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 values. 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 tested in a variety of environments with satisfactory results.

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 values. 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 advance in computational photography enables 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

Digital Photography X, edited by Nitin Sampat, Radka Tezaur, et. al., Proc. of SPIE-IS&T
Electronic Imaging, SPIE Vol. 9023, 902307 · © 2014 SPIE-IS&T · CCC code: 0277-786X/14/$18
doi: 10.1117/12.2038196

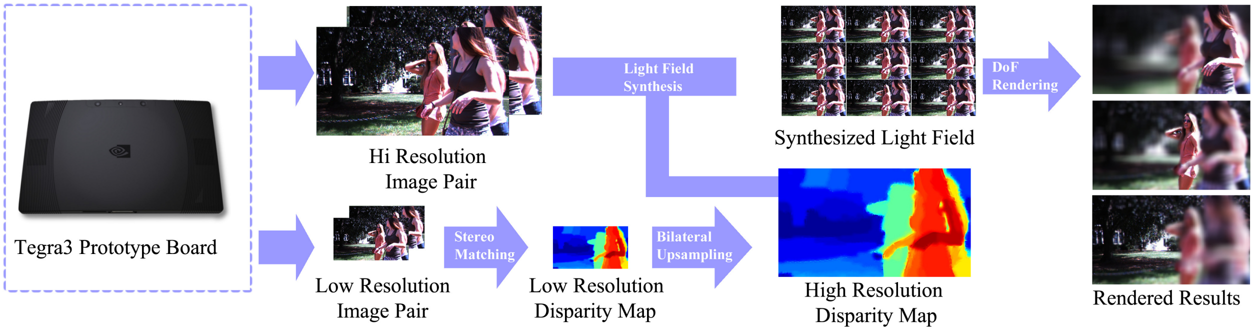


Figure 1. The NVIDIA Tegra3 prototype tablet and the processing pipeline of our software system. All modules are implemented on the Android 4.1 operating system.

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 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.

3. OVERVIEW

In this paper, we demonstrate that DoF effects can be rendered using low-cost stereo vision sensors on mobile devices. Our work is inspired by the algorithms proposed by Yu *et al.*[13,14] 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 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 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 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. 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. The 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[15] 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.[16] We computed the calibration parameters and saved 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. After the rectified images pairs are obtained, 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*.[15] 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

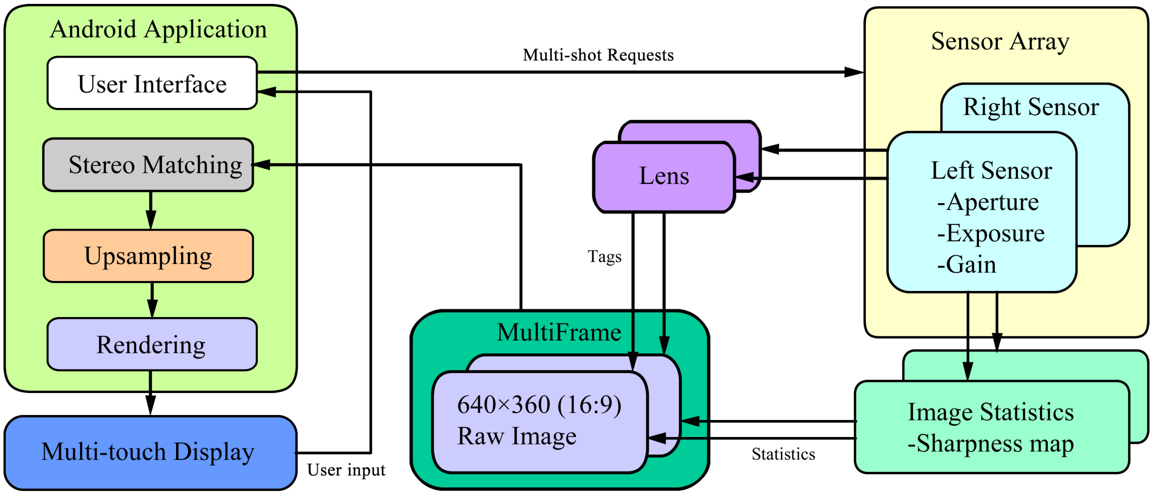


Figure 2. This diagram shows how our application interacts with the camera system. Our application accepts user input from the multi-touch screen, sends multi-shot requests to the sensors with desired parameters and then transfers the raw stereo image pairs to the stereo matching module. We then upsample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

for the camera sensors is being processed, if a wait is found and a condition is not satisfied, the sensor will halt until the condition is satisfied. On the other hand, if a release if 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. 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 the 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[17, 18] are usually fast, but introduces inaccurate boundaries or even bleeding artifacts. Global stereo estimation methods such as Graph Cuts (GC)[19] and Belief Propagation (BP)[20] have shown good results on complex scenes with occlusions, textureless regions and large depth changes.[21] 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.[22]

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:

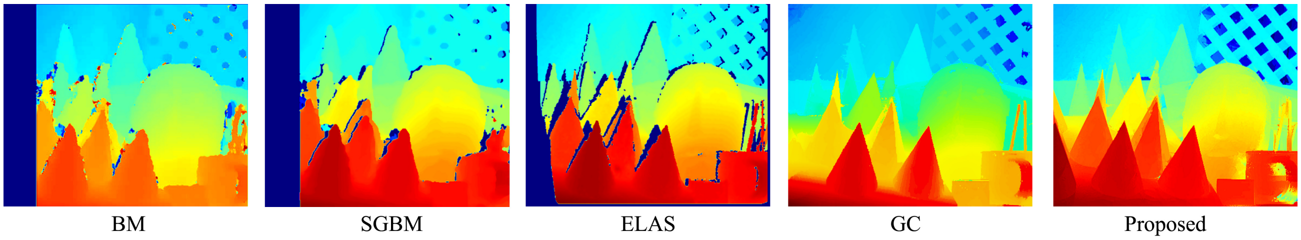
$$E\left(f\right) = arg\min_{f}\left(\sum_{p\in P} D_p\left(f\left(p\right)\right) + \sum_{\{p,q\}\in N} V_{p,q}\left(f\left(p\right), f\left(q\right)\right)\right) \tag{1}$$

The local minimization of function 1 can be efficiently approximated using the alpha-expansion presented in.[19] 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.

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

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),[17] efficient large-scale stereo (ELAS)[18] and Graph Cuts (GC).[19] 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 and ELAS on any given dataset, but requires an adequate choice of window size. Smaller window size may lead to larger bad pixel percentage while bigger window size 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. The depth edges are much more accurate compared to aforementioned methods with very little bleeding artifacts on occlusion boundaries. 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 the other methods which runs faster. Another reason is that we are running the GC algorithm on low-resolution imagery, and typically the running time is only 3 times slower than ELAS, which is not a huge difference. 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.



BM SGBM ELAS GC Proposed

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

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.*[23] 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 [22] *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 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 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. In our case, 5 iterations are sufficient to yield satisfactory results. Generally, it takes around 40 milliseconds to finish the 5 iteration steps on the tablet. Figure 5 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. Compared to other upsampling schemes, our method gives better boundary regions and effectively reduces blurring and aliasing artifacts by using the fine details from the high resolution color image.

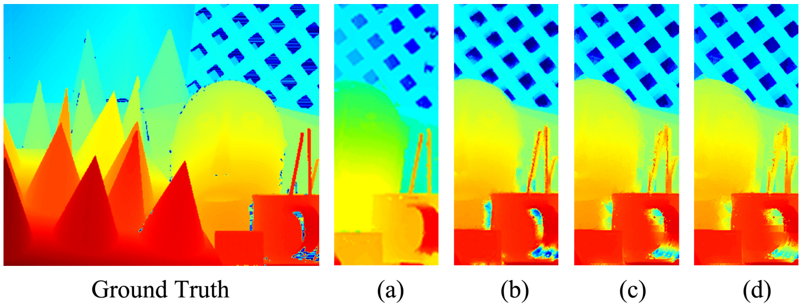


Figure 4. Comparison of results using different number of iterations.(a),(b),(c),(d) are using 0,5,10,20 iterations respectively.

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.[24,25] However, this method suffers

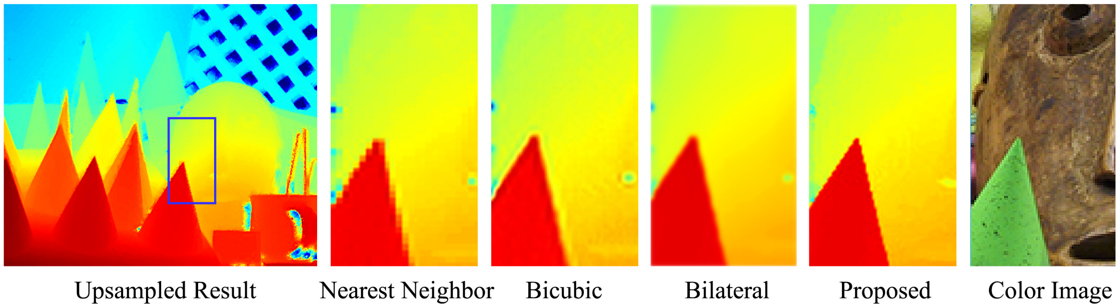


Figure 5. Comparison of our approach and other upsampling algorithms on the Middlebury cones dataset.

from strong intensity leakage and boundary bleeding artifacts. Other methods such as distributed ray tracing[26] and accumulation buffer[27] give more accurate results. However, these methods are computationally expensive and therefore only provide limited frame rates.

In this paper, we use a similar approach to[28] 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 take nearby pixels to fill the holes. This method may bring repeated patterns in the result image, but it is less problematic for DoF rendering since we are only shifting a little bit from the reference image. Figure 6 shows warped views of an indoor scene using the aforementioned warping and hole-filling algorithms.

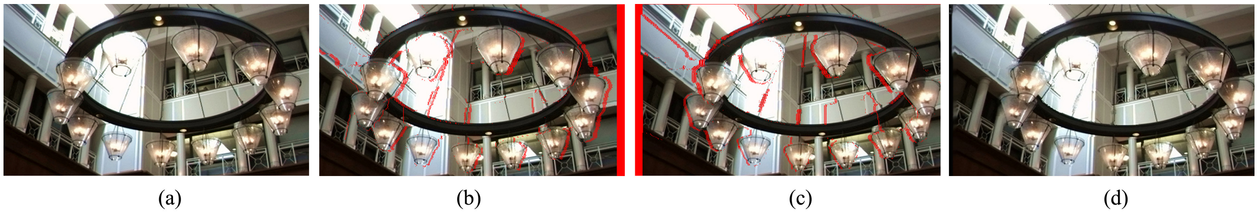


Figure 6. Synthesized light field view, missing pixels are marked in red. (a) input image (b) warped left side view (c) warped right side view (d) result image using our hole-filing algorithm, taking (c) as the input.

Since the image formed by a thin lens is proportional to the irradiance at pixel a,[29] 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 can easily map 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 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:

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

7. RESULTS AND ANALYSIS

We conducted extensive experiments on both indoor and outdoor scenes. Figure 7 shows the results generated by our system under different scene structures and illumination conditions. Row 1 and 2 demonstrate the ability of our system to handling real-time dynamic scenes; Row 3 shows the result on an outdoor scene with strong illumination and shadows; Row 4 displays the result on an indoor scene with transparent and textureless regions. The processing speed of one single frame varies 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. The user can keep taking pictures while the processing takes place in the background. Considering the current performance of major mobile 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 row 1 and 2 of Figure 7. 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 7 Row 1 and 2 show results of two frames from the same video sequence. Since we currently do not have 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 correspondence. Notwithstanding, our algorithm generates fairly accurate disparity maps and gives satisfactory refocusing results. The edges of the girls hand and arm are preserved when they are in focus and objects outside of the focal plane are blurred smoothly.

Figure 7 row 3 displays a scene of two girls 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. This, however, does not affect the overall refocusing result as the cars with similar disparity values are either all in focus (column 4) or blurred (column 3). 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 (column 4). Also, because the girls in row 3 are farther away from the camera compared to girls in row 1 and 2, the boundaries of girls in row 3 are coarser and fine details on the bodies are lost. Therefore, foregrounds in row 3 are more uniformly blurred compared to row 1 and 2. However, to an average user, these pictures are not visually different on a 10 inch screen.

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 wont cause problems such as over-exposure or shadows. Row 4 of Figure 7 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 (column 1). 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 column 4, 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.

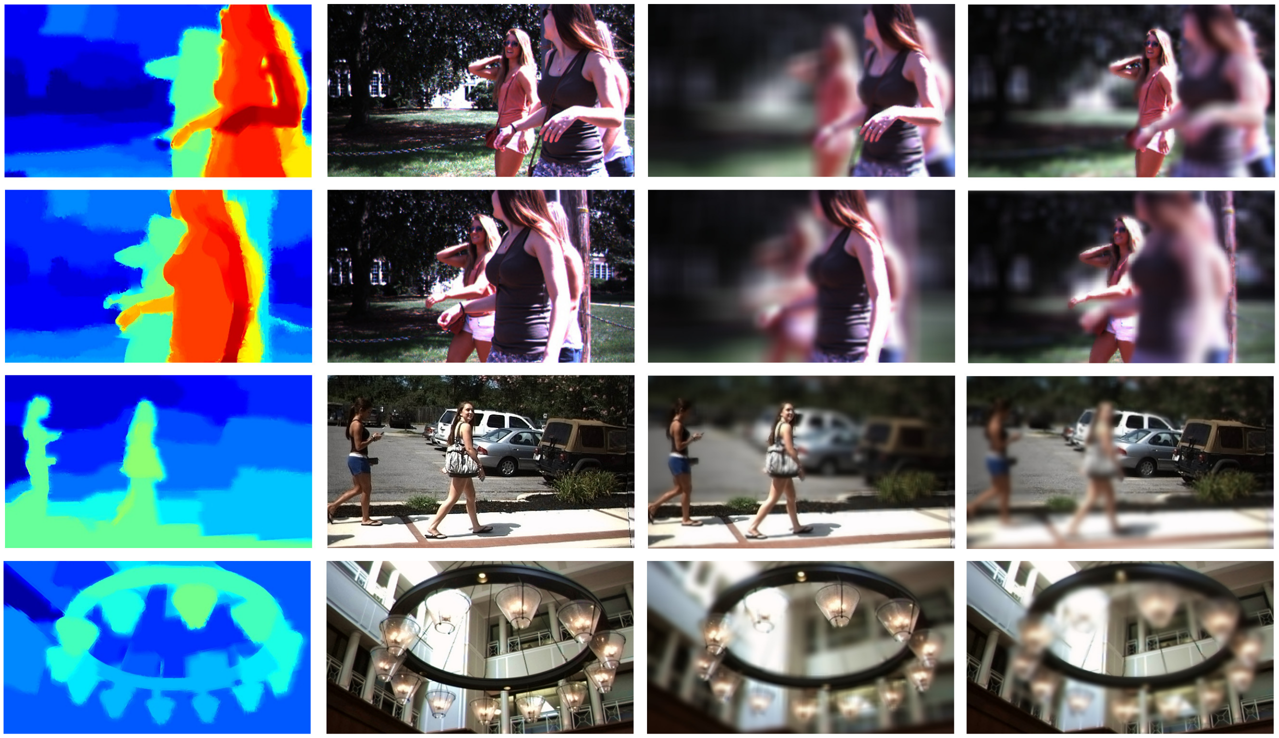


Figure 7. Input disparity map and rendered images of our system. Column 1: input high resolution disparity map. Column 2: reference view. Column 3,4: rendered DoF effects.

Our analysis illustrates that, of the four cases, our approach produces high quality disparity maps and generates promising dynamic DoF effects. Also, our system correctly handles complex scene structures with real world illumination conditions. Last but not least, we effectively reduce aliasing artifacts in out-of-focus regions by blending multiple synthesized light field views together.

8. CONCLUSION

We have presented an affordable solution for producing dynamic DoF effects on mobile devices. We compare the performance of popular stereo matching algorithms and design a hybrid resolution approach to speed up the matching process without losing too much quality. Also, we generate the synthesized light field by using the disparity warping scheme and effectively 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. We also plan to adapt our software to mobile devices such as LG Optimus 3D smartphone and Nintendo 3DS portable game console, as well as commercial point and shoot stereo cameras such as Sony Bloggie 3D or Fujifilm FinePix Real 3D. Finally, we would 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.," *Proceedings of Eurographics Symposium on Rendering* **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," *Proceedings of Computer Graphics International (CGI)* (2011).

[15] 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).

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

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

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

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

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

[21] 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).

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

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

[24] 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).

[25] 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).

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

[27] 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).

[28] 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).

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