Refocusing Plenoptic Images using Depth-Adaptive Splatting

Juliet Fiss
University of Washington
juliet@cs.washington.edu

Brian Curless
University of Washington
curless@cs.washington.edu

Richard Szeliski
Microsoft Research
szeliski@microsoft.com

Abstract

In this paper, we propose a simple, novel plane sweep technique for refocusing plenoptic images. Rays are projected directly from the raw plenoptic image captured on the sensor into the output image plane, without computing intermediate representations such as subaperture views or epipolar images. Interpolation is performed in the output image plane using splatting. The splat kernel for each ray is adjusted adaptively, based on the refocus depth and an estimate of the depth at which that ray intersects the scene. This adaptive interpolation method antialiases out-of-focus regions, while keeping in-focus regions sharp. We test the proposed method on images from a Lytro camera and compare our results with those from the Lytro SDK. Additionally, we provide a thorough discussion of our calibration and preprocessing pipeline for this camera.

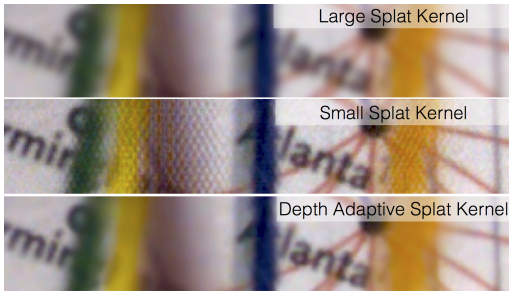

Figure 1. Plenoptic image with text in the background and colored strings in the foreground, rendered so that the text is in focus. The desired properties of the rendering are to have the foreground object antialiased and the text legible. Top: rendering with a large splat kernel. Middle: rendering with a small splat kernel. Bottom: rendering with an adaptive splat kernel.

1. Introduction

Light field imaging has seen growing interest in recent years, from both the artistic and scientific communities. Compact plenoptic cameras of the type described by Adelson and Wang [2], such as Lytro [17], allow computation of a smoothly-varying focal stack from a single exposure. Artistic applications of digital refocusing include providing a sense of depth on a 2D display and correcting errors in focus in photography [17, 22]. Recent scientific applications of digital refocusing include microscopy [14] and seafloor mapping [26].

Because plenoptic cameras work by trading off spatial resolution for angular resolution [10], plenoptic images suffer from reduced spatial resolution and also spatial aliasing [3, 31]. A specific kind of spatial aliasing occurs when the scene contains objects at different depths. When the image is digitally refocused on the background, images in the foreground may appear ghosted. Therefore, antialiasing is an important part of any digital refocusing algorithm. However, care should be taken when antialiasing to keep the in-focus regions sharp. This problem can be challenging for several reasons. First, light field aliasing artifacts are depth-dependent [3]. Also, some applications place additional constraints on refocusing methods. For example, rendered animated focal sweeps should appear temporally coherent (that is, smooth when transitioning from one plane of focus to the next). This paper presents a simple refocusing technique that addresses these kinds of image quality tradeoffs.

1.1. Related Work

Many varied light field rendering techniques have been proposed. A large body of work on image-based rendering addresses the problem of rendering the scene from [super-resolved] novel views [11, 12, 13, 19, 25, 27, 29, 31, 32]. Other related work considers the problem of digital refocusing [16, 21, 22, 23]. Almost all of these rendering techniques share a common preprocessing step: converting the raw plenoptic image captured on the sensor into an intermediate 4D representation $L(x, y, u, v)$, a parameterization described by Adelson and Bergen [1] and further explored in works such as *The Lumigraph* [11] and *Light field rendering* [13]. Depending on how this representation is sliced, it can be viewed as a set of either subaperture views or epipolar images [6]. This intermediate $L(x, y, u, v)$ representa-

tion is usually sampled uniformly in the (x, y) and (u, v) coordinates. However, because the actual rays captured on the sensor represent only a sparse sampling of the (x, y, u, v) rayspace, the intermediate representation contains many interpolated values. The next stage in the typical refocusing pipeline integrates over the u and v dimensions of these intermediate representations, either in the spatial [16, 23] or frequency [21] domain. Integrating over these interpolated values is necessary to correctly antialias the output, but leads to a blurry refocused image [13]. In contrast to the standard pipeline, a method that makes no attempt at antialiasing is presented in [15]; this method rearranges the subimages under each microlens like a puzzle, and severe spatial aliasing artifacts are obvious.

Several methods have been proposed to try to reconcile resolution gains with antialiasing. Cho *et al*. [8] compute a central view that is superresolved using neighboring views. The resulting rendering contains aliasing artifacts, which are corrected in a post-processing step using a learned dictionary. However, there are several drawbacks to using dictionary methods for interpolation. Such methods require training on the right kinds of images to get the desired results. Also, some dictionary methods may not be appropriate for certain scientific applications. Furthermore, correcting aliasing artifacts as a post-processing step may produce attractive results on still images, but the results may not be coherent from plane-to-plane. Therefore, video applications such as animated plane sweeps may not appear temporally coherent. Bishop *et al*. [3, 4] propose an iterative algorithm using depth estimates for antialiasing. Their algorithm repeatedly rearranges light field samples between subimages (the arrangement captured on the sensor) and subaperture views, performing depth map estimation on the views and depth-adaptive filtering on the subimages.

1.2. Our Approach

Our approach is theoretically similar to the method of Bishop *et al*. [3, 4]. However, we avoid the repeated rearrangement of samples, which would require many interpolation steps for a Lytro camera. Instead, we project rays directly from the raw plenoptic image captured on the sensor into the output plane of focus, without computing any intermediate representations such as subaperture views or epipolar images. We perform interpolation in the output space using splatting. Splatting [33] is a simple and well-known rendering technique that has been used before for image based rendering [11, 18]. It is highly parallelizable, and GPU implementations are available [7, 20, 34]. Varying the splat kernel allows us to explore the tradeoff between producing a result that is sharp at in-focus regions, versus antialiased at out-of-focus regions. At one extreme, making the splat kernel large enough to adequately antialias the image will produce output that is also blurry at in-focus regions. At the other extreme, using a splat kernel with a support of one pixel produces a refocused rendering that essentially rearranges the pixels of the input raw image. If we refocus with such a narrow splat kernel at depth d, objects in focus at depth d will be rendered correctly and sharp. However, regions of the image containing texture that is in focus at other depths will create aliasing artifacts, which may show up as lines or other patterns, depending on the depth. See Figure 1. As Ng observed [22], these kinds of artifacts are especially distracting in animations.

To reconcile the two extremes, we first estimate the depth at which each ray intersects the scene, using a measure of photoconsistency. In this plane sweep, a large splat kernel is used for every ray. To render the final output image, the size of the splat kernel for each ray is adjusted adaptively based on the output plane and the depth at which the ray is in focus. This adaptive interpolation method allows us to compute output that is antialiased in out-of-focus regions, but still sharp at in-focus regions.

We test our method on images from a Lytro camera, and compare our results with those from the Lytro SDK. Section 2 describes methods used to preprocess the raw Lytro image. Section 3 describes our splatting-based method for refocusing, and Section 4 describes our methods for refocusing and depth map estimation. Figure 4 shows a flowchart of our algorithm.

2. Preprocessing

Our preprocessing pipeline has two steps: (1) microlens pattern estimation and (2) flat fielding. The goal of microlens pattern estimation is to model the location of each microlens center. The goal of flat fielding is to correct for the complex vignetting pattern induced by the optics of the camera. Much of this vignetting is caused by the microlenses, as shown in Figure 2. Accurate flat fielding is especially important for our method, because we directly compare the color values of raw sensor pixels as a measure of photoconsistency when computing depth maps.

The vignetting pattern of the camera can be measured by imaging a uniform, diffuse white scene. Lytro cameras come loaded with many white images taken with different camera settings. Previous methods, including Dansereau *et al*. [9] and Cho *et al*. [8], use such white images in their preprocessing pipelines for the Lytro camera. We use the method described by [9] to extract a raw light field image from an LFP file, match it with a white image taken with the same camera, and demosaic both raw images. After demosaicing, we have the light field image containing content, I_c, and the white calibration image I_w. Our preprocessing pipeline is similar to those of [9] and [8], but with a few key differences.

First, our method for microlens pattern estimation works on almost any image I_c with a sufficient signal-to-noise ra-

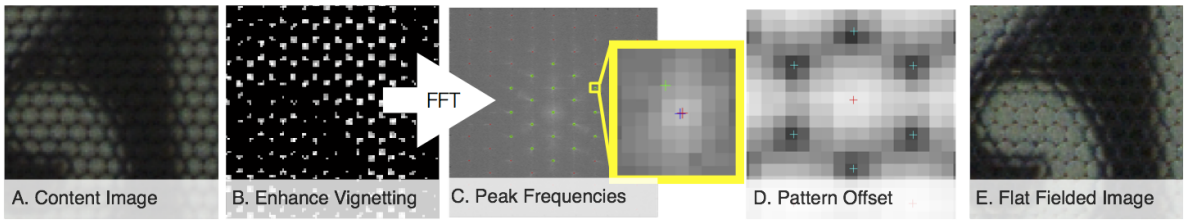

Figure 2. Microlens pattern estimation in the preprocessing pipeline. A. content image. B. content image after vignetting enhancement. C. frequency spectrum of the enhanced vignetting pattern, with peak locations marked. One peak is highlighted. If the microlens array were not rotated with respect to the sensor, the peak location would be near the green marker. The estimated local peak is marked in blue. The predicted peak location after solving for A is marked in red. D. Microlens pattern offset estimated in the spatial domain. Locations of centers are marked in red, and locations of vignetted points are marked in cyan. E. flat-fielded image.

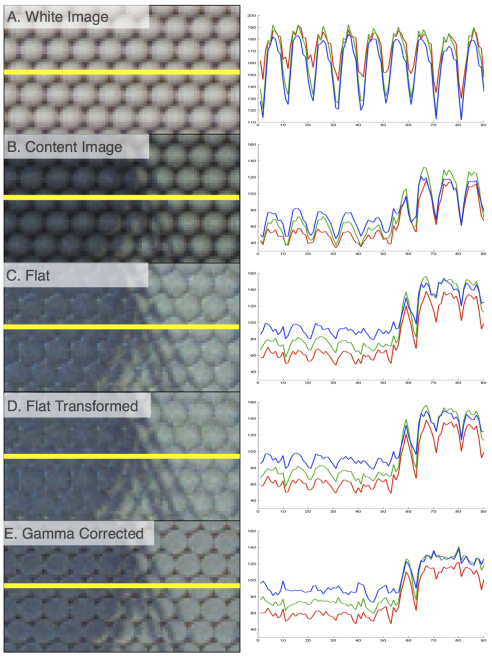

Figure 3. Flat fielding. Left: small sections of a plenoptic image, with a horizontal scanline (shown in yellow). Right: RGB profile of the horizontal scanline. Top to bottom: white image, content image, flat image (no alignment or gamma correction), flat image (alignment only), flat image (alignment and gamma correction). Before alignment, the flat image appears biased (the top part of the microlenses is lighter than the bottom part). Gamma correction removes most of the variation across microlenses due to the periodic microlens vignetting pattern.

tio, including those with complex scene content. The methods proposed by [9] and [8] are designed to work only on I_w, because they look for the local maximum of each microlens. In these previous methods, microlens patterns are estimated on I_w, and assumed to be the same for the matching I_c. In comparison, we are able to separately estimate the microlens pattern of both I_c and I_w, and then compute a transformation to align the images. We empirically observe slight misalignment between most (I_w, I_c) pairs, and our alignment step improves flat fielding results (Figure 2). Dansereau *et al*. perform flat fielding by dividing I_c by a low-pass filtered version of I_w [9], while Cho *et al*. [8] do not perform flat fielding. Our method precisely aligns I_w to I_c, and then divides I_c by a gamma-corrected version of I_w. We compute an optimal gamma for each image pair to minimize variation in the flat-fielded result. This gamma correction step helps compensate for nonlinearity of the camera's response curve and differing exposure levels between I_w and I_c.

2.1. Microlens Pattern Estimation

We model the locations of the microlens centers as a regular integer lattice $\mathbb{Z}^2$ to which an affine transformation T has been applied:

$$T = \begin{bmatrix} [A]_{2\times 2} & & t_x \\ & & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

The parameters of this transformation are unknown, primarily because the microlens array is rotated and translated a small amount with respect to the sensor. However, other issues, such as nonsquare pixels, could also affect this transformation. We estimate the 2×2 transformation matrix A in the frequency domain, and the offset $\mathbf{t} = [t_x \; t_y]^T$ in the spatial domain.

The microlens vignetting pattern is a strong periodic signal, with peaks in the frequency domain located at $A^{-T}\mathbb{Z}^2$. We use this property to solve for A. We first enhance the periodic signal of the microlens vignetting in the spatial domain. We convert the image to grayscale, apply a 13×13 minimum filter, and then threshold the result, using the 80th percentile gray level as an adaptive threshold. The result is a binary image where most of the content inside the microlenses has been set to zero, and the heavily vignetted

points are set to one (Figure 3:B). In the frequency domain, the peaks of this vignetting-enhanced image are easy to extract (Figure 3:C).

We are given that the Lytro camera microlens array has a hexagonal pattern, which is an integer lattice with basis vectors $[1, 0]^T$ and $[\frac{1}{2}, \frac{\sqrt{3}}{2}]^T$. We also assume that we know the lens pitch l, the physical distance between the microlens centers, and the pixel pitch p, the physical distance between pixels. For the Lytro camera, this information is provided in the metadata of an LFP file. We can therefore initialize

$$A_0 = \frac{l}{p} \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} \end{bmatrix}. \tag{2}$$

We look for local peaks in the frequency domain, within a radius R of the DC component, that fall near a sample of $A_0^{-T}\mathbb{Z}^2$ (Figure 3:B). For each peak, we fit a local quadratic to estimate the peak location with subpixel precision. If we store each peak location in the matrix P, we then have

$$A^{-T}\mathbb{Z}^2 = P \tag{3}$$

and can solve for A. This procedure is similar to the method Cho *et al*. [8] used to estimate the rotation of the microlens array with respect to the sensor. However, unlike Cho *et al*., we wish to avoid resampling the original image, so we do not perform a rotation.

Next, we estimate the offset of the microlens sampling pattern $\mathbf{t}$ in the spatial domain. Dansereau *et al*. [9] and Cho *et al*. [8] find the spatial location of microlens centers by looking for peak brightness within the microlenses. While this method works for white images, it is not designed to work on images with scene content. Instead, we look for the dark, highly vignetted points between microlenses, which exist regardless of scene content. These dark points occur in a hexagonal pattern around the microlenses (Figure 3:D). Given $\mathbf{t}$, the microlens centers C are located at $C(\mathbf{t}) = A\mathbb{Z}^2 + \mathbf{t}$. Heavily vignetted points occur above the microlens centers at $V_1(\mathbf{t}) = C(\mathbf{t}) + A\left[\frac{1}{3}, -\frac{2}{3}\right]^T$ and below the microlens centers at $V_2(\mathbf{t}) = C(\mathbf{t}) + A\left[-\frac{1}{3}, \frac{2}{3}\right]^T$. We can find an optimal value of $\mathbf{t}$ by interpolating the image at samples of $V_1 \cup V_2$:

$$\mathbf{t} = \arg\min_{\mathbf{t}'} \sum_{\mathbf{x} \in V_1(\mathbf{t}') \cup V_2(\mathbf{t}')} I(\mathbf{x})^2 \tag{4}$$

This algorithm works well for all white images, as well as most content images. However, the algorithm fails occasionally on content images that contain certain kinds of patterns or nonuniform illumination. In order to improve the algorithm's performance on all images, including these difficult cases, we insert one extra step into the algorithm for estimating the microlens pattern of I_c. We know that the microlens pattern of I_c is close to that of the matching I_w. Therefore, we first estimate the microlens pattern of I_w, and then just before the FFT, we set I_c to zero near the microlens centers of I_w. This step removes the distracting scene content, allowing the algorithm to find the correct A for I_c.

2.2. Flat Fielding

After separately estimating the microlens pattern transformations T_c for the content image I_c and T_w for the white image I_w, we can align I_w to I_c for flat fielding. We compute the transformation

$$T = T_c T_w^{-1} \tag{5}$$

and transform the white image, which we denote as $T(I_w)$. We resample with a high-quality interpolator (we use bicubic interpolation). If we calculate the flat-fielded image as $I = \frac{I_c}{T(I_w)}$, we notice that the vignetting is reduced only partially. Because the camera response is nonlinear, it is highly likely that I_c and I_w received different amounts of exposure. Applying a gamma correction to I_w before flat fielding works as a sort of pseudo-linearization and improves the flat fielding results (Figure 2). Thus, we compute:

$$I_\gamma = \frac{I_c}{T(I_w)^\gamma}. \tag{6}$$

We find an acceptable gamma through optimization. Our approach is to interpolate scanlines S from I that pass through rows of microlenses, and then examine the dominant frequencies in these scanlines. If flat fielding were to work perfectly, we would see very little of the vignetting frequency f_v in these scanlines. Therefore, we find the gamma that minimizes the ratio of f_v to the DC component in the scanlines.

$$\gamma = \arg\min_{\gamma'} \sum_{s \in S} \left(\frac{\int_{-\infty}^{\infty} I_{\gamma'}(s, x) e^{-2\pi i x f_v} dx}{\int_{-\infty}^{\infty} I_{\gamma'}(s, x) dx} \right)^2 \tag{7}$$

In practice, we use an FFT to compute the Fourier transform.

3. Splatting Method for Digital Refocusing

Given a 4D light field optically focused at F, $L_F(x, y, u, v)$, the refocused image at a virtual focal plane αF is:

$$E_\alpha(x, y) = \frac{1}{\alpha^2 F^2} \int \int L_F \left(u\left(1 - \frac{1}{\alpha}\right) + \frac{x}{\alpha}, v\left(1 - \frac{1}{\alpha}\right) + \frac{y}{\alpha}, u, v \right) du\, dv \tag{8}$$

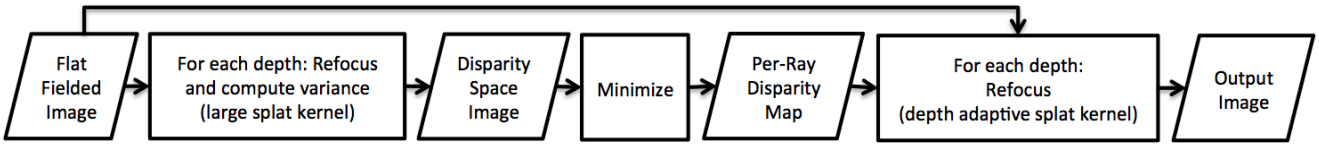

Figure 4. Flowchart of the proposed algorithm.

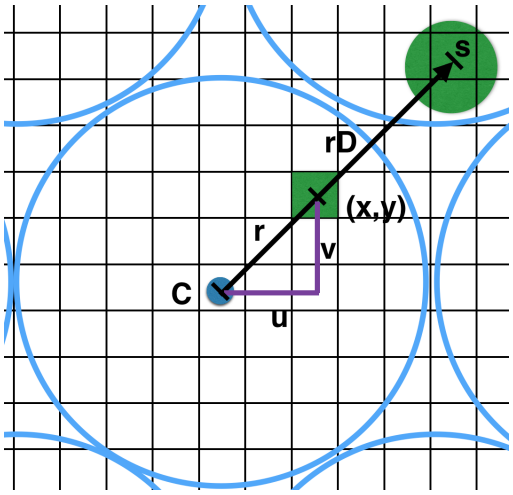

Figure 5. Splatting.

As explained by Ng [22], the scale factor $1/(\alpha^2 F^2)$ can be ignored.

Because only some rays are actually sampled by the sensor, the usual approach to implementing this integral is to first interpolate $L_F(x, y, u, v)$ with some regular sampling pattern, and then integrate over the many interpolated values. Our method implements the integral using a forward warp rather than an inverse warp. From the integral, we notice that the ray i with value $L_F(x_i, y_i, u_i, v_i)$ contributes some energy to the output image E_α at location

$$\mathbf{s}_{\alpha,i} = \left[\alpha\left(x_i - u_i\left(1 - \frac{1}{\alpha}\right)\right), \alpha\left(y_i - v_i\left(1 - \frac{1}{\alpha}\right)\right)\right] \tag{9}$$

We use $\mathbf{s}_{\alpha,i}$ as the splat location for ray i in output image E_α. Simplifying further, we see that α only serves as a dilation factor for the output image, so if we want output images of a fixed size, we can safely ignore it. This simplification is also described by Ng [22]. If we let

$$D_\alpha = \left(1 - \frac{1}{\alpha}\right) \tag{10}$$

where D_α is the disparity level for focal plane αF, then

$$\mathbf{s}_{\alpha,i} = (x_i - u_i D_\alpha, y_i - v_i D_\alpha)\,. \tag{11}$$

Figure 5 illustrates finding the splat location for a pixel. Our approach is to consider only rays $L_F(x, y, u, v)$ that were recorded on the sensor, and splat the color of these rays into the output image E_α at location $(x - uD_\alpha, y - vD_\alpha)$.

To illustrate the differences between the inverse warp and forward warp approaches, consider the subaperture image method for refocusing. This method chooses a set of (u, v) coordinate pairs (each coordinate pair defining one subaperture view), and interpolates (x, y) at each (u, v). Next, the subaperture images are shifted by (uD_α, vD_α) and accumulated. In short, this method can be described as: (1) interpolate, (2) shift, (3) accumulate. Our splatting-based method essentially swaps the order of the shift step and the interpolate step. First, the shift is performed on each ray individually by calculating its splat location $\mathbf{s}_{\alpha,i}$. Interpolation and accumulation are performed in the output space using splatting. Splatting can be viewed as a form of scattered data interpolation using radial basis functions.

We now discuss our implementation of this splatting approach. We model each pixel on the sensor as recording one ray $L(x, y, u, v)$, where x and y are the absolute coordinates of the pixel, and u and v are the coordinates of the pixel relative to the center of the microlens C. Because the pixel grid is slightly rotated with respect to the microlens array, u and v are not exactly horizontal and vertical pixel displacements. We use the flat-fielded image $I = I_\gamma$ as our estimate of ray colors.

$$L_F(x, y, u, v) \cong I(x, y, x - C, y - C) \tag{12}$$

If a ray i located at $\mathbf{x}_i = [x_i \;\; y_i]$ has a displacement $\mathbf{r}_i = [u_i \;\; v_i]$ from the microlens center, the ray is splatted to $\mathbf{s}_i = \mathbf{x}_i - D_\alpha \mathbf{r}_i$. Each ray i is associated with a splat kernel $k_{\alpha,i}(r)$ at disparity level D_α. Additionally, we use $c = T(I_w)^2$ as a confidence weighting for each ray. We are more confident in the color values of the pixels near the center of the microlens, because they have a higher signal-to-noise ratio than the heavily vignetted pixels. We set the resolution of the refocused images to be equal to the sensor resolution, which for the Lytro camera is 3280×3280 pixels. We compute the refocused image as

$$E_\alpha(\mathbf{x}) = \frac{1}{W_\alpha(\mathbf{x})} \sum_i I(\mathbf{x}_i) k_{\alpha,i}\left(|\mathbf{x} - \mathbf{x}_i - D_\alpha \mathbf{r}_i|\right) c_i \tag{13}$$

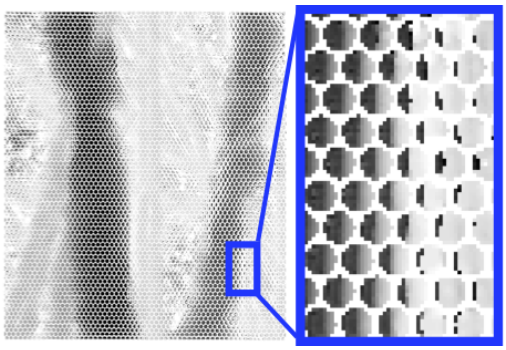

Figure 6. Per-ray disparity map for the image in Figure 9, with detail highlighted. A darker value indicates that the ray is in focus at a closer depth.

where

$$W_\alpha(\mathbf{x}) = \sum_i k_{\alpha,i}\left(|\mathbf{x} - \mathbf{x}_i - D_\alpha \mathbf{r}_i|\right) c_i \qquad (14)$$

The definition of the splat kernel may be changed to shape the appearance of the output refocused image, and may vary based on α and i. Kernel selection for scattered data modeling is a well-studied problem [24]. For the results shown in this paper, we use a radially symmetric linear kernel with a scaling parameter $b_{\alpha,i}$

$$k_{\alpha,i}(r) = \max\left(1 - \frac{r}{b_{\alpha,i}}, 0\right). \qquad (15)$$

In the next section, we describe how we initially fix $b_{\alpha,i}$ to be a large value when computing a depth map, and then adapt $b_{\alpha,i}$ based on depth to compute a refocused image that is both antialiased and sharp in in-focus regions.

4. Plane Sweep and Depth Estimation

In this section, we describe our plane sweep algorithm for digital refocusing and depth map estimation. First, we estimate the depth at which each ray intersects the scene; we find the disparity D_{α_i} with the highest photoconsistency for each ray i. We use a measure of photoconsistency based on variance. For each disparity level D_α, we calculate a mean (refocused) image E_α, and then calculate the SSD color difference (photoconsistency measure) between each ray and the mean image for that depth. Refocusing is performed using the method described in Section 3, using a splat kernel of a large, fixed size. The splat kernel must be large enough that aliasing artifacts are not visible in the refocused image, as such artifacts can interfere with the depth map computation step. We use $b_{\alpha,i} = l$, where l is the microlens pitch. Splatting with this large kernel produces a refocused image that is antialiased, but also blurry at in-focus regions.

Figure 7. Out-of-focus plane exhibiting artifacts. Top: jitter not applied. Bottom: artifact corrected with jitter.

Next, we construct a disparity space image (DSI). We again project each ray into the scene according to Equation 11. However, instead of splatting the ray color, we interpolate $E_\alpha(\mathbf{s}_i)$, the color of the mean image at the splat location. We then compute the SSD color difference (disparity cost) between the ray color and the mean image color:

$$SSD_{\alpha,i} = \|I(\mathbf{x}_i) - E_\alpha(\mathbf{s}_{\alpha,i})\|_2^2. \qquad (16)$$

We splat this color difference into DSI_α at $\mathbf{s}_{\alpha,i}$ using the same large splat kernel with $b_{\alpha,i} = l$.

Finally, we find the α with the lowest disparity cost for each ray in the input light field. We project each ray i into the scene and interpolate $DSI_\alpha(\mathbf{s}_{\alpha,i})$ for the ray at each value of α. We set

$$\alpha_i = \arg\min_\alpha DSI_\alpha(\mathbf{s}_{\alpha,i}) \qquad (17)$$

Figure 6 shows the per-ray disparity map for a small section of a plenoptic image.

To compute the final refocused image, we use a splat width $b_{\alpha,i}$ of a variable size. How $b_{\alpha,i}$ changes with α and i can be varied based on the desired depth-of-field effect. For the results shown in this paper, we use

$$b_{\alpha,i} = b_{min} + l\,|D_\alpha - D_{\alpha_i}|^n + w(\alpha) \qquad (18)$$

where b_{min} and n are constants (we use $b_{min} = 2$ and $n = 3.5$) and $w(\alpha)$ is used to increase the size of the splat kernel as α approaches the edge of the refocusable range of the camera. We use a sigmoidal function

$$w(\alpha) = \frac{p_0}{1 + e^{-\frac{(D_\alpha - p_2)}{p_1}}} \qquad (19)$$

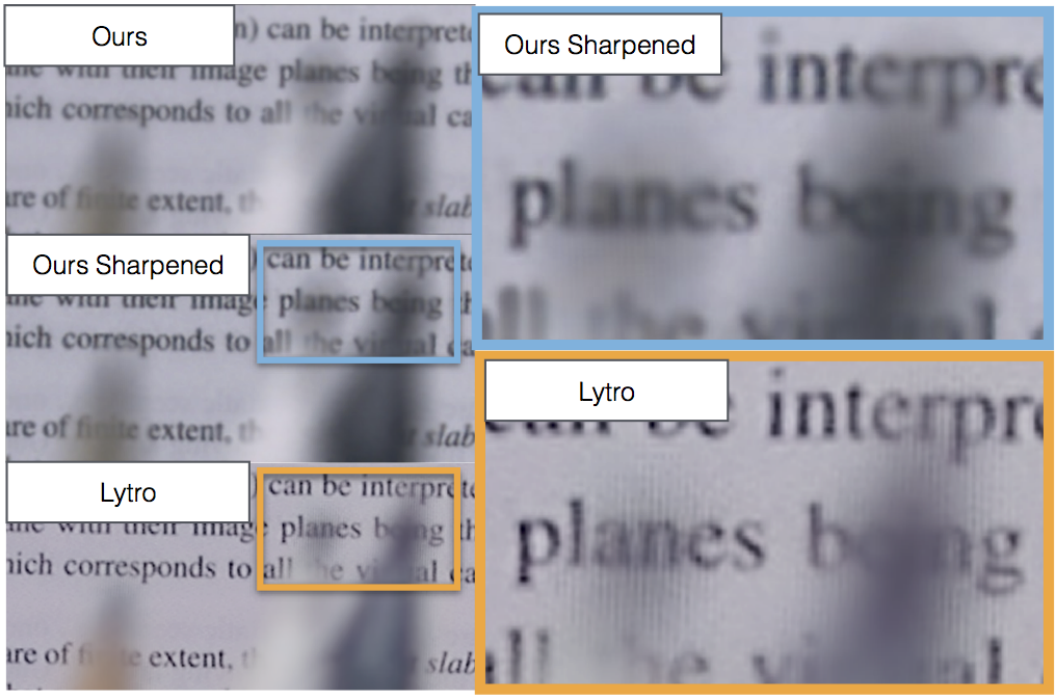

Figure 8. Left: refocusing results. Top: ours. Middle: ours, sharpened. Bottom: Lytro. Right: highlighted detail from our sharpened result (top) and Lytro result (bottom).

with parameters $p_0 = 15$, $p_1 = 20$, and $p_2 = 100$.

At some depths, especially nearing the edge of the refocusable range of the camera, lines or patterns appear as artifacts in the refocused result. In order to break up the artifact, we jitter the splat location of the rays:

$$\mathbf{s}_{\alpha,i} = \mathbf{s}_{\alpha,i} + [j_x, j_y] \tag{20}$$

where j_x and j_y are random variables drawn from $Uniform\left(-\frac{w(\alpha)}{2}, \frac{w(\alpha)}{2}\right)$. The resulting image is free of the patterning artifact, but can appear noisy or mottled for large $w(\alpha)$. To smooth out the result, rays can be splatted multiple times at different jittered locations. See Figure 7. In order to preserve coherency from plane-to-plane, the same jitter offsets should be used for each ray for each α.

5. Results

Figures 7 and 8 demonstrate refocusing an image at two planes of focus. Figures 9 and 10 demonstrate refocusing a second image at three planes. The images shown are cropped regions of interest of larger Lytro images. In Figures 8, 9, and 10, we compare the results of our method to those produced by the Lytro SDK. Our results are rendered at 3280 $\times$ 3280 pixels, while the Lytro images are rendered at 1080 $\times$ 1080 pixels. For this detail comparison, we have upsampled the Lytro result using bicubic interpolation. We apply a manual color correction to match the appearance of the Lytro results. The Lytro results appear sharper, but also noisier, than our results. The Lytro results also contain aliasing artifacts at depth discontinuities (Figures 8 and 9), patterning artifacts (Figure 10), and ringing artifacts from a sharpening filter. In Figure 8, we sharpen our result and compare with Lytro. For additional results, please see our supplemental material.

6. Limitations and Discussion

Our method has two major limitations. The first limitation arises from the design of the plenoptic camera itself. Plenoptic images can only be refocused through a certain range. Also, as described by Bishop *et al*. [5, 3], a specific plane of focus exists where the camera has very low spatial resolution. Near this plane and the edges of the refocusable range, the size of the splat kernel must be increased. For some applications, the best solution may be to skip over these planes completely when performing a plane sweep. The second limitation arises from the method used to estimate depth. In this paper, we use photoconsistency based on variance. We assume a scene that is mostly Lambertian, and depth is only estimated accurately in regions that contain texture. Any method for depth estimation may be used,

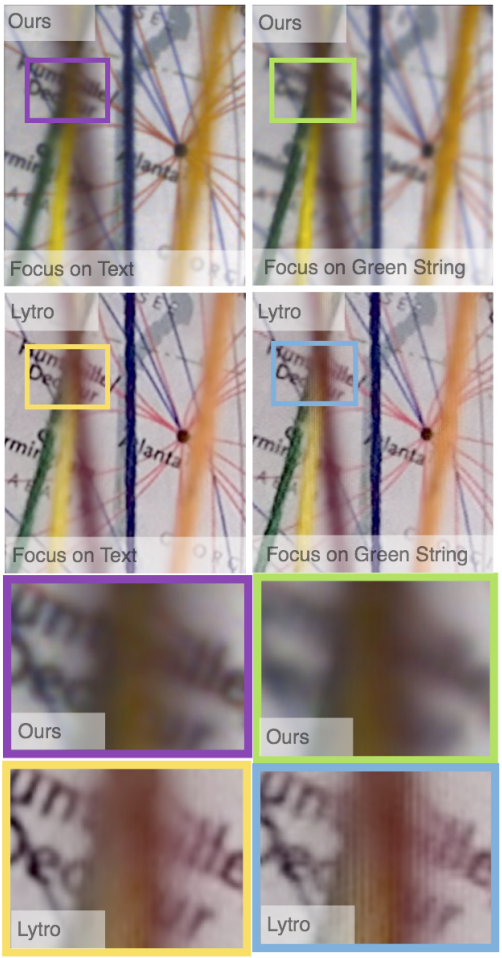

Figure 9. Refocusing on two nearby planes, with highlighted detail. Left: refocused on background (text). Right: refocused closer (on green string). Aliasing artifacts appear in the Lytro result when the green string is in focus. Our method does not display these artifacts, and varies smoothly from plane-to-plane.

and each will have its own strengths and limitations. When errors occur in the depth map, in-focus objects will be rendered less sharp than if the depth error had not occurred. In low texture regions, the size of the splat kernel does not noticeably affect the rendered result, so accurate depth information is not needed.

7. Conclusions and Future Work

In this paper, we have presented a simple method for refocusing plenoptic images that produces sharp and antialiased results. We have also presented a detailed description of our preprocessing and calibration pipeline for the Lytro camera. Our refocusing algorithm does not require computing or storing intermediate 4D light field representations, such as subaperture views or epipolar images. For rendering, our algorithm utilizes splatting, which can be implemented efficiently on the GPU. Animated focal sweeps computed using our method are smooth and temporally coherent. We plan to construct a GPU implementation in future work. We also plan to combine our calibration and refocusing algorithms with other methods for processing plenoptic images, such as improved demosaicing [35], distortion correction [9], alternative depth estimation methods [28, 30], and image-based rendering techniques.

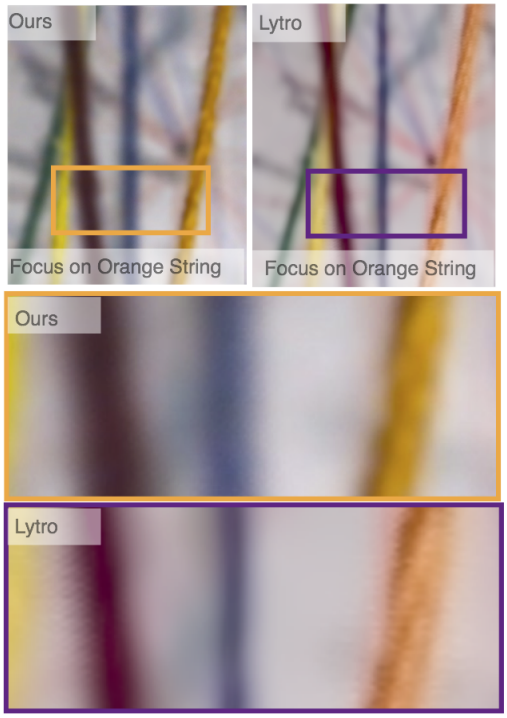

Figure 10. Refocusing on orange string, with highlighted detail.

Acknowledgements

We would like to thank Steve Marschner and Supasorn Suwajanakorn for their help on getting started with processing the Lytro image format, as well as the Lytro SDK team for their continued support. Thank you to the authors of Pujades *et al*. [25] for providing a preprint of their upcoming CVPR paper. We would like to acknowledge funding from Microsoft, the NVIDIA Graduate Research Fellowship, National Science Foundation grant IIS-0963657, and the University of Washington Animation Research Labs.

References

[1] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*, pages 3–20. MIT Press, 1991.

[2] E. H. Adelson and J. Y. A. Wang. Single lens stereo with a plenoptic camera, 1992.

[3] T. Bishop and P. Favaro. Plenoptic depth estimation from multiple aliased views. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1622–1629, 2009.

[4] T. Bishop and P. Favaro. The light field camera: Extended depth of field, aliasing, and superresolution. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(5):972–986, 2012.

[5] T. Bishop, S. Zanetti, and P. Favaro. Light field superresolution. In *Computational Photography (ICCP), 2009 IEEE International Conference on*, pages 1–9, April 2009.

[6] R. C. Bolles, H. H. Baker, David, and H. Marimont. Epipolarplane image analysis: An approach to determining structure from motion. In *Intern..1. Computer Vision*, pages 1–7, 1987.

[7] M. Botsch, A. Hornung, M. Zwicker, and L. Kobbelt. High-quality surface splatting on today's gpus. In *Proceedings of the Second Eurographics / IEEE VGTC Conference on Point-Based Graphics*, SPBG'05, pages 17–24, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[8] D. Cho, M. Lee, S. Kim, and Y.-W. Tai. Modeling the calibration pipeline of the lytro camera for high quality light-field image reconstruction. In *ICCV*, 2013.

[9] D. G. Dansereau, O. Pizarro, and S. B. Williams. Decoding, calibration and rectification for lenselet-based plenoptic cameras. In *CVPR*, pages 1027–1034, 2013.

[10] T. Georgeiv, K. C. Zheng, B. Curless, D. Salesin, S. Nayar, and C. Intwala. Spatio-angular resolution tradeoff in integral photography. In *In Eurographics Symposium on Rendering*, pages 263–272, 2006.

[11] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *In Proceedings of SIGGRAPH 96*, pages 43–54. ACM, 1996.

[12] C. Kim, H. Zimmer, Y. Pritch, A. Sorkine-Hornung, and M. Gross. Scene reconstruction from high spatio-angular resolution light fields. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 32(4):73:1–73:12, 2013.

[13] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, pages 31–42, New York, NY, USA, 1996. ACM.

[14] M. Levoy, R. Ng, A. Adams, M. Footer, and M. Horowitz. Light field microscopy. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 924–934, New York, NY, USA, 2006. ACM.

[15] A. Lumsdaine and T. Georgiev. Full resolution lightfield rendering. Technical report, Adobe, 2008.

[16] A. Lumsdaine and T. Georgiev. The focused plenoptic camera. In *In Proc. IEEE ICCP*, pages 1–8, 2009.

[17] Lytro. *The Lytro Camera*. http://lytro.com.

[18] W. R. Mark, L. McMillan, and G. Bishop. Post-rendering 3d warping. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics*, I3D '97, pages 7–ff., New York, NY, USA, 1997. ACM.

[19] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '95, pages 39–46, New York, NY, USA, 1995. ACM.

[20] N. Neophytou and K. Mueller. Gpu accelerated image aligned splatting. In *Volume Graphics, 2005. Fourth International Workshop on*, pages 197–242, June 2005.

[21] R. Ng. Fourier slice photography. *ACM Trans. Graph.*, 24(3):735–744, July 2005.

[22] R. Ng. *Digital Light Field Photography*. Phd thesis, Stanford University, CA, USA, July 2006.

[23] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. Light Field Photography with a Hand-Held Plenoptic Camera. Technical report, Apr. 2005.

[24] G. M. Nielson. Scattered data modeling. *IEEE Comput. Graph. Appl.*, 13(1):60–70, Jan. 1993.

[25] S. Pujades, B. Goldluecke, and F. Devernay. Bayesian view synthesis and image-based rendering principles. In *CVPR*, 2014 (to appear).

[26] C. Roman, G. Inglis, J. Vaughn, C. Smart, D. Dansereau, D. Bongiorno, M. Johnson-Roberson, and M. Bryson. New tools and methods for precision seafloor mapping. In *Oceanography*, volume 26, pages 10–15, 2013.

[27] H. Shum and S. B. Kang. Review of image-based rendering techniques. In *Proc. SPIE*, volume 4067, pages 2–13, 2000.

[28] M. W. Tao, S. Hadap, J. Malik, and R. Ramamoorthi. Depth from combining defocus and correspondence using light-field cameras. In *International Conference on Computer Vision (ICCV)*, Dec. 2013.

[29] S. Todt, C. Rezk-Salama, A. Kolb, and K. Kuhnert. Gpu-based spherical light field rendering with per-fragment depth correction. In *Computer Graphics Forum*, volume 27, pages 2081–2095, 2008.

[30] V. Vaish, R. Szeliski, C. L. Zitnick, S. B. Kang, and M. Levoy. Reconstructing occluded surfaces using synthetic apertures: Stereo, focus and robust measures. In *In Proc. Computer Vision and Pattern Recognition*, page 2331, 2006.

[31] S. Wanner and B. Goldluecke. Spatial and angular variational super-resolution of 4d light fields. In *European Conference on Computer Vision (ECCV)*, 2012.

[32] S. Wanner and B. Goldluecke. Variational light field analysis for disparity estimation and super-resolution. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(3):606–619, March 2014.

[33] L. Westover. Footprint evaluation for volume rendering. In *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '90, pages 367–376, New York, NY, USA, 1990. ACM.

[34] D. Xue and R. Crawfis. Efficient splatting using modern graphics hardware. *J. Graphics, GPU, & Game Tools*, 8(3):1–21, 2003.

[35] Z. Yu, J. Yu, A. Lumsdaine, and T. Georgiev. An analysis of color demosaicing in plenoptic cameras. In *CVPR*, pages 901–908. IEEE, 2012.