DOI 10.5162/sensoren2016/5.1.1

Potential and Challenges of using Computer Graphics for the Simulation of Optical Measurement Systems

Max-Gerd Retzlaff[1,2] (retzlaff@kit.edu), Johannes Hanika[1], Jürgen Beyerer[2,3], Carsten Dachsbacher[1]
[1]Karlsruhe Institute of Technology (KIT), Institute for Visualization and Data Analysis (IVD), Computer Graphics Group, Karlsruhe, Germany
[2]Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB), Karlsruhe, Germany
[3]Karlsruhe Institute of Technology (KIT), Institute for Anthropomatics and Robotics, Vision and Fusion Laboratory (IES), Karlsruhe, Germany

Abstract

Physically-based image synthesis methods, a research direction in computer graphics (CG), are capable of simulating optical measuring systems in their entirety and thus constitute an interesting approach for the development, simulation, optimization, and validation of such systems. In addition, other CG methods, so called procedural modeling techniques, can be used to quickly generate large sets of virtual samples and scenes thereof that comprise the same variety as physical testing objects and real scenes (e.g., if digitized sample data is not available or difficult to acquire). Appropriate image synthesis (rendering) techniques result in a realistic image formation for the virtual scenes, considering light sources, material, complex lens systems, and sensor properties, and can be used to evaluate and improve complex measuring systems and automated optical inspection (AOI) systems independent of a physical realization. In this paper, we describe an image generation pipeline for the evaluation and optimization of measuring and AOI systems, we provide an overview over suitable image synthesis methods and their characteristics, and we discuss the challenges for the design and specification of a given measuring situation in order to allow for a reliable simulation and validation.

Keywords: computer graphics, procedural modeling, image synthesis, optical measurement

Introduction

Current physically-based image synthesis techniques constitute a major leap compared to previously used, mostly phenomenological approaches. The simulation of light transport is at the core of physically-based image synthesis methods and crucial to generate images that are on par with images made by physical image acquisition systems. Light transport simulation nowadays is almost exclusively computed using Monte Carlo (MC) or Markov Chain Monte Carlo (MCMC) methods, which can account for complex light-matter interactions and naturally handle spectral emission, absorption, and scattering behavior (measured or derived from models) described by geometric optics. (MC)MC methods can also comprise the simulation of complex lens systems to accurately compute the resulting irradiance onto a virtual sensor.

Essentially, all (MC)MC rendering methods compute an estimate of the light transport by sampling, that is, stochastically generating, paths on which light propagates from light sources to sensors; their main difference being the path sampling strategy. Until sufficient convergence, the variance of this estimation is apparent as noise in the images. Because of this, even simplistic realizations of these methods are versatile and, in principle, capable to achieve the desired, and required, results. However, their application is only practical when an (MC)MC method is used which is well-suited for a given scenario; otherwise the computation time can easily become prohibitively long, even in seemingly simple cases. For example, one would choose different methods for computing light transport for complex high-frequency light transport phenomena (recognizable by multiple glossy or specular reflection) than for highly scattering media.

Particular attention has to be paid to the simulation of complex lens systems which can increase the computation time by orders of magnitude when implemented naively. We discuss the use of a state-of-the-art approach to efficient rendering with realistic lenses in the context of measuring and AOI systems.

As indicated, many different rendering algorithms and sampling strategies exist in the realm of (MC)MC methods, and they all exhibit different performance and noise characteristics, which are strongly linked to the type of

light-matter interactions and the geometry configurations occurring in a scene. As such, it is not straightforward for the non-expert to select the appropriate method. For this purpose, we identify light interactions and phenomena constituting different challenges for the image synthesis, and discuss state-of-the-art algorithms, as, for example, path tracing, bi-directional path tracing (BDPT), and others.

Rendering, rasterization, and ray tracing

Hughes et al. [8] define the term rendering very concisely as referring to the process of integration of the light that arrives at each pixel of the image sensor inside a virtual camera in order to compute an image.

There are two major strategies for determining the color of an image pixel: rasterization and ray tracing.

Ray tracing, also sometimes referred to as ray casting, determines the visibility of surfaces by tracing rays of light from the virtual view point, that is, the viewer's eye or the image sensor, to the objects in the scene. The view point represents the center of origin and the image a window on an arbitrary view plane. For each pixel of the image a view ray is sent originating in the view point through the pixel into the scene in order to find an intersection with a surface. By recursive application of this ray casting one can compute complex light interactions and global illumination, that is, indirect illumination including, among others, reflections and shadows.

Rasterization, on the other hand, projects geometric primitives one by one onto the image window. A depth buffer, also called z-buffer, is utilized to determine the closest and thus visible primitive for each pixel.

Usually, the perspective projection is carried out in three steps. First, the projection transformation, expressed in homogeneous coordinates, is applied. Afterwards, the projective coordinates are dehomogenized by the normalization transformation, mapping the view frustum to the unit cube. The resulting device coordinates can then be mapped to image coordinates by discarding the depth coordinate, as it is done for orthographic projection.

All primitives can be processed in parallel using a single instruction multiple data (SIMD) approach and minor synchronization via the depth buffer. This allows for a very fast pipelined hardware implementation in the form of modern graphics processing units (GPUs).

Simplified, one could say that ray tracing starts with the pixels and then determines ray intersections with the scene geometry, while rasterization starts with the geometry, projecting it onto the image plane. The availability of modern GPUs makes rasterization feasible for interactive real-time application.

Synthetic images in the context of optical inspection

In [19] we describe the idea of using computer graphics methods to allow systematic and thorough evaluation of automated optical inspection (AOI) systems. Instead of using real objects and acquisition systems, computer graphics methods are used to create large virtual sets of samples of test objects and to simulate image acquisition setups. We use procedural modeling techniques to generate virtual objects with varying appearance and properties, mimicking real objects and sample sets. Physical simulation of rigid bodies is deployed to simulate the placement of virtual objects, and using physically-based rendering techniques we create synthetic images. These are used as input to an AOI system instead of physically acquired images. This enables the development, optimization, and evaluation of the image processing and classification steps of an AOI system independently of a physical realization.

We demonstrated this approach for shards of glass, as sorting glass is one challenging practical application for AOI. In this paper, we focus on the aspects of image synthesis.

Real-time rendering of shard distributions

Our implementation includes real-time rendering of shard distributions by hardware rasterization on a GPU using OpenGL 4.2 [21].

As glass is an optically semi-transparent material, a method of transparency rendering is necessary. OpenGL itself only provides alpha blending that can be used to render transparent materials. But rendering transparency using alpha blending implies rendering the objects in sorted order. As sorting for every rendered frame is impractical and even not always possible (e.g., for mutually overlapping objects) order-independent transparency rendering (OIT) techniques have been invented.

Our previous implementation as presented in [19] used depth peeling (introduced by Everitt [2]), a robust hardware-accelerated OIT method, but meanwhile we replaced this with an implementation of a more powerful OIT technique making use of per-pixel concurrent linked lists (introduced by Yang et al. [24]). OIT rendering using depth peeling can store only a quite limited number of material interactions per image pixel requiring multiple rendering passes in case of many interactions. Per-pixel linked list OIT, on the other hand, allows storing a high number of interactions in a single rendering pass, and the realistic rendering of the breaking edges of glass shards may require a high number of interactions as the edges can be quite rough.

As our previous publications [19] and [20] focus on the generation of realistic virtual objects and scenes based on measured data as well

as the methodical procedure of using synthetic images for the evaluation of real AOI systems, we had used simplified models of the illumination and image acquisition for the image synthesis, e.g., ideal sensors and simplified optical image formation.
Most importantly, we have used RGB rendering. The synthesis was therefore reduced to only three values describing red, green, and blue norm stimuli, and could not reproduce spectral effects such as dispersion.

Replicating real image acquisition systems and spectral rendering

Recently, we have enhanced our real-time method to support spectral rendering and to replicate more physical aspects of real image acquisition systems, that is, we now support the simulation of real light sources and sensor responses of real image sensors.
Light emission and transport we now describe as full spectra. That is, we use the spectral data of real, measured light sources. The light interaction is computed using measured absorption spectra (of course, for other use cases reflection spectra are equally possible). The resulting spectral power distribution is multiplied with a color matching or sensor sensitivity function, as depicted in Fig. 2. Integration across the spectrum leads to the intensity values of each sensor sensitivity function, usually an RGB triplet in the RGB space of the simulated sensor that can be displayed.
In case of the CIE color matching functions [22] XYZ coordinates are computed that describe the color perception of a human being as standardized by the CIE 1931 standard colorimetric observer. The XYZ images can be converted to an RGB working space as, for example, the sRGB color space [9] and displayed on a calibrated screen. It thus can serve as a reference image for an expert user designing a AOI setup as it corresponds to the color impression one would perceive in the simulated viewing conditions, cf. Fig. 4.
Simulation of chromatic adaptation can account for the difference in white reference of

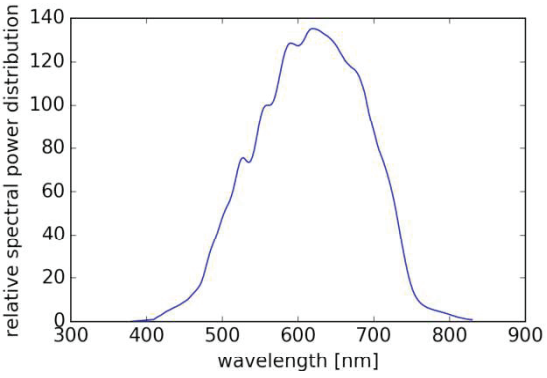

Fig. 1: Emission spectrum of a tungsten halogen lamp (Model 3900 by Illumination Technologies, Inc.).

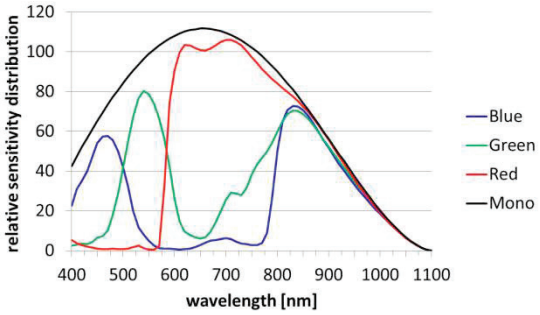

Fig. 2: Raw sensor response of the ELiiXA UC4/UC8 line scan camera by e2v.

the simulated light emission and of the computer screen that is used for viewing.

Light sources with bright spectral lines

While light sources such as incandescent light bulbs or halogen lamps have quite smooth spectra, see Fig. 1, the spectra of other light sources as, for example, fluorescent lamps exhibit a number of narrow bright peaks, as depicted in Fig. 3. Such a lighting condition in combination with certain sensors can result in images in which actual spectral differences cannot be perceived or easily detected anymore. This can be simulated with spectral rendering, and helps to design an acquisition setup that does not exhibit such problems by choosing a suitable combination of sensor and illumination, see Fig. 5 and compare to Fig. 4.
Of course, spectral rendering is especially important in the context of color filters that limit the light spectrum to a narrow spectral range combined with lighting by a non-uniform illumination.
For our real-time implementation we use a binning approach to spectral rendering. That is, the full spectrum of the visible light is quantized into bins of a certain width, thus the spectra are approximated by a step function. The spectra can either be approximated by equal width bins, which is a sufficient approximation for smooth spectra, or it can be adaptive, as necessary in the case of illumination with light depicting narrow bright peaks.

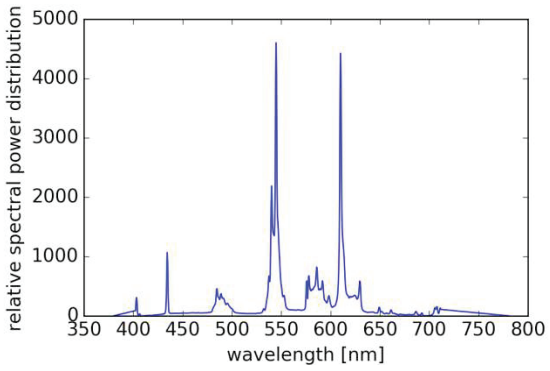

Fig. 3: Emission spectrum of a fluorescent ceiling light.

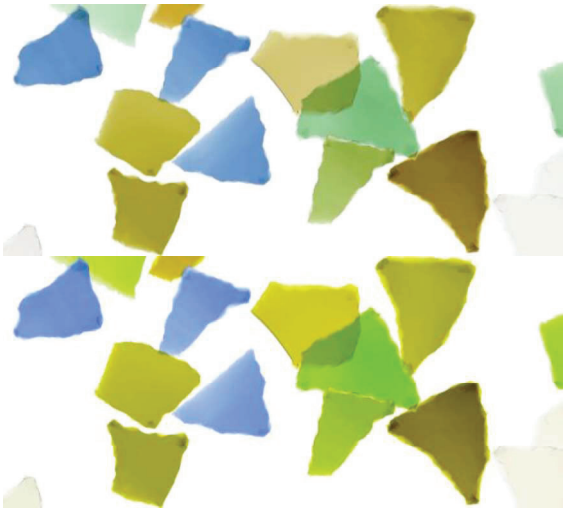

Fig. 4: Images rendered in real-time using the color matching functions of the CIE 1931 standard colorimetric observer. Top: CIE Illuminant D65, bottom: fluorescent illumination of Fig. 3.

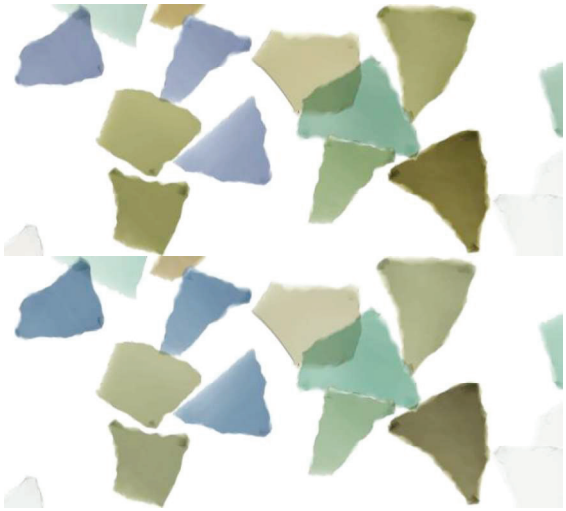

Fig. 5: Images rendered in real-time using the ELiiXA sensor sensitivity function depicted in Fig. 2. Top and bottom images illuminated as in Fig. 4.

Physically-based rendering

While it is possible to add support for more complex light interaction phenomena as, for example, refraction and dispersion – and has in fact been done in form of a proof of concept implementation for light refraction –, one has to accept that rasterization has its limits and at one point a ray tracing approach begins to be more feasible, and even more efficient.
Support of refraction can be added to a rasterization approach by ray marching but, obviously, it is much more straight forward to add it to a ray tracer. Dispersion constitutes a similar case. But the aforementioned point is surely exceeded when global illumination or simulation of real lens systems including monochromatic and chromatic aberration is asked for.

Generation of synthetic glass shards

Retzlaff et al. [19] describe a procedural model for glass shards based on an algorithm described by Martinet et al. [17]. We modified the algorithm to generate plausible shards with a certain smoothing, as waste glass is repeatedly relocated and this leads to rounded edges of the glass shards. In addition to this, our carving volume is smooth to allow fast intersection, while we achieve plausible breaking edges by displacement mapping of the intersection surfaces using hardware-accelerated tessellation shaders that were introduced in OpenGL 4.0.
These modifications make a shard generation in negligible computation time possible by adding surface detail in real-time. But there is also a downside. In case of considerable displacement, the polygonal mesh might get self-intersections or holes at the transition between intersection surface and original shard surface. Small irregularities can be fixed easily but bigger defects would require a remeshing of the polygon mesh, slowing down the generation considerably. Such erroneous meshes describe surfaces that are not physically possible, and thus present problems for physically-based rendering methods. As a result, the law of conservation of energy might get violated at such irregularities.
Therefore, we vouched for a new method of generating shards of broken glass, dropping the requirement of real-time generation and instead introducing a pool of precomputed shard meshes.
Most existing fracturing methods in the field of computer graphics are based on Voronoi partitions or tessellations, but many implementations generate only undetailed, flat intersection surfaces that are also not well-tessellated and thus not suitable for surface perturbation without remeshing. An example for this is the Cell Fracture implementation of the Blender 3D modeling software by the Blender Foundation.
One notable exception is the method introduced by Korndörfer [16]. It is also based on Voronoi partitioning but uses a 3D voxel grid and emphasizes on achieving a regular quad meshing of the boundary surface. The Voronoi regions are not based on the Euclidean distance but instead on the length of paths in a graph defined by the voxels. The shape of the regions can be controlled by weights of the graph edges. The method already supports random noise edge weights in order to generate irregular, i.e., plausible, surface structures, making it a suitable candidate for our task of generating of virtual glass shards.

Comparison of the suitability of Monte Carlo rendering techniques for AOI

In all their diversity, the (MC)MC rendering methods share the concept of stochastically

Fig. 6: A synthetic image of procedurally generated virtual glass shards generated by our Monte Carlo rendering framework.

creating paths, connecting the sensor to the lights. In this section, we discuss different rendering algorithms and sampling strategies that exist in the realm of (MC)MC methods.

Path tracing: The paths are starting from the camera, directions are sampled at every interaction, optionally, with next event estimation (NEE, Kajiya [13]). Path tracing is easy to implement and suitable if there are no caustics from small light sources and if direct connections to the light sources are possible.

Light tracing: Paths are starting from the light sources, direction sampling as with path tracing, again optionally with next event estimation (here: connection to camera). But NEE is problematic in context of specular surfaces and lens models. It works only well if light sources only emit to the surfaces visible to the camera.

Bi-directional path tracing (BDPT): The subpaths are starting from both ends, camera and light sources, with deterministic connections between nodes. It requires variance reduction via multiple importance sampling (MIS) to be practical, which unfortunately makes this method non-trivial to implement. BDPT works well in scenes where the deterministic connections are not blocked. This should usually be the case for AOI settings as we want to illuminate objects which are visible to both camera and light source.

Many-light methods (variants of BDPT): Create light subpaths and treat nodes as virtual point lights. This approach works very well for mostly diffuse scenes and is easy to implement. On the downside, it has problems with glossy surfaces and is even impossible to use with specular surfaces, cf. Dachsbacher et al. [3].

Photon mapping: Create light subpaths, record node locations (locations where energy impinges on surfaces), and use a density estimation to estimate energy per surface area. It works well with diffuse surfaces and caustics. For glossy surfaces, it (gracefully) degrades to path tracing. Photon mapping can be made robust and has the advantage of producing images with low noise levels, but the density estimation also causes a bias (systematic error), cf. Hachisuka and Jensen [5].

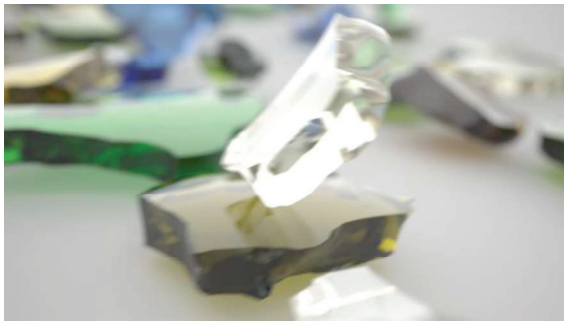

Fig. 7: A close-up view of the scene also displayed in Fig. 6 from a different angle.

More extensions and variants to bi-directional methods exist, as, e.g., Georgiev et al. [4].

Metropolis light transport (MLT): These methods use the Metropolis-Hastings (MH) algorithm (introduced by Hastings [7]) to sample the space of all possible light transport paths. Numerous variants exist, e.g., sampling in primary space (Kelemen [11]) or in path space (Veach [23]), as well as modern extensions such as Energy Distribution Raytracing, Manifold Exploration (Jakob [10]), Half-Vector space transport (Kaplanyan et al. [14]), and gradient domain methods (Kettunen et al. [15]). MLT methods have in common that they are very powerful in exploring difficult light transport phenomena (e.g., caustics). However, they have to be initialized repeatedly with independent MC samplers (e.g., BDPT) and thus rely on those to detect actual occurrences of said light phenomena. In image rendering, this results in images where individual components are rendered with little noise but all occurrences of the light phenomena are only found over time. Also, MLT methods share the property that they are difficult to implement (an exception is Kelemen [11]), and difficult to control.

Bringing it all together

Fig. 6 shows a scene of the shards generated with the method of Korndörfer [16] using our

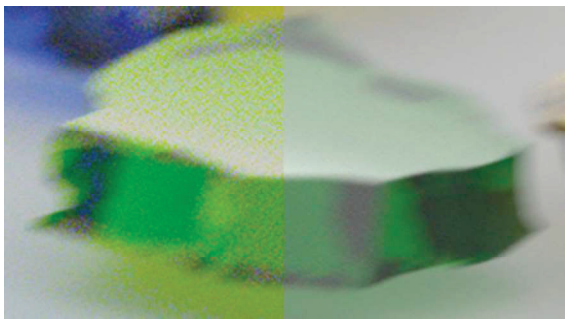

Fig. 8: Close-up view of a synthetic shard. Left: fluorescent illumination with bright spectral lines, as depicted in Fig. 3, right: CIE Standard Illuminant D65. Both images are rendered with 2048 samples per pixel.

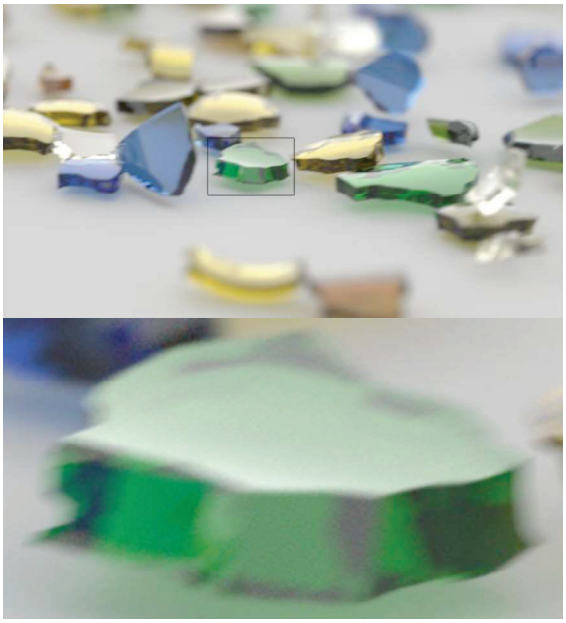

Fig. 9: Synthetic image and enlarged section (bottom) rendered using a thin lens model exhibiting a narrow depth of field (DOF).

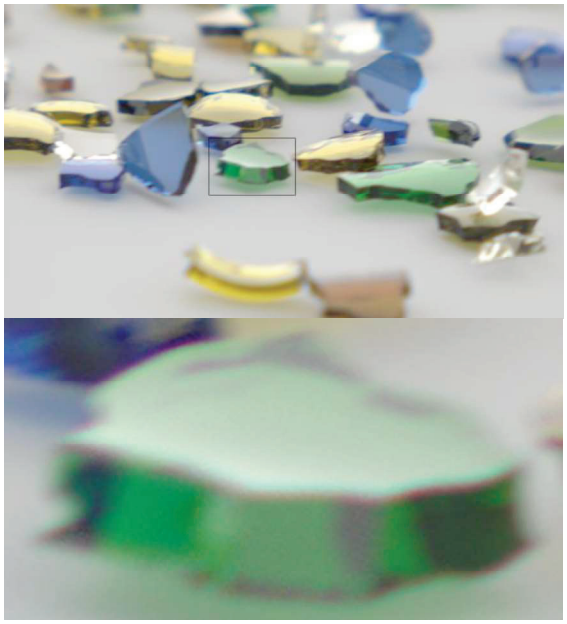

Fig. 10: Same scene as in Fig. 9 but using a simple Lensbaby-like lens, exhibiting a less narrow DOF but strong chromatic aberration (color fringing).

existing Monte Carlo ray tracing framework supporting global illumination, that is, a physically-based simulation of light transport, which inherently accounts for phenomena such as reflections and shadows. We have chosen path-tracing without NEE for our task as the AOI setup exhibits a very large light source.
As the ray tracing framework does not use spectral binning but instead supports full spectral rendering with Monte Carlo spectral sampling, it can simulate dispersion. Fig. 7 shows a close-up view of the same scene as in Fig. 6 (from a slightly different angle), exhibiting light refraction and chromatic dispersion.
It is interesting to note that the spectra of the illumination and interacting surfaces might also affect the rendering time or image quality. Uniform sampling of the emission spectrum leads to noisy images in case of spectra with bright narrow peaks compared to images using a more homogenous spectral illumination, as samples fall equally on every part of the spectrum. This is illustrated in Fig. 8.
Importance sampling that results in a higher sampling rate in regions with higher radiance while still being an unbiased estimation can leverage this problem without increasing the sampling count (and thereby increasing the rendering time).

Realistic lenses

In interactive image synthesis, often simple perspective projection is used that corresponds to the projection of a pinhole camera. Rendering systems also often use simplified approaches as the pinhole camera or the thin lens model. While it is possible to just include even a complex lens into the virtual scene and trace rays through it, this leads to very inefficient rendering, as, for straight forward ray tracing, 95% of all ray samples, or more, might not leave a real world lens system and enter the actual scene [6].
We use our method and implementation presented in Hanika and Dachsbacher [6] to support efficient rendering using a wide range of real world lenses with only negligible overhead compared to rendering with the thin lens model. Such real world lenses can consist of dozens of lens elements and an aperture, as, for example, a Double-Gauss lens as depicted in Fig. 11. Still, this approach can accurately compute the lens image and thus be used to simulate flaws and shortcomings of real lens systems.
Fig. 9 and Fig. 10 show a group of images of the same scene but using different lenses, the thin lens model and a simple Lensbaby-like lens (achromatic doublet), respectively. Both lenses have got different depths of field, and the image by the Lensbaby-like lens exhibits strong chromatic aberration, commonly referred to as color fringing.
To conclude this section, we present an overview of renderings with different lenses and color matching and sensor sensitivity functions,

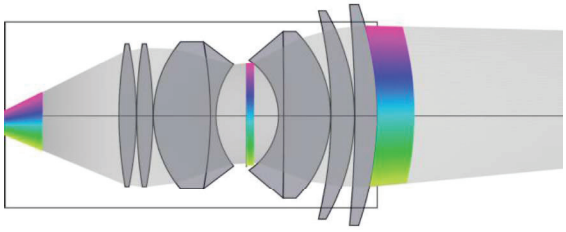

Fig. 11: Double-Gauss lens by Pierre Angénieux [1]

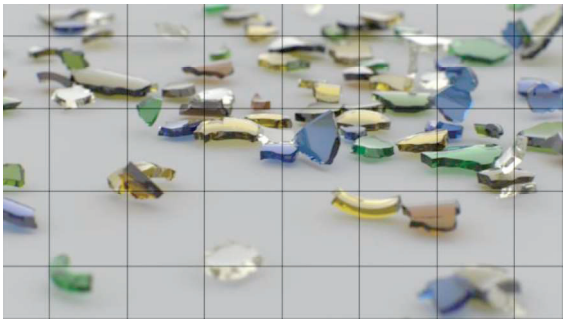

Fig. 12: Synthetic image using the simple thin lens model as perceived by the CIE 1931 standard colorimetric observer, CIE D65 hemispherical illumination.

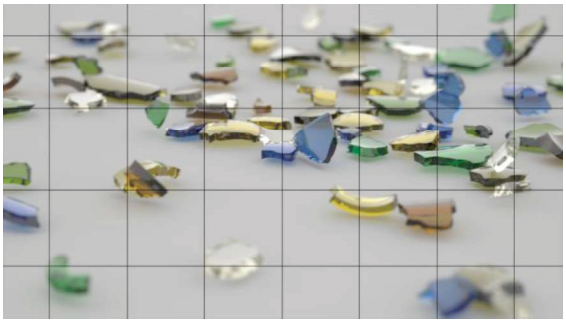

Fig. 13: Same as in Fig. 12 but using the Angénieux Double-Gauss lens [1], also depicted in Fig. 11. The black grid is overlaid to make the difference in distortion easily recognizable.

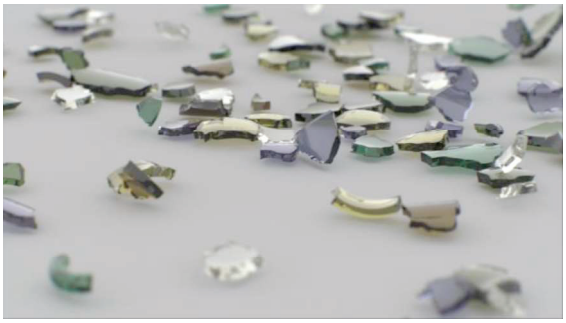

Fig. 14: Same as in Fig. 13 but detected by the ELiiXA sensor, depicted in Fig. 2.

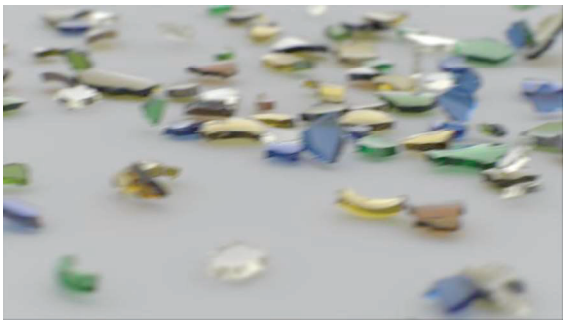

Fig. 15: Same as Fig. 13 but with motion blur.

respectively. Fig. 12 to Fig. 15 show the scene from a flat angle, while Fig. 16 to Fig. 19 show a top view of the same scene resembling the arrangement of the real image acquisition system. Actually, in the real system the shards always appear in a strong transmitted light as in Fig. 18 and Fig. 19.

As we generate the virtual scenes using physical simulation of gravity and collisions of rigid bodies, we can easily generate consecutive scenes separated by small time differences, and thus also render images with motion blur of the shards sliding along a sloping surface, as to be seen in Fig. 15.

Conclusion and future work

In summary, we described the entire image synthesis pipeline including all relevant components as well as their efficient and accurate implementation with regard to the simulation and validation of optical measuring systems.

The EMVA 1288 Standard for Machine Vision [12] describes a model for sensor noise. It is dissected as a combination of photon noise or, more generally, shot noise that can be described as the square-root of the signal, of dark noise governed by the dark current of thermally generated electrons, and of sensor readout noise. This model can be used to simulate the sensor noise of real sensors.

References

[1] Pierre Angénieux. Large Aperture Six Component Optical Objective. U.S. Patent 2,701,982, issued Feb. 15, 1955.

[2] Everitt Cass. Interactive order-independent transparency. Technical report, NVIDIA Corporation, May 2001.

[3] Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, and Jan Novák. Scalable Realistic Rendering with Many-Light Methods. Computer Graphics Forum, 33(1):88–104, 2014; doi: 10.1111/cgf.12256

[4] Iliyan Georgiev, Jaroslav Křivánek, Tomáš Davidovič, and Philipp Slusallek. Light transport simulation with vertex connection and merging. ACM Trans. Graph. (TOG) – Proc. of SIGGRAPH Asia 2012, 31(6):192:1–10, Nov. 2012; doi: 10.1145/2366145.2366211

[5] Toshiya Hachisuka and Henrik Wann Jensen. Stochastic progressive photon mapping. ACM Trans. Graph. (TOG) – Proc. of SIGGRAPH Asia 2009, 28(5):141:1–8, Dec. 2009; doi: 10.1145/1618452.1618487

[6] Johannes Hanika, and Carsten Dachsbacher. Efficient Monte Carlo rendering with realistic lenses. Computer Graphics Forum (Proc. of Eurographics), 33(2):323–332, April 2014; doi: 10.1111/cgf.12301

[7] W. Keith Hastings. Monte Carlo sampling methods using Markov chains and their applications. Biometrika 57(1): 97–109, April 1970; doi: 10.1093/biomet/57.1.97

[8] John F. Hughes, Andries van Dam, James D. Foley, Steven K. Feiner. Computer Graphics: Principles and Practice, 3rd edition. Addison-Wesley, 2015; ISBN: 978-0-321-39952-6

[9] IEC (International Electrotechnical Commission). Multimedia systems and equipment – Colour measurement and management – Part 2-1: Colour management – Default RGB colour space – sRGB. IEC 61966-2-1:1999; ISBN: 2-

8318-4989-6; ICS codes: 33.160.60, 37.080 – TC 100
[10] Wenzel Jakob and Steve Marschner. Manifold exploration: a Markov Chain Monte Carlo technique for rendering scenes with difficult specular transport. ACM Trans. Graph. (TOG) – Proc. of SIGGRAPH 2012, 31(4):58:1–13, July 2012; doi: 10.1145/2185520.2185554
[11] Csaba Kelemen, László Szirmay-Kalos, György Antal, and Ferenc Csonka. A Simple and Robust Mutation Strategy for the Metropolis Light Transport Algorithm. Computer Graphics Forum (Proc. of Eurographics), 21(3):531–540, Sep. 2002; doi: 10.1111/1467-8659.t01-1-00703
[12] Bernd Jähne. EMVA 1288 Standard for Machine Vision. Optik & Photonik, 5(1):53–54, 2010; doi: 10.1002/opph.201190082
[13] James T. Kajiya. The rendering equation. ACM SIGGRAPH Computer Graphics, 20(4):143–150, Aug. 1986; doi: 10.1145/15886.15902
[14] Anton S. Kaplanyan, Johannes Hanika, and Carsten Dachsbacher. The Natural-Constraint Representation of the Path Space for Efficient Light Transport Simulation. ACM Trans. Graph. (TOG) – Proc. of SIGGRAPH 2014), 33(4):102:1–13, July 2014; doi: 10.1145/2601097.2601108
[15] Markus Kettunen, Marco Manzi, Miika Aittala, Jaakko Lehtinen, Frédo Durand, and Matthias Zwicker. Gradient-domain path tracing. ACM Trans. Graph. (TOG) – Proc. of SIGGRAPH 2012, 34(4):123:1–13, Aug. 2015; doi: 10.1145/2766997
[16] Johann Korndörfer. Interaktive Zerlegung von Festkörpern in Bruchstücke. Diploma thesis. Universität Karlsruhe (TH), 2011.
[17] Aurélien Martinet, Eric Galin, Brett Desbenoit, and Samir Hakkouche. Procedural modeling of cracks and fractures. Proc. of the Shape Modelling International (Short Paper), pages 346–349, 2004; doi: 10.1109/SMI.2004.1314524
[18] Johannes Meng, Marios Papas, Ralf Habel, Carsten Dachsbacher, Steve Marschner, Markus Gross, and Wojciech Jarosz. Multi-Scale Modeling and Rendering of Granular Materials. ACM Trans. Graph. (TOG) – Proc. of SIGGRAPH 2015, 34(4):49:1–13, Aug. 2015; doi: 10.1145/2766949
[19] Max-Gerd Retzlaff, Josua Stabenow, Jürgen Beyerer, and Carsten Dachsbacher. Synthesizing images using parameterized models for automated optical inspection (AOI). tm – Technisches Messen, 82(5): 251– 61, April 2015. De Gruyter; doi: 10.1515/teme-2014-0041
[20] Max-Gerd Retzlaff, Josua Stabenow, and Carsten Dachsbacher. Synthetic image acquisition and procedural modeling for automated optical inspection (AOI) systems. Forum Bildverarbeitung 2014, pages 47–59, KIT Scientific Publishing, 2014; doi: 10.5445/KSP/1000043608
[21] Mark Segal and Kurt Akeley. The OpenGL Graphics System: A Specification (Ver. 4.2 (Core Profile) – April 27, 2012). Khronos Group, April 2012.
[22] Thomas Smith and John Guild. The C.I.E. Colorimetric Standards and Their Use. Transactions of the Optical Society 33(3):73–134, 1931–32; doi: 10.1088/1475-4878/33/3/301
[23] Eric Veach. Robust Monte Carlo Methods for Light Transport Simulation. Ph.D. dissertation. Stanford University, 1998; ISBN: 0-591-90780-1
[24] Jason C. Yang, Justin Hensley, Holger Grün, and Nicolas Thibieroz. Real-Time Concurrent Linked List construction on the GPU. Proc. of the 21st Eurographics Symposium on Rendering (EGSR'10), pages 1297–1304, 2010; doi: 10.1111/j.1467-8659.2010.01725

Fig. 16: Angénieux Double-Gauss lens [1], CIE D65 hemispherical illumination, CIE standard colorimetric observer.

Fig. 17: Same as Fig. 16 but detected by the ELiiXA sensor, depicted in Fig. 2.

Fig. 18: Same as Fig. 16 but with the fluorescent ceiling light of Fig. 3 as strong transmitted light as well as hemispherical illumination.

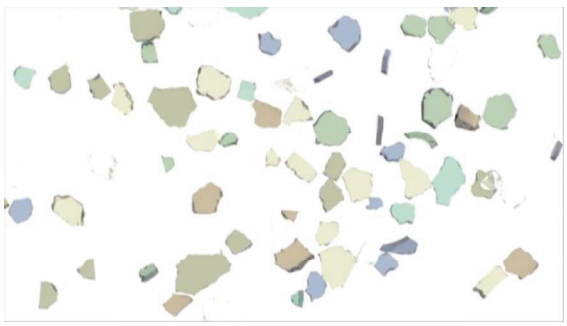

Fig. 19: Same as Fig. 18 but with the ELiiXA sensor, depicted in Fig. 2.