Fast Bokeh Effects Using Low-Rank Linear Filters

Tim McGraw

Abstract

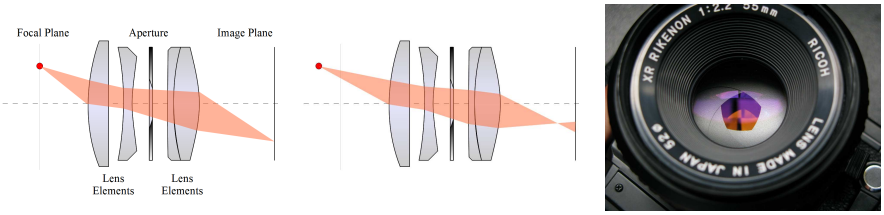
We present a method for faster and more flexible approximation of camera defocus effects **given a focused image of a virtual scene and depth map. Our method leverages the advantages of low-rank linear filtering by reducing the problem of 2D convolution to multiple 1D convolutions, which significantly reduces the computational complexity of the filtering operation.** In the case of rank 1 filters (e.g. the box filter and Gaussian filter) the kernel is described as 'separable' since it can be implemented as a horizontal 1D convolution followed by a 1D vertical convolution. While many filter kernels which result in bokeh effects **cannot be approximated closely by separable kernels**, they can be effectively approximated by low-rank kernels. We demonstrate the speed and flexibility of low-rank filters by applying them to image blurring, tilt-shift postprocessing, and depth-of-field simulation, and also analyze the approximation error for several aperture shapes.

Keywords: Bokeh Blur Filter Depth-of-field

1 Introduction

'Bokeh' (from the Japanese 'boke') refers to the visual quality of the **blurry, and out-of-focus** areas in photographs [18]. Bokeh effects are caused by multiple light rays from a single point in a scene passing through a camera aperture and striking different points on the image plane. **Very unfocused regions of the image may be characterized by a distinctive pattern of highlights roughly the shape of the camera aperture. This can be used for artistic effects** in photography when using lenses that have a shallow depth-of-field (i.e. a narrow range of distances in which objects are in focus). **For imaging systems that approximate a pinhole camera,** the whole image will be in focus and bokeh effects will be absent. However, bokeh effects can be added as a post processing step. This is the approach used in realtime graphics because images generated by polygon rasterization appear to be in focus everywhere.

The lens elements in a camera are designed to focus rays of light originating near the focal plane so that they converge on the image plane (see Figure 1a). Rays originating far from the focal plane may be spread into a circle of confusion (CoC) on the image plane (see Figure 1b).



(a) Point in focus (b) Point out of focus (c) 5-sided aperture.

Fig. 1: Focus of points in focal plane (a), and farther away (b). A lens with 5-sided aperture (c).

The aperture of the lens, formed by a number of pivoting blades, opens and closes to control the amount of light entering the camera (Figure 1c). The bokeh pattern is formed by the bundle of light rays that pass through the aperture. The shape of the bokeh is influenced by the number of blades, whether the blades are straight or curved, and the size of the aperture opening. Variations of light intensity within the bokeh are influenced by the optical properties of the lens elements.

The basic approach to simulating the bokeh effect is to integrate over the contributions of the many light rays that may be incident at each point on the image plane. Our approach is an image space technique which requires only a single image and depth map. Unlike other techniques based on separable filters which make restrictive assumptions about the nature of the bokeh, our method can simulate the effect of a wide variety of apertures and lenses. Some other methods based on stochastic sampling permit a more general modeling of the bokeh shape, but work with a sparse set of samples that can lead to noise artifacts. Our method produces images with no such noise artifact. By using low-rank linear filters we can combine the computational benefits of separable filters with the flexibility of stochastic sampling.

2 Previous Work

Camera depth-of-field (DoF) and bokeh effects have long been desirable features of computer graphics applications for developers attempting to emulate the styles seen in photography and filmmaking. Rather than presenting an extensive survey of techniques, we will instead focus on an overview of how various techniques handle the problem of approximating the integration over the space of incident light rays (for physically-based approaches), or adjacent pixels (for image-based techniques) at each point on the image plane.

Monte Carlo techniques. Cook [3] computed depth of field in a distributed ray tracer by stochastic sampling of rays traced from the camera lens into the scene. This approach can give realistic results in offline rendering applications,

because it naturally handles the well known intensity leakage and partial occlusion problems which plague DoF algorithms that require only a single pinhole camera image. Others [22, 2, 13, 14] have **presented single image techniques** which use the stochastic sampling technique to perform Monte Carlo integration over neighboring pixels. Haeberli [7] approached the problem by summing (in a hardware accumulation buffer) multiple pinhole images rendered from jittered viewpoints.

Splatting techniques. Splatting techniques are a direct approach to rendering DoF. These algorithms [12, 16, 25, 20, 24] proceed by rendering an alpha-blended sprite to the screen for each pixel in the frame buffer. The sprite is sized according to the diameter of the Coc, and the texture image generates the bokeh pattern. These techniques are fill rate intensive, and only feasible in realtime applications when the maximum Coc diameter is small.

Summed area tables. A summed area table (SAT) computed from an image holds the sum of all pixel values in the rectangle stretching from the origin to the destination pixel location. Once computed, the sums of values in arbitrary rectangular regions can be computed from 4 values in the SAT. Green and Hensley [6, 9] present fast methods for computing SATs on the **graphics processing unit (GPU)**, and apply the SAT to the DoF problem. However, these techniques are limited to simulating rectangular bokeh with no internal intensity variation.

Fourier transform. It is well known in the signal processing community that convolution may be implemented in the frequency domain as a simple multiplication [5]. Scofield [23] exploits this by computing the FFT of the image, performing multiplication, then computing the inverse FFT. However, this approach is limited since a convolution, in the strict sense, is filtering with a spatially invariant kernel. (Note that many works stretch the meaning of 'convolution' to include linear filters with spatially-varying kernels.) They overcome this limitation by partitioning the image into multiple regions over which the Coc is approximately constant, processing these separately, then recombining the filtered results.

Iterated or hierarchical convolution. By using repeated convolution [21] or hierarchical convolution of an image pyramid [11] it is possible to obtain the appearance of convolution with a large kernel. These techniques result in a limited range of bokeh shapes.

Solution of the diffusion/heat equation. It can be shown that the solution of the isotropic, homogeneous diffusion (or heat) equation can be obtained by Gaussian convolution [27]. Bertalmio and Kass [1, 10] approach the DoF problem by formulating a modified diffusion equation and solving it on the GPU.

Separable convolution. Separability is the property of some filter kernels that permits 2D filtering to be implemented by using a pair of 1D convolutions. This reduces the computational complexity, making it appealing for computing DoF in realtime applications. Our approach can be seen as improving and extending these methods by presenting an alternative to the separable filter since many desirable bokeh patterns do not have separable kernels. Zhou et al. [28], Hammon [8] and Lee et al. [15] use a separable Gaussian blur to accelerate the defocus effect, but the result is a very soft blur. McIntosh et al. [17] present a novel approach that is capable of generating polygonal bokeh, but their method has several restrictions that ours does not : the kernel is of uniform intensity throughout, the bokeh shape must be a union or intersection of parallelograms, and large overlapping bokeh do not blend properly.

Our low rank filter approach for bokeh effects is to approximate an arbitrary filter kernel as a sum of separable kernels. It is important to note that our low-rank linear filter approach can be used as part of many of the previously reported techniques for computing DoF from a single image by replacing the separable convolution or stochastic sampling step.

3 Low-rank and separable filters

The rank of a discrete filter kernel is the rank of its matrix representation, k. The rank of k is the number of linearly independent columns of k. If the rank of k is 1, a spatially invariant linear filter may be implemented using discrete convolution $I = I_0 * k = (I_0 * u) * v$ where I_0 is an input image, u is a vertical filter kernel represented by a column matrix and v is a horizontal kernel represented by a row matrix such that $k = uv$.

For a filter of arbitrary rank we may write

$$I = \sum_{i=1}^{r} \sigma_i (I_0 * u_i) * v_i. \tag{1}$$

The rank, r, of the filter kernel, k, and the values of σ, u and v can be determined by the singular value decomposition (SVD),

$$k = U \Sigma V^* \tag{2}$$

where u_i is the ith column of U, v_i is the ith column of V, and σ_i is $\Sigma_{i,i}$, the ith diagonal of Σ. The rank of k is given by the number of nonzero elements of Σ. The SVD can be used to determine a low-rank approximation to a filter kernel. Let $\tilde{\Sigma}$ be the same as Σ, but with all except the r largest singular values set to zero. Then $\tilde{k} = U \tilde{\Sigma} V$ is a rank r approximation to k. It can be shown that $\tilde{k}$ is the nearest rank r matrix to k in the sense of the Frobenius norm.

The quality of the low-rank approximation may also be computed from the singular values as

$$||k - \tilde{k}||_F = \sigma_{r+1}^2 + ... + \sigma_m^2, \tag{3}$$

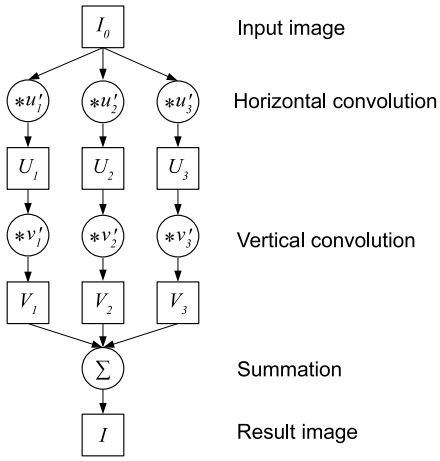


Fig. 2: Flow diagram of the rank 3 filtering process.

where $||\cdot||_F$ is the Frobenius norm, and m is the number of rows of k. The relative error of a low-rank approximation can then be defined as

$$\epsilon_{rel} = \frac{||k - \tilde{k}||_F}{||k||_F}. \tag{4}$$

Low relative error will lead to low-rank filtering results that are closer to the true 2D filtering results.

A flow diagram of the low-rank filtering process for rank = 3 is shown in Figure 2. Given the input image I_0 and the scaled singular vectors $u'_i = \sqrt{\sigma}_i u_i$, the intermediate results U_i are computed by discrete convolution in the horizontal direction. Each image U_i is then convolved with its corresponding scaled singular vector $v'_i = \sqrt{\sigma}_i v_i$ to obtain intermediate results V_i. These results are summed to obtain the final result, I. Note that the V_i intermediate results do not need to be stored if the image I is initialized to 0. In this case the vertical convolution results can simply be accumulated in I.

Compared to separable filters, low-rank filters allow us to create bokeh which have a wide variety of shapes and can simulate lens spherical aberration. Spherical aberration of the lens system can result in bokeh which have a nonuniform brightness - either lighter or darker at the edges. The bokeh shape is well approximated by the shape of the camera aperture. The shape and size of that aperture is controlled by the blades, which may be curved or straight, and vary in number from 2 to about 15. This results in bokeh shapes which are polygonal, or nearly circular in the case of an aperture with a large number of curved blades. With this in mind we investigated filter kernels with these shapes. Since it is possible to fit a camera lens with a special filter to give a photographer more control over the bokeh shape, we also investigated some filters which do

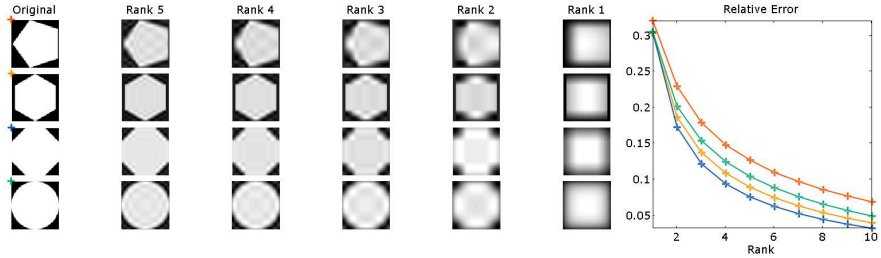


Fig. 3: Four filter kernels, their low-rank approximations, and a plot of relative error.

not correspond to aperture shapes.

3.1 Bokeh shape and matrix properties

Bokeh shapes which have a horizontal or vertical axis of symmetry will naturally have a lower rank since there will be rows (or columns) on either side of the axis that are equal. We orient our shapes within the filter kernel such that there is an axis of symmetry since we have observed that this reduces the error associated with the low-rank approximation. The low rank approximation for several filter kernels are shown in Figure 3. The rank of the original kernels (5-, 6-, 8-sided, and circular) are 55, 36, 38, and 39 respectively. In general, we have found that the lower rank kernels have lower relative error in their low rank approximations, even when the rank of the approximation is much lower than the actual rank of the original kernel. It is clear from Figure 3 that the 8-sided polygonal aperture shape leads to a filter kernel with the lowest relative errors for low-rank approximations with $2 \geq r \leq 10$. Filtering results for the 8-sided aperture are shown in Figure 4.

If the bokeh is symmetric about a diagonal axis such that the kernel, k, is symmetric in the matrix sense, then therefore has a Schur decomposition $k = Q\Lambda Q^*$. This decomposition permits more efficient storage of the 1D filter kernels since there will be horizontal and vertical kernels with equal components.

Many fast approaches to image blurring use box and Gaussian filters since the kernels are separable (have rank = 1), but in the context of camera defocus these kernels have some undesirable properties (see top row of Figure 5). A box filter will lead to square bokeh patterns, which are not commonly seen in photographic images. A Gaussian filter will lead to round bokeh patterns with soft edges, which are not always desirable. Our method permits arbitrary bokeh shapes to be used as filter kernels, and we analyze several of the most common shapes in this paper. By comparison, the polygonal and circular aperture filters result in a blur more consistent with photographic images (bottom row of Figure

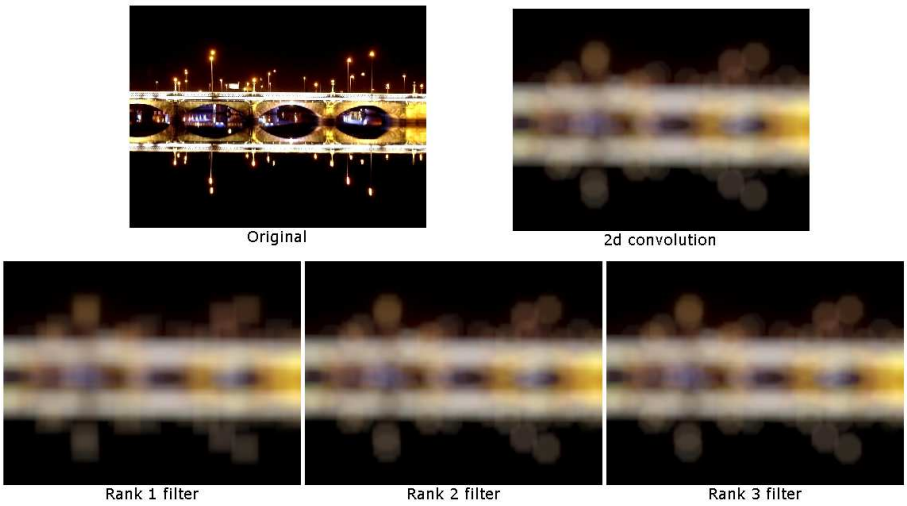


Fig. 4: Low-rank filtering results and the ground truth 2d convolution for the 8-sided polygonal aperture.

5).

Examples of filter kernels that simulate lens spherical aberration and curved aperture blades are shown in Figures 6 and 7, respectively. Figure 8 shows the filtering results for varying numbers of aperture blades and varying spherical aberration. The effect of spherical aberration and blade curvature on reconstruction error for rank = 3 is shown in Figure 9. A pattern seen in Figure 3 reoccurs here : the apertures with even numbers of blades have lower reconstruction error than those with an odd number of blades, with the 8-sided aperture having the minimum error. The plot of relative error vs. spherical shows that negative spherical aberrations (brighter in the center) lead to lower reconstruction error. We have observed that this trend continues for other ranks between 1 and 10. The plot of relative error vs. blade radius shows differing behavior between the apertures with even and odd numbers of curved blades. Increasing the blade radius (making the blades straighter) increases the low rank ($r = 3$) error for apertures with an odd number of blades, but decreases the error for apertures with an even number of blades. The circular aperture (approximated as a 60-sided aperture) falls between the two sets of curves, and has no dependence on blade radius. As expected, all relative errors converge to that of the circular apertures when the blade radius is the same as the circular aperture radius (1 in this case).

Figure 10 shows the results for several special filters. This demonstrates the generality of our technique with respect to aperture shape, and allows us to simulate the shaped bokeh that photographers create by using custom aperture disks [26].

The singular vectors for four filter kernels are shown in Figure 11 in order to

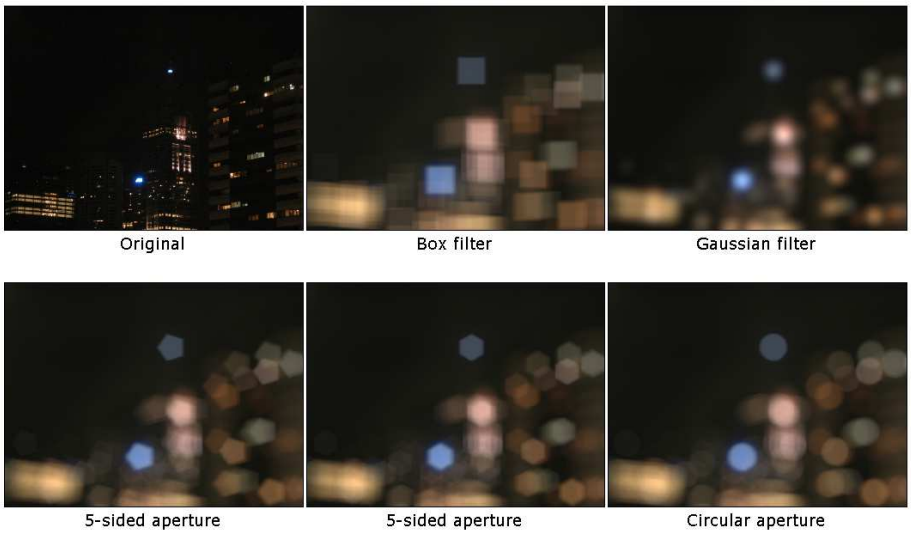


Fig. 5: Low-rank filtering results and the ground truth 2d convolution for the 8-sided polygonal aperture.

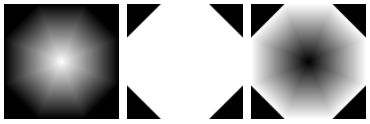


Fig. 6: Filter kernels for 8-sided aperture and lens spherical aberrations of -1 (left), 0 (center) and $+1$ (right).

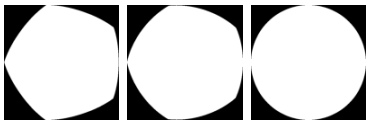


Fig. 7: Filter kernels for 5-sided aperture with curved blades of radius 2.0 (left), 1.5 (center) and 1.0 (right).

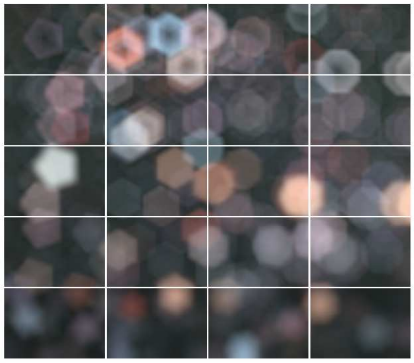


Fig. 8: Results of image blurring with low rank filters (r=4) of varying number of aperture blades (5, 6, 7, 8) left to right and spherical aberration (+1.0, +0.5, 0.0, -0.5, -1.0) top to bottom.

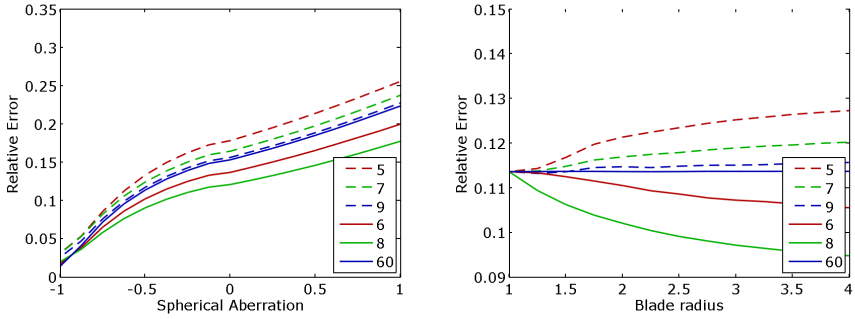


Fig. 9: Low-rank ($r = 3$) filter relative error vs. spherical aberration (left), and relative error vs. aperture blade radius (right) for apertures with number of blades = 5, 6, 7, 8, 9 and 60 (approximately circular).

Fig. 10: Results of image blurring with filters (r=4) of varying aperture shape (star, heart, flower, ring) left to right.

point out some potential optimizations. Aperture shapes with a horizontal axis of symmetry (all of the examples in Figure 11) have singular vectors u_i that are symmetrical about the midpoint, and if there is a vertical axis of symmetry the singular vectors v_i are symmetrical. This can be exploited to save memory when storing filter coefficients. This can be useful when implementing the algorithm on the GPU which has a relatively small area of constant memory. Also note that the 8-sided filter singular values u_i and v_i (as well as u_i of the 6-sided filter) have long runs of constant values near the middle of the domain. This suggests a possible optimization using summed area tables (or summed row / column tables). The potential benefit of such an optimization depends on many factors (e.g. hardware architecture, algorithm implementation). We do not consider this optimization any further in this work.

3.2 Comparison to stochastic sampling

Stochastic sampling of light rays can be used to generate physically-based DoF in ray tracing [3]. However, in image-based DoF [22], stochastic sampling of pixels can be used to reduce the number of samples of an image required when filtering. A drawback of this approach is that it can lead to noisy, grainy looking images. We compare 2 filtering implementations that use the same number of filter taps. The first implementation is a stochastic sampling technique using Poisson distributed samples within a hexagonal filter kernel. The locations of the samples are shown in Figure 12(a). The result of filtering is shown in Figure 12(b). For comparison, the result of rank 3 filtering is shown in Figure 12(c). Note that this image is free of the noise which is apparent in image 12(b).

Our method also exhibits better spatial locality of memory access than stochastic sampling. Since we are reading data from consecutive rows and columns of the image while filtering we will observe better cache performance.

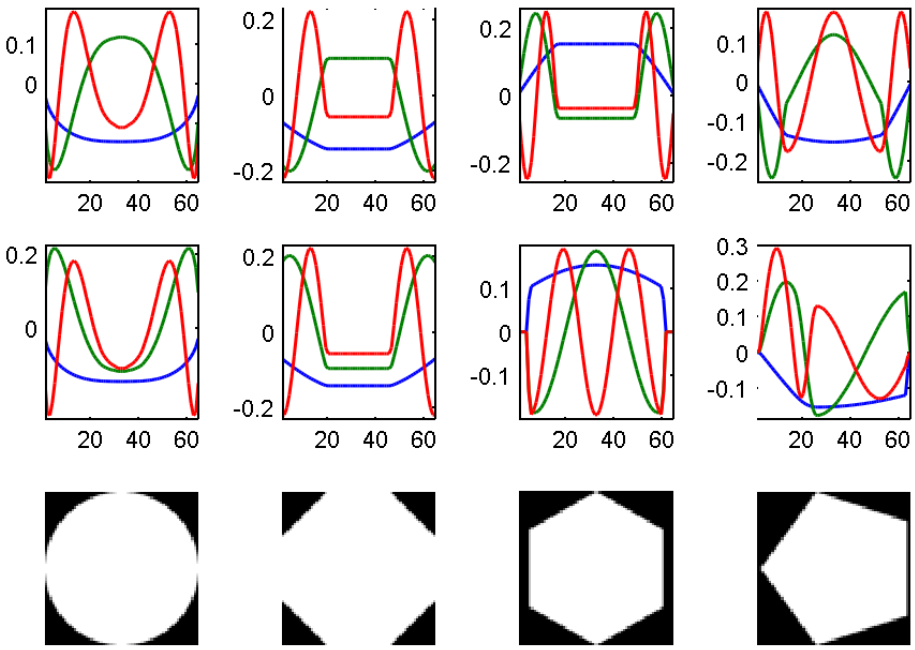
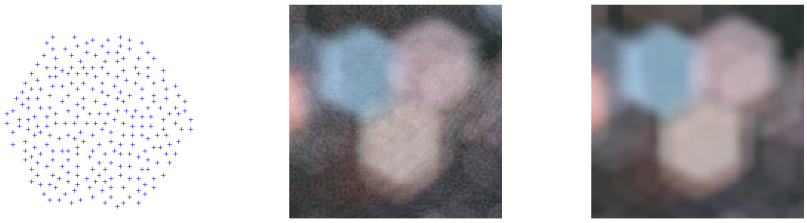


Fig. 11: The singular vectors of 4 filter kernels (bottom row). Top row is the singular vectors u_1, u_2, u_3, Middle row is the singular vectors v_1, v_2, v_3.



(a) Stochastic sample locations

(b) Detail from filtering result with stochastic samples

(c) Detail from low-rank linear filtering result

Fig. 12: Stochastic filtering and low-rank linear filtering with same number of filter taps.

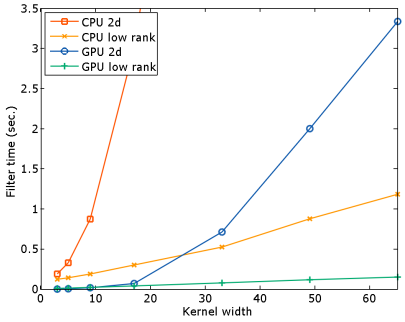


Fig. 13: CPU and GPU timing for 2D and low-rank filtering.

4 Results

Low-rank linear filter timing. Timing results for filtering implemented on the CPU and GPU as 2D convolution and as a sum of separable convolutions is shown in Figure 13. The image size was 1920×1080. For the GPU implementation we were able to achieve interactive framerates for filter kernels up to 65×65.

Tilt-shift postprocessing. Camera lenses that can translate and rotate relative to the image sensor are called 'tilt-shift' lenses. These lenses give the photographer control over perspective and focus in the final image. One application of these lenses is to create a shallow depth of field that causes the subject to appear to be miniaturized. This technique is sometimes referred to as 'tilt-shift photography', and can be simulated in postprocessing by designing a simple grayscale image representing Coc diameters. This is usually done using the gradient tool in a paint program, such as Photoshop. The photograph is then blurred with a spatially-varying kernel whose size equals the Coc diameter. Results of tilt-shift postprocessing using a low rank linear filter are shown in Figure 14. **Processing time for the** 727×554 **image was** 9.2 **ms.**

Linear filtering with a spatially-varying kernel is not a convolution since convolution is shift invariant. Convolution of an image with a constant 2D separable kernel will give identical results to a 1D horizontal convolution followed by a 1D vertical convolution, but implementing a 2D *spatially-varying* linear filter as a pair of 1D filters is prone to many visual artifacts. However, we find that when the filter kernel changes slowly (as it does in tilt-shift postprocessing) the impact is not noticeable. If the filter kernel changes abruptly, we can identify pixels near the discontinuities and perform 2D linear filtering there, as described in the next subsection.

Depth-of-field. In DoF applications we use a scene depth map to compute the Coc diameter which is used to determine filter width at each pixel. We use the

Fig. 14: Image created with tilt-shift postprocessing using a low rank (r=3) linear filter (left) and the corresponding image of Coc diameter (right). In the Coc image, dark intensities represent small Coc diameters, and bright colors are large Coc diameters. The black band corresponds to the focal plane.

Fig. 15: Depth image of a scene (left), edges detected in depth image (center), mask determined by blurring and thresholding the edge image (right). White pixels in the mask image show where a full DoF solution is computed. Everywhere else the low rank linear filter results are used.

equation for Coc diameter given by Demers [4],

$$Coc = a\frac{f(z_{focus} - z)}{z(z_{focus} - f)}, \tag{5}$$

where a is the lens aperture diameter, f is the focal length, z_{focus} is the distance to the focal plane, and z is the scene depth of the given pixel.

The depth map may contain discontinuities which are severe enough to cause artifacts in the low rank linear filter. To overcome this problem we flag pixels in the neighborhood of a depth discontinuity, and use a full depth of field calculation, including 2d filtering, in these areas. Pixels are flagged by performing edge detection on the depth image using finite differences, then blurring the edge image at the same time as the color image of the scene. An example depth image, edge image and mask are shown in Figure 15. We use the shader-based DoF technique proposed by Scheuermann [22] on the flagged pixels.

Our DoF results were generated using renderings produced by either the

physically-based renderer pbrt [19] or LuxRender. Both systems permit ray traced simulations of light transport. The camera model in pbrt only supports a circular aperture, but a variable number of blades are available in LuxRender. A color image and depth image were produced for each scene with no DoF by setting the camera aperture size to 0. These images were input to our proposed DoF technique. For comparison, a ground-truth image with DoF was ray-traced for each result. Tests were performed on a 3.40 GHz Intel Core i7 with 8GB RAM and AMD Radeon HD 7570 video card. Our DoF method was implemented in the OpenGL Shading Language.

Figure 16 shows the input color image of the Dragons scene, LuxRender results with a 6-sided aperture, and our results. A detail of the Dragons scene in Figure 17 shows that our method closely matches the bokeh of the raytraced image, while the technique proposed by Zhou et al. [28] which uses a separable Gaussian kernel results in a much softer blur.

Results for the School Corridor scene are shown in Figure 18. The incorrect softness of the blur seen on the lights in the ceiling when using the method of Zhou et al. is characteristic of all approaches utilizing Gaussian convolution, such as those by Hammon [8] and Lee et al. [15].

The Villa scene in Figure 19 shows how our technique can be used to give a plausible preview of bokeh effects for raytraced scenes. Since light rays contributing to out-of-focus areas of an image may come through the aperture from many possible directions, those parts of the image are typically undersampled and appear noisy. The Villa scene is shown in focus raytraced to 500 samples per pixel (spp), and some noise is still apparent in the image. The image raytraced with DoF required 5000 spp and also appears noisy, mostly in the unfocused regions of the background. However, our method applied to the focused image results in much less noisy bokeh due to the smoothing effect of the filtering process. The time required to raytrace the focused image and compute DoF was 1/10th of the time required to raytrace the scene with DoF. This approach to DoF permits the efficient previsualization of various aperture shapes from a single focused image. In Figure 20 we show 3 different aperture shapes simulated using our method compared with the raytraced solution. Note the color differences in the bokeh near the center of the images. Since our method is applied as a postprocess after tonemapping the true bokeh color is not always correct.

Figure 21 presents a problematic case for our algorithm. **Due to the camera being located near the branches of a tree,** there are a large number of depth discontinuities in the focal plane. This causes our technique to revert to a slow 2D convolution for many of the pixels in the resulting image. **Processing times for all results are shown in Table 1. The effect of the depth edges on the timing of our method can be clearly seen for the San Miguel scene. Although our method and Zhou's method are both many orders of magnitude faster than the ground truth, Zhou's method is always faster than ours since it relies on a linear filter of rank 1. However, our results had an average MSE which was** 31% **lower.**

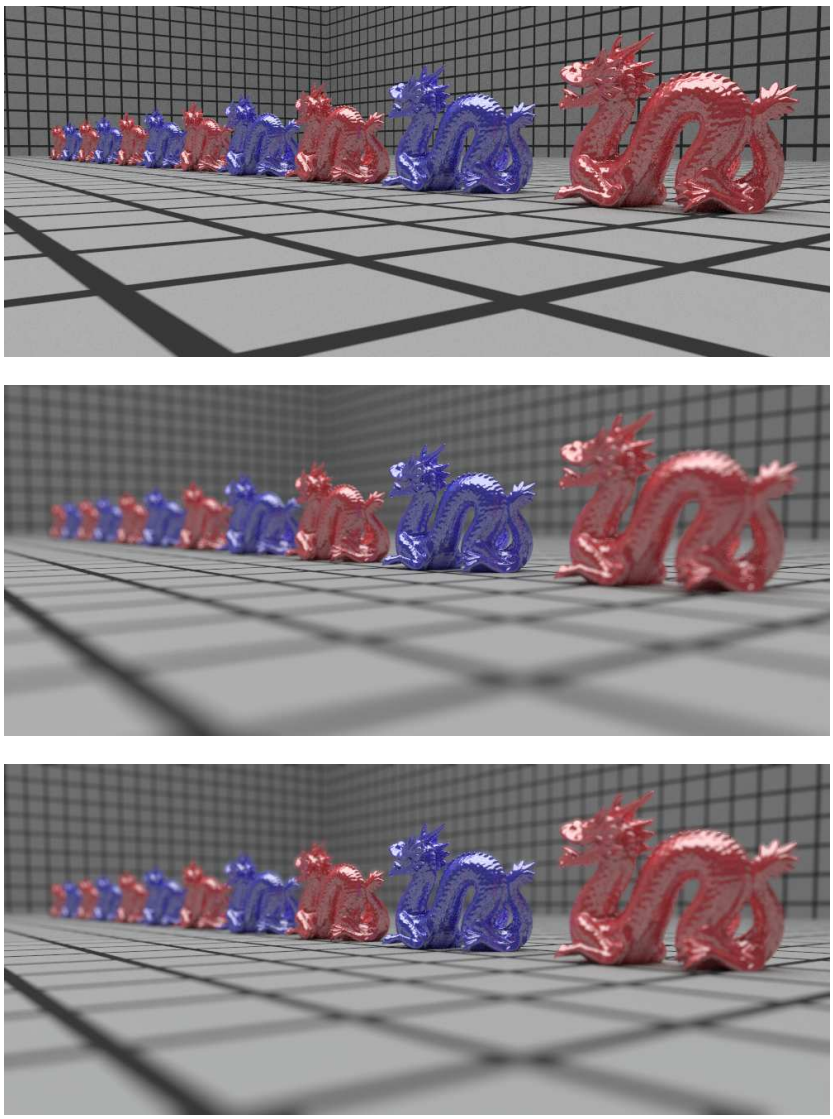


Fig. 16: Dragons scene raytraced with no DoF (top), rendered by raytraced with 6-sided aperture (center), our DoF results using low rank linear filtering with $r = 3$ and 6-sided aperture (bottom).

Fig. 17: Detail of Dragons scene. Raytraced (left), our method (middle), method of Zhou et al. (right).

Fig. 18: School Corridor scene. From left to right: raytraced with no DoF, raytraced with DoF, our method, Gaussian convolution.

Fig. 19: Villa scene raytraced with no DoF to 500 samples per pixel(left), raytraced with DoF to 5000 samples per pixel(middle), our method using 500 samples per pixel input(right).

Fig. 20: Details of Villa scene raytraced from another viewpoint (top), and DoF computed using our method (bottom). Aperture shape is circular (left), 5-sided (middle) and 6-sided (right).

Fig. 21: San Miguel scene raytraced by pbrt with DoF (left), our method (middle), depth edge mask (right). The high number of depth edges requires our solution to perform slow 2d convolution at many pixels.

Scene	Image size	Time (min) Raytraced	Time (ms) LRLF	MSE ($\times 10^{-3}$) LRLF	Time (ms) Zhou	MSE ($\times 10^{-3}$) Zhou
Dragons	1000×424	16.9	20.2	0.52	13.5	1.3
School Corridor	600×800	124	23.9	0.17	14.3	0.24
Villa	720×600	308	21.6	1.2	14.1	1.6
San Miguel	790×493	253	62.7	2.8	13.6	3.2

Tab. 1: **Timing results and mean squared error (MSE) for our proposed DoF method using low-rank linear filters (LRLF) and the method proposed by Zhou et al. Ground truth for MSE is computed by raytracing.**

5 Conclusions

In this paper we have presented a new technique for rendering **bokeh effects using low rank linear filters**. This technique can handle a variety of aperture shapes, unlike separable filters. Our technique can simulate lens spherical aberration which **causes nonuniform bokeh intensity**. We have also shown how the rank of the filter kernel may be selected by the user to **tradeoff speed for accuracy**. Unlike other **approaches which use stochastic sampling**, this method does not suffer from noise artifacts. We have also presented an analysis of several filter kernels which demonstrates **how to quantify the accuracy of the filter.**

Our results show that this technique works effectively within the framework of an existing DoF implementation to improve the visual quality of the results. We have also shown that this technique can be useful in an image processing context for both image blurring and tilt-shift postprocessing.

6 Acknowledgements

The authors would like to thank the providers of the following data. Dragon model: Stanford Scanning Repository, the San Miguel scene: Guillermo M. Leal Llaguno, the Villa: Florent Boyer, the School Corridor scene: Simon Wendche, Belfast at Night photo: rovingI, CBD at Night photo: Wilson Afonso.

References

[1] Bertalmio, M., Fort, P., Sanchez-Crespo, D.: Real-time, accurate depth of field using anisotropic diffusion and programmable graphics cards. In: 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on, pp. 767–773. IEEE (2004)

[2] Buhler, J., Wexler, D.: A phenomenological model for bokeh rendering. In:

ACM SIGGRAPH 2002 conference abstracts and applications, pp. 142–142. ACM (2002)

[3] Cook, R.L.: Stochastic sampling in computer graphics. ACM Transactions on Graphics (TOG) **5**(1), 51–72 (1986)

[4] Demers, J.: Depth of field: A survey of techniques. GPU Gems **1**(375), U390 (2004)

[5] Gonzalez, R.C., Woods, R.E.: Digital image processing. Prentice Hall Press (2008)

[6] Greene, S.: Summed area tables using graphics hardware. In: Game Developers Conference, vol. 6, p. 8 (2003)

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

[8] Hammon, E.: Practical post-process depth of field. GPU Gems **3**(28), 583–606 (2007)

[9] Hensley, J., Scheuermann, T., Coombe, G., Singh, M., Lastra, A.: Fast summed-area table generation and its applications. Computer Graphics Forum **24**(3), 547–555 (2005)

[10] Kass, M., Lefohn, A., Owens, J.: Interactive depth of field using simulated diffusion on a gpu. Pixar Animation Studios Tech Report **2** (2006)

[11] Kraus, M.: Using opaque image blur for real-time depth-of-field rendering. In: GRAPP, pp. 153–159 (2011)

[12] Krivanek, J., Zara, J., Bouatouch, K.: Fast depth of field rendering with surface splatting. In: Computer Graphics International, 2003. Proceedings, pp. 196–201. IEEE (2003)

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

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

[15] Lee, S., Jounghyun Kim, G., Choi, S.: Real-time depth-of-field rendering using anisotropically filtered mipmap interpolation. Visualization and Computer Graphics, IEEE Transactions on **15**(3), 453–464 (2009)

[16] Lee, S., Kim, G.J., Choi, S.: Real-time depth-of-field rendering using point splatting on per-pixel layers. Computer Graphics Forum **27**(7), 1955–1962 (2008)

[17] McIntosh, L., Riecke, B.E., DiPaola, S.: Efficiently simulating the bokeh of polygonal apertures in a post-process depth of field shader. Computer Graphics Forum **31**(6), 1810–1822 (2012)

[18] Merklinger, H.M.: A technical view of bokeh. Photo Techniques **18**(3) (1997)

[19] Pharr, M., Humphreys, G.: Physically based rendering: From theory to implementation. Morgan Kaufmann (2010)

[20] Potmesil, M., Chakravarty, I.: Synthetic image generation with a lens and aperture camera model. ACM Transactions on Graphics (TOG) **1**(2), 85–108 (1982)

[21] Rokita, P.: Fast generation of depth of field effects in computer graphics. Computers & Graphics **17**(5), 593–595 (1993)

[22] Scheuermann, T.: Advanced depth of field. GDC 2004 **8** (2004)

[23] Scofield, C.: 2 1/2 d depth-of-field simulation for computer animation. In: Graphics Gems III, pp. 36–38. Academic Press Professional, Inc. (1992)

[24] Shinya, M.: Post-filtering for depth of field simulation with ray distribution buffer. In: Graphics Interface, pp. 59–59. CANADIAN INFORMATION PROCESSING SOCIETY (1994)

[25] Soler, C., Subr, K., Durand, F., Holzschuch, N., Sillion, F.: Fourier depth of field. ACM Transactions on Graphics (TOG) **28**(2), 18 (2009)

[26] Tuttle, S., Hydeck, C.: Photo Craft: Creative Mixed Media and Digital Approaches to Transforming Your Photographs. F+W Media (2012)

[27] Weickert, J.: Anisotropic diffusion in image processing, vol. 1. Teubner Stuttgart (1998)

[28] Zhou, T., Chen, J.X., Pullen, M.: Accurate depth of field simulation in real time. Computer Graphics Forum **26**(1), 15–23 (2007)