# Fast Physically-Correct Refocusing for Sparse Light Fields Using Block-Based Multi-Rate View Interpolation

Chao-Tsung Huang, *Member, IEEE*, Yu-Wen Wang, Li-Ren Huang, Jui Chin, and Liang-Gee Chen, *Fellow, IEEE*

***Abstract*—Digital refocusing has a tradeoff between complexity and quality when using sparsely sampled light fields for low-storage applications. In this paper, we propose a fast physically-correct refocusing algorithm to address this issue in a two-fold way. First, view interpolation is adopted to provide photorealistic quality at infocus-defocus hybrid boundaries. Regarding its conventional high complexity, we devised a fast line-scan method specifically for refocusing, and its 1D kernel can be 30x faster than the benchmark VSRS-1D-Fast. Second, we propose a block-based multi-rate processing flow for accelerating purely infocused or defocused regions, and a further 3-34x speedup can be achieved for high-resolution images. All candidate blocks of variable sizes can interpolate different numbers of rendered views and perform refocusing in different subsampled layers. To avoid visible aliasing and block artifacts, we determine these parameters and the simulated aperture filter through a localized filter response analysis using defocus blur statistics. The final quadtree block partitions are then optimized in terms of computation time. Extensive experimental results are provided to show superior refocusing quality and fast computation speed. In particular, the run time is comparable to the conventional single-image blurring which causes serious boundary artifacts.**

## I. Introduction

Light-field cameras capture light rays with 4D information that consists of 2D spatial pixel positions in $(x, y)$ and 2D angular view/aperture positions in $(u, v)$ [1], [2], [3]. They can be implemented in different physical configurations such as large-baseline camera array [4], [5], hand-held camera with a micro-lens array [6], and monolithic camera array [7]. Their 4D light fields are mathematically equivalent under the thin-lens assumption and thus share the same novel applications. Digital refocusing in particular is a key application to provide novel 2D images that are focused at different focal planes and with different aperture sizes after capturing the light fields.

The most common artifact of digital refocusing is the aliasing effect in defocused regions that results from insufficient view/aperture sampling in $(u, v)$. For sparse light fields, view interpolation can compensate such insufficient sampling and provide physically-correct results by explicitly handling occlusions. However, the computation complexity is high in literature because complexed algorithms are adopted [8], [9], [10]. In contrast, dense light fields suffer only slight aliasing for their high view sampling rate, and depth-based pixel splatting is shown sufficient to alleviate the aliasing [11], [12]. However, the storage requirement for so many views will become a bottleneck for practical usage, especially video applications.

A classical fast method to avoid aliasing is the depth-dependent image blurring [13], [14]. It applies an adaptive convolution kernel on a single image to simulate defocused blurs. However, it also introduces discontinuous artifacts near object boundaries in infocus-defocus hybrid regions because no information is available for occluded pixels.

In this paper, we aim to provide a fast refocusing algorithm which can achieve photorealistic depth-of-field effects for sparse light fields. The target system in Fig. 1 has the following advantages: low storage using only few views, good refocusing quality as dense light fields provide, and fast computation speed comparable to the adaptive image blurring. The contribution of this paper is to greatly reduce the computation complexity of view-interpolation refocusing:

$$O(C_{vi} \cdot V \cdot I), \tag{1}$$

where $C_{vi}$ is the complexity of view interpolation, $V$ is the number of rendered views, and $I$ is the image/block size. Note that the total complexity increases bi-quadratically with the

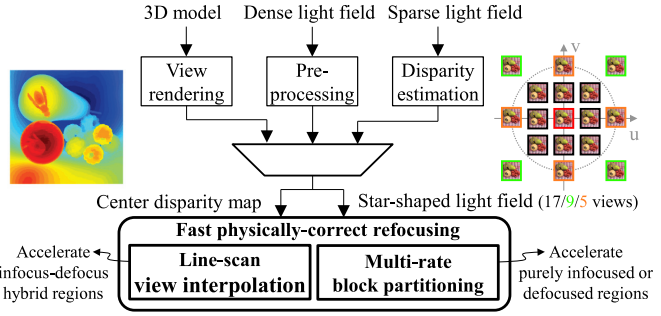



Fig. 1. Target system setup for fast physically-correct refocusing. Only a sparse star-shaped light field and the corresponding center disparity map are required for low-storage applications. They can be generated from different image sources. For example, dense light fields can be preprocessed through disparity estimation, super-resolution, view synthesis and denoising.

This work was supported by the Ministry of Science and Technology, Taiwan, R.O.C. under Grant No. MOST 103-2218-E-007-008-MY3.

C.-T. Huang is with the National Tsing Hua University, Hsinchu, Taiwan (e-mail: chaotsung@ee.nthu.edu.tw).

L.-R. Huang was with the National Tsing Hua University, Hsinchu, Taiwan, and is now with the MediaTek, Inc., Hsinchu, Taiwan.

Y.-W. Wang was with the National Tsing Hua University, Hsinchu, Taiwan, and is now with the Silicon Motion, Inc., Hsinchu, Taiwan.

J. Chin was with the National Taiwan University, Taipei, Taiwan, and is now with the PixArt Imaging Inc., Hsinchu, Taiwan.

L.-G. Chen is with the National Taiwan University, Taipei, Taiwan.

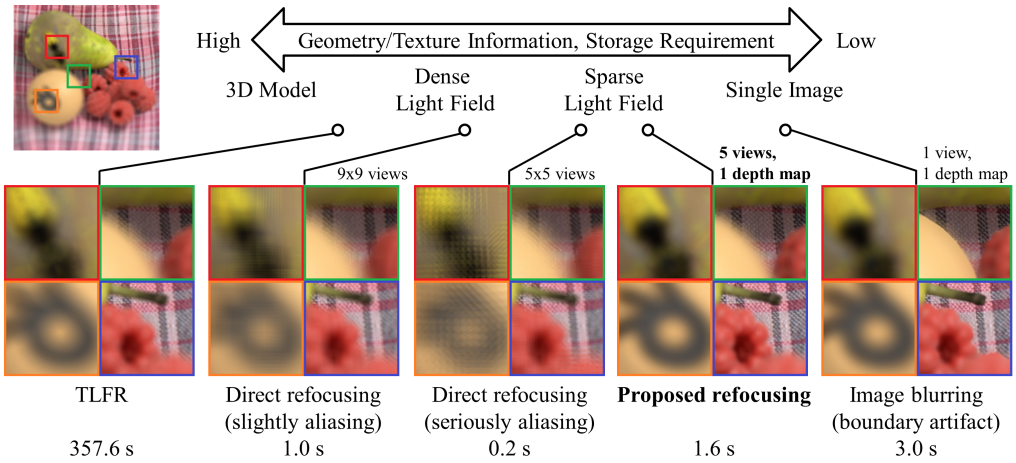



Fig. 2. Examples of digital refocusing for different image sources. The resolution of the tested light field *StillLife* from [15] is 768 × 768. Run times on a single-thread 3.4 GHz CPU are reported for purpose of comparing the relative computation complexity between different refocusing methods. Note that TLFR [20] needs only few seconds when accelerated by GPU. The proposed method provides similarly good quality using only five views compared to the highly complex TLFR. Its speed is comparable to the fast blurring method which has serious artifacts near the infocus-defocus hybrid boundaries of the wooden ball and berries.

image width/height because the required $V$ is proportional to $I$ for the same scene. Therefore, it is difficult for the conventional methods to efficiently support high-resolution images.

The fast refocusing algorithm is proposed in a two-fold way. First, a fast line-scan view interpolation algorithm was designed specifically for refocusing to reduce the complexity $C_{vi}$ which is especially high for the infocus-defocus hybrid regions. It is based on the fact that each rendered view will not be seen directly, so per-view artifacts can be tolerable and the view interpolation can be simple. Second, a block-based processing flow with multi-rate view interpolation was developed to reduce the unnecessary computation of the conventional frame-based refocusing. Each variable-size candidate block is processed differently because infocused regions only need few rendered views ($V\downarrow$) and defocused areas can be refocused in subsampled layers ($V \cdot I\downarrow$). To quantitatively determine these parameters and also the simulated aperture filter for avoiding aliasing and block artifacts, we performed a localized response analysis for the corresponding defocusing filters and constrained their differences from the ideal ones.

In this paper, we extend the preliminary ideas in our previous work [16], [17], which considered only 1D filter kernels, to a complete system with detailed theoretical analysis for 2D kernels. Also, extensive experimental results will be presented for more subjective and objective comparisons and different configurations of input views and subsampled layers. The rest of this paper is organized as follows. After summarizing related work in Section II, the filter response analysis is presented in Section III. Then the two building blocks, line-scan view interpolation and block-based multi-rate refocusing, are introduced in Sections IV and V respectively. Experimental results are shown in Section VI. Finally, the limitations and possible extensions of this work are discussed in Section VII, and concluding remarks in Section VIII.

## II. Related Work

### A. View Sampling for 3D Models

Distributed ray tracing [18] provides great depth-of-field effects but requires intensive computation for sampling lots of views. A common approach to save computation is stochastic view sampling, e.g. the low-discrepancy sampling in [19]. The temporal light field reconstruction (TLFR) in [20] further samples fewer views (e.g. 16) and reconstructs denser views (e.g. 256) according to depth information. Using a layered process for the reconstruction can be further 8-10x faster [21]. However, these methods still require aggressive GPU acceleration to achieve fast processing. In this paper, we target to greatly reduce such computing demand while retaining photorealistic depth-of-field effects.

Another approach, Fourier depth of field [22], adaptively samples not only views but also image pixels based on localized depth and texture spectrum analysis; however, the complexity overhead of the image-dependent analysis is high. In this paper, we perform the filter response analysis only for determining parameter selection rules and a simulated aperture filter in advance, and no computation overhead is required when refocusing.

### B. Pixel Splatting for Dense Light Fields

If the views are sampled densely enough, averaging shifted views directly forms a great refocused image by definition. The Fourier slice [23] was developed to accelerate this direct method for refocusing at different focal planes. However, in practice the view sampling rate is usually insufficient, and the direct refocusing will still cause aliasing artifacts as the 9×9 light field in Fig. 2. Depth-dependent pixel splatting can alleviate such aliasing for dense light fields [11], [12].

### C. View Interpolation for Sparse Light Fields

View interpolation is a common technique to reduce the worse aliasing effect when the sampling becomes more sparse

as the 5×5 light field in Fig. 2. However, the computation could be demanding, such as the tri-view morphing [8], hybrid geometric-/image-based synthesis [9], and inpainting-based hole filling [10]. The reader is referred to [24] for more details about view synthesis.

One well-known fast method is the VSRS-1D-Fast in the reference software 3D-HTM [25] for 3D-HEVC. It performs line-wise image warping and hole filling from both left and right viewpoints separately (in upsampled-by-four domain) and then combines them based on reliability. In this paper, we provide a much faster algorithm for refocusing which uses only one disparity map and simplified operations. In addition, the number of rendered views was set heuristically in [8], [9], [10]. Instead, we determine it systematically using the filter response analysis.

In [26], one light-field rendering method was proposed to reconstruct all rendered views from a small number of 1D viewpoint trajectories by recovering spectral sparsity. It can reduce sampling requirements; however, each 1D viewpoint trajectory is densely sampled to avoid aliasing. Also, the computation complexity is quite high, e.g. typical run times range from two to three hours for one light field using 70-84 CPU cores. In this paper, we target at sparse 2D viewpoints which have much fewer views, and the refocusing run times are of a few seconds for one light field using one CPU core.

### D. Image Blurring for Single-View Images

Depth-dependent image blurring is the most common technique for fast refocusing [13], [14], but it introduces obvious boundary artifacts [27] as shown in Fig. 2. The layered-depth processing in [28] can reduce such artifacts by applying alpha blending for neighboring depth layers. But it cannot help these single-image artifacts for large apertures and also loses details and gradients of the depth map. In [29], image deconvolution was applied for refocusing because the input image has obvious depth-of-field effects. In contrast, we assume the pinhole camera model for the light fields in this paper.

### E. Spectrum Analysis

Plenoptic sampling [30] provides a lower bound of the view sampling rate without light-field aliasing. On the other hand, a more generalized light-field model was discussed in [12] which considered detailed physical information such as sensor profile and finite aperture. In particular, it reasoned that the defocus blurs are time-varying, so only localized filter responses can be studied using Fourier analysis. In this paper, we study the localized 2D defocusing filters without any physical information of the imaging devices.

## III. Localized Defocusing Filter Analysis

Digital refocusing manipulates discrete pixels to approximate the ideal blurs resulting from a continuous aperture. In this paper, each block has two main parameters for refocusing: view-interpolation number $N$ and multi-rate subsampled layer $L$. A lower $N$ or a higher $L$ saves more computation but makes the blurs differ more from the ideal ones as shown in Fig. 3.

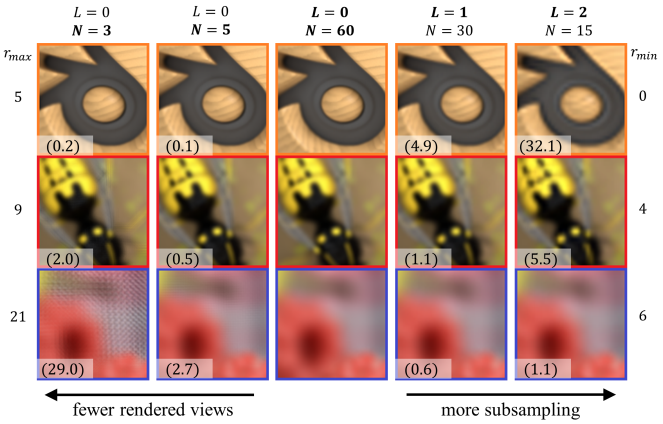



Fig. 3. Refocusing examples for varying $N$ and $L$. Center column: nearly ideal blurs (large $N$) for three image blocks with different local blur statistics ($r_{max}$ and $r_{min}$ denote the maximum and minimum defocus blur radii respectively). Other columns: refocused blocks using smaller $N$ or higher $L$ to save computation; numbers at corners represent the corresponding mean squared errors compared to the center column.

For them, we will first study the localized response and then derive their selection rules by limiting the difference from the ideal blurs to reduce both aliasing and block artifacts. The corresponding aperture filter will also be derived.

### A. Discrete Ideal Defocusing Blur

Assume the local region under consideration belongs to a Lambertian surface and has one single disparity value $d_p$. The disparity is defined as the spatial position difference of the same object between the views at $u = 0$ and 1. Let $d_t$ be the target infocused disparity and $u_t$ be the target aperture radius such that $\{(u,v)|\sqrt{u^2+v^2} \le u_t\}$ covers the considered aperture. Then the defocus blur has a radius $r$:

$$r = u_t \cdot |d_p - d_t|. \tag{2}$$

Given an aperture filter $a(u,v)$, we define its point spread function as $\frac{1}{r^2}a(\frac{x}{r},\frac{y}{r})$. When an explicit function is required for $a(u,v)$ in this paper, we will use the cosine fourth fall-off:

$$a(u,v;\theta) = \frac{1}{\pi\cos^2\theta}\frac{\cot^4\theta}{(u^2+v^2+\cot^2\theta)^2}\mathbb{1}_{\{u^2+v^2\le 1\}}, \tag{3}$$

where the first term is a normalization factor for intensity conservation, the second term is up to $\cos^4\theta$ when $u^2+v^2=1$, and the parameter $\theta$ is by default $0.26\pi$ if not mentioned. Then we define the discrete kernel $b[n,m]$ of the ideal defocusing blur for an image sensor of a 100% fill factor:

$$b[n,m] = \int_{n-0.5}^{n+0.5}\int_{m-0.5}^{m+0.5}\frac{1}{r^2}a(\frac{x}{r},\frac{y}{r})dxdy,\ n,m\in\mathbb{Z}, \tag{4}$$

where we assume the spatial sampling period in $(x,y)$ as one and use brackets $[\cdot]$ to express a discrete function for clarity.

### B. View-Interpolation Defocusing Approximation

The parameter $N$ denotes the number of rendered views along the aperture radius such that around $\pi N^2$ rendered views are interpolated on a regular grid of $(u,v)$ with a

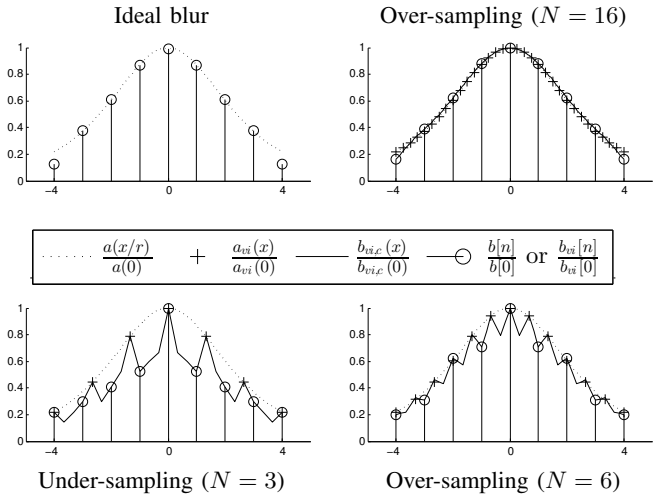



Fig. 4. Examples of view-interpolation defocusing filters for a defocus blur radius $r = 4$. The 1D case is presented for clarity. The responses are normalized in height for viewing. The discrete kernel $b[n]$ is shown for the ideal blur, and the approximated kernels $b_{vi}[n]$ for the cases with view interpolation.

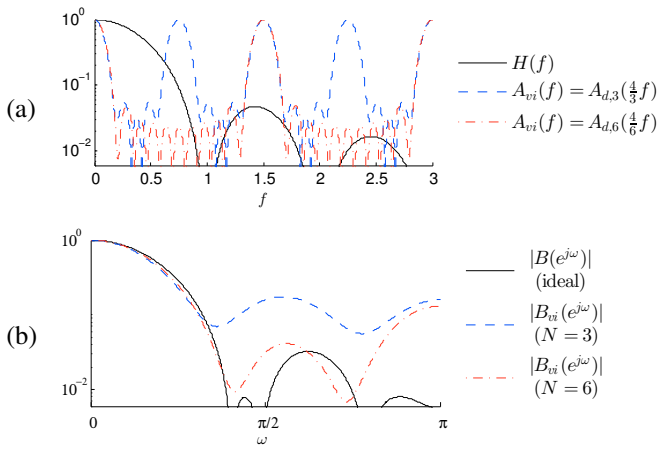



Fig. 5. Frequency responses of 1D view-interpolation defocusing blurs. $r = 4$ with under-sampled $N = 3$ and over-sampled $N = 6$. (a) Aliasing terms appear for $f > 0.5$ because $H(f) = \text{sinc}^2(f)$ is not the ideal box function, especially at where the view-interpolated $A_{vi}(f) = A_{d,N}(\frac{r}{N}f)$ has the peak replicas, i.e. $f = k\frac{N}{r}, k \in \mathbb{Z} \setminus \{0\}$. (b) Discrete defocus blur kernels. After discrete sampling, aliasing peaks locate near $\omega = 2\pi(k\frac{N}{r} - \lfloor k\frac{N}{r} \rfloor)$, especially for $k = \pm 1, \pm 2$ ($\omega = \frac{\pi}{2}$ and $\pi$ for $N = 3$, and $\omega = \pi$ for $N = 6$).

spacing of $\frac{u_t}{N}$. The corresponding defocusing blur is formed by three steps: view-interpolated $a_{vi}(x, y)$, spatial-interpolated $b_{vi,c}(x, y)$, and discretely sampled $b_{vi}[n, m]$. First, the regular-grid interpolated views correspond to a regular-grid point spread function $a_{vi}(x, y)$ with a spacing of $\frac{r}{N}$:

$$a_{vi}(x, y) = \sum_{i=-N}^{N} \sum_{j=-N}^{N} \delta(x - i\frac{r}{N}, y - j\frac{r}{N}) a_{d,N}[i, j], \quad (5)$$

where $a_{d,N}[i, j] \triangleq \frac{a(i/N, j/N)}{\sum_{i,j} a(i/N, j/N)}$ for normalizing the discrete aperture sampling. Second, an interpolation filter $h(x, y)$ is applied for continuous-discrete conversion:

$$b_{vi,c}(x, y) = h(x, y) * a_{vi}(x, y). \quad (6)$$

In this paper, we will use the triangular function for $h(x, y)$, i.e. bilinear interpolation, when an explicit expression is required. Finally, the discrete defocus kernel $b_{vi}[n, m]$ can be derived by discrete sampling and normalization:

$$b_{vi}[n, m] = \frac{b_{vi,c}(n, m)}{\sum_{n,m} b_{vi,c}(n, m)}. \quad (7)$$

When the interpolation number $N$ is much larger than the blur radius $r$, the resulting $b_{vi}[n, m]$ will be close to the ideal blur, e.g. $N = 16$ in Fig. 4. In contrast, an under-sampled $N < r$ will cause aliasing for insufficient view sampling. However, an over-sampled $N > r$ may also cause obvious aliasing due to the non-bandlimited $h(x, y)$, e.g. $N = 6$ in Fig. 4.

The aliasing effect can be further examined in frequency domain. According to (5), we have the equivalence for Fourier transforms: $A_{vi}(f_x, f_y) = A_{d,N}(\frac{r}{N}f_x, \frac{r}{N}f_y)$. Then the spatial interpolation in (6), $B_{vi,c}(f_x, f_y) = H(f_x, f_y)A_{vi}(f_x, f_y)$, introduces aliasing mainly due to the replicated main lobes of $A_{vi}(f_x, f_y)$ as shown in Fig. 5. The replicated peaks locate at $(f_x, f_y) = (k_x\frac{N}{r}, k_y\frac{N}{r})$ where $(k_x, k_y) \in \mathbb{Z}^2 \setminus \{(0, 0)\}$. As a result, the intensities of the aliasing terms mainly depend on two factors: the intensities of $H(f_x, f_y)$ at the replicated main-lobe peaks of $A_{vi}(f_x, f_y)$ and the main-lobe width of $A_{vi}(f_x, f_y)$.

### C. Selection Rule of View Interpolation Number $N$

In practice, most image blocks have more than one disparity value $d_p$. The aliasing effect of under-/over-sampling becomes inevitable because some $d_p$ exists such that $r \neq N$. Therefore, selecting $N$ is a tradeoff problem between quality and complexity. A simple rule to exclude under-sampling is $N = \lceil r_{max} \rceil$ where $r_{max}$ is the maximum blur radius in the target block. Instead, we allowed under-sampling for reducing computation, but the introduced aliasing is constrained with two criteria in terms of the sum of squared errors (SSE)

$$SSE_{vi} = \sum_{n,m} (b_{vi}[n, m] - b[n, m])^2. \quad (8)$$

Since the aliasing of over-sampling is mostly inevitable if there exists some $r < N$, its worst SSE is used as the first criterion. In addition, a sufficiently small threshold $\epsilon = 10^{-3}$ is chosen as the second criterion. From Fig. 6, the first criterion leads to a relaxed under-sampling rule, $N/r_{max} \geq 0.787$, which is valid only for $N \geq 4$ by the second criterion. Note that the worst over-sampled aliasing near $f = 1.5$ is consistent to the first side lobe of $H(f_x, f_y)$. For a continuous and non-decreasing transition to the simple rule for $N < 4$, we determined the selection rule as follows

$$N = \begin{cases} \lceil r_{max} \rceil, & r_{max} < 4 \\ 4, & 4 \leq r_{max} < \frac{4}{0.787} \\ \lceil 0.787 \cdot r_{max} \rceil, & \text{otherwise} \end{cases}. \quad (9)$$

For a large $r_{max}$, it saves 32% of rendered views compared to the simple rule with the same worst SSE due to aliasing.

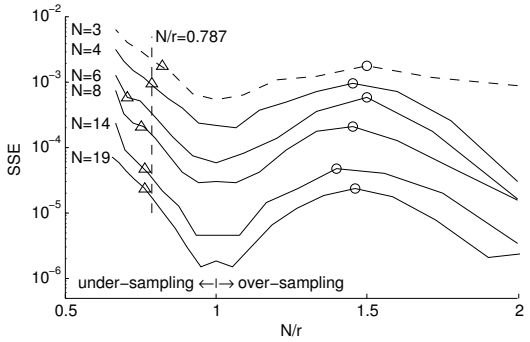



Fig. 6. SSE between $b_{vi}[n,m]$ and $b[n,m]$ for varying $N/r$. For each $N$, the 'o' marks the worst aliasing for over-sampling, and the '△' indicates the corresponding point for under-sampling with the same SSE.

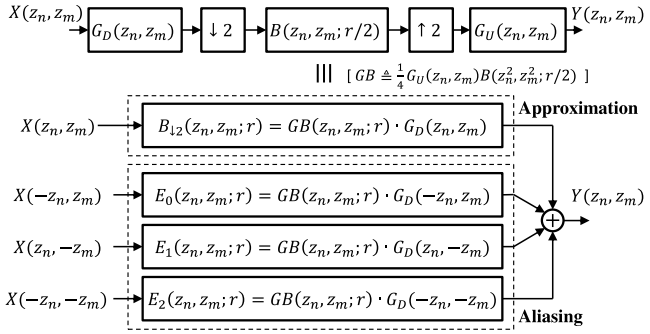



Fig. 7. Equivalent representations of the subsampled-by-two defocusing.

### D. Multi-Rate Defocusing Approximation

For a large blur, we can approximate it by refocusing in the subsampled-by-two layer with a half-size blur radius. This can reduce the refocusing complexity down to only 6.25% because the number of rendered views $V$ and image size $I$ in (1) both shrink to 25%. However, the equivalent defocusing blur will be different from the ideal one as shown in Fig. 7 where $G_D$ and $G_U$ denote the 2D downsampling and upsampling filters respectively. It consists of the defocus kernel $B_{\downarrow 2}(z_n, z_m; r)$ and the aliasing kernels $E_{i=0,1,2}(z_n, z_m; r)$. The former approximates the ideal blur based on the similarity between $B(z_n, z_m; r)$ and the lowpass filtered $B(z_n^2, z_m^2; r/2)$, and its accuracy is assessed by

$$SSE_{\downarrow 2} = \sum_{n,m} (b_{\downarrow 2}[n,m] - b[n,m])^2. \tag{10}$$

The latter part introduces aliasing terms through the bandpass filtered $B(z_n^2, z_m^2; r/2)$, and its effect is evaluated by

$$SSE_a = \sum_{i=0,1,2} \sum_{n,m} e_i^2[n,m]. \tag{11}$$

These two SSE numbers will get smaller when the lowpass band of $B(z_n^2, z_m^2; r/2)$ becomes narrower with an increasing $r$ as shown in Fig. 8. Note that the approximated $b_{\downarrow 2}[n,m]$ has a larger support and smaller high-frequency components than the ideal $b[n,m]$ due to the cascade of $G_D$ and $G_U$.

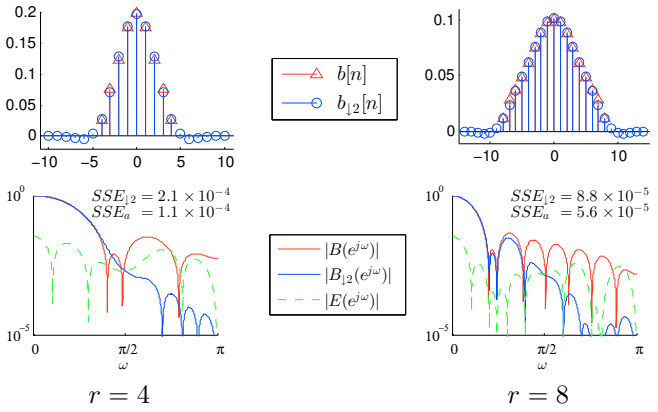



Fig. 8. Examples of subsampled-by-two defocusing blurs. The 1D case is presented for clarity, so there is only one aliasing kernel $E(z)$.

TABLE I
OPTIMIZED 2D FILTER PARAMETERS FOR SUBSAMPLED DEFOCUSING

| $r$ | $\beta_D$ | $\theta$ | $SSE_{\downarrow 2}$ | $SSE_a$ |
|---|---|---|---|---|
| 2.0 | 0.00 | $0.19\pi$ | $4.3 \times 10^{-3}$ | $1.4 \times 10^{-3}$ |
| 2.5 | 0.61 | $0.26\pi$ | $1.6 \times 10^{-3}$ | $7.4 \times 10^{-4}$ |
| **3.0** | **1.28** | $\mathbf{0.26\pi}$ | $\mathbf{6.7 \times 10^{-4}}$ | $\mathbf{1.4 \times 10^{-4}}$ |
| 3.5 | 1.44 | $0.27\pi$ | $3.0 \times 10^{-4}$ | $5.3 \times 10^{-5}$ |
| 4.0 | 1.49 | $0.28\pi$ | $1.3 \times 10^{-4}$ | $3.0 \times 10^{-5}$ |

### E. Filter Design for Multi-Rate Defocusing and Selection Rule of Subsampled Layer $L$

For efficient implementation, we adopted separable $G_D$ and $G_U$, so only their 1D kernels are considered. Short taps are preferred for supporting small $r$, so we used seven-tap Kaiser windows of parameters $\beta_D$ and $\beta_U$ to multiply the half-band $\mathrm{sinc}(x/2)$. To avoid high-frequency artifacts after upsampling, we set $\beta_U = 2.17$ such that $G_U(-1) = 0$. Then we jointly optimized the lowpass filter $G_D$ and the aperture filter $a(u,v;\theta)$ by minimizing $SSE_{\downarrow 2}$ for each $r$. The result is shown in Table I, and the accuracy gets better for larger $r$ as expected. Finally, we used the same threshold $\epsilon = 10^{-3}$ to determine if the subsampled defocusing can be applied. As a result, $r = 3$ was selected as the criterion, and the optimized parameters were chosen as $\beta_D = 1.28$ and $\theta = 0.26\pi$.

Let $r_{min}$ be the minimum blur radius in the target block. Then we determined the selection rule of the subsampled layer $L$, i.e. $\downarrow 2^L$, as follows

$$L = \begin{cases} 2, & r_{min}/2 \geq 3 \\ 1, & 3 \leq r_{min} < 6 \\ 0, & \text{otherwise} \end{cases}, \tag{12}$$

where subsampling-by-four ($L = 2$) is simply a cascade of subsampling-by-two ($L = 1$), and at most two subsampled layers are allowed in this paper. For purely defocused regions of sufficiently large $r_{min}$, the multi-rate refocusing can effectively accelerate by up to 16x ($L = 1$) or 256x ($L = 2$).

## IV. Fast View Interpolation for Star-Shaped Light Fields

For infocus-defocus hybrid blocks with a small $r_{min}$ and a large $r_{max}$, thousands of rendered views could be required to

retain photorealistic quality. This defies the usage of complex view interpolation methods which mostly have sophisticated hole filling and irregular memory access. In the following, we will introduce our computation-efficient algorithm which has simple 1D line-scan operations and efficient extension to a 2D aperture for sparse star-shaped light fields.

### A. Center-Based 1D Line-Scan View Interpolation

Using conventional methods has two complexity issues. First, they usually need disparity maps for all the captured views, which increases storage requirement and computing resource. Second, they use complex hole filling to enhance visual quality for occluded pixels which, however, will be averaged for refocusing and thus barely visible. For addressing these issues, we devised a center-based line-scan method specifically for refocusing. It uses only the center disparity map and applies simple hole filling.

Consider the view interpolation at horizontal grids which is performed line-by-line for each rendered view. The center view at $u = 0$ and a captured right view at $u = u_r > 0$ are used to interpolate rendered views at $u \in (0, u_r]$. The center-based line scan consists of two major steps as shown in Fig. 9: 1) implicitly shifting the right view for the target $d_t$ by changing pixel indexes without additional computation; 2) scanning the pixels in the center view from right to left in one single pass without z-buffering, as the $i_0$ to $i_8$ in Fig. 9, to calculate rendered pixels by forward warping and hole filling. Note that the case for left views $u < 0$ can be derived by symmetry.

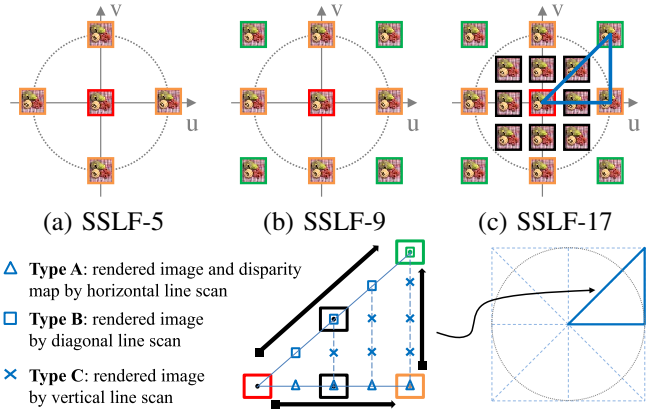
The first step above is for generating rendered pixels at integer positions such that they can be directly averaged for refocusing without additional pixel shifts. The second step simplifies the view interpolation in two respects. First, the forward warping always copies pixels from the center view (red point) for sharp infocused images. Second, for the hole filling we simply assume that the occluded region (green cross) is connected to its right neighbors, i.e. its disparity is estimated from the closest non-occluded pixel on the right.

*Quality assessment.* Fig. 10 shows an example of refocusing with a 1D aperture for clarity. The proposed line-scan method sometimes causes streak artifacts in the rendered views when the assumption for hole filling does not hold. However, the refocused image shows no obvious trace because the streaks all appear in the occluded regions and are averaged for refocusing with the objects occluding them.

*Speed assessment.* Our 1-D line-scan implementation is written in C++ and compared to the benchmark VSRS-1D-Fast in 3D-HTM 10.0 [25] which is optimized as an efficient view synthesis tool. Note that for each rendered pixel VSRS-1D-Fast needs to generate eight additional pixels because of the upsample-by-four processing and the two generated views from left and right viewpoints. Also, it requires many additional operations, including upsampling, downsampling, and reliability-based combination. In contrast, our line-scan method uses simple operations for each rendered pixel directly. Table II compares the computation throughputs for selected 5×1 light field subsets (detailed light-field information given in Table III). For VSRS-1D-Fast, we generated the corresponding 5×1 disparity maps in advance and interpolated each rendered view using the closest two available views. Since our software processes the RGB format (one pixel has three samples) and VSRS-1D-Fast works on the YUV 4:2:0 format, the throughput is normalized to sample/s for comparison. The line-scan method shows a stable speed gain across light fields of different characteristics. In summary, it can provide at least 30x speedup with good refocusing quality compared to VSRS-1D-Fast.

TABLE II
SPEED COMPARISON FOR 1D VIEW INTERPOLATION WITH $N = 16$ USING 5×1 LIGHT FIELD SUBSETS (SINGLE-THREAD 3.4 GHZ CPU)

| Light Field | Center-based Line Scan (Msample/s) | VSRS-1D-Fast (Msample/s) | Throughput Ratio |
|---|---|---|---|
| *Tsukuba* | 210.3 | 5.4 | 39.0x |
| *Lego Knights* | 202.5 | 6.5 | 30.9x |
| *Kid and Cat* | 205.0 | 6.6 | 31.0x |
| *Kid and Ball* | 201.9 | 6.6 | 30.6x |




(a) SSLF-5 (b) SSLF-9 (c) SSLF-17

(d) 2D aperture extension for SSLF-17

Fig. 11. Star-shaped light field (SSLF) configurations and 2D aperture extension from the line-scan view interpolation. (a) 5 views: center view at $(u, v) = (0, 0)$ and four views at $(\pm 1, 0)$ and $(0, \pm 1)$. (b) 9 views: four additional views at $(\pm 1, \pm 1)$. (c) 17 views: eight additional views at $(\pm 0.5, 0)$, $(0, \pm 0.5)$ and $(\pm 0.5, \pm 0.5)$. (d) Extends the 1D line scan to a triangular aperture using three different view types. A full 2D aperture is then composed of eight such triangles by symmetry.

### B. 2D Aperture Extension for Star-Shaped Light Fields

The above line-scan method has two benefits: simple operation and regular data access. For preserving these benefits when extending to a 2D aperture, we targeted the star-shaped light fields (SSLF) as shown in Fig. 11 where SSLF-5/-9/-17 denote the configurations of five, nine and seventeen views respectively. Regarding view interpolation for vertical or diagonal parallax, we rotated the images by 90° or 45° such that the parallax becomes horizontal and then applied the same 1-D line scan in Fig. 10. Note that for 45° rotation we generate a new image in which each horizontal line includes one corresponding diagonal line of the original image. Therefore, all rotation operations in this paper are lossless.

For SSLF-17, the center-based line scan is first extended for a triangular aperture with three different types of rendered

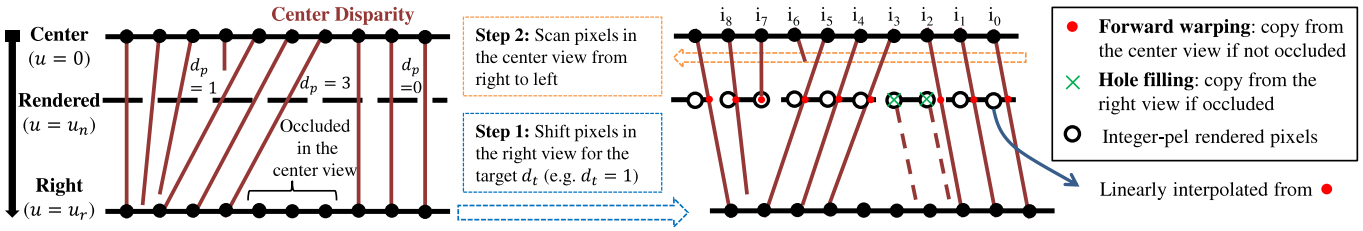



Fig. 9. Center-based 1D line-scan view interpolation using the center view at $u = 0$ and the right view at $u = u_r > 0$. For intermediate rendered views between them (on the trajectory of the solid black line on the left side), only the disparity of the center view is required. Such asymmetry is represented by the endpoints of the solid line: the square mark is for the center view and the arrow for the right view.

Refocusing by the center-based line-scan method

Refocusing by VSRS-1D-Fast

Fig. 10. Horizontal 1D aperture refocusing by view interpolation for the 5×1 light field subset of *Tsukuba*; $r = 19.5$ and $N = 16$ (left: refocused image; middle: cropped image; right: cropped rendered view at $u = -11/16$). Both refocused images show good quality without obvious artifacts. The proposed line-scan could generate rendered views with streak artifacts, as highlighted in the red circles, but they get blurred and unnoticeable in the refocused image.

views. Type A is 1D interpolated from the center and right views, and the disparity map is also generated (in a similar way) for later use. Type B is 1D interpolated from the center and top-right views using diagonal parallax. Then Type C is generated from Type A (as the center view) and Type B using vertical parallax, which fills the triangular aperture. Finally, a full 2D square aperture can be composed of eight symmetric triangles, and any aperture shape can be formed by skipping unnecessary views. For the lower-storage SSLF-9 and SSLF-5, we interpolated the missed views to form an SSLF-17 and then applied the same procedure as above.

## V. Block-Based Multi-Rate Refocusing

The conventional view-interpolation refocusing is applied on a frame-by-frame basis, and the parameter $N$ then depends on the maximum blur radius $r_{max}$ of the whole image. Therefore, the complexity is high and seems unnecessary in infocused regions that have small blurs. Therefore, we proposed a block-based processing flow to adapt to local blur statistics (similarly to the adaptive image blurring). Also, we included multi-rate layers to reduce computation for purely defocused regions with large $r_{min}$. Note that an extended border is required for each block to generate correct image pixels because outside pixels could splat their defocus blurs inwards. This computation overhead will also be considered.

The proposed processing flow is shown in Fig. 12 where five possible sizes of refocusing blocks (RB) are allowed (256×256−16×16). The whole image is processed by scanning the largest refocusing blocks (LRB) in a zig-zag scan order. The block-based refocusing is applied to the most computation-efficient LRB partitions. In the following, we will introduce the key components of the proposed flow: variable-size block analysis, timing-based partitioning, and layer-dependent deblocking.

### A. Variable-Size Block Analysis

For estimating the complexity of each RB, we need to know its refocusing parameters: the extended border width $W$ and height $H$, subsampled layer $L$, and interpolation number $N$. However, there is a complexity tradeoff between searching the extended borders and the refocusing based on them. By examining each pixel in a sufficiently large neighborhood of each RB, the smallest borders can be found but the complexity is high. On the contrary, the whole-image $r_{max}$ can be used for a quick guess, but the over-estimated borders cause significant computation overhead for refocusing. To balance the tradeoff, we proposed a fast border search method as follows.

Our method has two ingredients to accelerate the search. The first one is an indirect block-by-block approach for examining the blur splatting distance. For example, for each block we can record its maximum inside border which can splat to the right block and the maximum splatting distance to the right as shown in Fig. 13(b). Then the blocks on its right can use this information to find out their left extended border. Therefore, each target block can determine its extended border efficiently by aggregating such information from four directions. The key of acceleration here is that only two numbers are used to summarize the splatting effect for each block boundary. However, this also causes ambiguity of the border, and the resolution is the block size used for the examination. In this paper, we set the size to 8×8 for its balanced performance. The second ingredient for acceleration is the hierarchical merging shown in Fig. 13(c). In this way, the borders of all the RBs at the layer $L = 0$ can be determined recursively based on those of the 8×8 blocks.

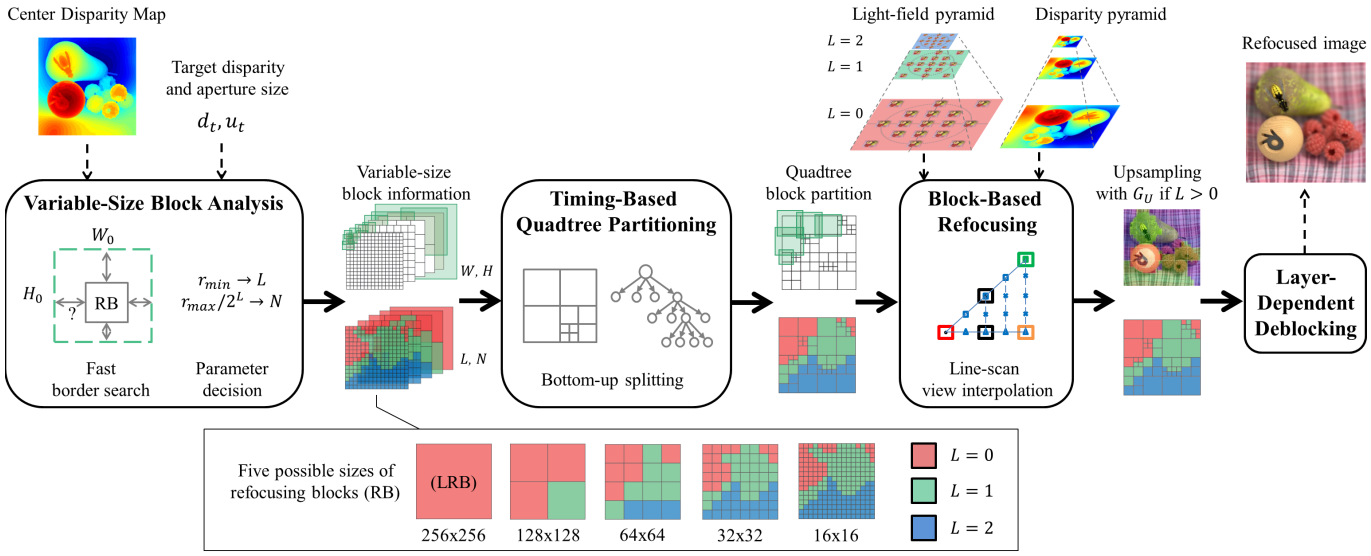



Fig. 12. Proposed block-based multi-rate refocusing flow. $W_0$ and $H_0$ denote the extended block width and height for layer $L = 0$.

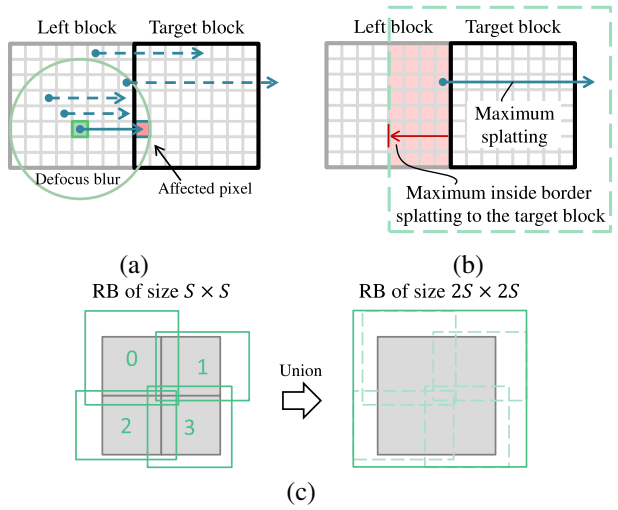



Fig. 13. Border search for RBs. (a) Defocus blurs from other blocks may affect the target block. (b) The extended border of the target block equals to the maximum inside border of neighbor blocks. (c) The border of a large RB can be constructed by its four corresponding half-size RBs.

Given the border of each RB, the subsampled layer $L$ is derived by substituting the local $r_{min}$ to (12). Then the interpolation number $N$ is derived by substituting the local maximum blur radius at the layer $L$, $r_{max}/2^L$, to (9). Finally, the extended block border width $W$ and height $H$ at the layer $L$ is derived by the previously determined border at the layer $L = 0$.

### B. Timing-Based Quadtree Block Partitioning

Larger RBs tend to include infocus-defocus hybrid boundaries which cause significant computation loading. In contrast, smaller RBs use local statistics more adaptively to save computation but suffer more from the overhead of the extended borders. To find the best LRB partition, we proposed to first estimate the computation complexity of each RB and then perform quadtree partition optimization.

For simplifying the estimation, we assumed the complexity of the 1D line-scan view interpolation is proportional to the total number of rendered pixels $P_S$:

$$P_S = W_S \cdot H_S \cdot V_S, \tag{13}$$

where $W_S$ and $H_S$ are the effective border width and height, and $V_S$ is the number of rendered views. This assumption can be verified by the results in Table II which shows the throughput of the 1D line scan is stable for different light fields. Then for 2D-aperture refocusing we considered the three different types of rendered views separately. A multiple linear model was used to estimate the execution time $t$ for one RB by

$$t = k_A P_A + k_B P_B + k_C P_C, \tag{14}$$

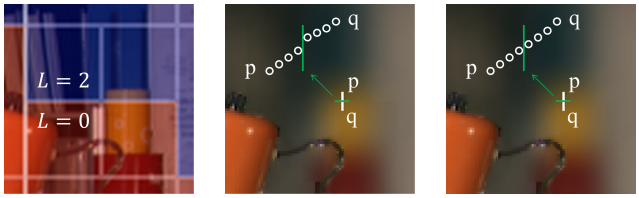
where the three variables, $P_{S=A,B,C}$, denote the total numbers of rendered pixels for the Type A, B, and C rendered views, and the $k_{S=A,B,C}$ are the corresponding parameters. Note that the run-time overhead of upsampling was found negligible compared to view interpolation and thus ignored in this model.

For multiple regression analysis, we performed block-based refocusing on the light field *Tsukuba* with many configurations: $(W, H)$ ranges from $(8, 8)$ to $(184, 184)$, and $N$ from 1 to 16. The measured run times were used for the analysis, and the parameters were derived as ($R^2 = 0.998$)

$$k_A = 95.5, k_B = 8.3, k_C = 15.3 \text{ (ns/pixel)}. \tag{15}$$

To verify the accuracy, we also applied these parameters to estimate the run times of the same configurations for light fields *Lego Knights* and *StillLife*. The tested $R^2$ values are 0.999 and 0.998 respectively, which shows the robustness of this model for light fields of different characteristics.

The complexity of each RB can then be estimated using the linear model and the derived $P_{A,B,C}$ from its refocusing




(a) (b) (c)

Fig. 14. Block effect for mismatched subsampled layers $L$. (a) Partitioned RBs: red for $L = 0$ and blue for $L = 2$. (b) Block artifacts at boundaries between different $L$, especially around the yellow can. The white circles denote the pixel intensities across such a boundary. (c) Block artifacts removed by the proposed deblocking filter.

parameters. The quadtree partition of one LRB is optimized by bottom-up splitting from 32×32 to 256×256. For each 32×32 RB, it will split into its four child 16×16 RBs if its complexity is larger than the sum of those of its child RBs; otherwise, it will be retained. Such RB splitting is performed recursively until the LRB 256×256 is reached. Note that the variable-size block analysis and this quadtree partitioning only induce negligible computation overheads.

### C. Layer-Dependent Deblocking Filter

Mismatched $L$ and $N$ at RB boundaries could lead to block artifacts. This possibility has been reduced by limiting the difference between defocusing filters with the small threshold $\epsilon$ when choosing them. However, the longer taps of the subsampled defocusing filters, as indicated in Fig. 8, could still introduce visible abrupt changes at boundaries. Fig. 14 shows an example for such artifacts with mismatched $L$.

Inspired by the adaptive deblocking filter in H.264/AVC [31], we used a similar deblocking flow which handles horizontal boundaries first and then vertical ones. For alleviating the abrupt changes, we devised a deblocking filter with linear interpolation as shown in Fig. 15 where $\{p_3, p_2, p_1, p_0, q_0, q_1, q_2, q_3\}$ denote the boundary pixels on two sides. In the following, we assume the RB containing the pixels $p_i$ uses a higher subsampled layer, i.e. $L_p \geq L_q$; for $L_p < L_q$, the process is similar by symmetry. A strong filter is used to smooth the longer filter taps for $L_p = 2$ by applying linear interpolation between pixels $p_0$ and $q_3$. To avoid affecting natural edges, it is applied only if the edge is considered artificial when all following conditions hold

$$|p_{0,c} - q_{3,c}| < \beta, \;\; c \in \{r, g, b\}, \tag{16}$$

where the subscripts $r/g/b$ denote the R/G/B components, and the value of $\beta$ is set to 35 in this paper. Similarly, a weak filter could be applied between $p_0$ and $q_1$ with conditions:

$$|p_{0,c} - q_{1,c}| < \beta, \;\; c \in \{r, g, b\}. \tag{17}$$

With the proposed layer-dependent deblocking filter, the block artifacts can be greatly reduced as shown in Fig. 14(c).

## VI. Experimental Results

Subjective assessment will be presented first to confirm the photorealistic refocusing quality for using view interpolation. Then objective comparisons in terms of peak signal-to-noise ratio (PSNR) and computing speed will be given to evaluate the performance of the proposed refocusing method. The block-based multi-rate flow will also be profiled in detail to break down the quality-complexity tradeoff. In addition, more refocusing results presented in video are available online[1].

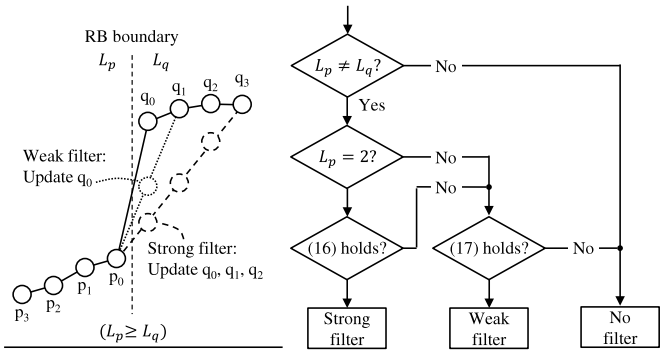



Fig. 15. Layer-dependent deblocking filter using linear interpolation. Left: Example of boundary pixels for which the left RB is refocused at a higher layer ($L_p$) than the right ($L_q$). Right: Decision flow for the deblocking filter.

### A. Test Configurations

Three variants of the proposed method were used:

1) **Block-based**: using the block-based multi-rate flow on SSLF-17(default)/-9/-5;
2) **Frame-based**: using frame-based view interpolation on SSLF-17 with $N$ selected by (9) for purpose of comparison;
3) **Benchmark**: using frame-based view interpolation with $N = 2 \cdot \lceil r_{max} \rceil$ as an objective quality benchmark.

We implemented them all in C++, and the programs ran on a single-thread 3.4 GHz CPU with 8 MB cache. The image blurring method was also implemented for comparison.

Fourteen light fields, as summarized in Table III, were tested, and the center views and disparity maps are shown in Fig. 16. Note that the disparity range, which determines the largest $r_{max}$, is a key factor for the computation complexity. In addition to standard datasets, two of the light fields were captured by the DragonFly camera in [16], four by the commercial Lytro ILLUM camera, and two rendered by the ray-tracing software pbrt[2]. For detailed realisticity, we used center disparity maps of sub-pixel precision. If not available, we derived them by applying multi-baseline stereo matching to the light fields.

### B. Subjective Quality

*1) Comparison with ray tracing:* The pbrt source code was modified to use the aperture filter derived in Section III-E. To generate refocused images as ground truth, we rendered 256 and 512 samples per pixel for *DOF-Dragons* and *Microcity* respectively. The results are shown in Fig. 17. The refocused images by the proposed method show very similar visual quality to the ground truth images, while those by the image blurring have obvious boundary artifacts.

[1] http://www.ee.nthu.edu.tw/chaotsung/blockbased_refocus
[2] http://www.pbrt.org

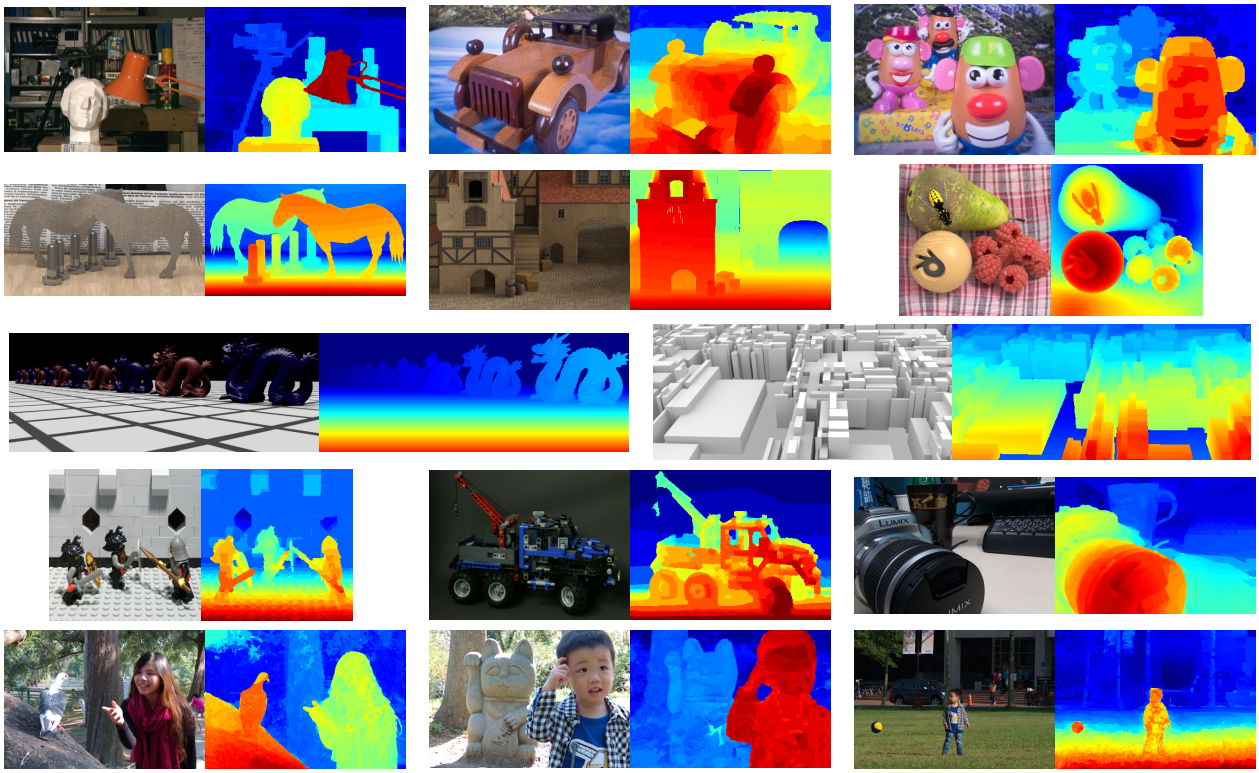


Fig. 16. Center views and disparity maps (in MATLAB jet color table) of the tested light fields. First row: *Tsukuba*, *Car*, and *Potato*. Second row: *Horses*, *Medieval*, and *StillLife*. Third row: *DOF-Dragons* and *Microcity*. Fourth row: *Lego Knights*, *Lego Truck*, and *Camera*. Fifth row: *Girl and Pigeon*, *Kid and Cat*, and *Kid and Ball*.

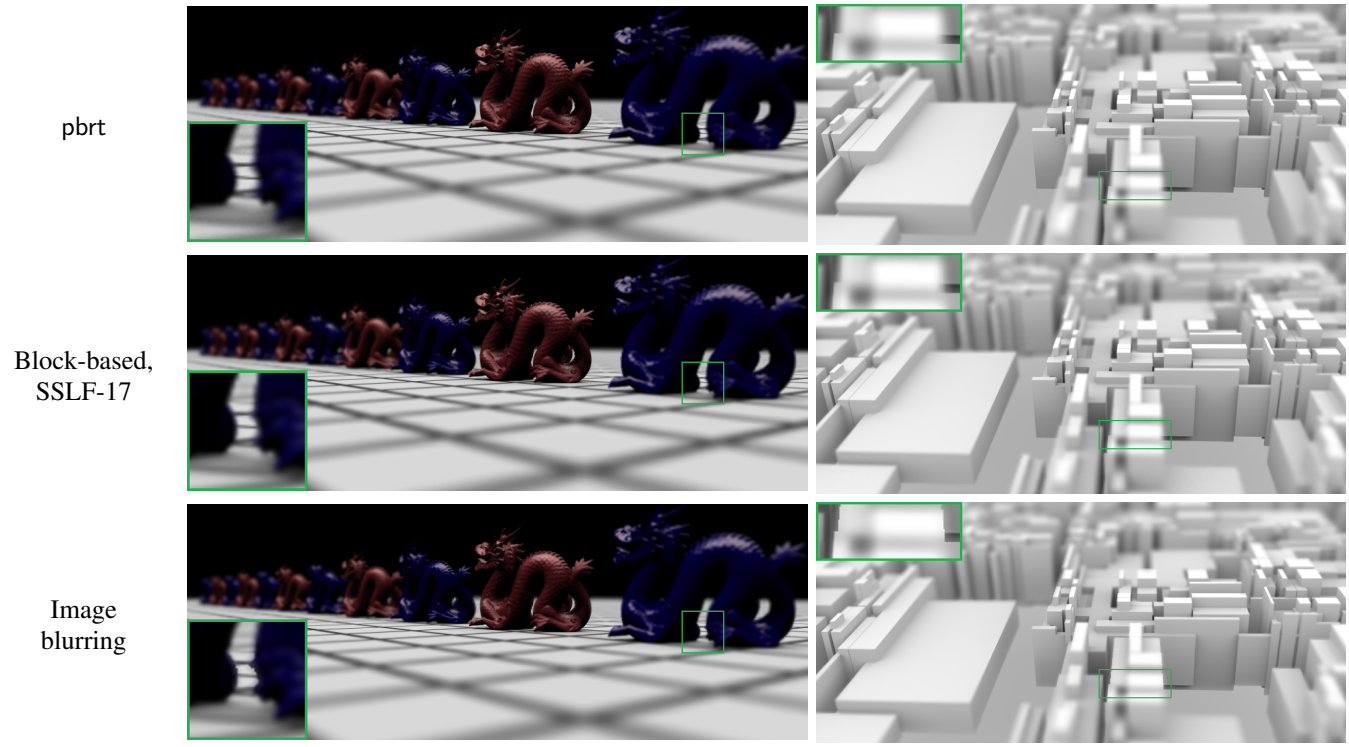



Fig. 17. Refocused images $(u_t = 1, d_t = 0)$ for *DOF-Dragons* (left column) and *Microcity* (right column). The PSNR values for the block-based refocusing and the image blurring are 39.9 dB and 38.1 dB respectively for *DOF-Dragons*, and 43.0 dB and 39.2 dB respectively for *Microcity*. These PSNR values are calculated with respect to the pbrt ground-truth images.

TABLE III
LIGHT FIELD TEST SET

| Light Field | Width | Height | Disparity Range | Source | Disparity Map |
|---|---|---|---|---|---|
| *Tsukuba* | 384 | 288 | 8.5 – 29.0 | Tsukuba[a] | Stereo matching |
| *Car* | 400 | 304 | 0.0 – 16.0 | DragonFly | Stereo matching |
| *Potato* | 400 | 304 | 1.5 – 10.0 | DragonFly | Stereo matching |
| *Horses* | 1024 | 576 | -5.0 – 3.5 | HCI[b] | HCI |
| *Medieval* | 1024 | 720 | -8.0 – 3.5 | HCI | HCI |
| *StillLife* | 768 | 768 | -11.0 – 10.0 | HCI | HCI + WMF[c] |
| *DOF-Dragons* | 1000 | 424 | -8.0 – 30.0 | pbrt[d] | pbrt |
| *Microcity* | 982 | 486 | -20.0 – 18.0 | pbrt[e] | pbrt |
| *Lego Knights* | 1024 | 1024 | -14.0 – 9.0 | Standford[f] | Stereo matching |
| *Lego Truck* | 1280 | 960 | -11.0 – 9.0 | Standford | Stereo matching |
| *Camera* | 2164 | 1504 | -24.0 – 26.5 | ILLUM | PT[g] + Upscale[h] |
| *Girl and Pigeon* | 2164 | 1504 | -7.0 – 14.5 | ILLUM | PT + Upscale |
| *Kid and Cat* | 2164 | 1504 | -20.0 – 20.0 | ILLUM | PT + Upscale + WMF |
| *Kid and Ball* | 2164 | 1504 | -8.0 – 10.0 | ILLUM | PT + Upscale + WMF |

[a] Tsukuba (http://vision.middlebury.edu/stereo/data/scenes2001/data/tsukuba/).
[b] HCI light field dataset [15] (http://hci.iwr.uni-heidelberg.de/HCI/Research/LightField/).
[c] Weighted median filter (WMF) to enhance disparity maps as discussed in Section VII.
[d] dof-dragons.pbrt in http://www.pbrt.org/scenes.html (256 low-discrepancy samples per pixel).
[e] microcity.pbrt in http://www.pbrt.org/scenes.html (512 low-discrepancy samples per pixel).
[f] Standford light field archieve (http://lightfield.stanford.edu/).
[g] Lytro Power Tools Beta (https://www.lytro.com/imaging/power-tools).
[h] Upscaling by two with a weighted mode filter.

*2) Comparison with TLFR:* The TLFR source code[3] provided by the authors was used for comparison. For using the software, we converted the SSLF-17 and the disparity map into the required format with 17 samples per pixel. The proposed refocusing can achieve similar depth-of-field effect with much less computation as shown in Fig. 2. In particular, the TLFR is designed for stochastically sampled inputs but not for the regularly and sparsely sampled light fields as our target system. For the challenging scene *Tsukuba* which has complex occlusion near thin objects, it introduces obvious artifacts near the lamp stand as shown in Fig. 18. The image blurring method also has serious artifacts for it.

*3) Comparison with real pictures:* We captured photos using a Panasonic GF2 camera with a similar field of view of the light field *Car* and then refocused the light field with a similar-size aperture ($u_t = 0.43$). The images are compared in Fig. 19 which shows that the proposed refocusing can deliver photorealistic depth-of-field effect as a real camera.

*4) Comparison with Lytro Desktop:* We compared our refocusing results to the commercial software Lytro Desktop[4] in Fig. 20. While the Lytro Desktop applies complex algorithms on the dense light fields captured by ILLUM, the proposed method can achieve similar quality using sparse SSLFs, especially for the infocus-defocus hybrid regions. Note that some artifacts appear near object boundaries when the disparity is not accurate enough, e.g. the cheek of the kid in Fig. 20.

### C. PSNR Comparison (w.r.t. Ray Tracing)

In the following, the distortions induced by the fast refocusing methods will be evaluated in terms of PSNR. Compared to the pbrt ground-truth images for *DOF-Dragons* and *Microcity*, the frame-based refocusing can achieve above 42 dB and the block-based refocusing mostly above 40 dB as shown in Fig. 21. The image blurring method is also compared and constantly has the worst PSNR.

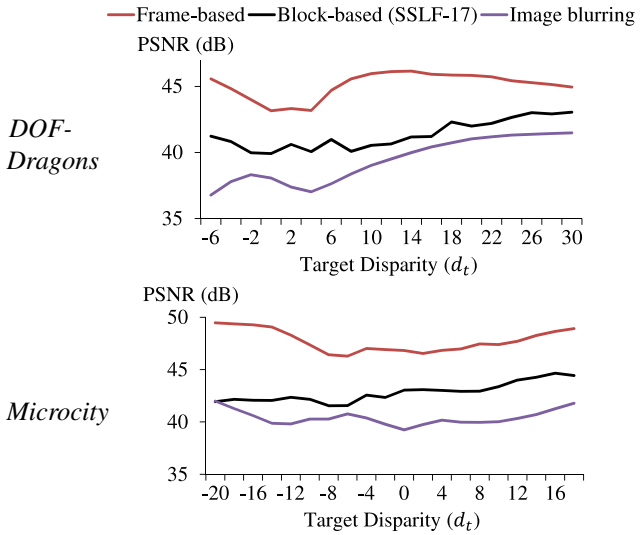



Fig. 21. PSNR (w.r.t. pbrt) vs. target disparity ($d_t$) for refocused images ($u_t = 1$) of light fields *DOF-Dragons* and *Microcity*.

### D. PSNR Comparison (w.r.t. Benchmark Configuration)

The ground-truth refocused images for other tested light fields are not available. For purpose of general evaluation, we used the benchmark configuration in Section VI-A as the basis for calculating PSNR in the rest of this paper. This choice is based on its great quality at object boundaries and nearly ideal blurring for defocused regions.

Detailed results for three selected light fields are shown in Fig. 22. The PSNR values of the frame-based refocusing are far above 50 dB; therefore, the selection rule of $N$ in (9) causes only negligible distortions. Regarding the block-based refocusing on SSLF-17, the PSNR values are mostly above 45 dB, and this shows that the subsampled refocusing delivers very similar results compared to the original one. For the image blurring method, boundary artifacts contribute most of the distortions. Therefore, its quality is highly related to disparity characteristics of the tested scenes. For example, the PSNR is below 35 dB for *Tsukuba* which has complex occlusion but can be above 45 dB for *Camera* which has smooth depth transition.

The cases of applying block-based refocusing on the low-storage SSLF-9 and SSLF-5 were also tested. The PSNR drops compared to SSLF-17 are shown at the bottom of Fig. 22. The results are also scene-dependent due to the similar reason for the image blurring, i.e. lacking of background information for correct hole filling. In particular, the worst cases occur when highly occluded background or objects are infocused. For example, refocusing at $d_t = 9.5$ (background) for the SSLF-5 of *Tsukuba* leads to a 1.8 dB drop; but it is hard to tell as shown in the bottom row of Fig. 18.

[3] http://groups.csail.mit.edu/graphics/tlfr
[4] https://illum.lytro.com/desktop

TLFR | Block-based, SSLF-17 | Block-based, SSLF-5 | Image blurring

Fig. 18. Refocused images ($u_t = 1$) for *Tsukuba* at the lamp (top row, $d_t = 27.5$) and the background (bottom row, $d_t = 9.5$).

GF2, 18-mm, f/4.3 (focused at foreground) | Block-based, $u_t = 0.43$ (focused at foreground) | GF2, 18-mm, f/4.3 (focused at background) | Block-based, $u_t = 0.43$) (focused at background)

Fig. 19. Similar depth-of-field effect for real pictures captured by GF2 and refocused images by block-based refocusing on *Car*.

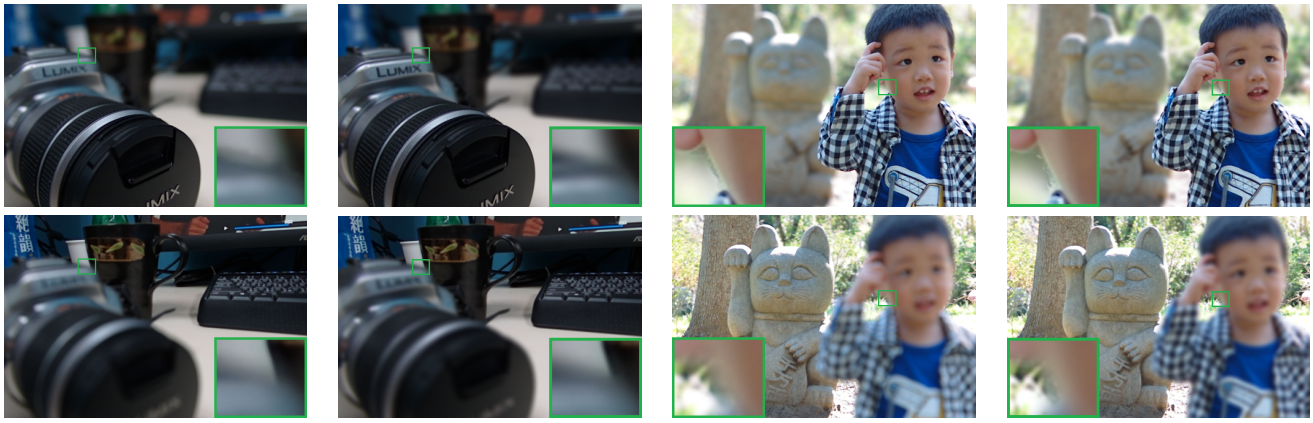
To summarize the scene-/disparity-dependent behaviors in few numbers, we calculated the expected values of PSNR over the distributions of the target disparity values $d_t$ for each scene. The frequency weighting can highlight the quality derived by focusing at major objects in the scene, e.g. the kid or cat in *Kid and Cat*. Also, it can remove outlier cases in which nothing is infocused or simply the $d_t$ is out of the disparity range. The results are summarized in the left columns of Table IV. The average numbers over the fourteen tested light fields are also given for different apertures sizes ($u_t = 1, 0.5, 0.25$), and the block-based refocusing shows stable and great performance constantly. Note that on average the SSLF-5 delivers very similar results compared to SSLF-9, which indicates a strong incentive to use SSLF-5 for saving storage.

### E. Speed Comparison

Fig. 23 shows the run-time details for the three selected light fields. The run times of the frame-based refocusing and image blurring both appear in convex lines along the target disparity. This reflects the fact that their computation complexity is quadratically proportional to the (maximum) defocus blur radius. In contrast, the results for the block-based refocusing are highly dependent on the disparity and scene characteristics. Note that the run times for SSLF-9 and SSLF-5 are nearly the same as those for SSLF-17 because the additional computation is merely interpolating few views.

For the *Camera* with smooth transition, the infocused RBs have small interpolation numbers $N$ and the defocused ones are mostly accelerated by subsampled refocusing at $L = 2$. Therefore, the speedup over the frame-based refocusing is quite significant and up to 34.4x on average. For the *Kid and Cat*, many RBs can be well accelerated as the case of *Camera*. But when focusing at the kid or cat, the RBs at infocus-defocus boundaries occupy lots of the computation time because $L = 0$ is assigned for the infocused parts and large $N$ is used for the large disparity difference. As a result, the speedup is 7.1x on average. Note that the block-based refocusing will run very fast if all of the objects are defocused, e.g. $d_t = 24$ for *Camera* and $d_t = 4$ for *Kid and Cat*, because nearly all RBs use $L = 2$ for acceleration. For a fair comparison, these cases are considered as outliers by using the expected values of run times over the distribution of $d_t$ to calculate the average numbers. For the complex scene *Tsukuba*, the block-based refocusing only improves computing speed slightly because most of RBs belong to infocus-defocus hybrid regions.

The results of all the tested light fields are summarized in the middle columns of Table IV. Most of the computation time of the block-based refocusing is contributed by the RBs refocused at the original layer $L = 0$. Moreover, since $L = 2$ only occupies less than 7.2% of the run time on average, this suggests that $L = 2$ is sufficient and a higher $L = 3$ will not improve the speed too much. The speed advantage of the proposed block-based refocusing is demonstrated in terms



Lytro Desktop, f/2 | Block-based, $u_t = 1$ | Lytro Desktop, f/2 | Block-based, $u_t = 1$

Fig. 20. Refocusing results for Lytro Desktop on dense light fields and the proposed refocusing on sparse SSLF-17 of *Camera* (left) and *Kid and Cat* (right). The images in the top row are refocused at foreground, and the ones in the bottom row refocused at background.

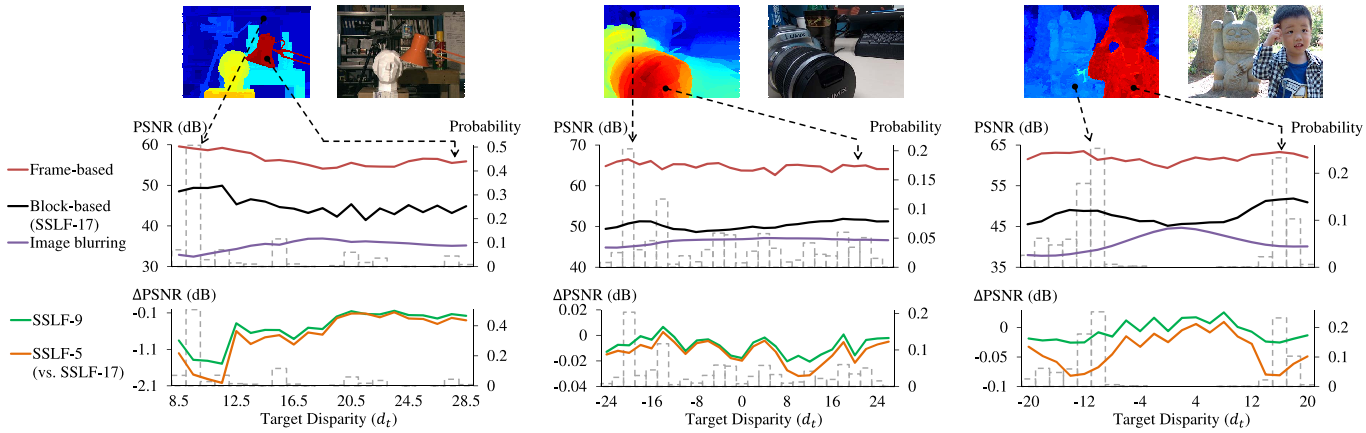



Fig. 22. PSNR (w.r.t. benchmark configuration) vs. target disparity $(d_t)$ for refocused images $(u_t = 1)$ of light fields *Tsukuba* (left), *Camera* (center), and*Kid and Cat* (right). The dotted histograms represent the possibility mass functions of discrete disparity values. The disparity maps in the top row are drawn using the MATLAB jet color table from blue (far) to red (near), and the curves in the bottom row are shown in terms of △PSNR for the block-based refocusing on SSLF-9 and SSLF-5.

TABLE IV
PSNR (W.R.T. BENCHMARK CONFIGURATION) AND SPEED COMPARISON

| Light Field | PSNR Comparison | | | | | Speed Comparison | | | | | | | | Block-Based Multi-Rate Flow Profiling | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Frame-Based | Image Blurring | Block-Based SSLF-17 | Block-Based SSLF-9 | Block-Based SSLF-5 | Frame-Based ($t_{FB}$) | Image Blurring ($t_{IB}$) | Block-Based ($t_{BB}$) | Run-Time Contribution to Block-Based Refocusing L=0 | L=1 | L=2 | Run-Time Ratio | | Frame-Based => Block-Based (SSLF-17, L=0) | | (L=0) => (L=0/1) | | (L=0/1) => (L=0/1/2) | | |
| | $\overline{PSNR}$ [a] (dB) | | | $\overline{\Delta PSNR}$ [b] (dB) | | $\bar{t}$ [c] (sec) | | | Percentage | | | $t_{FB}/t_{BB}$ | $t_{IB}/t_{BB}$ | Speed Gain | $\overline{PSNR}$ Drop | Speed Gain | $\overline{PSNR}$ Drop | Speed Gain | $\overline{PSNR}$ Drop | |
| *Tsukuba* | 58.0 | 33.6 | 47.7 | -1.0 | -1.3 | 1.7 | 0.2 | 1.1 | 92.0% | 4.9% | 3.1% | 1.5 | 0.2 | 1.0x | -3.5 dB | 1.4x | -3.8 dB | 1.0x | -2.9 dB | |
| *Car* | 56.9 | 39.7 | 45.8 | -0.1 | -0.1 | 1.0 | 0.2 | 0.3 | 76.1% | 19.0% | 4.9% | 3.3 | 0.6 | 1.1x | -2.8 dB | 2.7x | -6.3 dB | 1.1x | -2.1 dB | |
| *Potato* | 54.3 | 40.6 | 46.2 | -0.3 | -0.4 | 0.4 | 0.1 | 0.2 | 72.6% | 26.8% | 0.7% | 2.2 | 0.4 | 1.3x | -2.9 dB | 1.7x | -4.0 dB | 1.0x | -1.1 dB | |
| *Horses* | 54.7 | 41.6 | 47.0 | -0.3 | -0.2 | 2.8 | 0.3 | 0.8 | 80.6% | 18.4% | 1.0% | 3.7 | 0.4 | 2.2x | -3.6 dB | 1.6x | -3.9 dB | 1.0x | -0.2 dB | |
| *Medieval* | 57.1 | 45.0 | 50.1 | -0.1 | -0.1 | 5.3 | 0.4 | 0.7 | 69.1% | 29.2% | 1.7% | 7.5 | 0.6 | 3.3x | -4.2 dB | 2.2x | -2.6 dB | 1.0x | -0.1 dB | |
| *StillLife* | 57.6 | 41.3 | 46.1 | 0.0 | 0.0 | 9.4 | 1.5 | 1.3 | 75.7% | 17.1% | 7.3% | 7.1 | 1.2 | 1.7x | -4.2 dB | 3.5x | -5.3 dB | 1.2x | -2.0 dB | |
| *DOF-Dragons* | 59.2 | 39.6 | 43.5 | 0.0 | 0.0 | 21.6 | 4.0 | 1.6 | 66.3% | 18.9% | 14.8% | 13.7 | 2.5 | 1.3x | -2.2 dB | 6.2x | -8.3 dB | 1.7x | -5.2 dB | $u_t$=1 |
| *Microcity* | 61.2 | 40.9 | 45.2 | -0.1 | -0.1 | 19.5 | 2.4 | 3.3 | 78.2% | 14.9% | 6.9% | 5.8 | 0.7 | 1.5x | -3.3 dB | 3.2x | -8.7 dB | 1.2x | -4.0 dB | |
| *Lego Knights* | 60.6 | 39.3 | 49.2 | -0.2 | -0.2 | 23.0 | 2.9 | 2.1 | 78.1% | 13.6% | 8.3% | 10.9 | 1.4 | 2.6x | -5.0 dB | 3.4x | -3.9 dB | 1.2x | -2.5 dB | |
| *Lego Truck* | 59.7 | 44.7 | 50.8 | 0.0 | -0.1 | 24.4 | 5.2 | 2.8 | 81.8% | 10.6% | 7.6% | 8.7 | 1.9 | 1.8x | -2.7 dB | 3.8x | -3.5 dB | 1.3x | -2.7 dB | |
| *Camera* | 65.1 | 46.3 | 50.4 | 0.0 | 0.0 | 267.4 | 53.7 | 7.8 | 69.2% | 11.9% | 18.9% | 34.4 | 6.9 | 1.8x | -2.7 dB | 8.2x | -8.0 dB | 2.3x | -4.0 dB | |
| *Girl and Pigeon* | 60.1 | 43.9 | 49.1 | -0.1 | -0.1 | 64.4 | 7.9 | 5.8 | 80.3% | 11.0% | 8.6% | 11.1 | 1.4 | 2.4x | -3.3 dB | 3.5x | -6.1 dB | 1.3x | -1.6 dB | |
| *Kid and Cat* | 62.7 | 39.3 | 49.6 | 0.0 | -0.1 | 206.6 | 48.8 | 29.2 | 89.6% | 5.6% | 4.7% | 7.1 | 1.7 | 1.5x | -2.6 dB | 3.6x | -6.7 dB | 1.3x | -3.8 dB | |
| *Kid and Ball* | 62.3 | 48.7 | 49.5 | 0.0 | 0.0 | 55.2 | 7.0 | 2.8 | 71.7% | 16.3% | 12.0% | 19.4 | 2.5 | 2.6x | -3.2 dB | 5.1x | -6.2 dB | 1.5x | -3.4 dB | |
| Average ($u_t$=1) | 59.2 | 41.7 | 47.9 | -0.2 | -0.2 | 50.2 | 9.6 | 4.3 | 77.2% | 15.6% | 7.2% | 9.7 | 1.6 | 1.9x | -3.3 dB | 3.6x | -5.5 dB | 1.3x | -2.5 dB | |
| Average ($u_t$=0.5) | 57.8 | 44.0 | 49.8 | -0.4 | -0.4 | 17.2 | 2.6 | 1.5 | 78.0% | 16.2% | 5.7% | 8.5 | 1.2 | 2.2x | -2.9 dB | 2.9x | -3.7 dB | 1.2x | -1.4 dB | |
| Average ($u_t$=0.25) | 56.6 | 45.5 | 51.9 | -0.3 | -0.3 | 7.3 | 0.8 | 0.9 | 83.5% | 14.3% | 2.2% | 5.3 | 0.6 | 2.5x | -2.0 dB | 1.9x | -2.4 dB | 1.1x | -0.5 dB | |

[a] Expected value of PSNR over the distribution of the target disparity.

[b] Expected value of PSNR difference compard to SSLF-17.

[c] Expected value of run time over the distribution of the target disparity.

of the run time ratio. For the large aperture $u_t = 1$, it can achieve a 9.7x speedup on average compared to the frame-based refocusing and can be 1.6x as fast as the image blurring. The advantage becomes less for a smaller aperture, but in that case the run time is also faster and becomes a less important issue. Also, the speedup is more significant for light fields of higher resolution because it is easier to use larger RBs for saving computation. Therefore, the proposed block-based refocusing is in particular useful for the challenging task of large-aperture refocusing for high resolution images.

### F. Block-Based Multi-Rate Flow Profiling

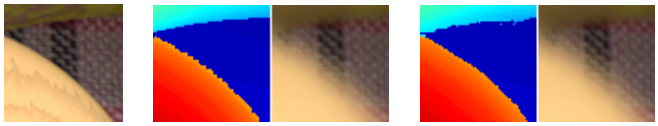
The performance of the proposed flow is profiled in the right columns of Table IV. Compared to the naive frame-based refocusing, the block-based refocusing adds three ingredients sequentially for acceleration. First, the block partitioning adapts to local blur statistics for infocused regions and provides a stable speed gain around 2x across different light fields and aperture sizes. Second, the subsampled-by-two refocusing ($L = 1$) accelerates defocused areas and boosts the most computation speed, e.g. up to 8.2x for *Camera* and 3.4x on average for $u_t = 1$. The speed gain becomes less for a smaller aperture due to smaller blurs. Finally, the further subsampled-by-four refocusing ($L = 2$) only helps when there are large blurs. The trend of PSNR drops is similar to the speed gains, which shows the quality-speed tradeoff for accelerating the refocusing.

## VII. Discussion

### A. Limitations

*Scene-dependent performance*. The proposed block-based refocusing can efficiently accelerate the processing of the scenes having smooth depth transition, such as *Camera* and *Kid and Ball*. The performance is less efficient for the scenes with distant objects or complex occlusion, like *Kid and Cat* and *Tsukuba*. The computation is dominated by the RBs at infocus-defocus boundaries which have large interpolation number $N$ and refocus at the original layer $L = 0$. Although the fast line-scan method was proposed for accelerating the view interpolation, these RBs still become the computation bottleneck.

*Disparity accuracy*. The image quality of the proposed method highly relies on accurate center-view disparity maps, especially for the object boundaries. Although stereo matching is beyond the scope of this paper, some boundary issues of the disparity maps need to be solved for photorealistic refocusing quality. Fig. 24 shows such an example. The HCI disparity map of *StillLife* was generated from a ground-truth 3D model, but it has zig-zag boundaries for the wooden ball. Some boundary pixels of the ball are assigned the wrong disparity values of the background curtain. As a result, the refocusing based on this disparity map causes those boundary pixels to be isolated in the defocused blurs of the ball. To address this issue by enhancing disparity edges between objects, we applied a bilateral weighted median filter for which the adaptive range weight is calculated by the pixel differences of hue and saturation in HSV domain. The resulting refocused images thus look more natural. For more details about how the disparity could affect refocusing quality, the reader is referred to [32].



Texture HCI disparity map WMF refined map

Fig. 24. Pixel isolation effect (cropped images of *StillLife*). The HCI disparity map has zig-zag boundaries for the wooden ball, and this results in some isolated pixels of the ball which should be defocused. After being filtered by a weighted median filter (WMF), the disparity map has a smooth boundary and thus reduces the artifacts.

*Bokeh*. The shallow depth-of-field effect on point light sources is considered as bokeh. In particular, good bokeh appears when the aperture filter is close to a uniform disc, e.g. long telephoto lenses. However, in this paper the aperture filter needs to be bandlimited to enable the subsampled refocusing. As a result, its filter kernel has fast fall-off boundaries; therefore, good bokeh cannot be simulated by the proposed multi-rate refocusing.

### B. Possible Extensions

In this paper, we introduce a block-based multi-rate processing flow for refocusing, and this system can be extended for different purposes. For example, the image blurring can replace the view interpolation for an ultra fast refocusing system. Or the lowpass filters $G_D$ and $G_U$ can be redesigned for an aperture filter of less fall-off to provide more obvious depth-of-field effect. To further resolve the computation bottleneck of infocus-defocus hybrid RBs, we may also modify the layered processing in [28] for light fields and then make it a new refocusing mode at the layer $L = 0$. It is possible because many such RBs are partitioned into small-size blocks in the proposed flow, and each of them may contain a simple disparity distribution with few significant bins to enable the layered processing. Moreover, the proposed refocusing flow can be extended to support video applications by considering temporal continuity in the selection rules of parameters $N$ and $L$.

The regular and localized block-based flow is in particular suitable for hardware acceleration. Unlike [20] and [21] which rely on expensive and power-hungry GPU, the proposed method is able to provide a cost-effective and low-power hardware design for mobile devices. As our future work, we will implement it into VLSI circuits to enable real-time and low-power refocusing applications for high-resolution sparse light fields.

## VIII. Conclusion

In this paper, we propose a block-based multi-rate refocusing algorithm to provide fast physically-correct refocusing for sparse light fields. For infocused blocks, it assigns small interpolation numbers $N$ to avoid unnecessary computation; for defocused ones, it applies refocusing at subsampled layers $L$ to greatly reduce the complexity. For blocks at infocus-defocus hybrid boundaries, view interpolation is adopted to

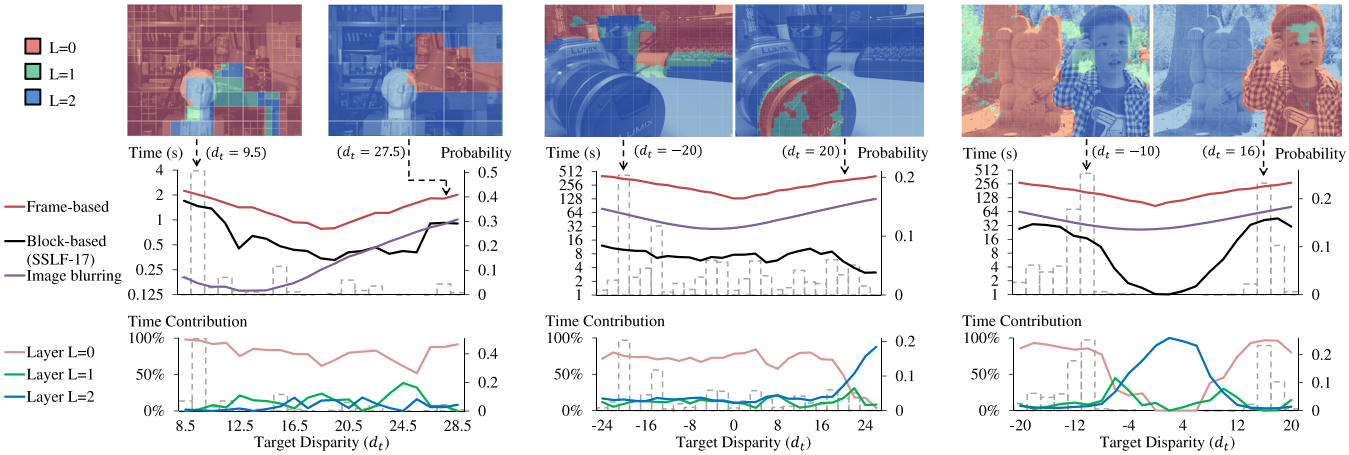



Fig. 23. Run time vs. target disparity $(d_t)$ for refocused images $(u_t = 1)$ of light fields *Tsukuba* (left), *Camera* (center), *Kid and Cat* (right). The timing-optimized RB partition maps are shown in the top row, and the run time contributions of different subsampled layers for the block-based refocusing are in the bottom row.

provide photorealistic depth-of-field effects, and the proposed fast line-scan method can achieve 30x as fast as the benchmark VSRS-1D-Fast. In addition, the key parameters $N$ and $L$ are determined systematically by a localized filter analysis to reduce aliasing and block artifacts. The experimental results based on fourteen tested light fields demonstrate the performance of the proposed method. While maintaining photorealistic refocusing quality, the block-based refocusing can provide a 9.7x speedup on average compared to the frame-based one. It also has comparable computation speed compared to the classical image blurring method. We also believe that this block-based framework can be extended for different purposes by including other refocusing methods such as image blurring or layered processing.

## REFERENCES


[1] M. Harris, "Focusing on everything," *IEEE Spectrum*, vol. 49, no. 5, pp. 44–50, 2012.

[2] M. Levoy, "Light fields and computational imaging," *IEEE Computer*, vol. 39, no. 8, pp. 46–55, 2006.

[3] C. Zhou and S. K. Nayar, "Computational cameras: Convergence of optics and processing," *IEEE Trans. Image Processing*, vol. 20, no. 12, pp. 3322–3340, 2011.

[4] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy, "High performance imaging using large camera arrays," in *Proc. SIGGRAPH*, 2005, pp. 765–776.

[5] A. Isaksen, L. McMillan, and S. J. Gortler, "Dynamically reparameterized light fields," in *Proc. SIGGRAPH*, 2000, pp. 297–306.

[6] R. Ng, M. Levoy, M. Bredif, G. Duval, M. Horowitz, and P. Hanrahan, "Light field photography with a hand-held plenoptic camera," in *CTSR 2005-02*. Stanford University, 2005.

[7] K. Venkataraman, D. Lelescu, J. Duparré, A. McMahon, G. Molina, P. Chatterjee, R. Mullis, and S. Nayar, "Picam: An ultra-thin high performance monolithic camera arrays," in *Proc. SIGGRAPH Asia*, 2013, pp. 166:1–166:13.

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

[9] C.-K. Liang, T.-H. Lin, B.-Y. Wong, C. Liu, and H. H. Chen, "Programmable aperture photography: Multiplexed light field acquision," in *Proc. SIGGRAPH*, 2008, pp. 55:1–55:10.

[10] B. Huhle, T. Schairer, P. Jenke, and W. Strasser, "Realistic depth blur for images with range data," in *Proceedings of the DAGM 2009 Workshop on Dynamic 3D Imaging*, 2009, pp. 84–95.

[11] J. Fiss, B. Curless, and R. Szeliski, "Refocusing plenoptic images using depth-adaptive splatting," in *IEEE International Conference on Computational Photography*, 2014, pp. 1–9.

[12] C.-K. Liang and R. Ramamoorthi, "A light transport framework for lenslet light field cameras," *ACM Trans. Graph.*, vol. 34, no. 2, pp. 16:1–16:19, Mar. 2015.

[13] P. Rokita, "Generating depth of-field effects in virtual reality applications," *IEEE Computer Graphics and Applications*, vol. 16, no. 2, pp. 18–21, 1996.

[14] Y. Taguchi, A. Agrawal, A. Veeraraghavan, S. Ramalingam, and R. Raskar, "Axial-cones: Modeling spherical catadioptric cameras for wide-angle light field rendering," *ACM Trans. Graph.*, vol. 29, no. 6, pp. 172:1–172:8, Dec. 2010.

[15] S. Wanner, S. Meister, and B. Goldluecke, "Datasets and benchmarks for densely sampled 4D light fiels," in *Vision, Modeling, and Visualization*, 2013, pp. 145–152.

[16] C.-T. Huang, J. Chin, H.-H. Chen, Y.-W. Wang, and L.-G. Chen, "Fast realistic refocusing for sparse light fields," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015, pp. 1176–1180.

[17] L.-R. Huang, Y.-W. Wang, and C.-T. Huang, "Fast realistic block-based refocusing for sparse light fields," in *IEEE International Symposium on Circuits and Systems*, 2016.

[18] R. L. Cook, T. Porter, and L. Carpenter, "Distributed ray tracing," in *Computer Graphics (Proc. ACM SIGGRAPH 84)*, vol. 18, 1984, pp. 137–145.

[19] M. Pharr and G. Humphreys, *Physically Based Rendering*. Morgan Kaufmann; 2 edition, Jul. 2010.

[20] J. Lehtinen, T. Aila, J. Chen, S. Laine, and F. Durand, "Temporal light field reconstruction for rendering distribution effects," *ACM Trans. Graph.*, vol. 30, no. 4, 2011.

[21] K. Vaidyanathan, J. Munkberg, P. Clarberg, and M. Salvi, "Layered light field reconstruction for defocus blur," *ACM Trans. Graph.*, vol. 34, no. 2, Mar. 2015.

[22] C. Soler, K. Subr, F. Durand, N. Holzschuch, and F. Sillion, "Fourier depth of field," *ACM Trans. Graph.*, vol. 28, no. 2, pp. 18:1–18:12, May 2009.

[23] R. Ng, "Fourier slice photography," in *Proc. SIGGRAPH*, 2005, pp. 735–744.

[24] S. Chan, H.-Y. Shum, and K.-T. Ng, "Image-based rendering and synthesis," *IEEE Signal Process. Mag.*, vol. 24, no. 6, pp. 22–33, 2007.

[25] *Test Model 8 of 3D-HEVC and MV-HEVC*. ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11 document JCT3V-H1003, Apr. 2014.

[26] L. Shi, H. Hassanieh, A. Davis, D. Katabi, and F. Durand, "Light field reconstruction using sparsity in the continuous Fourier domain," *ACM Trans. Graph.*, vol. 34, no. 1, pp. 12:1–12:13, Dec. 2014.

[27] J. Demers, "Depth of field: A survey of techniques," in *GPU Gems*. Addison-Wesley, 2004, pp. 375–390.

[28] F. Moreno-Noguer, P. Belhumeur, and S. Nayar, "Active refocusing of images and videos," *ACM Trans. Graph.*, Aug. 2007.

[29] W. Zhang and W.-K. Cham, "Single-image refocusing and defocusing," *IEEE Trans. Image Processing*, vol. 21, no. 2, pp. 873–882, 2012.

[30] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum, "Plenoptic sampling," in *Proc. SIGGRAPH*, 2000, pp. 307–318.

[31] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 614–619, Jul. 2003.
[32] J. T. Barron, A. Adams, Y. Shih, and C. Hernandez, "Fast bilateral-space stereo for synthetic defocus," in *Proc. CVPR*, 2015, pp. 4466–4474.

**Chao-Tsung Huang** (M'11) received the B.S. degree in electrical engineering and the Ph.D. degree in electronics engineering from the National Taiwan University, Taiwan, in 2001 and 2005 respectively.
He is now with the National Tsing Hua University, Taiwan, as an Assistant Professor. From 2005 to 2011, he was with the Novatek Microelectronics Corp., Taiwan, for developing multi-standard image and video codecs. He performed postdoctoral research on an HEVC decoder chip at Massachusetts Institute of Technology, Cambridge, MA, USA, from March 2011 to August 2012. He then worked on light-field camera design as his postdoctoral research at National Taiwan University, Taiwan, until July 2013.
His research interests include low-level computer vision and light-field signal processing, especially from algorithm exploration to VLSI architecture design, chip implementation, and demo system.
Dr. Huang serves as an Associate Editor for the *Springer Circuits, Systems and Signal Processing (CSSP)*. He was a recipient of the MediaTek Fellowship from 2003 to 2005.

**Yu-Wen Wang** was born in Kaohsiung, Taiwan. She received the B.S. and M.S. degrees in electrical engineering from Yuan Ze University, Taoyuan, Taiwan, and National Tsing Hua University, Hsinchu, Taiwan, in 2013 and 2015 respectively.
She is currently an Engineer with the Silicon Motion, Inc., Hsinchu, Taiwan.

**Li-Ren Huang** received the B.S. and M.S. degrees in electrical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2014 and 2016 respectively.
He is currently an Engineer with the MediaTek, Inc., Hsinchu, Taiwan.

**Jui Chin** received the B.S. and M.S. degrees in electronics engineering from National Taiwan University in 2012 and 2014 respectively.
He is currently with the PixArt Imaging Inc., Hsinchu, Taiwan. His research interests include computer vision and computational photography.

**Liang-Gee Chen** (M'86-SM'94-F'01) received the B.S., M.S., and Ph.D. degrees in electrical engineering from National Cheng Kung University, Tainan, Taiwan,in 1979, 1981, and 1986 respectively.
In 1988, he joined the Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan. Currently, he serves as the Chair Professor and received the National Professorship of Taiwan. Since May 2016, he is also the Political Deputy Minister of Ministry of Education, Taiwan, R.O.C.
His research interests include DSP architecture design, video processor design, and video coding systems. He has over 550 publications and 22 U.S. patents. He has conducted more than 100 technology transfers and helped two start-ups IPO successfully. He is the founder of Taiwan IC Design Society, the Chip Implementation Center (CIC), NTU SOC Center, and Graduate Institute of Electronics Engineering. In 2007, he established the Creativity and Entrepreneurship Program at National Taiwan University.
Dr. Chen has served on the Editorial Boards of many IEEE transactions, including *IEEE TRANSACTIONS ON VLSI*, *IEEE TRANSACTIONS ON VIDEO TECHNOLOGY*, and *IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS*. He was also invited as the Chair of the technical program committees for 2009 IEEE ICASSP and ISCAS 2012. In 2011, he received the Best Paper Award of IEEE Custom Integrated Circuits Conference (CICC).