https://doi.org/10.2352/EI.2025.37.7.ISS-285


# Lens Simulation for High Resolution and Multispectral Image using Distributed Multi-GPU Computing System

***Seonghyeon Kang, Jeongyong Shin, Sangmin Kim, Jeongwook Lee, Sung-Su Kim, Yitae Kim; S.LSI Division, Samsung Electronics; Hwaseong-si, Gyeonggi-do, Republic of Korea***

## Abstract

*In a lens-based camera, light from an object is projected onto an image sensor through a lens. This projection involves various optical effects, including the point spread function (PSF), distortion, and relative illumination (RI), which can be modeled using lens design software such as Zemax and Code V. To evaluate lens performance, a simulation system can be implemented to apply these effects to a scene image. In this paper, we propose a lens-based multispectral image simulation system capable of handling images with 200 million pixels, matching the resolution of the latest mobile devices. Our approach optimizes the spatially variant PSF (SVPSF) algorithm and leverages a distributed multi-GPU system for massive parallel computing. The system processes 200M-pixel multispectral 2D images across 31 wavelengths ranging from 400 nm to 700 nm. We compare edge spread function, RI, and distortion results with Zemax, achieving approximately 99% similarity. The entire computation for a 200M-pixel, 31-wavelength image is completed in under two hours. This efficient and accurate simulation system enables pre-evaluation of lens performance before manufacturing, making it a valuable tool for optical design and analysis.*

## Introduction

The optical properties and geometric shapes of the materials comprising a lens can be mathematically modeled. Tools such as Code V [1] and Zemax OpticStudio [2] are commonly used to handle these lens models. These simulation tools calculate various metrics for lens evaluation, including the f-number, focal length, and modulation transfer function. By enabling lens evaluation before production, they play a crucial role in optimizing lens design and performance.

However, what we get from using the simulation tools is only numerical result. What we achieve through lenses is a digital image captured by the lens. Therefore, it is necessary for lens evaluation to visually check the image obtained through the lens and provide a qualitative assessment.

Digital images are generated through a sequence of components: the camera lens, image sensor, and image signal processor. The image signal processor handles three-channel RGB images, which involve relatively small amounts of data and can be processed in a short time. However, for multispectral image simulation, three RGB channels are insufficient. Additionally, the resolution of images generated in camera lens simulations must exceed the sensor's resolution for accurate evaluation. Consequently, performing these simulations requires substantial computational power.

As the precision requirements for mobile lenses increase alongside the growing number of pixels in images, the need for advanced lens simulation tools becomes more critical. Lens evaluation is typically conducted using a two-dimensional chart image rather than a complex three-dimensional object. Therefore, in this paper, we implement a camera simulation that processes a 2D chart image into the camera's film representation.

## Related work

There are several approaches to addressing the camera simulation problem. One method involves estimating the ray-transfer function, which represents the input-output relationship of the lens, and incorporating it into the simulation [3][4]. This approach is particularly useful for simulating complex 3D environments. However, its performance depends on the chosen fitting function model, and due to the nature of function fitting, maintaining uniform accuracy across the entire lens area can be challenging.

Another approach to camera simulation involves separately modeling lens effects such as relative illumination (RI), distortion, and spatially variant point spread function (SVPSF) [5][6][7]. This method is well-suited for two-dimensional chart simulations and is the basis for our study. However, existing studies still face limitations in computational speed. Significant improvements are required to efficiently apply this approach to high-resolution 200MP images.

The high computational demand arises from the need to apply SVPSF, where each pixel has a unique point spread function (PSF), requiring spatial domain convolution. Several methods have been proposed to reduce the computational load [8][9]. However, these approaches inevitably result in some loss of accuracy due to PSF degradation.

In this study, we adopt a camera model based on [7], which minimizes PSF loss and ensures high precision. However, it is still not optimal in terms of computational speed. To address this, we implement a high-speed computing system capable of completing calculations within a few hours, even for 200MP images.

## Camera model

Our camera model is based on the pinhole camera model [10], which modifies only the size and aspect ratio of the input scene. While this model is simple, it has limitations in accurately representing real camera properties. To address these limitations, we integrate lens characteristics such RI, distortion, and SVPSF into our model. This enhancement enables more realistic camera simulations and allows for more precise lens performance evaluation.

Figure 1 presents a simplified representation of our camera model, which incorporates various lens effects. One such effect is RI, a phenomenon where the edges of an image appear darker than the center. The primary factor influencing RI is the chief ray angle (CRA), as the intensity of light decreases proportionally to $cos^4$ of the CRA due to the extended light path. Higher RI values indicate greater light transmission through the lens, resulting in brighter images and improved overall image quality.

Distortion occurs when an image captured by a lens from its actual form. This effect can result from various factors, with the most common types being radial distortion, caused by lens refraction, and tangential distortion, which arises from lens misalignment. Since our simulation system is specifically designed to analyze lens characteristics, we focus solely on implementing radial distortion. Radial distortion is commonly observed in wide-angle and fisheye lenses.

The PSF characterizes the spatial impulse response of a lens. Due to the wave nature of light, an impulse is spread out when projected onto the film, resulting in a blurred output image. Since the shape of the PSF varies depending on the pixel position within the image, it is referred to as the spatially variant point spread function SVPSF.

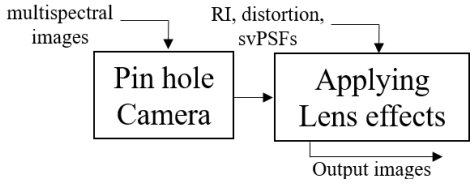



***Figure 1**. Camera model with the lens effect*

## Simulation in detail

Our method for applying lens effects is based on the approach described in [7]. In summary, RI is applied by adjusting the brightness values of the image, while distortion is implemented through nonlinear mapping.

Theoretically, applying the SVPSF requires extracting the PSFs for every pixel in the image space. However, this is computationally impractical. Instead, we extract PSFs only for pixels on a circular grid using an optical simulation tool (see Figure 2). By rotating these PSFs, we obtain values for the circular grid points. For pixels that do not lie on grid points, we estimate their PSFs using linear interpolation based on the four nearest grid points.

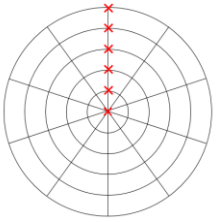


***Figure 2.** Circular grid on Film plane for PSFs*

Among various lens effects, applying the SVPSF presents the greatest challenge. The key contribution of our work lies in our ability to efficiently apply SVPSFs to a vast number of pixels in multispectral images. Our objective is to simulate images with 200 million pixels for each of 31 wavelengths, totaling 6.2 billion pixels. Since a unique SVPSF must be applied to each pixel, parallel GPU computing is essential for managing this massive computational workload effectively.

The main strategy for GPU computation is to upload as many operations as possible to the GPU. To achieve this, it is essential to set up the computational structure properly, and we have adopted the following strategy. Pixel on grid whose step size is the same as the PSFs, saved in 2D array. The corresponding PSFs are stored in a 2D array on the GPU. Note that this is not a 4D array, but rather a 2D array where each element is itself a 2D array for spatial convolution. The result of the element-wise multiplication of these arrays is accumulated in the output photon array. The pseudo code shows the process of applying SVPSFs in Algorithm below.

**Algorithm :** applying SVPSFs to the multispectral image

```
1 Procedure ApplySVPSFwitGPUs(input_photon, SVPSFs)
2    nGPU = the number of GPUs
3    H = Height of input_photon
4    W = Width of input_photon
5    nWave = the number of wavelengths
6    global output_photon = zeros(nWave, H, W)
7    H_psf, W_psf = Height and width of a SVPSF
8    ThreadPool thread[nGPU]
9    for w = 1, w <= nWave, w += 1 do
10      thread.clear()
11      for n = 1, n <= nGPUs, n += 1 do
12         Threads.GPUcomputing(n, input_photon[w,:,:],
SVPSFs)
13      end for
14      thread.waitforall()
15   end for
16   return output_photon
17 end procedure

18 procedure GPUcomputing (n, input_photon, PSFs)
19   global output_photon
20   H_psf, W_psf = Height and width of a SVPSF
21   for i = n, i <= H_psf, i += nGPU do
22      for j = 1, j <= W_psf, j += 1 do
23         P = input_photon[i:H_psf, H, j:W_psf, W]
24         PSF_array = PSFs corresponding to P
25         P_gpu = P /* Converting CPU to GPU */
26         PSF_gpu = PSF_array /* Converting CPU to GPU */
27         scene_gpu = P_gpu * PSF_gpu
28         /* Elementary wise multiplication of a 2D array of the
pixels, and a 2D array of the PSFs */
29         /*  scene_gpu forms a 2D array*/
29         output_photon += (scene_gpu to CPU)
30      end
31   end
32 end GPUcomputing
```

## Experiment and result

PCs are equipped with an Intel Xeon Gold 6430, 2T RAM, and 10 GPUs whose model name is NVIDIA RTA A6000. Our simulation code was implemented in Python. Our GPU code are based on CUDA, and to utilize multiple GPUs simultaneously, we used the GPUs in a thread for parallel processing.

In our experiments, we used Double Gauss 28-degree lens. The shape of the lens are in figure 3. The image size was set to 14,141 x 14,141 pixels, with a physical pixel size of 2um x 2um. The RI, distortion, and SVPSFs used in the simulation were extracted from Zemax Opticstudio. It covered 31 wavelengths, ranging from 400nm to 700nm with a step size of 10nm.

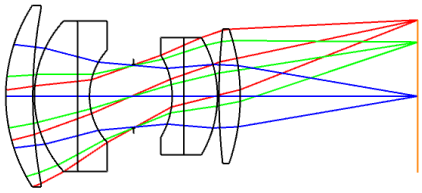


*Figure 3. Double Gauss 28-degree lens*

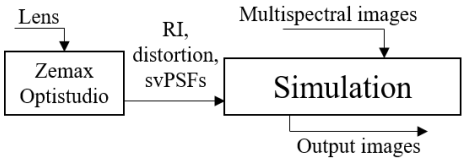



*Figure 4. Simulation system overview*

RI, distortion, and SVPSF are evaluated independently. RI is assessed by comparing the brightness values of images when a uniform scene is input. Distortion is evaluated using a 9×9 point array image as input. The centroid of each blob in the resulting image is considered the distorted coordinate, and its deviation from the expected values in Zemax OpticStudio is measured. SVPSF is challenging to evaluate directly for each pixel. Instead, we compare the edge spread function (ESF), which is the integral of the PSF. A slanted edge image is used for this comparison. The evaluation metrics include intersection over union for RI and ESF. Pixel distance is estimated for distortion.

**Table 1. Comparison results of the lens effects**

| Relative illumination | Distortion | Edge spread function |
|---|---|---|
| 99.9% | 0.1 pixel | 98.7% |

**Table 2. Consumed times of applying the lens effects**

| Size of PSF | Consumed time | |
|---|---|---|
| | RI and distortion | SVPSF |
| 50x50 | **27min 31sec** | **46min 15sec** |

Table 1 presents the quantitative results, showing that our simulation closely matches the results from Zemax OpticStudio. This indicates that the lens effects have been accurately applied as intended. The corresponding result graphs are displayed in table 3. Table 2 shows the computation times for applying lens effects. A total of 6 billion pixels were processed with PSF application, yielding results in under an hour. Even when combining all computations, the total processing time remained under two hours. These results demonstrate that our simulation tool excels in both accuracy and computational efficiency.

## Result and Further Work

In this paper, we developed a camera lens simulation system capable of processing 200MP images across 31 wavelengths. The simulation completes within a few hours, and verification results for key lens effects—including relative illumination (RI), edge spread function (ESF), and distortion—demonstrate high accuracy. These advancements enable efficient qualitative evaluation of various lenses, making a significant contribution to lens assessment and design.

By incorporating pixel patterns, demosaicing, and image signal processing, we can generate complete RGB digital images instead of simple photon values on the film. Additionally, various noise types—such as shot noise, dark current noise, and reset noise—can be applied during digital image processing. This enhancement enables our simulation tool to perform not only quantitative evaluations but also qualitative assessments of how these processes and noise factors impact the final digital image.

Our research is currently limited to processing 2D chart images into 2D film images. However, by incorporating depth information from 3D geometric models, we can extend our PSF computation algorithm to account for both spatial and depth variations. As a future direction, we plan to explore the conversion of 3D meshes into 2D depth images at a resolution of 200MP.

## Reference


[1] CODE V, https://www.synopsys.com/.

[2] Zemax Opticstudio, https://www.ansys.com/.

[3] Farrell, J. E., Catrysse, P. B., and Wandell, "Digital camera simulation,", Applied optics, vol, 51, issue 4, A80-A90, 2012.

[4] Goossens, T., Lyu, Z., Ko, J., Wan, G. C., Farrell, J., and Wandell, B., "Ray-transfer functions for camera simulation of 3D scenes with hidden lens design," Optics Express, vol. 30, issue 13, pp. 24031-24047, 2022.

[5] Maeda, P. Y., Catrysse, P. B., and Wandell, B. A., "Integrating lens design with digital camera simulation," Digital Photography, vol. 5678, pp. 48-58, 2005.

[6] Chen, J., Venkataraman, K., Bakin, D., Rodricks, B., Gravelle, R., Rao, P., and Ni, Y., "Digital camera imaging system simulation," IEEE Transactions on Electron Devices, vol. 56, issue 11, pp. 2496-2505, 2009.

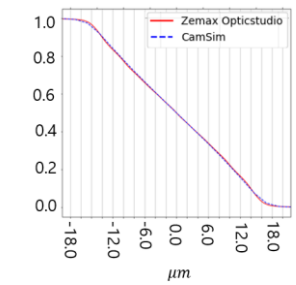
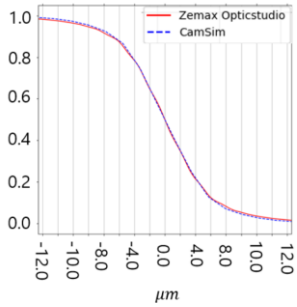
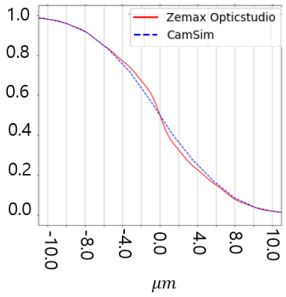
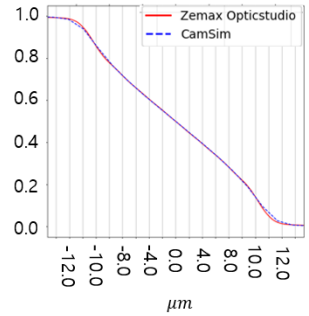
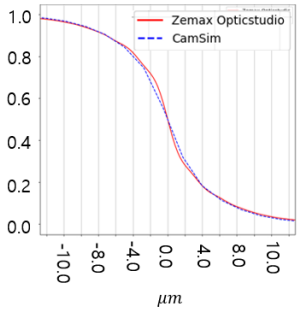
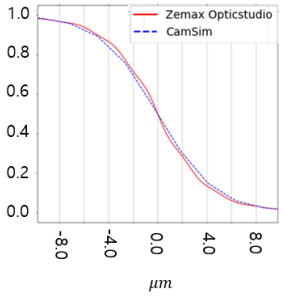
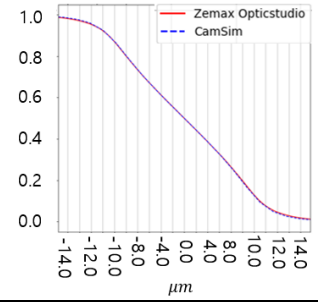
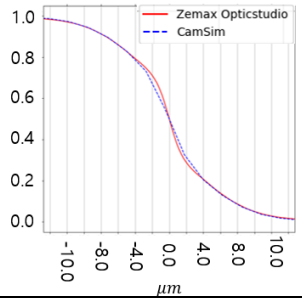
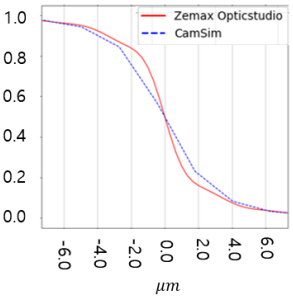
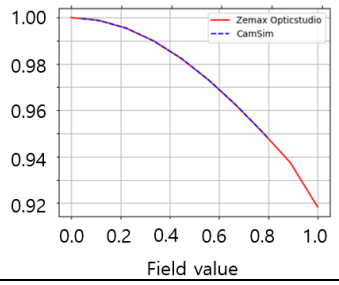
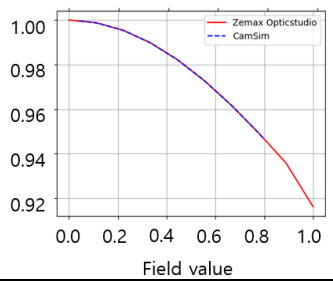
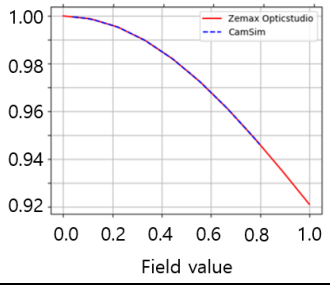
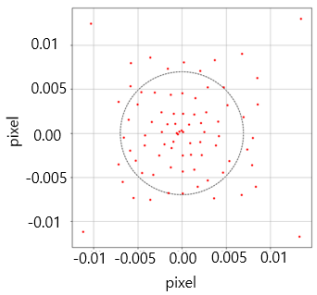
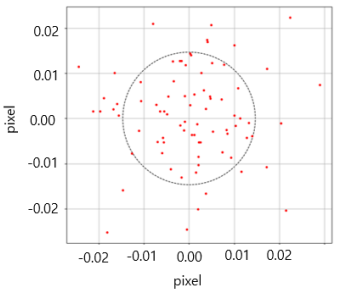
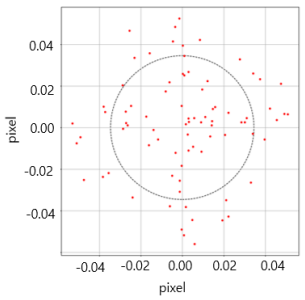
[7] ISetcam, https://github.com/ISET/isetcam.

[8] Novak, K., & Watnik, A. T., "Imaging through deconvolution with a spatially variant point spread function," Computational Imaging VI, vol. 173, pp. 24-40, 2021.

[9] Janout, P., Páta, P., Skala, P., & Bednář, J., "PSF estimation of space-variant ultra-wide field of view imaging systems," Applied Sciences, vol. 7, 151, 2017.

[10] R. Hartley and A. Zisserman, "Camera geometry and single view geometry" in Multiple View Geometry in Computer Vision, Cambridge, U.K.:Cambridge Univ. Press, pp. 151-236, 2004.

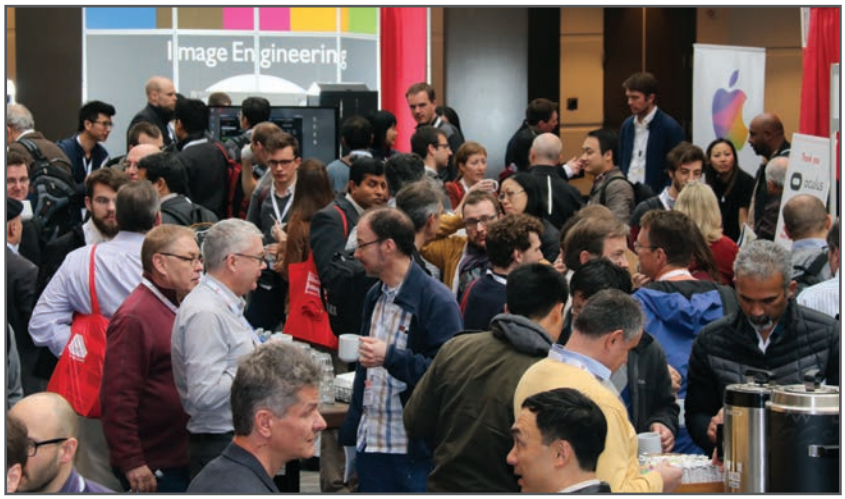
**Table 1. Result figures of the experiments**

| | | 450nm | 550nm | 650nm |
|---|---|---|---|---|
| Edge spread function | 0.0F |  |  |  |
| | 0.5F |  |  |  |
| | 0.8F |  |  |  |
| Relative illumination | |  |  |  |
| Geometric distortion | |  |  |  |

# JOIN US AT THE NEXT EI!

# electronic IMAGING

*Imaging across applications . . . Where industry and academia meet!*





**• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •**
**• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •**

## www.electronicimaging.org

ADVANCING IMAGING SCIENCE 75 1947 IS&T 2022 imaging.org