EUROGRAPHICS 2020 / U. Assarsson and D. Panozzo
(Guest Editors)

*Volume 39 (2020), Number 2*

# Spherical Gaussian Light-field Textures for Fast Precomputed Global Illumination

R. R. Currius[1], D. Dolonius[1], U. Assarsson[1], and E. Sintorn[1]

[1]Chalmers University of Technology, Sweden

**(a)** **(b)**

**Figure 1:** *Two scenes rendered with our method. The local light field for any fragment is available as a precomputed set of 16 Spherical Gaussians in a* light-field texture *(*$512 \times 512$*, 56MB). A similar texture contains the attenuation factor for a preconvolved environment map. The combined result is images with full global illumination for glossy surfaces rendered in just over a millisecond at 1080p resolution.*

**Abstract**

*We describe a method to use Spherical Gaussians with free directions and arbitrary sharpness and amplitude to approximate the precomputed local light field for any point on a surface in a scene. This allows for a high-quality reconstruction of these light fields in a manner that can be used to render the surfaces with precomputed global illumination in real-time with very low cost both in memory and performance. We also extend this concept to represent the* illumination-weighted environment visibility*, allowing for high-quality reflections of the distant environment with both surface-material properties and visibility taken into account. We treat obtaining the Spherical Gaussians as an optimization problem for which we train a Convolutional Neural Network to produce appropriate values for each of the Spherical Gaussians' parameters. We define this CNN in such a way that the produced parameters can be interpolated between adjacent local light fields while keeping the illumination in the intermediate points coherent.*

**CCS Concepts**

• ***Computing methodologies*** → ***Rendering; Ray tracing;***

## 1. Introduction

To achieve realistic computer generated images, the *indirect illumination* of each visible surface point must be accounted for. The current de-facto method for rendering such images is *path tracing*, where the Light Transport Equation [Kaj86] is numerically estimated. In real-time applications, even on high-end GPUs with dedicated ray-tracing hardware, only a few samples per pixel and frame are achievable. Recently, several de-noising techniques have been developed that reuse samples from adjacent pixels and frames [CKS*17; MMBJ17]. These techniques show great promise and allow for rendering scenes with fully dynamic lighting and materials. However, they are still much too expensive on mid or low-end hardware.

Therefore, in applications where lighting, geometry, and materials can be considered static, it is often preferable to rely on precomputing the indirect illumination in the scene and using ray tracing

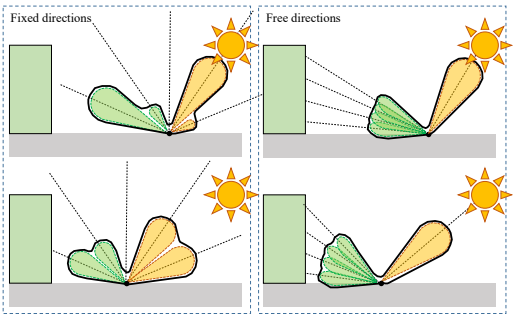



**Figure 2:** *Left: Using Spherical Gaussians with fixed directions [Pet16] (or Spherical Harmonics [RH01]) the incoming light is projected onto the directions being considered. High frequency changes in the illumination cannot be captured, and there will be visible aliasing as we interpolate between two receiving points. Right: With free directions a much higher quality can be obtained and interpolation can be free from aliasing.*

only for specific effects. When illumination can be pre-computed, the remaining questions are how to store a sufficiently dense sampling within a fixed memory budget, how to reconstruct the local light field, and how to convolve it with the *Bidirectional Reflectance Distribution Function* (BRDF) to get the reflected light.

Common choices of *Spherical Radial Basis Functions* (SRBFs) to store the light field are *Spherical Harmonics* (SHs) [RH01] and *Spherical Gaussians* (SGs) [TS06]. With SHs, a few coefficients are stored that describe a set of orthogonal functions on the sphere that can be combined to approximate the light field. With SGs a sum of gaussian lobes are used instead. Wang et al. [WRG*09] describe how the SVBRDF (Spatially Varying Bidirectional Reflectance Distribution Function) can be described in this form for each vertex, allowing for environment lighting in real time. SGs were used to encode light-field textures in the videogame *The Order 1886*, as described by Pettineo and Neubelt [Pet16]. The authors show that, with 12 SGs with *fixed* direction and sharpness (i.e. 36 floats), they can better represent the original light field than a 3-band SH representation (24 floats). Both methods benefit from expressing the BRDF in the same representation as the light fields, allowing for fast and efficient convolution with the incoming illumination.

Figure 2 illustrates a problem with using either SHs or SGs with fixed directions for approximating the incident illumination. Firstly, since the direction of the basis functions are fixed, the lobes can not be moved to where they are most useful. A much better reconstruction of the local light field can be obtained if lobes are concentrated where they are most needed. Secondly, as a source of illumination moves between two of these directions, the reconstructed illumination can only respond by modifying the amplitude, causing clearly visible aliasing in the reflections.

However, allowing for non-fixed directions is far from trivial. Optimizing only the amplitude can be solved with a linear least square solver. With arbitrary directions and sharpness the problem is much more complex. Additionally, it is imperative that the parameters of the SGs are interpolatable between, e.g., nearby light probes or texels in a light map.

The main contribution of this paper is an alternative approach to solving this optimization problem. Instead of optimizing the SG parameters directly, we train a *Convolutional Neural Network* (CNN) to generate them. Figure 3 shows an overview of our system. We start with a scene with a unique UV parametrization and a precomputed irradiance texture. The goal is to create another texture, the *light-field texture*, where every texel contains the SG parameters (axis, sharpness and amplitude) required to recreate the local light field. We first pathtrace the local light field from every texel's position and store it as a 2D *light-field image* to disk. Next, we train the CNN using these images as input to generate a set of parameters for a number of SGs. The sum of SGs are evaluated to predict the light-field image, and the error is backpropagated through the network. When the training has converged, the output SG parameters for each texel are saved as the light-field texture. A benefit of this approach is that, similarly to how an autoencoder works, the network will produce similar SG parameters for adjacent input light field images, and so a lookup in the light-field texture will produce plausible results when interpolated.

Once the light-field texture is created, it can be used to render the scene with indirect illumination in real time. A fragment shader fetches an interpolated set of gaussian parameters and very efficiently convolves this incident illumination with the BRDF to estimate the reflected radiance towards the camera.

As a second contribution we suggest an algorithm for allowing high-resolution glossy reflections from environment maps while taking visibility into account. A common approximation in real-time applications is to preconvolve the environment map with the *Normal Distribution Function* (NDF) and replace the expensive convolution with a single 3D texture lookup at runtime. The remaining terms of the light transport equation are moved outside of the integral and evaluated only for the perfect specular direction. The error of this estimation will be worse the rougher the material is, but in practice it works well for unoccluded reflection. As illustrated in Figure 4, using a preconvolved environment map is problematic when visibility is to be taken into account. Even if some representation of the local visibility is available, the convolution with the environment map must happen at run-time for correct results.

Inspired by the recent work by Heitz et al. [HHM18], we instead suggest rendering, for each texel and all directions, the preconvolved environment both with and without visibility. By taking their ratio we get a spherical function (represented by a 2D image) which we call the *illumination-weighted environment visibility*. These images are then compressed to spherical gaussians, as described above, and can be easily evaluated for any direction in the shader. Multiplying this result by the pre-convolved environment map gives us a high-quality estimation of the actual reflected light.

Together, these novel contributions allow us to render static, complex, scenes with glossy reflections from any viewpoint using high-resolution precomputed illumination and environment visibility stored as a set of a few spherical gaussians per texel. As shown in Figure 1, with 16 SGs per texel (56MB for a 512x512 light-field

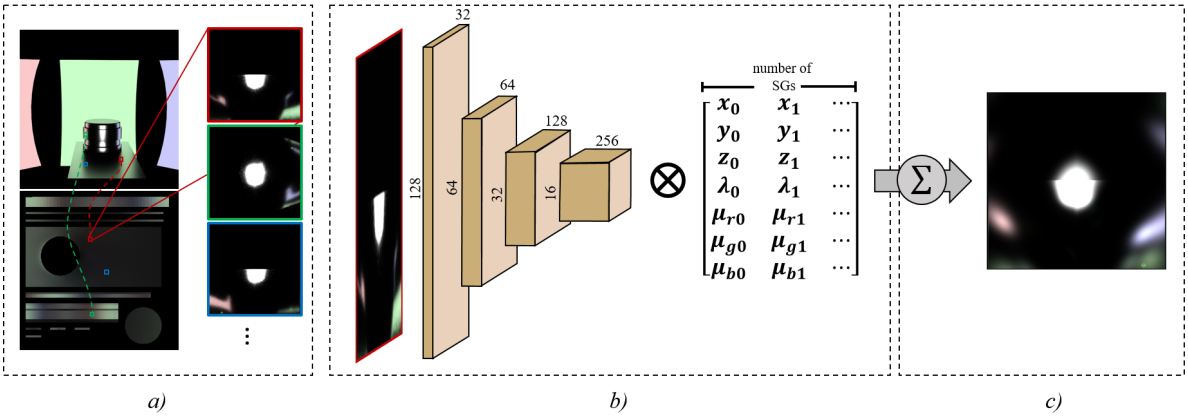



**Figure 3:** *Training a network to estimate spherical gaussian parameters. a) Given a scene and a UV unwrapping, the local light field (environment map) is pathtraced for every texel in the lightmap and stored as an image. b) These images are then passed through a Convolutional Neural Network where each layer consists of a convolution, max pooling and a ReLU activation. The output of final layer is passed through a fully connected layer to produce the parameters of each SG. c) Finally, the predicted local light field is calculated as the sum of these gaussians, and the error is backpropagated through the network. When the network is fully trained, the local light fields of each texel are run through the network again, and the predicted parameters are stored in the corresponding texel of a light-field texture.*

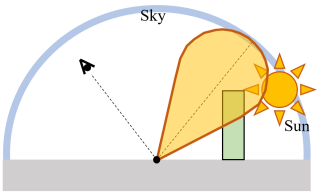



**Figure 4:** *Evaluating visibility only in the center of the BRDF lobe, to attenuate the preconvolved incoming radiance in that direction, can lead to significant light-leaking, as in this example where the surface should not reflect any sunlight.*

texture with 16-bit floats), we achieve smooth results, comparable to a pathtraced reference, in just over a millisecond on an RTX 2080 graphics card.

## 2. Previous Work

**Image-Based Lighting (IBL).** In 1976, Blinn and Newell [BN76] presented their work on *environment maps*, i.e. images representing the incoming radiance for a single point from all directions. For distant illumination, this technique is still in use today, usually extended by preconvolving the incoming radiance with the BRDF to allow for plausible glossy reflections [MS16]. It is common to render several environment maps at several points in the scene, which can then be blended together in an attempt to recreate the light field at an arbitrary point [SZ12]. Unless these *light probes* are extremely densely placed (requiring extensive amounts of memory), such methods will suffer from visibility errors. We refer the reader to a tutorial and survey of image-based lighting by Debevec [Deb06] for more details on image-based lighting.

**Irradiance and Precomputed Radiance Transfer.** For diffuse or very rough materials, light probes can be compactly described using Spherical Harmonics rather than a full environment map [RH01]. In *Precomputed Radiance Transfer* [SKS02], the *transfer function*, i.e., how the incoming radiance is transferred to a specific direction is precomputed. This allows for relighting of an object without recomputing the radiance transfer. This method has been extended to allow dynamic objects [SLS05] and to represent soft shadows [RWS*06].

Spherical Harmonics require many coefficients not to exhibit ringing artifacts when used to represent high-frequency functions, so they are limited to materials with high roughness. Tsai and Shih [TS06] represent both the transfer functions and the light sources with Spherical Gaussians, which allows for high-frequency lighting environments, but this method cannot easily handle spatially varying BRDFs and detailed reflections for rough materials are difficult to reconstruct. Green et al. [GKMD06] also compress the transfer function using Gaussians. Wang et al. [WRG*09] instead represent the BRDF as SGs and represent environment visibility as a *spherical signed distance function*. For environment lights they sample a preconvolved environment map, which will cause artifacts for rough lobes in certain lighting conditions (see Figure 4). To allow for dynamic scenes, Iwasaki et al. [IFDN12] approximate the geometry using spheres to create a visibility estimation which they can efficiently convolve with the lighting and BRDF.

Xu et al. introduced *Anisotropic Spherical Gaussians*, which are shown to produce better reconstructions of some functions with much fewer lobes [XSD*13]. While we do use anisotropic gaussians to represent the BRDF lobe in our real-time evaluation (see Section 6), we use a sum of isotropic gaussians to represent the local light field to avoid the extra amount of memory required.

None of these methods attempt to capture the local light field,

and thus they are not applicable to interreflections and global illumination rendering. In contrast, we suggest both a method for distant environment lighting with improved quality for rough BRDF lobes, and a method in which the precomputed local light field is reconstructed for every texel, allowing for very fast indirect illumination from any surface in the scene, as long as the scene, lighting, and materials can be considered static.

Xu et al. [XCM*14] derive an expression for a SG representing the reflected radiance from one triangle and of a node in a hierarchical representation of the scene, allowing for diffuse and glossy one-bounce interreflections at interactive frame-rates. In the work by Meder and Bruderlin [MB18], a hierarchy of *Virtual Spherical Gaussian Lights* (VSGLs) is generated by mip-mapping a *Reflective Shadow Map*. When shading, a predetermined number of VSGLs are importance sampled from the hierarchy and convolved with the BRDF, expressed as a SG. This method greatly improves the quality of reflections compared to standard Virtual Point Light sampling.

In several of these works [TS06; WRG*09; XSD*13] a method is required to fit a set of Spherical Gaussians to an environment map, which is achieved in an iterative process by first separately solving for directions and sharpness using the L-BFGS-B algorithm [ZBLN97] and then projecting the amplitude using a least-squares solver. In the work by Vorba et al. [VKŠ*14], the sphere of incoming radiance is projected to a 2D plane and a standard *Gaussian Mixture Model* (GMM) is used, rather than Spherical Gaussians. In their work on Normal Map Filtering [HSRG07], Han et al. instead use the von Mises-Fisher distribution to represent the *Normal Distribution Function* in filtered mipmap levels. Similarly to Green et al. [GKMD06], they add a term to the likelihood function that enforces coherency in directions for neighboring lobes, to allow for interpolation. All three use the *Expectation-Maximization* (EM) algorithm to efficiently estimate the gaussian parameters.

Vorba et al. [VKŠ*14] use bi-variate Gaussians to represent incoming radiance but in an off-line rendering context. They maintain a spatial cache of directional gaussian distributions to approximate a PDF for the incoming radiance. The renderer then uses only the closest cached distribution to importance sample new directions, so no interpolation between distributions is required.

**Real-Time Indirect Illumination.** The body of work on real-time indirect illumination is vast and spans decades. We refer the reader to the excellent STAR report by Ritschel et al. [RDGK12] for a detailed survey, and will only cover the most relevant works here.

Much recent work relies on rendering a very noisy image using real-time path tracing and denoising the results, e.g. by factoring the LTE and using carefully chosen filters [MMBJ17], or training a recursive autoencoder [CKS*17]. These methods can work very well but are still quite costly even on high-end hardware.

Faster, and more approximate, methods include Voxel Cone Tracing [CNS*11] where a low-resolution voxel representation of the scene is updated and ray-traced every frame, Photon Splatting approaches [ML09; MSK*16], and Light Propagation Volumes [KD10]. Despite often being able to produce very good results, these algorithms are rarely used in the industry due to their relatively high cost. More often, a combination of sparse precomputed illumination and very approximate screen-space methods, e.g., screen-space reflections [MM14] and screen space ambient occlusion [Mit07], are used. The work by McGuire et al. [MMNL17] falls somewhere between; precomputed environment maps, including normal and distance information, are calculated for sparsely placed light probes, which are then ray marched for each pixel to estimate the color of the reflecting surface.

**Neural network approaches.** Ren et al. [RWG*13] divide the scene into small sub-spaces and store a *Radiance Regression Function* (a small NN) in each, which approximates the outgoing radiance given the viewing direction, surface position, and surface normal. [GvSS17] shows that an image consisting of separate entities can be disentangled into one image per K objects by learning a separate representation vector for each object and a function (a neural network) that allows them to associate each pixel with a specific object. Somewhat similarly, in our method, a CNN learns to map features found in the input light-field images to specific SG parameters.

In the work of Hermosilla et al. [HMRR18], a sparsely sampled point cloud of the scene is processed by a Convolutional Neural Network to obtain abstract features. A second network is trained to process these features, along with the point cloud, to obtain, e.g., AO values for each point. A high-quality shaded image can then be produced, at interactive framerates, by feeding the network the visible points of a 2D image (the GBuffer). The method produces plausible values for points it has not previously seen. View-dependent global illumination is not handled by this method.

Our method is somewhat related to the problem of *inverse graphics* techniques, where the goal is to find scene parameters given observed images. Maximov et al. [MRF18] train a network that describes a *Deep Appearance Map* (DAM) which, given a normal and view direction, outputs the correct radiance for a specific material. They then train a separate network that, given an input photograph, can produce a new DAM very efficiently. Several recent papers have made use of a *differentiable renderer* [LADL18; LHJ19] which can compute derivatives of arbitrary scene parameters from the rendered image to find optimal values. In the work by Chen et al. [CGL*19], a target image is fed through a CNN to predict, e.g., vertex positions which are in turn processed by the differentiable renderer to produce an image. Through back-propagation, the CNN can be updated to improve the estimated vertex positions. Similarly, Wang et al. [WRM17] train a network to reproduce the outgoing radiance given a material, light and view direction. Since it is differentiable, they can then optimize these parameters for a target photograph, allowing for, e.g., inserting new objects in the image with plausible materials and lighting.

Interpolating between environment maps can arguably have similarities to constructing images for novel view points. There, *Deep Neural Network* (DNN) approaches have increasingly gained attraction [FNPS16; ZTF*18; KWR16; SWS*17; FBD*19]. However, these methods do not directly lend themselves for efficiently compressed light-field representations and, when applicable, real-time evaluation is much more expensive than our proposed method. In these methods, neural networks are used to predict the result, which is costly even with hardware acceleration. We only use a network to compute the SG parameters, which are then trivially

interpolated at run-time. DNNs have also been used for other related tasks, such as real-time light field reconstruction [CWZ*18; MKU13], approximate global illumination [TF17], and BRDF estimation from photos [AAL16], to mention a few.

## 3. Light-field Images

We store the gaussian parameters that approximate the local light field for each texel in a *light-field texture*, so all surfaces in the scene need a unique UV mapping. We follow a method similar to Rakhteenko's [Rak18] to obtain the positions and normals for each texel in the *light-field texture* while avoiding artifacts at seams and at points that lie inside other objects. Using these positions and normals, we compute a *light-field image*, a 2D image with the incident radiance projected from the sphere. For this we use a GPU-accelerated path tracer implemented using *Optix* [PBD*10]. We found an environment map size of $128 \times 128$ to be sufficient for the fidelity we can reconstruct and have used that size throughout the project. These light-field images are saved to disk in an uncompressed 16 bit float format, and sum up to tens of GBs for each of our test scenes.

## 4. Optimizing the SG parameters

A single spherical gaussian has the form: $G(\mathbf{v};\mathbf{u},\lambda,\mu) = \mu e^{\lambda(\mathbf{v}\cdot\mathbf{u}-1)}$, where $\mathbf{u}$ is the *axis* of the gaussian lobe, $\mu$ is the *amplitude*, and $\lambda$ is the *sharpness*. For each texel, $t$, we want to approximate each channel, $c$, of each pixel, $i$, in each light-field image, $T_t(\mathbf{v})$, as a sum of N spherical gaussians:

$$P_{ic} = \sum_{j}^{N} G(\mathbf{v}_i;\mathbf{u}_j,\lambda_j,\mu_{jc}) = \sum_{j}^{N} \mu_{jc} e^{\lambda_j(\mathbf{v}_i\cdot\mathbf{u}_j-1)}, \tag{1}$$

where $\mathbf{v}_i$ is a direction corresponding to pixel $i$ and depends on the spherical projection used. Therefore, the problem is to optimize all SG parameters such that the $L_2$ loss is minimized:

$$\sum_{t}^{\text{all texels}} \sum_{i}^{\text{all pixels}} \sum_{c}^{\text{all channels}} \left( \sum_{j}^{N} \mu_{tcj} e^{\lambda_{tj}(\mathbf{v_i}\cdot\mathbf{u_{tj}}-1)} - T_{tc}(\mathbf{v_i}) \right)^2. \tag{2}$$

The number of parameters to optimize scales with the number of texels in the light-field texture. Even with a very small light-field texture of $128 \times 128$ texels and 16 SGs, the number of free parameters to optimize is 1.8M. Additionally, if the gaussians for all texels in the light-field texture are optimized independently, the converged parameters can differ very much between neighboring texels in the light-field texture, resulting in severe visual artifacts when interpolated.

To enforce locally coherent sets of SGs to solve this, previous work has suggested explicitly aligning the axis of adjacent SGs during the optimization task [GKMD06; HSRG07]. We show in Section 7 that, for our problem, this slows convergence and either blurs the resulting reflections, or leaves undesirable artifacts along lines where the optimizer could not resolve conflicting axes.

Instead, we propose a novel formulation of the problem. Rather than trying to optimize the SG parameters directly, we train a Convolutional Neural Network to produce good SG parameters given an input light-field image (see Figure 3). The motivation for our approach is twofold: First, by making the parameters a function of the input image, we encourage similar images to produce similar parameters. This is not guaranteed but, just as an autoencoder will cluster similar images in latent space, our network will tend to make the SG parameters' trajectories locally continuous in the light-field texture, allowing for interpolation. Secondly, rather than training all SG parameters in isolation, the CNN is shared among all texels. Therefore, updating the network to perform better for one texel is likely to improve the result for similar inputs. As will be shown in Section 7, this improves convergence dramatically.

An overview of our network is provided in Figure 3. The input is a 2D light-field image obtained as above, to allow for 2D convolution. We use the octahedron projection suggested by Meyer et al., due to its simplicity [MSS*10]. Note that this projection does not give equal projected area in all directions, which must be accounted for during training. Each layer of the CNN consists of a convolution, max pooling, and ReLU activation. The output of the last layer is the input of a fully connected layer with $N \times M$ outputs, where $N$ is the number of SGs used and $M$ is the number of parameters per SG. Next, the output image is generated by evaluating the sum of gaussians defined by these parameters (Eq 1). The predicted light-field image is compared to the input image and the loss is propagated backward through the network. In the following paragraphs we will go through each of these steps in detail.

**Encoder Network.** During one epoch of training, each light-field image is passed through the CNN to produce the predicted SG parameters. Each CNN layer performs a convolution of $3 \times 3$-pixel spatial support and a ReLU activation, followed by $2 \times 2$ max pooling to produce a new image of half the size. The first convolution layer produces an image with 32 channels and each of the three subsequent layers doubles the number of channels, resulting in a final image of $8 \times 8$ feature vectors with 256 channels. This is then used as input to a fully connected layer, without activation function, that outputs an $N \times M$ matrix of real numbers, each taken to represent one of the M parameters in one of the N SGs. The hyperparameters of the network were found empirically, and kept as low as possible without introducing a visual degradation of the result of our more challenging scenes.

**Loss Function.** Once the constrained SG parameters are available, we can run a final kernel to reconstruct the predicted light-field image. For each pixel and channel we evaluate the sum of the predicted spherical gaussians using Eq 1. Since the input and the predicted images represent radiance, which may be of high dynamic range, we minimize the $L_2$ log loss function: $(\log(T_{ic}+1) - \log(P_{ic}+1))^2$. This ensures that very high energy values (e.g., directly visible light-sources or specular highlights) are not given too much importance compared to darker areas. The details of back-propagating the gradient of the $L_2$ log loss with respect to each parameter are given in the Appendix A.

At this step we take into account that the projection used is not area-preserving: to avoid some pixels having more weight, their gradient contribution needs to be scaled relative to their unprojected solid angle.

(a) (b) (c) (d) (e)

**Figure 5:** *Two of our scenes with: a) Only diffuse component from irradiance map, b) Reflections from preconvolved environment map, c) Environment visibility using 16 SGs, d) Interreflections using 16 SGs. e) Is a path-traced reference.*

**Constraints.** During backpropagation we enforce constraints on the generated parameters by modifying their gradients depending on the type of parameter: the axis of the SG are constrained to be of unit length, and the amplitude and sharpness are constrained to be positive. We will note that, while the axis *could* be expressed using only two values, e.g. spherical coordinates, this would cause discontinuities both for the training network and for the real-time renderer when interpolating between directions. Also, while a negative amplitude is not necessarily erroneous, we found that enforcing strictly positive amplitudes consistently improved our results.

**Optimizations.** There are two non-obvious optimizations we have employed in the training. First, a pixel of the input image contains the average incoming radiance from a small set of directions, rather than a single direction. This must be accounted for when training, otherwise the network can overtrain and produce unwanted artifacts. However, evaluating Eq 1 for several directions for each pixel would be very costly so, instead, we randomly jitter the direction used for evaluation and take a single sample, which we found to be sufficient to avoid overtraining. Secondly, the path-traced input images only have valuable information in the hemisphere centered on the normal. Therefore, we do not let directions, $\mathbf{v}$, below the normal, $\mathbf{n}$, contribute to the gradient at all.

## 5. Illumination Weighted Environment Visibility

Ignoring visibility, mirror reflections from an environment map can be achieved with a single texture lookup. For glossy materials, modern applications usually employ some kind of Torrance Sparrow BRDF [TS92], making the light reflected to the camera from the environment be:

$$L_o(\omega_o) = \int_\Omega \frac{D(\omega_h)G(\omega_i,\omega_o)F(\omega_o)}{4\,|\omega_o\cdot\mathbf{n}|\,|\omega_i\cdot\mathbf{n}|} L_E(\omega_i)V_E(\omega_i)\,|\omega_i\cdot\mathbf{n}|\,d\omega_i, \tag{3}$$

where $\omega_o$ is the direction to the camera, $\omega_h$ is the half vector, $\Omega$ is all directions on the hemisphere, $D$ is the Microfacet Distribution Function, $G$ describes attenuation due to masking and shadowing, $F$ is the fresnel term, and $\mathbf{n}$ is the surface normal. The visibility term, $V$, is often ignored. To achieve the look of a glossy material, without sampling the environment map excessively, a common trick is to preconvolve the $D(\omega_h)L_E(\omega_i)$ term for varying material roughnesses into a 3D texture, assuming a surface facing $\omega_o$, and then approximate the reflected light as:

$$L_o(\omega_o) = \left(\int_\Omega D(\omega_h)L_E(\omega_i)d\omega_i\right)\frac{G(\omega_r,\omega_o)F(\omega_o)}{4\,|\omega_o\cdot\mathbf{n}|\,|\omega_r\cdot\mathbf{n}|}\,|\omega_r\cdot\mathbf{n}|, \tag{4}$$

where $\omega_r$ is $\omega_o$ reflected around the normal. This approximation is increasingly incorrect for rougher materials, and for grazing viewing directions, but often looks plausible and is commonly used in practice. Alternatively, a *dominant* reflection vector can be calculated by shifting the specular reflection vector towards the normal at grazing angles [Seb14]

While our light-field texture could contain illumination from the environment, that would be the same for every point in the scene so, rather than spending SGs on reconstructing the environment map at every texel, it is preferable to make use of the existing high-resolution pre-convolved environment map. The most obvious approach might be to use a texture of sums of SGs to approximate the visibility function, $V_E(\omega_i)$, but for rough materials this would be insufficient, as illumination is contributed from a larger cone of directions.

Instead, we extend an idea recently published by Heitz et

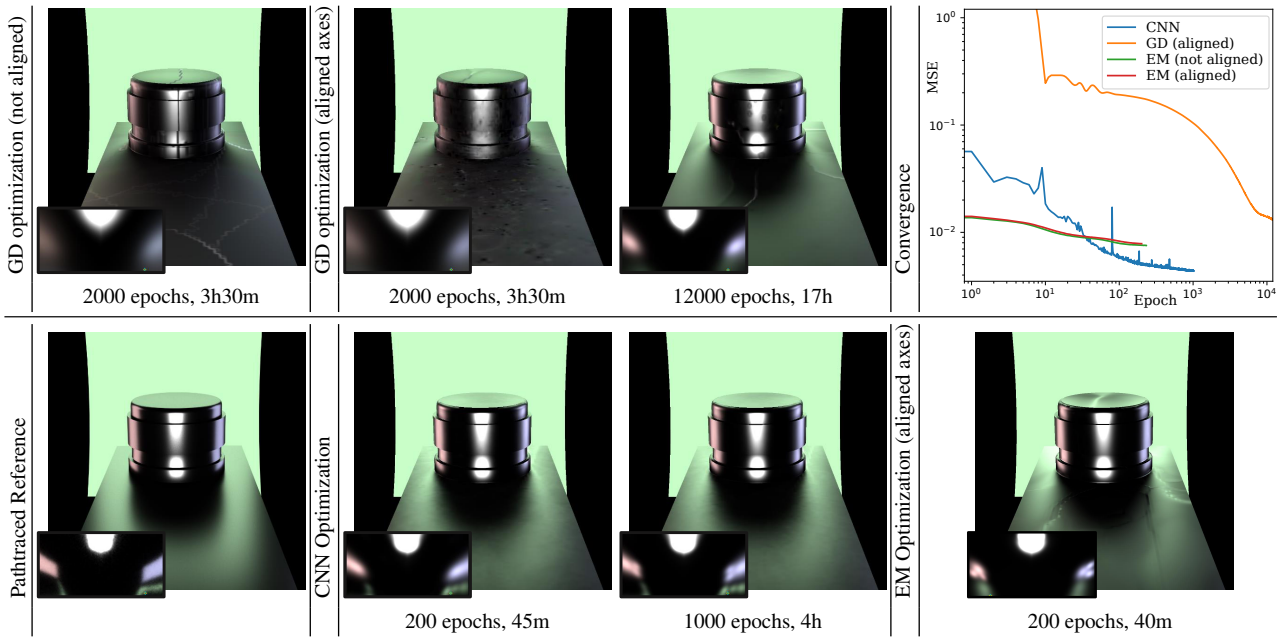



**Table 1:** *With no additional constraint, optimizing the SG parameters directly (EM or GD) causes disturbing artifacts between texels with very different solutions. Adding a regularization constraint (aligned axes) can alleviate this problem, but dampens the system, causing it to converge with a far from optimal result. By using a single CNN to produce all sets of SG parameters, local coherence between sets of SGs is enforced and the final MSE, which compares the input lightfield image with its obtained SG representation, is much smaller. The insets show the predicted light-field image for one texel (compare to the ground-truth light-field image in the inset of the pathtraced image).*

al. [HHM18], where correct soft shadows are computed by combining analytic area-light illumination and denoised, raytraced visibility. They suggest estimating the *illumination-weighted shadow*

$$W_S(\omega_o) = \frac{\int_\Omega R(\omega)L(\omega)V(\omega)d\omega}{\int_\Omega R(\omega)L(\omega)d\omega}, \tag{5}$$

where $R$ is the cosine weighted BRDF, $L$ is the incoming radiance from the light-source and $V$ is the visibility. This term is stochastically estimated and then multiplied by the exact analytical estimation of the unshadowed illumination, $\int_\Omega R(\omega)L(\omega)d\omega$.

In our case, we consider the incoming radiance from an environment map, rather than an area light, and we have no means of evaluating that analytically. We can, however, preconvolve the unoccluded environment map:

$$U(\omega_o) = \int_\Omega D(\omega_h)L_E(\omega_i)d\omega_i, \tag{6}$$

and store the result in a 3D texture. We then precompute the *illumination-weighted environment visibility*:

$$W_E(\omega_o) = \frac{\int_\Omega D(\omega_h)L_E(\omega_i)V(\omega_i)d\omega_i}{\int_\Omega D(\omega_h)L_E(\omega_i)d\omega_i}, \tag{7}$$

for each light-field texel using a path tracer. This function we also represent using SGs, trained as described above.

Finally, in the real-time shader, we can multiply the preconvolved environment illumination with this estimation and, again, approximate the remainder of the LTE using the perfect specular reflection direction:

$$L_o(\omega_o) = U(\omega_o)W_E(\omega_o)\frac{G(\omega_r,\omega_o)F(\omega_o)}{4\,|\omega_o\cdot\mathbf{n}|\,|\omega_r\cdot\mathbf{n}|}\,|\omega_r\cdot\mathbf{n}| \tag{8}$$

As shown in Table 3, this way we can achieve plausible, high-resolution, glossy reflections from an environment map with visibility at the small cost of one environment lookup and evaluating a sum of spherical gaussians. This technique can be used on its own or in combination with the method described in the previous section. Note that while we compute the full, three-channel, illumination-weighted environment visibility, it would also in many cases be sufficient to use a monochrome result, reducing the amount of memory traffic.

## 6. Real-time Algorithm

To render the images shown in this paper we have used a deferred shading pipeline and applied the lighting from our light-field textures and environment visibility in the global lighting pass.

The light-field and environment visibility textures containing the gaussian parameters are read from disk and stored in *texture arrays*. Although our light-field texture could be used for diffuse reflections as well, we instead use the existing precomputed irradiance light map, and use the light-field texture only for glossy reflections. All illumination in the scenes comes from emissive surfaces or the environment.

The steps to apply the illumination from the light-field texture, for each pixel in the fragment shader, are:

1. Fetch position, normal, uv-coordinates, and material properties from the G-buffer.
2. Look up irradiance in the precomputed texture and calculate diffuse reflection.
3. Calculate the (anisotropic) SG that represents the $D(\omega_h)$ term from the material properties.
4. Fetch one SG at a time from the light-field texture (7 parameters, i.e., two texture lookups per SG) and convolve it with $D$.
5. Calculate the other BRDF terms F, G, and the dot products in the divisor for the perfect specular direction.
6. Multiply the obtained terms to obtain the glossily reflected radiance for this SG and accumulate it to the total glossy reflected radiance from the light-field texture.

To evaluate and convolve the spherical gaussians, we follow Pettineo [Pet16], the relevant definitions from which have been included in Appendix B, and we refer the reader to the paper by Wang et al. [WRG*09] for a full derivation.

The steps to render the environment map reflections using the illumination-weighted environment visibility method are:

1. Fetch position, normal, uv-coordinates, and material properties from the G-buffer (re-use the information already obtained when applying light-field texture).
2. Based on material roughness, look up $U(\omega_o)$ from the preconvolved environment map.
3. Fetch each SG (two texture lookups per SG) from the visibility factor texture and evaluate it in the reflected direction. Accumulate the evaluated value to obtain the visibility factor $W_E(\omega_o)$ for that pixel.
4. Calculate glossily reflected radiance from the environment according to Eq 8.

## 7. Results

The evaluation of our method was performed on an Intel core i7-8700 with an RTX 2080 graphics card. The training is implemented using nVidia's CUDA and cuDNN, and the real-time renderer is implemented in OpenGL. All scenes are lit only by our proposed method, either from emitting surfaces or environment maps. Direct lighting can be orthogonally added with any standard method.

**Direct optimization of SG parameters.** We primarily compare our suggested method of using a CNN to generate the SG parameters to a direct optimization of the parameters using Gradient Descent (GD). We have chosen a Gradient Descent solver, as that lets us train using the same initialization, loss function, and parameter gradients, allowing us to evaluate the benefit of using a CNN in isolation. We have additionally performed one comparison with directly optimizing the gaussian parameters using *Expectation Maximization* (EM) [HSRG07; HZE*19], by normalizing the amplitude of the SGs so the integral over the sphere adds up to 1, letting us treat the sum of them as a von Mises-Fisher mixture. We have observed that EM can converge much faster and to a better MSE result than GD (see Table 1), even though it's goal is not to optimize for MSE.

As expected, adjacent sets of SG parameters can not be smoothly interpolated if the parameters are optimized in isolation. To remedy this, we add the regularization constraint suggested by Green et al. [GKMD06] to GD optimization, and an alignment term [HSRG07] to EM. In both algorithms, the axes of the SGs are pushed towards the average of adjacent texels' axes. Introducing a constraint alleviates the problems slightly but, when converged, the reflections still show strong artifacts along lines where one SG changes too quickly. If we increase the weight of the constraint further, it dampens the system and the training converges at a much higher MSE. The result is very blurry reflections.

In contrast, when training a CNN to generate the parameters, coherence between nearby sets of gaussians is maintained indirectly, still allowing parameters to change quickly when doing so does not affect the MSE. This leads to a much better MSE for the converged result and the images obtained when using the SGs for reflection are much closer to the pathtraced reference.

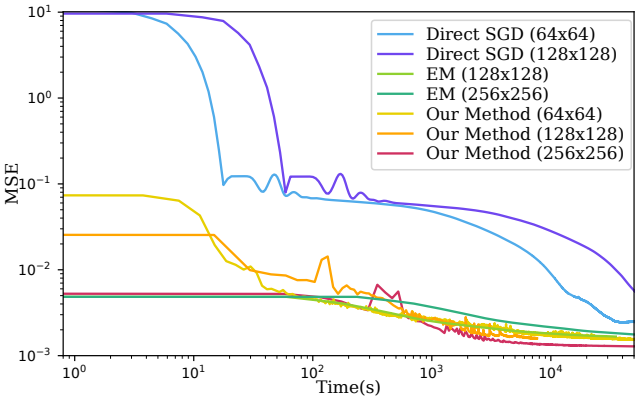



**Figure 6:** *Convergence for a simple scene with light-field textures of varying size.*

Using a CNN to produce the SG parameters also scales very well with the resolution of the light-field texture, as illustrated in Figure 6. Here, we can see that, while each epoch of training will take time proportional to the number of input light-field images, the time to convergence is essentially unaffected. Using our method, training is converged after approximately three hours, regardless of the light-field texture resolution. With a direct optimization of parameters (here without enforcing coherence between sets of SGs), the time to convergence is proportional to the number of input images. The average MSE of the final predicted light-field image is also much better with our method and, interestingly, improves with increased resolution.

**Quality.** Table 2 shows a comparison of our method for rendering images with precomputed light fields, using varying numbers of SGs per texel, and a path-traced reference. The scene is a simple test scene containing objects with a material of increasing roughness (0.2, 0.3, 0.4, and 0.5, GGX BRDF [WMLT07]) from left to right. The scene is illuminated by a number of emitting arcs that can be seen in the background. In the right column we see the light field as it was reconstructed for one of the pixels. The resolution of the light-field texture is rather small ($256 \times 256$), to show that the gaussians can be interpolated with very plausible results.

For the three rightmost objects (roughness $>= 0.3$), the reconstructed light-fields are sufficient to produce an image that matches

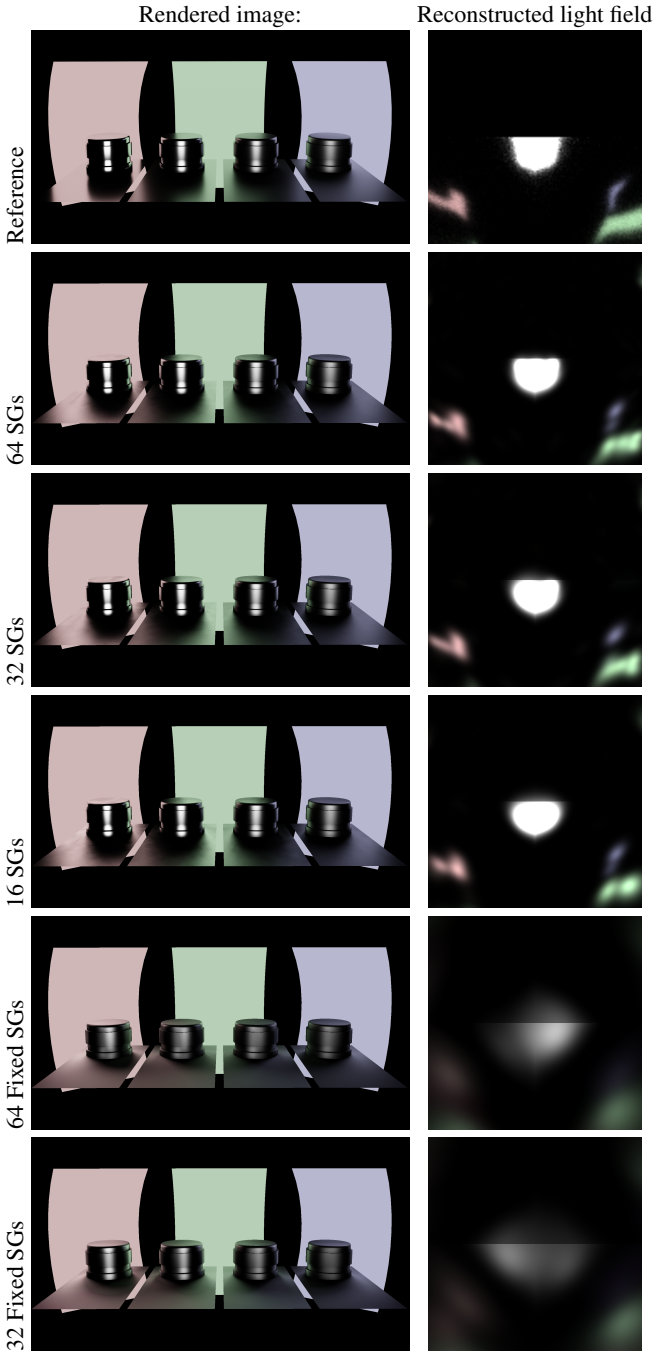



**Table 2:** *Quality of reflections compared to a path-traced reference for varying numbers of SGs per texel. In the right column is the reconstructed light field for one pixel.*

the path-traced reference quite well, even with 16 SGs. At lower roughness levels, the remaining errors become more obvious and on the clear, flat plane the reflections might not be acceptable even with 64 SGs. On a curved object, or a textured material (see, e.g., Figure 1b), the quality of highly glossy reflections can be quite sufficient with as few as 16 SGs. Looking at the leftmost plane, We can identify two main sources of error. First, since the reconstructed light-field image consists only of a sum of gaussians, straight, hard lines are difficult to reconstruct, leading to somewhat smudgy reflections. Secondly, in some places we can see what looks like folds in the flat plane. These artifacts appear when a SG changes direction quickly over a few pixels, which the network might deem necessary to reduce the overall error.

The two bottom rows show the results when using fixed directions as in previous work [Pet16]. Here, even the most rough material is clearly not comparable to the path-traced reference, and the errors are even more visible in motion, as can be seen in the accompanying video. This is not surprising when looking at the corresponding light-field image. The available SGs are necessarily spread uniformly over the sphere and the majority of them do not contribute at all.

In Table 3, we show a similar scene, but illuminated by an environment map, and using our precomputed illumination-weighted environment visibility method. While reflections are not quite as sharp as in the path-traced reference, our method works as a very convincing visibility estimator for any direction even with this quite challenging, high frequency, HDR environment map. Note in the right column that the reconstructed illumination weighted environment visibility is not a simple visibility map, but an attenuation factor for the preconvolved environment map.

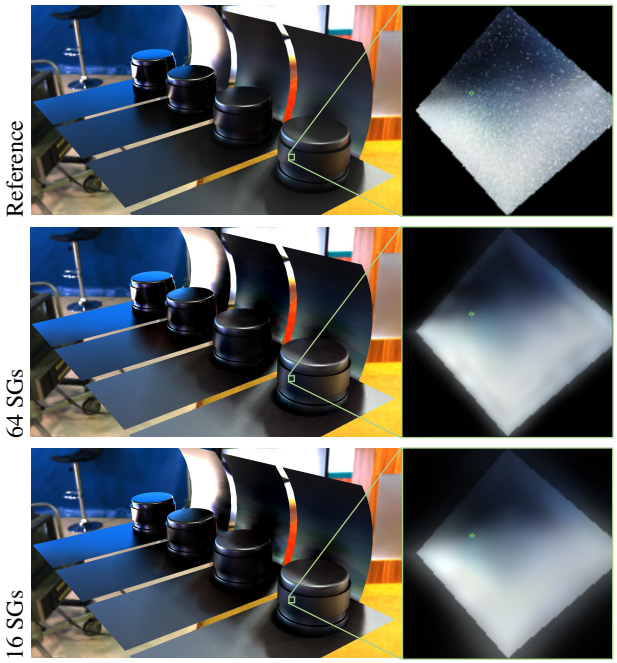
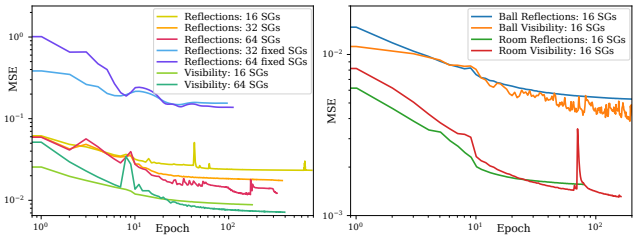



**Table 3:** *Quality of illumination weighted environment visibility compared to a path-traced reference for varying numbers of SGs per texel. In the right column is the reconstructed visibility for one pixel.*

**Convergence.** In Figures 7a and 7b, we show the loss as a function of the number of epochs the network has been trained. In all




(a) (b)

**Figure 7:** *Convergence of the networks trained for Tables 2 and 3, and for Figure 1.*

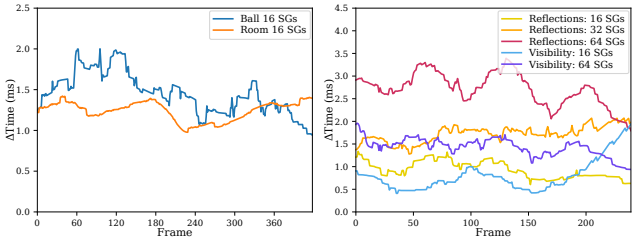



**Figure 9:** *The time taken for our deferred shading pass, including evaluation of indirect illumination from light-field textures, for each frame of the accompanying videos.*

of our tests, the MSE improves only very slightly after 128 epochs, and in general, a good result is obtained after 30 epochs. In the first graph, each epoch took approximately one minute to train, and in Figure 7b, which has a larger light-field texture ($512 \times 512$), each epoch took approximately five minutes. In Figure 7a, we also show the convergence when training for fixed directions. Here, we could reach convergence by directly optimizing the parameters with gradient descent.

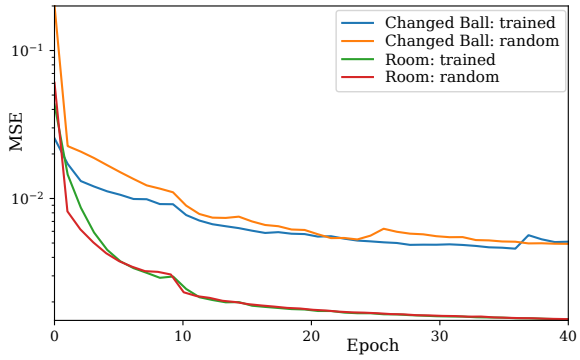



**Figure 8:** *Convergence when starting with a pre-trained network. The network is initialized with the values obtained from training for Figure 1b and then trained for two different scenes. The Changed Ball scene is very similar to the one the network is trained for, and the Room scene is the scene shown in 1a.*

To evaluate how general the trained network is, we have experimented with initializing the network weights with the converged weights for a different scene. The results are shown in Figure 8. We trained the scene shown in Figure 1b to obtain an initial network state and then trained two different scenes. One of these scenes was obtained by moving objects around in the original scene, and the other is the scene shown in Figure 1a. Although the MSE obtained after the first few epochs was slightly better than for random initialization, we did not find that a pre-trained network improved convergence in either case. We believe the old input images, although similar, do not contain sufficiently similar features for the network to generalize.

**Performance.** Finally, in Figure 9, we show the time taken to render each frame of the accompanying videos. All images are rendered at a resolution of $1920 \times 1080$, and the times shown are the time taken for the *Deferred Shading* pass, which evaluates all SGs (the total frametime includes an additional 0.2ms for rendering the GBuffer). Not unexpectedly, the performance is mostly proportional to the number of SGs evaluated. Since evaluating and convolving spherical gaussians is very cheap, the costly part of our approach is the number of texture fetches required. That memory is the bottleneck is further evidenced by the fact that performance improves significantly when we use 16 bit floating point values rather than 32 bit to describe our SG parameters. Since using 16 bit floats has no visible impact, that is what we have used in all meassurements in this paper. We also attempted to further reduce the size of our light-field textures by converting them to 8 bit values. This had a significant impact on quality, however, and did not improve performance much. To reduce the memory footprint further, it might instead be possible to use any of the compressed texture formats available in hardware, but we have not yet explored this further.

## 8. Conclusion and Future Work

We have shown that the quality of light-field textures, represented by Spherical Gaussians, can be greatly increased by allowing for arbitrary axes. We suggest training a Convolutional Neural Network to produce appropriate parameters for these SGs, rather than optimizing the spherical gaussians' parameters directly, and show that good results are obtained, for complex scenes, within a few hours of training. Additionally, we suggest a novel method for approximating environment visibility, by precomputing the *illumination weighted environment visibility*, and show that the same network can be used to create the SGs describing this function. Our real-time indirect illumination algorithm is extremely fast on modern high-end hardware and should perform well within real-time even on much older hardware or even portable devices.

Generating one of the converged ground-truth images shown in Figure 5e takes about 5 minutes on our RTX 2080 card, with an Optix renderer. A noisy, but recognizable, picture can be rendered within seconds. By significantly simplifying the allowed types of light-transport and scene-geometry, and making heavy use of temporal denoising filters, a pathtraced image can be obtained at interactive framerates (see e.g, QuakeRTX). For high-quality scenes and arbitrary lighting, fully dynamic solutions are still not available, however. Our method admittedly requires hours of baking and training as a preprocess (around 8 hours of baking and 6 of training), but allows for good quality global-illumination images for a time budget of 1-2 milliseconds per frame (see Figure 9).

In this work, we have concentrated on storing the SG parameters in two-dimensional textures, but another promising area would be to approximate densely placed light probes, which would allow dynamic objects to reflect the static scene. Although our examples do not require much memory for the light-field textures, a larger scene might require much higher resolution and then the memory cost of our method would naturally grow. Therefore, another interesting area of future work is to further compress the light-field data. This could be achieved as simply as using hardware compression for the textures, or it might be possible to take advantage of the coherency between texels in the light-field texture.

## 9. Acknowledgments

This work was supported by the Swedish Research Council under Grant 2014-4559, and 2017-05060.

## References

[AAL16] AITTALA, MIIKA, AILA, TIMO, and LEHTINEN, JAAKKO. "Reflectance Modeling by Neural Texture Synthesis". *ACM Trans. Graph.* 35.4 (2016) 5.

[BN76] BLINN, JAMES F. and NEWELL, MARTIN E. "Texture and Reflection in Computer Generated Images". *Commun. ACM* 19.10 (Oct. 1976), 542–547. ISSN: 0001-0782. DOI: 10.1145/360349.360353. URL: http://doi.acm.org/10.1145/360349.360353 3.

[CGL*19] CHEN, WENZHENG, GAO, JUN, LING, HUAN, et al. "Learning to Predict 3D Objects with an Interpolation-based Differentiable Renderer". *ArXiv* abs/1908.01210 (2019) 4.

[CKS*17] CHAITANYA, CHAKRAVARTY R. ALLA, KAPLANYAN, ANTON S., SCHIED, CHRISTOPH, et al. "Interactive Reconstruction of Monte Carlo Image Sequences Using a Recurrent Denoising Autoencoder". *ACM Trans. Graph.* 36.4 (July 2017), 98:1–98:12. ISSN: 0730-0301. DOI: 10.1145/3072959.3073601. URL: http://doi.acm.org/10.1145/3072959.3073601 1, 4.

[CNS*11] CRASSIN, CYRIL, NEYRET, FABRICE, SAINZ, MIGUEL, et al. "Interactive Indirect Illumination Using Voxel Cone Tracing: A Preview". *Symposium on Interactive 3D Graphics and Games*. I3D '11. San Francisco, California: ACM, 2011, 207–207. ISBN: 978-1-4503-0565-5. DOI: 10.1145/1944745.1944787. URL: http://doi.acm.org/10.1145/1944745.1944787 4.

[CWZ*18] CHEN, ANPEI, WU, MINYE, ZHANG, YINGLIANG, et al. "Deep Surface Light Fields". *Proc. ACM Comput. Graph. Interact. Tech.* 1.1 (July 2018), 14:1–14:17. ISSN: 2577-6193. DOI: 10.1145/3203192. URL: http://doi.acm.org/10.1145/3203192 5.

[Deb06] DEBEVEC, PAUL. "Image-based Lighting". *ACM SIGGRAPH 2006 Courses*. SIGGRAPH '06. Boston, Massachusetts: ACM, 2006. ISBN: 1-59593-364-6. DOI: 10.1145/1185657.1185686. URL: http://doi.acm.org/10.1145/1185657.1185686 3.

[FBD*19] FLYNN, JOHN, BROXTON, MICHAEL, DEBEVEC, PAUL E., et al. "DeepView: View Synthesis With Learned Gradient Descent". *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 2362–2371 4.

[FNPS16] FLYNN, JOHN, NEULANDER, IVAN, PHILBIN, JAMES, and SNAVELY, NOAH. "Deep Stereo: Learning to Predict New Views from the World's Imagery". *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), 5515–5524 4.

[GKMD06] GREEN, PAUL, KAUTZ, JAN, MATUSIK, WOJCIECH, and DURAND, FRÉDO. "View-dependent precomputed light transport using nonlinear gaussian function approximations". *In ACM Symposium on Interactive 3D graphics*. 2006, 7–14 3–5, 8.

[GvSS17] GREFF, KLAUS, van STEENKISTE, SJOERD, and SCHMIDHUBER, JÜRGEN. "Neural Expectation Maximization". *NIPS*. 2017 4.

[HHM18] HEITZ, ERIC, HILL, STEPHEN, and MCGUIRE, MORGAN. "Combining Analytic Direct Illumination and Stochastic Shadows". *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '18. Montreal, Quebec, Canada: ACM, 2018, 2:1–2:11. ISBN: 978-1-4503-5705-0. DOI: 10.1145/3190834.3190852. URL: http://doi.acm.org/10.1145/3190834.3190852 2, 7.

[HMRR18] HERMOSILLA, PEDRO, MAISCH, SEBASTIAN, RITSCHEL, TOBIAS, and ROPINSKI, TIMO. "Deep-learning the Latent Space of Light Transport". *Comput. Graph. Forum* 38 (2018), 207–217 4.

[HSRG07] HAN, CHARLES, SUN, BO, RAMAMOORTHI, RAVI, and GRINSPUN, EITAN. "Frequency Domain Normal Map Filtering". *ACM Trans. Graph.* 26.3 (July 2007). ISSN: 0730-0301. DOI: 10.1145/1276377.1276412. URL: http://doi.acm.org/10.1145/1276377.1276412 4, 5, 8.

[HZE*19] HERHOLZ, SEBASTIAN, ZHAO, YANGYANG, ELEK, OSKAR, et al. "Volume Path Guiding Based on Zero-Variance Random Walk Theory". *ACM Trans. Graph.* 38.3 (June 2019). ISSN: 0730-0301. DOI: 10.1145/3230635. URL: https://doi.org/10.1145/3230635 8.

[IFDN12] IWASAKI, KEI, FURUYA, WATARU, DOBASHI, YOSHINORI, and NISHITA, TOMOYUKI. "Real-time Rendering of Dynamic Scenes under All-frequency Lighting using Integral Spherical Gaussian". *Comput. Graph. Forum* 31 (2012), 727–734 3.

[Kaj86] KAJIYA, JAMES T. "The Rendering Equation". *SIGGRAPH Comput. Graph.* 20.4 (Aug. 1986), 143–150. ISSN: 0097-8930. DOI: 10.1145/15886.15902. URL: http://doi.acm.org/10.1145/15886.15902 1.

[KD10] KAPLANYAN, ANTON and DACHSBACHER, CARSTEN. "Cascaded Light Propagation Volumes for Real-time Indirect Illumination". *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '10. Washington, D.C.: ACM, 2010, 99–107. ISBN: 978-1-60558-939-8. DOI: 10.1145/1730804.1730821. URL: http://doi.acm.org/10.1145/1730804.1730821 4.

[KWR16] KALANTARI, NIMA KHADEMI, WANG, TING-CHUN, and RAMAMOORTHI, RAVI. "Learning-based View Synthesis for Light Field Cameras". *ACM Trans. Graph.* 35.6 (Nov. 2016), 193:1–193:10. ISSN: 0730-0301. DOI: 10.1145/2980179.2980251. URL: http://doi.acm.org/10.1145/2980179.2980251 4.

[LADL18] LI, TZU-MAO, AITTALA, MIIKA, DURAND, FRÉDO, and LEHTINEN, JAAKKO. "Differentiable Monte Carlo ray tracing through edge sampling". *ACM Trans. Graph.* 37 (2018), 222:1–222:11 4.

[LHJ19] LOUBET, GUILLAUME, HOLZSCHUCH, NICOLAS, and JAKOB, WENZEL. "Reparameterizing discontinuous integrands for differentiable rendering". *Transactions on Graphics (Proceedings of SIGGRAPH Asia)* 38.6 (Dec. 2019). DOI: 10.1145/3355089.3356510 4.

[MB18] MEDER, JULIAN and BRÜDERLIN, BEAT D. "Hemispherical Gaussians for Accurate Light Integration". *ICCVG*. 2018 4.

[Mit07] MITTRING, MARTIN. "Finding Next Gen: CryEngine 2". *ACM SIGGRAPH 2007 Courses*. SIGGRAPH '07. San Diego, California: ACM, 2007, 97–121. ISBN: 978-1-4503-1823-5. DOI: 10.1145/1281500.1281671. URL: http://doi.acm.org/10.1145/1281500.1281671 4.

[MKU13] MIANDJI, EHSAN, KRONANDER, JOEL, and UNGER, JONAS. "Learning Based Compression of Surface Light Fields for Real-time Rendering of Global Illumination Scenes". *SIGGRAPH Asia 2013 Technical Briefs*. 24. ACM, 2013, 24:1–24:4 5.

[ML09] MCGUIRE, MORGAN and LUEBKE, DAVID. "Hardware-accelerated Global Illumination by Image Space Photon Mapping". *Proceedings of the Conference on High Performance Graphics 2009*. HPG '09. New Orleans, Louisiana: ACM, 2009, 77–89. ISBN: 978-1-60558-603-8. DOI: 10.1145/1572769.1572783. URL: http://doi.acm.org/10.1145/1572769.1572783 4.

[MM14] MCGUIRE, MORGAN and MARA, MICHAEL. "Efficient GPU Screen-Space Ray Tracing". *Journal of Computer Graphics Techniques (JCGT)* 3.4 (Dec. 2014), 73–85. ISSN: 2331-7418. URL: http://jcgt.org/published/0003/04/04/ 4.

[MMBJ17] MARA, MICHAEL, MCGUIRE, MORGAN, BITTERLI, BENEDIKT, and JAROSZ, WOJCIECH. "An Efficient Denoising Algorithm for Global Illumination". *ACM SIGGRAPH / Eurographics High Performance Graphics*. HPG 2017. July 2017, 7. URL: https://casual-effects.com/research/Mara2017Denoise/index.html 1, 4.

[MMNL17] MCGUIRE, MORGAN, MARA, MIKE, NOWROUZEZAHRAI, DEREK, and LUEBKE, DAVID. "Real-time Global Illumination Using Precomputed Light Field Probes". *Proceedings of the 21st ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '17. San Francisco, California: ACM, 2017, 2:1–2:11. ISBN: 978-1-4503-4886-7. DOI: 10.1145/3023368.3023378. URL: http://doi.acm.org/10.1145/3023368.3023378 4.

[MRF18] MAXIMOV, MAXIM, RITSCHEL, TOBIAS, and FRITZ, MARIO. "Deep Appearance Maps". *ArXiv* abs/1804.00863 (2018) 4.

[MS16] MANSON, JOSIAH and SLOAN, PETER-PIKE. "Fast Filtering of Reflection Probes". *Comput. Graph. Forum* 35.4 (July 2016), 119–127. ISSN: 0167-7055. DOI: 10.1111/cgf.12955. URL: https://doi.org/10.1111/cgf.12955 3.

[MSK*16] MOREAU, P., SINTORN, E., KÄMPE, V., et al. "Photon Splatting Using a View-sample Cluster Hierarchy". *Proceedings of High Performance Graphics*. HPG '16. Dublin, Ireland: Eurographics Association, 2016, 75–85. ISBN: 978-3-03868-008-6. DOI: 10.2312/hpg.20161194. URL: https://doi.org/10.2312/hpg.20161194 4.

[MSS*10] MEYER, QUIRIN, SUSSMUTH, JOCHEN, SUSSNER, GERD, et al. "On Floating-point Normal Vectors". *Proceedings of the 21st Eurographics Conference on Rendering*. EGSR'10. Saarbrücken, Germany: Eurographics Association, 2010, 1405–1409. DOI: 10.1111/j.1467-8659.2010.01737.x. URL: http://dx.doi.org/10.1111/j.1467-8659.2010.01737.x 5.

[PBD*10] PARKER, STEVEN G., BIGLER, JAMES, DIETRICH, ANDREAS, et al. "OptiX: A General Purpose Ray Tracing Engine". *ACM Trans. Graph.* 29.4 (July 2010), 66:1–66:13. ISSN: 0730-0301. DOI: 10.1145/1778765.1778803. URL: http://doi.acm.org/10.1145/1778765.1778803 5.

[Pet16] PETTINEO, MATT. *The Danger Zone: SG Series*. 2016. URL: https://mynameismjp.wordpress.com/2016/10/09/sg-series-part-1-a-brief-and-incomplete-history-of-baked-lighting-representations/ (visited on 05/07/2019) 2, 8, 9, 14.

[Rak18] RAKHTEENKO, ARTHUR. *Baking artifact-free lightmaps on the GPU*. 2018. URL: https://ndotl.wordpress.com/2018/08/29/baking-artifact-free-lightmaps/ (visited on 05/07/2019) 5.

[RDGK12] RITSCHEL, TOBIAS, DACHSBACHER, CARSTEN, GROSCH, THORSTEN, and KAUTZ, JAN. "The State of the Art in Interactive Global Illumination". *Comput. Graph. Forum* 31.1 (Feb. 2012), 160–188. ISSN: 0167-7055. DOI: 10.1111/j.1467-8659.2012.02093.x. URL: https://doi.org/10.1111/j.1467-8659.2012.02093.x 4.

[RH01] RAMAMOORTHI, RAVI and HANRAHAN, PAT. "An Efficient Representation for Irradiance Environment Maps". *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '01. New York, NY, USA: ACM, 2001, 497–500. ISBN: 1-58113-374-X. DOI: 10.1145/383259.383317. URL: http://doi.acm.org/10.1145/383259.383317 2, 3.

[RWG*13] REN, PEIRAN, WANG, JIAPING, GONG, MINMIN, et al. "Global illumination with radiance regression functions". *ACM Trans. Graph.* 32 (2013), 130:1–130:12 4.

[RWS*06] REN, ZHONG, WANG, RUI, SNYDER, JOHN, et al. "Real-time Soft Shadows in Dynamic Scenes Using Spherical Harmonic Exponentiation". *ACM Trans. Graph.* 25.3 (July 2006), 977–986. ISSN: 0730-0301. DOI: 10.1145/1141911.1141982. URL: http://doi.acm.org/10.1145/1141911.1141982 3.

[Seb14] SEBASTIEN LAGARDE, CHARLES DE ROUSIERS. *Moving Frostbite to PBR, SIGGRAPH 2014 Course: Physically Based Shading in Theory and Practice.* Aug. 2014. URL: https://seblagarde.wordpress.com/2015/07/14/siggraph-2014-moving-frostbite-to-physically-based-rendering/ 6.

[SKS02] SLOAN, PETER-PIKE, KAUTZ, JAN, and SNYDER, JOHN. "Precomputed Radiance Transfer for Real-time Rendering in Dynamic, Low-frequency Lighting Environments". *ACM Trans. Graph.* 21.3 (July 2002), 527–536. ISSN: 0730-0301. DOI: 10.1145/566654.566612. URL: http://doi.acm.org/10.1145/566654.566612 3.

[SLS05] SLOAN, PETER-PIKE, LUNA, BEN, and SNYDER, JOHN. "Local, Deformable Precomputed Radiance Transfer". *ACM Trans. Graph.* 24.3 (July 2005), 1216–1224. ISSN: 0730-0301. DOI: 10.1145/1073204.1073335. URL: http://doi.acm.org/10.1145/1073204.1073335 3.

[SWS*17] SRINIVASAN, PRATUL P., WANG, TONGZHOU, SREELAL, ASHWIN, et al. "Learning to Synthesize a 4D RGBD Light Field from a Single Image". *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), 2262–2270 4.

[SZ12] SÉBASTIEN, LAGARDE and ZANUTTINI, ANTOINE. "Local Image-based Lighting with Parallax-corrected Cubemaps". *ACM SIGGRAPH 2012 Talks*. SIGGRAPH '12. Los Angeles, California: ACM, 2012, 36:1–36:1. ISBN: 978-1-4503-1683-5. DOI: 10.1145/2343045.2343094. URL: http://doi.acm.org/10.1145/2343045.2343094 3.

[TF17] THOMAS, MANU MATHEW and FORBES, ANGUS GRAEME. "Deep Illumination: Approximating Dynamic Global Illumination with Generative Adversarial Network". *CoRR* abs/1710.09834 (2017). arXiv: 1710.09834. URL: http://arxiv.org/abs/1710.09834 5.

[TS06] TSAI, YU-TING and SHIH, ZEN-CHUNG. "All-frequency Precomputed Radiance Transfer Using Spherical Radial Basis Functions and Clustered Tensor Approximation". *ACM Trans. Graph.* 25.3 (July 2006), 967–976. ISSN: 0730-0301. DOI: 10.1145/1141911.1141981. URL: http://doi.acm.org/10.1145/1141911.1141981 2–4.

[TS92] TORRANCE, K. E. and SPARROW, E. M. "Radiometry". Ed. by WOLFF, LAWRENCE B., SHAFER, STEVEN A., and HEALEY, GLENN. USA: Jones and Bartlett Publishers, Inc., 1992. Chap. Theory for Off-specular Reflection from Roughened Surfaces, 32–41. ISBN: 0-86720-294-7. URL: http://dl.acm.org/citation.cfm?id=136913.136924 6.

[VKŠ*14] VORBA, JIŘÍ, KARLÍK, ONDŘEJ, ŠIK, MARTIN, et al. "On-line Learning of Parametric Mixture Models for Light Transport Simulation". *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2014)* 33.4 (Aug. 2014) 4.

[WMLT07] WALTER, BRUCE, MARSCHNER, STEPHEN R., LI, HONGSONG, and TORRANCE, KENNETH E. "Microfacet Models for Refraction Through Rough Surfaces". *Proceedings of the 18th Eurographics Conference on Rendering Techniques*. EGSR'07. Grenoble, France: Eurographics Association, 2007, 195–206. ISBN: 978-3-905673-52-4. DOI: 10.2312/EGWR/EGSR07/195-206. URL: http://dx.doi.org/10.2312/EGWR/EGSR07/195-206 8.

[WRG*09] WANG, JIAPING, REN, PEIRAN, GONG, MINMIN, et al. "All-frequency Rendering of Dynamic, Spatially-varying Reflectance". *ACM Trans. Graph.* 28.5 (Dec. 2009), 133:1–133:10. ISSN: 0730-0301. DOI: 10.1145/1618452.1618479. URL: http://doi.acm.org/10.1145/1618452.1618479 2–4, 8.

[WRM17] WANG, TUANFENG Y., RITSCHEL, TOBIAS, and MITRA, NILOY JYOTI. "Joint Material and Illumination Estimation from Photo Sets in the Wild". *2018 International Conference on 3D Vision (3DV)* (2017), 22–31 4.

[XCM*14] XU, KUN, CAO, YAN-PEI, MA, LI-QIAN, et al. "A practical algorithm for rendering interreflections with all-frequency BRDFs". *ACM Trans. Graph.* 33 (2014), 10:1–10:16 4.

[XSD*13] XU, KUN, SUN, WEI-LUN, DONG, ZHAO, et al. "Anisotropic Spherical Gaussians". *ACM Trans. Graph.* 32.6 (2013), 209:1–209:11 3, 4.

[ZBLN97] ZHU, CIYOU, BYRD, RICHARD H., LU, PEIHUANG, and NOCEDAL, JORGE. "Algorithm 778: L-BFGS-B: Fortran Subroutines for Large-Scale Bound-Constrained Optimization". *ACM Trans. Math. Softw.* 23.4 (Dec. 1997), 550–560. ISSN: 0098-3500. DOI: 10.1145/279232.279236. URL: https://doi.org/10.1145/279232.279236 4.

[ZTF*18] ZHOU, TINGHUI, TUCKER, RICHARD, FLYNN, JOHN, et al. "Stereo Magnification: Learning View Synthesis Using Multiplane Images". *ACM Trans. Graph.* 37.4 (July 2018), 65:1–65:12. ISSN: 0730-0301. DOI: 10.1145/3197517.3201323. URL: http://doi.acm.org/10.1145/3197517.3201323 4.

**Appendix A:** Partial Derivatives for back propagation

We use the $L_2$ log loss function when training, defined as follows:

$$L(P) = (\log(T+1) - \log(P+1))^2, \tag{9}$$

where $P$ is the predicted value for a pixel in a light-field image, and T is the target value. The gradient of the $L_2$ log loss function with respect to this pixel is:

$$\frac{dL(P)}{dP} = -2\frac{\log(1+T) - \log(1+P)}{1+P} \tag{10}$$

However, we need to backpropagate the gradient of the $L_2$ loss with respect to each *parameter*. The chain rule gives us that the gradient of a specific parameter $\mathbf{p}_k$ is:

$$\frac{dL}{d\mathbf{p}_k} = \frac{1}{IC}\sum_{i,c}\frac{dP_{ic}}{d\mathbf{p}_k}\frac{dL(P_{ic})}{dP_{ic}}, \tag{11}$$

where $I$ is the total number of pixels and $C$ is the number of channels. Hence, for each pixel and channel, we must find the partial derivative of Eq 1 with respect to each of the parameters and add that contribution to the parameter's gradient. For the different parameter types, these derivatives are:

$$\frac{dP_{ic}}{d\mu_{jc}} = e^{\lambda_j(\mathbf{v}_i\cdot\mathbf{p}_j-1)} \tag{12}$$

$$\frac{dP_{ic}}{d\lambda_j} = \mu_{jc}e^{\lambda_j(\mathbf{v}_i\cdot\mathbf{p}_j-1)}(\mathbf{v}_i\cdot\mathbf{p}_j-1) \tag{13}$$

$$\frac{dP_{ic}}{dx} = \mu_{jc}e^{\lambda_j(\mathbf{v}_i\cdot\mathbf{p}_j-1)}\lambda_j x \text{ (identically for } y \text{ and } z) \tag{14}$$

**Appendix B:** BRDF as Spherical Gaussians and SG Convolution

In the real-time rendering of the reflection SGs we treat the BRDF function as a spherical gaussian to be able to convolve it with the spherical gaussians describing the incoming light field for a texel. These formulas have been adapted from [Pet16], where an in-depth explanation on how they are derived is also given.

We use the Cook-Torrance BRDF:

$$f(\omega_i,\omega_o) = \frac{F(\omega_o,\omega_h)\,G(\omega_i,\omega_o,\omega_h)\,D(\omega_h)}{4\,(\mathbf{n}\cdot\omega_i)\,(\mathbf{n}\cdot\omega_o)}, \tag{15}$$

and the following definition of a Spherical Gaussian:

$$G(\mathbf{v};\mu,\lambda,a) = ae^{\lambda(\mu\cdot\mathbf{v}-1)}. \tag{16}$$

The SG approximation to the $D$ term that we use is defined as follows:

$$D(\omega_h) = e^{-(arccos(\omega_h\cdot\mathbf{n})/r)^2} \approx G(\omega_h;\mathbf{n},\frac{2}{r^2},\frac{1}{\pi r^2}), \tag{17}$$

where r is the roughness of the material.

This gaussian is defined in the half-vector domain, and we need to convert it to the same domain as the SG it will be convolved with. To better represent the BRDF from directions approaching the surface plane we use an anisotropic transformation of the previous lobe in this step, as suggested by [Pet16]. For that, the following transformations are used:

$$\begin{aligned}\mu_w &= 2(\omega_o\cdot\mu_d)\mu_d - \omega_o\\ \lambda_w^x &= \frac{\lambda_d}{8\;max(\mu_d\cdot\omega_o, 0.0001)^2}\\ \lambda_w^y &= \frac{\lambda_d}{8}\\ a_w &= a_d\end{aligned} \tag{18}$$

This anisotropic spherical gaussian can be evaluated with:

$$G(\mathbf{v};[\mu_x,\mu_y,\mu_z],[\lambda_x,\lambda_y],a) = \tag{19}$$

$$= a\cdot max(\mathbf{v}\cdot\mu_z,0)e^{-\lambda_x(\mathbf{v}\cdot\mu_x)-\lambda_y(\mathbf{v}\cdot\mu_y)} \tag{20}$$

Where $\mu_x$ and $\mu_y$ are two orthogonal vectors that form a basis with $\mu_z = \mu_w$. Any will do, and they need to be transformed together with $\mu_z$ when applying equation 18.

We can convolve two spherical gaussians $G_1(\mathbf{v})$ and $G_2(\mathbf{v})$ as follows:

$$\int_\Omega G_1(\mathbf{v})G_2(\mathbf{v})d\mathbf{v} = \frac{4\pi a_1 a_2}{e^{\lambda_m}}\frac{sinh(||\mu_m||)}{||\mu_m||} \tag{21}$$

Where

$$\lambda_m = \lambda_1 + \lambda_2 \tag{22}$$

$$\mu_m = \frac{\lambda_1\mu_1 + \lambda_2\mu_2}{\lambda_1+\lambda_2} \tag{23}$$

To convolve each isotropic SG from the light-field texture with the anisotropic SG approximating the $D$ term, we use:

$$\begin{aligned}&\int_\Omega G_1(\mathbf{v};\mu_1,\lambda_1,a_1)\cdot G_2^A(\mathbf{v};\mu_2,[\lambda_2^x,\lambda_2^y],a_2)d\mathbf{v} =\\ &= \frac{a_1a_2\pi}{\sqrt{(\frac{\lambda_1}{2}+\lambda_2^x)(\frac{\lambda_1}{2}+\lambda_2^y)}}max(\mu_2^z\cdot\mathbf{v})e^{-(\lambda_2^x(\mathbf{v}\cdot\mu_2^x)^2+\lambda_2^y(\mathbf{v}\cdot\mu_2^y)^2)}.\end{aligned} \tag{24}$$