DOI: 10.2352/ISSN.2470-1173.2016.18.DPMI-031

Virtual DSLR: High Quality Dynamic Depth-of-Field Synthesis on Mobile Platforms

Yang Yang [1], *Haiting Lin* [1], *Zhan Yu* [2], *Sylvain Paris* [2], *and Jingyi Yu* [1]
[1] *University of Delaware, Newark, DE 19716, USA*
[2] *Adobe Systems Inc., San Jose, CA 95110, USA*

Abstract

Shallow depth-of-field (DoF) and a smooth bokeh are signature elements of Digital SLR cameras and high-quality lenses. Producing comparable effects on mobile platforms has long been challenging due to small sensor sizes and short focal lengths of mobile cameras. In this paper, we exploit depth sensing capabilities on emerging mobile devices and develop a new depth-guided refocus synthesis technique particularly tailored for mobile devices. Our technique takes coarse depth maps as inputs and applies novel depth-aware pseudo ray tracing. The depth maps can be obtained from mobile depth sensors, mobile stereo cameras and even from user-inputs. Our pseudo ray tracing scheme resembles light field synthesis and refocusing [21] but does not require actual creation of the light field and hence reduces both memory and computational overhead. At the same time, the scheme can overcome boundary bleeding and discontinuity artifacts observed in previous filtering techniques. Comprehensive experiments show that our approach can produce very high quality DoF comparable to the ones produced by DSLR and the Lytro light field cameras.

Introduction

Shallow depth of field (DoF) is a signature element in professional photography. It is used to effectively emphasize objects of interest while de-emphasizing the rest of the image via defocus blurs. For long, high quality DoF (smooth Bokeh, sharp boundary of in-focus objects, correct blurs of out-of-focus regions, etc) has been a luxury of high-end Digital SLR cameras with superior quality lenses.

The basic physics of DoF is well studied. Through a thin lens, a scene point p with depth z_p will project to a circular region on the sensor. Assuming that the thin lens has a focal length f, an aperture size D, and the sensor to lens distance s, the diameter c_p of the circular region is:

$$c_p = \frac{|s - s_p|}{s_p} D = sD\left|\frac{1}{z_p} - \frac{1}{z_s}\right|, \tag{1}$$

where $z_s = (1/f - 1/s)^{-1}$ and $s_p = (1/f - 1/z_p)^{-1}$ according to the thin lens law. The diameter c_p measures the size of blurs and it is linear to the absolute difference of the reciprocal distances $|1/z_p - 1/z_s|$.

While classical DSLR produces fixed-focus DoF, the emerging light field (LF) cameras such as Lytro provide post-capture refocusing capabilities. In a nutshell, a LF camera records the 4D ray space of the scene (2D spatial and 2D angular dimensions). These rays can be reassembled and integrated to produce desirable focus effects [5, 10, 14]. Existing LF cameras are based on the Lippmann design: it mounts a lenslet array in front of the sensor to multiplex the incident rays. Consequently, it has to trade between the spatial and angular resolutions. To avoid aliasing, the angular resolution needs to sufficient high. This leads to a low spatial resolution of the final refocused image.

Instead of capturing DoF, computer graphics techniques have been focused on synthesizing DoF through ray tracing, e.g., via a distributed ray tracer [1]. To ensure rendering quality, schemes in this category need to trace out a large number of rays for rendering each pixel. Early approaches such as the accumulation buffer [4] divide the lens aperture into sub-apertures and render each sub-aperture view through rasterization. Such techniques require rendering scenes from many different viewpoints hence place heavy work load on graphic rendering pipeline. More recent approaches directly filter a single view image [2, 8, 11] with spatially varying blur sizes based on the depths of the pixels according to Eq. 1. As shown in Fig. 8, these approaches are very fast but suffer from intensity bleeding and discontinuity because the blurring can go across (for background pixels) or be confined within (for foreground pixels) depth boundaries.

Most recent efforts attempt to combine ray-tracing with light field synthesis, by exploiting high-performance parallel computation on the GPU [7]. They first simplify the scene by decomposing the geometry into discrete image layers according to their depths and then perform ray tracing or filtering on the layers. [17]first synthesize a light field (an array of virtual views) from an in-focus image and its depth map and then apply ray integral. These approaches can produce very high quality DoF effects comparable to distributed ray tracing but at an interactive speed. However, such schemes either require special graphics hardware for performing specialized operations or a large texture memory for restoring intermediate results (e.g., the light field).

Our focus in this paper is to develop a feasible solution that can produce high quality DoF on mobile platforms. Compared with previous solutions, our solution is tailored to handle low computational power and low memory size of mobile devices and at the same time it exploits coarse depth sensing capabilities on emerging mobile platforms. Specifically, our technique takes coarse depth maps obtained through depth sensors (e.g., Google Project Tango), stereo sensor (e.g., HTC one smart phone), or even user drawn strokes as inputs. We then develop a novel depth-aware pseudo ray-tracing scheme that emulates the light field refocusing process while avoiding actual creation of the light field. Such a scheme does not require high-end graphics card or large memory sizes but still achieves high quality refocus rendering that avoids boundary bleeding and discontinuity commonly observed in previous filtering methods.

DOI: 10.2352/ISSN.2470-1173.2016.18.DPMI-031

In addition to providing pseudo adjustment to aperture shape and size, our refocus rendering solution can be further used to produce effects beyond regular DoF in a DSLR. For example, we can produce the tilt-shift effect, a unique feature of perspective control lens (or tilt-shift lens), through adjustment of the depth maps. Our technique can also produce optically impossible DoF by rendering multiple, non-adjacent depth layers in-focus. Finally, we can dynamically change the shape of the virtual aperture to produce customized Bokeh. We compare our techniques to commercial software and hardware solutions and comprehensive experiments show that our results outperform state-of-the-art software solutions and are comparable to the results from DSLR and light field cameras.

Depth-Guide Pseudo Ray Tracing

Traditional distributed ray tracing requires the complete geometric models. In contrast, our pseudo ray tracing technique uses an all-in-focus pinhole image and its depth map as input, where the depth map can be obtained in various ways on mobile platforms as shown in Section . In this section, we first elucidate and analyze brute-force image space ray tracing. Based on our analysis, we show how to improve its performance by employing statistical priors and then propose a new approximation scheme that is robust even with complex scene geometry.

Image Space Distributed Ray Tracing

As shown in Fig. 1(a), the image space ray tracing uses the input of a pinhole image and its depth map. We assume that the depth range of the scene is $[z_0, z_1]$ $(z_1 > z_0 > 0)$[1] and the sensor to pinhole distance is s. We render the DoF effects as if there were a virtual thin lens located at the pinhole position as shown in Fig. 1(b). We assume the virtual lens has a the focal length of f that satisfies $\frac{1}{f} = \frac{1}{s} + \frac{1}{z_1}$, *i.e.* the farthest plane z_1 is focused on the sensor. Further, we assume that the radius of the aperture is R. Under this virtual lens system, points on plane $z \in [z_0, z_1]$ will project to blur discs whose radius (i.e., the kernel size) is:

$$k = sR(\frac{1}{z} - \frac{1}{z_1}) \tag{2}$$

Recall that the largest kernel size on the senor $K = sR(\frac{1}{z_0} - \frac{1}{z_1})$ corresponds to plane z_0. The largest kernel size is also related to the intersection disc of the blue cone of rays (the cone that converges to a single point on the sensor) with plane z_0. Under the similitude relationship, the radius of this intersection disc can be computed $R\frac{z_1 - z_0}{z_1}$ or alternatively $\frac{z_0}{s}K$ as shown in Fig. 1(b).

Under this system setup, we can conduct dynamic refocusing by rendering an image on a virtual sensor plane at s_f that corresponds to focal plane z_f ($\frac{1}{s_f} + \frac{1}{z_f} = \frac{1}{f}$). As shown in Fig. 1(c), we use pixel indices in the original pinhole image as the central ray indices of the ray cones. This guarantees that the rendered refocused image will have an identical resolution of the input pinhole image and pixels at the same position in both images correspond to each other.

In the defocus result, for each pixel p, its final color will be the integration of all incident rays within a cone. In regular distributed ray tracing, we need to trace each ray into the scene. In image-space ray tracing, we instead seek to assign a color to each ray as some pixel from the input image. Geometrically, this can be done by first tracing out the ray into the scene as a 3D point and back trace the 3D point to the input pinhole camera. Under the Lambertian assumption, we can directly use its corresponding pixel in the input camera to assign the color of the ray.

Apparently to determine the ray color, it is essential to locate the intersection point of the ray with the scene. In the light field based approach [18], this is achieved via forward warping: the central view uses its depth map to warp to the neighboring light field views. A major disadvantage of such an approach is that, to ensure rays are densely sampled, one needs to virtually create/render a very large number of views and therefore the scheme requires multiple passes of rendering and large texture memory if it is conducted on the GPU.

Alternatively, we can determine the intersection by discretizing the path along the ray. Specifically, our goal is to determine, at each possible depth z along the ray, if its corresponding 3D point lies in either empty space or on an actual surface. This can be achieved by backprojecting the point to the pinhole view as a pixel and verify if the pixel's depth matches z. Assume that the ray "stops" at a signed distance r to the aperture center, for a point Q on the ray with its depth $z_Q \in [z_0, z_1]$, the signed distance between Q and the intersection point of the center ray with plane z_Q can be computed as $r\frac{z_Q - z_f}{z_f}$ (similar as the above example computing the largest kernel size). Under the similitude relationship, the projection q of Q will be at distance l_{pq} to the current rendering pixel p where $l_{pq} = \frac{s}{z_Q} \cdot r\frac{z_Q - z_f}{z_f} = sr(\frac{1}{z_f} - \frac{1}{z_Q})$.

Since the depth range of the scene is $[z_0, z_1]$, the ray segment inside the depth range will project to a line segment $[q_0, q_1]$ around p, where q_0, q_1 are the projections of Q_0, Q_1 with depth z_0 and z_1 respectively. In image space distributed ray tracing, we need to search for the front most intersection of the ray with the scene by intersection checking for each pixel $q \in [q_0, q_1]$ starting from q_0. If pixel q corresponds to a true intersection point, it should satisfy the intersection condition:

$$l_{pq} = sr(\frac{1}{z_f} - \frac{1}{z_q}), \tag{3}$$

where the depth z_q of pixel q is obtained from the depth input.

The analysis above is carried out in terms of depth. We can alternative map depth to disparity as $d = \frac{1}{z}$ and conduct a similar derivation. Specifically, we normalize the disparity range of the scene to $[0, 1]$, *i.e.* where 0 disparity corresponds to depth at ∞ and 1 corresponds to disparity at the nearest depth. If we directly use an aperture mask of size $K(= sR)$, we can simply the the intersection condition as:

$$l_{pq} = \frac{r}{R}K(d_f - d_q), \tag{4}$$

where $d_f = \frac{1}{z_f}$ and $d_q = \frac{1}{z_q}$. Since the left and right sides of Eq. 4 can be directly verified on the disparity map, our image space distributed ray tracing avoids reconstructing the scene geometry or warping the central view to a light field.

In fact, we can calculate the computational complexity of approach as $O(NMK)$ where N is the total number of pixels of the input image, M is the pixel number of the aperture mask (M corresponds to the area of the aperture thus has the scale of K^2),

[1] The maximum scene depth z_1 can be infinite.

DOI: 10.2352/ISSN.2470-1173.2016.18.DPMI-031

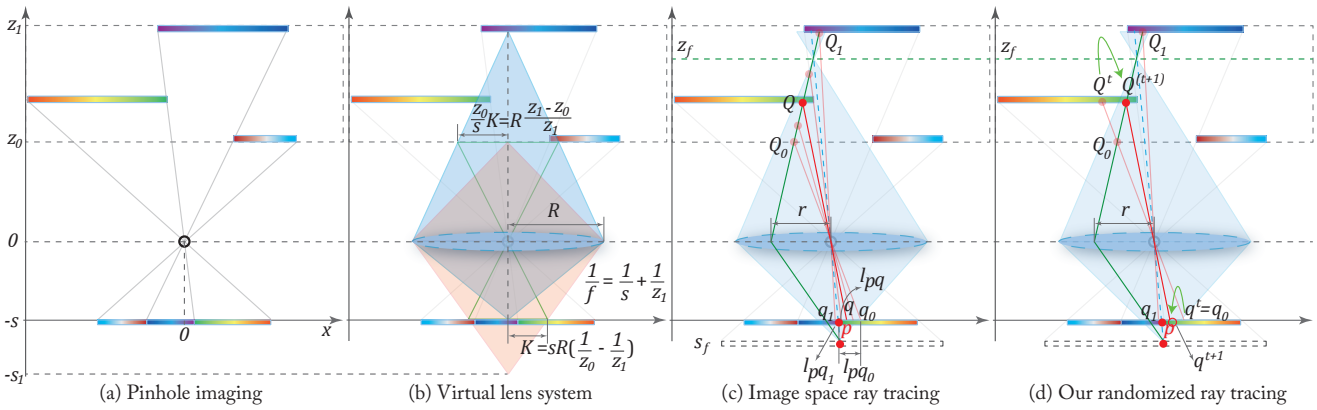
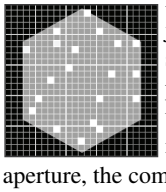


Figure 1: Image space distributed ray tracing and our method. (a) Pinhole imaging. (b) Virtual lens system for DoF rendering. (c) Image space ray tracing searches for the intersection point in the pixels on line segment $[l_{pq_0}, l_{pq_1}]$. (d) Our randomized intersection point searching method that utilizes smooth surface prior in real scenes.

and K accounts for intersection checking. Different from standard ray tracing on 3D models, the image space distributed ray tracing performed on image space suffers from two major problems. First, our scheme assumes that the xy dimensions of the scene (both the color image and the depth map) were discretized into pixels. Therefore the satisfaction of the intersection condition cannot ensure true intersection. To alleviate this problem, we relax the intersection condition to $|l_{pq} - \frac{r}{R}K(d_f - d_q)| \leq 0.5$, *i.e.,* within the size of a half pixel. Second, since we only use a single input image, we cannot determine the content (color or geometry) in the occluded regions. Our solution is to assume the surface (geometry and texture) is symmetric around the pixel of interest (being rendered), *i.e.,* when pixel at l_{pq} is occluded, we use pixel at $-l_{pq}$.

Randomized Intersection Searching

Image space distributed ray tracing is nearly as accurate as traditional geometry space ray tracing. However, it is also computationally expensive especially for the application of refocusing. Recall that M has the scale of K^2, therefore, the cost increases cubically the aperture radius ($O(NMK) = O(NK^3)$). For a large aperture, conducting complete distributed ray tracing is too expensive to be applicable to mobile devices. We present two schemes to reduce the computational cost. First, we reduce the number of ray samples through stochastic sampling methods such as jittered sampling for reducing M.



Instead of tracing every position of the aperture, jittered sampling first divides the aperture into several equalized grids, and then randomly samples one ray in each grid as shown in the left figure. Assume that the dimension of the grid array is a times less than the original dimension of the aperture, the computational cost is then reduced to $O(NMK/a^2)$.

While jittered sampling guarantees correct intersection computation for each ray sample, the sparsity and randomness of the sampling usually produces severe visual artifacts, especially for scenes of high contrast and variant colors. Since only a sparse set of rays are sampled, those rays should be representative for other unsampled rays in their grids. Several challenges, however, remain. First, rays in a grid may not be representable by a single sample. Second, random sampling cannot guarantee the quality/representativeness of the sampled ray. These two factors usually result in inconsistent blurs between neighboring pixels and incurs noise-type artifacts as shown in Fig. 8.

A more plausible approach is to simplify the intersection searching process, *i.e.*, to reduce K. We make use of the smooth surface assumption of real scenes to speed up intersection searching. When searching for the intersection point, instead of iterating pixel $q \in [q_0, q_1]$ and checking the intersection condition, we start from fixing d_q and directly compute the required l_{pq} using Eq. 4. Specifically, as illustrated in Fig. 1(d), for a ray sample at radius r, we initialize a d_q^t (superscript t indicates the iteration number) to be the depth of the pixel at distance l_{pq_0} to the current center pixel p. Given disparity d_q^t, we compute the distance l_{pq}^t. If the disparity of the pixel at distance l_{pq}^t equals to d_q^t, then we deem that the intersection has been found. Otherwise, we set $d_q^{(t+1)}$ to be the disparity of the pixel just located. The validation process repeats until the number of tests reaches a predefined threshold T. When terminates, the algorithm returns the color of the last pixel it has checked.

Since in real scenes, surfaces are usually continuous and smooth, the chance that nearby pixels having the same depth measure are very high. In other words, it is very likely that d_q^t equals to $d_q^{(t+1)}$ which results in early termination in intersection searching. This scheme can be alternatively interpreted as randomized intersection searching. The main limitation of this method is that it does not guarantee that we can find the correct (front most) or any intersection for a single ray sample. The returned pixel may not be a valid intersection. However, the chances of error is very low since one error pixel returned for a certain ray sample may be a valid intersection point for another ray sample and belongs to the overall set of the intersections. In other words, an invalid returned pixel does not necessarily introduce to the set an "outlier" sample: it still highly likely belongs to the set of intersections which should be counted towards the final blending for defocus rendering. To distinguish those samples from "outlier" samples, we call them "in-set" ray samples.

DOI: 10.2352/ISSN.2470-1173.2016.18.DPMI-031

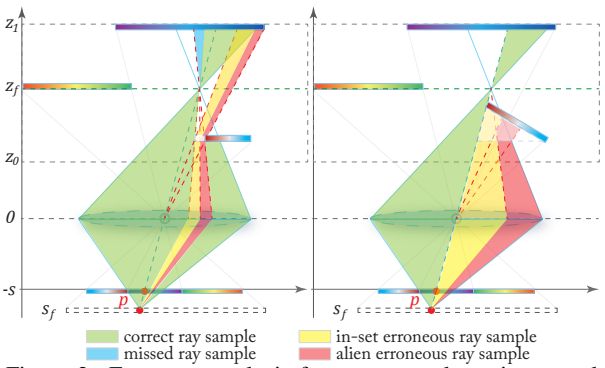
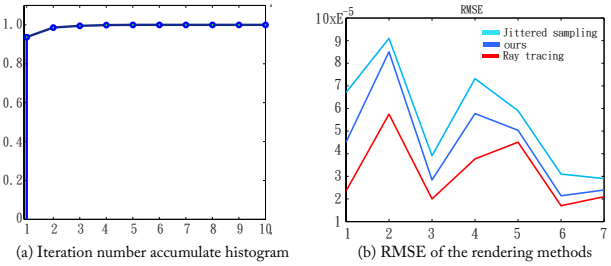


Figure 2: Error rate analysis for two examples using our algorithm. Error rate will be less regarding to the ray set (red only) than regarding to an individual ray (both red and yellow).



(a) Iteration number accumulate histogram (b) RMSE of the rendering methods

Figure 3: Error rate analysis.

To further handle bleeding from the background to a focused foreground object, when the returned pixel q is an invalid intersection and q is further away compared to the current center pixel p, *i.e.* $d_p > d_q$, we replace the returned invalid sample q with p. [2] In sum, even though the proposed randomized intersection searching algorithm has the limitation of returning erroneous samples, fusing the pixels from all ray samples still produces plausible and accurate DoF effects.

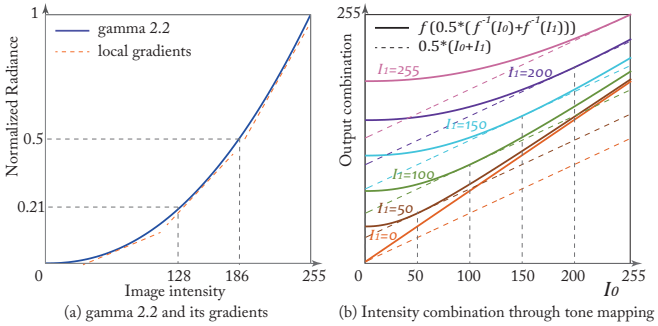
For the validation number threshold T, we choose $T = 1$ to significantly reduce the computational cost since our goal is to render on mobile devices. As a result, the computational cost reduces to $O(NM)$. In all our experiments, $T = 1$ is sufficient to produce high quality refocusing effects.

Fig. 2 illustrates two examples of the ray sampling results using our algorithm. The correct and erroneous ray samples are distinguished using different color shading. Recall that even though individual pixel-ray indexing may be wrong, the incorrectly fetched pixel will still correspond to some ray within the blur kernel, the overall error rate with respect to all rays within the kernel is very low.

Fig. 3(a) plots the iteration histogram, where the value of bin i shows the total number of pixels that find their correct samples corresponding to rays falling inside the aperture within i iterations. The histogram is computed based on several example images with their depth maps. From the histogram, it is clear that $T = 1$ is generally sufficient to achieve a low error rate.

Fig. 3 (b) plots the root mean square error (RMSE) of the rendered results from our algorithm, jittered sampling, and full ray tracing using the center image and its disparity map from each light field data set. The ground truth results are rendered using the complete light fields by integral photography. The plot shows that our algorithm achieves comparable quality to distributed ray tracing.

[2] The current center pixel p is guaranteed to belong to the set of intersections since it is the intersection point of the center ray with the scene.



(a) gamma 2.2 and its gradients (b) Intensity combination through tone mapping

Figure 4: The effects of response function in intensity fusing. (a) Gamma 2.2. (b) Comparison between intensity combination through tone mapping and linear combination on two intensities I_0 and I_1. Different color corresponds to different I_1 intensity value.

Weighted Color Blending

In real camera photography, the transformation from linear scene radiance J to the digital image intensity I is highly nonlinear due to high dynamic range compression and on-board color processing.This nonlinear transformation is usually modeled as response function f: $I = f(J)$. An SLR camera integrates radiance instead of intensity on the sensor. Ideally, correct DoF synthesis should be conducted on radiance $J = f^{-1}(I)$ and produce the final rendering I' as $I'(p) = g_p(I) = f(\frac{1}{|\Omega_p|}\sum_{q\in\Omega_p} f^{-1}(I(q)))$, where p,q index the pixels in the image, Ω_p denotes the blur kernel centered at p and $|\Omega_p|$ computes its cardinality. We use $g_p(I)$ to denote the overall transformation. However there are two obstacles that prevent the direct application of the transformation $g_p(I)$. First, the response function for the input image is usually unknown. As common practice, one may use gamma 2.2 as approximation [9]. But the camera response function can be very different from gamma 2.2 function [6]. Second, a single response function is inadequate to model the color processing performed on-board. Due to the aforementioned two main reasons, the Gamma 2.2 approximation results in color shift as shown in Fig. 5. In order to avoid the color shifting and at the same time mitigate nonlinearity artifacts, we adopt a weighted blending scheme that directly integrates pixels from image I with content aware weighting.

We first examine the property of the overall transformation function $g_p(I)$ using gamma 2.2 function as an example. As shown in Fig. 4(a), the gradients within high intensity regions are steeper than the ones in low intensity regions. This indicates that high radiance is compressed much more than low radiance. Compared to directly integrate on intensity, when converted back to radiance, the integrated radiance that combines low and high radiances will be closer to the higher radiance as illustrated in the example Fig. 4(b). In other words, the overall effect of $g_p(I)$ is to give higher intensity a larger weight. Thus we adopt the following

DOI: 10.2352/ISSN.2470-1173.2016.18.DPMI-031

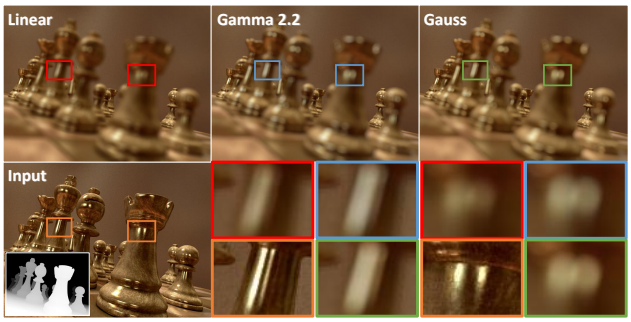


Figure 5: DoF results using different fusion methods with a circular aperture: direct linear blending, gamma 2.2 based blending and our gaussian weighted color blending scheme.

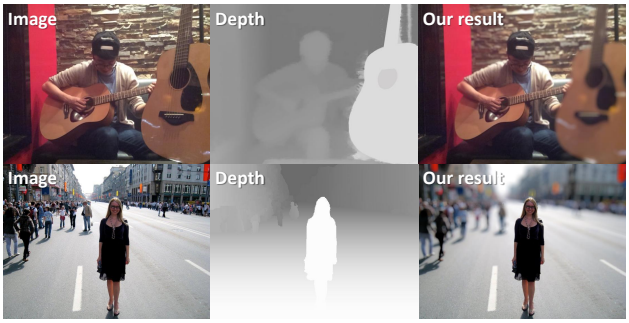


Figure 6: Depth acquisition methods: first row, the depth sensor; second row, single image depth learning.

blending scheme:

$$I'(p) = \frac{1}{W}\sum_{q\in K_p} w_q I(q), \quad w_q = \exp(-\frac{(I(q)^{max}-255)^2}{2\sigma^2}), \tag{5}$$

where $I(q)^{max} = \max\{I_r(q), I_g(q), I_b(q)\}$ is the max intensity among all the color channels, the normalization term $W = \sum_{q\in K_p} w_q$ and σ is a user defined parameter to control the weighting behavior. Fig. 5 compares different blending methods. While linear blending that directly averages the intensities dims the high lights and gamma 2.2 based blending $g_p(I)$ makes the color bluish, our proposed gaussian weighted color blending renders a more realistic results.

Depth Maps as Inputs

Depth sensor The depth sensing capabilities are becoming popular on emerging smart phones and tablets. For example, Google Project Tango [3], HTC One [22] are exploring integration with time-of-flight depth sensors. Structure company [15] designs infrared sensors attachable to iPad's. As a prototype, we use a structure sensor which captures color image of resolution 640×480 and an infrared depth image of resolution 320×240.

We adopt the similar scheme in [17] to upsample the depth map. The low resolution depth map can be upsampled to high resolution using bilateral up sampling while preserve the boundary discontinuity. We show an example of our depth sensor result in the first row of Fig. 6.

Single image depth learning There is also an emergence of techniques that recover depth from a single color image through machine learning [12, 13]. Variant monocular depth cues from a pinhole image are explored including the scale, ground plane, vanishing points. Especially combined with object detection, we can get a depth map with clear boundary from single image for our application. Specifically, we adopt the method from [16] for its edge preservation ability. We show an example of single image depth learning result in the second row of Fig. 6.

Depth generation interface When no depth sensor is available or the learning method fails, we use a depth generation interface for user to draw depth map efficiently, taking the advantages of modern multi-touch screens equipped in most mobile devices. Basically, users use selection and (gradient or solid) filling tools for depth map generation. The most important aspect of the depth map in DoF rendering we want to assure is clear sharp boundary. As shown in Fig. 7, we provide stroke based selection tool which adopts paint selection technique for quick and clear object selection. Most scenes have ground plane whose depth can be generated using gradient filling. This ground plane can be first paint as background depth layer. Users only need to assign several depth layers for objects placed on/above the ground plane. While the user move his/her finger on the image, the object marked by the trajectory is selected with clear boundary. The user then can fill the selection with a suitable depth value.

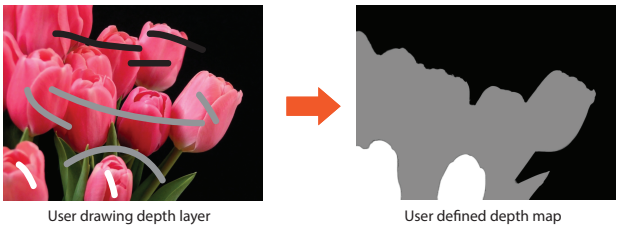


Figure 7: Stroke based user depth generation.

Experimental Results

Our algorithm is tested on an iPad mini with 1.3Ghz A7 dual core CPU, 1GB memory, and a PowerVR G6430 GPU. We conduct variant experiments to show the running complexity, rendering quality of our algorithm compared to full ray tracing, jittered sampling, LF rendering [17, 20] and image space filtering [11] techniques. We also show the super flexibility of our algorithm in rendering images. It is very convenient in our rendering framework to control settings such as aperture size and shape, focusing regions. Using our algorithm and tools, we can easily generate special effects such as tilt-and-shift, artificial focusing.

Performance Table 1 summarizes the complexities and the running time of all the algorithms. Our proposed algorithm has the complexity same as that of the most efficient image space filtering technique, and produces high quality DoF comparable with the most accurate ray tracing technique. Recall that LF rendering technique [17,20] utilizes efficient rasterization process in graphic pipeline to render multiple views of the scene. It is hence faster than ray tracing, 5 fps vs. 3 fps, to produce a similar rendering quality. The memory usage of LF rendering can be further reduced to $O(N)$ if we accumulate the intermediate views directly

©2016 Society for Imaging Science and Technology
DOI: 10.2352/ISSN.2470-1173.2016.18.DPMI-031

Algorithms	Comp. Cost	Memory Cost	Running time (fps)
Image space blurring	$O(NM)$	$O(N)$	15
LF rendering	$O(NM)$	$O(NM)$	5
Ray-tracing	$O(NMR)$	$O(N)$	3
Jittered sampling	$O(NMR/a^2)$	$O(N)$	15
Ours rand. searching	$O(NM)$	$O(N)$	15

Complexity and Running time analysis. Where N: number of pixels of result; M: number of pixels of the aperture mask; a: constant reduction scale; R: aperture radius. Typical resolutions of the image and the aperture mask for the evaluation is 720×540 and 41×41 respectively.

to the output buffer. However, it needs to fill in the holes in the rendered view due to depth warping. In contrast, our method automatically fills the missing information through symmetric assumption. Furthermore, from table 1, we observe LF rendering is less efficient than image space blurring and our method. This is because the implementation requires multiple pass rendering which becomes a severe bottleneck. In contrast, our method is a single pass algorithm.

Rendering Quality Fig. 8 compares the rendering results of various algorithms (full ray tracing, jittered sampling, LF rendering [17, 20] and image space filtering [11]). We observe that our rendering quality is comparable to the full ray tracing and the LF rendering, while jittered sampling results exhibit noise-type artifacts and image space filtering results exhibit boundary bleeding and discontinuity artifacts.

Next, we compare our method with real cameras. We demonstrate that our method produces a high quality DoF effect comparable to expensive DSLR cameras and LF cameras. Fig. 9 compares our proposed algorithm with images captured by a Canon 60D with lens 3.5 and the latest LF camera Lytro illum. Compared to a DSLR camera which can only achieve fixed focus, our method can conduct post-capture refocus same as the LF camera (Fig. 10). The major difference though is that we only use a mobile camera which is significantly cheaper than the Lytro camera and at the same time we can achieve very high resolution rendering at a high speed. Our technique also offers a richer set of effects as shown in Fig. 11 and Fig. 13.

Figure 10: Dynamic refocusing comparison between the Lytro illum results (first row) and our method (second row).

Bokeh Effect The weighted fusing technique is especially demanding in Bokeh effect rendering where light sparks are of significantly higher radiance than their surroundings. The parameter in weighted fusing scheme is adjustable to further refine the Bokeh effects. We compare different σ values in Fig. 12. With smaller σ, the Bokeh becomes more prominent.

Figure 11: Rendering results with different aperture shapes indicated on the up right corners.

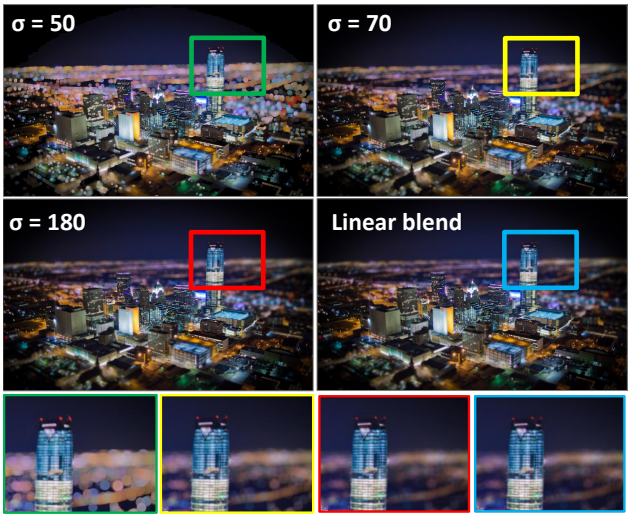


Figure 12: Comparing the rendering Bokeh results with different σ.

Special effects We further show the super flexibility of our method through special effects rendering. Tilt-shift photography is a technique for selective focus and is often used for simulating a miniature scene. In real cameras, special lens are required to achieve such effects. Tilt makes the lens plane forms an angle to the image plane which results in same effects of depth compression. Our algorithm can easily render Tilt-and-shift effects by modifying the depth input. Fig. 13 (a) shows an example where the depth range is compressed along a user specified line.

We also show examples contains artistic focus effect which cannot be achieved using a DSLR camera or a LF camera. Regular cameras can only focus at one depth layer. In our framework, we can simultaneously focus on the objects at different depth layers by assigning them with same disparity values. In this way, we can make some depth based illusions more realistic. Fig. 13(b)

DOI: 10.2352/ISSN.2470-1173.2016.18.DPMI-031

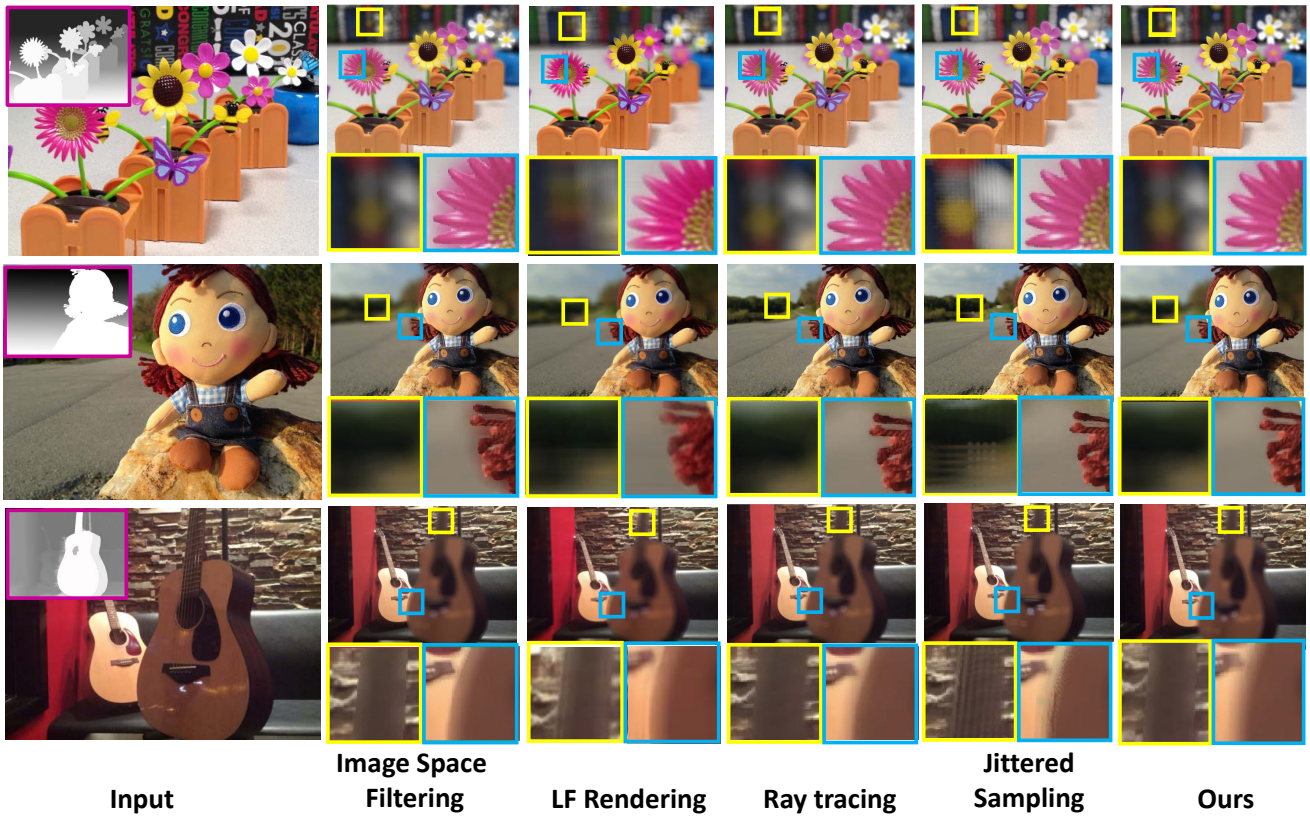


Figure 8: DoF comparisons between different rendering methods.

and (c) shows such examples. Actually, we can focus at any regions by providing depth maps regardless We show an example of center focusing for visual attraction in Fig. 13(d).

Conclusions

In this paper, we have proposed a computation and memory efficient DoF rendering algorithm suitable for mobile devices. Our approach uses an all-focus image and its depth map as inputs and can produce a wide range of high quality DoF effects. We exploit the depth sensing capabilities on emerging smart phones and tablets and the multi-touch interface for continent depth map acquisition. Compared with other software solution, our depth-guided pseudo ray tracing is as efficient as image space blurs but can achieve quality comparable to distributed ray tracing. Compared to real SLR and LF cameras, our solution is based on mobile cameras which are significantly cheaper. In addition to regular DoF rendering, we have demonstrated our technique to produce challenging or physically impossible effects.

References

[1] Cook R. L., Porter T., Carpenter L.: Distributed ray tracing. In *SIGGRAPH* (1984), ACM.

[2] Earl Hammon J.: Practical post-process depth of field. In *GPU Gems 3*, Nguyen H., (Ed.). Addison-Wesley, 2008, pp. 583–605.

[3] Project tango. https://developers.google.com/project-tango/.

[4] Haeberli P., Akeley K.: The accumulation buffer: Hardware support for high-quality rendering. In *SIGGRAPH* (1990), ACM.

[5] Isaksen A., McMillan L., Gortler S. J.: Dynamically reparameterized light fields. In *SIGGRAPH* (2000), ACM.

[6] Kim S. J., Lin H. T., Lu Z., Susstrunk S., Lin S., Brown M. S.: A new in-camera imaging model for color computer vision and its application. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (2012).

[7] Lee S., Eisemann E., Seidel H.-P.: Real-time lens blur effects and focus control. *ACM Trans. Graph.* (2010).

[8] Lee S., Kim G. J., Choi S.: Real-time depth-of-field rendering using anisotropically filtered mipmap interpolation. *IEEE Transactions on Visualization and Computer Graphics* (2009).

[9] Lin H., Kim S. J., Susstrunk S., Brown M. S.: Revisiting radiometric calibration for color computer vision. In *Computer Vision (ICCV), 2011 IEEE International Conference on* (2011), IEEE.

[10] Ng R.: Fourier slice photography. In *SIGGRAPH* (2005), ACM.

[11] Rokita P.: Real-time Depth of Field Rendering via Dynamic Light Field Generation and Filtering. *Computer & Graphics* (1993).

[12] Saxena A., Chung S. H., Ng A. Y.: Learning depth from single monocular images. In *NIPS* (2005), MIT Press.

[13] Saxena A., Chung S. H., Ng A. Y.: 3dd depth reconstruction from a single still image. *Int. J. Comput. Vision* (2008).

[14] Soler C., Subr K., Durand F., Holzschuch N., Sillion F.: Fourier depth of field. *ACM Trans. Graph.* (2009).

[15] Structure sensor. http://structure.io/.

[16] Wang P., Shen X., Lin Z., Cohen S., Price B., Yuille A.: Towards unified depth and semantic prediction from a single image. In *CVPR* (2015).

DOI: 10.2352/ISSN.2470-1173.2016.18.DPMI-031

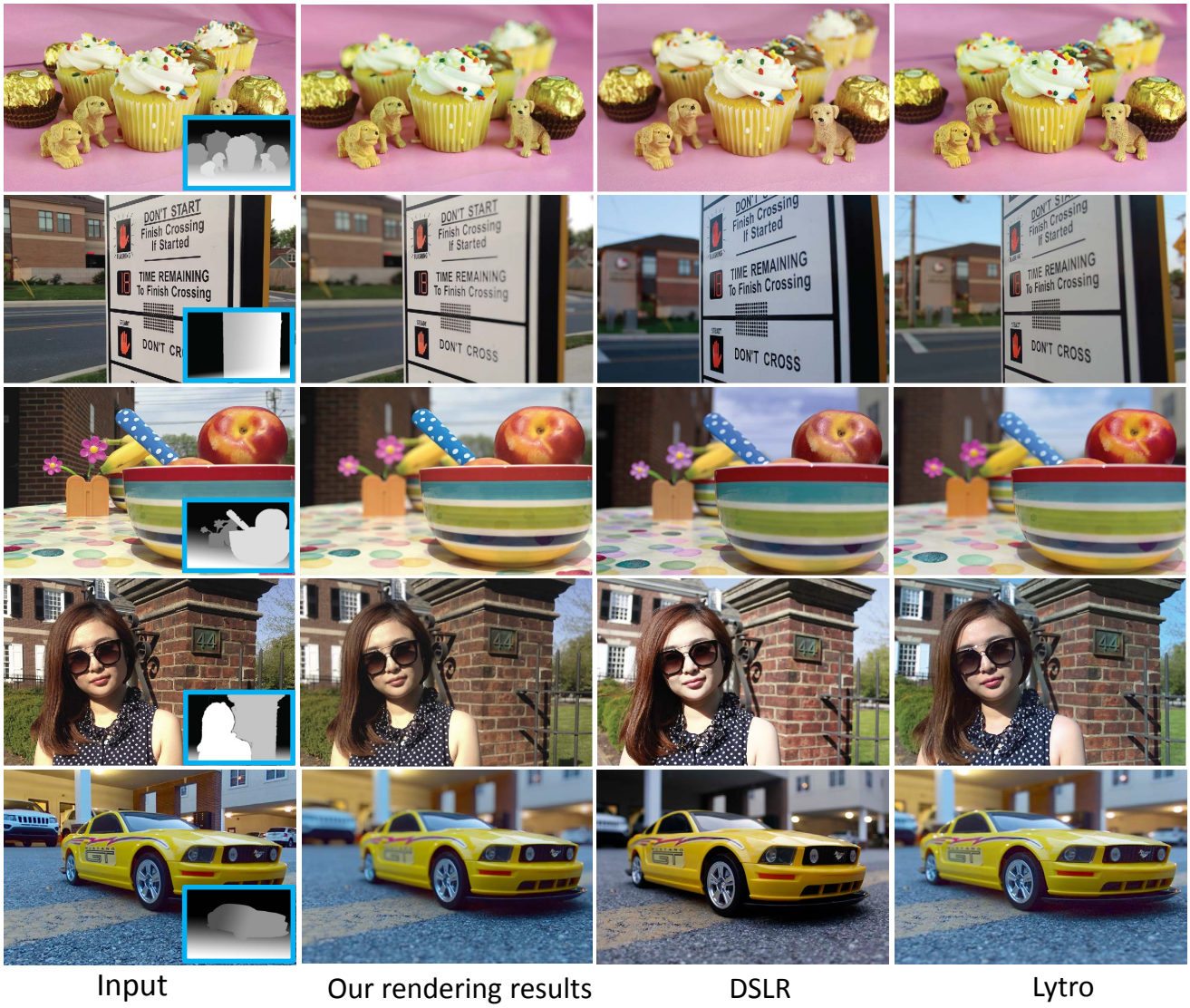


Figure 9: DoF comparisons between a DSLR camera and Lytro Illum.

[17] WANG Q., YU Z., RASMUSSEN C., YU J.: Stereo vision based depth of field rendering on a mobile device. In *SPIE conference of Digital Photography X, 2014* (2014).

[18] YU Z., GUO X., LING H., LUMSDAINE A., YU J.: Line assisted light field triangulation and stereo matching. In *In Proceedings of the Thirteenth International Conference on Computer Vision* (2013).

[19] YU Z., THORPE C., YU X., GRAUER-GRAY S., LI F., YU J.: Dynamic depth of field on live video streams: A stereo solution. In *In proceedings of Computer Graphics International* (2011).

[20] YU X., WANG R., YU J.: Real-time Depth of Field Rendering via Dynamic Light Field Generation and Filtering. *Computer Graphics Forum* (2010).

[21] YU Z., THORPE C., YU X., GRAUER-GRAY S., LI F., YU J.: Dynamic depth of field on live video streams: A stereo solution. In *In proceedings of Computer Graphics International* (2011).

[22] HTC One. https:http://www.htc.com/us/smartphones/htc-one-m8/.

Author Biography

Yang Yang received his MEng degree from the Stevens Institute of Technology, in 2012. He is now a PhD student at Department of Computer and Information Sciences, University of Delaware. His research interests include computer vision and computer graphics.

Haiting Lin received his PhD degree from School of Computing, National University of Singapore in 2003. He is now working as a Postdoc at Department of Computer and Information Sciences, University of Delaware with Prof. Jingyi Yu.

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

DOI: 10.2352/ISSN.2470-1173.2016.18.DPMI-031

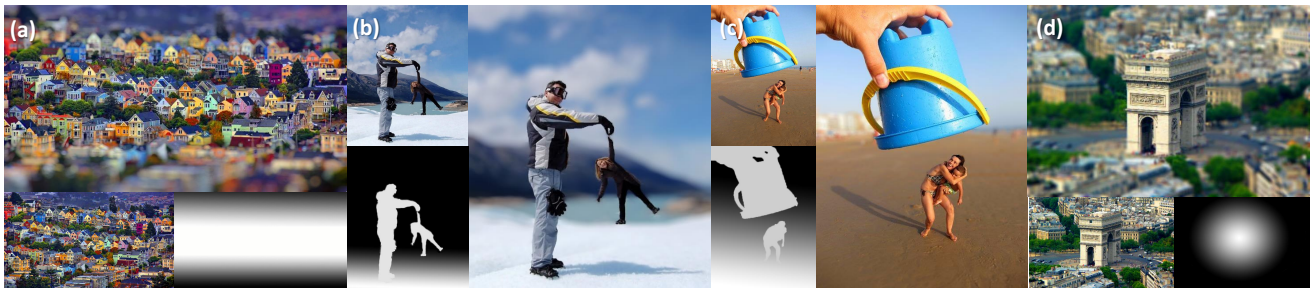


Figure 13: Special effects.

Sylvain Paris is a researcher at Adobe, working in the Advanced Technology Labs. He received his PhD degree from INRIA, France. His research focuses on extracting information from photographs and videos.

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