using the full resolution color image as guidance image. The resolutions of the four datasets (Tsukuba, Venus, Teddy, Cones) are 384×288, 434×383, 450×375, 450×375, respectively. If not specified, raw image size of each individual dataset will be the same for the remainder of this paper. Noncc: bad pixel percentage in non-occluded regions; All: bad pixel percentage in all regions; Disc: bad pixel percentage in regions near depth discontinuities
.

	Tsukuba	Venus	Teddy	Cones
	noncc all disc	noncc all disc	noncc all disc	noncc 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
GC	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

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 while bigger 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 Figure 3, we can still identify the curved surface area of the cones from results generated by SGBM and ELAS, but the same area looks almost flat in BM. SGBM and ELAS are 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 Figure 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 GC gives smooth transitions between regions of different disparity values. According to Table 2, the GC 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 GC rather than other methods which runs faster. Another reason is that we are running the GC algorithm on low-resolution imagery. According to Table 1, the running time is around 250 ms, which is still acceptable compared to 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 result 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 C. Tomasi *et al.*[24] is a simple, non-iterative scheme for edge preserving smoothing which uses both a spatial kernel and a range kernel. However, for low signal-to-noise ratio (SNR) images, this algorithm cannot keep the edge information very well. A variant called joint bilateral filter introduced by J. Kopf[23] *et al.* 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

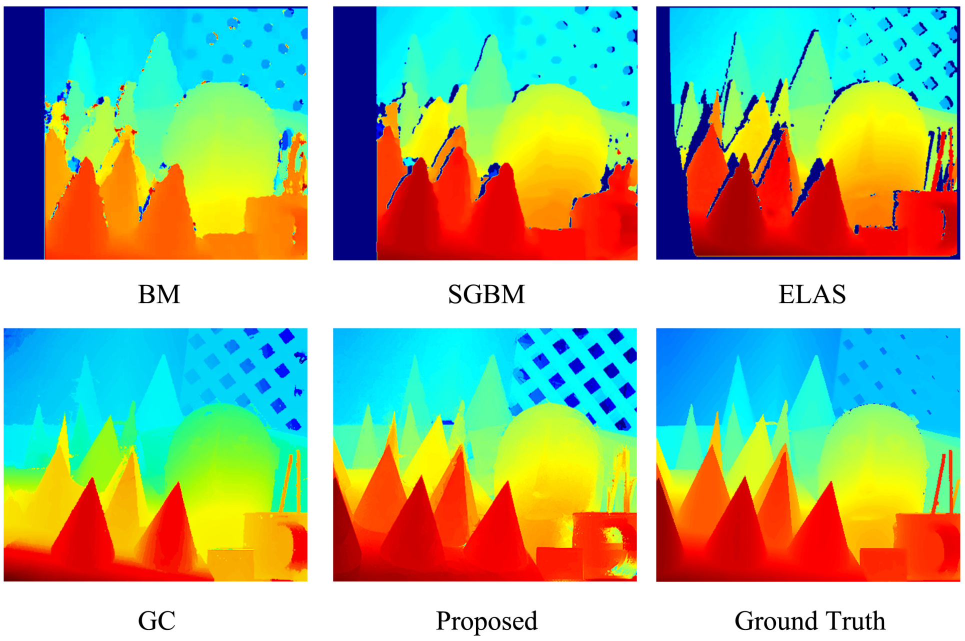


Figure 3. Comparison of our approach and other popular stereo matching algorithms.

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:

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

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. Firstly, blurring 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 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.

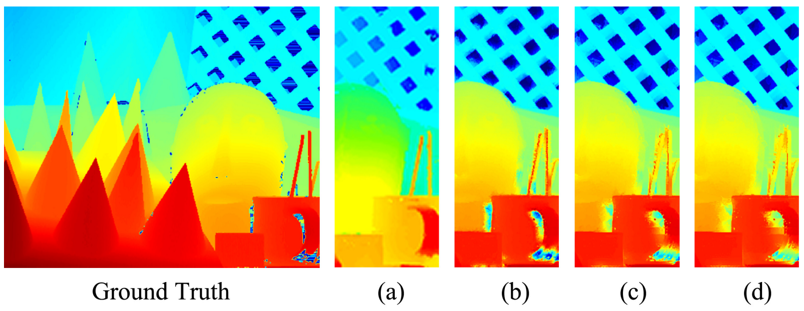


Figure 4. Comparison of results using different number of iterations.(a),(b),(c),(d) are using 0,5,10,20 iterations respectively.

Figure 4 shows results after different number 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 starts 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 Figure 5, the quality of the disparity map can be improved during the first 5 or 6 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 starts 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 Figure 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 5 iterations should return good results under a variety of scene structures. Generally, it takes around 40 milliseconds to finish the 5 iteration steps on the tablet. Figure 6 illustrates the detailed view of our result compared to other standard upsampling methods. Because DoF effects are most apparent around 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.

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
	noncc all disc	noncc all disc	noncc all disc	noncc 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

6. DEPTH OF FIELD RENDERING

Once we obtained the high resolution disparity map, we can proceed to synthesize dynamic DoF effects. Previous studies suggested that real time DoF effects can be obtained by applying a spatially varying blur on the color image and use 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 paper, we use a similar approach to[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 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

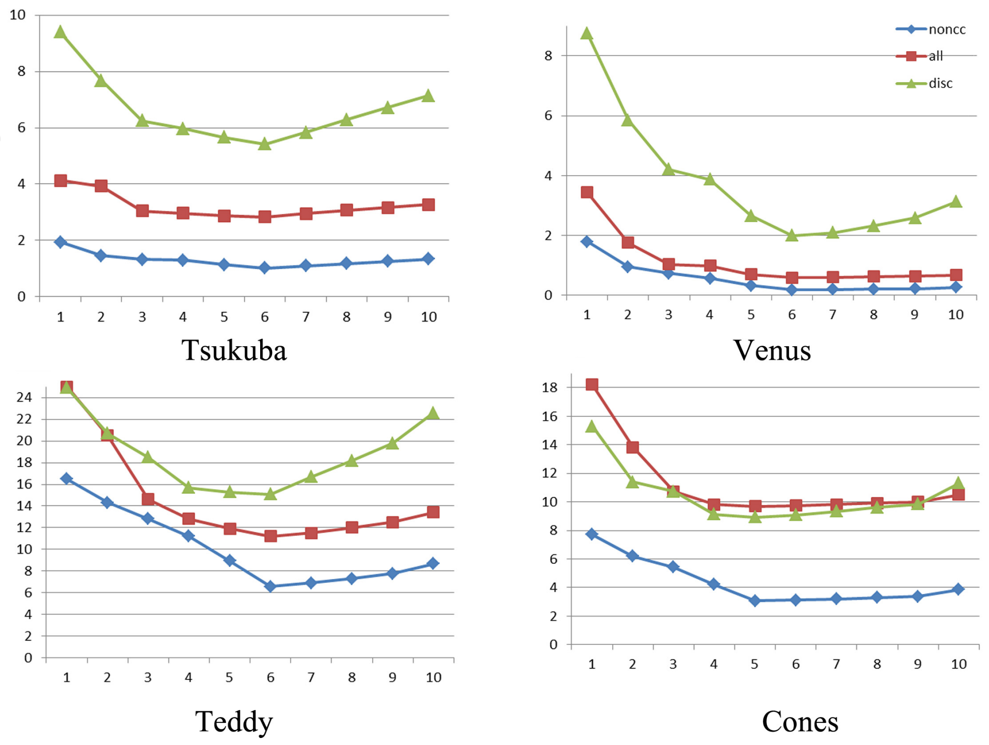


Figure 5. Evaluation of the disparity maps using different number of Joint Bilateral Upsampling iterations on the Middlebury stereo dataset. The horizontal axis shows the number of iterations and the vertical axis shows the bad pixel percentage.

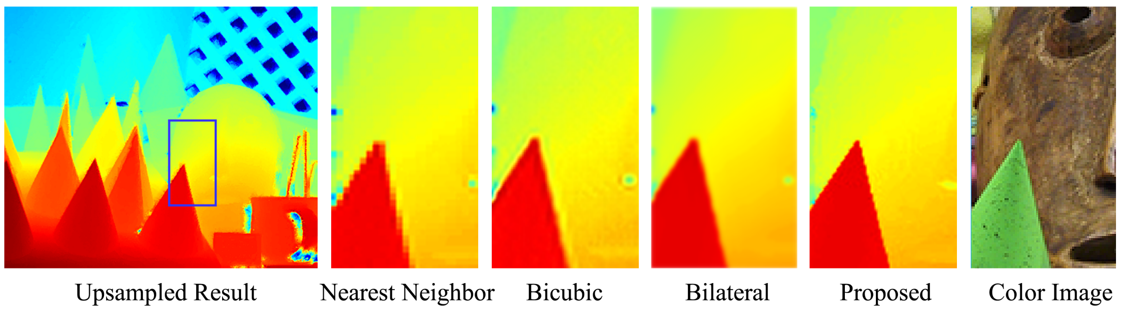


Figure 6. Comparison of our approach and other upsampling algorithms on the Middlebury cones dataset.

take nearby pixels to fill the holes. Note that this module is only used for generating pleasing individual views for the user to randomly shift the perspective. In the final rendering process, missing rays are simply discarded and the filled pixels are not used. Figure 7 shows 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 $L_{out}(s,t,u,v)$ to represent out-of-lens light field and $L_{in}(s,t,u,v)$ to represent in-lens light field, the pixels in this image can be obtained as a weighted integral of all incoming radiance through the lens:

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

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 depth with corresponding disparity γ_f by finding

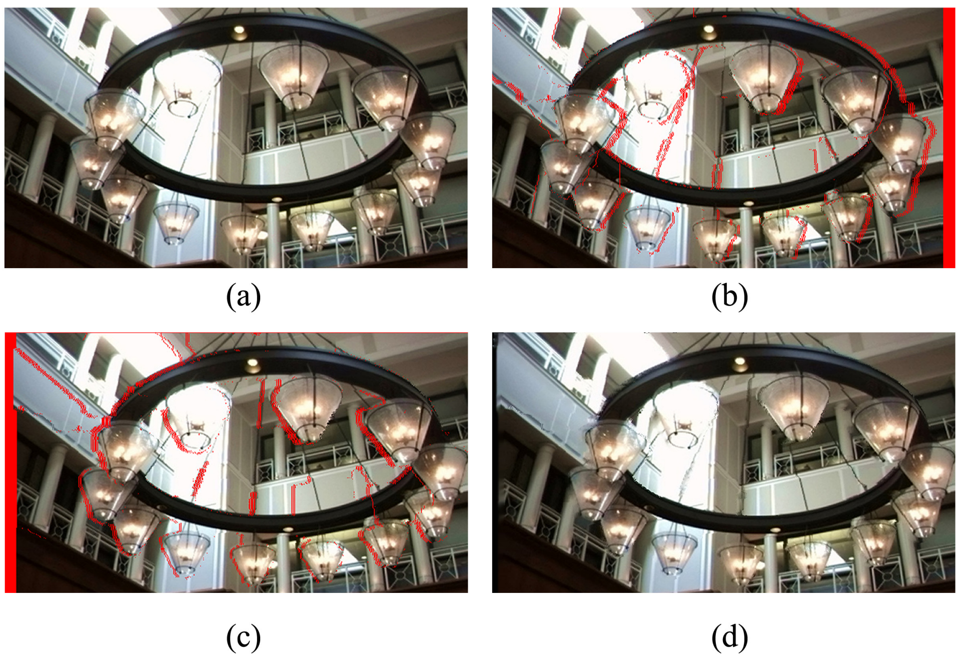


Figure 7. Synthesized light field view, missing pixels are marked in red. (a) input image (b) warped left side view (c) warped right side view (d) result image using our hole-filing algorithm, taking (c) as the input.

the pixel index in camera R_{st} using equation 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:

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

Figure 8 shows 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.

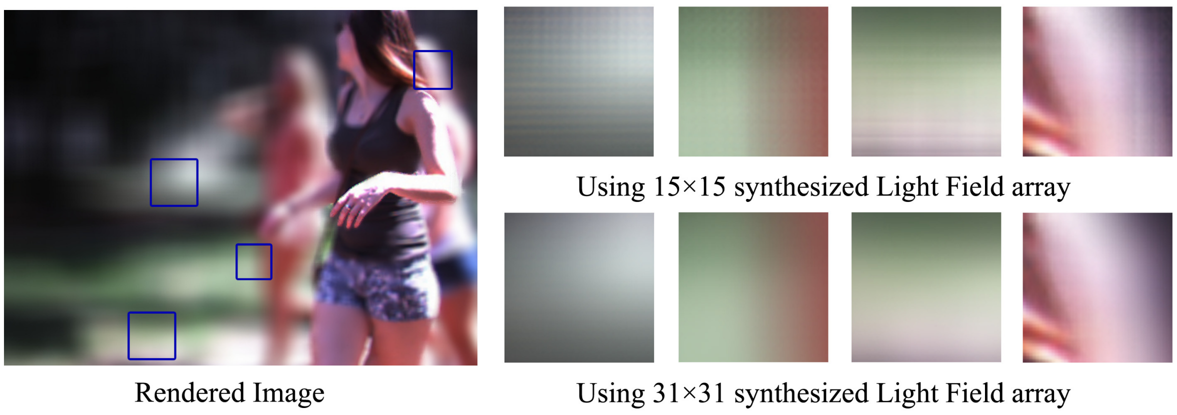

Rendered Image
Using 15×15 synthesized Light Field array
Using 31×31 synthesized Light Field array

Figure 8. Comparing rendering results with different sizes of the synthesized Light Field array.

6.2 Comparison of our method of 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 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 the background (the building). For Fig. 9 (a, b), we assume the camera to be focused at the background and for Fig. 9 (c, 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 Fig. 9 (a, b). Suppose P_b is a point on the background building and 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 the 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 and attempts to use rays originating from both the foreground and background to calculate 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. 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 (Fig. 9 (c,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 reasonable approximations of the ground truth. For the last case, consider a point P_f on the foreground 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 produce 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.

Fig. 10 shows results of our method and single image blurring on an outdoor scene. As mentioned before, our method reduces artifacts on boundary regions compared to 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)) while 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. Figure 11 and 12 shows the results generated by our system under different scene structures and illumination conditions. Scene 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 vary from less than a second to several hundred seconds, depending on 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

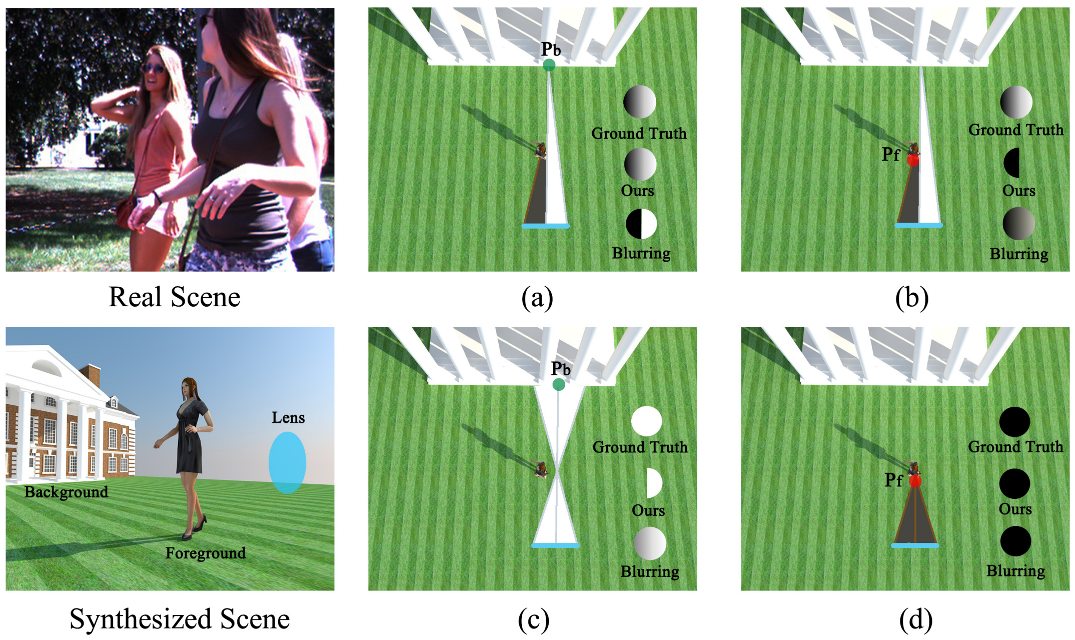


Figure 9. Causes of different boundary artifacts. See Section 6.2 for details.

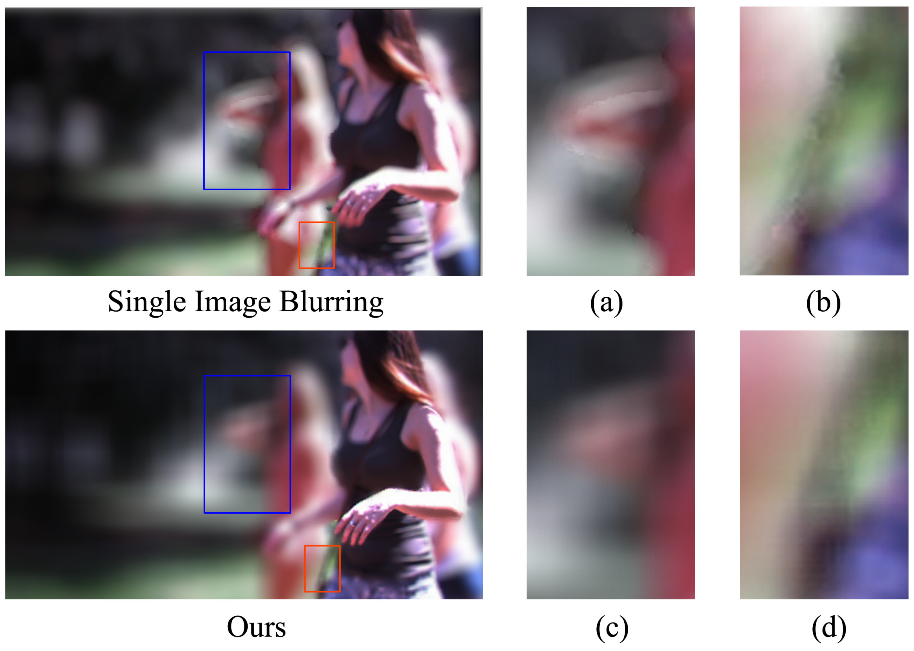


Figure 10. Comparison between our method and single image blurring. Single image blurring methods suffer from intensity leakage (a) and boundary discontinuity (b) artifacts. Our method (c,d) reduces these artifacts.

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 render them offline, as can be seen in scene 1 and 2 of Figure 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 Graph Cuts algorithm and further improve processing speed without introducing much performance penalty.

We first demonstrate our system in dynamic outdoor scenes. Figure 11 shows 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.

Figure 12 scene 3 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 five meters. 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 part 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 to the women in scene 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 to scene 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 won't cause problems such as over-exposure or shadows. Scene 4 of Figure 12 shows 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 Figure 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.

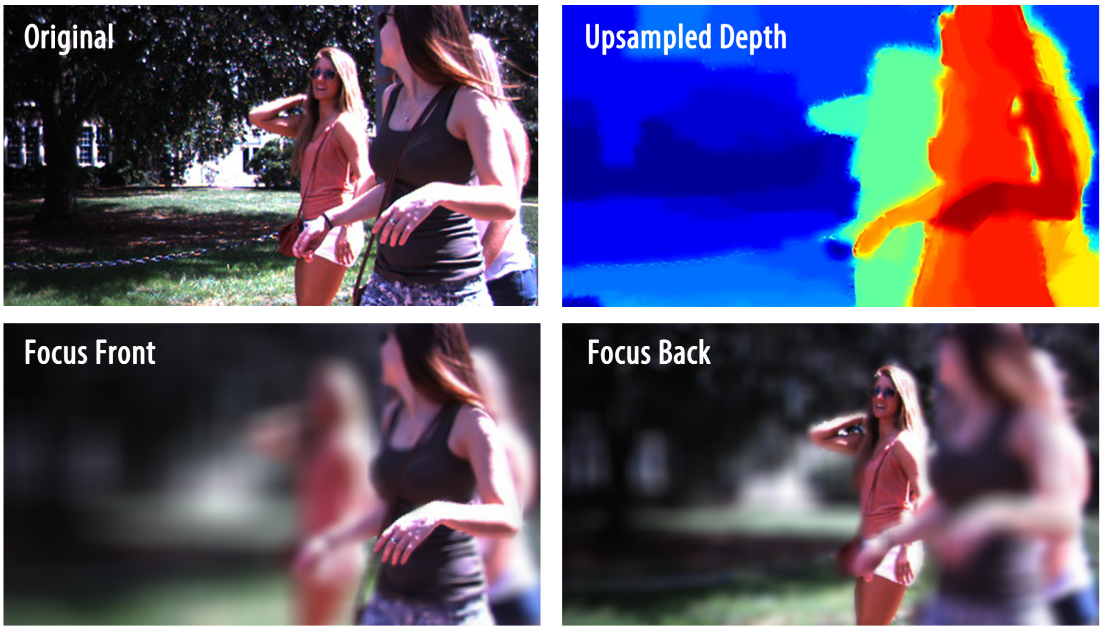
The discussion here is based on our own captured data and it is hard to evaluate rendered results because of the lack of ground truth. To address this problem, we conducted subjective rating tests with twenty people. Among these people, ten have a computer vision or graphics background and the remaining have no expertise in the related field. For convenience and clarity, the rating is done on a 0-9 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 at Table 4. The average rating of the non-expert 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 result images in Figure 10, we reduce aliasing artifacts in out-of-focus regions by blending multiple synthesized light field views together.

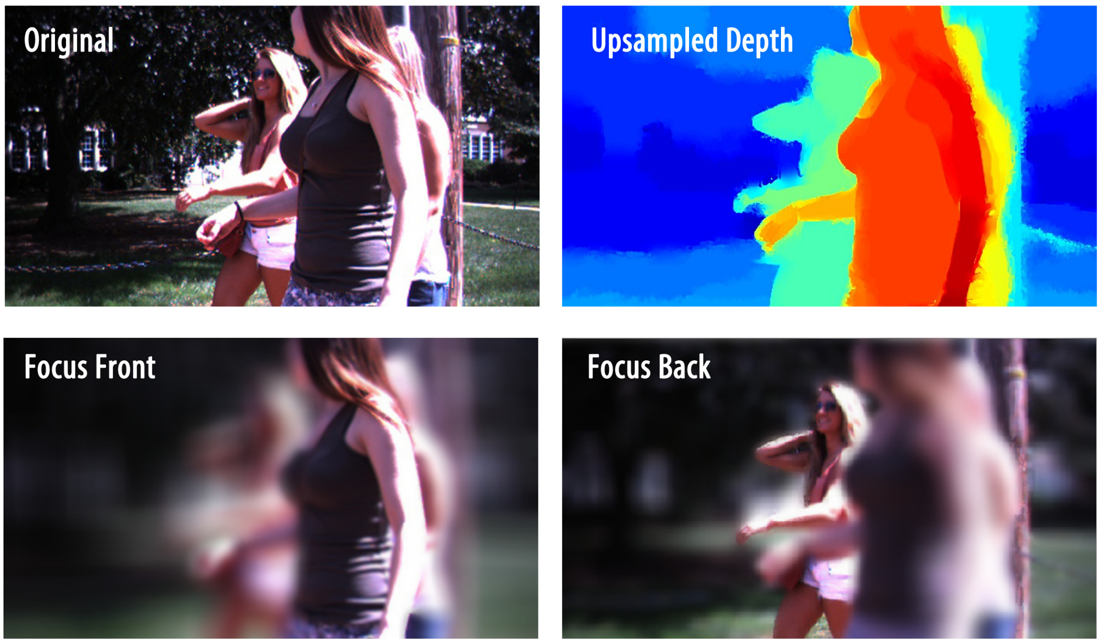
Table 4. Results of subjective quality rating tests.

User ♯	1	2	3	4	5	6	7	8	9	10	Average
Non-experts	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

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 3D camera to capture a set of stereo images and generate shallow DoF images with refocus capabilities at 6MP resolution, as shown in Figure 13 and

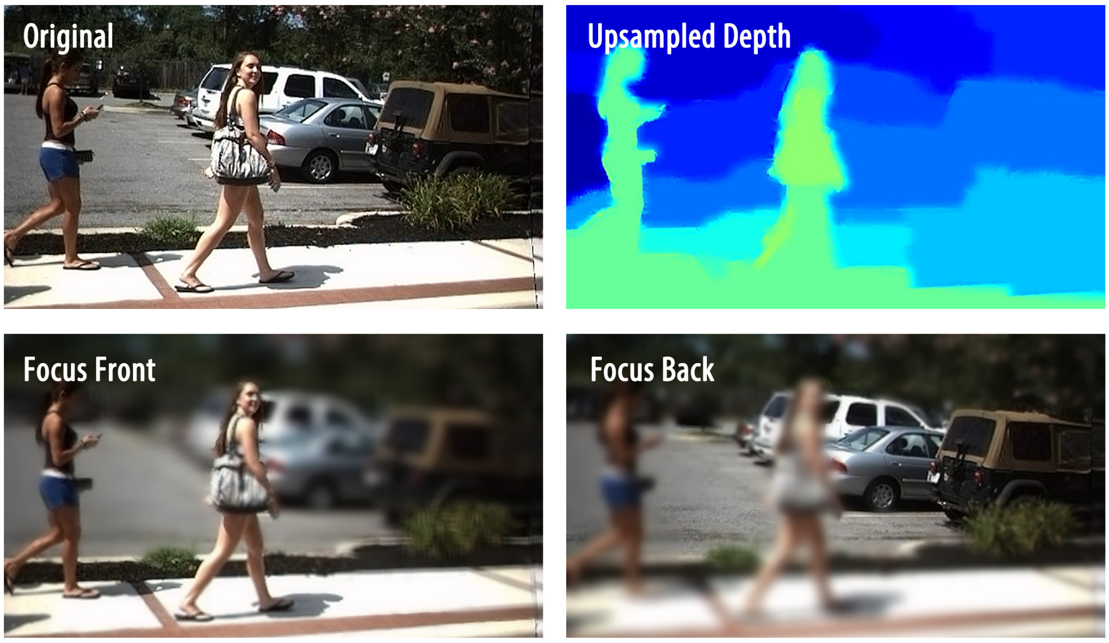


Scene 1

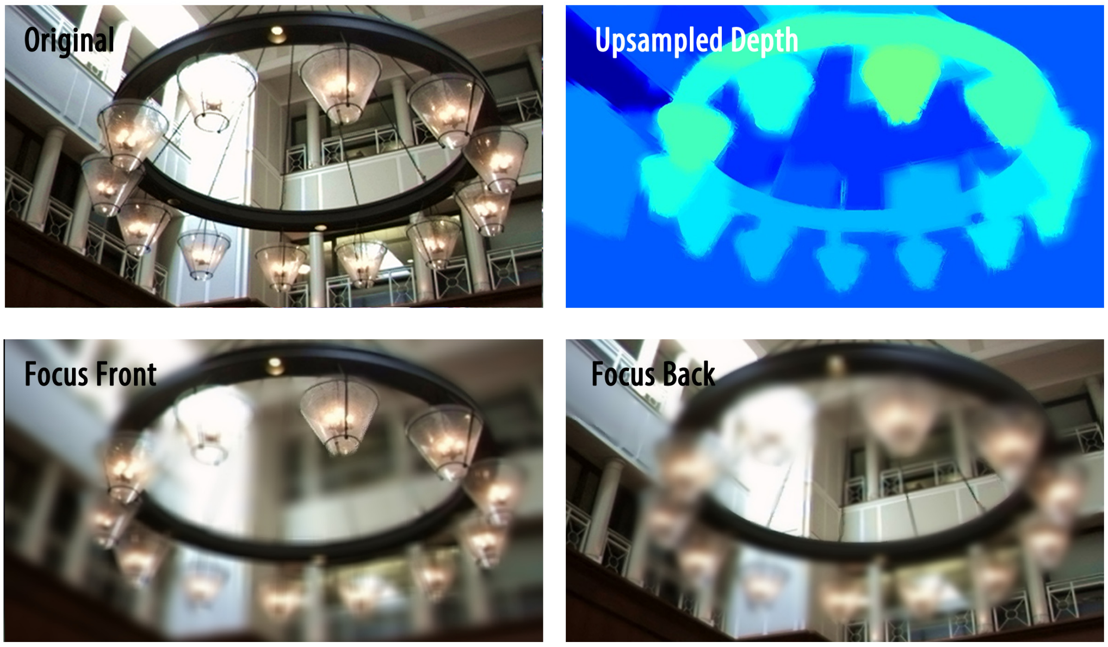


Scene 2

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



Scene 3



Scene 4

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

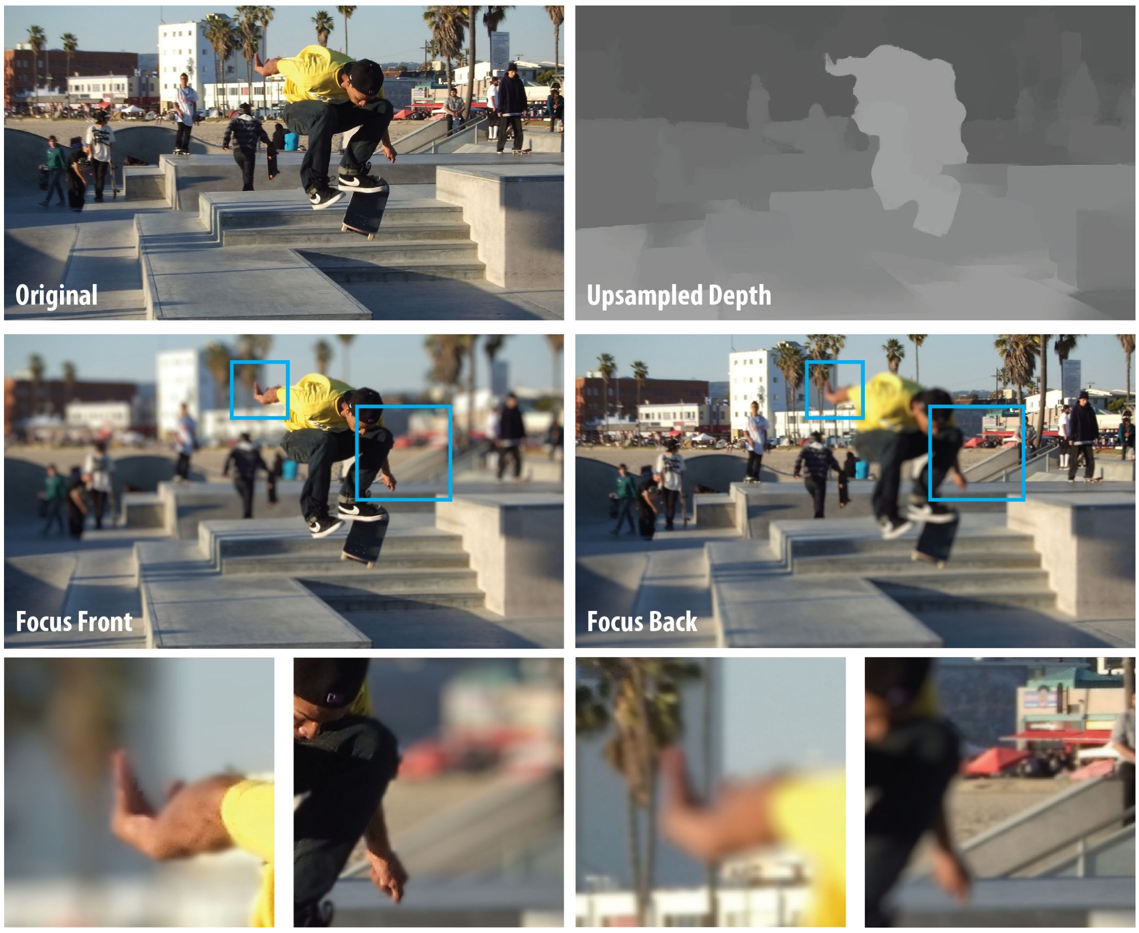


Figure 13. Our result on a skateboard scene at 6MP captured by Fujifilm FinePix Real 3D camera. Courtesy of Design-Design.[32]

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 Graph Cut algorithm and our current depth upsampling is largely relying on the initial estimation. In the future, we plan to employ depth error correction into our upsampling scheme. Figure 14 shows a scene of a sculpture in a shopping mall. Dispite 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 dollars. We compare the performance of popular stereo matching algorithms and design a hybrid resolution approach to try to improve both speed and accuracy. Also, we generate the synthesized light field by using a disparity warping scheme and render 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.

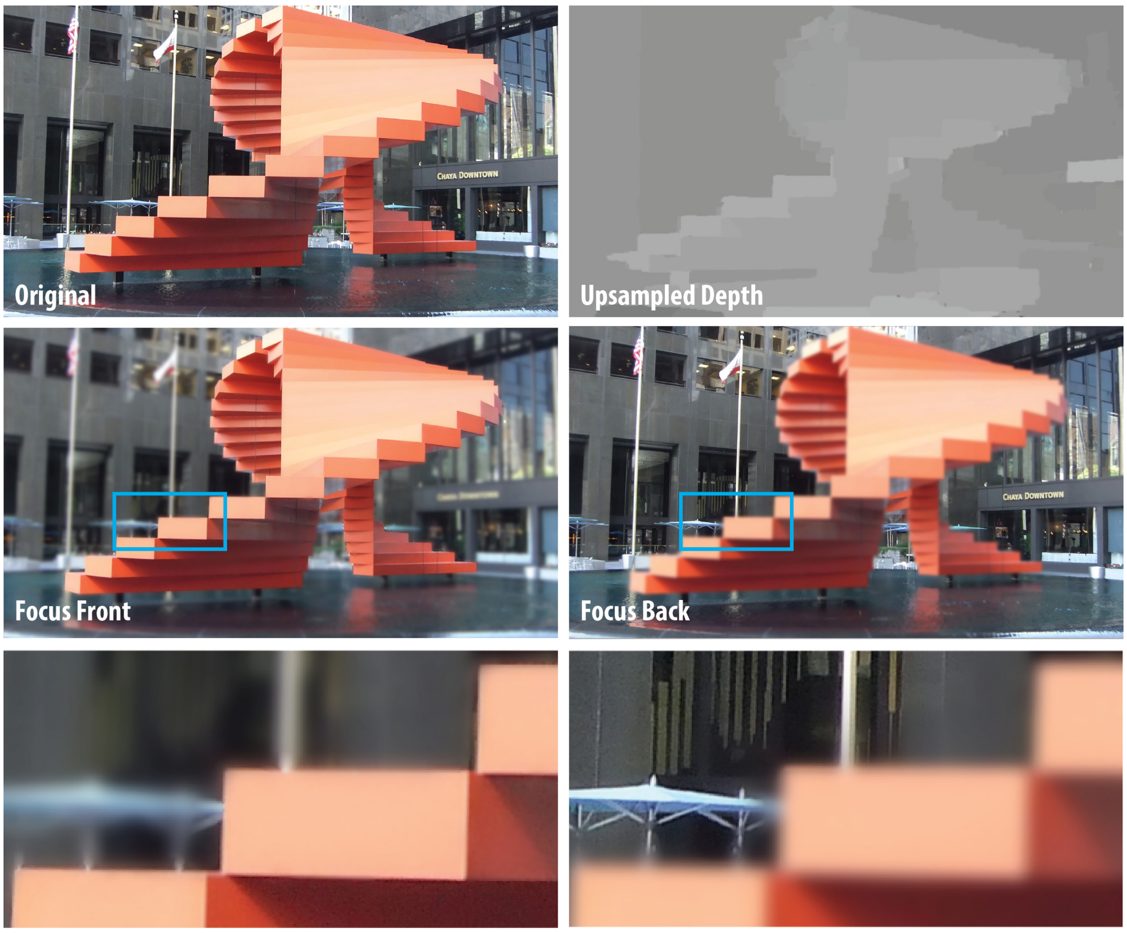


Figure 14. Our result on a sculpture scene at 6MP captured by Fujifilm FinePix Real 3D camera. Courtesy of Design-Design.[32]

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] Ng, R., Levoy, M., Brédif, M., Duval, G., Horowitz, M., and Hanrahan, P., "Light field photography with a hand-held plenoptic camera," *Computer Science Technical Report CSTR* **2**(11) (2005).

[2] Georgiev, T., Zheng, K. C., Curless, B., Salesin, D., Nayar, S., and Intwala, C., "Spatio-angular resolution tradeoffs in integral photography.," *Rendering Techniques* **2006**, 263–272 (2006).

[3] Georgiev, T. and Lumsdaine, A., "Focused plenoptic camera and rendering," *Journal of Electronic Imaging* **19**(2), 021106–021106 (2010).

[4] Kolmogorov, V. and Zabih, R., "Computing visual correspondence with occlusions using graph cuts," in [*Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*], **2**, 508–515, IEEE (2001).

[5] Adams, A., Jacobs, D. E., Dolson, J., Tico, M., Pulli, K., Talvala, E.-V., Ajdin, B., Vaquero, D., Lensch, H., Horowitz, M., et al., "The frankencamera: an experimental platform for computational photography," *ACM Transactions on Graphics (TOG)* **29**(4), 29 (2010).

[6] Lippmann, G., "La photographie intgrale," *Comptes-Rendus,Acadmie des Sciences* **146**, 446–451 (1908).

[7] Wilburn, B., Joshi, N., Vaish, V., Talvala, E.-V., Antunez, E., Barth, A., Adams, A., Horowitz, M., and Levoy, M., "High performance imaging using large camera arrays," in [*ACM Transactions on Graphics (TOG)*], **24**(3), 765–776, ACM (2005).

[8] Yang, J. C., Everett, M., Buehler, C., and McMillan, L., "A real-time distributed light field camera," in [*Proceedings of the 13th Eurographics workshop on Rendering*], 77–86, Eurographics Association (2002).

[9] Veeraraghavan, A., Raskar, R., Agrawal, A., Mohan, A., and Tumblin, J., "Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing," *ACM Transactions on Graphics* **26**(3), 69 (2007).

[10] Lytro, "www.lytro.com."

[11] Georgiev, T., Yu, Z., Lumsdaine, A., and Goma, S., "Lytro camera technology: theory, algorithms, performance analysis," in [*IS&T/SPIE Electronic Imaging*], 86671J–86671J, International Society for Optics and Photonics (2013).

[12] PelicanImaging, "www.pelicanimaging.com."

[13] Yu, Z., Yu, X., Thorpe, C., Grauer-Gray, S., Li, F., and Yu, J., "Racking focus and tracking focus on live video streams: a stereo solution," *The Visual Computer* , 1–14 (2013).

[14] Yu, Z., Thorpe, C., Yu, X., Grauer-Gray, S., Li, F., and Yu, J., "Dynamic depth of field on live video streams: A stereo solution,"

[15] Boykov, Y., Veksler, O., and Zabih, R., "Fast approximate energy minimization via graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **23**(11), 1222–1239 (2001).

[16] Sun, J., Zheng, N.-N., and Shum, H.-Y., "Stereo matching using belief propagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **25**(7), 787–800 (2003).

[17] Tappen, M. F. and Freeman, W. T., "Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters," in [*Ninth IEEE International Conference on Computer Vision*], 900–906, IEEE (2003).

[18] Scharstein, D. and Szeliski, R., "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision* **47**(1-3), 7–42 (2002).

[19] Troccoli, A., Pajak, D., and Pulli, K., "Fcam for multiple cameras," in [*IS&T/SPIE Electronic Imaging*], 830404–830404, International Society for Optics and Photonics (2012).

[20] Zhang, Z., "A flexible new technique for camera calibration," *Pattern Analysis and Machine Intelligence, IEEE Transactions on* **22**(11), 1330–1334 (2000).

[21] Hirschmuller, H., "Accurate and efficient stereo processing by semi-global matching and mutual information," in [*IEEE Computer Society Conference on Computer Vision and Pattern Recognition*], **2**, 807–814, IEEE (2005).

[22] Geiger, A., Roser, M., and Urtasun, R., "Efficient large-scale stereo matching," in [*Computer Vision–ACCV 2010*], 25–38, Springer (2011).

[23] Kopf, J., Cohen, M. F., Lischinski, D., and Uyttendaele, M., "Joint bilateral upsampling," *ACM Trans. Graph.* **26**(3), 96 (2007).

[24] Tomasi, C. and Manduchi, R., "Bilateral filtering for gray and color images," in [*Sixth International Conference on Computer Vision, 1998.*], 839–846, IEEE (1998).

[25] Lee, S., Eisemann, E., and Seidel, H.-P., "Depth-of-field rendering with multiview synthesis," in [*ACM Transactions on Graphics (TOG)*], **28**(5), 134, ACM (2009).

[26] Lee, S., Eisemann, E., and Seidel, H.-P., "Real-time lens blur effects and focus control," *ACM Transactions on Graphics (TOG)* **29**(4), 65 (2010).

[27] Cook, R. L., Porter, T., and Carpenter, L., "Distributed ray tracing," in [*ACM SIGGRAPH Computer Graphics*], **18**(3), 137–145, ACM (1984).

[28] Haeberli, P. and Akeley, K., "The accumulation buffer: hardware support for high-quality rendering," in [*ACM SIGGRAPH Computer Graphics*], **24**(4), 309–318, ACM (1990).

[29] Yu, X., Wang, R., and Yu, J., "Real-time depth of field rendering via dynamic light field generation and filtering," in [*Computer Graphics Forum*], **29**(7), 2099–2107, Wiley Online Library (2010).

[30] Stroebel, L., Compton, J., Current, I., and Zakia, R., [*Photographic Materials and Processes*], Focal Press, Boston/London (1986).

[31] Chai, J.-X., Tong, X., Chan, S.-C., and Shum, H.-Y., "Plenoptic sampling," in [*Proceedings of the 27th annual conference on Computer graphics and interactive techniques*], 307–318, ACM Press/Addison-Wesley Publishing Co. (2000).

[32] Design-Design, "http://www.design-design.co.uk/sample-mpo-3d-images-for-television-display."