Rendering Realistic Lens Flare

Huixuan Tang

1. Introduction

Lens flare is an unintended effect caused by rays passing through the camera lens in an unintended way. It is due to inter-reflection of lens elements of a compound lens and diffraction and adds various artifacts to photos, namely:

- numerous ghost images with various colors and (sometimes) ringing structure
- a starburst pattern on the image center along with a haze across the entire image.

Fig. 1 shows a real lens flare in a photography.

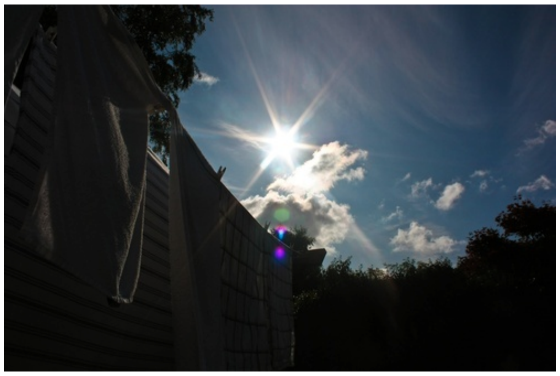


Fig.1 A photo containing lens flare. [1]

Although considered undesirable to lens designer, lens flare is sometimes delibrately used in photography and filming to invoke a dramatic or realism sense, e.g. [1]. For the same reason they are also commonly seen in games and photo editing tools to make the photos look more interesting.

In this project, I extend PBRT[2] to render realistic lens flares based on [3-5]. My implemen-tation reads in a lens prescription (See Table. 1 for an example) and output two separate layers of the image: the rendering of the scene and the lens flare. The layers are then linearly combined to produce the final image with a lens flare. See Fig. 11 in Section 6 for an example.

```
# D-GAUSS F/2 22deg HFOV
# US patent 2,673,491 Tronnier"
# Moden Lens Design, p.312"
# Scaled to 50 mm from 100 mm

Radius     Axpos   N      Aperture  v-no
29.475     3.76    1.67   25.2      47.1
84.83      0.12    1      25.2      -
19.275     4.025   1.67   23        47.1
40.77      3.275   1.699  23        30.1
12.75      5.705   1      18        -
0          4.5     1      17.1      -
-14.495    1.18    1.603  17        38.4
40.77      6.065   1.658  20        57.0
-20.385    0.19    1      20        -
437.065    3.22    1.717  20        48.1
-39.73     0       1      20        -
```

Table 1. An example of lens prescription

The implementation of the technique involves several components:

Accurate ray-tracing Given an indicent ray, the rays are traced though a composite lens with all its optical elements so that ray pathes would be nearly accurate.

Generating monochromatic ghosts There are two alternatives for generating ghosts in the image. One is to generate random ray samples from the light source and use photon mapping techniques to interpolate the irradiance on the sensor [4]. The other is to trace sparse rays intersecting the front element on a quad-grid to capture the warping from the aperture stop to the sensor, and texture map the aperture image onto the film. In my implementation I tried both approaches and found the latter to produce more persuading ghost images.

Handling color variation Due to dispersion, rays of different wave lengths are refracted differently at each lens elements, leading rainbows at the borders of the ghosts. Besides, lens coating eliminates reflected light of specfic wavelengths, causing color variations among the ghost images.

Involving diffraction According to wave optics, the starburst pattern is due to Fraunhofer diffraction and the ringing of the ghosts are due to Fresnel diffraction. The intepretation of [6] provides a unified framework for conveniently calculation of the diffraction in image with fractional Fourier transform . Thus diffraction can be rendered by plugging this step in before the texture mapping step in ghost generation.

2. Generating Ghosts

2.1 Tracing rays

I follow [3] to trace rays fron one end of the lens to another. This is by iteratively intersecting the ray with each lens surface; deciding whether the ray runs out of the aperture of the surface; and changing the direction of the ray by law of refraction/reflection.

The forward (camera sensor to front element) path tracing corresponds to the GenerateRay() function of PBRT and are input to surface integrators to render the scene. It creates realistic DoF effect to the scene image. The backward path tracing is called a separate routine for ghost generation, as described in the following.

2.2 Ghost generation by photon mapping[5]

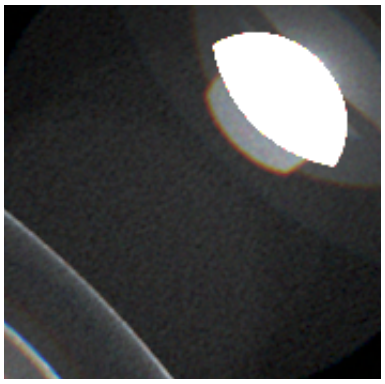
My first attempt to generate ghost is by photon mapping. In this approach, random samples are first generated on the front element of the lens. A sampled point along with a (sampled) point from the light source decides the direction and radiance of the indicent ray. The backward ray tracing algorithm computes the intersection of the ray with the aperture stop and the film and the resulting information is stored on a photon map. Finally, for each pixel on the camera film, the sensor irradiance is calculated by retrieving neighboring photons and interpolating them to get the final ghost.

However, since the size resulting ghost are often very large, a significant number of ray samples are required to produce a ghost image of high quality. Fig.2(a) shows a ghost image produced by photon-mapping based method with 2,00,000 ray samples. Despite the large number of ray samples used, the resulting ghost images still look noisy.

2.3 Ghost generation by texture mapping[4]

Alternatively, ghost can be generated tracing sparce rays on the sensor and interpolating the aperture intersection coordinates for the nearest sampled points. This is by first rasterizing the front pupil with a regular quad-grid, tracing rays through each vertex of the grid. The path tracing algorithm form an "truncated aperture image" at the front pupil, which determines if individual rays associated with a dense discreterization (512x512) of the front pupil can pass through the lens or not. It also provides a sparse correspondence from the front pupil coordinate to the sensor coordinate. Finally, the ghost is computed by mapping each cell of the truncated aperture image to the sensor. To account for the ray distributions, the sensor irradiance of each pixel is normalized by the area of the cell on the sensor.

Fig. 2 shows an example ghost rendered with the texture mapping approach. It samples 32x32 grid and generates much smoother results. The problem of using a low sampling rate on the front pupil potentially cause stripe artifacts in the ghost. The problem can be handled by rasterizing the front pupil with a denser grid, but this increases computational time. Sec. 5 addresses the problem by local refinement of the grid.


(a) Ghost generated by photon mapping

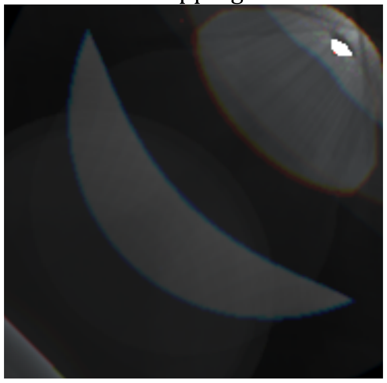

(b) Ghost generated by texture mapping

Fig. 2 Two approaches for generating ghosts

3. Coloring the ghost

Both the path a light travels and its reflectivity at a lens surface depends on its wavelength, and this causes color variations within a ghost and among different ghosts. When rendering the lens flare, I sample a number of (3-7) wavelengths and compute the ghost image for that specific wavelength. After all these images are computed, the intensity of the pixel at different wavelengths give the SPD of the image, and can be used to generate the final color of the ghost. Fig. 3 shows an example rendered lens flare image showing both variations.

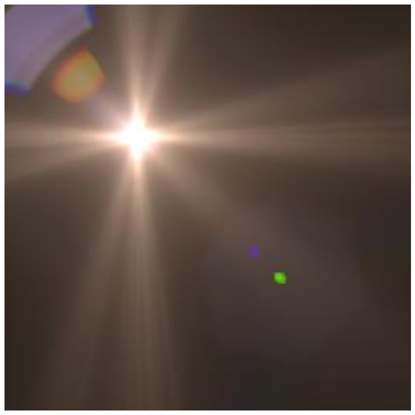


Fig. 3 Color variation of ghost images for the lens prescription of Fig. 1.

3.1 Dispersion

Light of different wavelength travels at a different speed in dispersive media. Accordingly, a beam of ray arriving at an interface between two mediums may diverge due to the wavelength-dependent variation in their relative refractive index. Typically refractive index is high for shorter wavelengths, and can be quantified by the Abbe number:

$$v = \frac{n_d - 1}{n_F - n_C},$$

where n_d, n_F and n_C are the refractive indices of the material at the wavelengths of the Fraunhofer d-, F- and C- spectral lines (587.6 nm, 486.1 nm and 656.3 nm respectively).

The refractive index as a function of wavelength can also be described by an empirical formula, such as the first-order Cauchy approximation:

$$n(\lambda) = A + \frac{B}{\lambda^2}$$

In my implementation, the input lens prescription includes a column v for the Abbe number, and a column n_d for the refractive index at the d- line. These information are used to transform into the coefficients A and B for the Cauchy approximation.

3.2 Anti-reflection Lens coating

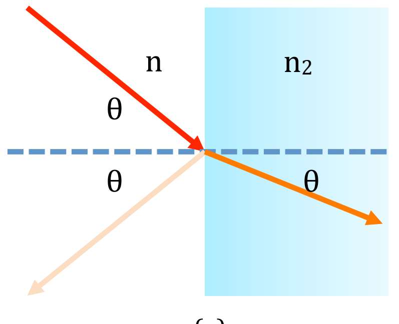
The proportion of light refracted at a lens interface is given by the Fresnel equation

$$R_s = \left[\frac{n_1 \cos\theta_i - n_2 \cos\theta_t}{n_1 \cos\theta_i + n_2 \cos\theta_t}\right]^2, T_s = 1 - R_s,$$

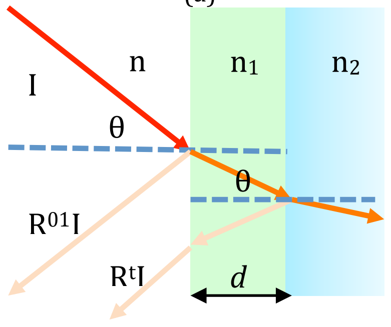
$$R_p = \left[\frac{n_1 \cos\theta_t - n_2 \cos\theta_i}{n_1 \cos\theta_t + n_2 \cos\theta_i}\right]^2, T_p = 1 - R_p,$$

where R and T denotes reflection and transmission (refraction), and the subscripts s and p denotes polarizations. The other notations are illustrated by Fig. 4(a).

To reduce reflected light, anti-reflection lens coating adds a thin layer of material at the interface so that a train of light wave arriving at the interface would split into two waves. The coating is deliberately designed so that the phase difference between the two waves is $\pi/2$ and they can cancel out each other (Fig.4 (b)). For this purpose, the desired refractive index of the coating $n_1 = max(\sqrt{n_0 n_2}, 1.38)$ and its thickness $d = \lambda_o/4/n_1$, where λ_o is the wavelength of light whose reflectivity is minimized.



(a)



(b)

Fig. 4 Interpretation of Quarter-wave coating

The reflectivity with a layer of refractive index n_1 and thickness of d can be computed by combining the p- and v-polarized reflectivity. I followed [4] by assuming the p- and s- polarization of the incident light is equal, thus

$$R = (R_s + R_p)/2.$$

Each component R_d is the addition of the two reflected waves:

$$R_d = (R_d^{01})^2 + (R_d^t)^2 + R_d^{01} R_d^t \cos\left(\frac{4\pi}{\lambda}\Delta\right),$$

where R_d^{01} and $R_d^t = R_d^{12}(T_d^{01})^2$ are the reflectivity from the first and second interface; and the relative phase delay $\Delta = n_1(\mathrm{d}/\cos\theta_1 - d \sin\theta_0)$.

4. Involving diffraction

The ringing and starburst patterns are both due to diffraction – a phenomenon when light passes though a small-scale geometry such as the aperture.

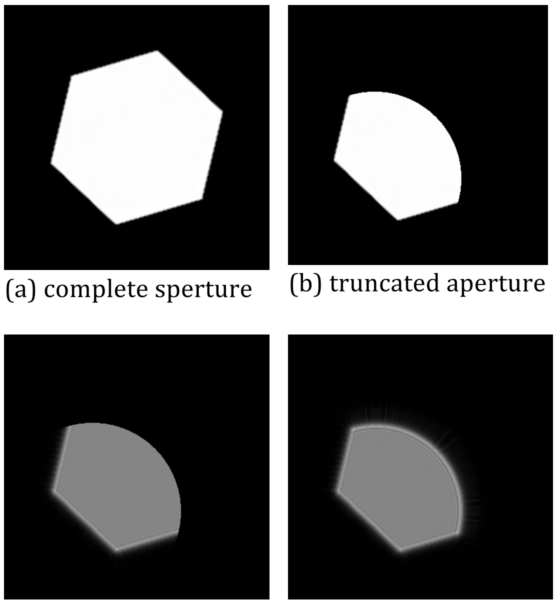
4.1 ringing pattern

It was shown by [5] that for an optical system of axially aligned spherical lenses, the amplitude distribution of light can be easily computed from fractional Fourier transform (FrFT):

$$(\mathcal{F}^{\alpha}q)(u) = \int_{-\infty}^{\infty} B_{\alpha}(u,u')q(u)du$$

$$B_{\alpha}(u,u') = \frac{exp[-i(\pi\hat{\phi}/4-\phi/2)]}{\sqrt{|\sin\phi|}} \cdot$$
$$exp[-i\pi(u^2 cos\phi - 2uu' \csc\phi + u'^2\cos\phi)]$$

where $\phi = \frac{a\pi}{2}, \hat{\phi} = \operatorname{sgn}(\sin\phi)$ is the phase transform determined by radius and positioning of the lenses, as well as wavelength of the light.


(a) complete sperture (b) truncated aperture

(c) ghost by [4] (d) ghost by my approach

Fig.5 The need to apply FrFT to incomplete aperture image. The hexagon (a) aperture is cut by circular lens barrel and causes the effective aperture to be like the mask in (b). Performing FrFT to (a) and than doing the truncation would make the ringing inconsistent with the ghost shape. My approach address the problem by applying FrFT to (b).

Similar to [4], I perform FrFT transform with a small order that is linearly dependent on the F-number and wavelength of the image. However, rather than transforming the complete aperture mask, I choose to apply FrFT to the "truncated aperture image" cut by both the diaphragm and lens barrel. As shown in Fig. 5, this allows the ringing to match with the shape of the ghost better. This modification also makes it possible to extend my implementation to deal with lenses with multiple stops.

4.2 Starburst

The starburst pattern corresponds to the far-field approximation of the diffraction, and the diffracted image can be gained by taking the Fourier spectrum of the image. The scale of the spectrum is linearly dependent on the size of the aperture. To make the starburst pattern more interesting, the image of the starburst pattern is randomly rotated by a small angle for several times and the final starburst is the linear combination of the rotated copy of the starbursts. Fig. 6 shows an example of such starbursts. Interestingly, because there are a lot of non-zeros in the spectrum, it provides a feel of haze throughout the image.

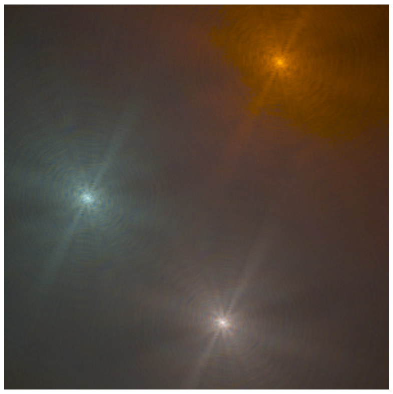


Fig. 6 Starburst image of three separate light sources.

5. Balancing Computational Cost and Quality

The computational cost for path-tracing based lens flare rendering is a significant multiply of the DoF generation model in PBRT. First, the number of paths through the lens due to a single incident ray scales exponentially with number of elements in the lens and the number of reflections along the path. Second, the area of each ghost can be a considerable proportion of the full-image size. Although the texture mapping based approach can render reasonable ghosts at most times, it still creates artifacts when the ray grid is highly deformed.

To address the first issue, I followed [4] to reduce the number of apertures by : (1) only taking into account light pathes with at most two reflections, (2) At a narrow aperture, ignore paths passing the stop for three times, i.e. the two interfaces for reflection are on different sides of the stop; (3) compute rough area of intersection on the sensor from the ray tracing results, and ignore ghosts whose area is too large (half the area of the output image), considering such ghost images to be too dim to show.

To address the second issue, I perform a local refinement of the ray mesh on the front pupil.
I start with a 16x16 quad-mesh on the front pupil and divide each triangular cell into 4 smaller cells if its vertices do not convey enough information for it to texture map its interior. The division process terminates when the interior of a cell can be well interpolated, or the area of the mesh is small enough to be negligible. Fig.7 shows an example of such division.

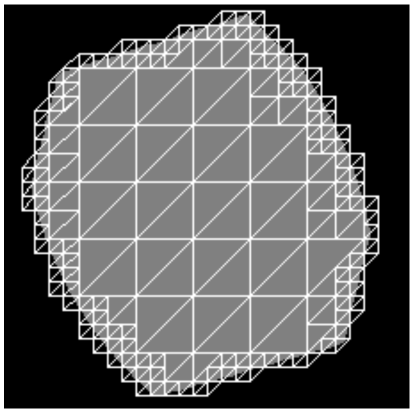


Fig. 7 Example of local mesh refinement when forming aperture. The shade area shows the final filling of the truncated aperture.

In the procedure of forming the "truncated aperture texture", a cell stops division when all rays associated with the vertices have passed through the lens. In the procedure of mapping the texture to the sensor, the testing condition is whether the deformation of the cell is approximately linear. This is by taking a sparse sampling within the cell, computing intersection with the sensor with these rays, and test if they are close enough to the interpolated positions estimated from the three vertices.

The real-time lens flare rendering paper [4] also talked about using symmetry of the aperture to reduce computational complexity. However, when rendering lens flares of creative bokeh (see section 6 for details), this constrains maynot hold. Therefore in the experiments I ignore this effect.

6. Results

In this section I show some additional results of lens flare rendering. On a Macbook with Intel Core 2Duo 2.53GHz processor, rendering a 500x100 lens flare image for a simple lens with 20 ghosts and a single light source takes less than 1 minutes. More complex lenses and more light sources increases the computational time, but is still within an acceptable range (5-10 minutes). Note that all the computations are done on a single core, therefore, there is still a large space of improvement in speed using multiple cores or GPU computation.

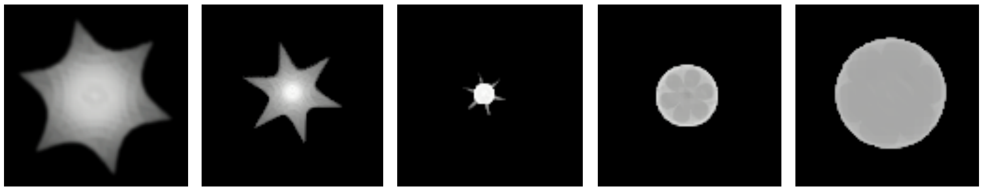


Fig. 8 Through-focus Bokehs of a 100mm Double Gaussian lens. To better illustrate the structure of the bokehs, only the center 100x100 pixels of a 7mmx7mm/300x300 pixels is shown. The image is taken by a 100mm F1.39 Double Gaussian lens[7] at five focus settings (distance from the back element ranges from 91.5mm to 95.5 mm), with a hexagon shaped aperture stop.

Optical aberrations A by-product for my implementation is the realization of optical aberrations, an effect due to the deviation of the actual ray paths from the linear approximation of lens maker's equation. Due to this effect, rays emitting from a single point light source would focus at multiple depths, causing non-uniform "bokehs" (main image of a single point light source) in the image. Fig. 8 shows a sequence of bokehs of a light source at 400mm whose image in the image center. The resulting image shows both interesting shapes and non-uniform structures.

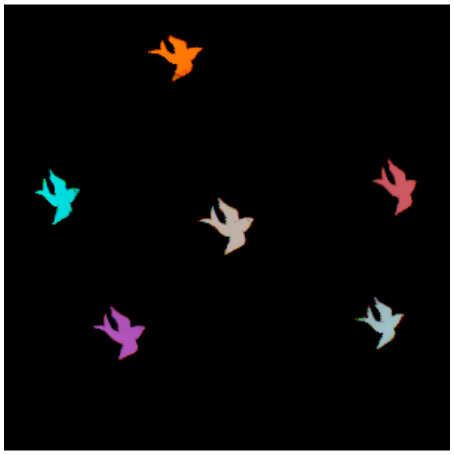
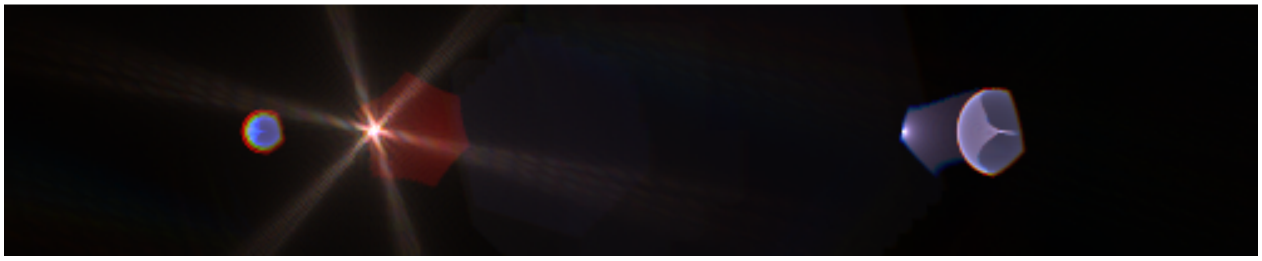


Fig. 9 Spatially varying bokeh shape. In this example ghost images are not shown to clearly demonstrate the bokeh shapes. The lens layout is the same as the one used for Fig. 8. All light sources are placed at 420mm.

Another characteristics of optical aberration is the spatially varying deformation of the bokeh. Fig. 9 shows an scene containing a few light sources. Note the warping of the deformation mask for bokehs in the corner. This example also illustrates the use of creative stop shapes to make the image look more interesting.

Lens flare of various lenses Fig. 10 shows two additional lens flares generated by an old double Gaussian lens as well as a model commercial lens prototype. The rendered flares exhibits well defined ghost shape and little noise with or without complex deformations of the ray grid.

Rendering lens flare in a scene As lens flare deals with interaction between the ray and the lens, the scenes in previous examples only contains a few point light sources. Fig. 11 shows an example where lens flare in rendered with a general scene. In my implementation, I choose to outputs two layers of images, a scene image storing the irradiance coming from objects in the scene, and a lens flare image containing the starburst pattern and the ghosts. These two components are added together to create the final rendered image.


Double Gaussian 100mm, F/1.39 [7]

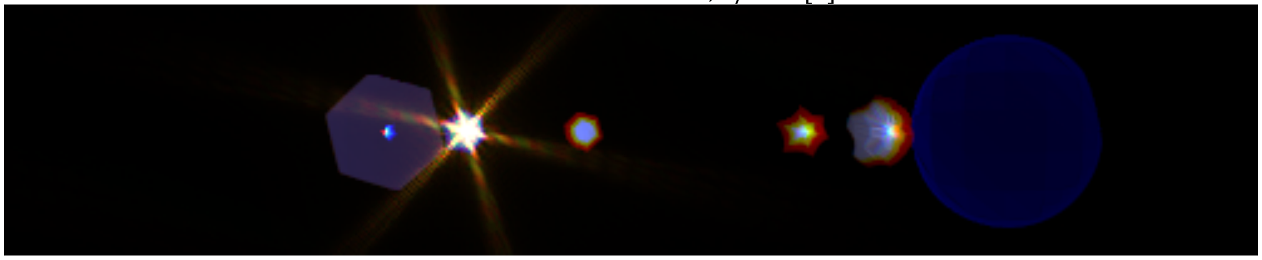
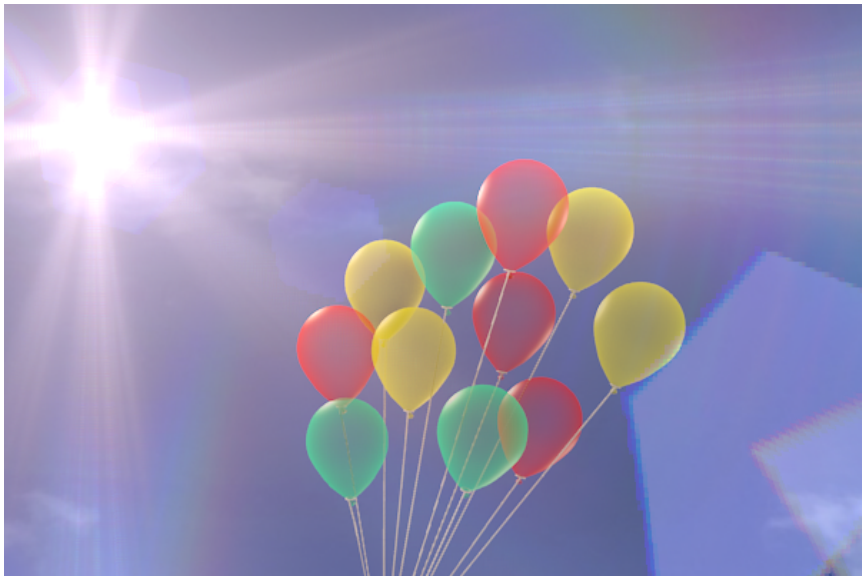
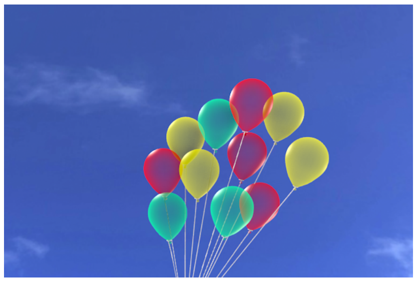

Canon 85mm F/1.2 [8]

Fig. 10 Two additional lens flares.


(a) The final output of lens flare rendering


(b) Scene image

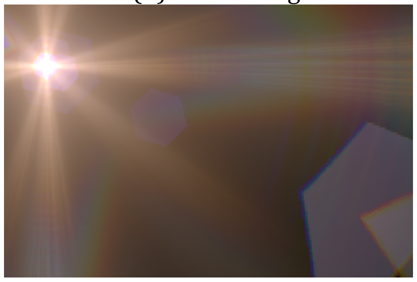

(c) Lens flare image

Fig.11 Lens flare generated for a 100mm Double-Gauss lens at F/8.

7. Conclusion

In this project, I investigated various techniques related to rendering lens flare, which occurs in the form of multiple ghosts with color variations as well as a starburst pattern around the main image of the light source.

The challenge of rendering lens flare lies in the high dimensionality of rays associated with this effect, causing a lot of noise sampling based methods. Therefore, I decide to use texture mapping based approach [4] for rendering, which includes three steps:

(1) Divide the front pupil into a sparse quad-mesh and trace rays through each vertex of the mesh; to handle dispersion, indices of refraction and Fresnel reflectivity are evaluated per wavelength and ray tracing is performed per- wavelength too.
(2) Based on the ray tracing results, compute an aperture truncation texture that maps densely discretized front pupil coordinates to a binary indicator of light transportation; To account for ringing, transform this texture with FrFT.
(3) Map the aperture image to the sensor plane, normalize the solid angle accordingly.

To improve the efficiency of the method, insignificant ghosts are ignored during the rendering. To speed up rendering of individual ghosts, I designed a local mesh refinement algorithm to adaptively subdivide the mesh where lacks information and needs more rays to be traced.

My implementation is able to render interesting lens artifacts including optical aberrations starbursts, and lens flares of complex deformations. However, it still has a few limitations. First, the approximation of diffraction using FFT and FrFT of a constant order is very rough. As stated in Appendix, this problem can potentially be solved by generalizing the FrFT theory to complex valued ordered transforms. Second, although the adaptive subdivision method typically works very well, I found it problematic when dealing with apertures with holes and sharp corners .The only way to resolve this issue is to sample the front pupil more aggressively. Finally, I have tried to adapt my algorithm to area light sources by approximating it with a number of point light sources, it typically generates a lot of artifacts due to insufficient sampling. As a future work, it is worthwhile to see if this can be approximated by blurring each ghost image with the distribution of intersection between the sensor and rays from various samples on the light source.

Acknowledgement

My implementation is build up from both PBRT and the starter code of assignment 348b Assignment 3 of Year 2011(https://graphics.stanford.edu/wikis/cs348b-11/Assignment3). By the time this report in completed, I just noticed that in this year, the assignment requires an simple implementation of lens flare, equivalent to the content till section 2.3 in this report.

Reference

[1] Shooting Challenge: Lens Flare Gallery, http://gizmodo.com/5562538.

[2] M. Pharr and G. Humphreys, “Physically Based Rendering: from Theory to Implementation”, Second Edition, Elsevier 2010.
[3] C. Kolb, D. Mitchell, and P. Hanrahan, “A realistic camera model for computer graphics,” presented at the the 22nd annual conference, New York, New York, USA, 1995, pp. 317–324.
[4] M. Hullin, E. Eisemann, H.-P. Seidel, and S. Lee, “Physically-based real-time lens flare rendering,” *SIGGRAPH '11: SIGGRAPH 2011 papers*, Aug. 2011.
[5] A. Keshmirian, “A physically-based approach for lens flare simulation,” 2008.
[6] H. Ozaktas, “Fractional Fourier optics,” *JOSA A*, 1995.
[7] W. J. Smith, “Modern Lens Design”. McGraw-Hill Professional, 2005.
[8] Japan patent 2011-253050.

Appendix. Discussion on implementation of diffraction

I have attempted to use the original theory of Fractional Fourier Transform to compute the order for the transform. It would be nice to have this functionality because the distance rays of different ghosts travels can be fairly different, and should correspond to different orders of the transform. Here is an extreme example: consider a highly defocused “main” image of the light source: its diffraction image should be a ringing pattern rather than a starburst. The orders of these two patterns are very different: the ringing pattern is close to 0, while the starburst is 1.

The Fractional Fourier Transform theory[6] states that for two spherical surfaces of radius R_1 and R_2 with a spacing of d, their amplitude distribution of light $q_1(x)$ and $q_2(u)$ are related by

$$q(x'/s_2) = s_1 A(d, \lambda, \phi) \cdot (\mathcal{F}^{2\phi/\pi} q_1)(x/s_1),$$
$$A(d, \lambda, \phi) \ = \ \frac{exp(i2\pi d/\lambda) exp\left[i(\pi\widehat{\phi}/4 - \phi/2)\right]\sqrt{\lceil \sin\phi \rceil}}{\sqrt{i\lambda d}}.$$

The coordinate scaling factors and the phase of transform are function of d, R_1 and R_2:

$$s_1 = \frac{\sqrt{\lambda d}}{\sqrt[4]{g_2/g_1 - g_2^2}}, s_2 = \frac{\sqrt{\lambda d}}{\sqrt[4]{g_1/g_2 - g_1^2}}, \phi = \csc^{-1}\frac{s_1 s_2}{\lambda d}.$$

Here $g_1 = 1+d/R_1$, $g_2=1-d/R_2$.

Thus, ideally given a lens prescription, we can find the phase of transform ϕ_k between each pair of surfaces. Since the order of FrFT is additive, the order of transform on the final image is the sum of the phases ϕ_k along the light path.

However, to get a real valued solution for the scaling factors and the transformation phases (equivalently, the order) requires both $g_2/g_1 - g_2^2$ and $g_1/g_2 - g_1^2$ to be positive. But substituting the variables with existing lens prescriptions shows this is not the case. Potentially, this difficulty can be resolved by finding complex-valued orders, but I ended up just implementing the technique in [4], which is a very rough approximation to the diffraction.