

Lens Flare Prediction based on Measurements with Real-Time Visualization

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Walch Andreas, BSc

Matrikelnummer 0926780

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Prof. Dr.Dr.h.c. Purgathofer Werner

Mitwirkung: Dipl. Ing. Luksch Christian

Dipl. Ing. Dr. Maierhofer Stefan

Dipl. Ing. Mag. Schwärzler Michael

Wien, 2. März 2017

Walch Andreas

Purgathofer Werner

Lens Flare Prediction based on Measurements with Real-Time Visualization

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Walch Andreas, BSc

Registration Number 0926780

to the Faculty of Informatics

at the TU Wien

Advisor: Prof. Dr.Dr.h.c. Purgathofer Werner

Assistance: Dipl. Ing. Luksch Christian

Dipl. Ing. Dr. Maierhofer Stefan

Dipl. Ing. Mag. Schwärzler Michael

Vienna, 2nd March, 2017

Walch Andreas

Purgathofer Werner

Erklärung zur Verfassung der Arbeit

Walch Andreas, BSc
Zur Spinnerin 53/8/1

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 2. März 2017

Walch Andreas

Acknowledgements

Many thanks to all the people at the *VRVis* and the *Vienna University of Technology* who supported and guided me during my studies. In particular to Werner Purgathofer and Stefan Maierhofer, who supervised my work on this master thesis. I want to thank Michael Schwärzler and Christian Luksch for their useful hints and comments.

I want to thank the *Vienna University of Technology* and the *Faculty of Informatics* for the financial support, which allowed me to acquire the needed hardware for this project.

Many thanks to all my friends and university colleagues who helped and accompanied me on my way through all the exercises and exams. In particular to Attila Sabzo and Simon Brenner, who proved to be awesome team members and true friends.

Finally, I want to thank my family for their support, help and patience. Special thanks to my parents for providing me the possibility to study and their never-ending trust and patience.

I want to dedicate this work to my former supervisor Robert F. Tobler.

Kurzfassung

In dieser Diplomarbeit wird eine neue Methode zur Visualisierung von realistischen Linsenreflexionen (Lens Flare) entwickelt. Linsenreflexionen werden durch verhältnismäßig helle Lichtquellen im Bild sichtbar und entstehen durch interne Reflexionen im Linsensystem der Kamera. Das Auftreten von Lens Flare wird meist als störend empfunden, da sie das eigentlich aufgenommene Motiv überblenden und den Gesamtkontrast verringern. In künstlich generierten Bildern werden Lens Flare jedoch oft nachgeahmt, um einen höheren Grad an Realismus zu suggerieren. In der Computergraphik wird für die Generierung von Lens Flare oft das gesamte Linsensystem einer Kamera simuliert. Die exakte Beschaffenheit aller eingebauten optischen Elemente, im speziellen der Anti-Reflexions-Beschichtungen, sind hingegen meist nur schwer einzusehen und können oft nur geschätzt werden. Die Qualität der Simulation hängt stark von den verfügbaren Beschreibungen ab. Die Simulation ist daher sehr unflexibel.

Die in dieser Arbeit entwickelte Visualisierungsmethode ist nicht auf die exakte Beschreibung des Linsensystems einer Kamera angewiesen, da sie direkt mit den aufgenommenen Daten der Kamera arbeitet. Durch Analysieren der aufgenommenen Bilder kann ein Model erstellt werden. Das Model kann für Prognosen von Lens Flare unter verschiedenen Beleuchtungssituationen verwendet werden. Dank GPU-Implementierung ist es für Realtime-Visualisierungen geeignet.

Abstract

Lens flare is a visual phenomenon caused by interreflection of light within a lens system. This effect is often undesired, but it gives rendered images a realistic appearance. In the area of computer graphics, several simulation based approaches have been presented to render lens flare for a given spherical lens system. An accurate model of the lens system and all its components is crucial for a physically reliable result. Since the effect differs from camera to camera, these methods are not flexible, and the internal parameters – especially the anti-reflection coatings – can only be approximated.

In this thesis we present a novel workflow for generating physically plausible renderings of lens flare phenomena by analyzing the lens flares captured on a camera. Furthermore, our method allows to predict the occurrence of lens flares for a given light setup. This is an often requested feature in light planning applications in order to efficiently avoid lens flare prone light positioning. A model with a tight parameter set and a GPU-based rendering method allows our method to be used in real-time applications.

Contents

Kurzfassung	ix
Abstract	xi
Contents	xiii
1 Overview	1
2 Introduction	3
2.1 Problem Definition	3
2.2 Aim of the Work	5
3 Background - Lens Flares	9
3.1 Camera and Lens System	10
3.2 Lens Flare Elements	12
3.3 Physiological effects	16
3.4 Countermeasures to Avoid Lens Flare	17
4 Related Work	23
4.1 Lens System Models	24
4.2 Light Propagation Formulation	26
4.3 Camera Models	27
4.4 Lens Flare Simulation and Rendering	28
5 Data Acquisition	37
5.1 Sampling A Lens Flare	37
5.2 Assumptions	38
5.3 Prototype Setup	38
6 Ghost Rendering Primitive - Lens Flare Model	41
6.1 Analyzing the Sampled Ghosts	41
6.2 Requirements For the Rendering Primitives	42
6.3 Creating the Model	42
	xiii

7	Optimization - Finding the Ghost	49
7.1	Cost Function	50
7.2	Optimization Strategy	53
7.3	Finding Lens Flare in the Acquired Images	54
8	Real-Time Visualization	57
8.1	Interactive Visualization	57
8.2	Lens Flare Incorporated Into An Application	58
9	Lens Flare Occurrence Prediction	61
9.1	Occurrence Estimator	61
9.2	Context	62
10	Results	63
10.1	Result - Acquisition	63
10.2	Result - Lens Flare Model	66
10.3	Result - Optimization	68
11	Conclusion and Future Work	75
	List of Figures	77
	List of Algorithms	78
	Acronyms	79
	Bibliography	81

CHAPTER 1

Overview

To generate lens flare renderings based on measurements, we introduce a novel workflow. The thesis provides background information and current rendering approaches. Our whole workflow is described in detail. The thesis is structured as follows:

Chapter 2 of this thesis states the problem definition and the aim of the work. By providing an example case, we demonstrate how affected fields, like light planing, could benefit from our solution.

Chapter 3 gives background information on lens flares and cameras – especially, where lens flare occurs, and how its appearance in terms of shape or color is influenced by the lens system. Additionally, the associations with lens flare and possible countermeasures are described.

Chapter 4 explains the available optical system models, and how these models are incorporated in the field of computer graphics in terms of camera models. The progress of lens flare rendering over the years, up to the current state-of-the-art simulations, are described in detail.

Chapter 5 describes how the data for our data-driven workflow is generated. The developed prototype and its components are discussed thoroughly.

Chapter 6 discusses how lens flare can be described by a model and what assumptions the model has to fulfill to generate a valid visual representation. The developed lens flare model is necessary for the next stages of our workflow to compress the captured data and for real-time rendering.

Chapter 7 gives insight on how the lens flare rendering elements (Chapter 6) can be used to optimally describe a sample from the acquisition stage (Chapter 5). The developed optimization algorithm and its cost function as well as the encountered problems are described in detail.

In chapter 8 the results from the best fitted models (Chapter 7) of the captured data (Chapter 5) can be used to describe the appearance of lens flare elements over the captured range. To generate a smooth real-time visualization, each parameter of the model is compressed over the whole range by a function, which allows to fill the gap for uncaptured sub-images.

Chapter 9 describes how the real-time visualization (Chapter 8) can be reused to predict the occurrence of lens flare for a given camera-light constellation.

Chapter 10 presents results from the different workflow stages.

In chapter 11, we conclude our work and point out possible future improvements.

Introduction

This chapter describes the problem definition and lists possibly affected application fields and scenarios.

2.1 Problem Definition

This work is a practical approach to gain information on lens flare occurrences for a lens system for which no specifications are known. Lens flare is a visual effect that appears due to internal reflections within a lens system or due to scatterings caused by lens material imperfections(Chapter 3). Lens flares and glares reduce the image's information content, since actual image features are overdrawn by them (Figure 2.1). Apart from these visible lens flare shapes and artifacts, the output image is also overcast by haze, which heavily influences the image quality. For these reasons, lens flare is often regarded as a disturbing artifact, and camera producers therefore develop countermeasures to reduce their visual impact (Chapter 3.4). In other applications though, lens flares are used as a stylistic element, and are therefore a desired effect in photography or movies.

The intensity of lens flare heavily depends on the camera to light source constellation. In controlled environments like broadcast or movie studios, the light sources can be adjusted and set up according to their requirements – but in general, most light setups are static and cannot be easily changed. The light placement for static scenes are defined in a planning or construction stage. In case of large scale projects like sport arenas, museums or industry buildings, a vast number of lights have to be placed in the light planning stage. While in sport arenas live TV broadcasts play an important role, museums and industry facilities have to deal with surveillance systems. In both examples, many cameras with different lens systems and lights are included, which implies the possibility of lens flare occurrence. Both affected fields are sensible to image quality, making the occurrence of lens flare therefore an undesired effect. While ignoring the possible occurrence of lens flare in the planning stage, affected combinations have to be rearranged afterward.

2. INTRODUCTION

Modifications in a late stage can become quite costly, especially if equipment has to be changed or the building has to be modified. In case of flexible camera to light constellation, the adjustment is still time consuming, and often based on trial-and-error methods.



Figure 2.1: Outdoor lens flare¹

A lens flare-aware light planning stage allows the light designer to carefully elaborate an optimal camera and light setup to avoid lens flares. In such an early phase, the costs are quite low, and problematic constellations can easily be prevented. For inevitable constellations, the lens flare occurrence can at least be minimized by choosing the optimal camera or by changing the light source. Moving lights or cameras can also benefit from lens flare occurrence prediction by analyzing the occurrence over time or moving ranges. In outdoor scenes, the sun triggers lens flares quite easily due to its extreme brightness. In case of live sport broadcasts, the sun trajectory can be included into the lens flare occurrence prediction for the given time span. By knowing the time dependent sun position, the camera to light constellation can be simulated over the whole broadcasting period. This allows to find the optimal camera position for a single camera setup in order to avoid lens flares. Additionally, for multiple camera setups, the timetable of which cameras to use can be optimized in advance by suggesting a switching order.

¹Source: <https://commons.wikimedia.org/w/index.php?curid=642778>

One approach to predict lens flare occurrences is simulation. A precise description of the lens system and all its properties are essential to carry out a physically correct simulation. The lens flare characteristics depend heavily on the lens system parameters. Especially the anti-reflection coatings (Chapter 3.4) have a great impact on lens flare occurrence. But these carefully designed anti-reflection coatings are mostly kept under wraps by the camera manufacturers, or cannot be accessed easily for any camera system, because the quality of the lens system depends on them. Due to this inaccessible or hardly available information and incomplete knowledge about the lens system, in most cases only the camera producers themselves are able to carry out a completely accurate simulation. In general, the lens system parameters can only be approximated, which leads to erroneous simulations. Another aspect to consider when using simulations for lens flare prediction, the simulation run-time increases drastically the more complex the lens system gets.

Simplified or even artificially generated lens flare effects, as used in computer games, are not suitable for lens flare prediction. These lens flare renderings are primarily focused on artistic aspects and visual output, but not much attention is paid to their physical accuracy (Chapter 4.4.2).

Another way to predict lens flare occurrence can be done by examining real-world measurements. The camera and all its hidden parameters can be interpreted as a black box. The input is a bright light source, and the black box's output are raw images. The raw images contain all lens system-dependent transformations of the input light, including physically accurate lens flares. This approach does not rely on any internal specifications of the lens system, and is therefore very flexible and independent from the camera manufacturers. To generate a model from measurements, a lot of data has to be captured, analyzed and compressed, though.

Due to the shortcomings of the above-mentioned simulation based approaches, we intend to find a data driven workflow to generate a physically accurate simulation of lens flares. Furthermore, there exists no tool in the light planing stage so far to predict the occurrence of lens flares. Such a tool could enrich the planning stage to be more efficient in many aspects: Apart from decreasing overall cost and planning time, the solution's quality could also be increased. The development and need of such a tool can be underlined by scenarios as depicted in Figure 2.2, where a live football broadcast is disturbed by lens flare.

2.2 Aim of the Work

The overall goal of this work is to find a workflow to predict lens flare occurrences for a certain camera light constellation without knowing any specific internal parameters of the lens system (except known camera parameters, like focal length and aperture stop). A measurement-driven approach seems to be the only option left, which is physically accurate, and does not rely on any approximations.

²Source: Sky Bundesliga HD1, Borussia Dortmund vs. FC Bayern München at 28.04.2015



Figure 2.2: Football live broadcast with disturbing lens flares²

The workflow contains multiple stages, which have to be first specified and later optimized. As it is a data-driven approach, the first step is to gather sample data of lens flares for different light camera constellations. To efficiently capture the required large amount of data, a hardware setup has to be developed (Chapter 5). Afterwards, the sampled data has to be analyzed (Chapter 3), and a model has to be found to represent the lens flare elements (Chapter 6). The model's parameters have to be adjusted to best fit the sampled data (Chapter 7). By combining the results of the best fitted models, a compressed

representation for the whole captured range can be generated. This compression can be further be used for real-time visualization (Chapter 8) and lens flare occurrence prediction (Chapter 9).

The proposed method can be of great use in the planning stage, allowing to place luminaries as well as cameras in a more effective way by avoiding lens flare artifacts. Additionally, this prediction can also be useful to find especially likely lens flare constellations to produce artistic camera shots.

Background - Lens Flares

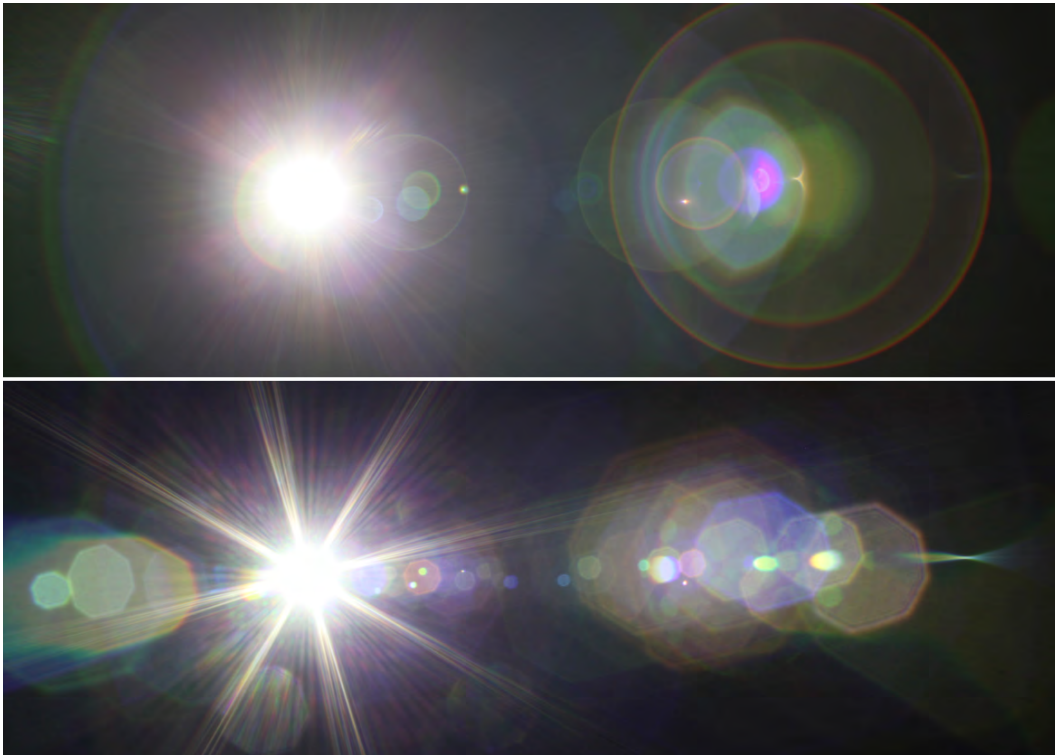


Figure 3.1: Complex lens flare [HESL11]

Lens flare, also known as *veiling glare* in the field of photography and video, is a visual effect which can be observed while a lens system receives light from a considerably brighter light source than the rest of the scene [HESL11]. The lens flare shape and appearance changes from camera to camera, and strongly depends on the built-in lens

system. Still, lens flare effects can also differ within the same camera when adjusting its parameters (Chapter 3.1). However, the characteristics of the occurring lens flare elements are generally similar in most cases [Kes08].

Lens flare is often regarded as an undesired effect, as it reduces the image quality and information gain. While the bright light source is visible in the field of view, the close area around the light source is heavily affected by haze and glare. In case of strong haze, the image's contrast can be noticeably reduced [Mch05]. The close surroundings of the bright light source overexpose the image background drastically. Lens flare is caused by interreflections within the lens system (Chapter 3.2) and can even occur while the light source is outside of the angle of view due to stray light (Figure 3.2).

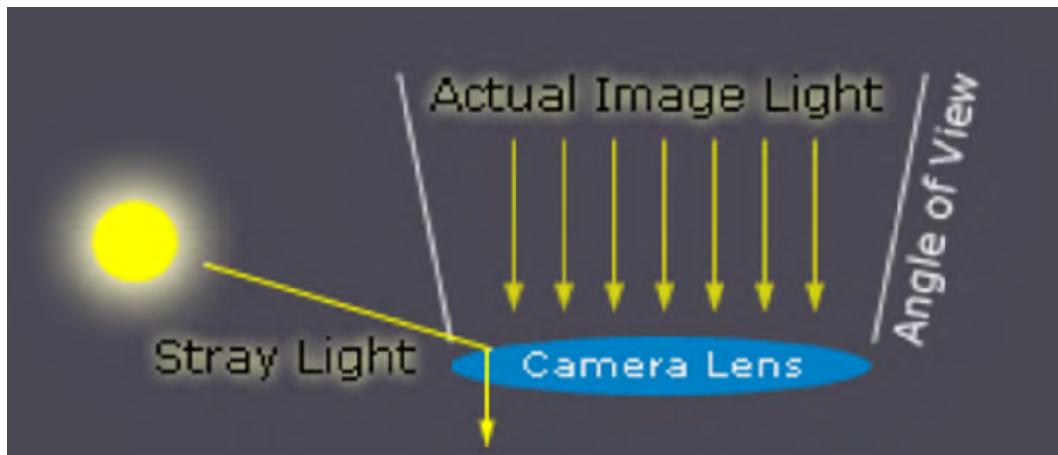


Figure 3.2: Illustration of stray light and angle of view [Mch05]

On the other hand, lens flare can also be used intentionally like in movies [Woe09] as part of an artistic expression, or in photography by careful composition. Animation movies [Pix08] and recent computer games [Tow12] use lens flares as stylization element, or to imply a higher degree of realism.

3.1 Camera and Lens System

A camera is an optical device, consisting of a *lens system* and a light sensitive sensor to record images. The main changeable camera parameters are *aperture*, *exposure time*, *film speed* and optionally a focal length (Figure 3.3). Each of these parameters influences the final image in a visually noticeable way, and depending on their combined overall configuration, they generate different visual effects. The focal distance of a camera depends on the lens system's arrangement and lens properties. Zoom-able cameras allow to change the focal distance by moving lenses or lens groups within the lens system. While changing the lens groups, the non-image-forming paths are also affected (Chapter 3.2.1). The aperture controls the opening size, through which light must travel to reach the sensor (Figure 3.14). Focal length and aperture in combination define the *depth of field*.

The sensor size and the focal length form the *angle-of-view*. The exposure time sets the amount of time the sensor is exposed to light. In case of a moving object and adequate exposure time, the object can be mapped multiple times onto the image plane (i.e. the sensor). These multiply captured light rays generate a blurred shape behind the moving object, also known as *motion blur*. The film speed is responsible for the sensitivity of the film or sensor.

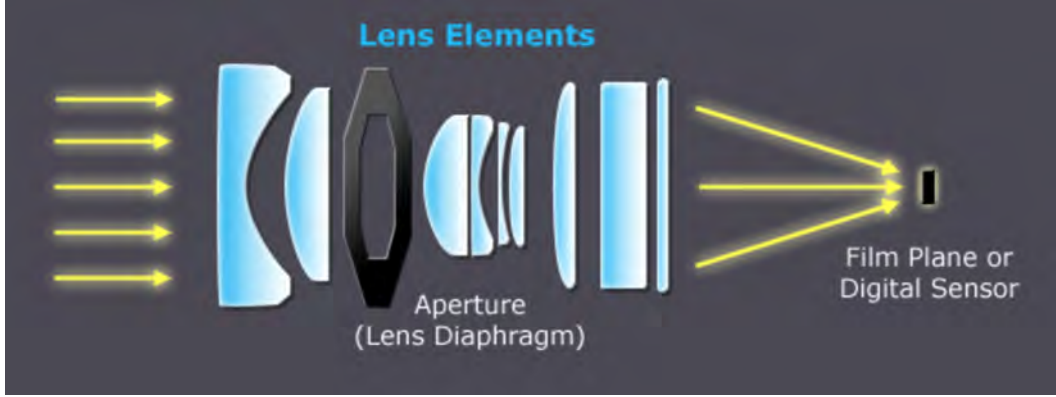


Figure 3.3: Camera model illustration [Mch05]

The simplest camera consists of a single simple convex lens, but suffers heavily from *optical aberrations*. Optical aberrations can defocus and distort the output image. Unfortunately, a simple convex lens cannot eliminate these unwanted transformations and is therefore not used in consumer cameras. To reduce optical aberrations, an arrangement of multiple lenses along the *optical axis* is used (Chapter 3.2.3). The shape and material properties of each lens affects the behavior of the passing light. A lens is made out of a translucent material and can be used to focus light onto a certain point, or to disperse light into different directions. *Snell's law* describes the angles of incidence θ_i and refraction θ_t while light passes through an interface of two different materials, depending on the material's refraction index n_1 and n_2 (Equation 3.1).

$$\frac{\sin \theta_i}{\sin \theta_t} = \frac{n_2}{n_1} \quad (3.1)$$

The refraction index describes how different wavelengths change their direction while passing another medium. The *Fresnel Equation* (Equation 3.2) describes the amount of reflected light for polarized incoming light. In case of unpolarized light, the amount of reflected light R is simply the average of R_s and R_p . The reflection depends on Snell's angles θ_i and θ_t and on the material's refraction indices n_1 and n_2 . The amount of refracted light (transmitted) T is due to the conservation of energy $T = 1 - R$.

$$R_s = \left| \frac{n_1 \cos \theta_i - n_2 \cos \theta_t}{n_1 \cos \theta_i + n_2 \cos \theta_t} \right|^2 \quad R_p = \left| \frac{n_1 \cos \theta_t - n_2 \cos \theta_i}{n_1 \cos \theta_t + n_2 \cos \theta_i} \right|^2 \quad (3.2)$$

Visible light consists of a range of wavelengths, which get bent into different directions according to the Fresnel Equation. This leads to *chromatic aberrations*, where the incoming white light gets refracted and is split up into multiple visible colors. To control and reduce unwanted reflections within a lens system, camera producers carefully design anti-reflection coatings. The anti-reflection coatings have an impact on the lens flair's color. Further details in chapter 3.4.

3.2 Lens Flare Elements

Several visible lens flare effects originate from the center of the light source. The effect can be distinguished into three different kind of element types. The visibly most prominent effect is a *star-shaped* element, where multiple bright streaks are radiating outwards from the light source center [Kes08]. This glare streaks are caused by diffraction at the edge of the aperture (Figure 3.7). For non-circular aperture shapes, the number of flares depends on the n -polygon. Perpendicular to each edge are yielding two spikes 180° apart [Kin92]. In case of an even number of edges, the flares overlap, resulting in n streaks, while an odd number of edges results in $2 * n$ flares (Figure 3.4).

The close area around the light source is also affected from a less distinct glare, called *halo*, which fades out gradually from the light source center. According to [Cha07], the halo effect is caused by intense scattering of light in the lenses.

The other prominent effect is a series of circles and rings aligned on a line, which crosses the light source center and the image center. The shapes on this array are so-called *ghosts*, and their position and shape differ strongly from camera to camera. These ghosts are the most complex lens flare elements, and their occurrence is non-trivial to predict compared to the other source-centered lens flare elements.

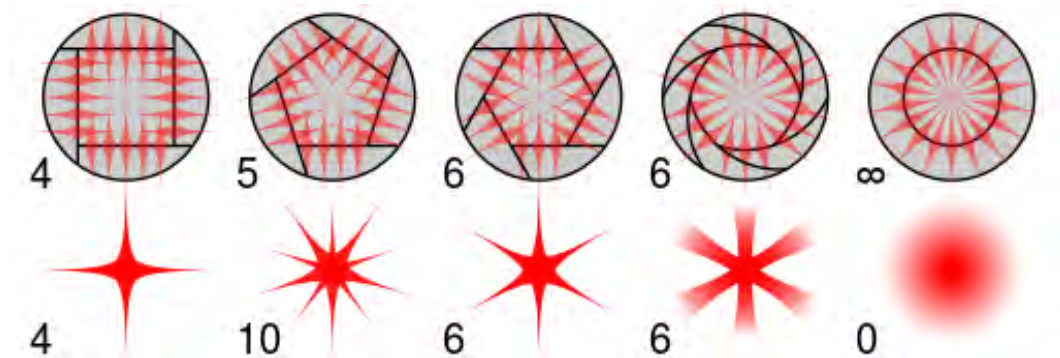


Figure 3.4: Comparison of diffraction spikes for different apertures¹

¹Source: By Cmglee (Own work) [CC BY-SA 3.0 (<http://creativecommons.org/licenses/by-sa/3.0>)], via Wikimedia Commons

3.2.1 Ghosts Occurrence

In a lens system, it is intended that the image-forming light rays find a direct way to the sensor while passing all lens elements to create the desired image. But in reality, imperfections in the lens material and certain lens combinations can cause internal reflections. As a result, the light passes the lens system in an unexpected way while getting reflected multiple times within the system (Figure 3.5). If these non-image-forming paths reach the sensor, they form visible artifacts called ghosts [HESL11].

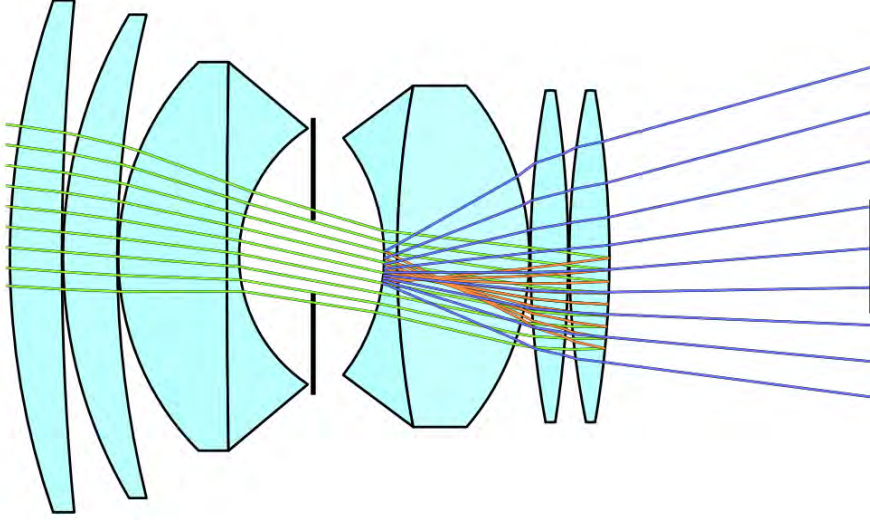


Figure 3.5: A possible double-reflection path [HESL11]

The number of ghosts depends on the possible reflections within a lens system. Only an even number of reflections have the possibility to reach the sensor, while an odd number of reflections are most likely to end up at the entrance pupil [LE13]. According to Koren [Kor07] the formula 3.3 states the exact number of possible ghosts, where R is the total reflections, and M the number of air-to-glass surfaces. In case an additional filter is applied, the number even increases to $R_{withFilter} = R + 4M + 1$. More complex lens systems contain a lot of lens groups, and therefore offer a higher number of possible non-image-forming paths than simpler systems. Zoom-able lens systems generally tend to be more complex than prime systems (Figure 3.6). Therefore, when not considering the camera construction quality, prime systems are less likely to be affected by lens flares than zoom-able devices.

$$R = (2M - 1) + (2M - 2) + (2M - 3) + \dots + 1 = M(2M - 1) = 2M^2 - M \quad (3.3)$$

Still, not all possible reflection paths carry enough energy to produce a visible ghost in the resulting image. According to the *Fresnel Equation* (3.2), the light energy is split

up at each material-intersection into reflected and refracted energy, which implies that each reflection step reduces the amount of energy carried on by the light ray. Therefore, only paths with a low number of even reflections are able to carry enough energy to have an impact on the resulting image [HESL11]. In zoom-able lens systems, the focal length can be adjusted by moving the lenses within the lens system. These lens system changes affect also the ghost forming paths, which causes the ghosts to change their occurrence even within the same camera.

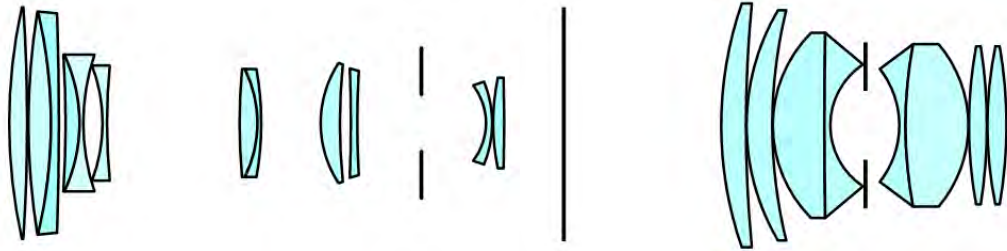


Figure 3.6: Geometry comparison of zoom systems - Nikon 80-200mm f/2.8 vs prime system - Angenieux Biotar 100mm f/1.1 [HESL11]

3.2.2 Ghost Shape

Each valid ghost path has to pass the aperture at least once. The aperture shape alters the appearance of the resulting ghost. A path which passes the aperture multiple times can even have a more complex shape due to multiple clippings by the aperture. The aperture controls the opening size via mechanical components. They are intended to adjust the circular-like opening to various sizes. The components are usually blade-shaped, and by increasing the number of used components, the desired circular shape can be better approximated. In case of a fully opened aperture (smallest f -stop), the shape is a perfect circle, as the blades rotate behind the aperture ring of the camera body. While increasing the f -stops (reducing the opening size), the blades are moved and rotated together (Figure 3.14). According to the number of blades, the ghosts have a regular n -polygon-like shape (Figure 3.7).

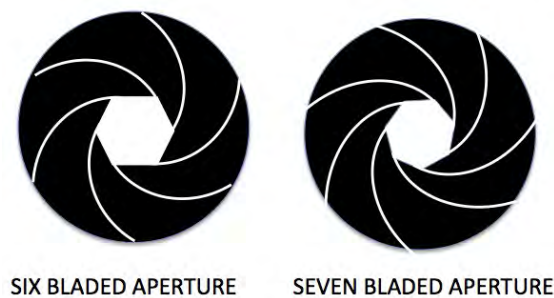


Figure 3.7: Different aperture shapes caused by varying blade count²

Higher-end lenses put a lot of effort into developing mechanical components that maintain the circular aperture shape at all aperture-stops. Therefore, the visual appearance of the ghosts can also differ within the same camera when changing the aperture-stop. Additionally, the shape of the ghost can also be clipped by the entrance pupil's shape of the lens system [HESL11]. The ghost's shape is featured by ringings close to the edges, which are caused by *near-field Fresnel diffraction* [JKL⁺16].

3.2.3 Ghost Position

The camera position and its orientation and light position is vital for the position of the ghosts on the image plane. Depending on the enclosed angle α between the camera's optical axis and camera to light-vector the non-image-forming light paths reach the sensor at different spots (Figure 3.8). The ghosts are more or less aligned on a ray, which runs through the image center and the light source center [Kes08]. The small offsets are likely to be caused by lens irregularities or imperfections. While rotating the camera toward the light source, the ghosts are getting closer together. On the other hand, while rotating the camera away from the light source, the ghosts start to fan out, and they are less likely to overlap (Figure 3.9). When the camera directly faces the light source, the ghosts overlap each other completely.

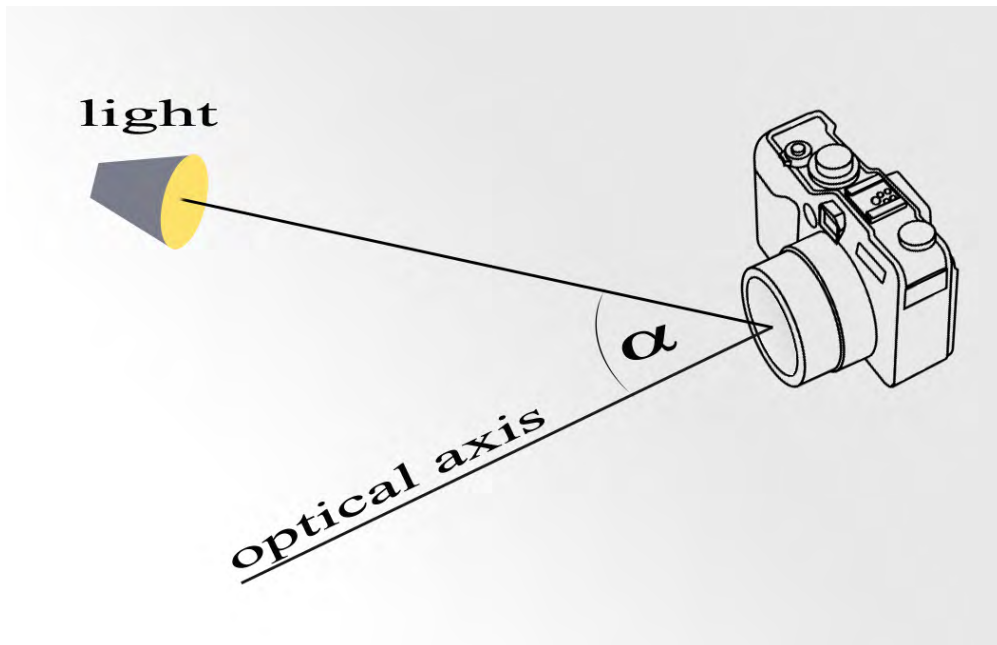


Figure 3.8: Illustration of a camera light constellation³

²Source: <http://www.tested.com/tech/photography/2286-how-your-dslr-camera-lens-aperture-really-w>

³Camera-Model By Rama (Own work) [CeCILL (http://www.cecill.info/licences/Licence_CeCILL_V2-en.html)], via Wikimedia Commons

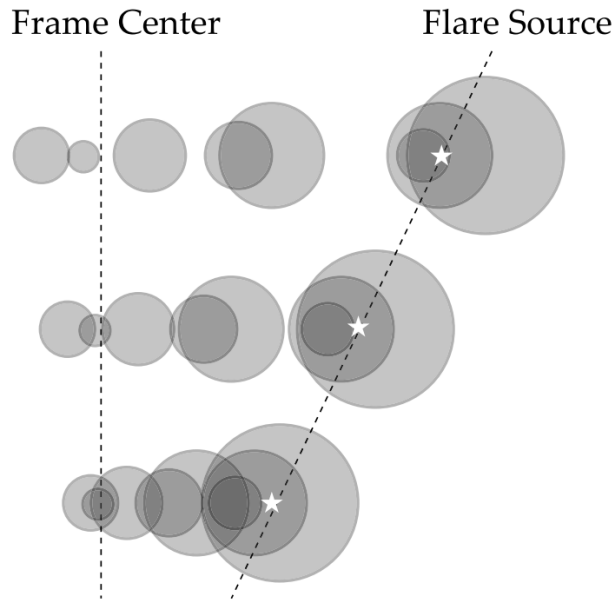


Figure 3.9: Illustration of how ghosts behave while the camera is rotated [Kes08]

As mentioned in chapter 3.1, camera lenses are usually aligned on the optical axis, which coincides with the axis of *rotational symmetry* and the image center. The distance of the projected light on the image plane and the image center is equivalent to the angle between light position and camera orientation. While maintaining a certain angle and rotating either the camera or changing the light position, the ghost's positions are constant with regards to the image center and rotate accordingly. The rotational symmetry property simplifies the investigation of the ghosts' behavior drastically, as their behavior only differs while changing the distance of the optical axis to the light source. This allows to capture the complete behavior by simply rotating the camera from the light source along the horizon away.

3.3 Physiological effects

As lens flare is a real-world phenomena in lens system dependent imagery, their occurrence is well-known. The absence of lens flare in artificially generated images can reduce the perceived realism. Like other lens system dependent effects (depth-of-field or motion blur), lens flare is a well-trained visual perception and helps the viewer to extract additional information from an image or scene. The occurrence of lens flare indicates the presence of very bright light sources within a scene – even while the light source is outside of the camera frustum. By observing the ghosts behavior, the unknown light source position can be approximated outside of the camera frustum. Overexposed areas can also indicate

lens flare, as lens flare and bright light sources can clearly exceed the physical boundaries of a display device [RIF⁺09].

3.4 Countermeasures to Avoid Lens Flare

In various applications, the occurrence of lens flare is regarded as an undesired phenomena. Therefore, camera manufacturers try to eliminate or to reduce their occurrence rate and intensity. There are many ways to reduce lens flares either from the technical side or from photography theory.

3.4.1 Staging

By avoiding certain constellations, the occurrence of lens flare can be drastically reduced. This can be the result of an effective light planing stage, where camera and light source are well-placed in advance to avoid lens flare. The simplest solution is to place the bright light source behind the camera, but this changes the appearance of the scene extremely and probably in an undesired way. In another scenario, where the camera or light source cannot be moved freely within the scene, a well-used scene composition can reduce lens flare – for example by staging the scene elements in a way that the light source is obscured from the camera, which can help to avoid or at least reduce lens flares (Figure 3.10).



Figure 3.10: Scene composition can be used to reduce lens flare [Kor07]

3.4.2 Lens Hood

Lens flare is often caused by stray light. Camera manufacturer therefore try to design the barrel of the lens system in a more efficient way to prevent stray light from entering (Figure 3.11). In most cases, the regular barrel can be extended by a *lens hood*. The lens hood surrounds the entrance pupil and extends the barrel's shape into the scene. In case of heavy stray light within the scene, the lens hood can be attached in front of the camera. The theoretically best way to prevent any stray light from entering the lens system would be an extension to the furthest object following closely the angle of view. Therefore, the longer the extension, the more efficient it can blocks stray light. The lens hood is bound to the angle of view of a lens system. For zoom-able cameras, the lens hood is restricted by the widest possible angle of view – otherwise it would be visible in the resulting image. For wide-angle and zoom-able cameras, the lens hood cannot be optimized as easily as for long focus systems due to their large or changing viewing angle.

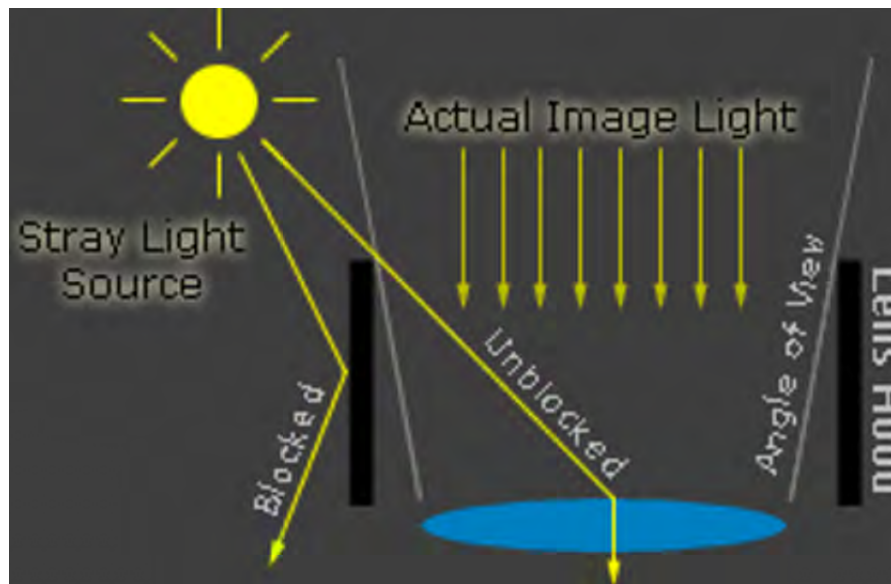


Figure 3.11: Illustration of how a lens hood can reduce stray light [Mch05]

More complex lens hood shapes also include the sensor ratio and reduce mechanical vignetting. Generally, the barrel inside as well as the lens hood itself is covered by a non-reflective material to prevent light from getting reflected in the first way [Mch05]. A lens hood is highly effective to reduce lens flare caused by stray light, but cannot reduce lens flare from visible light sources within the angle of view.

3.4.3 Anti-reflection coatings

A lens system, exposed to a bright light source within its angle of view, offers the incoming light many possible reflection paths to the sensor. Each reflection reduces the transported energy, but a sufficient bright light can still cause lens flare. Uncoated air-to-glass surfaces

reflect about 4% of the incident light. By using a simple anti-reflection coating, the reflection can be reduced to around 2%. Complex coatings composed of multiple layers can further reduce the reflectivity to 1% or less [Kor07] (Figure 3.12). By reducing the overall reflection possibilities, lens flare is less likely to occur.

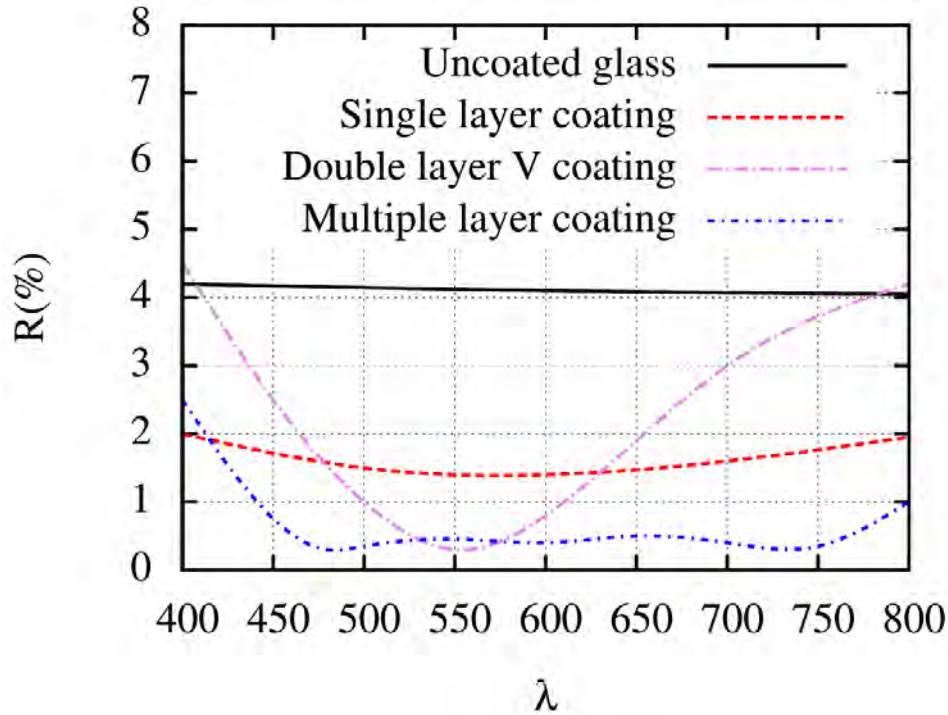


Figure 3.12: Reflectivity per wavelength and coating composition [SDHL11]

While light is getting reflected within the lens system, the reflected waves can superimpose and interfere with each other. Anti-reflective coatings intend to modulate light waves in way that the reflected light interferes with other light waves to cancel themselves out. This can occur if light waves share the same amplitude, but have the opposite phase. The incident angle and wavelength of the incoming light as well as the lens material are responsible for the reflection behavior of the incoming light. The anti-reflection coatings are designed to minimize reflections of a certain wavelength, causing the remaining reflected light to change its visible color. The color of the ghosts therefore depends on the anti-reflection coatings and their constellation.

The reflectivity of a lens system is therefore bound to the anti-reflection coating and their composition. To design an efficient coating combination is a complex optimization task and needs a high expertise in the field of optics. A wide range of parameters like light direction have to be considered. Therefore, the anti-reflection coatings are a seal of quality for the lens system, and the manufacturers keep their exact composition mostly unpublished [HESL11]. Figure 3.13 shows how much impact lens coatings can have, and

how the optimization process has been improved in the recent years (both images are shot under similar condition and settings) [Hen15].



Figure 3.13: Comparison of coatings [Hen15]. Top: 1980's Vivitar Zoom Bottom: 2014 Sony FE28-70

3.4.4 Aperture and Camera Type

The choice of camera type and used aperture stop can also reduce the amount of ghosts. In case of an high f -stop, the narrow opening restricts the possible light paths (Figure 3.14). The number of ghost forming light paths are therefore also strongly reduced. The possibility of a non-image-forming path to get reflected multiple times through the

aperture is also reduced. Therefore, the shape of the ghost is more likely to be rather simple than complex.

The lens system complexity often depends on the camera type. Prime cameras (fixed focal length) tend to be less complex than zoom-able devices. The lens system complexity limits the possible amount of inter-reflections paths within a camera. Hence, the amount of ghosts is generally smaller for less complex camera types than for more complex cameras [Kor07].

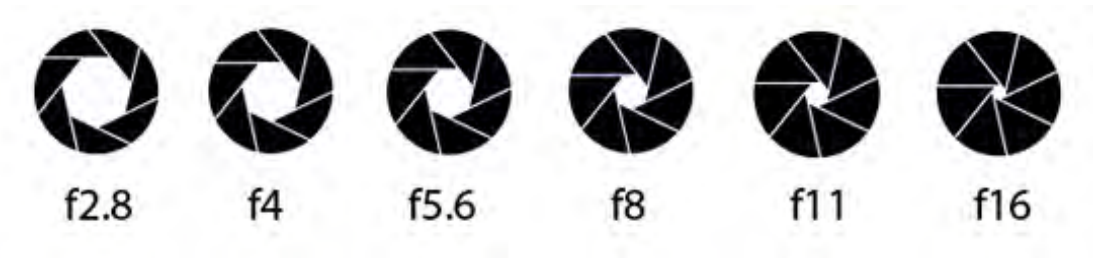


Figure 3.14: Different aperture stops and the resulting aperture shapes⁴

⁴Source: <https://picsart.com/blog/post/use-aperture-to-your-advantage-part-two/>

Related Work

A major research field in computer graphic deals with the rendering of realistic images. The light exchange and the light interaction within a virtual environment are important factors for the generation of a plausible image depicting the real world. These illumination computations depend on how well the different models describe the real world. The material model describes the behavior of light interactions within scene objects. The light transport can be significantly modified by the object's material properties. A too simplified material model can only approximate the complex material properties. Anti-reflection coatings have a complex reflectance and refracting behavior, and need a more sophisticated material model. The camera model mimics the characteristics of a real world optical system and is responsible for mapping the virtual scene onto the virtual image plane.

Depending on the quality of the camera model (Chapter 4.3) and the material model, the resulting image can become increasingly physically plausible. Simplified camera models lack of realism, as they cannot reproduce familiar visual real-world phenomena, like *depth-of-field*, *motion blur* (Chapter 3.1) and lens flares (Chapter 3). These effects are important visual cues and imply a high degree of realism. Depending on the application, some of these effects can be simulated by a well described camera model or can be approximated in a post-processing step.

In the movie industry, special effects are a frequently used tool to enrich real world images with artificially generated images. In case of a too simplified camera model, the rendered image cannot be seamlessly merged with real-world images, because artifacts are visible. An accurate simulation of cameras and sensors is also of high importance for *machine vision* or *optical system design*.

4.1 Lens System Models

In the field of optic design, various lens models have been developed to describe optical systems and their components.

4.1.1 Thin Lens Model

A camera contains a set of lenses (Chapter 3.1), which manipulate the incoming light by refraction and reflection. To simplify a simulation process, the *thin lens approximation* assumes the thickness of a simple lens, which is similar to the lens system characteristic, to be infinitesimally small. The refraction within the lens body is therefore not considered, and the passing rays are refracted only once at a single plane, the *principal plane* (Figure 4.1).

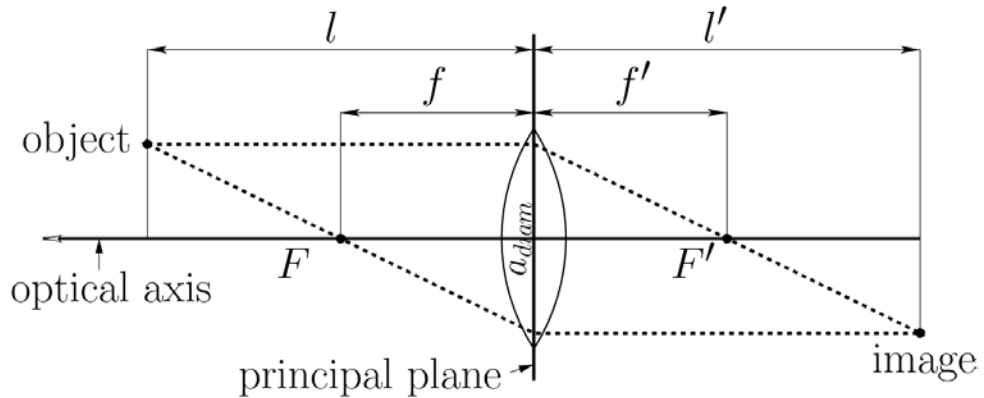


Figure 4.1: Thin lens model [BHK⁺03]

Light rays passing the *focal point* F and F' are refracted in parallel to the optical axis to the other side. The distance between the center of the lens and the focal point is called *focal length* f and f' . Points of a certain distance l to the principal plane are projected on the other side of the lens to a certain distance l' . If the image plane is equal to the distance l' , the point lies on the *focus plane* and its projection is a sharp point on the image plane. Points which do not lie on the focus plane are projected onto the image plane as blurred circles, as their projected points lie in front or behind the image plane. This simplified lens model is capable of generating depth-of-field, but is only appropriate for lens system simulations where the thickness of the lens is relatively small compared to the focal length [BHK⁺03].

4.1.2 Thick Lens Model

The thin lens model is suitable for approximating a lens system by one circular symmetric lens with negligible thickness, but real camera systems are based on a complex set of

various lenses, where the thickness has a great visual impact. To describe a camera in a more accurate way, the lens system can be better approximated by a *Thick Lens Model*. The thick lens model includes the thickness of a lens into its model to describe the light traversal through the lens in a more accurate way.

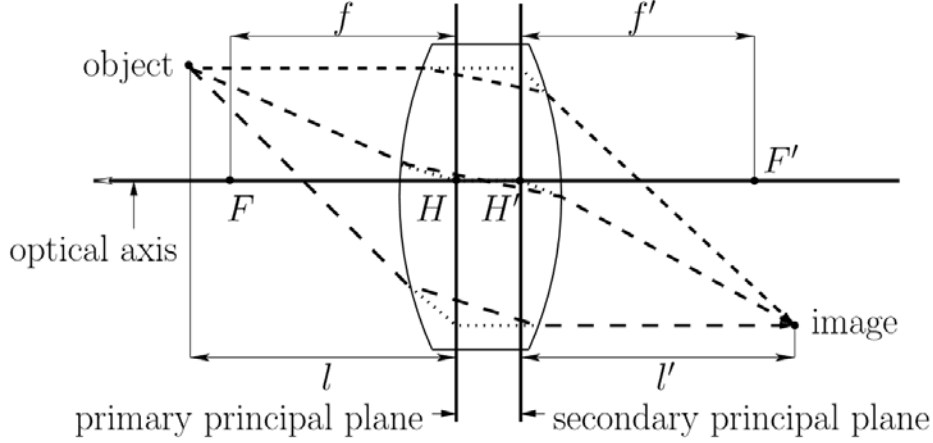


Figure 4.2: Thick lens model [BHK⁺03]. The dashed lines show the actual path of light rays. The dotted lines show how the light path is approximated by the thick lens model.

In contrast to the thin lens model, each side of the lens is described by its own principal plane (primary principal plane H and secondary principal plane H'). The distance between primary and secondary principal plane describes the lens' effective thickness. The thickness of the model can also be negative, if the secondary principal plane is in front of the primary principal plane [KMH95]. Each principal plane has its focal point F and respective F' , and the resulting focal distance f and f' (Figure 4.2). Rays intersecting the focal points F or F' refract (as described in the thin lens model chapter 4.1.1) and pass the principal plane parallel to the optical axis. The light propagation through the lens is thereby described by two thin lens models, where the light passes the gap between the principal plane parallel to the axis [BHK⁺03].

4.1.3 Optical System Description

An optical system can be specified algebraically up to a certain point. In case of spherical lenses, each lens can be described by its signed radius (convexity/concavity), thickness, material property and aperture stop position (Figure 4.3). These details are of high importance for a realistic light propagation simulation through a lens system. The data sheets are available from specialized sources [SO05] or from available patents. Unfortunately, in most cases these data sheets are not complete – especially the exact composition of the used anti-reflection coatings are not publicly available.

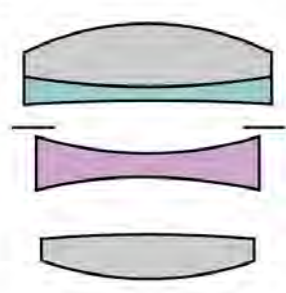
radius	thickness	material	radius	
42.970	9.8	LAK9	19.2	
-115.33	2.1	LLF7	19.2	
306.840	4.16	air	19.2	
aperture	4.0	air	15.0	
-59.060	1.870	SF7	17.3	
40.930	10.640	air	17.3	
183.920	7.050	LAK9	16.5	
-48.910	79.831	air	16.5	

Figure 4.3: Optical system description [SO05]

4.2 Light Propagation Formulation

For ray tracing techniques various models can be used to simulate the light flow through an optical system. By assuming a lens model to fulfill certain criteria or by accepting a certain amount of approximations, different methods are available. The general approach for ray tracing applications is to express the ray to interface intersection geometrically or analytically.

4.2.1 Ray Transfer Matrix

While accepting *paraxial approximation*, which assumes that all rays have a small angle and distance to the optical axis, an optical system can be approximated by a linear system [Gre04]. The optical system can be split up into its containing optical interfaces, where each interface also represents a linear system. A linear system can be expressed as a matrix. As all components of an optical system can be formulated as matrices, their concatenation results in a single matrix, which describes the whole system. A path through the whole system can be evaluated in constant time by applying the single matrix to its input vector. The transformation of a light ray by an interface can be expressed as a 2x2 *Ray Transfer Matrix* [LE13]. The translation T , reflection L and refraction R ray transfer matrices are defined for simple optical components. Where d is the distance from the optical ray, n_1 and n_2 are the refraction indices, and R the radius of curvature [Hen15].

$$T = \begin{pmatrix} 1 & d \\ 0 & 1 \end{pmatrix} \quad R_{curved} = \begin{pmatrix} 1 & d \\ \frac{n_1 - n_2}{R \cdot n_2} & \frac{n_1}{n_2} \end{pmatrix} \quad L_{curved} = \begin{pmatrix} 1 & 0 \\ -\frac{2}{R} & 1 \end{pmatrix}$$

4.2.2 Polynomials

Polynomials are a generalization of ray transfer matrices (first order approximation) and can be formulated for any desired order. The possible higher order allows to describe also non-linear effects and aberrations. The polynomials express the light propagation analytically through a geometric optical system in a similar fashion as transfer matrices. Generally, the method can be used for any shape as long as the ray intersection has a valid analytical solution. The ray intersection with an optical element can be interpreted as a function. By concatenating the functions (using function output as input for the next function) the ray propagation can be completely described. This concatenation increases the complexity of the resulting polynomial expression, but by truncating terms of a certain degree the net complexity can be kept constant. This allows to approximate an optical system to a certain degree without increasing the complexity [HHH12].

4.3 Camera Models

To allow a higher degree of realism, more advanced camera models have to be used within the rendering process. The physically more reliable camera models are based on optical descriptions to describe the light transformation within a lens system.

4.3.1 Camera Obscura

The theoretical model of the *Camera Obscura* (pinhole camera) can be analytically described and can be used as an artificial camera model. The Camera Obscura model is based on an infinitesimal aperture (pinhole). The aperture is also the center of projection and perfectly maps each point of the scene onto the image plane. Therefore all objects of the scene appear to be in sharp focus [BHK⁺03]. Based on this theoretical model, a real camera can be crafted, but due to the extremely small aperture the resulting images are too dim to observe [HSS97]. This simple camera model is able to produce perspective-projected images without any distortion. Due to its simplicity, it is an often used camera model in computer graphics, but it does not incorporate real lens data to describe the lens system or any camera parameters.

4.3.2 Camera Model Incorporating Optical System Descriptions

Kolb et. al. [KMH95] developed a realistic camera model using the thick lens model to describe a lens system. By using distributed ray-tracing, the parameter of the thick lens model can be approximated quite well. The model also considers aperture stops for a valid depth-of-field and includes exposure time by weighting the ray accordingly for correct irradiance mapping. Vignetting and non-linear geometric transformations like fisheye-lenses can be depicted. The model falls short in terms of wavelength-dependent effects such as chromatic aberrations and internal reflections as well as lens imperfections and diffraction.

Steinert et. al [SDHL11] introduce an advanced path traced rendering approach. The full spectral path tracing algorithm converges well due to efficient Monte Carlo weighting. The approach also includes lens design data to describe the lens system. To simulate various aberrations and wavelength-dependent effects, laws of physical optics are considered by the simulation. Diffraction-based effects, like the star-shaped lens flare effect caused by the aperture shape, can be evaluated for rays within a certain range around the aperture by geometrical diffraction theory. This approach allows to simulate lens flares and chromatic aberrations and can be used as a reference application due to its exact and non-optimized ray propagation formulation.

Hanika et. al. [HD14] present an advanced camera simulation approach based on polynomials. This work extends the previous work of Hullin et. al. [HHH12], which used higher polynomials to describe light propagation through a geometric optical system. The more sophisticated approach by Hanika et. al. [HD14] improves the simulation quality by precisely solving the polynomial approximation of an optical system. The simulation is based on bidirectional path tracing, including multiple importance weight sampling. To find suitable weights, the Jacobians of the polynomials is used. The optimized simulation process allows to produce results close to the ray traced ground truth. This approach supports vignetting, all aberrations and faithfully reproduces point spread functions. Furthermore it allows to change the focus by shifting the sensor. Fresnel's laws are not considered while evaluating ray transmissions, which results in constantly too bright images – but as lens manufacturers intend to craft high transmission lens systems, the results are quite close to reality.

Joo et. al. [JKL⁺16] propose an efficient ray-tracing method to include aspherical lenses. The previously mentioned approaches are all based on spherical lens models. An aspherical lens has non-linear deviations to its spherical base curvature. Modern lens systems include aspherical lenses due to their good convergence of peripheral rays on the focus plane and to reduce aberrations. To optimize such non-linear lenses is not a trivial task that needs high expertise and care in optics. To find the intersection point of a ray with an aspherical lens, Joo et. al. use a *bracketing*-based root-finding method. For efficient use, the initial interval is evaluated by approximating the aspherical lens by a tight proxy geometry. The simple sample-based ray intersection method can be used for aspherical lenses as well as for more general lens forms. Fresnel diffraction is approximated by a *fraction Fourier Transform* to generate a realistic *bokeh* effect.

4.4 Lens Flare Simulation and Rendering

Lens flare simulations strongly depend on the underlying optical system model. An optical system solely described by thin or thick lens model is not capable of simulating lens flare or complex optical effects (aberrations of higher order). Complex light paths within the system cannot be described accordingly, therefore this description approximations are too crude. Nonetheless, these lens approximation models are still useful to reduce the complexity of the simulation process in various simulation approaches. Depending on the

application's intentions, different kinds of optical system models are used.

Other applications are primary concerned with the correctness of the simulation process and therefore tend to use very precise lens systems descriptions. These applications focus on quality evaluations of a lens system and not mainly on rendering (Chapter 4.4.1). For other applications, the rendering of lens flare is of high importance. Depending on the need for realism, different approaches have been developed. A possible way to render lens flare is based on artistically inspired heuristics, which are not based on any real-world data. For these applications, the correct occurrence and shape of lens flare is negligible, because they only want to mimic the lens flare effect in a pretty and fast way (Chapter 4.4.2). In contrast to artistic lens flare renderings, there are also renderings inspired by a high degree of realism. The visual appearance of the lens flare elements as well as their correct position is a result of a physically-inspired simulation (Chapter 4.4.3).

4.4.1 Lens Flare Simulations without Rendering

An high degree of physical correctness in the simulation of a lens system is of great importance in the field of optical lens design. Based on physically accurate simulation results, the optical lens designer is able to optimize the lens system in a more efficient way. It allows to fulfill various criteria, like performance requirements and cost constraints. High-end designing tools for optical systems like *ZEMAX* or *Code V* include various optical effects, like wave-optical effects caused by diffraction and interferences. The main focus of such tools is to provide correct simulation results, but they are not intended to generate real-time renderings and only provide a very general solution [Toc07]. Figure (4.4) shows an example how *ZEMAX* visualizes a lens system and its scattered light in a spot diagram.

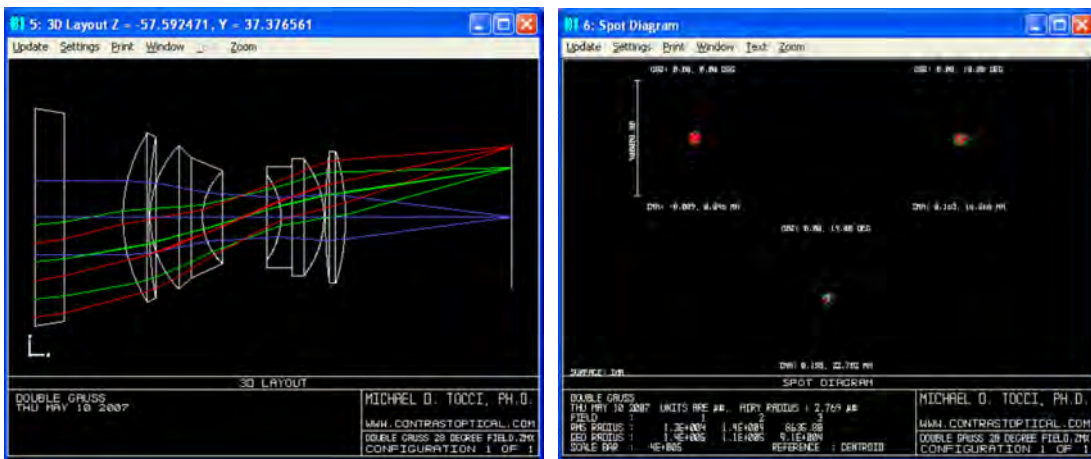


Figure 4.4: Lens description and simulation [Toc07]

4.4.2 Artificial Lens Flare Rendering

Artificially created lens flare renderings are not based on lens flare simulation. The hereby used lens flare sprites are artificially generated and do not claim to be physically correct. The main interest of lens flare sprites are to enrich the scene with a pretty lens flare effect and to indicate the presence of bright light within a scene. In the creation process of these sprites, artists try to mimic the shapes and colors of lens flares. The artist are not strictly bound by physical laws and therefore certain details can be simply skipped or can be drawn emphasized or even exaggerated to depict some pretty aspects of real lens flares.

Kilgard et. al. [Kil00] suggest to blend the pre-computed lens flare textures into the frame-buffer as a post-processing effect common in real-time applications. The colorful lens flare effect is achieved by blending different red, green and blue textures together. The lens flare sprites are arranged along a line through the image center following an ad hoc displacement function. Star-like textures are directly blended over the light source position to achieve a pretty, colorful lens flare effect. Figure 4.5 shows basic gray-scale textures and a result of this approach.

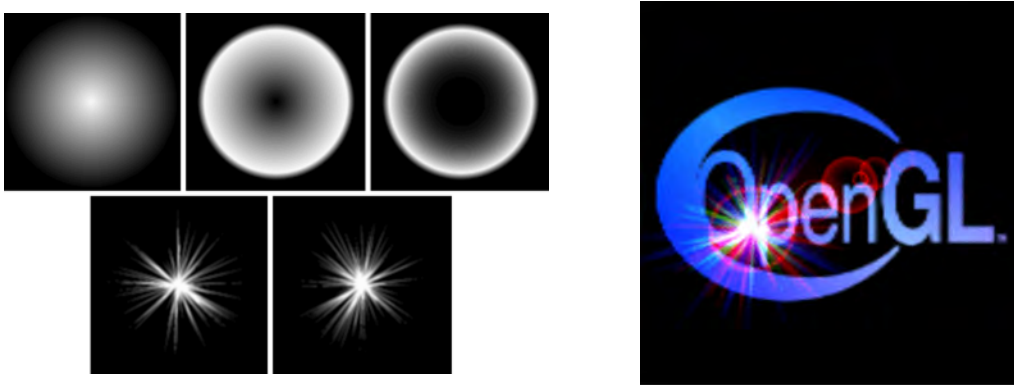


Figure 4.5: Gray-scale lens flare sprites and resulting lens flare effect [Toc07]

For a more realistic behavior of the lens flare effect, camera movement has to be taken into account to manipulate the lens flare sprites. King [Kin00] achieves to include camera movement by changing the opacity and size of the lens flare sprites according to the distance of the projected light to the image center. While the light source is closer to the image center, the flare elements become larger and more opaque. On the contrary, while moving the light source farther away from the projection center, elements shrink and become more transparent.

To control brightness variation of a sprite based lens flare effect, Maughan [Mau01] suggests to scale the intensity of the effect by the amount of visible light pixels in the rendered image. In case of a non-occluded light source, the lens flare effect becomes quite prominent. While moving the light source behind occluding scene elements, the amount of visible light pixels decreases and the lens flare effect scales down. Due to performance



Figure 4.6: Lens flare sprites used in *Serious Sam 2* [Sek04]

issues Sekulic [Sek04] recommends to use the occlusion queries features of common Graphical Processing Unit (GPU)-Application Programming Interfaces (APIs) to speed up the evaluation of visible light source pixels. Figure 4.6 shows how a sprite-based lens flare effect is used in a computer game.

Another way to describe lens flare elements is demonstrated by Alspach [Als09], where the shapes of the lens flare elements are described by a set of vector shapes. The vector-based representation is a combination of different basic vector elements, such as *halos*, *rings* and *rays*. The vector shapes within a group maintain their association to each other while manipulating the whole group by user inputs. Common editing operation such as moving and resizing can be applied to the whole vector group.

4.4.3 Physically-Based Lens Flare Simulation and Rendering

Realistic lens flare rendering is coupled with an optical system simulation. Depending on the rendering quality, more or less approximations are acceptable. Simulation and rendering time can be a critical criteria for real-time applications or while restarting the whole simulation due to lens system changes (zooming, aperture).

Chaumond [Cha07] implements lens flare rendering based on simple path tracing. The resulting lens flare images are very noisy, because only a few ray are actually contributing to the lens flare effect. To improve the convergence of lens flare artifacts, the path tracing algorithm starts from the scene light towards the camera. This results in a nearly dark scene, but better convergence of the ghosts. This approach already considers the aperture shape, but due to poor convergence, the aperture shape does not have a visible impact

in the resulting rendering. Lens flare coating as well as wave effects are not included.

Kesmirian [Kes08] uses photon-mapping to render lens flare. The used camera model is based on Constructive Solid Geometry (CSG) primitives. This approach is able to render ghosts, but it converges really slowly. Lens flare coatings are not included, therefore the color of the ghosts cannot be determined. Wave-optical effects are also not considered in this approach. Figure 4.7 shows a synthetic flare generated with this method.

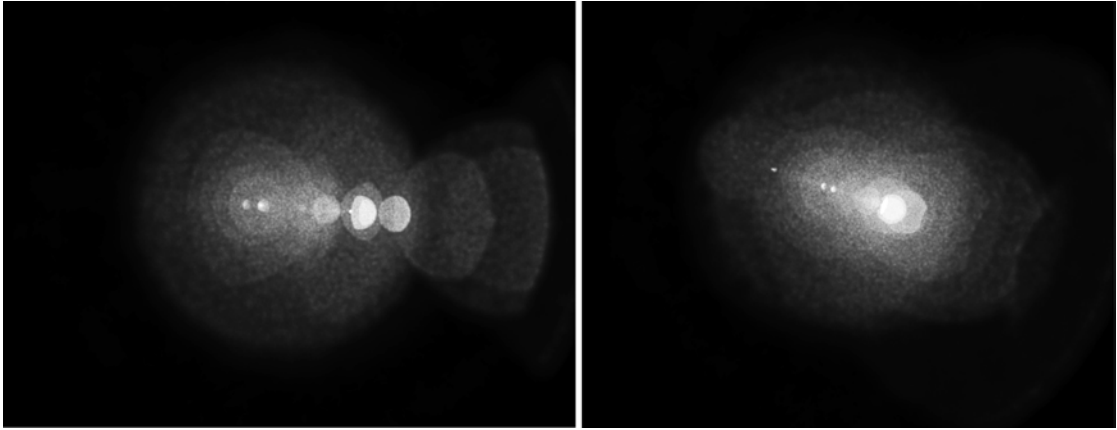


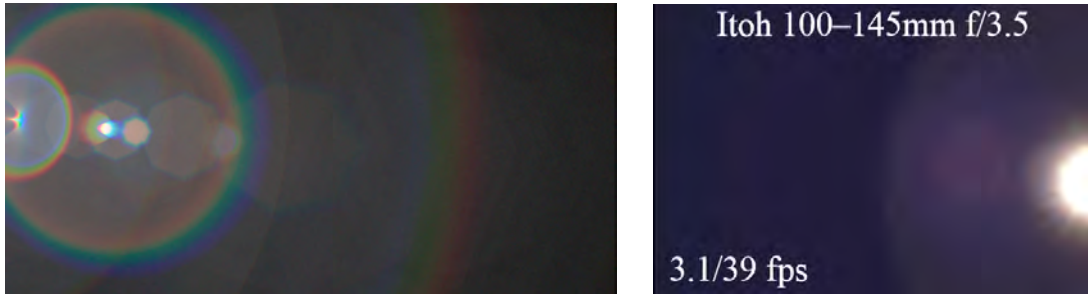
Figure 4.7: Synthetic lens flare simulation (left: 143min | right: 149min) [Kes08]

Steinert et. al [SDHL11] present a full spectral camera model based on lens design data (Chapter 4.3.2). The lens flare ghosts are simulated by introducing an optimized path generation for non-image-forming paths. The colors of the ghosts are mapped according to the unknown, but well approximated lens coatings. By incorporating geometrical theory of diffraction, the star shape lens flare can also be simulated (Chapter 3.2). To simulate this effect efficiently, only rays within a certain range from the aperture edges are considered. Due to the complex simulation process, simulation time is beyond the requirements of real-time applications. Figure 4.8 shows different lens flare elements (aperture diffraction and ghosts) and a scene with the combination of both effects.



Figure 4.8: From left to right: diffraction simulation (30min) | ghosts simulation (130min) | armadillo scene with lens flare overlays [SDHL11]

Hullin et. al. [HHH12] present an improved ray tracing approach based on a polynomial representation of the lens system (Chapter 4.2.2). Lens flare can be efficiently simulated by expressing lens flare forming paths as polynomial systems. The resulting systems are used to pre-generate machine code to allow an efficient execution. The entrance pupil is randomly sampled to map incoming light rays onto the sensor without generating visible patterns or holes. For each ray, an additional polynomial is evaluated to find the position where it passes the aperture. This allows to discard aperture blocked rays from the final solution and to incorporate the aperture shape for the ghosts simulation. For more complex light paths, which pass the aperture several times, additional polynomials evaluations are necessary to verify a ray is not blocked by the aperture. For efficient computation, this approach assumes lenses to be rotationally symmetrical to reduce overall complexity. Code generation for the polynomial systems is quite time consuming (up to 8 minutes), but afterwards, the rendering can be done quite fast (up to 4 seconds). Aberrations can be simulated by using polynomials of degree 3 or higher, but diffraction cannot be simulated by this approach. To reduce sampling noise, the *Jacobian* of the polynomial system can be used to determine appropriate orientations and kernel size for a *Gaussian filter*. This approach is more efficient than the one proposed by Steinert et. al. [SDHL11], but also coarser due to its approximations. The method reaches interactive performance, but is not suitable for a real-time application. Hanika et. al. [HD14] improve the precision for this polynomial approach, but have not further investigated the visual difference of their rendering compared to this approach. Figure 4.9a shows the lens flare rendering of the *Itoh* zoom lens in comparison to Hullin et. al. [HESL11].



(a) Polynomial based rendering by Hullin et. al. [HHH12] (b) Texture based rendering by Hullin et. al. [HESL11]

Figure 4.9: Itoh zoom lens

In *Physically-Based Real-Time Lens Flare Rendering* by Hullin et. al [HESL11], a ray tracing-based simulation approach for lens flare rendering is presented. The camera model, based on real lens design data, is efficiently traced by *ray bundling*. Ray bundling reduces the needed amount of rays to a sparse set by grouping neighboring rays to a single ray. To accelerate the simulation process, the sparse set of incoming light rays are restricted to a rectangular region, depending on the entrance aperture. The region's size is chosen to include all rays that might reach the sensor. Each ray records its traversal through the system, and when finally reaching the sensor, a grid with neighboring rays

can be defined. The ray grid is used to interpolate missing rays in between without the need to actually simulating them.

The rendering is based on realistic textures, which incorporate the aperture shape and approximate the aperture diffraction by the *Fraunhofer pattern* as well as the surrounding ring pattern of the ghost by the *Fresnel approximation*. The texture generation is performed in a pre-processing step. For complex lens systems, the simulation is quite time consuming (up to 20min). The rendering time of the flare is quite low (near real-time frame rates can be achieved) due to pre-generated lens flare textures. The color of ghosts is considered within the simulation process by approximating anti-reflecting coatings. Lens imperfections are artificially generated by slightly offsetting the ghosts from the light-to-image-center ray. The approach gives the user the possibility to manipulate lens flare textures (color) to fulfill artistic requirements, and to adjust the simulation trade-off between quality and speed by skipping certain flares. Figure 4.10 (bottom) shows a complex lens flare generated by this approach.

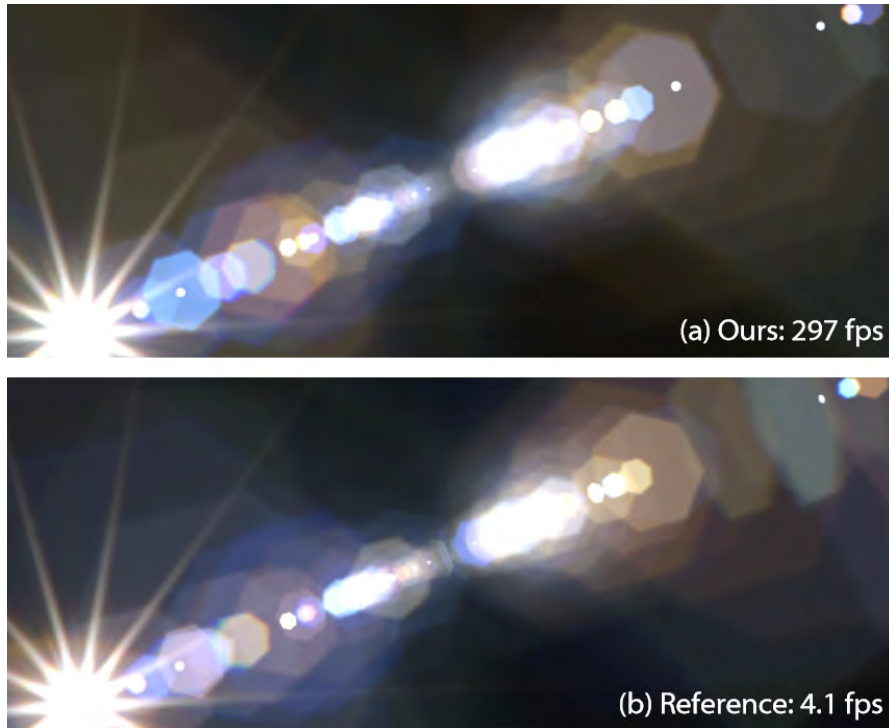


Figure 4.10: Lens flare results from Lee et. al.[LE13] (top) and Hullin et. al. [HESL11] (bottom)

Lee et. al. [LE13] improve the work of Hullin et. al [HESL11] regarding simulation speed, but with coarser quality (Figure 4.10). The lens simulation is based on a first-order approximation of the ray transfer within an optical system. This approximation can be described by ray transfer matrices (Chapter 4.2.1). To find a possible ghost path (*flare*

matrix), the complete light propagation path has to include an even number of reflection matrices. The flare matrix is used to directly project a quad from the entrance pupil onto the sensor in constant time. The projected quad holds all transformations for a specific ghost and can be re-used as a ghost sprite. As the shape of a ghost strongly depends on the aperture, the aperture shape is integrated in the propagation path as well as the entrance pupil. The flare matrix is split up into two parts: One describes the path before the aperture, the other one the path from the aperture to the sensor. In between, the pre-sampled aperture texture is used to block rays accordingly. To reduce unlikely reflection paths where the aperture is crossed multiple times, the reflection is assumed to occur either before or after the aperture. The intensity of the ghost is scaled by the projected quad and its color is sampled by one ray in the ghost center (one simulation run for each color channel separately). The direct light is approximated as in Hullin et. al. [HESL11]. Due to the first-order approximation (minimal version of polynomial approximations), nonlinear deformation, aberrations and topological changes cannot be expressed correctly, but generally only a few lens flare elements are affected by non-linearity. The overall processing time depends on the lens system's complexity, but is usable for real-time applications (261Hz to 1615Hz). The introduced ray propagation model can be used for sprite based lens flare rendering (Chapter 4.4.2) to describe a realistic displacement function. For acceleration purposes, only paths of two reflections are simulated (due to the high energy loss per reflection (Chapter 3.2.1), and ghosts with low intensities are culled. Hennessy [Hen15] implemented this approach and suggests to scale the ghost's intensity by the system's effective aperture and to sample a ghost's color by a random angle for each color channel to reduce the chance of complete internal reflections.

Joo et. al. [JKL⁺16] present a method to simulate lens imperfection caused during the manufacturing process. Lens imperfections can arise by grinding or polishing in the crafting process, or by dirt in the lens material or on the lens surface. Even a few nanometers of deviation can result in visible artifacts. An ideal parametric model does not include such extrinsic imperfections, and their incorporation is not trivial. Joo et. al. simulate the swinging movements of the polishing and grinding tools to generate an imperfection texture. The imperfection texture can be used in the simulation process as an offset look-up to bias the refraction directions of light paths. This approach is capable of dealing with aspherical lenses, which are more prone to crafting imperfections than spherical lenses. The simulation process is able to reproduce lens flare, but is very costly due to a large number of lens samples to reduce noise. To allow real-time lens flare rendering, this method can be used to generate lens flare textures which are then applied in a sprite based rendering approach like proposed by Lee et. al. [LE13] (Figure 4.11).

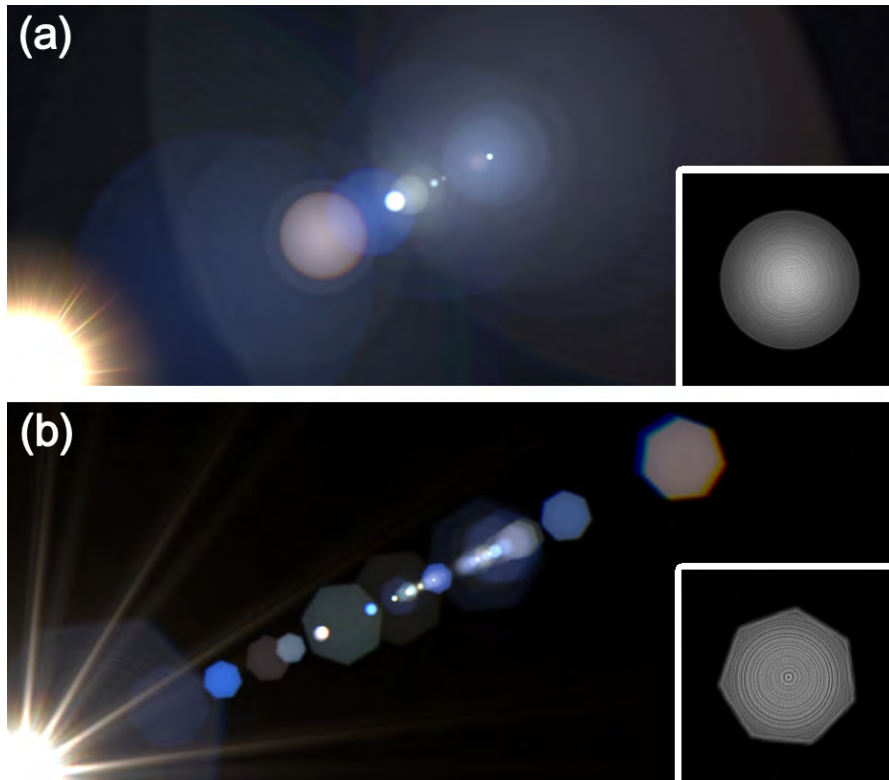


Figure 4.11: Example of imperfection textures from Joo et. al. [JKL⁺16] applied onto Lee et. al. [LE13] lens flare renderings



Data Acquisition

The aim of this work is to retrieve a possible workflow to predict and render lens flare, especially ghosts, for any lens system without any specific information of the lens system's internal structure (Chapter 2.2). The only necessary parameters for this approach are the generally available camera parameters: focal length and aperture stop. The first building block of this workflow is the data acquisition stage.

The camera can be interpreted as a black box, and its output images contain all transformations caused by the lens system. Recent work in the field of physically-based lens flare rendering show, that the lens system must be described as precisely as possible to generate a physically accurate lens flare effect (Chapter 4.4.3). Unfortunately, the lens system specifications are hardly available. Additionally, the specifications are not complete. Especially the composition of the anti-reflection coatings, which are essential for the appearance and occurrence of lens flare, are kept secret. The constellation of camera to light source is essential for the ghosts positions and their appearance (Chapter 3.2.3). To sample the behavior of lens flare, the camera to light constellation has to be changed in discrete steps (rotating the camera). While using a sufficient small step size, the ghost's positions do not vary too much between neighboring samples. This allows to reconstruct the behavior of each ghost over the sampled range.

5.1 Sampling A Lens Flare

While capturing a bright point light source within a dark neutral environment, the chance to depict a clear lens flare effect in the output-image increases. A ghost's intensity is related to the amount of energy, which reaches the sensor via the non-image forming paths (Chapter 3.2.1). Each reflection on the path reduces the carried energy. To produce a valid data sample, a sufficiently bright light source is needed, to also depict ghosts with low intensity. A longer exposure time can also be used to compensate a less bright light source, but increases the acquisition time. On the other side, a too long exposure results

in oversaturated areas within an image, which do not hold any usable information. To ensure reliable data, raw-data as captured by the sensor is used in the further workflow. Therefore, the images are not tone-mapped or compressed. To increase the possible range of intensities, multiple images with different exposure times (0.25s/1s/4s) are sampled and combined to a High Dynamic Range (HDR) image (performed by *Photoshop 5 - HDR-Tool*). The HDR images are converted from the camera-specific RAW-format to the common open-source format *OpenEXR* from Industrial Light and Magic [LM14].

5.2 Assumptions

For an effective data acquisition, some assumption have to be made in advance to limit and reduce the amount of required samples, but still ensure to capture all essential information of a lens flare.

Symmetry: The lens system is assumed to be radial symmetrical to its optical axis. Therefore it is enough to rotate the camera only around one axis to sample the complete lens flare behavior (Chapter 3.2.3). The only requirement is that the light source center is crossed by the camera's optical axis. This reduces the amount of possible samples drastically.

Aperture: To capture as many ghosts as possible, the aperture is fully opened, resulting in a perfect circle aperture shape (it is hidden behind the aperture housing). Therefore, the ghost shapes are not so complex and can be better approximated in the next workflow stages.

Light Source: The used light source has to simulate a point light. The incoming rays at the entrance pupil are therefore assumed to be parallel. To fulfill this assumption, the used light is chosen to have a narrow output angle and small exit pupil (approximately 10mm).

Stray Light: To capture stray-light caused ghosts, the camera's optimized lens hood is not applied.

Imperfections: To avoid extrinsic imperfections, like dust on the entrance pupil or on the sensor, both camera elements are carefully cleaned before sampling lens flare.

5.3 Prototype Setup

The prototype consists of a camera and a bright light source, each placed on a tripod in such a manner, that the optical axis of the camera is on the same height as the light source. The camera is rotated around its *no-parallax point* to avoid parallax shifts over the range of captured images [Rik06]. The fixed distance of the camera to the light source is chosen carefully to ensure that the camera's entrance pupil is fully, but as tightly as

possible, illuminated by the light source. To generate a valid range sampling, the optical axis has to cross the light source center. To ensure this position, our acquisition setup starts with exactly that critical constellation. The light centered camera start position allows to rotate the camera in any direction, while still recording all important ghost movements. The rotation sampling is performed along the horizon, because the camera format is generally wider in the horizontal dimension, than in the vertical dimension.

5.3.1 Automatic Sampling

In our first attempts, we tried to manually rotate the camera in small rotation steps. By manually rotating the camera (even using the tripod's angle marks and physical locks) the sampling steps cannot be assured to be evenly spaced. To ensure equal rotation steps, we used a programmable motorized panorama rotation head from *Syrp* [Syr16]. The panorama head offers an efficient and dense sampling rate (0.2°), which should be sufficient enough to capture all necessary ghost movements.

5.3.2 Test Light Source and Environment

To find an appropriate test light source is not trivial. Regular light bulbs are too large to simulate a point light in a reasonable distance. Unfortunately, the tested flashlights are too weak to trigger lens flare in reasonable time (around 30 seconds or more as depicted in Figure 5.1). Our own crafted test light based on a single Light-Emitting Diode (LED) is more suitable, but unhandy to mount on the tripod (still 10 seconds, but better as the flashlight attempt as Figure 5.2 shows). Finally, the quest of finding a more powerful and small light source lead us to a medical light system, which is designed for endoscopy examination (*Olympus CLV-S* with a OES Xenon Light Source). The xenon light source is extremely bright and allows to capture visible lens flare within seconds (fast and good response within 1 second as Figure 5.3 shows).

The tripods are clearly visible in the first test attempts, because our setup room is not comparable to a perfect black box. By covering the surrounding and tripods with a black



Figure 5.1: Flashlight test setup and result (30 seconds)

5. DATA ACQUISITION

curtain, the background noise can be drastically reduced (Figure 5.4). To create our final data set we used a *Canon EOS 5D Mark II* (1 second | f/1.4 | 24mm | ISO 100) and the endoscopy light system (Olympus CLV-S). The setup's complexity is intentionally kept simple and general to be reusable for any mountable camera model or lens system.



Figure 5.2: LED test setup and result (10 seconds)



Figure 5.3: Medical light test setup and result (1 second)



Figure 5.4: Final setup with curtain and automatic panorama head

Ghost Rendering Primitive - Lens Flare Model

The captured samples of the *Data Acquisition Stage* (Chapter 5) show the physically correct mapping of lens flares by a specific lens system. Unfortunately, the samples do not contain any description about their content. From the capturing stage, only the position of the light source within the image is known (due to the capturing angle), but the ghosts' positions are completely unknown. For further use in our workflow, the position and shape of the ghosts have to be extracted from the sampled image data. We use a simple model for ghosts consisting of shape, position and color parameters. To best fit the elaborated ghost model to the sampled data (in the next stage of the workflow, see Chapter 7), the model is capable to generate a visual representation. The visual representation is directly implemented on the GPU to ensure a fast rendering, which is important for an efficient optimization (Chapter 7) and real-time visualization (Chapter 8).

6.1 Analyzing the Sampled Ghosts

According to the stated assumptions (Chapter 5.2), the ghosts in the captured data are more or less circular or elliptical. Additionally, the images share a mirror-axis along the captured horizon, except for some small variations (imperfections). Generally, in case of an odd sided n -polygon-shaped aperture, the mirror property is not given. The position of the light source's height varies a little bit over the range. This can be lead back to a non-perfectly balanced mass center of the camera while rotating. The ghosts seem to be split-able into a left and right side while surveying the shape. This visual clue may originate from the horizontal sampling and clipping by the entrance pupil and aperture. The left and right side share a common height with a smooth transition. The color and intensity of each ghost varies with the distance to the visible border of the ghost shape,

but the transition is generally quite smooth. Each ghost appears to have a dominant color with slight variations. Usually, the border holds the brightest color and intensity values and starts to fade out to both sides. Torus-like shapes are generated, while the intensity of the border fades out to black on each side. In some cases the inner area does not fade out completely, resulting in various filled ghost shapes.

6.2 Requirements For the Rendering Primitives

The required parameters for a ghost rendering primitive are as follows:

Shape: Due to horizontal sampling, the shape of the ghost can be clipped horizontally by the entrance pupil. Therefore, the left and right side have to be treated differently. Each side has to offer enough flexibility to approximate the aperture shape (according to our assumptions, a circle). The position of the ghost is set by the center coordinates in image space.

Color: The color or intensity depends on the distance to the visible ghost border, while the transfer function can vary from ghost to ghost.

6.3 Creating the Model

The general position of a ghost can be expressed by a position vector (M_x and M_y). Therefore, this previously mentioned requirement can easily be included into a model. The intensity or color is mapped by a minimal distance function to the ghost's border. The border can become quite complex, especially as the left and right side can differ from each other while still sharing a smooth transition. According to our assumption, the aperture shape is a perfect circle, but the resulting ghost shapes are more complex than a simple circle. To be able to describe the whole ghost shape, either a rather complex curve is required, or multiple circular segments have to be smoothly combined.

6.3.1 Shape

The *Bézier curve* can satisfy the previously mentioned requirements, as it allows to describe a huge variation of circular shapes and their combination. The Bézier curve depends on a control-polygon. In contrast to other curve models like splines, Bézier curves start and end in control points, which allows to append multiple Bézier curves in an easy fashion. The number of control points, used for weighting the curve describe the degree of the Bézier curve. A *cubic Bézier curve* is defined by four control-points: a start point P_0 , two additional control points P_1 and P_2 , which mainly influence the slope, and an end point P_3 . The weighting for a point B on a cubic Bézier curve is formulated by Equation 6.1, while the parameter t within its range $[0, 1]$ describes all points between the start and end point (Figure 6.1).

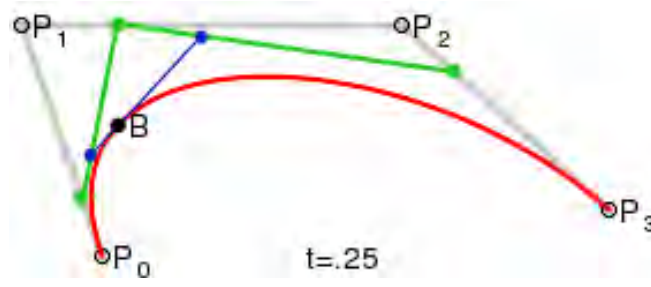


Figure 6.1: Cubic Bézier curve - point construction at $t = 0.25$.¹

$$C(t) = \sum_{i=0}^3 \binom{3}{i} t^i (1-t)^{3-i} P_i \quad t \in [0, 1] \quad (6.1)$$

While combining two Bézier curves it is possible to generate a very smooth transition by placing the control points appropriately. To ensure that the curves have a C^1 continuity at the assembly point, the closest control point of each curve and the assembly point have to share a common line. To ensure G^1 continuity the control points have to be placed evenly spaced from the assembly point on the line. To reduce the overall complexity of the ghost shape, only cubic Bézier curves are used within our model. Cubic Bézier curves are flexible enough to describe various forms and can be simply spit up into multiple segments by inserting additional control points into the previous control polygon. While various shapes can be formed by cubic Bézier curves, a perfect circle can only be faithfully approximated by appending four cubic Bézier curves [2]. Four curve segments are also the lower limit to be able to represent the encountered ghosts shapes, therefore this restriction does not increase the complexity of our model.

In case of a mirror-able aperture shape (horizontal aligned) the upper or lower part of the ghost can simply be mirrored. The left L and right R side of the ghost have to be individually changeable. The height h of the ghost is defined by the common assembly points of the left and right ghost side. The curvature of each side can be separately adjusted by the control points C_l and C_r . To reduce the parameter the control points are restricted to be shifted only axis-aligned to support C^1 continuity. The model is capable to generate simple curve segments and even a line. Figure 6.2 visualizes the concept of the shape model. The green points represent the control points for C_r and the red points for C_l .

To represent any n polygonal aperture shape (blade edges are not restricted to be flat), more Bézier curve elements could be used to describe a more complex ghost shape.

¹Source: By Philip Tregoning (Constructed with ImageMagick) [Public domain], via Wikimedia Commons

²<http://spencermortensen.com/articles/bezier-circle/>

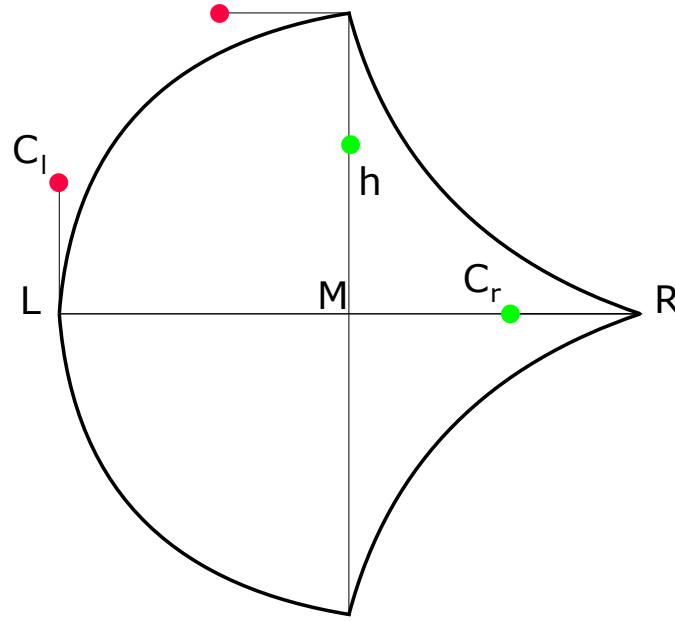


Figure 6.2: Shape model for a lens flare element

6.3.2 Intensity

The intensity or color of a point depends on its signed distance to the ghost's shape. The shape of a ghost is described in our model by multiple appended cubic Bézier curves (Chapter 6.3.1).

The minimal distance from a given point to a cubic Bézier curve cannot be directly solved analytically, but different heuristics can be used to retrieve the point B on the cubic Bézier curve $C(t)$ (Equation 6.1), which has the minimal distance $d(t) = |C(t) - Q|$ to the query point Q .

A simple approach is to loop over the curve in discrete steps n and calculate in each iteration i the point on the curve $C_i = C(t + i * \frac{1}{n})$. For each sampled point the distance is calculated $d_i = |C_i - Q|$, and the smallest encountered distance is assumed to be the minimal distance. This method strongly depends on the sample density and can only approximate the minimal distance very roughly and by coincidence. Only a really dense sampling rate increases the chance to find the actual minimal distance or at least a quite good approximation, but this approach is highly inefficient.

To reduce the number of samples and to further increase the quality of the minimal distance approximation a *divide and conquer* based approach, like *binary search*, is used instead of brute force sampling. Binary search iteratively splits the parameter range into two equal parts and evaluates the distance separately for each mid-point. The range, which holds the smaller value at its mid-point, is used for the next iteration. In each iteration the possible range of t (starting with $[0, 1]$) is halved ($range = 0.5^i$).

This reduces the needed iterations from $O(n)$ to $O(\log n)$ and allows to approximate the minimal distance of our restricted cubic Bézier curve efficiently. The restriction of our shape model allows to further accelerate the splitting process by reducing the possible range around the unknown parameter t more efficiently. The split ratio between the projected query point onto the connection line of start and end point, allows to reduce the range more efficiently than by simply splitting it into halves.

Another more complex, but exact and more general method to find the closest point B on the cubic Bézier curve $C(t)$ can be stated by the following equation 6.2.

$$B = C(t) \text{ where } t = \operatorname{argmin}_{t'} |C(t') - Q| \quad (6.2)$$

The absolute distance d of a query point Q to the points on the Bézier curve $C(t)$ depends on t and is defined as $d(t) = |C(t) - Q|$. Finding the extremes of the first derivative of the distance function $\frac{\partial d(t)}{\partial t}$ allows to reduce the possible solution to a polynomial root finding problem. The derivative is a fifth-order polynomial in t , therefore up to five possible solution exist. The solutions have to be first filtered by excluding complex solutions and solutions which are outside of the defined range $[0, 1]$. The remaining solutions values for t are used to evaluate the absolute distances for each solution separately. The calculated distances are compared to each other to find the exact minimal distance.

The precision of the less complex iterative binary search based approach depends on the amount of iterations. To find the sufficient amount of iterations, we compared the iterative approach to the analytic approach. After 12 iterations the difference of the binary search method is negligible to the analytic method. The lightweight iterative method is suitable to be executed directly on the GPU, while the complex analytical approach can be more efficiently evaluated on the Central Processing Unit (CPU). Therefore we use the iterative method to ensure efficient minimal distance calculation per pixel.

The **sign** of the distance indicates whether the query point lays inside or outside of the curve. The sign of the distance can be evaluated by basic 2D-trigonometry (Equation 6.3), where B is the closest Bézier-point, B' its derivative and Q the query point.

$$\operatorname{rot}_{\text{tangent}} = \operatorname{norm} \begin{pmatrix} -B'_y \\ B'_x \end{pmatrix} \quad (6.3)$$

$$\operatorname{sign} = \operatorname{sign}(\operatorname{rot}_{\text{tangent}} \cdot (B - Q))$$

After defining the signed distance for each point its intensity or color value is applied by a **transfer function**. Color can be interpreted as combination of multiple intensity channels (RGB-model: red, green and blue channel), therefore the intensity transfer function can be applied to each channel separately. A transfer function maps a distance value to an intensity value by an arbitrary function. According to our analysis of the occurring lens flare elements in the acquisition stage, the brightest color or highest

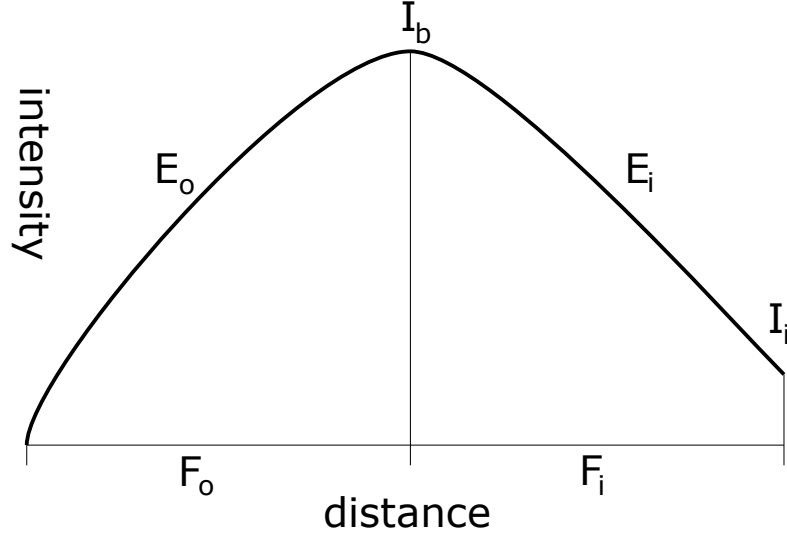


Figure 6.3: Transfer function for a lens flare element

intensity values rest upon the ghost border and fade out to a constant value or completely. Equation 6.4 describes the intensity transfer function of our model. Value I_b represents the intensity value at the border and I_i the intensity value for the inner area and d the signed distance value. The fade-out range is limited by F_i the inner falloff range and respectively F_o for the outer. The fade-out slope can be adjusted by the exponents E_i for the inner fade-out function and respectively E_o for the outer. *Lerp* interpolates the border intensity value I_b to the desired intensity (I_i or 0.0) value by the scale factor S . Figure 6.3 visualizes the transfer function.

$$S = \begin{cases} 1.0 & \text{if } d = 0 \\ \frac{|d|}{F_o} & \text{if } d < 0 \\ \frac{|d|}{F_i} & \text{if } d > 0 \end{cases} \quad (6.4)$$

$$I = \begin{cases} \text{Lerp} (I_b, 0.0, \text{Clamp} (0.0, 1.0, S^{E_o})) & \text{if } d < 0 \\ \text{Lerp} (I_b, I_i, \text{Clamp} (0.0, 1.0, S^{E_i})) & \text{else} \end{cases}$$

The overall minimal signed distance from a given point to all curves can be found by evaluating the minimal signed distance for each curve respectively and choose the smallest distance. Figure 6.4 depicts two Bézier curves and the closest distance to a query point.

This model ensures that positive distance values (points laying inside of the ghost) are always affected by the intensity transfer function, while the outer-side of the ghost is fading out completely. Negative distance values farther away than F_o are not anymore affected by the transfer function. Without any restrictions, the distance evaluation and the coupled intensity transfer function has to be calculated for all pixels of the image.

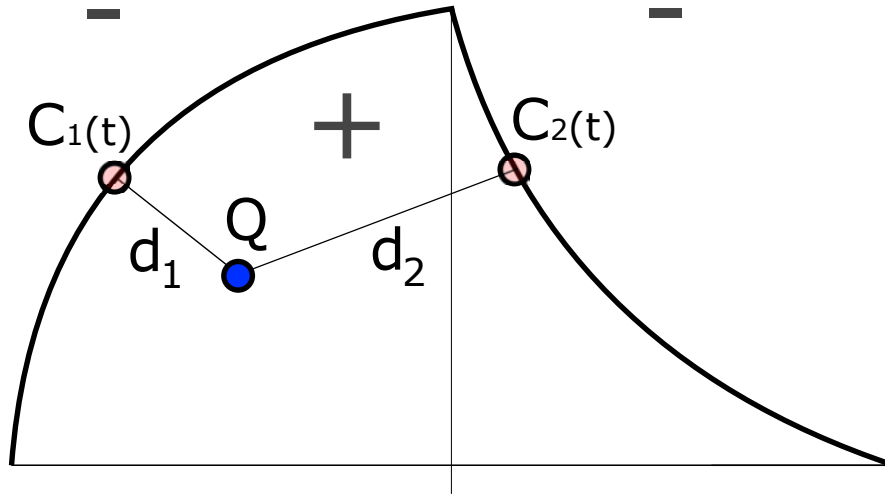


Figure 6.4: Minimal distances from a query point to the Bézier curves

This is extremely inefficient. By defining a tight bounding box around the ghost, which holds all possible affected points by the transfer function, the amount of tested pixels can be drastically reduced.

This shading model is capable to approximate a lot of observed details from the sampled ghosts, but can be simply extended to describe additional details. To render the ghost primitives efficiently, the whole shape generation and shading is directly performed on the GPU. Especially our advanced binary-search based distance calculations benefits extremely from the efficient parallelism in the fragment stage. As each ghost solely describes light, multiple ghosts can be additively blended into a single image.

In regard to the upcoming stages of the workflow (Chapter 7) we tried to create a powerful and flexible model to describe various ghosts, but also limit the model's parameter set to a minimum. To ensure a fast and efficient execution, various optimizations are included in the GPU based rendering.

Optimization - Finding the Ghost

The previously presented lens flare rendering primitives (Chapter 6) are able to generate an artificial representation of a lens flare element based on the underlying model. By adjusting the model's parameters, a lens flare element of the acquisition stage (Chapter 5) can be quite well approximated or even completely described. The problem is to find the optimal parameter set for the lens flare model. Some special details are maybe not depict-able by the model, and therefore the best possible approximation has to be found.

The difference between the actual, captured image and our rendered lens flare representation has to be as small as possible. To somehow express this difference and its cost, a so called *cost function* is used (Chapter 7.1). While reducing this cost function, the difference decreases, and as a result the images become more alike. This reduction process is a typical optimization problem, where the global minimum of the cost function represents the best solution. To avoid a wrong behavior, the cost function has to describe the optimal solution precisely (Chapter 7.1.2). Within the optimization process, the cost function is evaluated very often, and a fast calculation method is needed to find the solution in reasonable time. The data input for the cost function can be generated in short time (lens flare rendering primitives) or can be simply accessed (sample of the acquisition stage). As the cost function deals with images, it is obvious to use a GPU to benefit from its optimized image processing and image representations (Chapter 7.1.3).

The simplest approach to solve this problem is a *brute force* based approach, where all possible parameter constellations are evaluated and compared – but this is highly inefficient. To reduce the amount of samples, several optimization strategies are developed, strongly depending on the cost function's properties. Iterative search techniques are quite common for complex cost functions: After each iteration, the output is compared to the previous state, and according to the improvement potential, the parameters for the next iterations are chosen (Chapter 7.2). The optimization strategies vary on how the parameter sets are chosen and updated. Depending on the optimization problem and the search strategy, the correct solution can be found faster, or can be better approximated.

The optimization strategy uses the cost function to find the optimal parameter set to describe a lens flare element in an image. To support the optimization process to find the best solution, the initial values for the model parameter are crucial. After obtaining the optimal parameter set for a sample of the acquisition stage, the just evaluated parameter sets can be reused as initial values for the neighboring samples (Chapter 7.3). Because the user is included in setting the initial parameters, and parameters for neighboring samples are extrapolated from the first result, this is a semi-automatic approach.

7.1 Cost Function

To solve an optimization problem, it is important to know what impact changes cause. To distinguish between good and bad decisions, a numeric indicator is necessary. The cost functions maps the improvable space into a numeric value. A reduction of the cost value indicates a good decision, while unpromising changes increase the cost. In our case, the cost function represents the similarity between the rendered image and the captured reference image. The similarity can also be expressed as the difference of two images. The digital representation of images are raster images, consisting of pixels. To calculate the difference between two images, the image can be compared pixel wise. If two pixels share the same value, their difference is zero, and this pixel does not contribute any additional cost. The simplest difference cost function can be calculated according to equation 7.1, where N is the number of a all pixels and X and Y are the pixels of the two input images.

$$cost_{abs} = \frac{1}{n} \sum_{i=0}^n |X_i - Y_i| \quad (7.1)$$

While using this basic difference function, the convergence behavior of the simulation is quite slow. To boost the impact of changes, the square of the differences is used for a non-linear mapping (Equation 7.2).

$$cost_{MSE} = \frac{1}{n} \sum_{i=0}^n (X_i - Y_i)^2 \quad (7.2)$$

The Mean Squared Error (MSE) improves the convergence of the cost function by weighting small difference less than larger ones. This metric is quite stable and successful for artificially generated tests (comparison of two rendered images), but fails on real-world data (comparison of rendered and captured image).

7.1.1 Encountered Problems

While solely using equation 7.2, the cost function is not able to distinguish between **background** and lens flare elements. Background noise as well as in- or out-fading lens flare elements state the biggest problem for the basic cost function. Therefore, the

current cost function (Equation 7.2) is not good enough. Additional features have to be included, or other pre-processing steps have to be applied (Chapter 7.1.2).

Color can also be a source of error, as every color space is based on multiple, dependent channels. To generate a certain color, each channel has to be set to a certain value. To describe a similar color, several channel values have to be changed individually. For example, in the RGB-color space, the euclidean distance between two color is not really similar or comparable to the human perception. We also investigated the HSV-color space, because it seems to be suitable for color interpolation. However, the three HSV parameters are not independent. Therefore, our optimization, which expects independent parameters, does not converge. The problem of divergence never arose in our artificial tests (based on promising initial values), because the test color varied only a little bit from the original color in the same color space. The interpolation between two alike colors is therefore often not linear.

To solve the color problem, either all channels have to be combined to one intensity value (gray-scale image), or each channel has to be optimized separately. To not further increase the complexity for our optimization problem, both RGB-images are converted to a gray-scale intensity image. This reduces the lens flare model to represent the border color only by one intensity, which is now completely independent.

7.1.2 Advanced Cost Function

The previously defined cost function (Equation 7.2) is not capable of distinguishing between background (noise) and actual lens flare elements.

A possible way to reduce the background noise is to apply a **smoothing** filter (Gaussian Filter Kernel) to both images – but this erases some details. While using a **threshold** the background is clipped away completely and erases all information below the threshold value. Additionally, the threshold value has to be chosen very carefully to not destroy the actual lens flare elements. Finding a suitable threshold value is not trivial.

As the current cost function is based on a quite simple pixel difference, a more sophisticated difference function can be used. In the field of image compression, the Structural Similarity (SSIM) index is used to describe the perceived similarity of a compressed image to its original [WBSS04]. SSIM composes three different comparison measurements (luminance, contrast and structure) to form its index, but unfortunately these rather costly additional features (caused by window-based sampling) do not improve the cost function, as it behaves quite similar to the basic MSE cost function.

The cost function can also be adapted by combining it with another cost function. The current cost function only considers the differences of intensity values, but does not pay any attention to shapes. The shape of a lens flare element is limited by a border edge. The edges of an image can be extracted by applying image filtering techniques. Following the idea to use intensity differences to evaluate image similarity, the edge images are also a valid comparison component. The discrete *Laplacian Filter* kernel is

able to detect horizontal, vertical and even diagonal intensity changes, mainly caused by recognizable edges (i.e. the gradient changes its direction). To improve the edge filtering result, the noise in the input images has to be first smoothed out by a *Gaussian Filter* kernel. The Gaussian filter kernel is used to blur and to “wash out” an image. The Laplacian filter kernel responds to gradient changes and therefore maps an edge as two one pixel wide lines to the result image (see figure 7.1 second column). While calculating the difference between two edge images, the chances are quite low to perfectly fit the one pixel wide edges. In most cases, the slender edges only cross each other, but not map completely. While increasing the edge width using an another Gaussian Filter, the chance rises that two edges are overlapping, or to affect the output while being close to each other (Figure 7.1 right column). This edge difference cost function and the MSE cost function are linearly combined and turn out to converge also with real-world data.

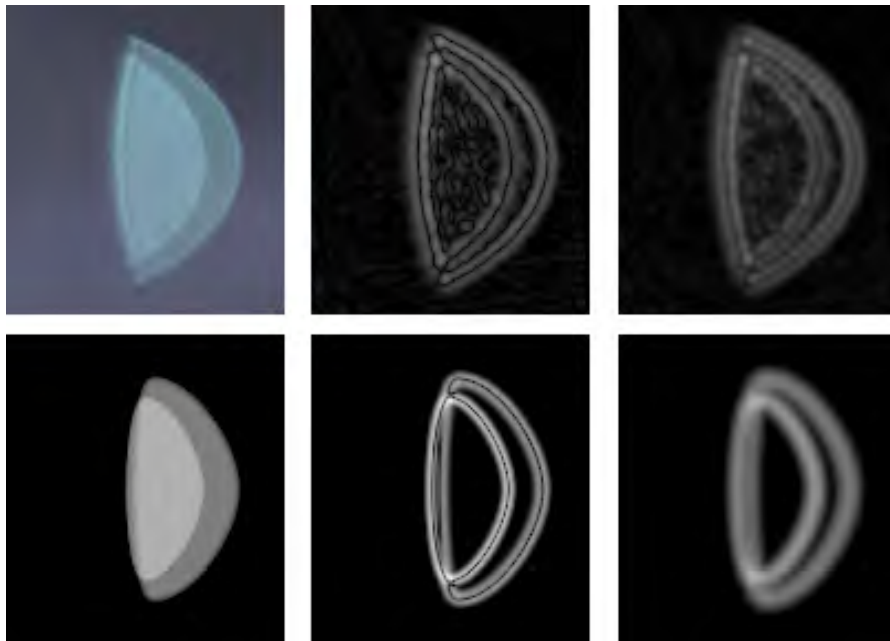


Figure 7.1: Edge cost function creation. Top row shows a captured ghost of the acquisition stage, while the bottom row visualizes a model based rendering. The first column is the input file. The second column is the result of a Gaussian-Laplacian filtering. The third column presents the result of the final edge cost. An additional Gaussian kernel is applied to the second column to increase overlapping in the difference images. For visualization purposes the edge images are brightened up.

7.1.3 Performance

The cost function is evaluated very often by the optimization algorithm, and therefore its execution time has to be as short as possible. The GPU allows to efficiently work with images or textures. The mean square error can be efficiently calculated in the

fragment stage of the rendering pipeline by writing the difference pixel value into a result texture. *Mipmap pyramids* provide an efficient representation to change the scale of a texture. The ground level of the pyramid represents the original image, while every layer successive downscales the image. The top of the mipmap pyramid is one single pixel, which represents the averaged value for the whole image. Applied to a difference texture, the mean difference of all pixels can be evaluated very efficiently.

Image filtering is based on convolution, where the filter kernel represents a discrete convolution matrix. To filter an image, the filter kernel calculates a value for each pixel based on the neighborhood weighting. The number of needed neighboring pixels depends on the kernel size, and heavily influences computation time. Both Gaussian and Laplacian filter kernel are 2D-kernels, and therefore the neighborhood evaluation is more complex than for a simple 1D kernel. Fortunately, the Gaussian filter kernel is separable and can be efficiently evaluated by executing the 1D-kernels sequentially. This property of the Gaussian filter allows to speed up the blur and smoothing operations. Unfortunately, the edge filtering operation, based on the Laplacian filter kernel, is not separable and has to be executed without an optimization.

7.2 Optimization Strategy

To generate a rendering similar to the captured image, the optimal parameter set for the model has to be found. The problem is to find the unknown parameter settings. The cost function maps the quality of the current solution to a numeric value, which indicates how close the actual state is to the wanted solution. Various approaches are able to derive additional information from the cost function, but unfortunately our cost function is complex, and the model parameters are not totally independent. For complex cost functions and a large number of model parameters (optimizing multiple lens flare primitives), an iterative optimization method is the only option to find the optimal solution efficiently. An iterative approach successively generates better approximations of the optimal solution. In each iteration, the current cost function output is compared to the last state. In case of less cost compared to the old cost, the new parameter changes are accepted and are used as start values for the upcoming iteration.

To ensure fast convergence, the optimization strategy has to decide on new values accordingly. The gradient of the cost functions is a good indicator to decide if a parameter has to be increased or decreased. *Multivariate Gradient Descent* is an iterative optimization algorithm, which follows the gradient (the steepest descent) to the next minimum. Due to its high complexity, the gradient of our cost function cannot be derived in closed form (too many variables), but the gradient can be numerically approximated by calculating the *central differential quotient* for each parameter (Equation 7.3).

$$\frac{\Delta y}{\Delta x} := \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x} \quad (7.3)$$

The numerical gradient approximation allows to increment or decrement each parameter correctly for the given state. The step size influences the convergence speed drastically. By using a too small step size, unnecessary iterations are carried out before reaching the solution. On the other hand, by choosing a too large step size, the optimal solution may not be reached or is only badly approximated. Therefore, for each model parameter, an individual step size is defined, as the parameters vary in range and magnitude. A global *learning rate* is defined (as used in machine learning approaches) to scale all step sizes. To ensure a fast convergence for not optimally initialized parameter values, the learning rate increases the step sizes. A large learning rate is used as long as possible to ensure a fast approximation. After various iterations, the parameters approximate the optimal solution quite well, but the long step size prevents the parameters to actually find a better solution. Therefore, after an unsuccessful iteration, the learning rate is reduced for the next iteration. Depending on the reduction factor of the learning rate, the optimal solution can be approximated more efficiently.

There are multiple update strategies to decide which parameters are manipulated within an iteration. A simple strategy chooses one parameter randomly in each iteration, but this approach only converges slowly with our data sets. To ensure that each parameter is equally often changed, the parameters are changed in a fixed order. Therefore, in each iteration, every parameter is changed once, but the parameter update can be differently applied to the model within an iteration. The optimized parameters can either be directly incorporated into the model before evaluating the remaining parameters (*online update*), or each parameter is tested against the current state, and after the iteration, all parameter values are updated at once (*batch update*) [CM06]. For our approach, the online-update is the more efficient strategy, because our model defines a strict order of its parameters (shape before shade).

7.3 Finding Lens Flare in the Acquired Images

By choosing initial parameters close to their optimal value, the optimization process has only to fine-tune the solution, and is less likely to get stuck in a local minimum. To ensure appropriate initial values the values are carefully chosen manually. To prevent other lens flare elements to influence the start parameters, an image containing multiple distinct ghosts is used. Furthermore, to support the user find valid initial parameters, multiple views are offered. Figure 7.2 depicts the user interface of initial parameter finding. The user interface fully supports our user-driven parameter finding workflow.

After the optimization process, the fine-tuned parameters represent the inspected lens flare as precisely as possible. A certain degree of inaccuracy remains because the model might not be capable of representing some details. The optimal model allows to compress the captured image content to a finite parameter set. This strong compression is of high importance for the visualization (Chapter 8) and occurrence prediction (Chapter 9).

In the acquisition stage, the samples are captured along the horizon, while the camera is rotated away from the light source. Depending on the sampling rate, the lens flare

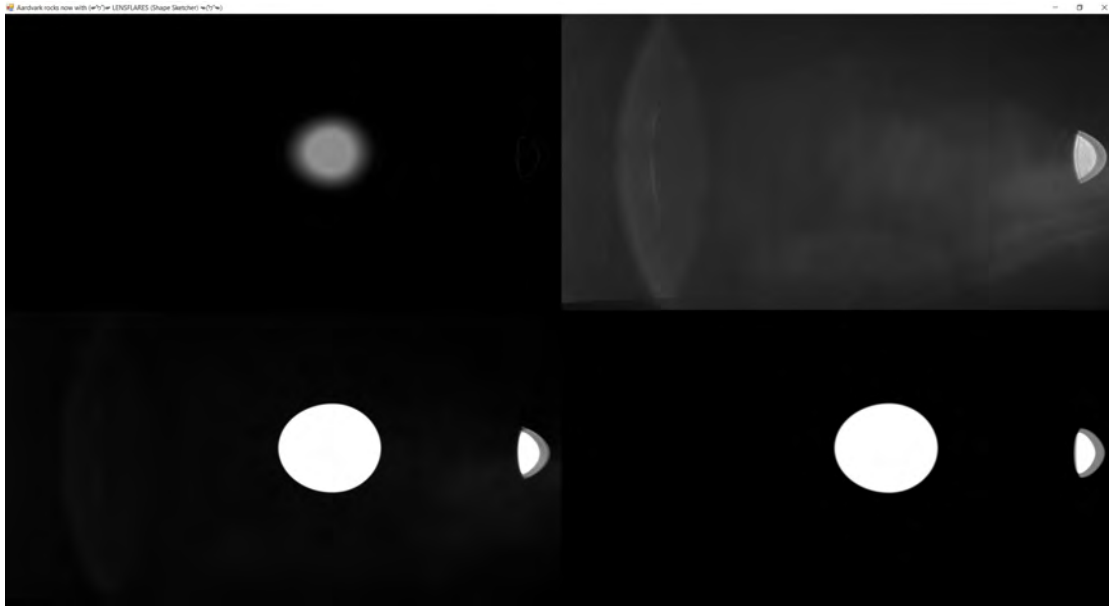


Figure 7.2: User interface of the user-driven initial parameter approach. Top left: cost function of the current state – bright pixel indicate error. This view is mainly used to find the optimal shade parameters of the model. | Top right: captured sample | Bottom right: current rendering (where the big white ghost is placed intentionally for visualization purpose) | Bottom left: captured sample with rendering overlay. The overlay allows the user to fine tune the shape parameters

elements only slightly differ from one sample to the next. While using a dense sampling rate, the difference between two samples is only marginal. The neighbor relationship is crucial for the optimization process, because it allows to re-use the just found optimal parameter set as start values for the closest neighbor samples. This reduces the amount of manually chosen initial parameter drastically, but the quality for the next initial parameters strongly depend on the sampling rate.

To improve the precision for the initial parameters of the next neighbor sample, more than only sample can be used. As the behavior of lens flare is smooth over the whole range, like a continuous signal, the parameter changes for several samples can be described by a function. For the first sample no information is available, therefore the initial parameters have to be set manually. The next neighbor sample is already able to re-use the previously found parameter set like a constant function. For the third sample, the two previously matched parameter sets define a linear function, which can be extrapolated to estimate the initial parameter. For each additional available matched sample, the function becomes more precise. A polynomial can be used, to best fit the available matched parameters values in a *least-squares* sense, to define the prediction function. To avoid over-fitting and extreme extrapolated predictions, the degree of the polynomial has to be kept as low as possible.

To ease the manual initial parameter setting process, a sample of the range is chosen, where the lens flare element of interest is clearly visible. While rotating the camera, the lens flare element may disappear or move out of the frustum. While tracing a lens flare element over a range of samples, other lens flare elements may fade-in and disturb the examined lens flare element. To correct the intensity contribution of other lens flare elements, the in-fading lens flare is ignored as long as it cannot be clearly identified. After a possible identification of the disturbing lens flare element, the parameters are set again manually. To correct the contribution of the freshly modeled lens flare element in the previously optimized samples, the sampling order is simply reverted. While reverting the sample order, the fade-in of a lens flare element is changed to a fade-out.

Real-Time Visualization

The optimization stage (Chapter 7) is able to generate a best-fitted model for a lens flare element (Chapter 6) for each sample of the acquisition stage (Chapter 5). Fortunately, the model is capable to render its lens flare elements in an efficient way so it can be used for real-time rendering. To generate a visualization for a given sample, all lens flare elements are directly rendered into one image. An interactive visualization allows the user to change the camera to light constellation. To find the optimal visualization for a certain constellation, different approaches can be used (Chapter 8.1). The interactive visualization can be included into a 3D software while using the visualization as a scaled overlay in the post processing stage (Chapter 8.2). The scale factor of the overlay depends on the camera parameters and the light source.

8.1 Interactive Visualization

The lens flare occurrence and appearance depends on the camera to light constellation (Chapter 3.2.3). The simplest approach to visualize a certain constellation is to find the closest sample for a given constellation, and use its compressed model representation directly. This simple approach strongly depends on the sampling rate of the camera to light constellation in the acquisition stage. A rough sampling leads to visible jitters, because the appearance of lens flare changes noticeably between two neighboring samples. To avoid these visible jitters the discrete step size of the sampling has to be reduced, but this increases the amount of work for the whole workflow extremely.

To avoid an extremely dense sampling rate and to ensure a smooth transition of lens flare over a range of constellations, the matched samples have to be interpolated. As mentioned in chapter 7.3, while changing the camera to light constellations, lens flare acts like a continuous signal. The simplest approach to avoid the visible jitters is to interpolate the parameter values between the neighboring samples. This simple neighbor interpolation reduces the jitter, but the transition over multiple samples may still be

uneven. To improve the smoothness over a range of samples, the original signal of the lens flare has to be better approximated. Therefore, for each parameter of the model a smooth function is necessary. The optimally matched parameter sets for each constellation are used to approximate the unknown function by a polynomial based on *Least Squares Fitting*. The resulting polynomial is guaranteed to be continuous and while trying to restrict the polynomial to a low order, the risk of overfitting can be reduced. Furthermore, the look-up complexity for a certain constellation is reduced for each parameter by one polynomial evaluation.

8.2 Lens Flare Incorporated Into An Application

The interactive lens flare visualization can be incorporated into any 3D application as long as the camera to light constellation is available. To create a lens flare effect, the lens flare visualization has to be generated and rotated accordingly. To allow a realistic lens flare effect, the virtual camera of the 3D application has to simulate the camera parameters of the captured camera. Especially the focal length and aperture shape have to be the same to produce a physically plausible lens flare effect. The lens flare visualization can be integrated into the 3D application within a post process. Therefore, the final visualization is scaled to match the brightness of the virtual scene and then added to the 3d application's rendering.

8.2.1 Constellation Angle

While knowing the camera position C_{pos} , camera orientation $C_{forward}$ and the light position L_{pos} , the **rotation angle of the capturing process** (constellation angle) is defined by equation 8.1. The rotation angle α is applied to parameter functions to generate the optimal lens flare visualization.

$$constellation = \arccos (|C_{forward}| \cdot |L_{pos} - C_{pos}|) \quad (8.1)$$

8.2.2 Lens Flare Orientation

According to our assumptions, the lens system is radially symmetrical to its optical axis (Chapter 5.2). Therefore, the lens flare rendering can be rotated around its image center to match the **roll angle** of the light source and the camera's horizon. To rotate the rendered image counter-clockwise around the camera's optical axis, the rotation angle β is formulated by the following equation 8.2, where the light position L_{pos} is transformed into the view space L_{vs} .

$$roll = \arctan 2 (L_{vsY}, L_{vsX}) \quad (8.2)$$

8.2.3 Aperture and Focal Length

Most camera models (Chapter 4.3) in computer graphics applications integrate the **focal length** as a changeable parameter to define the camera's frustum. It is not possible to simply scale our lens flare rendering to match a different field-of-view, because lens flare occurrence depends on the captured focal length, and the occurrence can differ strongly. The **aperture stop** also affects the occurrence of lens flare and has a great impact on the brightness. For a realistic lens flare effect, the camera model has to match the sampled aperture stop and aperture shape to generate the corresponding ghost shapes. In some camera models, the aperture shape is not included, but the aperture stop is still important for the brightness.

8.2.4 Brightness

The brightness of an image depends on various camera parameters like (**aperture stop**, **film speed**, **exposure time**) and on the **light sources**. A physically based camera model is able to simulate the brightness which is caused by the intrinsic camera parameters (aperture stop, film speed and exposure time). The brightness ratio between lens flare visualization and scene rendering is important to be able to add the visualization in the 3D application's output.

In case the virtual camera and light source of the 3d application match the used parameters of the acquisition stage, the brightness is equal and the lens flare visualization is simply added to the scene rendering. In case any parameter differs from its assumed reference value, the brightness has to be corrected. The film speed *ISO* describes the sensitivity of the sensor or film. While doubling the sensitivity, the brightness is also increased by the same amount. This also applies to the exposure time *EXP* (seconds). While exposing the sensor twice as long to the light, the image becomes also twice as bright. The *f*-number describes the ratio of the aperture diameter and the focal length. The focal length has to be constant, as mentioned previously (Chapter 8.2.3). Therefore, only the aperture is changeable. Each aperture stop is described with a corresponding $\frac{1}{f}$ -number. While decreasing the aperture diameter by $\frac{1}{\sqrt{2}}$, the *f*-number increases, but the incoming light is halved. The incoming light at the entrance pupil also influences the brightness of the lens flare. In case the luminance ($l = \frac{\text{candela}}{\text{m}^2}$) and the solid angle α of both lights sources are known, a brightness ratio can be formed. Equation 8.3 describes the brightness scale factor B_{scale} between the lens flare rendering F and the rendered scene R . The brightness factor is used to scale the lens flare visualization accordingly to match the scene brightness.

$$B_{scale} = \frac{f_F^2}{f_R^2} * \frac{ISO_R}{ISO_F} * \frac{EXP_R}{EXP_F} * \frac{l_R * \alpha_R}{l_F * \alpha_F} \quad (8.3)$$

8.2.5 Wavelengths Distribution - Light Type

The incoming light type has an impact on the lens flare's occurrence, because the anti-reflection coatings are designed for specific wavelengths. Therefore, the virtual light source has to match the sampled light's wavelength distribution to guarantee a realistic lens flare. Unfortunately, each light type has its unique wavelength distribution. To ensure a realistic lens flare effect, the whole workflow has to be executed for each light separately. By using a normed light source with a known wavelength distribution, other light sources with a similar distribution can be roughly approximated.

Lens Flare Occurrence Prediction

The interactive visualization allows to generate an artificial lens flare representation for any camera to light constellation within the captured range (Chapter 8). The visualization of the lens flare elements is intended to be as physically accurate as possible, but due to approximations (lens flare element representations may not be able to capture all details) and acceleration strategies (interpolating of model parameters), only a physically plausible representation can be ensured. To predict the occurrence of lens flare, the real-time visualization can be re-used, where each model parameter is described over the whole range by a function. The interpolatable parameter functions are used to calculate an occurrence estimator (Chapter 9.1). The occurrence estimator is then used to predict the impact of lens flare for a certain constellation within a known scene (Chapter 9.2).

9.1 Occurrence Estimator

For a given constellation, the real-time visualization allows to retrieve various statistical values, which are used to form an estimator. The precision of all estimators heavily depends on how precisely the acquired image has been described by the lens flare models (the current implementation only supports ghosts). The brightness scale factor tries to adjust the brightness of the lens flare visualization to match the brightness of the virtual scene approximately (Chapter 8.2).

The real-time visualization is based on models with interpolatable parameter functions. The same idea on how to compress the model parameters to a parameter function (least square fitting) can also be applied to each statistical term (*statistic function*). To generate various statistic functions, for each best-fitted sample of the acquisition stage, various statistical terms have first to be evaluated and then approximated by *least square fitting*. The intensity values of the resulting lens flare rendering are useful to form good occurrence estimators. The *minimum* and *maximum* intensity values are important to define contrast. The *sum* and *average* of all intensity values are able to approximate

brightness. The *logarithmic average* is more stable against outliers and is often used in tone mappers to describe the key for a scene [RSSF02].

9.2 Context

All previously described occurrence estimators represent a numerical value, but without any context or bounds, these values are not really meaningful. Depending on the context, different statements can be made.

While comparing the current occurrence estimator to the estimators over the whole sampled range, the current constellation can be rated. By inspecting the statistic function of a certain estimator, global extremes can be derived and also used to express the quality of the current state. To classify the estimator, the range between maximum and minimum of the inspected estimator are subdivided into several bins.

The virtual scene can be used as a context to describe the impact of lens flare. While knowing all necessary camera and light parameters, the scene brightness is estimated. While comparing the logarithmic average of the scene to the lens flare overlay, the ratio is used as a glare indicator. The presence of lens flare reduces the contrast of an image. The *ideal contrast* K_{ideal} from a light is given by equation 9.1.

$$K_{ideal} = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} \quad (9.1)$$

While lens flare occurs within the scene, the *real contrast* K_{real} decreases, because the lens flare intensity I_{flare} is added equally to I_{min} and I_{max} (Equation 9.2).

$$K_{real} = \frac{I_{max} - I_{min}}{I_{max} + I_{min} + 2I_{flare}} \quad (9.2)$$

The ratio between real contrast K_{real} and ideal contrast K_{ideal} can be used to describe the change of the contrast while lens flare occurs (Equation 9.3) [Kes08].

$$\frac{K_{real}}{K_{ideal}} = \frac{I_{max} + I_{min}}{I_{max} + I_{min} + 2I_{flare}} \approx 1 - \frac{2I_{flare}}{I_{max} + I_{min}} \quad (9.3)$$

Results

In this chapter the results for each stage of the workflow are presented. While implementing the workflow, the results of already working stages gave great insight and helped a lot to decide how to tackle the next stages.

10.1 Result - Acquisition

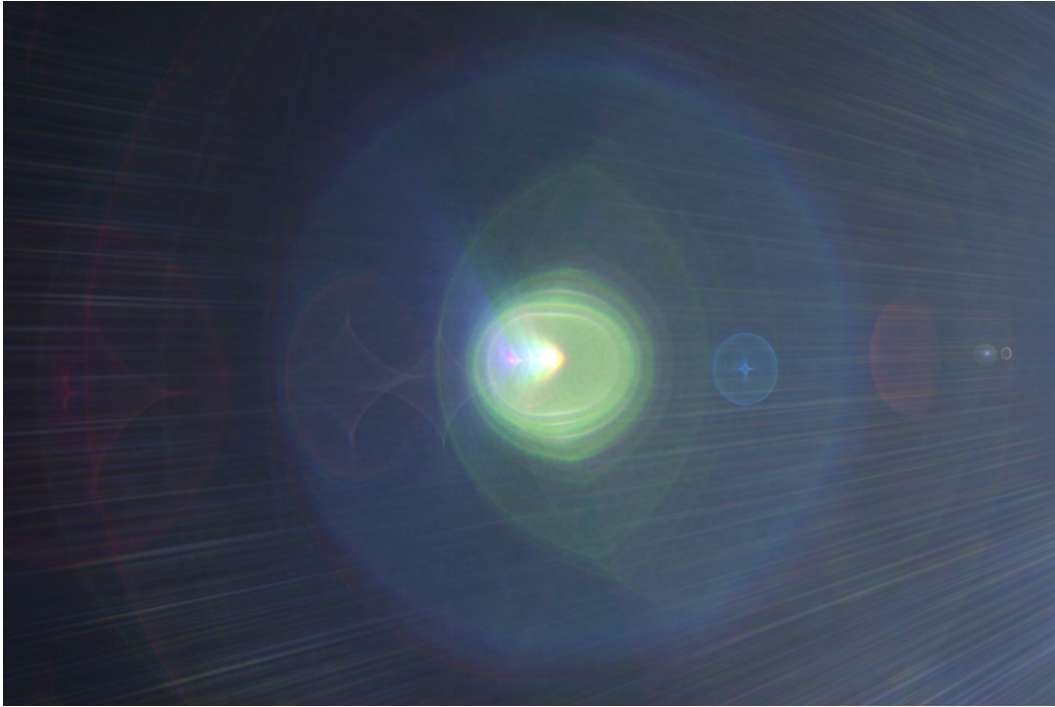
The following images are the examples results from the acquisition stage. Figure 10.1a shows the start constellation, where the camera is directly oriented into the light source. In this constellation the ghosts are centered around the light source and overlap each other. While rotating away from the light source the ghost are distinguishable as figure 10.1b depicts. Figure 10.2a shows the complex ghosts from the previous figure 10.1b in detail. Figure 10.2b gives a rough overview how lens flare behaves, while the camera is rotated away from the light source.



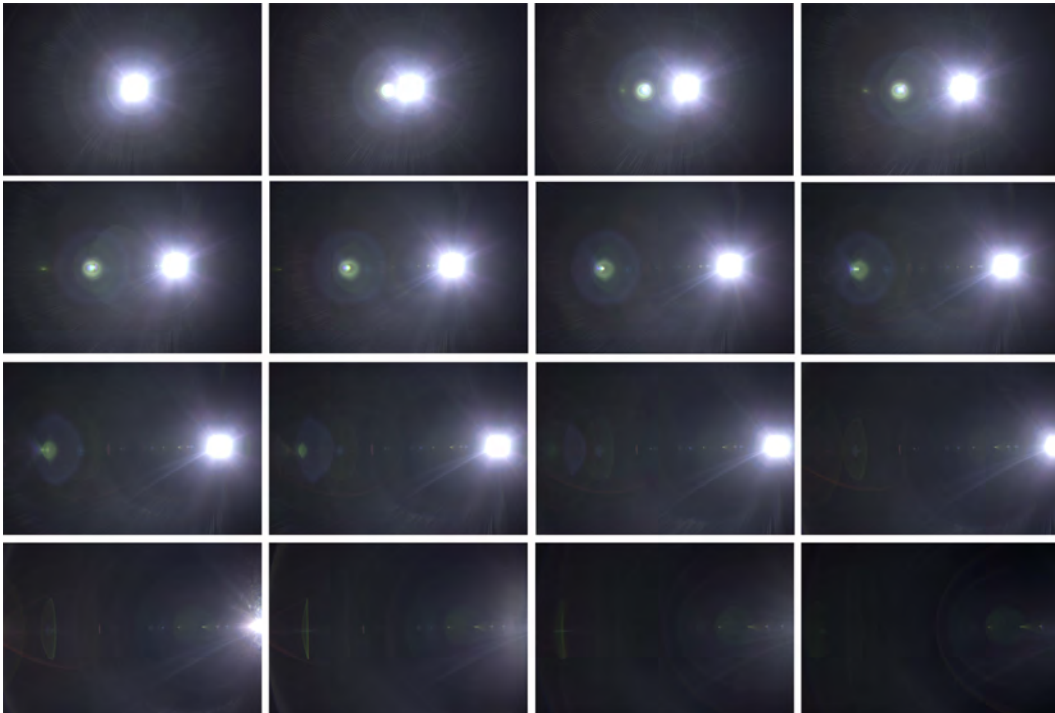
(a) Camera oriented into the light source



(b) Camera rotated 22.8° – distinguishable ghosts



(a) Closeup of ghost at 22.8°



(b) Rough overview of our sampled data set

10.2 Result - Lens Flare Model

The lens flare model holds a dense parameter set and is very flexible to render various ghost shapes. Figure 10.3 shows how the appearance of a ghost can change, while only manipulating the distance function (see chapter 6.3.2 for parameter information).

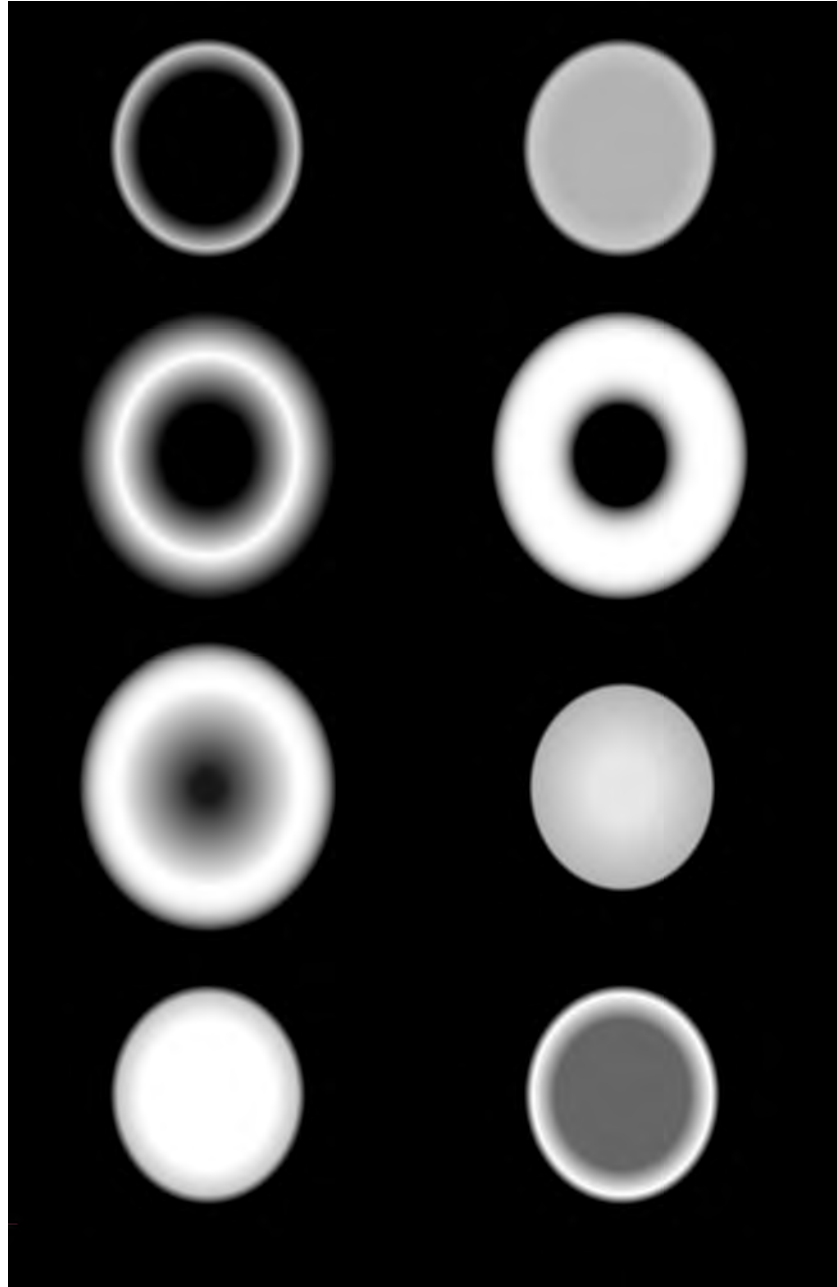


Figure 10.3: Circular shaped ghost with different distance function

Figure 10.4 depicts a selection of shapes, which our model is capable to render (see chapter 6.3.1 for parameter information).

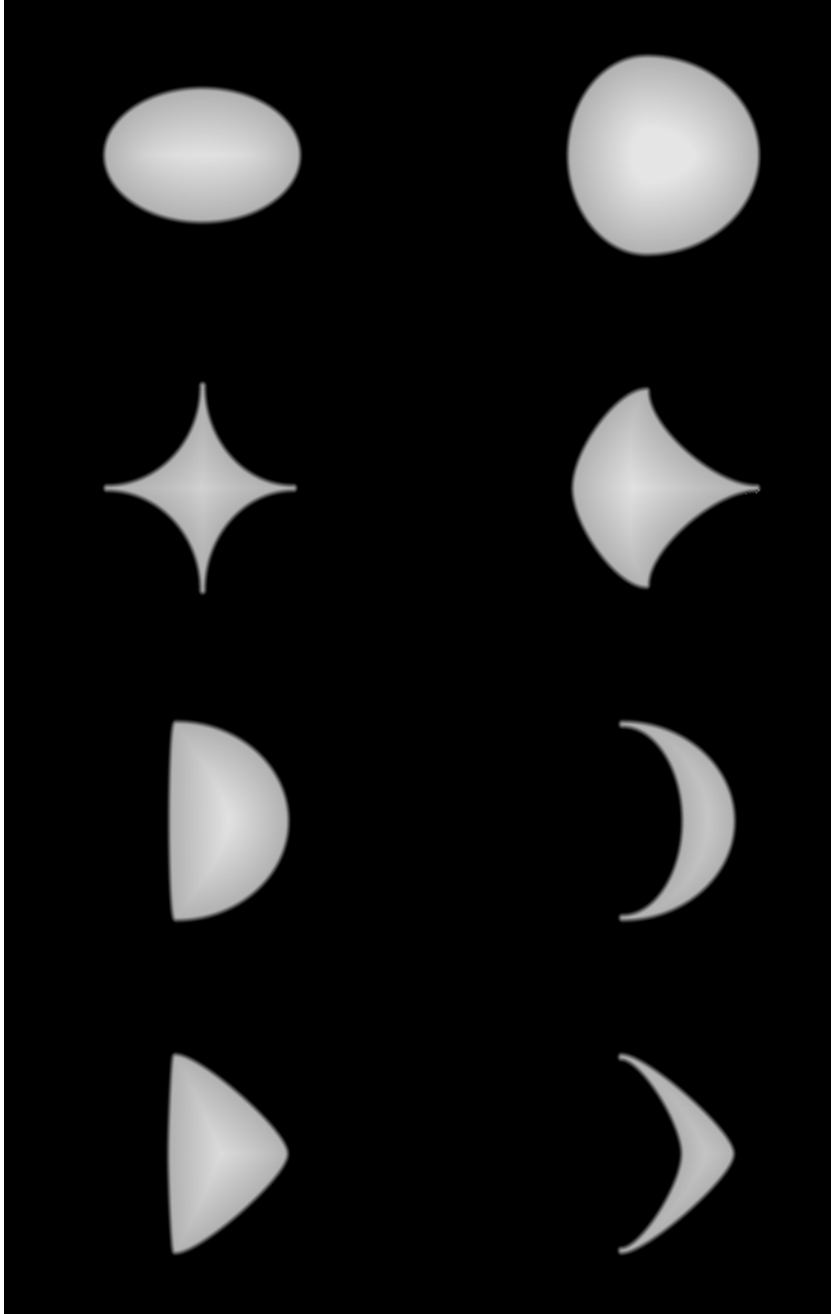


Figure 10.4: Various ghost shapes generated with our model

10.3 Result - Optimization

The optimization algorithm uses the cost function to fine-tune the model parameters to best fit a sample. Figure 10.5 visualizes the optimization process between neighboring samples and their corresponding models.

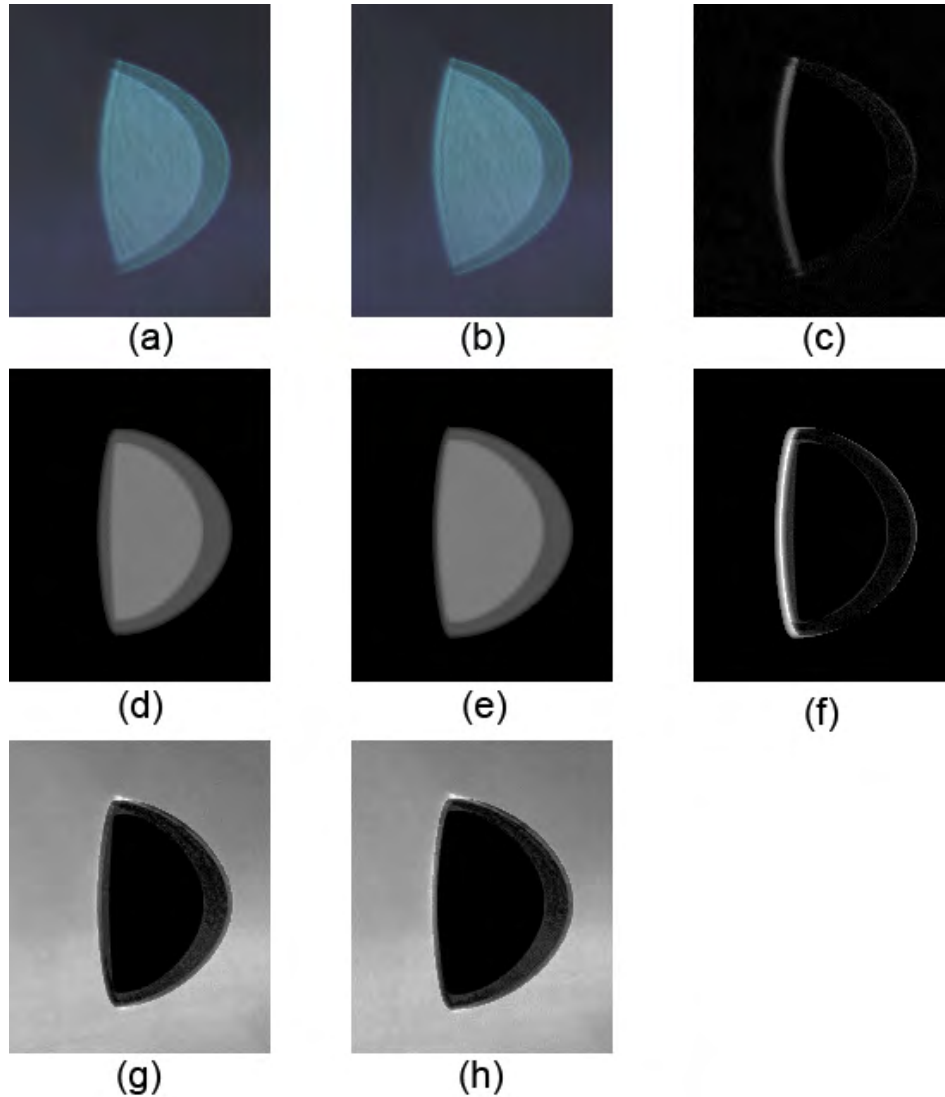


Figure 10.5: Optimization process from manually placed model to the neighboring sample. Images (a) and (b) are succeeding samples of the acquisition stage. Image (d) is a rendering of a rough hand tuned model for the first sample (a). The parameters in (d) are the initial values for the optimization of (b). The result of the optimization is depicted in image (e). Images (c/d/g/h) visualize the differences of the row or column images (bright areas indicate difference).

Figure 10.6 shows the progress of the cost function and its components while optimizing the model of the previous figure 10.5.

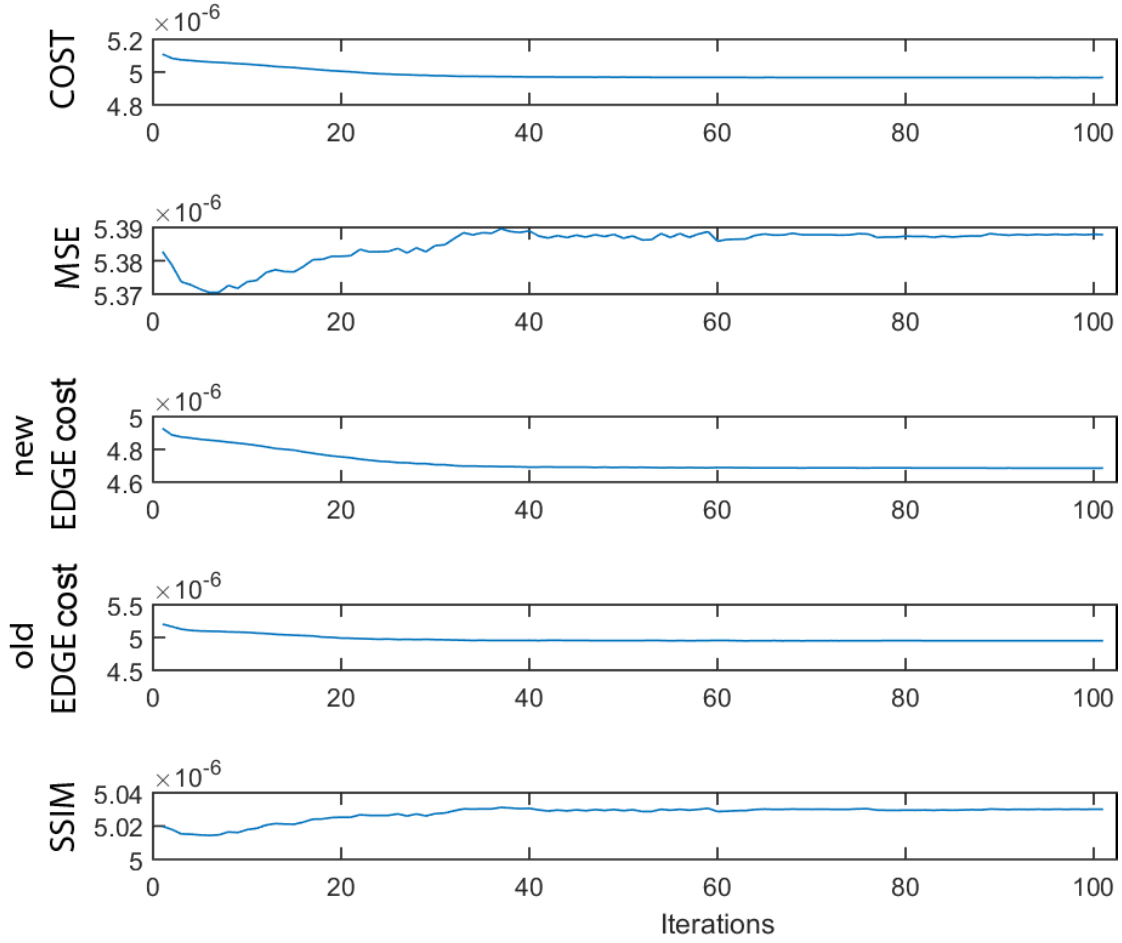


Figure 10.6: Progress of the cost function and its components, while best fitting the model depicted in figure 10.5(d) to (e). The first row shows the cost, which is a linear combination of the *MSE* cost (second row) and the *new edge cost* (third row). The fourth row shows the *old edge cost* (Chapter 7.1). The last row shows *SSIM* index, which behaves similar to MSE, but is more expensive to compute. SSIM and the old edge cost are only depicted to visualize their behavior. The increase of the MSE after a few iterations is caused by the new edge cost. The new edge cost ensures that the MSE does not emphasize background noise too much.

Figure 10.7 shows the resulting parameter value of a range optimization of 13 samples.

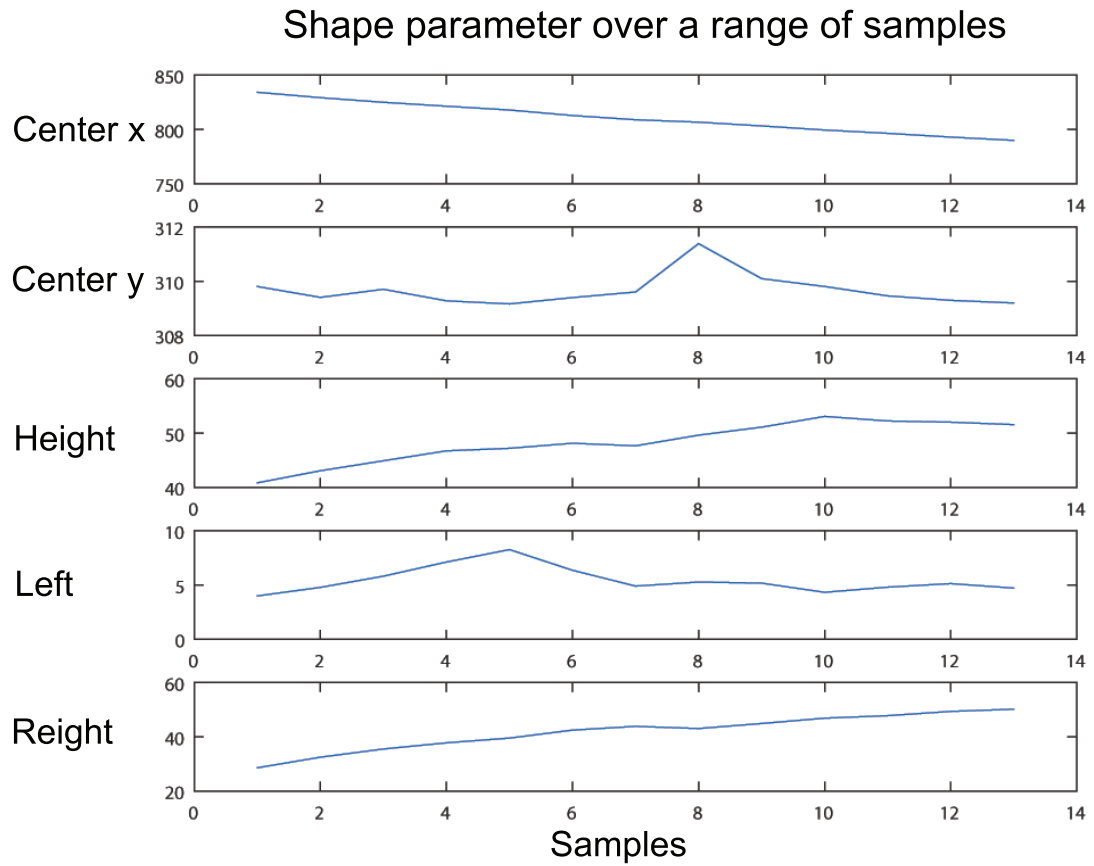


Figure 10.7: Shape parameter values over a range of 13 samples. The coordinates of the center in image space describe a linear movement to the left side (decrease of x value and nearly steady y value). The height slowly increases while the left control point to the element is nearly static. The right control point slowly increases.

To generate a smooth real-time visualization the resulting values of a range optimization have to be described by a polynomial. The approximation of the values to a function is done by *Least Squares Fitting*. The polynomial allows to bridge neighboring samples. To demonstrate the impact of the polynomial order, the center parameter of the previously discussed range optimization (Figure 10.7) are used as an example.

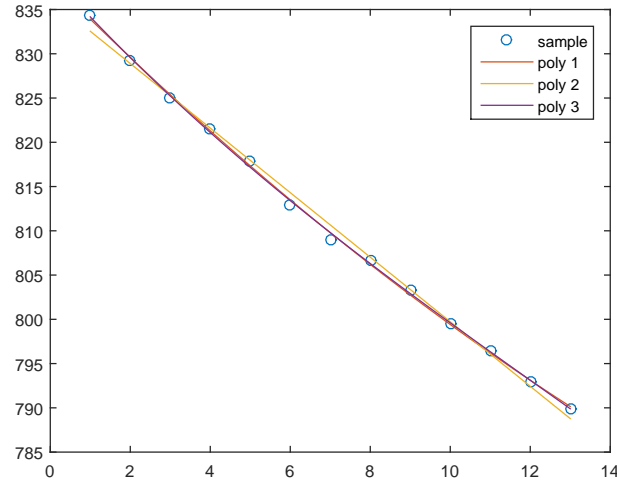
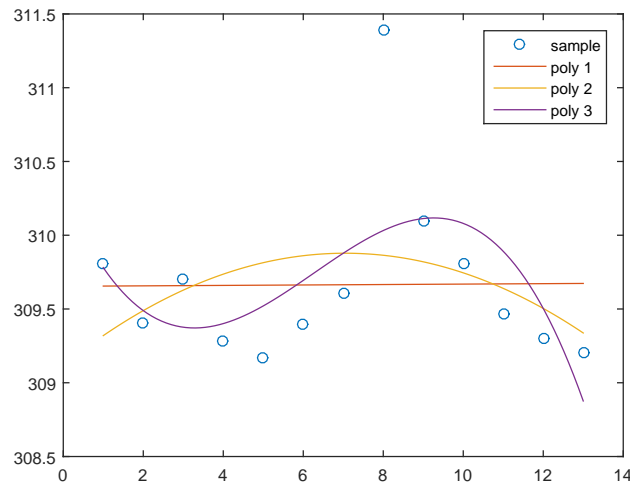
(a) Lens flare element center x -coordinate(b) Lens flare element center y -coordinate

Figure 10.8: *Least Squares Fitting* of the center parameters over 13 samples. A proper polynomial order has to be chosen to avoid overfitting (poly 3) or underfitting (poly 1). Polynomial order 2 is suitable for our case.

Figure 10.9 shows the live visualization of the optimization process.

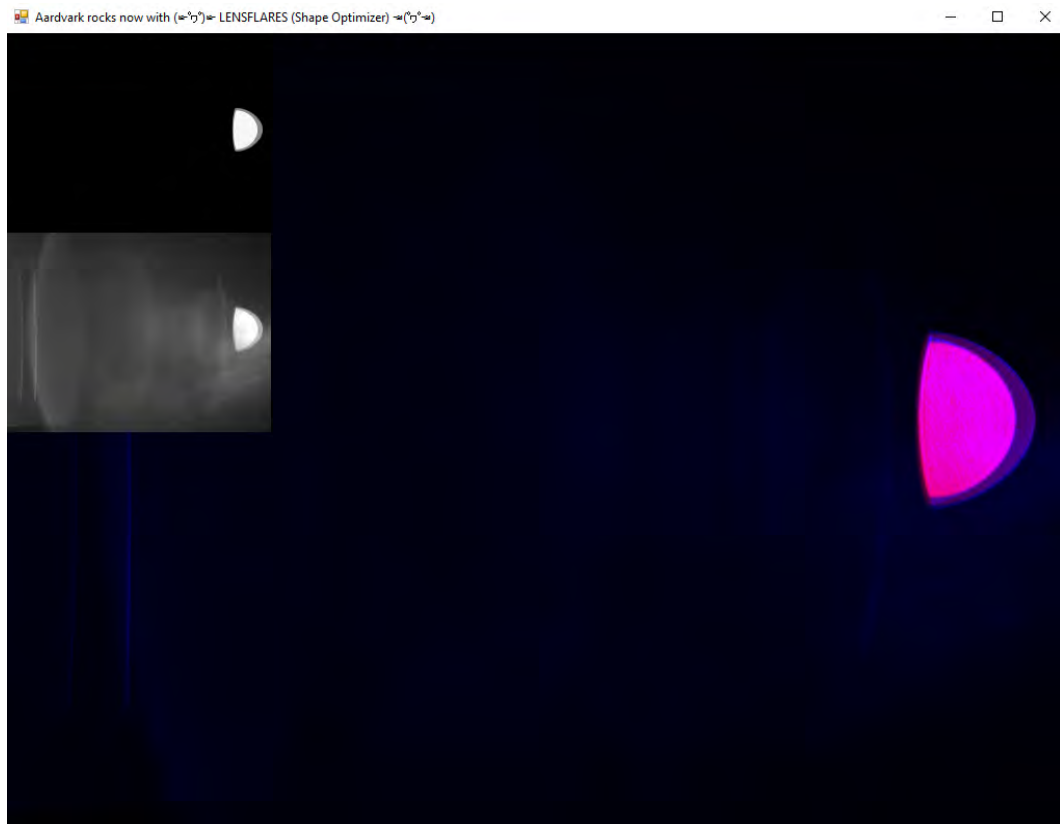


Figure 10.9: Optimization visualization. Top left is the current state of the model based rendering. Middle left is the input image. The large image shows the overlay of the input image (blue) and the current rendering (red). The user easily recognizes divergence and convergence.

The performance evaluation in figure 10.10 shows the rendering performance versus element resolution.

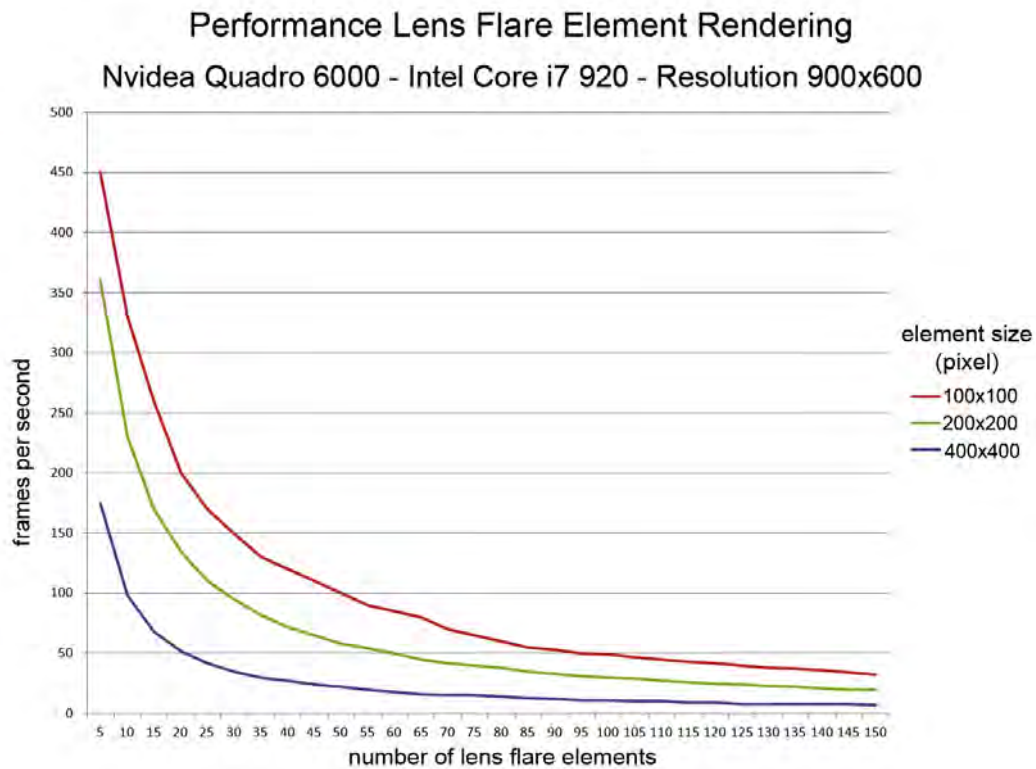


Figure 10.10: Rendering performance strongly correlates with the element resolution. Real-time performance can be achieved for more than 100 elements when using smaller resolutions. In real world applications only few lens flare elements are outstandingly large.

Conclusion and Future Work

In this thesis we describe a whole workflow on how to create a physically plausible lens flare rendering from measurements. The workflow is designed from bottom-up in stages (acquisition - optimization - rendering - prediction), and allows to stop at a certain stage, if the current result already fulfills the application's goal. The lens flare quality of simulation based approaches heavily depends on the exact description of the lens system and often approximations have to be used (especially for the anti-reflection coatings). Our workflow is based on raw measurement data, and therefore the whole light traversal through the optical system is captured without any approximations. This allows us to avoid the retrieval of hardly available internal parameters of the lens system and its complex components. The developed measurement setup is camera independent and easy to use. Our lens flare model is able to emulate the appearance of captured lens flare elements in the acquired data set, and can be easily extended to depict more details. The model can be used for various applications and is a good description of a lens flare element. The model-based representation is manually tweaked to approximate a sampled lens flare element. To increase the similarity of the rendered lens flare element to its sampled counterpart, we present an optimization strategy including a cost function. To reduce the complexity and create a continuous description of the lens flare behavior over the sampled range, we present an efficient way to compress the model data. We present an efficient implementation to render the lens flare elements in real-time. While using the proposed compression technique, a smooth visualization can be achieved. We describe how the lens flare visualization can be integrated into other rendering applications as a post processing effect. In the last stage we sketch how the impact and occurrence of lens flare can be expressed and predicted.

This thesis describes a basic workflow to render lens flare based on measurements, but there are many possible improvements to increase performance and quality at various stages.

By capturing more samples in the acquisition stage, also non-radial symmetrical lens system can be approximated. The appearance of lens flare is influenced by the focal length and aperture stop. To capture all possible variations, all aperture stops and focal lengths have to be sampled as well, which increases the net complexity drastically. To reduce this effort, some aperture stops and focal lengths might be skipped and later reconstructed by interpolation. By finding the optimal capture settings for a given light source, HDR may become redundant, which could lead to a shorter acquisition time.

The current lens flare model is restricted to a fully open aperture (circle-shaped), and can be extended to represent also more complex aperture shapes and resulting lens flare elements. The model's transfer function can also be changed and adapted to describe additional details. The current optimization stage is only able to handle intensity values, while we suggest to include color by inspecting every color channel separately. By finding a way to correctly interpolate color, color can be treated like intensity values as one channel. The color of the ghost heavily depends on the anti-reflection coatings and the incoming wavelengths, different kinds of light sources can be used to detect their influence. To fully reconstruct a captured sample of the acquisition stage, models for the *star-shape* and *halo* lens flare elements have to be developed.

To ensure a fast convergence in the optimization stage, currently the initial parameters are set manually. To find good initial values, (semi-)automatic methods can be developed. A suggestion to find better initial shape parameters is to apply a scan-line algorithm onto the edge images of a lens flare element. To avoid invalid parameter constellations which break the model design or do not affect the rendering, regularization can be used to "punish" certain parameter changes. The cost function can also be further improved by improving its convergence behavior and by making it less error-prone. The currently used optimization strategy can also be further improved by tweaking the used optimization parameters or by dividing the optimization process into several subroutines (optimizing the shape based on edge images first, and then solve the intensity problem in a second run). Many other approaches can be used to track the ghost elements over the whole range, like *structure from motion* or *neural networks*.

List of Figures

2.1	Outdoor lens flare with ghosts - from snty-tact (Talk). Taken at Yosemite National Park, California, U.S., CC BY-SA 3.0	4
2.2	Football live broadcast with disturbing lens flares	6
3.1	Complex lens flare	9
3.2	Stray light is not intended to enter the lens system	10
3.3	Simple camera model showing all essential optical elements (lenses, aperture and sensor)	11
3.4	Comparison of diffraction spikes for different apertures	12
3.5	Illustration of a possible double-reflection path	13
3.6	Geometry comparison of zoom systems - Nikon 80-200mm f/2.8 vs prime system - Angenieux Biotar 100mm f/1.1	14
3.7	Different aperture shapes caused by varying blade count	14
3.8	Illustration of camera light constellation	15
3.9	Illustration of how ghosts change their position while the camera is rotated. .	16
3.10	The effective use of scene elements and their composition can reduce lens flare	17
3.11	Illustration of how a lens hood can reduce stray light	18
3.12	Reflectivity per wavelength and coating composition	19
3.13	Comparison of coatings quality over the years. Both image are taken with similar settings	20
3.14	Different aperture stops and the resulting aperture shapes	21
4.1	Thin lens model	24
4.2	Thick lens model	25
4.3	Optical desing by Brendel (USP 2854889) $f/2.8$, 100mm effective focal length	26
4.4	Lens description and simulation in <i>ZEMAX</i>	29
4.5	Gray-scale lens flare sprites and resulting lens flare effect	30
4.6	Lens flare sprites used in Serious Sam 2	31
4.7	Synthetic flare simulation by Keshmirian [Kes08]	32
4.8	Simulation result of diffraction and ghosts [SDHL11] and example scene with combined lens flares	32
4.9	Itoh zoom lens	33

4.10	Lens flare results from Lee et. al.[LE13] (top) and Hullin et. al. [HESL11] (bottom)	34
4.11	Example of imperfection textures from Joo et. al. [JKL ⁺ 16] applied onto Lee et. al.[LE13] lens flare renderings	36
5.1	Flashlight test setup and result (30 seconds)	39
5.2	LED test setup and result (10 seconds)	40
5.3	Medical light test setup and result (1 second)	40
5.4	Final setup with curtain and automatic panorama head	40
6.1	Cubic Bézier curve construction	43
6.2	Shape model for a lens flare element	44
6.3	Transfer function for a lens flare element	46
6.4	Minimal distances from a query point to the Bézier curves	47
7.1	Edge cost function - input images (left) edge cost function (middle) improved edge cost function (right)	52
7.2	Interface of the user-driven initial parameter approach	55
10.3	Circular shaped ghost with different distance function	66
10.4	Various ghost shapes generated with our model	67
10.5	Optimization process from manually placed model to the neighboring sample	68
10.6	Cost function of the optimization process (depicted in figure 10.5)	69
10.7	Shape parameter values over a range of 13 samples	70
10.8	<i>Least Squares Fitting</i> of the center parameters over 13 samples.	71
10.9	Optimization GUI	72
10.10	Performance of lens flare element rendering	73

Acronyms

API Application Programming Interface. 31

CPU Central Processing Unit. 45

CSG Constructive Solid Geometry. 32

GPU Graphical Processing Unit. 31, 41, 45, 47, 49, 52

HDR High Dynamic Range. 38, 76

LED Light-Emitting Diode. 39

MSE Mean Squared Error. 50–52

SSIM Structural Similarity. 51

Bibliography

- [Als09] Ted Alspach. Vector-based representation of a lens flare, 2009. US Patent 7,526,417.
- [BHK⁺03] Brian A. Barsky, Daniel R. Horn, Stanley A. Klein, Jeffrey A. Pang, and Meng Yu. Camera models and optical systems used in computer graphics: Part i, object based techniques. In *Proceedings of the 2003 International Conference on Computational Science and its Applications*, 2003.
- [Cha07] Julian Chaumond. Realistic camera - lens flare, 2007. <https://graphics.stanford.edu/wikis/cs348b-07/JulienChaumond/FinalProject>.
- [CM06] Bishop Christopher M. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [Gre04] John E Greivenkamp. *Field Guide To Geometrical Optics*, volume 1. SPIE Press Bellingham, Washington, 2004.
- [HD14] Johannes Hanika and Carsten Dachsbacher. Efficient monte carlo rendering with realistic lenses. In *Computer Graphics Forum*, volume 33, pages 323–332. Wiley Online Library, 2014.
- [Hen15] Padraic Hennessy. Implementation notes: Physically based lens flares, 2015. <https://placeholderart.wordpress.com/2015/01/19/implementation-notes-physically-based-lens-flares/>.
- [HESL11] Matthias Hullin, Elmar Eisemann, Hans-Peter Seidel, and Sungkil Lee. Physically-based real-time lens flare rendering. *ACM Trans. Graph.*, 30(4):108:1–108:10, July 2011.
- [HHH12] Matthias B Hullin, Johannes Hanika, and Wolfgang Heidrich. Polynomial optics: A construction kit for efficient ray-tracing of lens systems. In *Computer Graphics Forum*, volume 31, pages 1375–1383. Wiley Online Library, 2012.
- [HSS97] Wolfgang Heidrich, Philipp Slusallek, and Hans-Peter Seidel. An image-based model for realistic lens systems in interactive computer graphics. In *Graphics Interface*, volume 97, pages 68–75, 1997.

- [JKL⁺16] Hyuntae Joo, Soonhyeon Kwon, Sangmin Lee, Elmar Eisemann, and Sungkil Lee. Efficient ray tracing through aspheric lenses and imperfect bokeh synthesis. In *Computer Graphics Forum*, volume 35, pages 99–105. Wiley Online Library, 2016.
- [Kes08] Arash Keshmirian. *A Physically-Based Approach for Lens Flare Simulation*. ProQuest, 2008.
- [Kil00] Mark J. Kilgard. Fast opengl-rendering of lens flares, 2000. <https://www.opengl.org/archives/resources/features/KilgardTechniques/LensFlare/>.
- [Kin92] Rudolf Kingslake. *Optics In Photography*, volume 6. SPIE Press, 1992.
- [Kin00] Yossarian King. 2d lens flare. In Mark DeLoura, editor, *Game Programming Gems*, pages 515–518. Charles River Media, Inc., Rockland, MA, USA, 2000.
- [KMH95] Craig Kolb, Don Mitchell, and Pat Hanrahan. A realistic camera model for computer graphics. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 317–324. ACM, 1995.
- [Kor07] Norman Koren. Veiling glare (lens flare), 2007. <http://www.imatest.com/docs/veilingglare/>.
- [LE13] Sungkil Lee and Elmar Eisemann. Practical real-time lens-flare rendering. In *Computer Graphics Forum*, volume 32, pages 1–6. Wiley Online Library, 2013.
- [LM14] Industrial Light and Magic. Openexr, 22014. <http://www.openexr.com>.
- [Mau01] Chris Maughan. Texture masking for faster lens flare. In Mark DeLoura, editor, *Game Programming Gems 2*. Charles River Media, Inc., Rockland, MA, USA, 2001.
- [Mch05] Sean Mchugh. Understanding camera lens flare from cambridge in colour, 2005. <http://www.cambridgeincolour.com/tutorials/lens-flare.htm>.
- [Pix08] Pixar. The imperfect lens: Creating the look of wall-e. wall-e three-dvd box, 2008.
- [RIF⁺09] Tobias Ritschel, Matthias Ihrke, Jeppe Revall Frisvad, Joris Coppens, Karol Myszkowski, and H-P Seidel. Temporal glare: Real-time dynamic simulation of the scattering in the human eye. In *Computer Graphics Forum*, volume 28, pages 183–192. Wiley Online Library, 2009.
- [Rik06] Littlefield Rik. Theory of the “no-parallax” point in panorama photography, 2006. <http://www.janrik.net/PanoPostings/NoParallaxPoint/TheoryOfTheNoParallaxPoint.pdf>.

- [RSSF02] Erik Reinhard, Michael Stark, Peter Shirley, and James Ferwerda. Photographic tone reproduction for digital images. *ACM transactions on graphics (TOG)*, 21:267–276, 2002.
- [SDHL11] B Steinert, Holger Dammertz, Johannes Hanika, and Hendrik PA Lensch. General spectral camera lens simulation. In *Computer Graphics Forum*, volume 30, pages 1643–1654. Wiley Online Library, 2011.
- [Sek04] Dean Sekulic. Efficient occlusion culling. *GPU Gems*, pages 487–503, 2004.
- [SO05] Warren J Smith and Others. *Modern Lens Design*, volume 2. McGraw-Hill New York, 2005.
- [Syr16] Syrp. Genie mini motion controller, 2016. <https://syrp.co.nz>.
- [Toc07] Mike Tocci. Quantifying veiling glare (zemax users’ knowledge base), 2007. <http://www.zemax.com/os/resources/learn/knowledgebase/quantifying-veiling-glare>.
- [Tow12] Justin Towell. A brief history of the most over-used special effect in video games: Lens flare, 2012. <http://www.gamesradar.com/brief-history-most-over-used-special-effect-videogames-lens-flare/>.
- [WBSS04] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions On Image Processing*, 13:600–612, 2004.
- [Woe09] Meredith Woerner. J.j.abrams admits star trek lens flares are “ridiculous” (interview), 2009. <http://io9.gizmodo.com/5230278/jj-abrams-admits-star-trek-lens-flares-are-ridiculous>.