INTERPOLATING SPECULAR HIGHLIGHTS FROM REFLECTANCE FIELD DATA

Using Gaussians to interpolate specular highlights in a Lagrangian frame of reference

by

ANDOR MICHELS

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science at the Delft University of Technology, to be defended publicly on Thursday, December 4, 2025 at 14:30.

November 2025

Andor Michels – *Interpolating specular highlights from reflectance field data,*
Using Gaussians to interpolate specular highlights in a Lagrangian frame of reference, November 2025

SUPERVISORS:

Ricardo Marroquim	TU Delft
Elmar Eisemann	TU Delft
Baran Usta	TU Delft
Ruben Wiersma	ETH Zurich

ABSTRACT

Reflectance fields have a limited number of discrete lighting directions that can be used for lighting design. Multiple methods for interpolating these reflectance fields have been proposed to get an approximation of the missing lighting directions. However, these methods can struggle with specular highlights, because they are more sensitive to changes in lighting direction. We propose a method that solves this problem by representing the specular highlights as Gaussians and interpolating them, framing the problem as a Lagrangian formulation. We interpolate these Gaussians by optimizing their parameters twice: first for each lighting direction, then for every neighboring lighting direction. By doing this, we can efficiently interpolate specular highlights in real-time. This enables the designer to design the specular highlights and their movements with immediate feedback.

CONTENTS

LIST OF SYMBOLS

I_i	Reflectance field image	7
I'	Interpolated result image	7
ω_i	Light direction for I_i	7
ω'	Polled light direction to interpolate	7
G_i	Gaussian set representing S_i	7
G'	Interpolated Gaussian set	7
$G_{i\to j}$	Gaussian set G_i retargeted on S_j	7
S_i	Specular highlight component of I_i	7
S'	Interpolated specular component	9
D_i	Diffuse component of I_i	9
D'	Interpolated diffuse component	9
$\tilde{D}_i$	Approximation of D_i	9
x, y	x and y image coordinates	10
$n(x, y)$	Object normal at pixel coordinate (x, y)	10
$k_d(x, y)$	Diffuse intensity at pixel coordinate (x, y)	10
$k_a(x, y)$	Ambient intensity at pixel coordinate (x, y)	10
$\bar{n}(x, y)$	Normal $n(x, y)$ multiplied by the diffuse intensity $k_d(x, y)$	10
t_{seg}	Threshold value for image segmentation	10
L, L'	Set of valid lighting directions for image segmentation	10
l_i	Segmentation loss for I_i	10
$\tilde{m}$	Median loss value	10
a_p	Amplitude of Gaussian p	12
$\boldsymbol{\mu}_p$	Mean (position) of Gaussian p	12
$\boldsymbol{\sigma}_p$	Standard deviation of Gaussian p	12
σ_x, σ_y	x and y components of $\boldsymbol{\sigma}$	17
θ_p	Rotation of Gaussian p	12
$\mathcal{L}_{est}$	Estimation loss function	12
$\mathcal{L}_{ret}$	Retargeting loss function	17
$\mathcal{L}'_{fit}$	Image fitting part of the retargeting loss function	15
$\mathcal{L}'_{str}$	Structure preservation part of the retargeting loss function	16

$\mathcal{L}'_{ene}$	Energy preservation part of the retargeting loss function	17
$\mathcal{G}(\mathbf{x}, G)$	Gaussian rasterization function	12
$g(\mathbf{x})$	Gaussian function	13
t_{est}	Threshold value for creating new Gaussians	13
Σ	Covariance matrix	14
R_θ	Rotation matrix for rotation θ	14
Λ	Diagonal matrix with squared standard deviations	14
$h(x)$	Limit function	16
γ	Power of change in the limit function	16
m	Position of change in the limit function	16
ρ_{pq}	Euclidean distance between Gaussians p and q	16
s_{ij}	Scale difference between G_i and $G_{i\to j}$	16
p	Gaussian in a Gaussian set	12
$\hat{p}$	Gaussian in Gaussian set G_i corresponding to p in retargeted Gaussian set $G_{i\to j}$	16
$c_{fit}, c_{str}, c_{ene}$	Weights used to balance l_{fit}, l_{str}, and l_{ene}	17
$\ddot{\mu}, \ddot{\theta}$	Mean (position) and rotation of Gaussian connected components	18
A, B, C	Interpolation triangle vertices	19
$\lambda_A, \lambda_B, \lambda_C$	Barycentric coordinates corresponding to triangle vertices A, B, and C	19
O	Interpolation circle center	22
$\theta^{\Delta}_{A\to B}$	Difference in rotation between G_A and $G_{A\to B}$	22
$\bar{\theta}$	Angle between the direction of a Gaussian and the direction from the Gaussian to circle center O	23
w, w'	Weight used to determine which direction the interpolation rotation will be	22
ξ	Inverse distance parameter in propagation matrix	23
η_p	Total rotation propagation from Gaussian p	23
ϕ_{pq}	Rotation parity between Gaussians p and q	23
M	Propagation matrix for deciding interpolation rotation direction	23

1 INTRODUCTION

Specular highlights are vital for the perception of an object and the context the object is in, since they give visual cues on the shape of an object and sources of light surrounding it (Fleming et al., 2004; Norman et al., 2004). The movement of these specular highlights can also be very effective in adding additional visual cues, since more of the shape of the object is accentuated. Lighting designers can use these visual cues to highlight aspects of an object to make it look more attractive or to associate certain feelings with the object. Examples can be a specular highlight on the edge of a new watch, giving feelings of luxury and sophistication, or highlights moving quickly along the lines of a new sports car, giving associations with speed and aggression. Therefore, being able to design these highlights is highly desirable.

Figure 1.1: Example image of a specular highlight used to highlight the shape of an object. Photograph by Guy Sie (2009).

However, to be able to design these highlights, a designer needs complete control of the light sources' positioning and shape, which requires a lot of skill, labor, and expensive lighting setups.

A possible tool for additional control is the use of reflectance fields. Reflectance fields are sequences of images recorded on a light stage, and store light reflectance information at every point on an object from different light directions. The use of reflectance fields allows the subject to be recorded only once and the lighting design to be done afterward, giving more flexibility to the designer without needing continuous access to the subject or the lighting stage. Over the years, much research has gone into developing techniques that make the acquisition of reflectance fields easier (Debevec et al., 2000; Kinsman, 2016; Maček et al., 2022; Masselus et al., 2002), making reflectance fields more accessible than they used to be.

However, reflectance field data is limited by their sampling density: the number of lighting directions available. The sampling density is often limited to decrease the cost of equipment and the needed data storage (Fuchs et al., 2007). Especially specular highlights suffer from this lower density, since they are more reactive to changes in lighting direction. This sampling density can be increased using different reflectance field interpolation techniques, which can then be used in combination with a light map to place the subject in a different lighting situation. However, these techniques focus primarily on softer lighting (Debevec & LeGendre, 2022), where sharp specular highlights are less important.

When designing moving highlights, this lower sampling density is problematic, since it would cause specular highlights to jump from one known lighting direction to another. The appearance of the specular highlights at the intermediate lighting directions is unknown. Techniques exist to increase the sampling density by interpolating the appearance from existing lighting directions (B. Chen & Lensch, 2005; Fuchs et al., 2007). However, they still primarily focus on creating denser reflectance fields, without allowing for querying specific lighting directions. These methods can be extended to query single lighting directions, but they may require significant processing for each queried lighting direction, which makes them unsuitable for real-time feedback for interactive lighting design.

This work proposes a method for interpolating reflectance field data, specifically focused on moving specular highlights smoothly during a video with high user controllability in real-time. The goal is to make it easy to query the appearance of lighting directions in such a way that subsequent images with slowly changing lighting directions show naturally moving specular highlights.

We do this by representing the specular highlights with 2D Gaussians and interpolating these to get the unknown lighting directions. Gaussians are a natural choice, since they can approximate the shape of a specular highlight very well. This way, we change the domain of the problem from an Eulerian representation where pixels have an amount of highlightness, to a Lagrangian representation, where the specular highlights are built of components having position and shape. Other methods like Gaussian splatting (Kerbl et al., 2023) use a similar representation to make manipulation of the components intuitive and fast.

We present a novel method of interpolating these Gaussians, where we preserve the shape and movement direction of specular highlights. This method is computationally fast, so it can be used for immediate feedback when designing specular highlights.

2 RELATED WORK

REFLECTANCE FIELD ACQUISITION In our pipeline, we work with reflectance fields. Reflectance fields (Debevec et al., 2000) describe the relation between incoming and outgoing light around the surface of an object. This relation is described using a four- or higher-dimensional function (two incoming light directions and two outgoing directions or pixel coordinates), and more dimensions can be added depending on the needed flexibility in lighting and viewpoint (e.g., camera position). Generally, 4D reflectance fields only allow for changes in lighting situations, while higher-dimensional functions allow for novel view synthesis. Related representations, such as light fields (Levoy & Hanrahan, 1996) or the Lumigraph (Gortler et al., 1996), are usually 4D or 5D functions that allow for viewpoint changes but assume fixed lighting.

The first reflectance fields were acquired by rotating a light source around the object and using one or more cameras to capture the light reflecting off the object (Debevec et al., 2000). Later methods use a sophisticated light stage consisting of a dome with uniformly placed light sources surrounding the object and high-speed cameras to capture live performances (Guo et al., 2019). Garg (2006) developed a setup where reflectance fields can be acquired more efficiently by projecting patterns that utilize the independence of elements in the light transfer function. This allows for significantly faster acquisition of 8D reflectance fields.

For applications only focused on relighting, like this thesis, many researchers have used 4D reflectance fields captured with a fixed camera and a point light as a light source. Kinsman (2016) provides an easy-to-build, dome-shaped system for capturing reflectance fields of small objects. Malzbender et al. (2001) use a similar system to capture light-direction-dependent textures. Schechner et al. (2003) proposes a method for multiplexing the lighting directions to improve image quality. Masselus et al. (2002) developed a method for capturing reflectance fields using a handheld light source, making the acquisition more affordable. Maček et al. (2022) created a similar system using a smartphone flashlight as the light source. Our pipeline works with these more easily obtainable reflectance fields, but we mainly focus on interpolating the reflectance fields rather than acquiring the reflectance field itself.

REFLECTANCE FIELD INTERPOLATION Because of practical limitations, acquired reflectance fields are often relatively sparse. Lower-

dimensional reflectance fields are often used where possible to lower the complexity of acquisition, while the number of light directions is limited to speed up acquisition. There have been several attempts at resolving this issue. A popular method is to represent the reflectance field using a continuous function. Malzbender et al. (2001) use a polynomial per pixel to model the reflectance of surfaces under varying lighting conditions, allowing for the computation of the reflectance under arbitrary lighting directions. They initially ignore specular lighting in their acquisition and add it at a later stage using a separately acquired bump map.

Similarly, Maček et al. (2022) interpolate a very sparse and uneven reflectance field of a person using a model of a human head. They use this model to detect sharp geometric transitions that may cause occlusions and to calculate approximate surface normals. Both of these systems would require either a known geometry or a separate recording of a bump map, which our system does not have.

Recently, there have been several attempts at representing reflectance fields using machine learning models (Bi et al., 2020; Meka et al., 2019; Sun et al., 2020; Xu et al., 2018), encoding the reflectance field in a black box model. These methods can significantly decrease the number of required lighting directions, but they require extensive training. The specular highlights on the resulting interpolated images are generally less defined than the compared ground truth, and do not provide any tools for directly manipulating specular highlights.

More conventional direct interpolation methods for reflectance fields also exist. B. Chen and Lensch (2005) propose a method focusing on interpolating shadows between different lighting directions. They assume minor illumination differences between neighboring light directions except for shadows, ignoring specular highlights. This way, they separate the shadows from more static elements and use an optical flow scheme to interpolate shadows. Fuchs et al. (2007) show that a similar optical flow scheme can be applied for specular highlights. They can interpolate any novel lighting direction from three surrounding known lighting directions using a double linear interpolation. These interpolation schemes are relatively expensive to compute. This can work well for static lighting situations where unknown lighting directions can be pre-defined, but may be slow for dynamic lighting direction querying.

GAUSSIAN REPRESENTATION This thesis proposes using Gaussian functions to approximate the specular highlights for interpolation. Kerbl et al. (2023) have shown that Gaussians can be used for representing real-world 3D scenes. They also provide a differentiable Gaussian rasterizer, allowing for efficient optimization of the Gaussian representation. E. Chen et al. (2025) use a similar technique, optimizing Gaussians in 2D image space for interpolating between stitched

images taken at different times. They also split the image into separate reflectance and shading components using available metadata, e.g., time, sun position, and cloud coverage. Zhang et al. (2025) use 2D Gaussians to represent and compress full images. These methods show that images can be efficiently represented using Gaussians, where the Gaussians describe certain features of the environment. We will use similar techniques to represent only the specular component of the images and interpolate between them.

3 METHOD

We present a method for the efficient interpolation of specular highlight images for any lighting direction. An overview of the method is presented in Figure 3.1.

The input of our method is the reflectance field data of the object, consisting of the set of reflectance field images R captured under lighting directions d. We let I_i and ω_i denote individual reflectance field images with their corresponding lighting direction. We also need a single lighting direction ω' that we want to query, which only needs to be known during the interpolation phase.

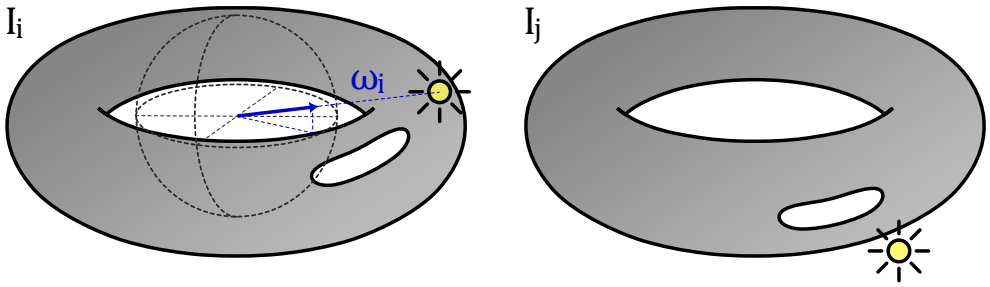
For simplicity, we assume that the lighting in the reflectance field images comes from lights at a sufficient distance to resemble directional lights. If the light sources are too close, they can cause divergence of lighting angles across the object for a single lighting direction.

We also assume the light sources to have no inherent shape. As long as the actual shape of the light sources is close to circular, we do not expect this to have a significant effect on the highlight direction assumptions made later in this thesis. The reflectance field images should also be radiometrically linear.

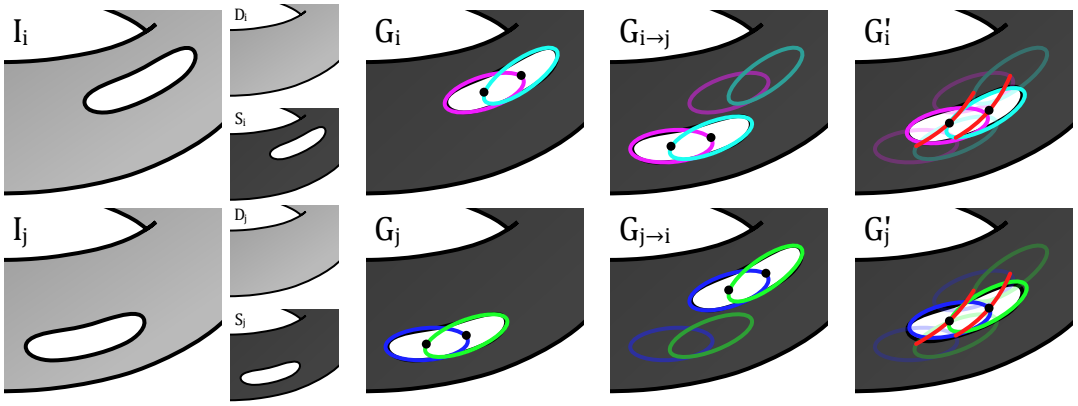
Our output is an interpolated image I' showing the appearance of the specular highlights under the queried lighting direction ω'. We achieve this result by extracting the specular highlight image S_i for each known lighting direction ω_i, representing these specular highlights as sets of Gaussians G_i and moving these to create the set of Gaussians G' corresponding to the queried lighting direction ω'. This way, we transform the data from an Eulerian formulation to a Lagrangian formulation, where we treat the specular highlights as objects with properties like position and shape. This helps with efficient and intuitive interpolation of the specular highlights.

We perform the interpolation by applying a barycentric interpolation scheme using the Gaussians that represent the different specular highlight images. Specifically, we interpolate the Gaussian parameters of neighboring Gaussian sets. However, one set of Gaussians cannot easily be interpolated with another set of Gaussians, since it is not trivial to identify which Gaussians in neighboring sets correlate to each other, and the number of Gaussians per set can differ.

We solve this problem by retargeting all sets of Gaussians G_i on the neighboring specular highlight images S_j to create additional Gaussian sets $G_{i\to j}$, as seen in Figure 3.1. We do this while keeping the same number of Gaussians and a similar structure. This does result in a worse fit on the specular highlight image, but these additional sets



(a) Input Images

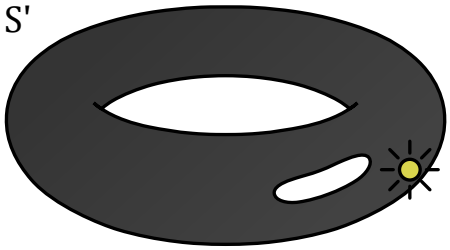


(b) Image segmentation

(c) Gaussian estimation

(d) Gaussian retargeting

(e) Gaussian interpolation



(f) Interpolated specular highlight

Figure 3.1: Illustrations representing the full flow of our method for interpolating specular highlights. **(a)** shows the input images (I_i, I_j) under different lighting directions (ω_i, ω_j). **(b)** shows the segmentation of these images into diffuse (D_i, D_j), and specular components (S_i, S_j). **(c)** shows the fitting of Gaussian sets (G_i, G_j) on the specular components (S_i, S_j). **(d)** shows the re-fitting of these Gaussian sets on the neighboring specular components (S_j, S_i). **(e)** shows the interpolation of these Gaussian sets to get interpolated Gaussians (G'_i, G'_j). **(f)** shows the resulting interpolated specular highlight (S').

can be used in combination with the original Gaussian set for the interpolation.

This results in the following steps for our method, as seen in Figure 3.1:

1. Apply **image segmentation** to extract the specular highlights;
2. **Estimate** Gaussians to represent the specular highlights;
3. **Retarget** the Gaussians on neighboring light directions;
4. **Interpolate** and rasterize the Gaussians.

For simplicity, all computations with images in this thesis are done on grayscale images, not considering color, though this pipeline can be easily extended to support color.

3.1 IMAGE SEGMENTATION

As established before, diffuse reflection and specular highlights react differently to moving light sources. Therefore, we want to treat them separately. In the remainder of this thesis, we focus on interpolating specular highlights between frames and assume we can use pre-existing methods for diffuse interpolation.

To treat these components separately, we must first isolate them. This involves decomposing each reflectance field image I_i into its diffuse image D_i and specular highlight image S_i such that $I_i = S_i + D_i$. Similarly, we can create an interpolated reflectance field image I' using interpolated components S' and D', although the reconstruction of D' is not covered by this thesis.

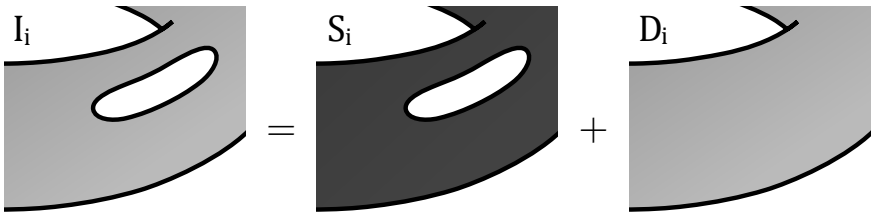


Figure 3.2: The reflectance field image I_i can be decomposed into specular highlight image S_i and diffuse image D_i.

These diffuse images contain all reflections that remain relatively stable under moving light sources. This means we also include elements of indirect illumination.

We extract the specular highlights by creating a slightly brightened approximation of the diffuse component $\tilde{D}_i$. We use this approximation to split the image using

$$D_i = \min(I_i, \tilde{D}_i) \tag{3.1}$$

$$S_i = I_i - D_i, \tag{3.2}$$

isolating the specular highlights.

To get a reasonably accurate approximation of the diffuse component, we assume the diffuse reflection to be ideal Lambertian reflection

and therefore follow Lambert's cosine law, with some additional ambient reflection to account for inaccuracies and to capture indirect illumination. As demonstrated by Nayar et al. (1990), modeling a purely Lambertian reflectance component using a cosine creates a good representation from which geometric information like surface normals can be extracted.

Per pixel, we calculate the diffuse component as a cosine by using the formula

$$\tilde{D}_i(x,y) = k_d(x,y)n(x,y) \cdot \omega_i + k_a(x,y), \tag{3.3}$$

where we have the unknowns $n(x,y)$ as the normal of the object at the location shown in the pixel at (x,y), and $k_d(x,y)$ and $k_a(x,y)$ as the diffuse and ambient intensities at position (x,y). We can combine the diffuse intensity and normal into a single term $\bar{n}(x,y)$ to get the simplified formula

$$\tilde{D}_i(x,y) = \bar{n}(x,y) \cdot \omega_i + k_a(x,y). \tag{3.4}$$

If we have four or more valid lighting directions, we can find optimal values for $\bar{n}(x,y)$ and $k_a(x,y)$ using linear least squares to recreate I_i. However, we cannot use all available lighting directions, since lighting directions that light the pixel from the back will have a negative value for Equation 3.4. Therefore, we exclude any lighting directions where $I_i(x,y) < t_{seg}$, where we empirically set $t_{seg} = 0.05$. t_{seg} needs to be higher than 0 to counteract indirect lighting, but low enough to include as many valid lighting directions as possible. This leaves us with the set of valid lighting directions L.

Even with the potentially negative approximation values ignored, some pixel values in I_i may contain specular highlights or shadows that would skew the fitting. Assuming these values are in the minority, we can filter them out by further restricting the set of used lighting directions in an iterative fashion. For each direction, we compute the current loss $l_i = |I_i - \tilde{D}_i|$, then determine their median value $\tilde{m}$. We now recalculate $\tilde{D}_i(x,y)$ using a new set of lighting directions L' where we only keep the directions if $l_i \leqslant 2\tilde{m}$. We continue this until m converges, i.e., until L' stops changing.

The resulting $\tilde{D}_i(x,y)$ is an approximation of the diffuse lighting, which needs to be transformed into an upper bound. We use the function

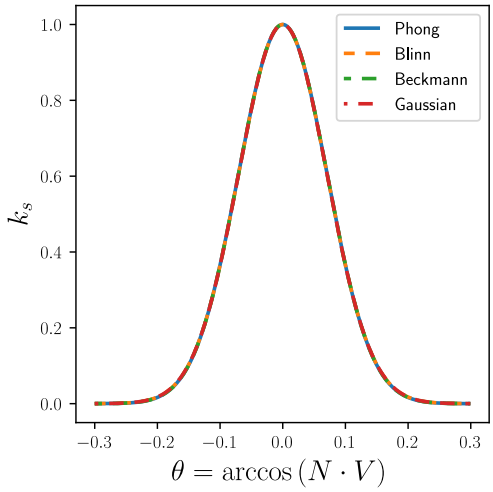
$$\tilde{D}_i = \max(\tilde{D}_i, 0.05) + 0.01. \tag{3.5}$$

This adjusted approximation is then used in Equation 3.1 to extract the final diffuse and specular components.

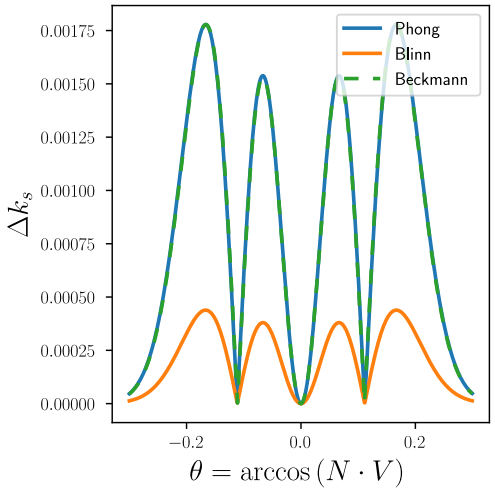
This method is specifically tailored to reflectance field data, since it uses the lighting information of all different lighting directions, making it invariant to object color, unlike image decomposition methods

that only rely on a single image. It also performs well for our dataset. However, in practice, any intrinsic image decomposition method can be used, like the method developed by Fuchs et al. (2007).

3.2 GAUSSIAN ESTIMATION



(a) Specular response k_s for different reflectance models compared to a Gaussian.



(b) Absolute differences between the reflectance model responses and the Gaussian.

Figure 3.3: Specular intensity responses k_s from different reflectance models compared to a Gaussian **(a)** and the difference Δk_s between these models and the Gaussian **(b)**. Model and Gaussian parameters were chosen to be as close to the Phong shading model with exponent $n = 50$. θ is the angle between the view direction V and the surface normal direction N. The view direction and light direction were chosen to be identical for simplicity. The resulting Gaussian is at most around 0.18% off the other models.

We use a set of Gaussians to approximate the specular highlight image. Gaussians are sometimes used in specularity models because they can accurately describe the distribution of the orientations of microfacets that cause the specular highlight (Beckmann & Spizzichino, 1963). As can be seen in Figure 3.3, specularity from different distribution functions by Phong (1975), Blinn (1977), and Beckmann and Spizzichino (1963) can easily be approximated using a single Gaussian. This means that, for specular highlights on simple geometries, the main structure of the highlight can be approximated well using a single Gaussian, as opposed to multiple overlaying Gaussians at the same position needed to approximate the shape of more complex functions. This means that we do not need to worry about using multiple Gaussians to model the natural falloff of specular highlights, and

multiple Gaussians only need to be used to model the larger structure of the specular highlight.

An additional advantage of using Gaussians is that the shape and direction of the Gaussians correlate with the magnitude and direction of curvature of the underlying geometry. We observe that the more curved the geometry is, the narrower the specular highlight will be in that direction. In turn, the fitted Gaussians will be more elongated and oriented orthogonal to the curvature, as can be seen in Figure 3.4. So this direction and the dimensions of the Gaussian correspond to the principal directions and principal curvature of the geometry. With this information, we can extrapolate in which direction specular highlights will be moving when a light source is moved, which we will discuss in Section 3.4.1.

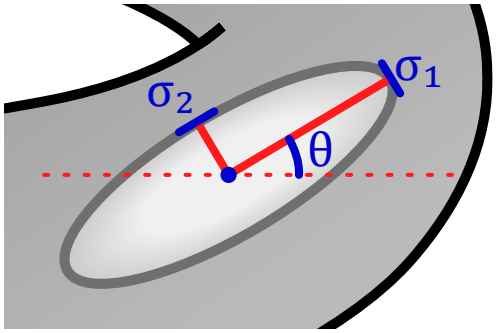


Figure 3.4: A specular cylinder showing the principal direction aligning with θ and the size of σ showing high and low principal curvature.

We represent the specular highlight images S_i with sets of Gaussians G_i by optimizing a number of Gaussians to fit the specular highlight images. We minimize the loss function

$$\mathcal{L}_{\text{est}}(G_i) = \sum_{x,y} \left| \log_2 \left(\mathcal{G}\left([x,y]^T, G_i\right) + 1 \right) - \log_2 \left(S_i\left(x,y\right) + 1 \right) \right|, \quad (3.6)$$

where we define $S_i(x,y)$ as the pixel value of the specular highlight image at position (x,y). We also define the Gaussian rasterization function

$$\mathcal{G}(\mathbf{x}, G) = \sum_{p \in G} g(\mathbf{x}; a_p, \boldsymbol{\mu}_p, \boldsymbol{\sigma}_p, \theta_p), \quad (3.7)$$

with Gaussian parameters a_p, $\boldsymbol{\mu}_p$, $\boldsymbol{\sigma}_p$, and θ_p for Gaussian p in Gaussian set G. We optimize for these Gaussian parameters using the Adam optimizer (Kingma, 2014).

We are mainly focused on matching the shape of the specular highlights rather than matching the highest values. The higher values will often be clipped to the maximum brightness in the final output. Additionally, the shape of the highlight is primarily derived from the highlight values between 0 and 1, whereas values higher than 1 do not contain as much visual information. Therefore, we use the

transformation function $\log_2(x+1)$. This function has the desirable property that $x \approx \log_2(x+1)$ for $x \in [0,1]$, while for larger values of x, the resulting values increase more slowly. Because of this, matching the shape has a higher weight than matching the peaks.

Additionally, this conforms to Fechner's law on how human perception is logarithmic in relation to external stimuli (Fechner, 1948). Consequently, the resulting difference values are closer to how humans would perceive this difference.

When we later want to interpolate these Gaussians, we will allow them to move partially independently from each other. This has the disadvantage that these Gaussians can move away from each other and cause tearing: gaps that form when the Gaussians do not move at the same speed. An example is shown in Figure 3.5. To prevent this, we want to represent the specular highlights with as few Gaussians as possible. This decreases the number of possible places where tearing can occur, decreases the chance of Gaussians re-ordering themselves, and increases the necessary coverage of each Gaussian, meaning the individual Gaussians better represent prominent features of the specular highlights. This also means that we get more information about the curvature of the object encoded in each Gaussian, giving us more accurate information to base the movement of the specular highlights on, decreasing the chance of tearing further.

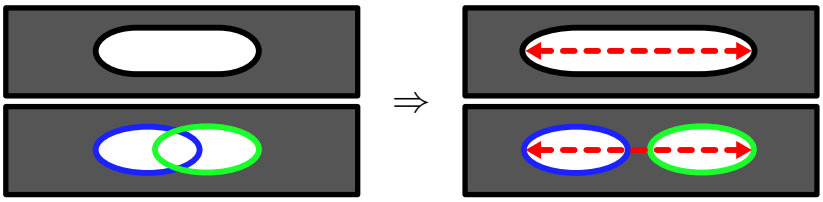


Figure 3.5: Gaussians can move independently during retargeting. In some circumstances, this can cause gaps to form between Gaussians.

To achieve this, we dynamically limit the number of Gaussians in the Gaussian sets. This is done by iteratively adding Gaussians at different times during the optimization. These Gaussians are positioned where the missing pixel value exceeds a threshold t_{est}, and start with a size calculated to cover as much of the missing area as possible. The optimization process will grow or shrink the Gaussians to fit, so it is not crucial to have the initial Gaussian parameters at the optimal values. Similarly, Gaussians are removed that fall below a threshold for contributing to the loss function. This way, the number of Gaussians is limited while creating a set that completely covers the specular highlights.

Throughout this thesis, we use 2D Gaussians of the form

$$g(\mathbf{x}) = a^2 \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}-\boldsymbol{\mu})\right), \tag{3.8}$$

where a^2 is the amplitude, $\boldsymbol{\mu}$ is the position, and Σ is the covariance matrix of the Gaussian.

In this formulation, the covariance matrix Σ is required to be positive semi-definite to have meaning. When performing an unconstrained optimization on the values of Σ, this can cause problems, as noted by Kerbl et al. (2023). The individual values of the covariance matrix also have very little physical meaning as they relate to the specular highlights. To combat these issues, we build the covariance matrix

$$\Sigma = R_\theta \Lambda R_\theta^T \tag{3.9}$$

$$\Lambda = \operatorname{diag}\left(\boldsymbol{\sigma}^2 + \epsilon\right), \tag{3.10}$$

where R_θ is the rotation matrix for rotation angle θ, $\boldsymbol{\sigma}^2$ is the variance, and ϵ is a small constant added to avoid division by 0 while allowing for unconstrained optimization. This covariance matrix will always be positive semi-definite and allows us to manipulate physical characteristics like size and rotation directly.

That leaves us with the parameters a for amplitude, $\boldsymbol{\mu}$ for position, $\boldsymbol{\sigma}$ for size, and θ for rotation to optimize.

Contrary to conventional Gaussian definitions, we use a^2 to ensure positive Gaussians without additional methods. A negative amplitude can make sense for other image representation methods, but when moving these Gaussians at a later stage, this can be detrimental to image stability.

These Gaussians are fully differentiable and quick to compute on a GPU, making it easy to use standard optimization techniques like gradient descent. An optimized differentiable 3D Gaussian rasterizer for Gaussian splatting is presented by Kerbl et al. (2023). We created a similar differentiable rasterizer for 2D Gaussians in CUDA.

3.3 GAUSSIAN RETARGETING

We need to find out where the Gaussians need to be moved to during interpolation to create a good interpolated specular highlight image. Since there is no one-to-one relation between the Gaussians in different Gaussian sets after the estimation step, we cannot directly interpolate their parameters, and we need to find a different way to move each Gaussian. We do this by creating Gaussian sets $G_{i\to j}$ that have the same number of Gaussians and a similar structure as Gaussian set G_i but are retargeted to the specular highlight image S_j as shown in Figure 3.6. Since this creates a one-to-one relation between Gaussians in the different sets, these Gaussians can be interpolated between each other.

To understand which Gaussian sets need to be retargeted on each other, we apply a spherical Delaunay triangulation to all known lighting directions. We say that two Gaussian sets G_i and G_j are neighbor-

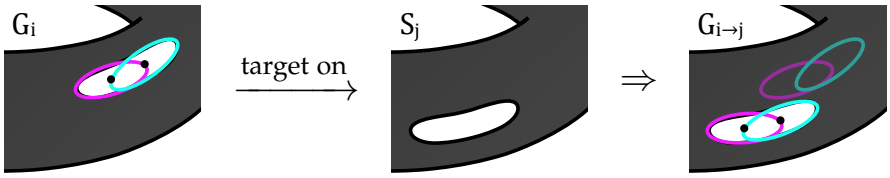


Figure 3.6: Gaussians in Gaussian set G_i get their parameters optimized to represent specular highlight image S_j. This results in the retargeted Gaussian set $G_{i\to j}$.

ing iff their respective lighting directions ω_i and ω_j share an edge in this Delaunay triangulation. All neighboring Gaussian sets need to be retargeted on all their neighboring specular highlight images. This way, every triangle in the Delaunay triangulation will have all its vertices retargeted on each other, which is needed for the barycentric interpolation scheme in Section 3.4.

We perform the retargeting by optimizing the Gaussian set G_i again on the specular highlight image S_j to create the new Gaussian set $G_{i\to j}$. Like in Section 3.2, we optimize all available Gaussian parameters using gradient descent on a similar loss function. However, in this situation, we need to focus on more aspects:

- **Similarity to the target specular image**: The Gaussians should resemble the specular highlight image;
- **Similar relative structure**: During interpolation, the structure and shape of the Gaussian source and target sets should be as similar as possible to ensure that there is a smooth change;
- **Similar purpose**: A Gaussian should represent a similar aspect of specular highlight, to ensure it still represents that aspect during the interpolation.

3.3.1 Similarity to the target specular image

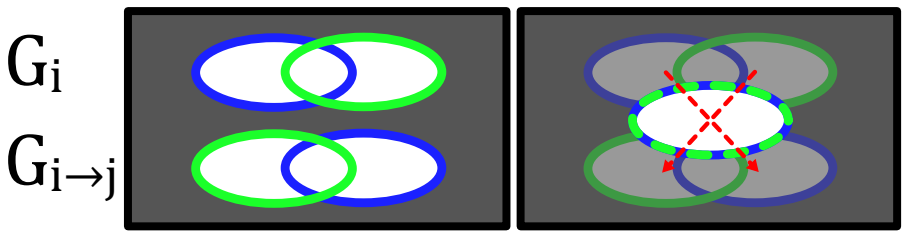
To move the Gaussians to their new target positions, we use the same loss function as seen in Section 3.2:

$$\mathcal{L}'_{\text{fit}}(G_{i\to j}) = \sum_{x,y} \left|\log_2\left(\mathcal{G}\left([x,y]^T, G_{i\to j}\right)+1\right) - \log_2\left(S_i\left(x,y\right)+1\right)\right| \tag{3.11}$$

3.3.2 Similar relative structure

Without additional constraints, the paths of the retargeted Gaussians may cross over each other, causing the interpolated positions of the Gaussians halfway through the interpolation to collapse and form an unrelated highlight shape as seen in Figure 3.7. To combat this, we

introduce a loss on the change of the distance between individual Gaussians.



(a) G_i and $G_{i\to j}$ with a different order.

(b) The interpolation paths of the Gaussians, with the interpolated Gaussians.

Figure 3.7: When the order of the Gaussians changes during retargeting as seen in **(a)**, the Gaussians will not be interpolated correctly and can collapse into a single Gaussian halfway through, as seen in **(b)**. This significantly changes the shape of the interpolated specular highlight.

When Gaussians are organized along a line, this ensures the order of the Gaussians is preserved. When Gaussians are ordered in a triangular structure, this same triangular structure is preserved.

We create the relative structure loss function

$$\mathcal{L}'_{str} = \sum_{p\in G_{i\to j}} \sum_{q\in G_{i\to j}} h(\rho_{pq}) * (\rho_{pq} s_{ij} - \rho_{\hat{p}\hat{q}})^2 \tag{3.12}$$

where ρ_{pq} is the Euclidean distance between Gaussians p and q, and $\hat{p}$ and $\hat{q}$ are the original Gaussians from G_i before retargeting to p and q. We use the function $h(x)$ and parameter s_{ij} to manipulate the area of effect and scaling of distances between Gaussians.

Since we are primarily focused on local structure, we only want to take changes between nearby Gaussians into account. This way, we can bend the larger structure while preserving the smaller detail. For this, we use the differentiable limit function

$$h(x) = \frac{1 + \tanh(\gamma * (m - x))}{2} \tag{3.13}$$

where γ and m are the power of change and the threshold limit, which have been empirically set to 4 and 15, respectively. Increasing the value of γ causes a steeper drop above the threshold, but makes the derivative more erratic, causing unexpected behavior.

Specular highlights can also change size, meaning the Gaussians should be able to change their distances from each other uniformly. To allow this to happen, we add the scaling factor

$$s_{ij} = \frac{\sum_{p,q\in G_{i\to j}} \rho_{pq}}{\sum_{\hat{p},\hat{q}\in G_i} \rho_{\hat{p}\hat{q}} + \epsilon}. \tag{3.14}$$

This allows all distances between Gaussians to scale together, allowing the growing and shrinking of the specular highlights while keeping the same shape.

3.3.3 Similar energy

Finally, we also need the specular highlights to keep representing similar features. Bright, large Gaussians should not change into dim, small Gaussians. There are ways in which Gaussian can change size, for example, when a specular highlight becomes narrower. Therefore, penalizing changes to size or amplitude directly can cause incorrect specular highlight shapes. As an alternative, penalizing the overall change in brightness of the Gaussian allows these factors to have trade-offs between each other, while ensuring the total energy is preserved. This brightness can be calculated by integrating the Gaussians, giving us the function

$$\mathcal{L}'_{ene} = \sum_{p \in G_{i \to j}} \left(\int_{\mathbb{R}^2} g_{\hat{p}}(\mathbf{x})\, d\mathbf{x} - \int_{\mathbb{R}^2} g_p(\mathbf{x})\, d\mathbf{x} \right)^2. \tag{3.15}$$

The integral of a Gaussian can be calculated using the formula

$$\int_{\mathbb{R}^2} g(\mathbf{x}) d\mathbf{x} = 2\pi a^2 \sigma_x \sigma_y, \tag{3.16}$$

where $\boldsymbol{\sigma} = [\sigma_x, \sigma_y]^T$.

3.3.4 Combining the loss function

When combining the different parts of the loss function, there is a balance that needs to be considered. That gives the loss function

$$\mathcal{L}_{ret}(G_{i \to j}) = \underbrace{c_{fit}\mathcal{L}'_{fit}}_{\text{image fit}} + \underbrace{c_{str}\mathcal{L}'_{str}}_{\text{structure}} + \underbrace{c_{ene}\mathcal{L}'_{ene}}_{\text{energy}}. \tag{3.17}$$

where c_{fit}, c_{str}, and c_{ene} are hyperparameters to balance the three parts of the optimization function.

3.3.5 Connected components

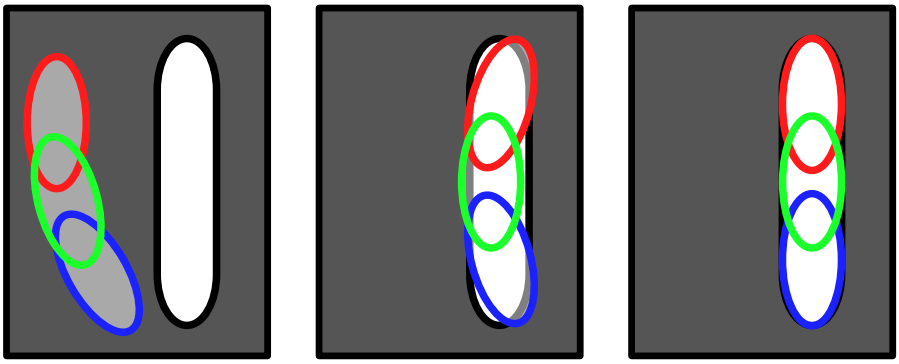
Since specular highlights are not necessarily at the same location in neighboring specular highlight images, it is likely that, at the start of the Gaussian retargeting step, some Gaussians are too far away from brighter specular highlight values for the gradient descent to move them towards the specular highlights effectively. In these cases, they are completely surrounded by black pixels. With the previously discussed constraints, this means the Gaussians will usually contract,

clumping together in a single point to minimize their visual footprint, decreasing all their distances uniformly, and uncontrollably changing their size and rotation. This effect will slowly revert when the Gaussian eventually finds a specular highlight, but this is a slow process, and significant changes to the size and rotation can often still be visible after the retargeting step.

To combat this, we split the Gaussian retargeting step into two phases: first, optimizing overlapping Gaussians together on their size and rotation, and secondly, optimizing the Gaussian parameters individually when they are close enough to the specular highlights as discussed before.

To move groups of Gaussians together, we first need to analyze which Gaussians form connected components that represent a single specular highlight. Since Gaussians are unbounded functions, we need a threshold that tells us which Gaussians are touching. Since the level sets of individual Gaussians are ellipses, we can select a threshold that defines an ellipse per Gaussian and calculate if the ellipses for two Gaussians intersect, which can be done using the formulation by Eberly (2000). This gives us groupings of Gaussians, the connected components, that can be moved together.

These connected components are only optimized on their position $\ddot{\mu}$, which can be added to the positions of the individual Gaussians in the connected components, and rotation $\ddot{\theta}$, which is a rotation around a central point to make sure the structure of the connected component does not change.



(a) Before the retargeting, Gaussian set G_i can be far from the target highlight.

(b) First, whole connected components are optimized, keeping their relative position.

(c) After that, individual Gaussians are optimized.

Figure 3.8: We optimize the connected components **(b)** before optimizing the individual Gaussians **(c)**.

3.4 GAUSSIAN INTERPOLATION

As a final step, we interpolate the known Gaussians to get the interpolated specular highlights. We use the Delaunay triangulation from Section 3.3 and query our lighting direction ω' to get the barycentric coordinate $(\lambda_A, \lambda_B, \lambda_C)$ within a triangle ABC, represented by Gaussian sets G_A, G_B, and G_C.

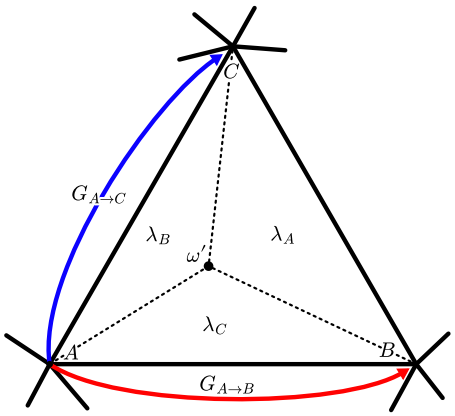


Figure 3.9: Illustration of a triangle within the Delaunay triangulation showing the corners ABC, the polled light direction ω', the barycentric coordinate $(\lambda_A, \lambda_B, \lambda_C)$, and the direction of the retargeted Gaussians $G_{A\to B}$ and $G_{A\to C}$.

Since the accuracy of the retargeted Gaussians is lower than that of the original, estimated Gaussians, we want an interpolated result close to a vertex to be formed mainly by the estimated Gaussian. We achieve this by applying a double barycentric interpolation.

The first interpolation is between the parameters of the estimated Gaussians and their corresponding retargeted Gaussians. For example, we interpolate between the parameters of G_A, $G_{A\to B}$, and $G_{A\to C}$ to get G'_A, the interpolated Gaussians coming from vertex A.

The second interpolation is between the three resulting Gaussian sets using

$$G' = \lambda_A G'_A + \lambda_B G'_B + \lambda_C G'_C, \tag{3.18}$$

where multiplying a Gaussian set by λ is equivalent to multiplying λ by its amplitude a, and adding Gaussians is equivalent to combining their sets. This set of Gaussians can be rendered to get specular highlight image S', representing the specular highlight for the queried direction ω' using

$$S'(x, y) = \mathcal{G}([x, y]^T, G'). \tag{3.19}$$

This makes sure that images close to a known specular highlight are mostly represented by the initially estimated Gaussians instead of the retargeted Gaussians that are subject to more constraints and

therefore have lower visual similarity. It also allows blending between the different sets of Gaussians.

Other methods, such as the method proposed by Fuchs et al. (2007), use two linear interpolations to get the intermediate result. However, this requires the first set of interpolated Gaussians to be retargeted on the third set of Gaussians and vice versa, which is an expensive calculation that cannot be precomputed, since it depends on the requested lighting direction.

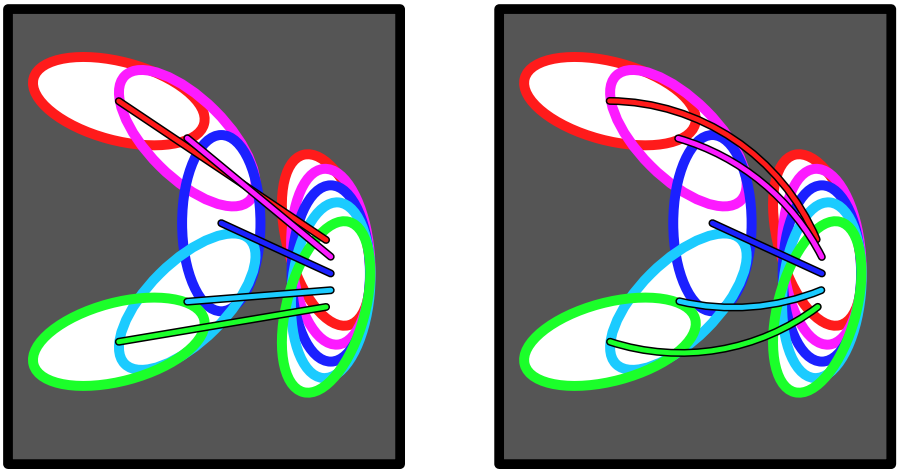
During the first interpolation, the interpolated Gaussian sets G'_A, G'_B, and G'_C are created by interpolating their parameters individually. For simplicity, we only write down the formulae to create G'_A, but the others follow the same structure. Gaussian parameters a and σ are interpolated through linear barycentric interpolation using the formula

$$a'_A = \lambda_A a_A + \lambda_B a_{A\to B} + \lambda_C a_{A\to C} \tag{3.20}$$

$$\sigma'_A = \lambda_A \sigma_A + \lambda_B \sigma_{A\to B} + \lambda_C \sigma_{A\to C}. \tag{3.21}$$

The parameter θ is also calculated through a barycentric interpolation. However, there are two valid rotations for any start and end θ: clockwise and counterclockwise. The direction of rotation will be calculated based on the path the Gaussians take in Section 3.4.1.

Doing the same with position μ can result in unnatural movement of Gaussians, as can be seen in Figure 3.10. The Gaussians need to follow a more natural path, guided by the shape of the object.



(a) Straight paths **(b)** Circular paths

Figure 3.10: Illustration of a highlight shrinking and moving to the right. The paths taken by individual Gaussians are represented by lines. On the left are straight paths where the red and pink paths cross each other on their way to their final position. On the right are circular paths as described in this section.

3.4.1 Movement paths

When calculating the new position of the Gaussians, we start by looking at the interpolation between two of the three vertices in the Delaunay triangulation, starting between G_A and $G_{A\to B}$. We can model all interpolated positions for each Gaussian between these two vertices as a smooth path going from its position in G_A to its position in $G_{A\to B}$. An example of these paths is shown in Figure 3.10.

When we look at how specular highlights move on an object when designing highlights around edges on an object, we can dissect the movement of a highlight into two orthogonal components: movement across the edge and movement along the edge.

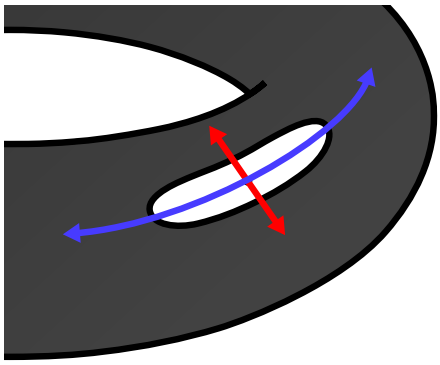


Figure 3.11: Illustration of a specular highlight being able to move across the edge **(red)** and along the edge **(blue)**.

MOVEMENT ACROSS THE EDGE means the specular highlight moves laterally from one side of the edge to the other side. The path that follows this movement is orthogonal to the direction of the edge and can be approximated with a linear translation.

MOVEMENT ALONG THE EDGE means the path of the specular highlight follows the shape of the edge on the object. The shape of this edge can be complex, but since the local geometry of the object can be approximated using the direction of the Gaussians when using point lights, as shown in Section 3.2, we can estimate the edge direction at the positions of these Gaussians. Most importantly, the Gaussian is elongated in the direction of the edge, meaning the direction, or rotation, of the Gaussian and the direction of the edge should match.

Since the rotation of the Gaussian is in the direction of the edge at any point along the edge, we can conclude that the change in rotation of the Gaussian should be the same as the change in direction of the path. Since the rotation θ will be linearly interpolated, this change in rotation and the path's change in direction will be constant, meaning the path should follow the arc of a circle. This arc is defined such that μ_A and μ_B are its start and end points, respectively, and the circle's

center point $\mathbf{O}$ is constructed such that the angle $\angle \boldsymbol{\mu}_A \mathbf{O} \boldsymbol{\mu}_B$ is equal to the difference between θ_A and θ_B. The formula for interpolating the path becomes

$$\boldsymbol{\mu}'_{A\to B} = R_{\lambda_B \theta^{\Delta}_{A\to B}}(\boldsymbol{\mu}_A - \mathbf{O}) + \mathbf{O}, \tag{3.22}$$

where $R_{\theta^{\Delta}_{A\to B}}$ is the rotation matrix defined by the interpolated rotation difference $\lambda_B \theta^{\Delta}_{A\to B}$ between Gaussians in G_A and $G_{A\to B}$. Then the interpolated rotation becomes

$$\theta'_A = \theta_A + \lambda_B \theta^{\Delta}_{A\to B} + \lambda_C \theta^{\Delta}_{A\to C}. \tag{3.23}$$

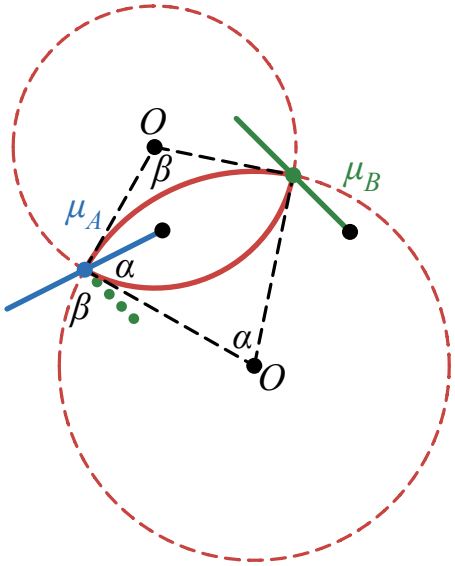


Figure 3.12: Construction of the circular paths, where $\boldsymbol{\mu}_A$ and $\boldsymbol{\mu}_B$ are the start and end points of the path respectively, $\mathbf{O}$ are the possible center points of the circle, and α and β are clockwise and counterclockwise rotation angles respectively. The angles $\boldsymbol{\mu}_A \mathbf{O} \boldsymbol{\mu}_B$ are equal to the difference in rotation of the Gaussians, represented as direction lines.

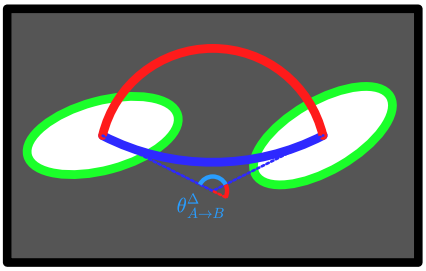
As noted before, the rotation $\theta^{\Delta}_{A\to B}$ has two valid rotation directions: clockwise and counterclockwise. Both result in valid interpolations, and both cause different circular paths as seen in Figure 3.12. Which to take depends on the angle of the rotation and the paths of neighboring Gaussians.

In general, we want Gaussians close to each other to follow similar paths with similar rotation angles to make sure Gaussians do not collapse or tear during interpolation. To achieve this, we define a weight w such that $w \leqslant 0$ corresponds to a clockwise rotation, while $w > 0$ corresponds to a counterclockwise rotation. The further from 0, the more confident the direction is. This weight will later be propagated to neighboring Gaussians. We define the weight as

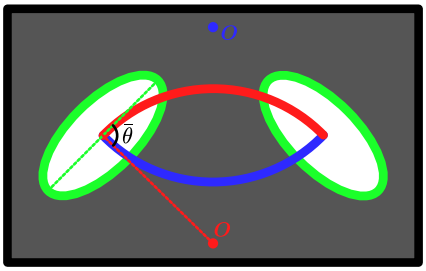
$$w = \frac{\cos(\theta^{\Delta}_{A\to B}) + \theta^{\Delta}_{A\to B} - \pi/2}{\pi/2 - 1} + \frac{\cos(\bar{\theta})^2 - 0.5}{10} \tag{3.24}$$

where $\bar{\theta}$ is the angle between the direction of the Gaussian and the direction from the Gaussian towards centerpoint **O**.

This weight consists of two distinct parts, as shown in Figure 3.13. The first part favors straight paths over semicircular paths. When $\theta^{\Delta}_{A\to B}$ is around $\pi/2$, we want this to be around 0, so other Gaussians can have a more significant impact during propagation. The second part slightly favors moving the Gaussians along the edge.



(a) Straighter paths are preferred. Here the bottom blue path is selected.



(b) Paths in the direction of the Gaussians are preferred. Here the top red path is selected.

Figure 3.13: The decision for which path to take depends on two aspects: **(a)** it prefers straighter paths, and **(b)** it prefers paths in the direction of the Gaussians.

We propagate weights to nearby Gaussians based on the distance between these Gaussians. We define the function that specifies relations between Gaussians

$$\xi_{pq} = \frac{1}{1 + \rho^2_{pq} + \rho^2_{\hat{p}\hat{q}}}, \tag{3.25}$$

where ρ_{pq} is the Euclidean distance between Gaussians p and q. When two Gaussians are at the same position, the value is 1, while for Gaussians further away, ξ_{pq} approaches 0.

We use this relation metric in the propagation matrix M defined as

$$\eta_p = \sum_q \xi_{pq} \tag{3.26}$$

$$\phi_{pq} = \text{sign}\left(\frac{\pi}{2} - |\theta_p - \theta_q|\right) \tag{3.27}$$

$$M_{pq} = \begin{cases} 1/\eta_p, & p = q \\ \phi_{pq}\xi_{pq}/\eta_p, & p \neq q. \end{cases} \tag{3.28}$$

Here η_p is a normalization factor to make sure values stay within the range $[-1, 1]$ and ϕ_{pq} ensures rotation parity. This is needed because circular paths with the same rotation direction are not necessarily the most alike, as demonstrated in Figure 3.14. ϕ_{pq} will propagate the inverse weight when opposite rotation directions are more similar.

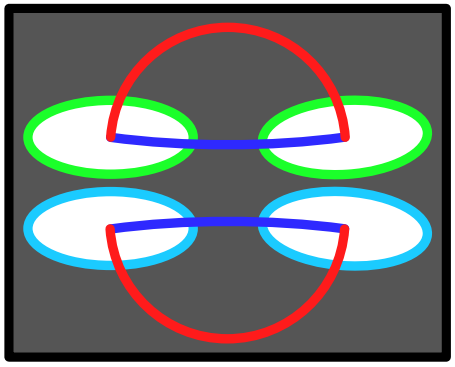


Figure 3.14: Example of a situation where the same rotation direction does not mean the paths are the most similar. The two blue circular paths are only a few degrees off each other, but have different rotation directions. The rotation parity factor ϕ_{pq} is used to compensate for this.

We apply the matrix multiple times to the weights to propagate the values using

$$w' = M^n w, \tag{3.29}$$

where we set $n = 2$ empirically. Based on whether the final weight is positive or negative, we take the clockwise or counterclockwise rotation path.

When we have calculated w and used its sign to select which $\mathbf{O}$ to use, we can interpolate the new position

$$\boldsymbol{\mu}'_{A\to B} = R_{\lambda_B \theta^{A}_{A\to B}}(\boldsymbol{\mu}_A - \mathbf{O}) + \mathbf{O}. \tag{3.30}$$

To get to the final position within the triangle, we can treat the newly found interpolated positions as displacements by calculating $\boldsymbol{\mu}_{A\to B} - \boldsymbol{\mu}_A$ and $\boldsymbol{\mu}_{A\to C} - \boldsymbol{\mu}_A$, and adding them to the original position. This gives the interpolation formula

$$\boldsymbol{\mu}' = \boldsymbol{\mu}'_{A\to B} + \boldsymbol{\mu}'_{A\to C} - \boldsymbol{\mu}_A. \tag{3.31}$$

3.4.2 Combining

A final image I' can be created by adding the interpolated specular highlight image S' to the interpolated diffuse image D'.

3.5 PERFORMANCE CONSIDERATIONS

In order to get immediate feedback on the designed specular highlights, the interpolation computation needs to be close to real-time. In order to achieve this, the following considerations have been made:

PRECOMPUTATION All steps in the process except for the interpolation can be precomputed. Because of this, only the performance of the Gaussian interpolation step has an impact on the speed of feedback to the user. Since all operations in this step are relatively inexpensive, the interpolated images can be computed quickly. This also means that all precomputation can be done on more powerful hardware, and the design on less capable devices.

STORAGE All precomputed information, except for the diffuse images, is efficient in storage, since we are only storing Gaussian parameters. Storage does not scale linearly with resolution. Given that we need to store the results of the Gaussian estimation and the Gaussian retargeting, we only need to store $7n - 12$ Gaussian sets, where n is the number of input lighting directions. Assuming an average of 25 Gaussians per Gaussian set and 256 lighting directions, it comes down to approximately 2.2MB of storage. This is significantly lower than methods like Fuchs et al. (2007) who would theoretically need to store intermediate results, since they rely on full images at every step in the process.

RESOLUTION SCALING Another advantage of using a Lagrangian formulation is that the intermediate computation is largely independent of the resolution of the input images. Gaussian sizes and positions can be scaled to do any part of the method on a lower resolution, and scale back up for the end result. However, doing so during the precomputation step can cause lower detail specular highlights.

4 EXPERIMENTS

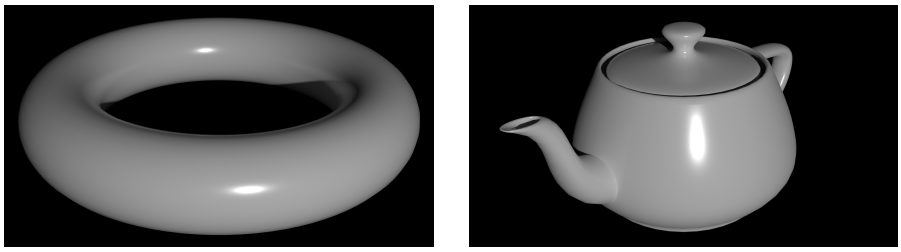
In our experiments, we aim to verify the presented specular highlight interpolation method and compare it to a baseline implementation.

In our experiments, we primarily aim to test the following:

1. If the interpolation method creates interpolated specular highlights of high quality comparable to known specular highlight images;
2. If the Gaussian estimation method produces high-quality specular highlight images with relatively few Gaussians;
3. If the Gaussian interpolation method can give real-time interpolations on request;
4. How the method reacts to changes in variables and different situations.

IMPLEMENTATION The experiments were run on a desktop computer with an AMD Ryzen 7 3700X 8-core processor, 32GB of DDR4 RAM, and an Nvidia GeForce RTX 3080 graphics card with 12GB of VRAM.

All methods have been implemented in Python 3.10 using PyTorch, and the Gaussian rasterization and differentiation are implemented using custom CUDA kernels. Code for both the full method and the Gaussian rasterizer are made available online.[1]



(a) Donut **(b)** Utah teapot

Figure 4.1: Example images from the used datasets.

USED DATASETS All experiments are run on synthetic datasets rendered with Blender. These datasets contain 250 lighting directions

1 An implementation of the method is available at github.com/ACMichels/RF-highlight-interp and the rasterizer is available at github.com/ACMichels/gaussian-drawing

uniformly distributed on the sphere using the Fibonacci lattice algorithm. Each image is a 480x270 HDR image. Example images from the datasets can be seen in Figure 4.1.

For some experiments, specific lighting directions and their specular highlights are highlighted. These reference lighting scenes are shown in Figure 4.2.

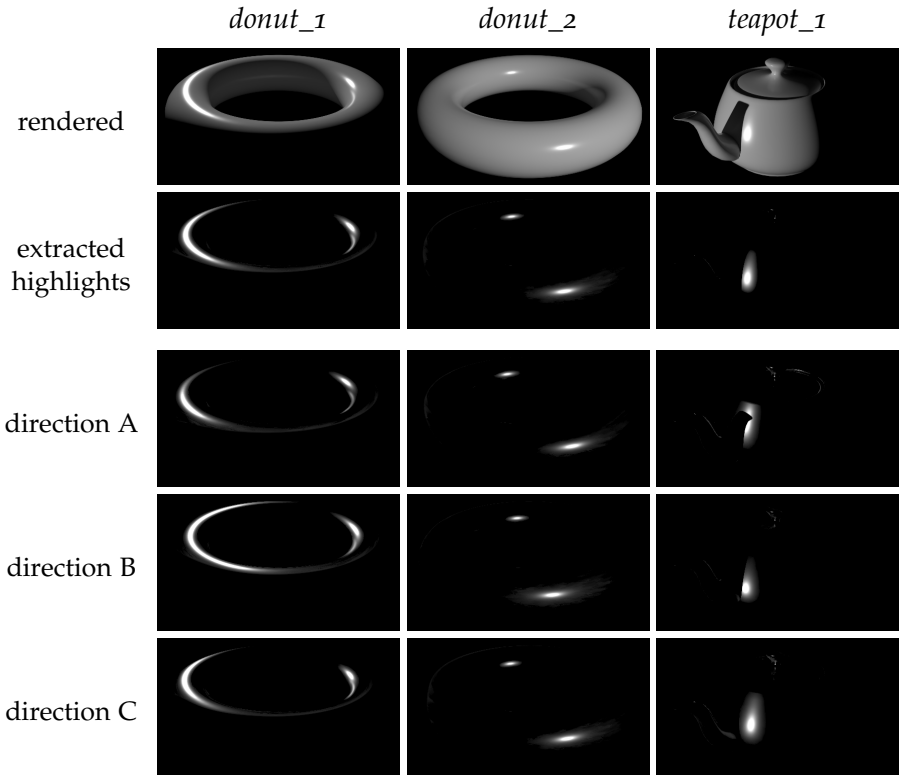


Figure 4.2: The reference scenes from the used datasets, including their extracted highlights and the known directions used for the interpolation.

USED INTERPOLATION METHODS In our comparisons, we use three different methods:

- **Circular** path method: The method described in Section 3.
- **Straight** path method: The method described in Section 3 without the circular paths. Instead, we calculate the position μ' using linear interpolation based on the barycentric coordinates, similar to α' and σ' in Equations 3.20 and 3.21 respectively. We still calculate the interpolated rotation θ' using the method described in Section 3.4.1. This allows us to avoid an artificially higher or lower error caused by different choices in rotation direction.

- **Alpha blend** method: A baseline implementation using alpha blending of the known highlights based on the barycentric coordinates.

USED METRICS We compare the methods with the rendered specular highlights using the peak signal-to-noise ratio (PSNR). The higher the PSNR, the better the reconstruction quality. However, these values should only be used to compare results from the same input scene. Between different datasets, the PSNR values vary with different highlight intensities and sizes. Therefore, all comparisons made with the PSNR will be within the same dataset between different interpolation methods or parameters.

Some experiments show a range of PSNR values by showing the mean and the 5th and 95th percentiles.

All interpolations are done on barycentric coordinates $\lambda_A = \lambda_B = \lambda_C = 1/3$ to show the situation furthest from a known lighting direction. When all of the λ values are 0 or 1, the highlights are known, and we expect the PSNR to be highest, whereas if they are close to 1/3, they are furthest from a known highlight, and we expect the PSNR to be lowest. This relation is further explored in Experiment 4.5.4.

4.1 QUALITATIVE RECONSTRUCTION COMPARISON

We evaluate the quality of our interpolation results by comparing them with rendered reference images.

We specifically compare the reference scenes from Figure 4.2, showing the advantages and disadvantages of the different techniques. For these scenes, we calculate the PSNR and visually show the difference between the interpolated and rendered specular highlights.

We hypothesize that the circular path method outperforms both the straight path and the alpha blend method, while the straight path method outperforms the alpha blend method. Specifically, we expect the placement of the circular path to be better than that of the straight path, especially in situations where the direction of curvature changes significantly, for example, in the *donut_1* reference scene.

Results

Table 4.1 presents the comparisons between the interpolated and rendered images, including their PSNR. In all three shown scenes, our method improves upon the alpha blend of the input images, and the circular path improves on the straight path.

The difference images in scene *donut_1* clearly show the advantage of using the circular paths. The straight paths cut off the bend, causing two separate bands where the highlight is supposed to be and where

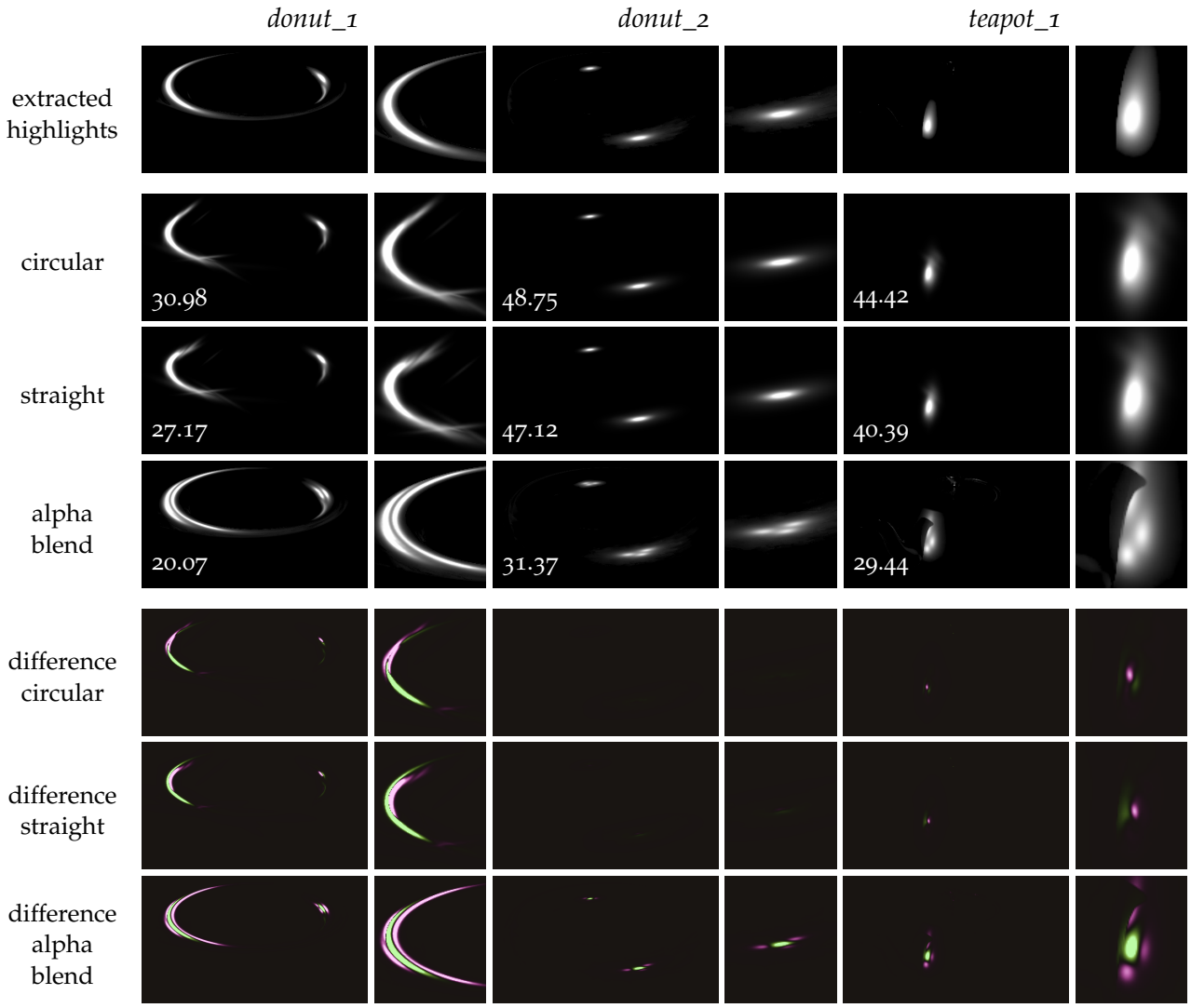


Table 4.1: Comparison of interpolated lighting directions, showing a render, a barycentric blend of the known images, our method using straight paths, and our method using circular paths. The number in the bottom left of each image represents the PSNR compared to the rendered highlight (higher is better).

it is placed by the interpolation algorithm. The circular paths improve the positioning by overlapping these bands.

Also shown in scene *donut_1* is the ability to interpolate the highlight shape correctly. The three known specular highlights vary in shape and size. This is also clearly visible in the alpha blend method, where the specular highlight is too large compared to the known highlight. Both the circular and straight path methods produce a highlight of the correct shape and size.

Scene *donut_2* shows simpler highlights in the shape of singular Gaussians. These are fitted well, causing a very high PSNR compared to the alpha blended highlights. The circular paths also improve the

score compared to the straight path, meaning the position of the highlight on the object is slightly more accurate.

Scene *teapot_1* shows that the shadow of the teapot spout is avoided relatively well without any actions taken to account for shadows. This probably happens because the left side of several of the input images contains a hard line representing the transition between highlight and shadow. Additionally, the shadow is always on the left side of the highlight, meaning there are no Gaussians moving over the shadow. Investigating how the method responds to different kinds of shadows requires further research.

The overall reconstruction quality of the scenes is not optimal. Specifically, scenes *donut_1* and *teapot_1* show clear differences between the interpolated image and the reference image. Scene *donut_1* shows visible artifacts from the interpolation in the form of individual Gaussians sticking out and the ends pointing in different directions.

4.2 QUANTITATIVE RECONSTRUCTION COMPARISON

To determine if the results from Experiment 4.1 generalize, we want to compare other lighting directions to assess the overall reconstruction quality of our methods.

To assess this overall reconstruction quality, we plot the PSNR of all 496 intermediate lighting directions in the datasets in the violin plots in Figure 4.3. Some images were excluded where no specular highlights were detected, resulting in infinitely large PSNR values.

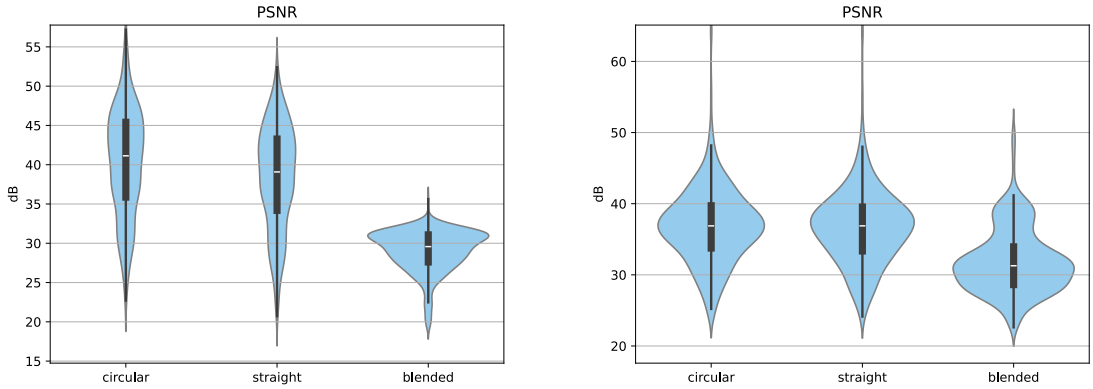
We expect the trends seen in Experiment 4.1 to continue. Specifically, we again expect the circular path method to improve on the straight path method, which in turn should improve on the alpha blend method.

Results

Figure 4.3a shows a clear difference between the three methods. There are some situations where our method performs worse than the alpha blend method, but the vast majority of lighting directions are handled better by the circular path method.

Figure 4.3b shows some worse PSNR values, but mostly improvements compared to the alpha blend method, although the improvements are more moderate. The improvement of the circular path method compared to the straight path is not there.

These less pronounced results can mainly be explained by the required sizes of the Gaussians. The more minor details in the teapot with very high curvature, e.g., the lid, handle, and spout, will result in smaller, narrower highlights. To accurately model these highlights, a



(a) Donut

(b) Utah teapot

Figure 4.3: The distribution of PSNR values for the donut **(a)** and teapot **(b)** datasets run on the circular path method, the straight path method, and the alpha blend method.

higher number of smaller Gaussians is needed, which is prevented by our method in order to attain the properties needed as described in Section 3.4.1. These specific sections of the teapot dataset are further explored in Experiment 4.3 and the effect of creating smaller Gaussians is explored in Experiment 4.5.3.

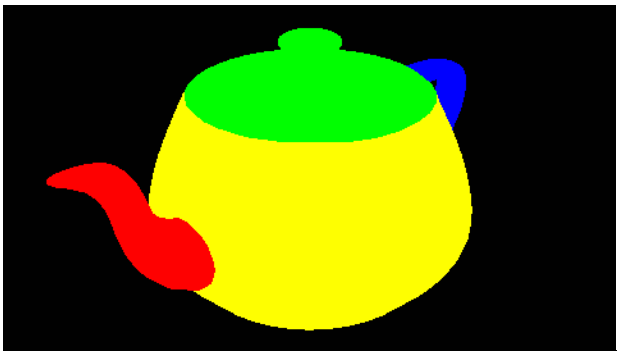
4.3 SEGMENTED RECONSTRUCTION COMPARISON

We want to know how the interpolation method responds to different types of geometry. This information can help explain the results from Experiments 4.1 and 4.2.

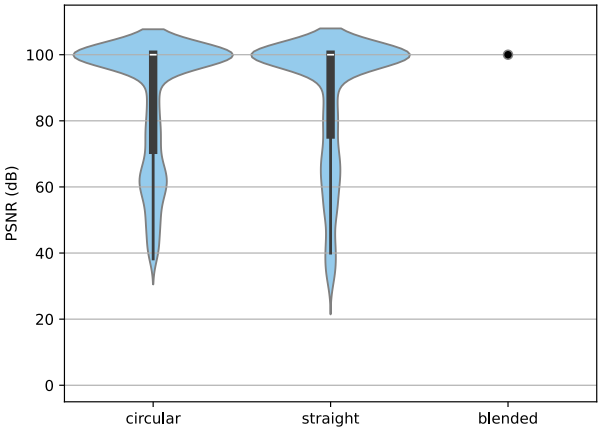
We segment the pixels of the teapot dataset into the parts shown in Figure 4.4. For each of these segments, we calculate the distribution of PSNR values similar to Experiment 4.2.

When segmenting the image, there are often parts that do not contain any specular highlights in both the reference image and the interpolated highlights, resulting in a perfect reconstruction and an infinite PSNR. We capped these values at 100. Images that have values this high would otherwise already represent a nearly imperceptible decrease in quality.

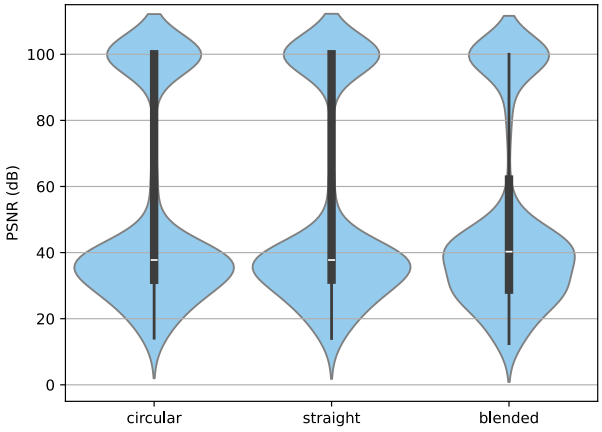
We hypothesize that we will still see improvements for all segments. However, we expect them to be more pronounced for the main body, since this would correspond most to the donut dataset because its curvature is similar.



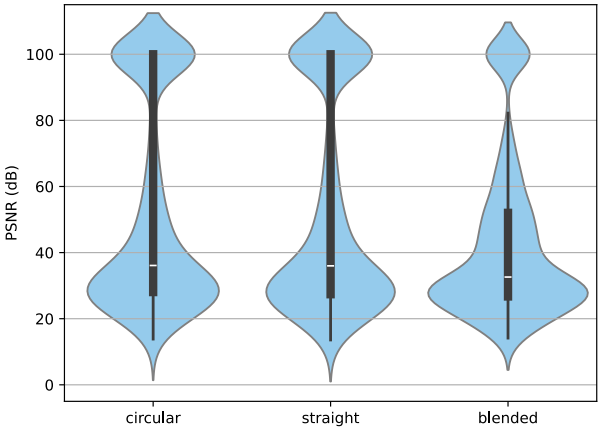
(a) Segmentation image



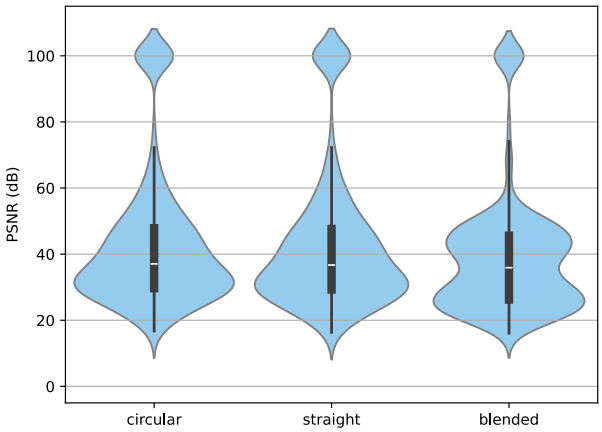
(b) Background PSNR (black)



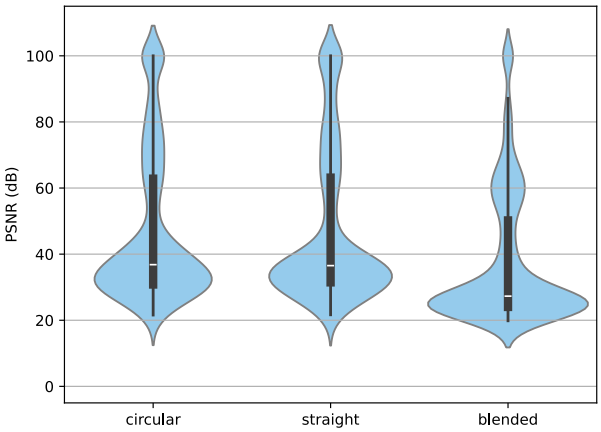
(c) Lid PSNR (green)



(d) Handle PSNR (blue)



(e) Spout PSNR (red)



(f) Body PSNR (yellow)

Figure 4.4: The segmentation image used for the experiment **(a)** and the distribution of PSNR values for each segment **(b...f)**, run on the circular path method, the straight path method, and the alpha blend method.

Results

These results in Figure 4.4 show a marginal improvement for all parts, whereas the improvement for the body is more pronounced. Since the majority of the pixels contain the body, this result dominates the aggregated result in Figure 4.3b of the previous experiment, where the results of all parts are combined.

The lid, containing the most detailed curvature, shows the most significant decrease in PSNR. This is primarily caused by the Gaussian estimation step not modeling small or dim parts of specular highlights. This is further explored in Experiment 4.5.3.

Notably, our methods have a larger number of perfect matches with a PSNR of 100 compared to the alpha blend method. This happens when the interpolated scene does not contain any specular highlights, but one or more of the input images for the barycentric blend do contain highlight pixels. These pixels will always be blended to a non-zero value, causing a slight difference between the reference and interpolated highlights. These highlights are often not captured by our Gaussian estimation step, since they are generally very dim. In cases where they are, they exhibit the same behavior as for the alpha blend method.

4.4 SPEED BENCHMARKING

To see if we can use this method for real-time lighting design, we test the computation time per component on multiple datasets.

The image segmentation, Gaussian estimation, and Gaussian retargeting steps can take a significant amount of time to compute but can be precomputed as mentioned in 3.5. All these steps, except the segmentation, are linear in terms of the number of input images, since we need n Gaussian estimation steps and $6n - 12$ retargeting steps (derived from the Euler characteristic) for n input images.

On the other hand, we expect the Gaussian interpolation step to be fast enough to give close to real-time results.

Results

Results of the benchmark are shown in Table 4.2.

In total, all preprocessing combined took approximately 65 minutes for these datasets, with some variation caused by differing numbers of Gaussians. This is relatively slow and can be improved on in future iterations through more parallelization and improved memory handling. However, this slow processing does not need to be a dealbreaker. Per dataset, these steps only needs to be run once, and the results can be stored for later use.

	Time per iteration (ms)			Total time (s)	
	donut	teapot	#iterations	donut	teapot
Loading data	68 ms	36 ms	input image (250)	17 s	9.0 s
Segmentation	1519 ms	1590 ms	total input (1)	1.5 s	1.6 s
Estimation	1296 ms	1357 ms	input image (250)	324 s	339 s
Retargeting	2401 ms	2335 ms	triangle edge (1488)	3573 s	3474 s
Interpolation	**12 ms**	**11 ms**	output image (496)	-	-
Total				3915 s	3824 s

Table 4.2: Amount of time spent by each part of the method for each dataset. Times are in ms per iteration or in seconds for the full method. The #iterations column shows the number of iterations performed. The interpolation step in **bold** need to be real-time and has not been added to the total time, since it cannot be precomputed.

The Gaussian interpolation is fast enough to be used in real-time for both datasets. This means it meets our speed requirement and could be used to get real-time feedback to a designer.

4.5 ANALYSIS

4.5.1 Estimation loss function analysis

To achieve the best possible interpolation results, we want to know how the Gaussian estimation loss function behaves and how it compares to similar loss functions.

We investigate the effect of several loss function norms on the estimated Gaussians. First, we visually compare a small set of interpolation results and compare their number of Gaussians. Second, we quantitatively compare the resulting number of Gaussians from the different loss functions for the complete datasets. A lower number of Gaussians can have positive effects for the final interpolation result as described in Section 3.2. This relationship is further explored in Experiment 4.5.5.

Because we are only looking at the Gaussian estimation step, we use lighting direction A for scenes *donut_1*, *donut_2*, and *teapot_1* instead of the full interpolation triangle.

We compare the following norms:

1. L2 norm: $\sqrt{\sum (x_1 - x_2)^2}$
2. abs norm: $\sum |x_1 - x_2|$
3. L2 norm with $\log_2(x+1)$: $\sqrt{\sum (\log_2(x_1+1) - \log_2(x_2+1))^2}$

4. abs norm with $\log_2(x+1)$: $\sum |\log_2(x_1+1) - \log_2(x_2+1)|$

Loss function 4 is the one used in our method in Equation 3.6. The other norms are constructed to build up from a more conventional L2 norm to our selected norm.

We hypothesize that functions 2 and 3 will produce a lower number of Gaussians than function 1, since these focus more on the value range that is important for the interpolation. Similarly, we expect function 4 to give the lowest number of Gaussians, while keeping visual quality high enough.

Results

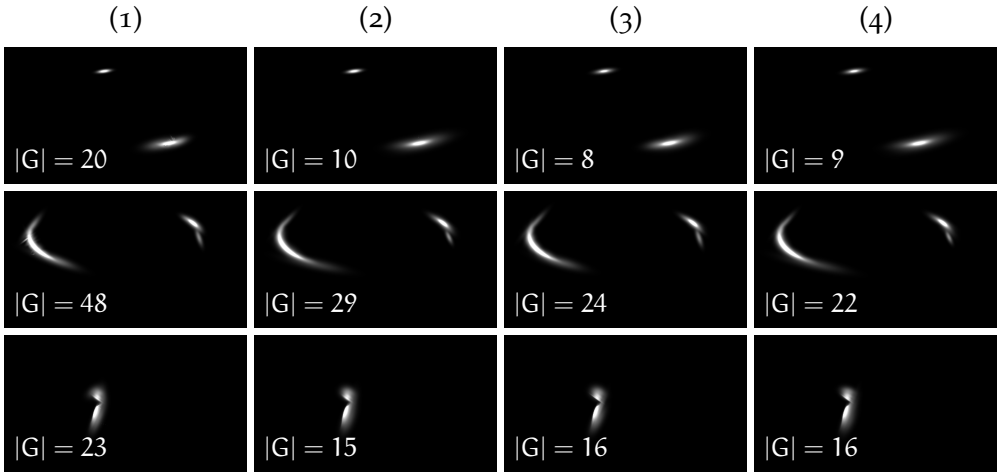


Figure 4.5: Comparison of different loss functions. (1) and (3) use the L2 norm, while (2) and (4) use the absolute norm. Additionally, (3) and (4) transform the input using $\log_2(x+1)$. The number in the bottom left of each image is the number of Gaussians used.

A visual comparison between the different loss functions can be seen in Figure 4.5. Column (1) shows that MSE creates the highest number of Gaussians and exhibits visible artifacts, as highlighted in Figure 4.6. Gaussians are placed perpendicular to the direction of the specular highlights, likely to compensate for the high peak values, which MSE disproportionately targets. The other loss functions perform better, both visually and in terms of the number of Gaussians.

Figure 4.7 similarly shows that MSE results in the highest number of Gaussians.

The log MSE loss function performs best when looking at only the number of Gaussians. This is different from the hypothesis. We expected the absolute methods to outperform the MSE methods, since MSE would theoretically still heavily penalize the higher values. However, it creates a better fit on the lower values, resulting in fewer Gaussians being created.

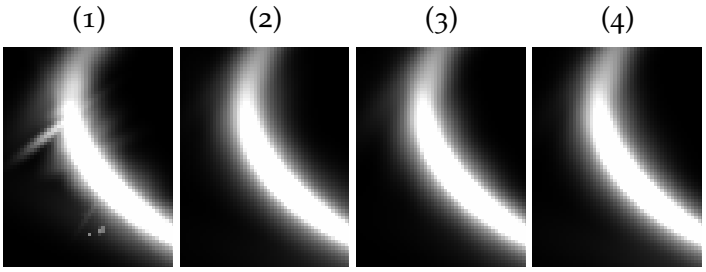
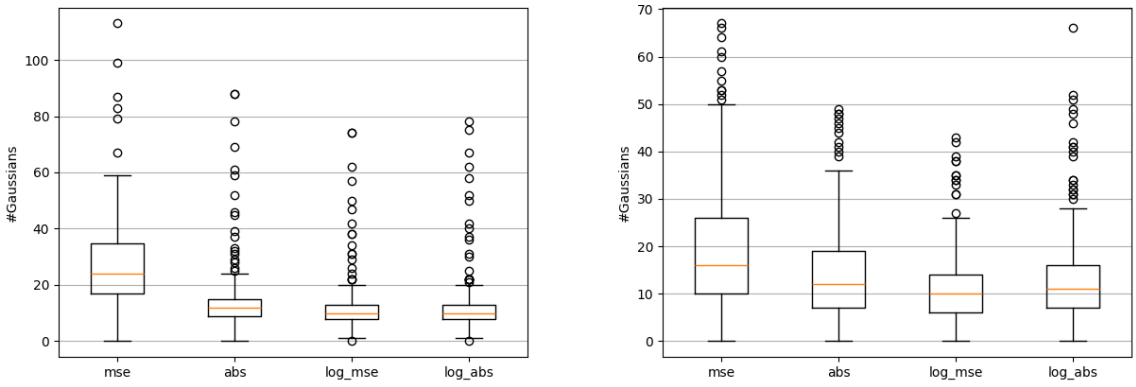


Figure 4.6: Zoomed in comparison of different loss functions for scene *donut_2* from Figure 4.5. The MSE method shows artifacts of small Gaussians, while the other methods do not.



(a) Donut **(b)** Utah teapot

Figure 4.7: Distributions of the resulting number of Gaussians for different estimation loss functions.

4.5.2 Retargeting loss function analysis

The Gaussian retargeting loss function described in Section 3.3 is combined in Equation 3.17 with three weights: c_1, c_2, and c_3. These combine the different parts of the loss function with these purposes:

1. **Target**: Make the Gaussians look as much as possible like the target image.
2. **Structure**: Keep the relative structure of the Gaussians the same.
3. **Purpose**: Make each Gaussian represent similar features.

We want to understand how these three parts are related to each other and how they influence the resulting reconstruction quality. We analyze this by varying the three weights when interpolating scenes *donut_1* and *teapot_1*.

Because the Adam optimizer we use for Gaussian retargeting is invariant to scaling of input values and gradients (Kingma, 2014),

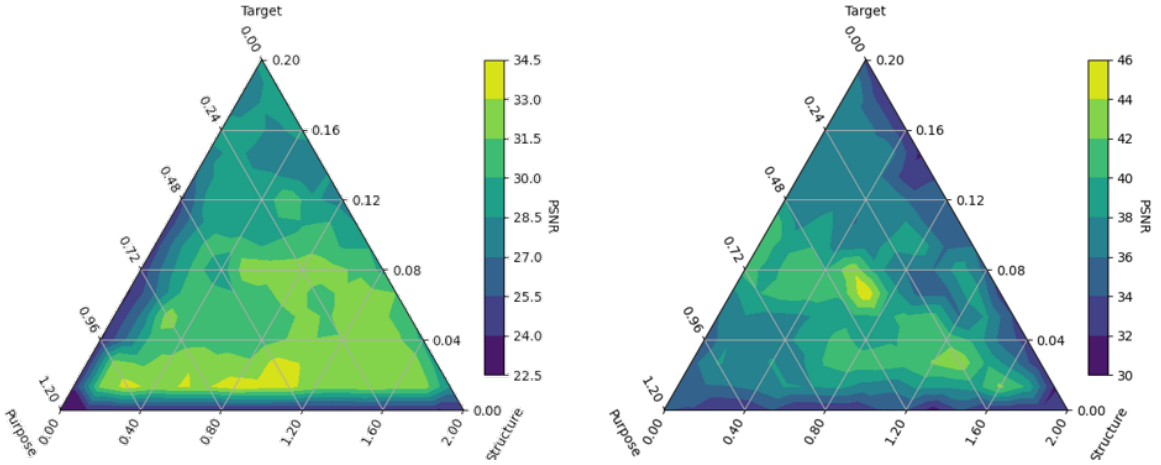
we can apply a normalization on the weights without changing the optimization dynamics. This effectively makes the search space 2-dimensional, which we can visualize in a ternary plot.

We plot the PSNR scores for the two lighting directions and varying weights in ternary plots. Instead of the resulting values adding to 1, we applied different scaling to the axes to center an area of interest and explore the search space around it. This is necessary because the amplitudes of the individual parts of the loss function have no relation to each other. I.e., one weight being double the other does not mean one is twice as important as the other. Weights can be orders of magnitude apart to balance each other correctly.

We hypothesize that all three parts of the loss function are necessary for good scores. The PSNR around the edges, where individual weights are zero, is expected to be significantly lower than towards the center of the plot.

We hope to find the generally best values to use. In that case, we can confidently use these weights across different datasets without the need for re-optimization of hyperparameters. If the values are not generally optimal, we still expect to find optimal values for individual lighting directions.

Results



(a) Donut **(b)** Utah teapot

Figure 4.8: Ternary plots showing the PSNR values for different weights. At the edges, parts of the loss function are completely turned off. Axes are scaled to center on an area of interest.

Figure 4.8 shows the resulting search spaces. The teapot dataset (Figure 4.8b) shows a clear peak in the center of the plot, indicating a local optimum for this lighting direction. This combination of weights is also used for the other experiments. In contrast, the donut scene Figure (4.8a) does not show a peak in the same position, but requires

a lower target weight. Therefore, in the current formulation of the loss function, there is no single general optimal combination of weights.

Whether this combination of weights is optimal for the whole teapot dataset is not currently measured. This would require further research.

Around the edges of the ternary plots, where one or more of the loss components are set to 0, the PSNR values drop sharply. This means that all three parts of the loss function are necessary for the interpolation to get good results.

4.5.3 Number of Gaussians

The number of Gaussians created by our method primarily depends on the dynamic Gaussian creation threshold described in Section 3.2. When, during optimization, pixels are found where the error exceeds the threshold t_{est}, an additional Gaussian is added. Increasing and decreasing t_{est} respectively decreases and increases the number of created Gaussians.

We want to understand how this parameter t_{est} and the number of Gaussians affect the reconstruction quality. To achieve this, we calculate interpolations on all interpolation triangles of the donut dataset, varying the threshold t_{est}. We compare the different methods and the reconstruction quality of the estimation step.

We hypothesize that a smaller value t_{est} means a larger number of Gaussians. We expect a higher number of Gaussians to result in a higher PSNR for the Gaussian estimation step, as previous work by Zhang et al. (2025) has shown that a sufficiently large number of Gaussians can accurately represent full-color images.

However, we expect a high number of Gaussians to cause more interpolation artifacts because of tearing or incorrectly moved Gaussians. Additionally, we expect a higher number of Gaussians to mean a lower quality gain from the circular path method compared to the straight path method, because individual, smaller Gaussians contain less information about geometry than larger Gaussians.

Results

Figure 4.11 shows the resulting PSNR ranges. A lower threshold t_{est}, resulting in a higher number of Gaussians, led to a better Gaussian estimation result. This is in line with our hypothesis.

Since the alpha blend method does not rely on the Gaussians at all, the resulting PSNR range stays constant. For any tested threshold, the alpha blend method performs worse than the circular and straight path methods.

There is a visible peak in PSNR values for the circular and straight path methods for $0.1 < t_{est} < 1.0$. Below this peak, the excessive number of Gaussians causes visual artifacts, decreasing the reconstruction

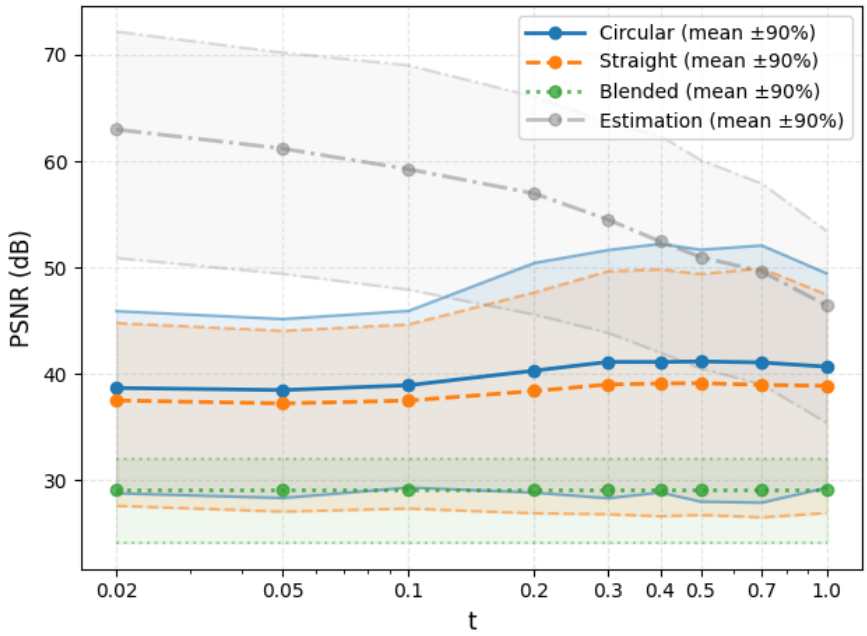


Figure 4.9: Ranges of PSNR values for a varying threshold for Gaussian creation t_{est} after the Gaussian estimation step and for the re-targeting methods, represented by the distribution mean, 5th percentile, and 95th percentile. t_{est} is on a log scale.

quality as described in the hypothesis. Higher values cause a lower reconstruction quality because there are not enough Gaussians to represent the whole structure of the specular highlights sufficiently, as can be seen in Figure 4.10.

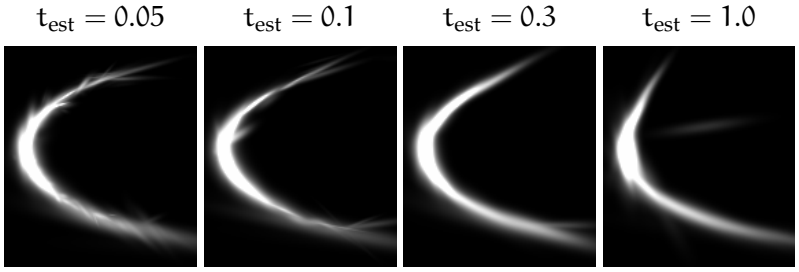


Figure 4.10: Zoomed-in example images from *donut_2* for different threshold values t_{est}. For $t_{est} = 0.05$ and $t_{est} = 0.1$, Gaussians can be seen poking out of the specular highlight. For $t_{est} = 1.0$, there are not enough Gaussians to fully model the specular highlight. $t_{est} = 0.3$ shows a sweetspot inbetween with fewer artifacts.

Contrary to our hypothesis, there is still an advantage in using the circular path method for a large number of Gaussians. If the advantage were lost, the circular and straight method PSNR values would approach each other for a lower t_{est}. We do see a marginal increase in distance between the two methods around $t_{est} = 0.2$ and

$t_{est} = 0.5$ for the upper 95th percentile, though this is less visible in the average PSNR values. This can have two different probable causes:

1. There may be a smaller subset of lighting directions that can more efficiently exploit the circular path method compared to the straight path method. This would make sense, since not all interpolation directions require moving along structures resembling curved edges.

2. There appears to still be some geometric information encoded in the larger numbers of Gaussians. This is probably because there are still some large Gaussians that are used to cover the majority of the specular highlights' area, with smaller Gaussians for minor corrections. These large Gaussians contain the geometric information, which can be exploited by the circular path method. To find out to what extent this is the case, further research is needed.

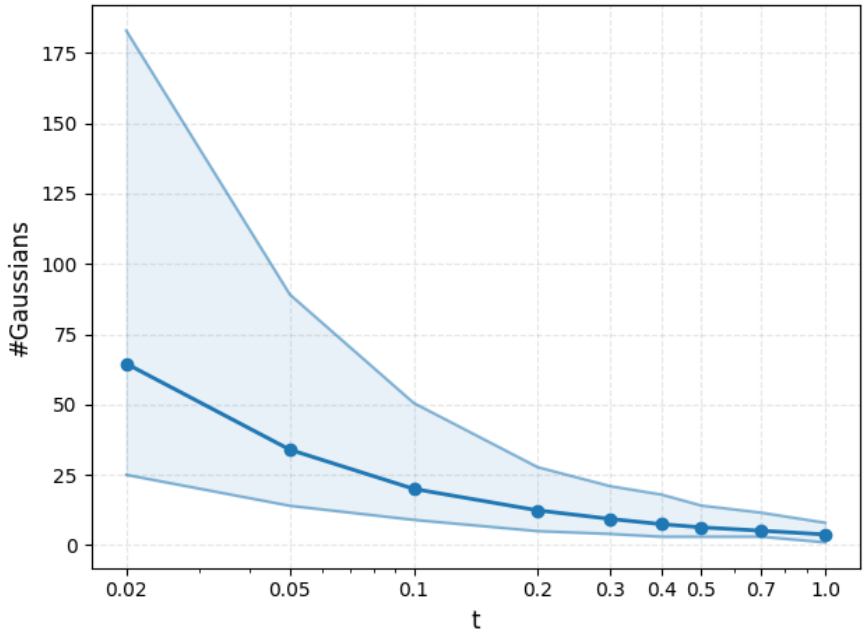


Figure 4.11: The number of Gaussians generated using different threshold values t_{est}. t_{est} is on a log scale.

Figure 4.11 shows that decreasing t_{est} quickly increases the number of Gaussians. Since this can increase computation time and complexity of any further analysis, it can be interesting to select a high enough threshold.

Comparing these observations, there seems to be an optimal value for t_{est} around 0.2 and 0.7. The ranges of PSNR values are relatively consistent within these bounds. For the other experiments, $t_{est} = 0.2$ has been used to keep the resulting Gaussian estimation as good as

possible without decreasing the reconstruction quality of the interpolation steps.

This experiment does not assess whether these results are generalizable between datasets. The optimal threshold is mainly dependent on the range of values we are interested in, which we set to $0-1$. This should not change between datasets. Depending on the level of detail in the dataset, there could still be variance in optimal values. Ideally, the dynamic generation of Gaussians should be able to handle such situations, but further research is needed to confirm this.

4.5.4 Interpolation position within the triangle

The accuracy of an interpolated specular highlight depends on its position relative to the surrounding known specular highlights. We would like to understand how the resulting reconstruction quality varies within an interpolation triangle. Therefore, we calculate the accuracy of multiple interpolations with varying barycentric coordinates within a single triangle in the Delaunay triangulation.

We hypothesize that interpolating further from a known highlight results in a lower reconstruction quality. Interpolations close to a known specular highlight should give more accurate results, since the interpolation result is dominated by the Gaussians from the closest specular highlight. At the known lighting directions, the final accuracy depends only on the accuracy of the Gaussian estimation step, not on the Gaussian interpolation step.

We use the barycentric coordinates corresponding to the lowest average reconstruction accuracy to test the interpolation method in other experiments. This makes the resulting PSNR values more dependent on the interpolation step and less on the accuracy of the Gaussian estimation step.

Results

As shown in Figure 4.12, the PSNR values are highest closer to the known highlights, dropping off towards the center. For this specific triangle, the lowest PSNR values are not in the center of the triangulation but more towards the known specular highlight A. This is because specular highlights B and C are more similar to each other than to specular highlight A. The accuracy at the center of the interpolation is still close to this minimum.

Before performing the interpolation, the location of the lowest reconstruction accuracy is not known. The actual center accuracy is still significantly lower than the accuracy closer to the corners. Therefore, other experiments on the interpolation accuracy focus on interpolations at the center coordinates $\lambda_A = \lambda_B = \lambda_C = 1/3$.

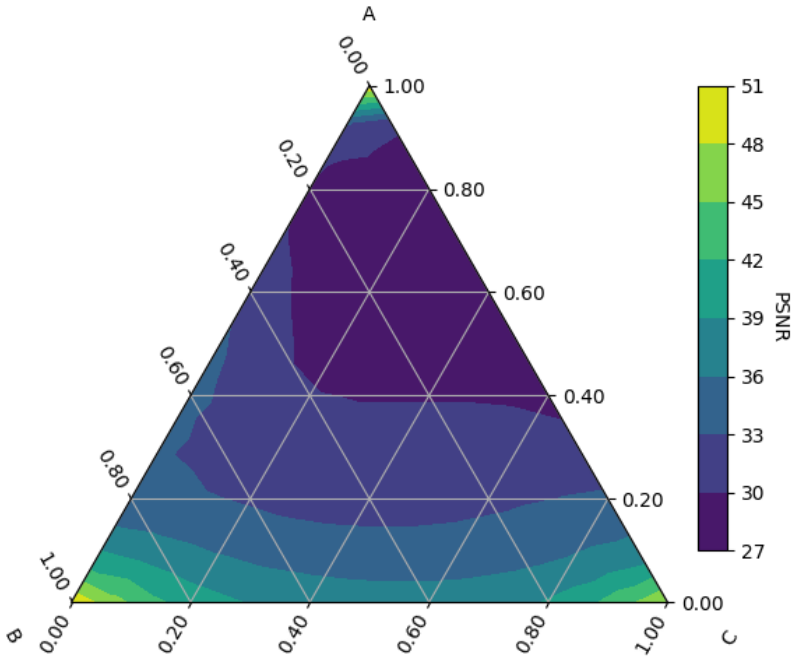


Figure 4.12: PSNR values at different barycentric coordinates for the interpolation triangle of example image *donut_1*. Corners A, B, and C represent the known specular highlights.

4.5.5 Reflectance field density analysis

Sparser reflectance fields are easier to capture and require less storage space. However, a sparser reflectance field can hurt the reconstruction quality of our interpolation. We want to understand how this tradeoff affects our interpolation method and how low we can go.

To make this analysis, we generate additional reflectance field datasets with increasing numbers of lighting directions, using the donut model. We run the interpolation algorithms on these datasets for all triangles in the Delaunay interpolation and calculate their PSNR values.

Modern light stages for reflectance field acquisition produce between 156 and 1111 lighting directions (Debevec, 2012). However, for high-density light stages, lights are often combined to produce around 250-350 usable light directions, speeding up acquisition and reducing the required storage space.

We hypothesize that a higher number of lighting directions significantly increases the reconstruction quality of the interpolation method, though with diminishing returns, limited by the estimation quality. The alpha blend method should generally perform worse, but can surpass the other interpolation methods for a sufficiently high number of Gaussians, since the alpha blend method captures all details that the other methods do not represent when near the known highlights.

Results

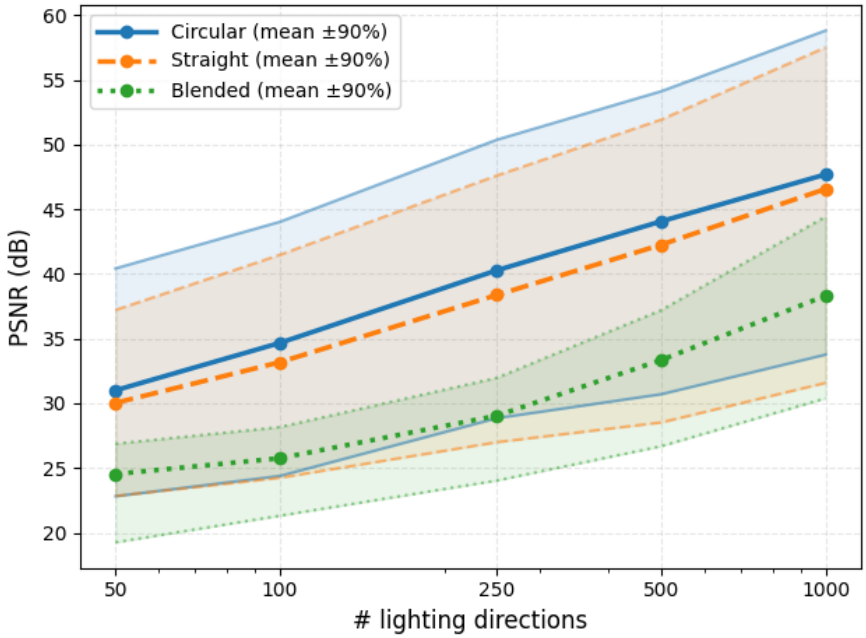


Figure 4.13: Ranges of PSNR values for selected numbers of lighting directions in the reflectance fields, represented by the distribution mean, 5th percentile, and 95th percentile. The number of lighting directions is on a log scale.

Figure 4.13 shows that, across a range of realistic reflectance field densities, the circular path method outperforms the straight path method and the alpha blend method. As densities increase, the alpha blend method improves more quickly, but the point at which it overtakes the circular path method requires an unrealistically high number of lighting directions.

There are clear diminishing returns, requiring approximately a doubling of the number of lighting directions to improve the average PSNR by 4 points. This increase is relatively constant, showing that there are no inherent threshold densities that dramatically affect the resulting reconstruction quality.

Since this increase is relatively constant, it is always advantageous to have more lighting directions. There is no specific threshold below which the results are significantly worse.

4.5.6 Effect of changing resolution

Since the coordinate system used by the Gaussians is normalized, i.e. independent of image size, the interpolation can be performed on a different resolution than the Gaussian estimation and retargeting steps.

Two resolutions can be varied when following our method:

1. **The working resolution** used for the Gaussian estimation and retargeting step.
2. **The interpolation resolution**, which determines the resolution of the final result.

Our goal is to analyze how the reconstruction quality varies with different resolutions. This is relevant because lower working resolutions reduce computation time, while higher interpolation resolutions may be necessary for the required output resolution.

We calculate the Gaussian estimation and retargeting steps for scene *donut_1* using a range of working resolutions, then interpolate these with another range of interpolation resolutions. We plot the resulting PSNR values in a heatmap. These PSNR values are calculated by comparing an interpolated image with a reference image of the same resolution. This way, images with the same amount of detail but different resolutions should get a similar PSNR.

We hypothesize that a working resolution higher than the interpolation resolution won't further increase the reconstruction quality. Contrarily, a lower working resolution than interpolation resolution can lead to a lower reconstruction quality, as the interpolation introduces pixels that were not optimized for. This can lead to Gaussians not correctly covering all required highlight pixels.

Results

The results shown in Figure 4.14 do not support our initial hypothesis. The first observation is that the working resolution of 480x270 results in significantly higher PSNR values than higher or lower resolutions. This suggests that, despite normalizing the coordinate system, the hyperparameters used for Gaussian estimation and retargeting, highlighted in Experiments 4.5.3 and 4.5.2, don't work for different resolutions. This is likely because the specular highlight structures are represented by different numbers of pixels. Consequently, different hyperparameters need to be selected when working in a different working resolution.

Besides this dependency on the hyperparameters, the results from the highest working resolution are higher than those for the remaining resolutions, which suggests that a higher working resolution does result in a better reconstruction quality. However, this is not apparent when comparing resolutions 640x360, 320x180, and 240x135, which give higher PSNR values for lower resolutions. Since these values could also highly depend on the hyperparameters, more research is needed to make this conclusive.

Interestingly, reconstruction quality remains consistent when changing the interpolation resolution, suggesting that there is generally no

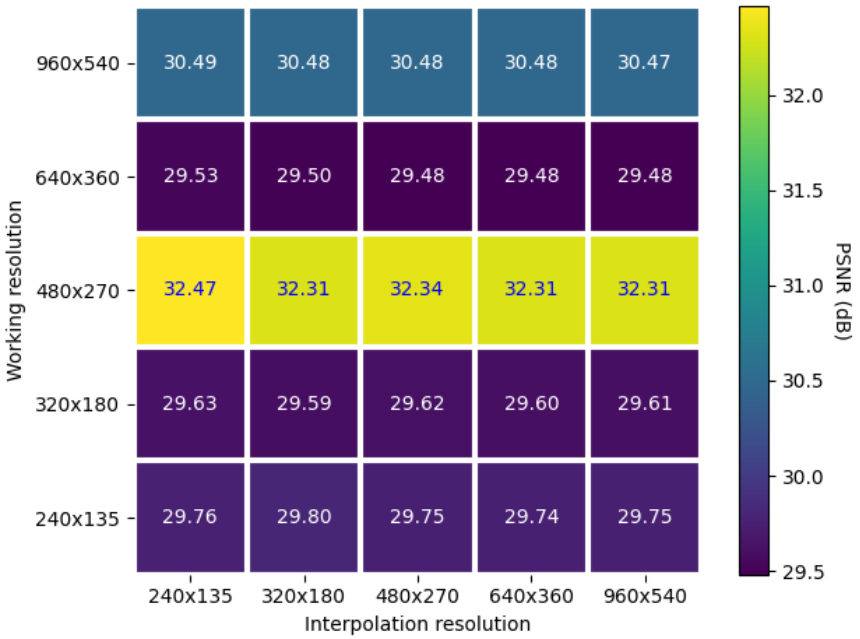


Figure 4.14: PSNR scores for various combinations of working and interpolation resolution.

loss in quality when increasing the interpolation resolution. Further experimentation is needed to determine whether this generalizes to even higher resolutions.

4.6 DISCUSSION

The experiments demonstrate that the method is effective for handling larger specular highlights. However, the method struggles to represent narrow specular highlights on more detailed, high-frequency geometry. As a consequence, the current method is not ideal for arbitrary scenes. The main problem is that the Gaussian estimation step described in Section 3.2 is too scale-dependent to be broadly applicable. Further research can be done to improve this part to create Gaussians on different scales while maintaining the property of using a minimal number of Gaussians to represent specular highlights.

The current method also creates visual artifacts when Gaussians are not positioned correctly. Since these Gaussians are not bounded by any underlying information during interpolation, it is even possible for Gaussians to be moved partially outside the object in the scene. In further research this could be mitigated by post-processing or bounding the Gaussians during retargeting and interpolation.

Additionally, the experiments show that applying circular paths generally increases the reconstruction quality. This primarily means that the additional underlying geometric information we can extract

gives us better positioning of interpolated specular highlights. Representing the specular highlights in a Lagrangian framework facilitates the extraction of this information, making it easier to accommodate non-linear movements of specular highlights. Other methods that we discussed in Chapter 2 do not use this, but rely primarily on straight interpolation paths.

We also found that the interpolation step is fast enough to be real-time, meaning that, if desired, it can be implemented in design software and easily used after precomputation is complete.

The method relies significantly on a number of hyperparameters, which the experiments have shown to be highly dependent on scene and resolution. This makes it less adaptable to different situations. Further research can be done to determine if the optimizations can be made more generally applicable.

5 LIMITATIONS AND CONCLUSION

5.1 LIMITATIONS

NO DIFFUSE LIGHTING RESULTS This thesis focuses solely on interpolating specular highlights, whereas images typically also contain diffuse lighting. We decided to focus only on specular highlights because they presented the most significant opportunity for improvement and best lend themselves to being represented as Gaussians.

However, this means the results are shown as only specular highlights without any diffuse lighting as context clues for visual assessment. Further research can be done to determine if a similar method can be applied to diffuse lighting interpolation and if there are benefits in doing so.

COMPARISON METHODS Due to the time limitation and lack of code availability, we do not have any implementation for the reflectance field interpolation methods presented in Chapter 2. Therefore, we do not have any well-performing interpolation methods to compare our method against.

Instead, we used a simple baseline implementation for comparisons. This has helped to illustrate in which areas our method performs well and where it struggles.

Additionally, we have both the straight and circular path versions of our method. Since the straight path method interpolates along a similar path to the other methods presented, and the circular path method builds on that with a different interpolation direction, we can infer that the advantage of the circular path method can be used to further improve the other methods. Further research is needed to determine if this is indeed the case.

5.2 CONCLUSION

We presented a method for interpolating specular highlights in reflectance field data and conducted experiments to validate the method and explore its functionality. The method shows promise in applying a Lagrangian frame of reference when interpolating specular highlights. It allows for easier extraction of movement directions of specular highlights along edges when underlying geometry information is not available.

However, the current method does not provide optimal results, showing visual artifacts and missing specular highlights. Because of this, more work is needed before a similar method can be used effectively by designers.

BIBLIOGRAPHY

Beckmann, P., & Spizzichino, A. (1963). *The scattering of electromagnetic waves from rough surfaces*. Pergamon Press.

Bi, S., Xu, Z., Srinivasan, P., Mildenhall, B., Sunkavalli, K., Hašan, M., Hold-Geoffroy, Y., Kriegman, D., & Ramamoorthi, R. (2020). Neural reflectance fields for appearance acquisition. https://doi.org/10.48550/arXiv.2008.03824

Blinn, J. F. (1977). Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.*, *11*(2), 192–198. https://doi.org/d95b64

Chen, B., & Lensch, H. P. (2005). Light source interpolation for sparsely sampled reflectance fields. *Proc. Vision, Modeling and Visualization*, 461–469.

Chen, E., Kovačič, Ž., Aggarwal, M., & Davis, A. (2025). Pocket timelapse. *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*. https://doi.org/10.1145/3721238.3730594

Debevec, P., & LeGendre, C. (2022). Hdr lighting dilation for dynamic range reduction on virtual production stages. *ACM SIGGRAPH 2022 Posters*. https://doi.org/n9zp

Debevec, P. (2012). The light stages and their applications to photoreal digital actors.

Debevec, P., Hawkins, T., Tchou, C., Duiker, H.-P., Sarokin, W., & Sagar, M. (2000). Acquiring the reflectance field of a human face. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, 145–156. https://doi.org/10.1145/344779.344855

Eberly, D. (2000). Intersection of ellipses. *Geometric Tools*. https://www.geometrictools.com/Documentation/IntersectionOfEllipses.pdf

Fechner, G. T. (1948). Elements of psychophysics, 1860. *W. Dennis (Ed.), Readings in the history of psychology*, 206–213. https://doi.org/10.1037/11304-026

Fleming, R. W., Torralba, A., & Adelson, E. H. (2004). Specular reflections and the perception of shape. *Journal of vision*, *4*(9), 10–10. https://doi.org/10.1167/4.9.10

Fuchs, M., Lensch, H. P. A., Blanz, V., & Seidel, H.-P. (2007). Superresolution reflectance fields: Synthesizing images for intermediate light directions. *Computer Graphics Forum*, *26*(3), 447–456. https://doi.org/d43gf8

Garg, G. (2006). *Efficiently acquiring reflectance fields using patterned illumination* [Doctoral dissertation, Stanford University].

Gortler, S. J., Grzeszczuk, R., Szeliski, R., & Cohen, M. F. (1996). The lumigraph. *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 43–54. https://doi.org/10.1145/237170.237200

Guo, K., Lincoln, P., Davidson, P., Busch, J., Yu, X., Whalen, M., Harvey, G., Orts-Escolano, S., Pandey, R., Dourgarian, J., et al. (2019). The relightables: Volumetric performance capture of humans with realistic relighting. *ACM Transactions on Graphics (ToG)*, *38*(6), 1–19. https://doi.org/10.1145/3355089.3356571

Kerbl, B., Kopanas, G., Leimkühler, T., & Drettakis, G. (2023). 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, *42*(4), 139–1. https://doi.org/gstrtb

Kingma, D. P. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. https://doi.org/10.48550/arXiv.1412.6980

Kinsman, T. (2016). An easy to build reflectance transformation imaging (rti) system. *The Journal of Biocommunication*, *40*(1), e4. https://doi.org/10.5210/jbc.v40i1.6625

Levoy, M., & Hanrahan, P. (1996). Light field rendering. *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, 31–42. https://doi.org/10.1145/237170.237199

Maček, N., Usta, B., Eisemann, E., & Marroquim, R. (2022). Real-time relighting of human faces with a low-cost setup. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, *5*(1), 1–19. https://doi.org/10.1145/3522626

Malzbender, T., Gelb, D., & Wolters, H. (2001). Polynomial texture maps. *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 519–528. https://doi.org/10.1145/383259.383320

Masselus, V., Dutré, P., & Anrys, F. (2002). The free-form light stage. *Proceedings of the 13th Eurographics Workshop on Rendering*, 247–256. https://doi.org/10.1145/1242073.1242275

Meka, A., Haene, C., Pandey, R., Zollhoefer, M., Fanello, S., Fyffe, G., Kowdle, A., Yu, X., Busch, J., Dourgarian, J., Denny, P., Bouaziz, S., Lincoln, P., Whalen, M., Harvey, G., Taylor, J., Izadi, S., Tagliasacchi, A., Debevec, P., … Rhemann, C. (2019). Deep reflectance fields - high-quality facial reflectance field inference from color gradient illumination. *ACM Transactions on Graphics (Proceedings SIGGRAPH)*, *38*(4). https://doi.org/10.1145/3306346.3323027

Nayar, S., Ikeuchi, K., & Kanade, T. (1990). Determining shape and reflectance of hybrid surfaces by photometric sampling. *Robotics and Automation, IEEE Transactions on*, *6*, 418–431. https://doi.org/10.1109/70.59367

Norman, J. F., Todd, J. T., & Orban, G. A. (2004). Perception of three-dimensional shape from specular highlights, deformations of

shading, and other types of visual information. *Psychological Science*, *15*(8), 565–570. https://doi.org/10.1111/j.0956-7976.2004.00720.x

Phong, B. T. (1975). Illumination for computer generated pictures. *Commun. ACM*, *18*(6), 311–317. https://doi.org/bkfrm9

Schechner, Nayar, & Belhumeur. (2003). A theory of multiplexed illumination. *Proceedings Ninth IEEE International Conference on Computer Vision*, 808–815 vol.2. https://doi.org/10.1109/ICCV.2003.1238431

Sie, G. (2009). Ticino Big Pilot B/W [Photograph]. https://flic.kr/p/6ktcyu

Sun, T., Xu, Z., Zhang, X., Fanello, S., Rhemann, C., Debevec, P., Tsai, Y.-T., Barron, J. T., & Ramamoorthi, R. (2020). Light stage super-resolution: Continuous high-frequency relighting. *ACM Transactions on Graphics (TOG)*, *39*(6), 1–12. https://doi.org/10.1145/3414685.3417821

Xu, Z., Sunkavalli, K., Hadap, S., & Ramamoorthi, R. (2018). Deep image-based relighting from optimal sparse samples. *ACM Trans. Graph.*, *37*(4). https://doi.org/10.1145/3197517.3201313

Zhang, X., Ge, X., Xu, T., He, D., Wang, Y., Qin, H., Lu, G., Geng, J., & Zhang, J. (2025). Gaussianimage: 1000 fps image representation and compression by 2d gaussian splatting. In A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler, & G. Varol (Eds.), *Computer vision – eccv 2024* (pp. 327–345). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-72673-6_18