Vision, Modeling, and Visualization (2014)
J. Bender, A. Kuijper, T. von Landesberger, H. Theisel and P. Urban (Eds.)

Progressive Spectral Ray Differentials

Oskar Elek[1,2] Pablo Bauszat[3,1] Tobias Ritschel[1,2] Marcus Magnor[3] Hans-Peter Seidel[1,2]

MPI Informatik[1] MMCI Saarland University[2] TU Braunschweig[3]

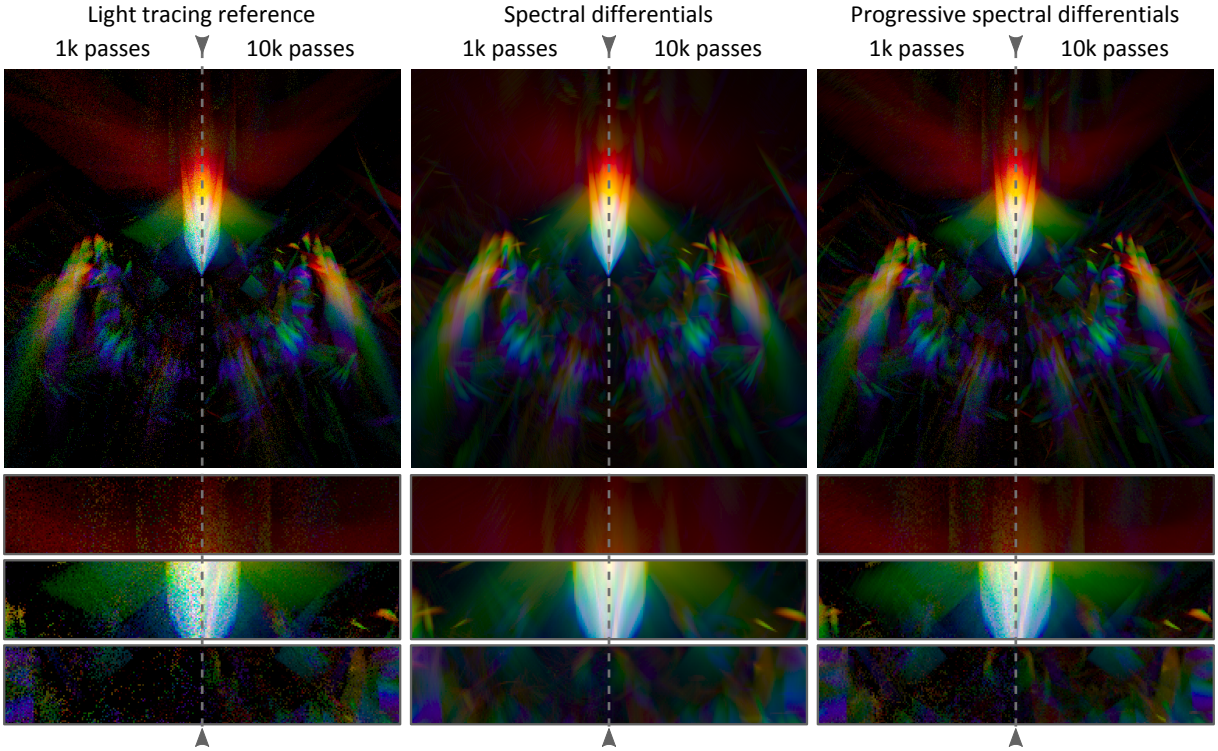

Figure 1: *Complex caustic resulting from chromatic dispersion on a brilliant-cut gemstone with a very wide refractive index range (from 2.2 for the violet end to 1.2 for the red end of the spectrum). We use a standard unbiased MC solution (light tracing) with stochastic spectral sampling as a reference, which produces very noisy results and does not fully converge even after a high number of passes. Spectral ray differentials lead to a smooth solution and an order of magnitude faster convergence, the solution is however biased, which is manifested as blur of the caustic features (cf. insets below). Our proposed progressive spectral differentials converge almost as fast and lead to an unbiased solution in the limit.*

Abstract

Light travelling though refractive objects can lead to beautiful colourful illumination patterns resulting from dispersion on the object interfaces. While this can be accurately simulated by stochastic Monte-Carlo methods, their application is costly and leads to significant chromatic noise. This is greatly improved by applying spectral ray differentials, however, at the cost of introducing bias into the solution. We propose progressive spectral ray differentials, adapting concepts from other progressive Monte-Carlo methods. Our approach takes full advantage of the variance-reduction properties of spectral ray differentials but progressively converges to the correct, unbiased solution in the limit.

Keywords: dispersion rendering, spectral ray differentials, progressive Monte-Carlo methods

1. Introduction

Interaction of light with a refractive material interface (e. g., glass) causes a change of its initial direction, leading to interesting phenomena such as caustics. If the material is dispersive this direction is further wavelength-dependent, causing chromatic rainbow-like phenomena. These are often visually attractive, but just as often undesired as well – especially in optical systems where they cause chromatic aberrations, degrading their performance. It is therefore important to understand and accurately simulate these phenomena.

Stochastic Monte-Carlo methods can be trivially extended to support dispersion, albeit dispersive caustics as the most representative phenomenon from this category are handled better by the forward-tracing methods [Arv86, Jen01], than by the backward-tracing ones [Kaj86, Tho86]. Regardless, spectral rendering introduces an additional dimension (the spectral domain) into the rendering equation. The prevalent strategies to sample the spectrum are stochastic sampling, which leads to chromatic noise, and regular sampling, which causes aliasing artifacts [EBR*14, WND*14]. In general this also introduces computational overhead, as more samples are required for a solution to converge. Methods that cache illumination, such as photon mapping, also require more memory to store the spectral information [Lai07].

To alleviate this issue the concept of spectral ray differentials (SRD hereinafter) has recently been proposed [EBR*14]. SRD extend the widely-adopted framework of ray differentials introduced by Igehy [Ige99], which has since been generalized to paths and the distribution phenomena associated with them [SW01, SFES07, FD09]. All ray differentials, including the spectral variant, trace partial derivatives of a ray within a certain domain. A first-order approximation then uses this information for a better reconstruction of a given phenomenon (dispersion in case of SRD). However, applying the first-order approximation introduces bias into the solution; this is precisely the problem which we aim to resolve.

Reducing bias in global illumination solvers has naturally received significant attention. Arguably the most prominent example is photon mapping [Jen01], which can produce an unbiased solution but only if an infinite number of photons be stored. To resolve this limitation the progressive photon mapping was proposed by Hachisuka et al. [HOJ08, HJ09], and later improved to remove the need of storing and maintaining local photon statistics [KZ11]. Furthermore, Jarosz et al. [JNT*11] extended this framework to the volumetric photon beam estimation. Photon mapping suffers from bias because its density estimation takes place in a region of a certain finite size around the point of interest. The central idea of progressive methods is to gradually decrease the size of this region so that the bias diminishes with increasing number of traced photons. An added value that progressiveness can provide is an interactive manipulation and feedback [LSK*07, DKL10], allowing, for instance, fast changes to the scene parametrization.

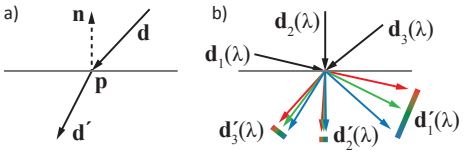

Figure 2: (a) *Refraction for a single monochromatic ray and* (b) λ*-dependent dispersive refraction of three spectral rays.*

We propose progressive spectral ray differentials as an effort to remove the bias caused by the first-order approximation in SRD. The basic idea is, in analogy with progressive photon mapping, to decrease the reconstruction scale of SRD with increasing number of illumination samples that are traced though the scene. Specifically, we

- combine the probabilistic progressive photon mapping framework [KZ11, JNT*11] with spectral ray differentials [EBR*14] (Sec. 3), and
- demonstrate the ability of this approach to converge to a corresponding unbiased Monte-Carlo solution, with specific focus on dispersive caustics (Sec. 4).

2. Overview

In this section we briefly summarize the physics of dispersion and rephrase the concepts of spectral ray differentials and progressive rendering methods to provide background to our work.

Refraction and dispersion A light ray hitting an interface of a dielectric object is partially refracted (according to Fresnel's law). This changes its direction $\mathbf{d}$ and additionally causes its spectral components to spread (disperse) in case the index of refraction (IOR) of the object depends on the wavelength of the light in vacuum λ. According to Snell's law the refracted direction $\mathbf{d}'$ will be

$$\mathbf{d}' = \eta\,\mathbf{d} - \mu\mathbf{n}, \quad \text{where}$$

$$\mu = \eta\,(\mathbf{d}\cdot\mathbf{n}) + \sqrt{1-\eta^2+\eta^2\,(\mathbf{d}\cdot\mathbf{n})^2}$$

Here, η is the ratio of the λ-dependent refractive indices of the original and the entered medium, and $\mathbf{n}$ is the interface normal. Fig. 2 illustrates this relationship.

Spectral ray differentials Given a spectral ray $\mathbf{R}(\lambda) = (\mathbf{p},\mathbf{d})$ defined by its position $\mathbf{p}$ and direction $\mathbf{d}$ its spectral differential is a pair of partial derivatives $\frac{\partial\mathbf{p}}{\partial\lambda}$ and $\frac{\partial\mathbf{d}}{\partial\lambda}$. This information is traced with the ray itself and updated on every transport, reflection or refraction event (see Fig. 3). The classic ray differentials [Ige99] can be used for the transport and reflection directly, since these interactions are λ-independent. For refraction, differentiating Snell's law yields the following

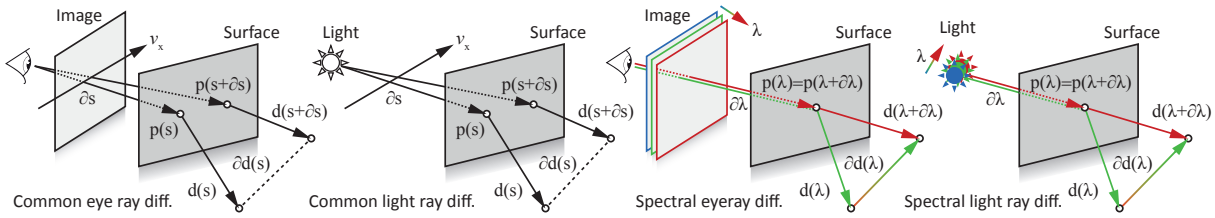

Figure 3: *Comparison of the classic* (left half) *and spectral* (right half) *ray differentials. Classic differentials define the change of ray in respect to a spatial coordinate (which can relate to an initial position on the image plane or a light source), while spectral differentials in respect to a position in the spectrum (in other words, wavelength). Reproduced with permission from [EBR*14].*

expression:

$$\frac{\partial \mathbf{d}'}{\partial \lambda} = \frac{\partial \eta}{\partial \lambda}\mathbf{d} + \eta\frac{\partial \mathbf{d}}{\partial \lambda} - \frac{\partial \mu}{\partial \lambda}\mathbf{n} - \mu\frac{\partial \mathbf{n}}{\partial \lambda} \tag{1}$$

The derivative $\frac{\partial \mu}{\partial \lambda}$ is defined in Appendix A for clarity. The differential of η depends on the model used to derive the λ-dependent material IOR; the expressions for several standard models are covered by [EBR*14]. On the other hand the normal differential $\frac{\partial \mathbf{n}}{\partial \lambda}$ relates only to the surface geometry and therefore is sufficiently described by Igehy [Ige99].

Contrary to the directional differential, the positional differential $\frac{\partial \mathbf{p}}{\partial \lambda}$ is not influenced by refraction, since, obviously, the interaction does not displace the ray position. As described by the classic ray differentials the differential of position changes only during transport (i. e., when the ray travels through empty space).

Progressive rendering On the high level the premise of all progressive methods is to rapidly produce a coarse solution and then progressively refine it until a certain quality is met. Progressiveness is a natural part of most unbiased rendering algorithms (e. g., in path tracing it amounts to simply tracing more samples). In biased algorithms, however, additional considerations are necessary to ensure that the bias decreases over time and the solution converges correctly.

We opted to employ the probabilistic framework of Knaus and Zwicker [KZ11] and Jarosz et al. [JNT*11] due to its simplicity and general applicability. Here it is assumed that the rendering algorithm proceeds in passes; each pass corresponds to tracing a fixed-sized, relatively small batch of samples. These are then continually averaged, producing a converging solution.

Denoting the error of pass i as ϵ_i, the average error after N passes is $\overline{\epsilon}_N = \frac{1}{N}\sum_{i=1}^{N}\epsilon_i$. Since each pass is assumed to be independent, the errors ϵ_i can be interpreted as realizations of a random variable $\overline{\epsilon}_N$. The variance (noise) and expected value (bias) of $\overline{\epsilon}_N$ can then be expressed as

$$\mathrm{Var}[\overline{\epsilon}_N] = \frac{1}{N^2}\sum_{i=1}^{N}\mathrm{Var}[\epsilon_i] \quad \text{and} \quad \mathrm{E}[\overline{\epsilon}_N] = \frac{1}{N}\sum_{i=1}^{N}\mathrm{E}[\epsilon_i].$$

An algorithm converges to a stable solution if $\mathrm{Var}[\overline{\epsilon}_N] \to 0$ as $N \to \infty$. The main difference between unbiased and biased methods is that the average error $\mathrm{E}[\overline{\epsilon}_N] = 0$ in unbiased methods at any point during the solution computation; for biased methods this does not hold. However the bias in approaches such as progressive photon mapping is *consistent*, which means that also $\mathrm{E}[\overline{\epsilon}_N] \to 0$ as N approaches infinity and therefore the result converges to the true unbiased solution. This is achieved by decreasing the size of the density estimation region in each pass by a small factor; this of course increases the variance of each pass $\mathrm{Var}[\epsilon_i]$, however the *overall* variance of the solution $\mathrm{Var}[\overline{\epsilon}_N]$ still decreases.

Despite unbiased approaches being mathematically superior to the biased ones, often in practice the latter are preferred. The reason is their usually better robustness, meaning that the *actual* error of the solution $\overline{\epsilon}_N$ tends to be smaller in biased algorithms, albeit its expected value $\mathrm{E}[\overline{\epsilon}_N]$ is non-zero. As we will demonstrate, this is also the case in dispersion rendering and the primary motivation to use spectral ray differentials in a progressive manner.

3. Progressive Spectral Ray Differentials

The application of spectral ray differentials consists of two stages: tracing and reconstruction. During the tracing stage the differential, a pair of 3D vectors $(\frac{\partial \mathbf{p}}{\partial \lambda}, \frac{\partial \mathbf{d}}{\partial \lambda})$ associated with the ray $\mathbf{R}(\lambda) = (\mathbf{p}, \mathbf{d})$, is maintained as summarized in Sec. 2. The reconstruction step then uses the differential to make predictions about the spatial ray changes in dependence on changes in its spectral coordinate λ. The spatial displacement is obtained by applying the first-order approximation:

$$\mathbf{R}(\lambda + \Delta\lambda) - \mathbf{R}(\lambda) \approx \Delta\lambda\frac{\partial \mathbf{R}(\lambda)}{\partial \lambda} =: \Delta\mathbf{R} \tag{2}$$

This defines a linear 1D footprint of the sample, with the spatial extent between $(\mathbf{p} - \Delta\mathbf{p})$ and $(\mathbf{p} + \Delta\mathbf{p})$. Splatting of line primitives is one way to reconstruct such samples [EBR*14].

Note that $\Delta\lambda$ acts as a user-controlled parameter here. Its increase will enhance the variance reduction of SRD but also increase bias, since this scales up the filter footprint. And, vice-versa, if $\Delta\lambda \to 0$ then the sample footprint shrinks, down to unbiased point estimates at $\Delta\lambda = 0$. Performing this

shrinking automatically is obviously the key ingredient to enable progressive rendering with spectral ray differentials.

Estimate Regardless of the specific reconstruction procedure, the radiance L in the direction $\mathbf{d}_x$ within the sample footprint can be estimated as

$$L(\mathbf{p}_x, \mathbf{d}_x, \Delta\lambda) \approx k_{\Delta\lambda}(u) \cdot \gamma, \tag{3}$$

where $\mathbf{p}_x$ lies on the line between $(\mathbf{p} - \Delta\mathbf{p})$ and $(\mathbf{p} + \Delta\mathbf{p})$. Eq. 3 expresses the reconstructed radiance as a kernel estimate from the sample contribution γ (which combines the flux carried by the sample modulated by the surface BRDF and geometry terms). The kernel $k_{\Delta\lambda}(u)$ with the support size of $\Delta\lambda$ depends on the distance $u = \|\mathbf{p} - \mathbf{p}_x\|$. Photon mapping traditionally uses the Gaussian kernel, although in the context of SRD a simpler box kernel $k_{\Delta\lambda}(u) = \frac{H(u+\Delta\lambda) - H(u-\Delta\lambda)}{2 \cdot \Delta\lambda}$ proves sufficient [EBR*14].

The error ϵ_L of the estimate provided by Eq. 3 can be stated explicitly as the difference between the estimated and the true radiance (at a given position and direction):

$$L(\mathbf{p}_x, \mathbf{d}_x) = L(\mathbf{p}_x, \mathbf{d}_x, \Delta\lambda) - \epsilon_L(\Delta\lambda) \tag{4}$$

Analysing the statistical behaviour of ϵ_L in dependence on $\Delta\lambda$ is the key step to derive the progressive update of $\Delta\lambda$.

Error analysis Thanks to the formulation of the estimate and error in Eq. 3 and Eq. 4 we can now directly apply the results of the one-dimensional probabilistic framework of Jarosz et al. [JNT*11]. The primary result of their analysis are estimates for the expected value and the variance of $\epsilon_L(\Delta\lambda)$:

$$\mathrm{E}[\epsilon_L(\Delta\lambda)] \approx \Delta\lambda \cdot \mathrm{E}[\gamma] \cdot C_1 \tag{5}$$

$$\mathrm{Var}[\epsilon_L(\Delta\lambda)] \approx \frac{(\mathrm{Var}[\gamma] + \mathrm{E}[\gamma]^2) \cdot C_2}{\Delta\lambda} \tag{6}$$

Here C_1 and C_2 are constants that depend on the specific kernel shape and sampling strategy. These estimates are derived for reconstructing a single sample at a given query location, but it can be shown that they also generalize to a combined reconstruction from multiple samples with different size. For additional details please refer to [JNT*11].

The most important conclusion to draw from Eq. 5 and Eq. 6 is the linear relationship between the reconstruction scale $\Delta\lambda$ and the resulting error characteristics. Specifically if $\Delta\lambda$ *decreases* than the expected error will *decrease linearly* while its variance will *increase linearly*. The first relationship is necessary for the progressive solution to consistently converge, while the increasing variance will be compensated for by the increasing overall number of samples constituting the solution. As will be shown next the proper balance between these two relationships is achieved when $\Delta\lambda$ decreases *sub-linearly*.

Progressive update of $\Delta\lambda$ Under the conditions described previously, Knaus and Zwicker show [KZ11] that a progressive convergence can be achieved by imposing the following ratio of variance between successive passes:

$$\frac{\mathrm{Var}[\epsilon_{i+1}]}{\mathrm{Var}[\epsilon_i]} = \frac{i+1}{i+\alpha}, \tag{7}$$

where $\alpha \in [0,1]$ is a user parameter controlling how steep the progression should be. Due to the inverse proportion between the variance and the reconstruction scale $\Delta\lambda$ (Eq. 6) we get an update rule for the reconstruction scale at pass i:

$$\frac{\Delta\lambda_{i+1}}{\Delta\lambda_i} = \frac{\mathrm{Var}[\epsilon_i]}{\mathrm{Var}[\epsilon_{i+1}]} = \frac{i+\alpha}{i+1} \tag{8}$$

The intuition behind the parameter α is quite simple: for α close to 1 the reconstruction scale will decrease slowly, hence leading to only a slow variance growth. This implies that the solution becomes smooth quickly, but at the cost of only a slow decrease of bias. The opposite is true for small α.

Note that Eq. 8 is independent on the number of samples traced per pass. This is obviously undesired, as different number of samples per pass will yield different results. Jarosz et al. proposed the addition of a parameter M (denoting the number of samples per pass) that compensates for this by applying the reconstruction scale update M-times after each pass (note that the original equation in [JNT*11] contains a typo [Jar14]):

$$\frac{\Delta\lambda_{i+1}}{\Delta\lambda_i} = \prod_{j=1}^{M} \frac{(i-1)M + j + \alpha}{(i-1)M + j + 1} \tag{9}$$

This way the parameter α becomes independent of the pass size and corresponds to changing $\Delta\lambda$ for every individual sample, albeit with a delayed application at the end of each pass. This is especially beneficial in GPU implementations where tracing small sample batches and frequent $\Delta\lambda$ updates would heavily underutilize the available computational resources.

In addition, the recurrent relationship in Eq. 9 can be reformulated to compute $\Delta\lambda_i$ explicitly:

$$\Delta\lambda_i = \Delta\lambda_1 \cdot \frac{1}{iM} \prod_{k=1}^{iM-1} \frac{k+\alpha}{k}$$

However, we deem this less efficient than Eq. 9 as the number of evaluated operations grows with each successive pass.

3.1. Application

Tracing ray differentials (both classic and spectral) is independent on the specific Monte-Carlo method, and is described in detail in [EBR*14] and rephrased in Sec. 2. Different rendering algorithms then apply this information depending on their specific illumination reconstruction methods.

Light tracing traces illumination samples from the light source. Upon landing on an opaque receiver surface at position $\mathbf{p}$ these are connected ("splatted") to the sensor by a projective mapping $\rho \in \mathbb{R}^3 \rightarrow \mathbb{R}^2$. Spectral differentials are applied by extending the basic point samples to 1D line segments from $(\mathbf{p} - \Delta\mathbf{p})$ to $(\mathbf{p} + \Delta\mathbf{p})$ (cf. Sec. 3) and splatting each segment using ρ. The progressive variant simply traces

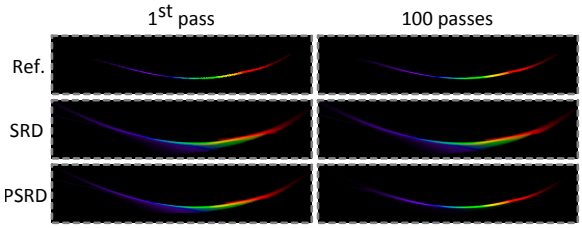

Figure 4: *Narrow beam of light dispersed by a curved object.*

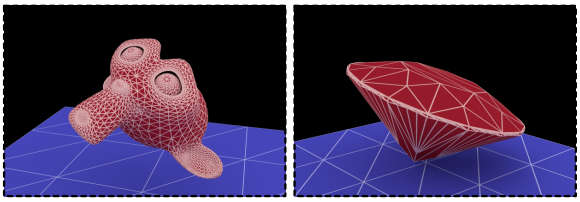

Figure 5: *Scene compositions used for Fig. 6 and Fig. 8. Refractive objects are marked red, while blue corresponds to an opaque diffuse receiver.*

light samples in passes and decreases the reconstruction size at each pass according to Eq. 9. This directly decreases the magnitude of $\Delta\mathbf{p}$, converging to point samples.

Eye tracing operates similarly to light tracing except that the sensor is sampled instead. In each pass, rays are started at sensor positions $\mathbf{p}_e$ and traced through the scene, yielding receiver intersections and the corresponding spectral differentials. These again define 1D sensor footprints from $(\mathbf{p}_e - \rho(\Delta\mathbf{p}))$ to $(\mathbf{p}_e + \rho(\Delta\mathbf{p}))$ due to the reciprocity of light transport. Although splatting is still applicable, it is less practical here since the samples already correspond to sensor (and image) locations directly. Alternatively it is possible to store the SRD statistics ($\Delta\mathbf{p}$ and optionally multilateral filtering weights) in separate images and after the tracing stage apply a 1D spatially-variant linear filter to the solution of each pass, before it is averaged with the overall solution. Such filtering is very fast thanks to its low dimensionality, but assumes the SRD statistics are reasonably smooth (relatively to the filtering scale). However, in contrast to the original SRD the progressive variant does not suffer from the additional bias due to averaging of the differential statistics, as each pass is filtered separately and only then accumulated.

Photon mapping combines the two above approaches, connecting them via a kernel density estimate which is subject to the previously described progressive methods (Sec. 1). Progressive SRD add an additional dimension to this estimate, which results in an anisotropic density estimation kernel as opposed to the standard isotropic (e. g., Gaussian) kernel.

4. Results and Analysis

To analyse progressive spectral ray differentials in detail we have chosen light tracing with line splatting (Sec. 3.1) as the representative algorithm. This is because the effects of applying PSRD can be isolated and examined most clearly, as no other sources of bias are present (compared to the alternatives). On the other hand the results of this analysis apply to the other approaches as well, since the basic concepts behind the PSRD reconstruction are the same.

The core of our implementation is a GPU-based light tracer written in OpenGL and GLSL. The splatting is performed by rasterising line primitives procedurally generated from point samples in a geometry shader. All our results have been generated on the NVidia GeForce GTX 770 graphics card. We universally trace $M = 200$k samples per pass, which in all examples amounts approximately to 50 ms. All presented images are rendered at the resolution of 512^2. For converting spectral intensities to RGB colours the standard pipeline of transforming to an intermediate CIE XYZ space by XYZ colour matching functions [CIE32] has been used.

Visual analysis We first analyse PSRD for a simple setting where a thin pencil of light is dispersed by a curved object. This results in a simple, narrow, colourful caustic (Fig. 4) which is bent by the dispersive object's geometry (first row). This introduces a nonlinearity that cannot be precisely handled by the first-order approximation of the standard SRD, resulting in a biased blurry reproduction (second row). In comparison PSRD, despite initially containing this bias as well, over time converge to the correct shape (third row).

For a more realistic analysis, we examine dispersive caustics produced by two complex objects – the Blender monkey (Fig. 6) and a brilliant-cut gem (Fig. 8). Both of these objects have IORs ranging between 1.6 (violet end) and 1.4 (red end) in the visible spectrum and are illuminated by a constant directional white light-field from the top. Fig. 5 shows a schematic depiction of both scenes. We consider these objects well representative – the monkey model is curved and has both convex and concave areas, while the gem model has many angled facets that cause multiple internal reflections and hence a significant divergence of the incoming light.

Our aim has been to examine the rate of convergence (visually and numerically, see below) of SRD and PSRD in comparison to the point-sampled reference, as well as the bias of the resulting solutions. The SRD solution uses a constant $\Delta\lambda$ value corresponding to $1/6$ of the visible spectrum range (cf. [EBR*14]). The PSRD solution starts with the same $\Delta\lambda$ value, which is then progressively updated using Eq. 9 with $\alpha = 0.9$. All the solutions are shown after selected numbers of passes, as well as converged to a stable state.

The conclusions we can draw from this experiment are in agreement with the theory discussed in Sec. 3. The point-based reference solution is extremely noisy, especially in the areas with low sample density, and therefore requires a high

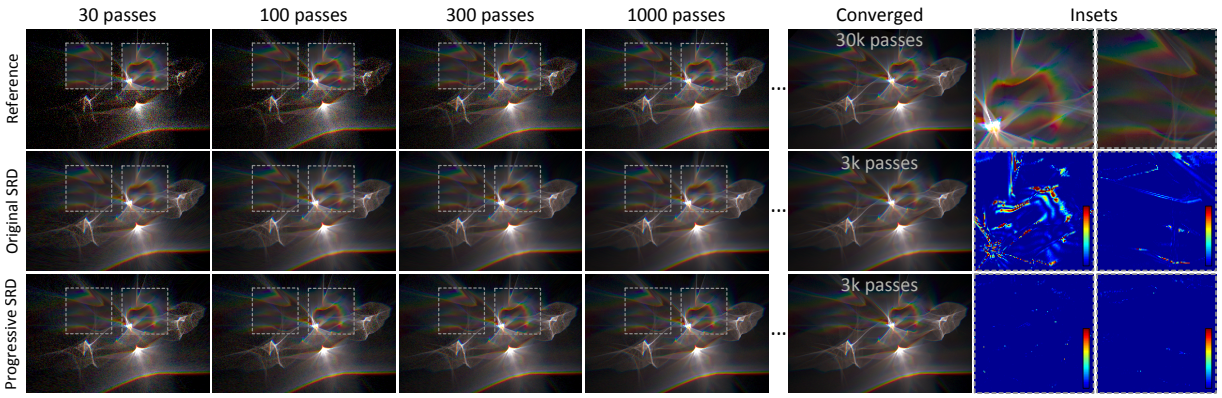

Figure 6: *Convergence of the Blender monkey caustic. We compare the stochastic reference solution to the SRD and PSRD solutions (which converge about an order of magnitude faster). Pieces of the respective converged solutions have been inserted into the partial solutions to enable direct visual comparison of the residual variance. On the right we then show false-colour-coded RMSE differences of the insets between SRD / PSRD solutions and the reference.*

number of passes to converge. In agreement with the results of [EBR*14], SRD produce an exceptionally smooth solution, requiring about 1–1.5 orders of magnitude less passes (and samples thereof). The results however contain visible bias manifested as spatial blurring of features, similarly to Fig. 4 (cf. difference images). In contrast the PSRD solution, as expected, has a slightly slower convergence rate but the results are visually indistinguishable from the reference.

Although the bias produced by SRD is often subtle it will generally increase with the dispersion magnitude, but also with distance the light travels after the interaction, as dispersion is an angular phenomenon. In Fig. 1 we demonstrate this by artificially increasing the IOR range, which leads to a much stronger dispersion. Here the bias is immediately apparent, causing blurring or even different shaping of the caustic features.

Numerical analysis Knaus and Zwicker [KZ11] and Jarosz et al. [JNT*11] have focused on numerically verifying the probabilistic linearity relationships (Eq. 5 and Eq. 6) by measuring the behaviour of several individual pixels of their respective solutions. These results apply in our case as well, and therefore we are mainly interested in the behaviour of our PSRD solution as a whole, to complement the previous analysis. For this we have measured the mean squared error of the entire image for both SRD and PSRD solutions, both in comparison to a converged reference solution. For our PSRD solution we also analyse its behaviour for three different values of the parameter α. These measurements (corresponding to the scenes in Fig. 6 and Fig. 8) are plotted in Fig. 7.

It is immediately apparent that the SRD solution converges the fastest, but not to the expected error value of 0, resulting in bias. Our PSRD solution empirically turns out to be optimal for α values around 0.9, at least for the specific experiments

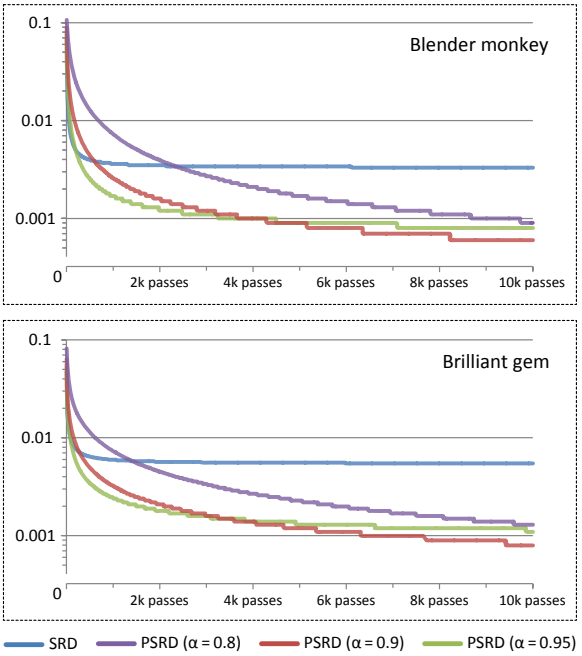

Figure 7: *RMSE plots showing the convergence over 10k passes of the monkey (Fig. 6) and gem (Fig. 8) scenes. It is obvious that SRD converge very fast but to a biased solution, while PSRD, regardless of the actual value of parameter α, approach the correct solution at the rate characteristic for other Monte-Carlo approaches.*

conducted here. For smaller α, such as 0.8 or less, the algorithm regresses to very small $\Delta\lambda$ values too quickly, leading to higher variance and slower convergence. And naturally, the opposite holds for α values close to 1.

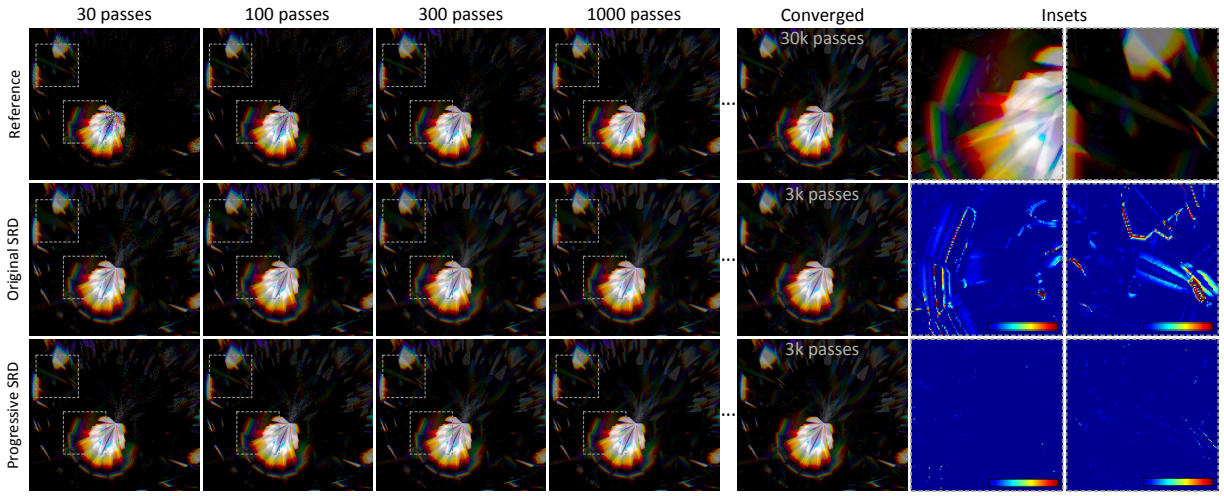

Figure 8: *Convergence of the brilliant gem caustic. We compare the stochastic reference solution to the SRD and PSRD solutions (which converge about an order of magnitude faster). Pieces of the respective converged solutions have been inserted into the partial solutions to enable direct visual comparison of the residual variance. On the right we then show false-colour-coded RMSE differences of the insets between SRD / PSRD solutions and the reference.*

5. Discussion and Conclusion

Our analysis in Sec. 4 demonstrates that the convergence of SRD-based approaches is much better than the basic point-based solution. This naturally comes from the fact that each linear sample footprint generally covers many pixels in the resulting image. Since the size of the footprint is defined in the world space and not in the image space, an interesting side effect is that the convergence of both SRD and PSRD becomes even better if the resolution of the rendered image be higher (because the number of pixels covered by a particular sample footprint increases with resolution as well).

Same as all progressive photon mapping approaches our algorithm is controlled by a scalar parameter α. Although the intuition behind this parameter is easily grasped, the actual value optimal in a given scene configuration is not easy to determine without prior experimentation. Hence a very interesting direction for future work is an automatic determination of an optimal α value. This might be based on a suitable statistical error metric, similar to the work of Hachisuka et al. [HJJ10]. It is also conceivable that different α values could be used in different parts of the scene instead of a single global value.

In addition to the applications in standard Monte-Carlo rendering pipelines (Sec. 3.1), other, more specific areas of rendering might benefit from our work. Examples of these include simulation of light transport in optical systems, most prominently human eye [KMN*04, RIF*09] and artificial lens systems [HESL11, HHH12]. The application of PSRD is even more beneficial in rendering meta-materials or exotic materials with high IOR ranges (Fig. 9) which produce images with a certain artistic value. Here the dispersion is much stronger; therefore, while the standard solution will produce even more noise than usually, spectral differentials will maintain their good convergence properties since their footprints grow proportionally with dispersion magnitude.

Conclusion The introduction of spectral ray differentials by Elek et al. [EBR*14] has enabled a very rapid, near interactive way of working with dispersive refraction. The user-controlled reconstruction size parameter $\Delta\lambda$ makes it possible to first render dispersion coarsely (with larger bias) and fast, and then by decreasing this value achieving a more accurate, slower-converging solution. Our progressive extension of SRD enables a fluent transition between these two modes, so that users can still start with a fast and coarse solution which then automatically converges to the correct, unbiased one. In general these results extend the growing family of ray differentials introduced by Igehy [Ige99], further demonstrating the usefulness of this concept. We look forward to see further applications of these approaches to other, more advanced phenomena.

References

[Arv86] ARVO J.: Backward ray tracing. In *Developments in Ray Tracing. ACM SIGGRAPH Course Notes* (1986), pp. 259–63. 2

[CIE32] CIE: Commission internationale de l'Eclairage proceedings. Cambridge University Press, 1932. 5

[DKL10] DAMMERTZ H., KELLER A., LENSCH H.: Progressive point-light-based global illumination. *Comp. Graph. Forum 29*, 8 (2010), 2504–15. 2

[EBR*14] ELEK O., BAUSZAT P., RITSCHEL T., MAGNOR M., SEIDEL H.-P.: Spectral ray differentials. *Comp. Graph. Forum (Proc. EGSR) 33*, 4 (2014). 2, 3, 4, 5, 6, 7, 8

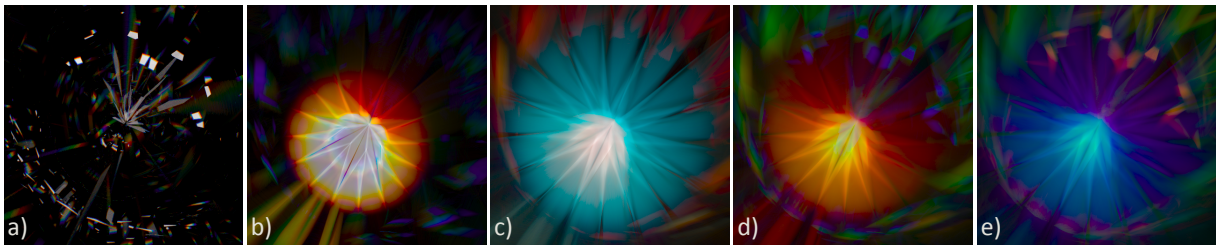

Figure 9: *Caustic by a diamond* (a) *(violet-to-red IOR range 2.5–2.4) and multiple caustics resulting from different meta-materials with IOR ranges 1–1.5* (b)*, 0.05–2.4* (c)*, 0.5–2.5* (d) *and 3.5–0.1* (e)*. The scene composition is identical to Fig. 8.*

[FD09] FABIANOWSKI B., DINGLIANA J.: Interactive global photon mapping. In *Comp. Graph. Forum (Proc. EGSR)* (2009). 2

[HESL11] HULLIN M., EISEMANN E., SEIDEL H.-P., LEE S.: Physically-based real-time lens flare rendering. *ACM Trans. Graph. (Proc. SIGGRAPH) 30*, 4 (2011), 108. 7

[HHH12] HULLIN M. B., HANIKA J., HEIDRICH W.: Polynomial optics: A construction kit for efficient ray-tracing of lens systems. *Comp. Graph. Forum (Proc. EGSR) 31* (2012), 1375–83. 7

[HJ09] HACHISUKA T., JENSEN H. W.: Stochastic progressive photon mapping. *ACM Trans. Graph. (Proc. SIGGRAPH) 28* (2009), 141. 2

[HJJ10] HACHISUKA T., JAROSZ W., JENSEN H. W.: A progressive error estimation framework for photon density estimation. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 29* (2010), 144. 7

[HOJ08] HACHISUKA T., OGAKI S., JENSEN H. W.: Progressive photon mapping. *ACM Trans. Graph. (Proc. SIGGRAPH Asia) 27* (2008), 130. 2

[Ige99] IGEHY H.: Tracing ray differentials. In *Proc. SIGGRAPH* (1999), pp. 179–86. 2, 3, 7

[Jar14] JAROSZ W.: private communication, June 2014. 4

[Jen01] JENSEN H. W.: *Realistic image synthesis using photon mapping*. AK Peters, Ltd., 2001. 2

[JNT*11] JAROSZ W., NOWROUZEZAHRAI D., THOMAS R., SLOAN P.-P., ZWICKER M.: Progressive photon beams. *ACM Trans. Graph. 30* (2011), 181. 2, 3, 4, 6

[Kaj86] KAJIYA J. T.: The rendering equation. In *Proc. SIGGRAPH* (1986), pp. 143–50. 2

[KMN*04] KAKIMOTO M., MATSUOKA K., NISHITA T., NAEMURA T., HARASHIMA H.: Glare generation based on wave optics. In *Proc. Pacific Graphics.* (2004), pp. 133–40. 7

[KZ11] KNAUS C., ZWICKER M.: Progressive photon mapping: a probabilistic approach. *ACM Trans. Graph. 30* (2011), 25. 2, 3, 4, 6

[Lai07] LAI A.: A compression method for spectral photon map rendering. In *Proc. WSCG* (2007). 2

[LSK*07] LAINE S., SARANSAARI H., KONTKANEN J., LEHTINEN J., AILA T.: Incremental instant radiosity for real-time indirect illumination. In *Proc. EGSR* (2007), pp. 277–86. 2

[RIF*09] RITSCHEL T., IHRKE M., FRISVAD J. R., COPPENS J., MYSZKOWSKI K., SEIDEL H.-P.: Temporal glare: Real-time dynamic simulation of the scattering in the human eye. *Comp. Graph. Forum (Proc. Eurographics) 28*, 2 (2009), 183–92. 7

[SFES07] SCHJØTH L., FRISVAD J. R., ERLEBEN K., SPORRING J.: Photon differentials. In *Proc. GRAPHITE* (2007), pp. 179–86. 2

[SW01] SUYKENS F., WILLEMS Y. D.: Path differentials and applications. In *Proc. EGSR* (2001), pp. 257–68. 2

[Tho86] THOMAS S. W.: Dispersive refraction in ray tracing. *The Visual Computer 2*, 1 (1986), 3–8. 2

[WND*14] WILKIE A., NAWAZ S., DROSKE M., WEIDLICH A., HANIKA J.: Hero wavelength spectral sampling. *Comp. Graph. Forum (Proc. EGSR) 33*, 4 (2014). 2

Appendix A: Spectral Ray Differentials

For completeness we rephrase the expression for updating the spectral ray differential (Sec. 2). The directional differential $\frac{\partial \mathbf{d}}{\partial \lambda}$ is updated as

$$\frac{\partial \mathbf{d}'}{\partial \lambda} = \frac{\partial \eta}{\partial \lambda}\mathbf{d} + \eta\frac{\partial \mathbf{d}}{\partial \lambda} - \frac{\partial \mu}{\partial \lambda}\mathbf{n} - \mu\frac{\partial \mathbf{n}}{\partial \lambda}$$

with

$$\frac{\partial \mu}{\partial \lambda} = \frac{\partial \eta}{\partial \lambda}\theta + \eta\frac{\partial \theta}{\partial \lambda} + \frac{-\eta\frac{\partial \eta}{\partial \lambda} + \eta\frac{\partial \eta}{\partial \lambda}\theta^2 + \eta^2\theta\frac{\partial \theta}{\partial \lambda}}{\sqrt{1-\eta^2+\eta^2\theta^2}}$$

$$\text{and} \quad \frac{\partial \theta}{\partial \lambda} = \frac{\partial \mathbf{d}}{\partial \lambda}\cdot\mathbf{n} + \mathbf{d}\cdot\frac{\partial \mathbf{n}}{\partial \lambda}$$

Here, $\theta = \mathbf{d}\cdot\mathbf{n}$. Note that $\mathbf{p}$, $\mathbf{d}$, $\mathbf{n}$ and η are all functions of λ (directly or indirectly). The positional differential $\frac{\partial \mathbf{p}}{\partial \lambda}$ on the other hand is not influenced by refraction.

For a more detailed description (in particular for how to compute the derivatives of $\mathbf{n}$ and η) please refer to [EBR*14].