Scene Editing using Polarization

Real-World Scene Editing using a Polarization-based Intrinsic Image Decomposition

Amir Zaidi

TU Delft

Scene Editing using Polarization

Real-World Scene Editing using a Polarization-based Intrinsic Image Decomposition

by

Amir Zaidi

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday August 30th, 2022 at 11:30 AM.

Student number:	4704258	
Project duration:	November 30th, 2021 – August 30th, 2022	
Thesis committee:	Prof. dr. E. Eisemann,	TU Delft, supervisor
	Prof. M. Weinmann,	TU Delft
	X. Zhang,	TU Delft

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

This thesis was written as part of the graduation procedure at Delft University of Technology. The topics of polarization and scene editing fit within the context of Computer Graphics and Visualization, and combining the two allowed me to present my capabilities for the Master's degree in Computer Science.

Various topics are explored and combined throughout the thesis to compose an algorithm which may bring new interest to the usage of the linear polarization of visible light in the field of computer vision. First, a simple but intuitive definition of the polarization information that is measured by a camera is introduced. Second, an information propagation loop estimates two separate light components of the image: the diffuse light and the specular light. Last, these separate light components are used in existing scene editing operations to showcase the value of the decomposition of the two components.

Prof. Dr. Elmar Eisemann, who has been my professor for the Bachelor's course Computer Graphics and the Master's courses Computer Graphics and Seminar Computer Graphics, has guided me during the thesis alongside Prof. Michael Weinmann, who offered to start attending the presentations on the progress and extensively helped with improving the overall quality of the thesis. These two members are joined on the thesis committee by Assistant Prof. Xucong Zhang, who has agreed to be included as the third and final member.

Amir Zaidi
Delft, August 2022

Contents

1 Introduction

In the field of computer vision, the difficulty of processing the illumination in an arbitrary scene creates complications for a machine's understanding of the scene. Computer vision algorithms commonly use input images acquired by a camera, where the only measured quantity is the amount of light the camera's sensor receives without any additional information about the shadows, occlusion, highlights, light directionality and interreflections, to help with scene processing and understanding. This thesis applies polarization state measurements of the camera's incoming light to implicitly extract information on these illumination phenomena through a polarization-based intrinsic image decomposition. The decomposition estimates the diffuse and specular components of the incoming light. These components can be used to improve existing scene post-processing operations such as material editing and tonemapping.

Shadows and highlights, both found in nearly every scene, are complex phenomena that can confuse a machine if it is not fine-tuned to handle the input data. For example, dark patches in images could be either objects with a light-absorbing surface material, or hard shadows created by blocked directional light. These hard shadows could be classified and removed using local segmentation algorithms. However, the more common type of shadow is the more difficult to segment soft shadow, fading from light gray to pitch black in occluded areas due to ambient occlusion, creating fuzzy shadow boundaries. These are harder to remove due to the challenge in estimating the gradient. Opposite on the luminosity histogram, highlights range from minuscule bright white points to large circular areas barely brighter than their surroundings. There are view-dependent appearance differences in the characteristics of the highlights, because both the incoming and outgoing light, also called irradiance and radiance, are highly directional. This directionality is what creates the appearance of highlights and causes their visual displacement when the view changes. The directional behaviour of the light creating the highlights depends on the materials of the objects they bounce off and the surface texture of those objects on a microscopic level. Without a perfect model of the objects and the scene, predicting the appearance and position of the highlights in a scene is at least as difficult as the classification of shadows.

Because the light intensity on the sensor is the only directly available data from a camera, algorithms usually process the intensity directly without attempting to understand the illumination. There are global illumination models which can render computer-modelled scenes with accurate lighting. Their implementations emulate all the real-world phenomena created by illumination separately, and then merge them for the final output image. However, this is generally a one-way pipeline, and cannot be inverted to separate the components again from only the output image. Work on an inverse rendering pipeline is ongoing, but the existing attempts at such a pipeline either require additional material and lighting information, or use a neural network to guess this information. An approximation of such an inverse model that does not require additional input information could provide insight into the decomposition of camera samples, which are similar to the output image of a global illumination pipeline. Calculating which illumination components are merged into the measured light intensity is a difficult task, due to the lack of constraints on what the values of the components can be. Still, a good approximation would improve many computer vision algorithms that process camera samples from real-world scenes, as they would then have direct access to an estimation of each of the illumination components separately.

The concept of intrinsic images was introduced as a way to decompose an image into separate components, which can then either be summed or multiplied to find the input image again. The intrinsic compo-

nents can take many different forms, but they are always orthogonal characteristics of the input image, with no data shared between them. The characteristics do not have to be related to the illumination. For instance, a decomposition into small-scale and large-scale features is a valid intrinsic image decomposition.

One particular decomposition that is useful to find more data on the illumination is the diffuse-specular intrinsic image decomposition, which separates the diffuse object colors invariant to the viewpoint from the specular highlights varying with viewpoint. Although this decomposition has been attempted with only one light intensity input image, there is usually not enough information to complete such a decomposition, and additional information of the scene is required for good results. For example, the scene could be captured multiple times with different directional light sources in each sample. This provides a way to approximate the diffuse component, and subsequent images from that scene can then be directly decomposed by subtracting the previously measured diffuse component.

Setting up the illumination sources to compute a diffuse image is a time-intensive operation that might even be impossible when the external illumination cannot be controlled. An alternative that does not require controlling the illumination would massively speed up the process. The polarization of visible light might be the key to finding the necessary information for the diffuse-specular decomposition without any specific illumination requirements. The 1990s saw a surge of research interest in polarization due to its correlation with illumination angle and material type. Measuring polarization is trivial with a linear polarization filter in front of the camera, but merely knowing the amplitude and phase of the linear polarization is not enough to directly find shadow and highlight information. The theories that were developed two decades ago were either never tested in real systems, or merely tested on a small data set due to a lack of processing power. The increase in publicly accessible processing power since then has made it possible to now execute the previously proposed algorithms for the extraction of additional scene information using polarization in seconds rather than hours.

This thesis builds upon one of the previously developed theories on the separation of the diffuse and specular intrinsic images using only a set of four linear polarization-state measurements and without further knowledge of the scene. The diffuse-specular decomposition is first crudely approximated using the correlation between two different polarization state parameters. A pyramid-based information propagation algorithm then iterates towards a locally optimal separation between the diffuse and specular component. This thesis additionally presents multiple post-processing operations to change the materials and illumination in the scene, and shows how tonemapping algorithms can retain more lighting cues by using the diffuse and specular intrinsic images.

2

Theoretical Background

This section introduces the fundamental basics related to the proposed approach and the necessary notations.

2.1. Pixels and Vectorized Operations

In this thesis, an image I is a collection of data points aligned in a two dimensional lattice grid, where each data point on this grid is a pixel p. In such a grid, a pixel p can be indexed by its two integer coordinates (x_p, y_p). The first coordinate x_p refers to the horizontal location of the pixel on the grid, ranging from 0 to the width minus one. The second coordinate y_p refers to the vertical location of the pixel, ranging from 0 to the height minus one. The intensity or value of pixel p is historically denoted with the notation I_p, but we use the notation $I(p)$ where p is a function argument, while the subtext a in I_a is instead used to refer to the image named a. The intensity $I(p)$ can be single-dimensional or multi-dimensional depending on the type of data the image I describes. For example, a color image from a camera typically has three dimensions per pixel p, while a monochrome image only has one dimension. The only constraint on the number of dimensions per pixel is that it must be fixed for the entire image.

Arithmetic operations can be applied to the intensities $I(p_i)$ of one or more pixels p_i and use the multi-dimensional intensities of these pixels as vectors. $I(p) + I(q)$ linearly adds the intensities of p and q using vector addition regardless of the amount of dimensions. Whenever the specific pixel p is irrelevant and such an operation is applied at once on all pixels of an image I, the pixel identifier argument p is left out of $I(p)$ and only I is used. For example, if I_1 and I_2 are two images of the same size, adding $I_1(p)$ and $I_2(p)$ for every pixel p can be written with the simplified notation $I_{1+2} = I_1 + I_2$. Such an operation is vectorized, meaning that there is no interdependence between pixels, and all resulting pixels can be calculated in parallel.

2.2. Color Dimensions

Sensing and processing a bundle of light is difficult for a machine due to the amount of information contained in any bundle of light. Light is a large sum of waves, where each wave has its own wavelength λ. This sum of waves can be described through a spectral power distribution, which is a function mapping each wavelength λ to its respective intensity $L(\lambda)$ in the final waveform. However, wavelengths are not discretized. Due to their continuous nature, the spectral power distribution of incoming light is infinitely dense and cannot be directly processed by a digital computer, requiring a reduction to a finite amount of dimensions. Rather than attempting to measure the entire spectral power distribution, cameras can integrate the spectral power distribution into one or more discretized dimensions. Most consumer cameras apply tristimulus integration to reduce the spectral power distribution to three dimensions, similar to how human eyes perceive color with a red, a green and a blue component. Such cameras capture exclusively how humans perceive the scene rather than all the wavelengths in the spectral power distribution creating that perception. Data in this 'RGB color space' can be uniquely displayed to human eyes again without any loss of color information. However, the dimensionality reduction raises the question of whether theories developed for the complete spectral power distribution $L(\lambda)$ would still work on the integrated data I captured by these cameras.

If the spectral power distribution at one pixel p is $L(p, \lambda)$ and $R(f(\lambda))$ is a function integrating $f(\lambda)$ with a spectral sensitivity peaking at 'red' light wavelengths, then the first component of $I(p)$ is $I^R(p) = R(L(p, \lambda))$.

$I^G(p)$ and $I^B(p)$ are defined similarly for the green and blue components of $I(p)$. Because R is defined to be an integration, replacing $f(\lambda)$ with a linear combination of two functions will give the same result as integrating the two functions separately:

$$R(f(\lambda)+g(\lambda)) = R(f(\lambda)) + R(g(\lambda)) \tag{2.1}$$

This holds regardless of the sensitivity curve of R. $I(p)$ is a three-dimensional vector, with each component separately having this linearity:

$$\begin{aligned} I(p) &= \begin{bmatrix} R(L(p,\lambda)) & G(L(p,\lambda)) & B(L(p,\lambda)) \end{bmatrix} \\ &= RGB(L(p,\lambda)) \end{aligned} \tag{2.2}$$

If L is a summation of two separate spectral power distributions $M+N$, then the resulting I will also be a summation of two three-dimensional components that are the separate tristimulus integrations of M and N:

$$\begin{aligned} I(p) &= RGB(L(p,\lambda)) \\ &= RGB(M(p,\lambda)+N(p,\lambda)) \\ &= RGB(M(p,\lambda)) + RGB(N(p,\lambda)) \\ &= I_M(p) + I_N(p) \end{aligned} \tag{2.3}$$

This shows that reasoning about summed values in the three-dimensional measurement $I_M + I_N$ is analogous to reasoning about summed spectral power distributions $M+N$. All theories developed for the full range of light wavelengths will work on three-dimensional color data from a sensor as long as the calculations performed are linear.

2.3. Intrinsic Image Decomposition

The diffuse-specular decomposition is only one existing intrinsic image decomposition out of many. A general intrinsic image decomposition is a separation of a sum $I_1+I_2+...+I_n = I$ or multiplication $I_1 * I_2 * ... * I_n = I$ of separate components I_i back into components from only input image I, and each component I_i denotes a unique characteristic of I. Each component I_i is then one intrinsic image of I, revealing exclusively the characteristic that the intrinsic image represents. The diffuse-specular intrinsic image decomposition is such a separation of two characteristics relevant to our goal of understanding the illumination in the scene. In this decomposition, image $I = I_d + I_s$ is decomposed into the two intrinsic images I_d and I_s, which are the angle-invariant diffuse image created by diffuse light, and the angle-variant specular image created by specular light.

The diffuse-specular decomposition is modelled based on the components of radiance, the outgoing light from objects in the physical world. On a microscopic level, the light L emitted at one point of an object into an arbitrary direction is, according to the dichromatic reflectance model, the summation of the diffuse emission from the body of the object L_d and the specular reflection from the surface of the object L_s:

$$L(\lambda) = L_d(\lambda) + L_s(\lambda) \tag{2.4}$$

where L, L_d and L_s are the spectral power distributions of the total radiance and the separated diffuse and specular radiance components.

Since this is a linear operation, the theory of Section 2.2 is applicable. This implies that such a linear diffuse-specular decomposition is correct after a reduction to any number of color dimensions:

$$I = I_d + I_s \tag{2.5}$$

Assume all objects are ideal diffuse radiators following Lambertian reflectance. At any pixel p, the diffuse component $I_d(p)$ is then only dependent on the angle of the light source to the object but independent of the angle of the camera to the object, while the specular component $I_s(p)$ is highly dependent on this angle of the camera to the object, shifting from zero to the local maximum when the angle changes. This property can be used by applying minor shifts to the camera position and comparing the difference to find $I_s(p)$. However, our approach does not use any camera movement, and instead focuses on polarization for this separation.

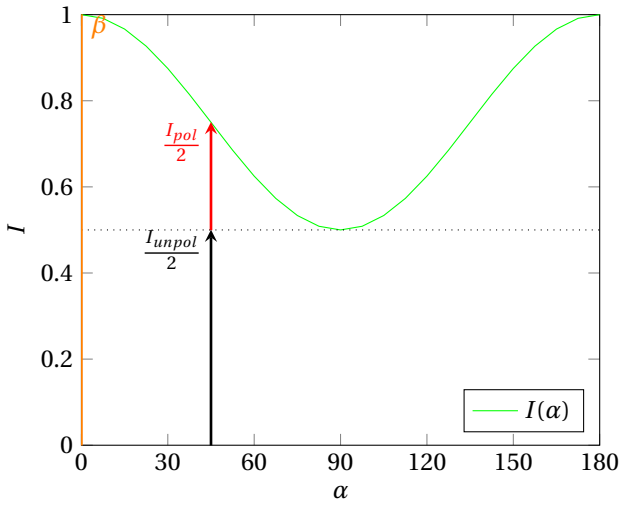

Figure 2.1: One color dimension of a possible configuration of the linear polarization of light: $I(\alpha) = \frac{1}{2} + \frac{1}{2} * (1 + cos(2 * \alpha))$, where the linear offset $I_{unpol} = 1$, the cosine size $I_{pol} = 1$ and the cosine starting phase $\beta = 0°$.

2.4. Linear Polarization of Light

The linear polarization of light is a useful property for the diffuse-specular decomposition. Non-conductive objects, also called dielectric objects, almost exclusively radiate diffuse light without polarization. This then implies that all polarized light must be part of the object's specular light, which gives additional constraints on the specular light when the amount of polarized light is known.

Linear polarization can be understood as a direction of photon oscillation in the 2D plane orthogonal to the direction the light is travelling towards. Measuring the direction of the polarization of one or more photons is achieved through a linear filter. A photon polarized in precisely the same orientation as the filter will always pass the filter, while a photon polarized in the orthogonal orientation will never pass it. Any photon with an orientation in between follows the rules of quantum superposition, where the measurement collapses the superposition to either a pass or block state. The chance of passing the filter follows a cosine function dependent on the orientation of the filter relative to the orientation of the photon. We use ° as the symbol to denote an angle in degrees, such that $180° = \pi$. This cosine function has a period of 180°, with α the orientation of the filter and α_i the polarization orientation of photon i:

$$P_i(\alpha) = \frac{1 + cos(2 * (\alpha - \alpha_i))}{2} \tag{2.6}$$

One photon can only be captured once, so measurements are done on a large bundle of light, where the result finds how many photons have passed the filter (see Figure 2.1). This is an addition of many different cosine functions P_i, which either causes constructive interference or destructive interference depending on their respective α_i values:

$$I(\alpha) = \frac{I_{unpol}}{2} + \frac{I_{pol}}{2} * (1 + cos(2 * (\alpha - \beta))) \tag{2.7}$$

For illustration, if all photons are randomly polarized in some direction, I_{pol} will be 0, and the average intensity of the light passing will be half of the incoming light: $I(\alpha) = \frac{1}{2} I_{unpol} = \frac{1}{2} I$. If half of the photons are polarized horizontally and the other half randomly, 75% of the light will pass a horizontal filter, and only 25% will pass a vertical filter.

2.5. Soft Decision Making

Local patches on object parts with similar material characteristics should have similar polarization information, and segmenting such patches to use the average of the polarization state of an entire patch would help in making a polarization-based decomposition algorithm more robust to sensor noise. However, segmenting object patches with hard boundaries requires making definitive decisions for where to place those boundaries. This approach towards segmentation for any purpose has drawbacks due to the difficulty in finding

a decision threshold that works for a majority of scenarios and with varying noise levels. An alternative to segmentation is using weighted inclusion of all pixels into local patches, where the weights are determined by how likely any pair of nearby pixels are to be part of one patch. The bilateral filter is an applicable filter for such soft decision making. It is a filter that by default selectively blurs an image by using two blurring criteria: the spatial distance between pixels, quantified by a function $f(\Delta x)$, and the tonal distance between pixels, quantified by a function $g(\Delta I)$. These two functions are usually normal distributions with a manually chosen standard deviation:

$$f(\Delta x) = e^{(\Delta x)^2/\sigma_f^2} \tag{2.8}$$

$$g(\Delta I) = e^{(\Delta I)^2/\sigma_g^2} \tag{2.9}$$

The two blurring criteria are multiplied to create one blur weight in the bilateral filter, which then compares each pixel q from the input image with each pixel p from the input image, and then sums all p weighted by the previously defined blur weight:

$$J_q = \frac{1}{k(q)} \sum_{p \in \Omega} f(p-q) * g(I(p) - I(q)) * I(p) \tag{2.10}$$

$k(q)$ is a normalization term, which is needed because the weights do not sum to 1:

$$k(q) = \sum_{p \in \Omega} f(p-q) * g(I(p) - I(q)) \tag{2.11}$$

Ω denotes the set of all pixels in the full input image I. q, one pixel from this set, is identified by the spatial coordinates (x_q, y_q) of the input pixel, and p by the spatial coordinates (x_p, y_p) of another pixel being compared. $I(q)$ is the pixel intensity of the input pixel q, and $I(p)$ is the pixel intensity of the pixel being compared. Both $I(q)$ and $I(p)$ can contain any number of color dimensions.

The basic bilateral filter has a neighbourhood as large as the entire image for each pixel q, but this is computationally expensive and unnecessary due to the spatial weight f becoming close to zero at large distances. In a real system, Ω can be replaced by $N(q)$ which denotes a square neighbourhood around a pixel q, usually with a size of 3-by-3, 5-by-5, or 7-by-7 pixels:

$$J_q = \frac{1}{k(q)} \sum_{p \in N(q)} f(p-q) * g(I(p) - I(q)) * I(p) \tag{2.12}$$

$$k(q) = \sum_{p \in N(q)} f(p-q) * g(I(p) - I(q)) \tag{2.13}$$

Algorithm 1 reveals how the bilateral filter can be vectorized, which is a powerful property due to the widespread adoption of GPUs. Each operation processes an entire image at once. This algorithm can additionally be adopted into a fully parallelized per-pixel kernel that does not require any synchronization breakpoints until the end of the algorithm.

Algorithm 1 Vectorized Bilateral Filter

```
procedure VECTORIZED-BILATERAL(I, r, f, g)
    weights ← 0 * I
    totals ← 0 * I
    for dy ← −r to r do
        for dx ← −r to r do
            I_shift ← shift(I, [dx dy])
            w ← f([dx dy]) * g(I − I_shift)
            weights ← weights + w
            totals ← totals + w * I_shift
        end for
    end for
    return totals/weights
end procedure
```

3

Related Work

Related work includes developments regarding intrinsic image decompositions, most notably the diffuse-specular decomposition, and the usage of polarization information to improve this decomposition. Furthermore, work regarding decision-making is relevant for the decomposition algorithm itself. Finally, the work in three applications of this decomposition is important to evaluate its beneficiality in computer vision. These different types of works will be discussed in the following sections.

3.1. Diffuse-Specular Decomposition

There are several works that attempted to solve the diffuse-specular intrinsic image decomposition in the past. Shafer et al. (Shafer, 1985; Klinker et al., 1988) formulated multiple properties of the diffuse and specular components of images from real-world scenes, and used these properties to extract the highlights. They derived that tristimulus integration over all wavelengths of light to reduce the infinitely dense color space to a three-dimensional color space does not change the linear nature of the specular highlights being added to a diffuse image. They then made various assumptions about the real-world scenes in their samples to find the diffuse and specular components from only one single image. Most of these assumptions are unnecessary when the polarization of the light is known. One assumption that is still relevant is that due to this additive nature of the diffuse and specular light, the colors perceived from a dielectric constant-material object lie in a parallelogram in 3D space. The parallelogram is defined by the two basis vectors I_d and I_s (see Figure 3.1). We do not explicitly calculate a local parallelogram, but use it as a reason for why the diffuse color vector direction and specular color vector direction are locally constant.

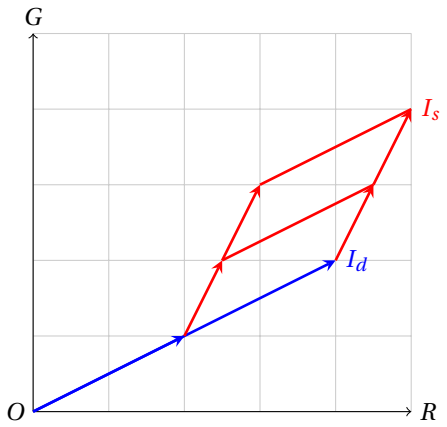

Figure 3.1: Parallelogram in *RG* color space created by the addition of non-parallel color vectors I_d and I_s. In *RGB* color space, I_d and I_s have three dimensions, but create a similar parallelogram. The angle of the plane defined by the bounds of the parallelogram in three dimensions sets limits on the possible values of I_d and I_s.

3.2. Polarization-Based Decomposition

Wolff (L. Wolff, 1989; L. Wolff, 1990) experimented on the properties of polarization data, and laid the groundwork for its incorporation into decomposition and surface estimation algorithms. Light can be both circularly and linearly polarized, but Wolff limited his testing to linearly polarized light. He then devised a way of measuring linear polarization using four different orientations of a linear polarization filter in 45 degree increments: 0 (vertical), 45, 90 (horizontal) and 135 degrees. Using this data, he could calculate both the phase and amplitude of the polarization by fitting a cosine wave onto the four measurements using a minimization of mean squared error approach. Wolff then used these cosines to set a lower bound on the Fresnel ratio for every pixel in the image, which is an accurate estimation for the Fresnel ratio itself when the diffuse component is small. From these lower bounds he could segment objects into material types, as the Fresnel ratio is much higher for dielectric materials such as plastic than for other materials such as metal. Wolff estimates the diffuse intrinsic image by plotting the polarization data of pixel patches in polarization coordinate space, with the two coordinates set as the lowest and highest possible pixel intensity, and searching for a linear slope across all data points. The parameters of the linear slope give the Fresnel ratio, which can then be used to estimate the diffuse component. The method employed by Wolff requires a segmentation of the image to determine which pixels are part of one patch to calculate the linear slope. It also assumes that the diffuse component is constant across this patch, which is not necessarily true.

Nayar et al. (Nayar et al., 1993) created a reflection model, based on Wolff's research, that can be used to decompose an image into the diffuse and specular components using polarization imaging, which they used for stereo vision in Bhat and Nayar, 1994. They found that the specular light can be decomposed into the specular constant term I_{sc}, the amount of specular light present on average when the polarization filter is fully rotated, and the specular varying term I_{sv}, the amount of polarized light that can pass through or be blocked by the filter depending on the filter orientation. The diffuse-specular decomposition problem can then be reformulated to finding I_{sc}, as I_{sv} is known from the measurements. Nayar et al. then used the assumption that the Fresnel ratio is locally similar to propagate an estimation of the ratio from highly specular points onto neighboring points, while labeling points with a low amount of polarization exclusively diffuse. Their algorithm has multiple iterations per pixel, but fixes the value of the ratio estimation for a pixel once it is confident in the estimation. This thesis is inspired by the work in Nayar et al., 1993, and builds upon their foundation. Wolff and Nayar (L. B. Wolff et al., 1998) later worked on a complete theory on reflection models for digital simulations, using their previously developed theory.

Schechner et al. (Y. Schechner et al., 2001; Y. Y. Schechner et al., 2003) used the work of Wolff and Nayar to remove haze from a scene in daylight. By measuring the polarization state of incoming light and assuming that the degree of polarization is determined by the amount of haze, it becomes possible to isolate this haze. Our thesis does not assume that the degree of polarization is exclusively determined by haze, and instead assumes all polarized light is created by specular reflections.

Takatani et al. (Takatani et al., 2018) combine the data from a linearly polarized filter and a circularly polarized filter to classify materials. They control the illumination by additionally placing a linearly polarized filter in front of the illumination source to create fully polarized incoming light when taking measurements, followed by the same experiment with a circularly polarized filter in front of the illumination source. The polarization phase of light is affected differently by bounces of photons on different types of materials, and this can be used for material classification and decomposition. While controlling the illumination is ideal for experimentation, our thesis focuses on a more general approach that does not require controlling the illumination.

3.3. Bilateral Filter Extensions

A soft decision-making framework can counteract the effect of local optima in a hard decision-making iterative estimation algorithm labeling pixels as having fully converged, which fixes their value at the local optimum. If the next iteration then finds a different optimum, the previously found value cannot be modified anymore. In the case of a soft decision-making algorithm, the pixels in the local optimum would be assigned a very low modification weight in the next iteration if the local optimum is considered an accurate estimation already. However, given enough iterations these pixels could leave the local optimum and converge to a different value instead. Such a soft-decision making framework can be based on a simple local filter that processes a current estimate into a new estimate.

Filters are operators that calculate each pixel of a new image based on one or more input images. One simple example of a filter is the smoothing filter, which equally blurs each pixel of an input image to reduce noise.

However, a naive smoothing filter has the drawback of also blurring edges and other details, which might improve the quality of the result when preserved depending on the use-case. An edge-preserving smoothing filter could accomplish this by selectively blurring the areas with a low amount of details. Tomasi and Manduchi (Tomasi and Manduchi, 1998) proposed such an edge-preserving smoothing filter and named it the bilateral filter, which quickly became popular due to its speed and ease of implementation.

An important later extension of the bilateral filter was the cross bilateral filter formulated in Eisemann and Durand, 2004, also called the joint bilateral filter in Petschnigg et al., 2004. The cross/joint bilateral filter uses a different input image K for the weighted values than is used for the weight calculation. Such an extension is useful when local soft decisions have to be made from one image to filter another image. In Eisemann and Durand, 2004, the soft decision determines how to weigh a sum of an image taken with a flashlight and an image taken without flashlight. Extending the cross/joint bilateral filter with one additional change makes it a perfect candidate for a soft decision-making framework that works for an arbitrary decision criterion.

3.4. Real-World Scene Editing

The previous work on diffuse-specular decomposition was mostly focused on the removal of highlights as the final goal. However, such a decomposition has multiple applications in the context of real-world scene editing, with the following relevant works that can be used to test the performance of the decomposition.

Stroia-Williams et al. (Stroia-Williams et al., 2010) presented a material editing technique using an estimated albedo of objects. They separated the shading from the albedo using statistical analysis, and then directly replace the old albedo with the desired new albedo. They call this replacement 'reflectance transfer'. The replacement is fast enough to be used for real-time video editing. This technique may be combined with a new albedo estimation based on the diffuse-specular decomposition to improve the quality of the reflectance transfer.

Phong (Phong, 1975) developed a lighting model for computer rendered images, which summed an ambient, diffuse and specular component for the output image. Blinn (Blinn, 1977) modified the specular component of Phong's model to remove the hard boundary on the specular component when the observer was close to the light source. This became known as the Blinn-Phong model, which is still used in rendering pipelines. Although this model was developed for rendering rather than modelling real life, the Blinn-Phong model is similar to the decomposition in this thesis, with the difference being a separation between the ambient and diffuse components I_a and I_d, which we sum into only the diffuse component I_d. The Blinn-Phong model is relevant for this thesis due to its definition of material shininess in computer generated scenes.

Durand and Dorsey (Durand and Dorsey, 2002) created a fast tonemapping operator using the bilateral filter. This tonemapping operator reduces the high dynamic range of a picture from a camera, to be displayed on a standard dynamic range monitor. Having the specular light separated could improve the performance of their tonemapping operator.

Meylan et al. (Meylan et al., 2007) developed an inverse algorithm that detects and extends the dynamic range of already tonemapped specularities for display on high dynamic range monitors. With a diffuse-specular decomposition, the algorithm could be directly applied without needing any specularity detection. Their work also gives insight into how highlights can be mapped to change their dynamic range without affecting their perception.

4

Our Method

The main goal of our algorithm is to decompose I into I_d and I_s by using measurements of the polarization state. These measurements only provide information on the linear polarization defined in Section 2.4, and do not directly provide enough information to find I_d and I_s. The diffuse-specular decomposition in Nayar et al., 1993 converts the measurements into the parameters of a cosine, which can be used to find the color vector direction of I_s. We use a similar conversion to transform four measurements into a 'polarization cosine'. The amplitude parameter I_{amp} of this polarization cosine, which has one value for each color dimension, can then be weighted with a weight map k to calculate $I_s = k * I_{amp}$ and $I_d = I - I_s$. Initially, the values for the weight map are estimated using tile-based correlation, which is a fast estimation that is accurate when the specular light only consists of highlights on objects. Starting from this estimation, our method uses an iterative approach to improve the weight map under two local constraints similar to Nayar et al., 1993: Fresnel ratio constancy and diffuse color vector direction constancy. We refer to this step as the 'local optimization' step, because it optimizes the current estimate by applying these two local constraints. A new functional bilateral filter is combined with a vectorized line intersection for the local optimization step. These two techniques make the decomposition more robust to sensor noise than previous works. Additionally, a Gaussian pyramid is applied to the information propagation to accommodate a wide range of input image sizes, and to introduce a quality-runtime trade-off which can be modified for different use-cases. Figure 4.1 presents a high-level visual overview of the subroutines of our method.

4.1. Polarization Cosine

Polarization measurements fitting on a cosine has been known since the work by Wolff (L. Wolff, 1990) and was used by Nayar et al. (Nayar et al., 1993) and Schechner et al. (Y. Schechner et al., 2001), but no name

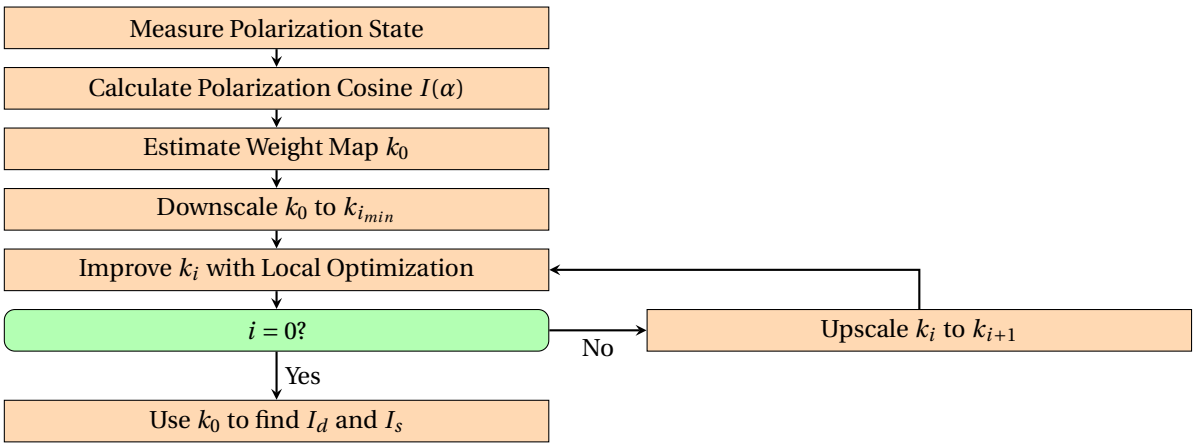

Figure 4.1: Flowchart of the subroutines used in our method. Each orange block is executed, while each green block is a decision point. The arrows indicate the flow between the subroutines.

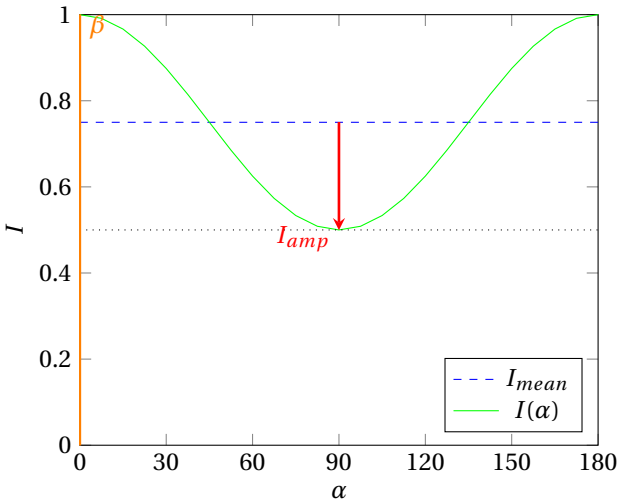

Figure 4.2: The linear polarization of light plotted as a polarization cosine: $I(\alpha) = \frac{3}{4} + \frac{1}{4} * cos(2 * \alpha)$, where the linear offset $I_{mean} = \frac{3}{4}$, the cosine size $I_{amp} = \frac{1}{4}$ and the cosine starting phase $\beta = 0°$. This is an alternative formulation to the plot in Figure 2.1.

has been given to the cosine itself. We refer to it as the 'polarization cosine' $I(\alpha)$, which consists of a discrete offset I_{mean}, an amplitude I_{amp}, and a polarizing filter phase β where the output is maximal (see Figure 4.2):

$$I(\alpha) = I_{mean} + I_{amp} * cos(2 * (\alpha - \beta)) \tag{4.1}$$

I_{mean} will be referred to as the polarization mean, and I_{amp} as the polarization amplitude. Even though the polarization cosine has the phase β, the phase information is not necessary for the diffuse-specular decomposition and can be discarded. Equation 4.1 is a simplification of Equation 2.7 where $I_{mean} = \frac{I_{unpol} + I_{pol}}{2}$, and $I_{amp} = \frac{I_{pol}}{2}$. Although this formulation was mentioned by Nayar et al., using these two cosine parameters directly is different from their usage of the parameters to find the cosine's lower and upper bounds I_{min} and I_{max} which they then use for their method.

Each pixel has a value for the polarization mean and amplitude in each color dimension. I_{mean} and I_{amp} vary between color dimensions, which is crucial to the eventual decomposition, as the vector direction of the color vectors $I_{mean}(p)$ and $I_{amp}(p)$ at a pixel p can be compared to find more information about the illumination at pixel p. With the assumptions that only specular light can be polarized and each color channel has an equal amount of relative polarization, the vector direction of the specular light $I_s(p)$ at a pixel p has to be equal to the vector direction of the polarization cosine amplitude I_{amp}, which is the difference in intensity across polarization angles. These two assumptions are valid for most dielectric objects. Because the vector direction of I_s and I_{amp} is equal, I_s can be written in terms of I_{amp} by adding a scaling factor k:

$$I_s = k * I_{amp} \tag{4.2}$$

with $k \geq 1$, because the polarization measured in I_{amp} has to be part of the specular light I_s by our assumptions.

The decomposition is then simplified to finding the scalar k for every pixel. We call the resulting values for k the amplitude weights, and the image of all these values the amplitude weight map. Finding k is similar to finding p in Nayar's algorithm, which is used to calculate $I_d = I_{min} - p * k = I_{min} - p * (I_{max} - I_{min})$. k and p are closely related, as $p = (k-1)/2$.

We use these amplitude weights rather than Fresnel ratios, the standard in previous works, due to the non-linear nature of the Fresnel ratios. In extreme circumstances, the Fresnel ratio could be infinite when the incoming light is fully polarized, while the amplitude weight k will always be a finite number. Diffusion of amplitude weights will give better results in the cases where the Fresnel ratio of one pixel is close to infinity while the surrounding pixels have Fresnel ratios that are orders of magnitude smaller, as using amplitude weights guarantees that the propagation of values does not have to handle this edge case of infinite ratios. Amplitude weights also have better total value conservation properties than Fresnel ratios because of the non-linearity of the latter. Even though we do not use the Fresnel ratios, they can be directly calculated from

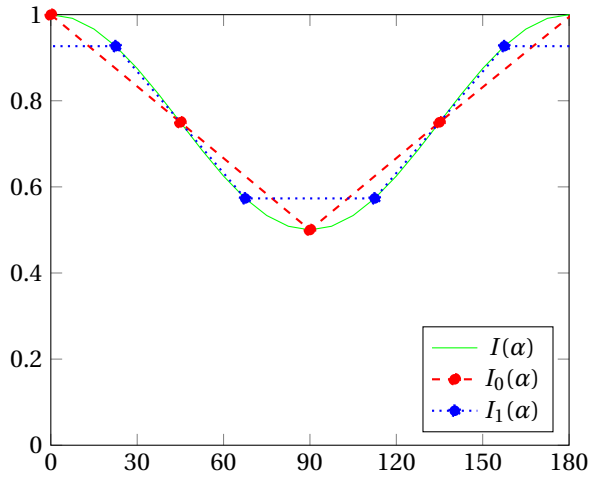

Figure 4.3: Measuring the polarization cosine $I(\alpha)$ with four samples. $I_0(\alpha)$ and $I_1(\alpha)$ show two possible sets of discrete measurements with 45° intervals for α. Our measurements use the points indicated by $I_0(\alpha)$ to find $I(\alpha)$.

Figure 4.4: I_{mean} (left) and I_{amp} (right) for sample 'zoom'. The blue tiles on the floor are strongly polarized due to the angle of the sun, while the foliage only has polarized highlights.

only the amplitude weights:

$$\begin{aligned} F &= \frac{I_s + I_{amp}}{I_s - I_{amp}} \\ &= \frac{k * I_{amp} + I_{amp}}{k * I_{amp} - I_{amp}} \\ &= \frac{(k+1) * I_{amp}}{(k-1) * I_{amp}} \\ &= \frac{k+1}{k-1} \end{aligned} \tag{4.3}$$

4.2. Measuring Polarization

The polarization cosine in a scene is measured using Wolff's (L. Wolff, 1990) setup with four orientations of a linear polarizing filter in 45° increments (see Figure 4.3). The four RGB images captured by a camera are merged into 4-dimensional data $I(\alpha, x, y, c)$. The first dimension is the filter orientation, the second and third the spatial coordinates, and the fourth the color band. At pixel $p = (x_p, y_p)$ and color band c, $I(\alpha, x_p, y_p, c)$ is a polarization cosine. The data captured corresponds to $I(0°, x, y, c)$, $I(45°, x, y, c)$, $I(90°, x, y, c)$ and $I(135°, x, y, c)$.

Because the signal is a cosine wave, previous experiments in L. Wolff, 1990 and Nayar et al., 1993 took a number of pixel intensity measurements with a rotating linearly polarized filter and then fit a cosine onto

the measurements using a mean squared error solver. We instead use a FFT-4 function on the input data, because it can be parallelized in hardware for a rapid transformation towards frequency space. The FFT-4 gives complex Fourier coefficients X_0 to X_3. Since our input consists of real numbers, X_3 is irrelevant, and X_2 can be ignored as any overtone of the cosine wave is noise. X_0 and X_1 are the two coefficients of interest. X_0 contains the discrete offset of the cosine wave I_{mean}, which can also be calculated as the average of the four measurements for $I(\alpha)$:

$$I_{mean} = \frac{I_{unpol} + I_{pol}}{2} = |X_0| \tag{4.4}$$

X_1 contains the amplitude and phase for the cosine term in the polarization cosine:

$$I_{amp} = |X_1| \tag{4.5}$$

$$I_{pol} = 2 * I_{amp} \tag{4.6}$$

$$\beta = \angle(X_1) \tag{4.7}$$

Calculations with I_{mean} and I_{amp} are simpler than the same calculations with I_{unpol} and I_{pol} without losing any information, thus we use the former. The phase β is discarded, but could be used to estimate the local surface orientation.

The peak value of the polarization cosine is I_{max} and the minimum is I_{min}. Rarely, if I_{max} or I_{min} is approximately equal to one of the measured polarization states, it is possible to read them directly and calculate the amplitude of I_{pol}. Guaranteeing that this happens requires infinite measurements, rather than four. It is more reliable to calculate them directly through the previously found I_{mean} and I_{amp}:

$$\begin{aligned} I_{max} &= I_{mean} + I_{amp} \\ &= \frac{I_{unpol} + I_{pol}}{2} + \frac{I_{pol}}{2} \\ &= \frac{I_{unpol}}{2} + I_{pol} \end{aligned} \tag{4.8}$$

$$\begin{aligned} I_{min} &= I_{mean} - I_{amp} \\ &= \frac{I_{unpol} + I_{pol}}{2} - \frac{I_{pol}}{2} \\ &= \frac{I_{unpol}}{2} \end{aligned} \tag{4.9}$$

Only three out of the previously mentioned four measurements at 45° intervals are strictly necessary to reconstruct any arbitrary cosine wave with a constant offset created by polarization. If three measurements are taken instead of four and, for example, only $I(0°)$, $I(45°)$, $I(90°)$ are known, $I(135°)$ can be directly calculated:

$$I(135°) = (I(0°) + I(90°)) - I(45°) \tag{4.10}$$

Afterwards, the four measurements can be processed by the FFT-4.

We found one edge case where sensor noise and discretization of all four measurements gave $I_{amp} > I_{mean}$, and therefore $I_{min} < 0$. An example that can cause this edge case is $I(0°) = 1$, $I(45°) = 1$, $I(90°) = 0$, $I(135°) = 0$. In a real-world scene this should never be a valid measurement, as this would imply that $I(\alpha) < 0$ for $90° < \alpha < 135°$, while a measurement of negative pixel intensity cannot occur due to negative light intensities not existing. However, the algorithm needs to handle this case, thus it reduces I_{amp} when $I_{amp} > I_{mean}$:

$$I'_{amp} = \min(I_{mean}, I_{amp}) \tag{4.11}$$

Dynamic Range Extension Well-exposed camera samples might have highlights that are clipped at one or more of the *RGB* colour channels. If the camera does not have the dynamic range to retrieve unclipped highlights in each of the filter orientations, it is possible to partially restore this data using the other orientations. At any color band of any pixel it is known that all measurements should lie on the polarization cosine. I_{mean} could be calculated using only the half of the measurements with the lowest values plus one. If there would be 64 measurements, only the lowest 33 would be required. In our case with four measurements, only the three lowest measurements are necessary. Once I_{mean} is known, the higher half of the measurements can be restored by mirroring the lower half.

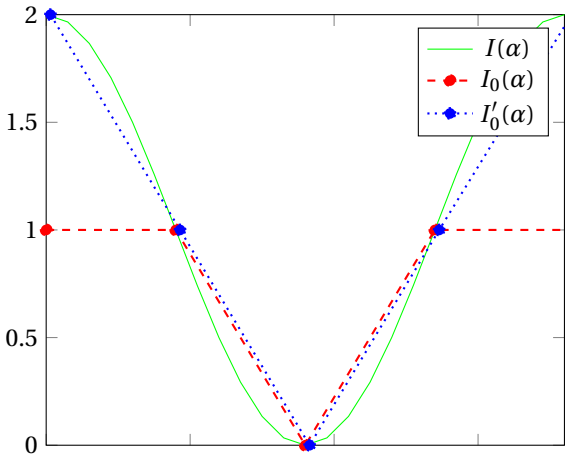

Figure 4.5: When $I(\alpha)$ is measured using the four sampling points of $I_0(\alpha)$, the first sample is clipped at 1.0. Using the the other three samples which are not clipped, the single clipped sample can be restored to 2.0.

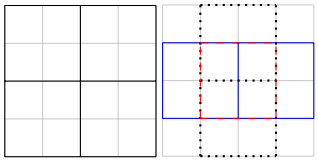

Figure 4.6: Both I_{mean} and I_{amp} are divided in N-by-N tiles with overlap. The black squares, blue squares, dotted squares and red square are all valid 2-by-2 tiles in this 4-by-4 example image. $N = 16$ in our implementation.

To illustrate of how this is added to an implementation with four measurements, we take the example of the peak amplitude being at $I(0°) = 2$. The ground truth is $[2;1;0;1]$, but the clipped measurements will give $[1;1;0;1]$ due to the upper bound at 1.0. The clipped sample can be restored by calculating I_{mean} from $I(45°)$ and $I(135°)$; $I_{mean} = (1+1)/2 = 1$, and then subtracting $I(90°)$ from twice the mean; $I(0°) = 2 * I_{mean} - I(90°) = 2 * 1 - 0 = 2$. Using this technique, the dynamic range can be increased by up to a factor of two in ideal conditions with maximum polarization (see Figure 4.5).

This process is automated by calculating each measurement $I'(\alpha)$ as a function of the other three measurements. The final values are then the maximum of the original values and the recalculated values: $I''(\alpha) = \max(I(\alpha), I'(\alpha))$.

4.3. Tile-Based Estimate

After finding I_{mean} and I_{amp} in Section 4.2, k can be estimated using the local correlation between the I_{mean} and I_{amp} across a patch of pixels. The underlying assumption is that any correlation between the two is caused by specularities, and choosing k to maximally reduce the correlation results in a perfect decomposition between I_d and I_s. Due to image noise, calculating the local correlation for one pixel p requires a large window around that pixel for an accurate estimate. This is computationally expensive, and is therefore optimized by only calculating the correlation for a subset of pixels. From this subset of pixels the remaining pixels are estimated through interpolation.

Rather than defining the subset of pixels for which the correlation is calculated, we define the windows around the pixels as tiles, and directly use these tiles for our calculations. I_{mean} is divided into N-by-N tiles, where N is a small constant, chosen to be 16 in our implementation due to the input image sizes in our experiments where this is approximately 1% of the image size in one dimension. For higher precision, additional overlapping N-by-N tiles are added in between the first set of tiles. For example, if the image is 4-by-4 and N is 2, every possible 2-by-2 square of the image becomes one tile, for a total of $3 * 3 = 9$ tiles that are of size 2-by-2 (see Figure 4.6). The process is then repeated for I_{amp}. A tile with index (a, b) will be addressed as $T(a, b, x, y, c)$, where (x, y) are the spatial coordinates inside the tile, and c is the color band.

Each tile T from both the mean and amplitude image is average-compensated to T':

$$T_{avg}(a,b,c) = \frac{\sum_{x=0}^{N-1}\sum_{y=0}^{N-1} T(a,b,x,y,c)}{N^2} \tag{4.12}$$

$$T'(a,b,x,y,c) = T(a,b,x,y,c) - T_{avg}(a,b,c) \tag{4.13}$$

A cosine window w is then applied to the data in each tile in both the horizontal and vertical dimension. This reduces the impact of data from the edges of tiles and increases the weight of the center of tiles.

$$w(x) = \frac{1}{2} - \frac{1}{2}cos(2\pi\frac{x+\frac{1}{2}}{N}) \tag{4.14}$$

$$T''(a,b,x,y,c) = w(x) * w(y) * T'(a,b,x,y,c) \tag{4.15}$$

The last three arguments (x,y,c) are discarded and the entire tile $T''(a,b)$ is used as a large $N*N*C$-sized vector. The vectors $T''_{mean}(a,b)$ and $T''_{amp}(a,b)$ can be compared using any correlation analysis. For our implementation, a basic vector-projection was used. One assumption made for this tile-based estimate by choosing a basic vector-projection is that a mean tile T_{mean} can be directly decomposed into the corresponding amplitude tile weighted by a constant, which becomes the specular component $T_s = k * T_{amp}$, and an orthogonal tile T_d that will make up the diffuse component. Due to their orthogonality, this gives $T_d \cdot T_s = 0$ where $A \cdot B = A_0B_0 + A_1B_1 + ... + A_nB_n$ denotes the dot product between two vectors A and B. k can be found by projecting the mean tile onto the amplitude tile, to find the correlation:

$$\begin{aligned} k_{tile}(a,b) &= \mathbf{proj}(T''_{mean}(a,b), T''_{amp}(a,b)) \\ &= \frac{T''_{mean}(a,b) \cdot T''_{amp}(a,b)}{T''_{amp}(a,b) \cdot T''_{amp}(a,b)} \end{aligned} \tag{4.16}$$

where $\mathbf{proj}(A,B)$ is the scaling factor for vector B that makes up the projection of vector A onto vector B. This means that $(A - B * \mathbf{proj}(A,B)) \cdot B = 0$.

The coefficients $k_{tile}(a,b)$ found are the initial estimates for the amplitude weights. These coefficients are not directly available for each pixel $p = (x_p, y_p)$ in the image, so they are interpolated between tile centers. Due to the downscaling in Section 4.6, linear interpolation should suffice. In our implementation, we use cosine interpolation to find $k(p)$ for each pixel p.

The assumption that I_d and I_s are uncorrelated and therefore have orthogonal tile vectors often does not hold. Predominantly on edges and gradient textures the correlation can be positive, causing the tiles on these patches of objects to result in a higher amplitude weight k than intended. Nevertheless, the tiles where this assumption does hold give an amplitude weight estimation that is correct without any further calculations, which is why it is used as a starting point for the algorithm. An alternative correlation analysis could look at object patches using the functional bilateral filter for a more accurate first estimate for k. However, this would also be computationally more expensive, while the basic correlation analysis already provides a good starting point.

4.4. Functional Bilateral Filter

Nayar et al. (Nayar et al., 1993) iteratively use an information propagation step to improve their weight map until convergence. For each pixel that has not converged, the best diffuse component estimations of neighbouring pixels are used to improve the diffusion component estimation of the pixel. This has the two drawbacks that the pixels might find a local optimum that is unequal to the global optimum, and the information propagation being slow across the image. To replace their information transfer step we introduce a 'functional bilateral filter', which does not stop iterating if it converges at a local optimum and does not make any hard decisions for propagation area boundaries.

The original bilateral filter (Tomasi and Manduchi, 1998) only filters one input image I by comparing pixels from that input image I (Equation 2.10). The cross/joint bilateral filter (Eisemann and Durand, 2004; Petschnigg et al., 2004) extends this by having a separate input image K:

$$J_q = \frac{1}{k(q)} \sum_{p \in \Omega} f(p-q) * g(I_p - I_q) * K_p \tag{4.17}$$

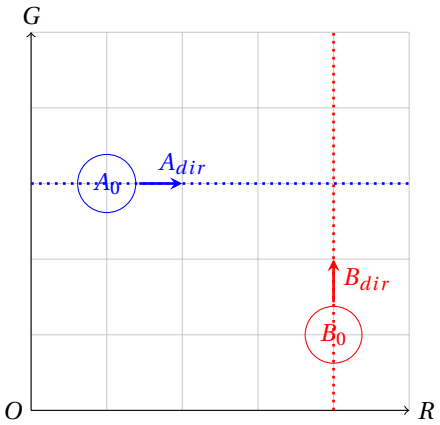

Figure 4.7: *RG* color space intersection of two arbitrary non-parallel lines A and B can be solved with a system of two equations, using a 2-by-2 matrix with A_{dir} and $-B_{dir}$ as the column vectors. In *RGB* color space a 3-by-3 matrix is needed to solve the system with three equations, because an additional third line C is needed to ensure the lines always intersect.

The 'functional bilateral filter' is a generalized variant of the cross/joint bilateral filter, and uses a separate input image $\kappa(p,q)$ for each pixel shift $p-q$, which may be calculated directly during execution of the algorithm:

$$J_q = \frac{1}{k(q)} \sum_{p \in \Omega} f(p-q) * g(I_p - I_q) * \kappa(p,q) \tag{4.18}$$

The individual weights between pixels $f(p-q) * g(I_p - I_q)$ are unchanged from the original bilateral filter. This results in $k(q)$ being unchanged as well. The difference between the cross/joint bilateral filter and the proposed filter is the dependency of the input image on the spatial shift vector. The input image $\kappa(p,q)$ can be different for each spatial shift $p-q$, and can be implemented as a function that is ran in parallel for each pixel being filtered.

Our implementation also has an additional change compared to the original bilateral filter, where the weight $g(I_p - I_q)$ is changed to zero when $\kappa(p,q)$ gives an invalid value. Valid values are characterized by being inside a certain range, such as being non-infinite or being between 0 and 1. In our algorithm, values between the lower and upper bound k_{min} and k_{max} of the amplitude weight are valid. The previous definition can be updated as follows:

$$J_q = \frac{1}{k(q)} \sum_{p \in \Omega} f(p-q) * g'(p,q) * \kappa(p,q) \tag{4.19}$$

$$g'(p,q) = \begin{cases} g(I_p - I_q) & \text{if valid}(\kappa(p,q)) \\ 0 & \text{otherwise} \end{cases} \tag{4.20}$$

The functional bilateral filter can be used to calculate a weighted sum of any local function on pixel pairs with the edge-preserving feature of the bilateral filter. It is a generalization of the normal and cross/joint bilateral filter, as defining $\kappa(p,q) = I_p$ makes it equal to the normal bilateral filter, and defining $\kappa(p,q) = K_p$ makes it equal to the cross/joint bilateral filter. Our implementation also uses the optimization where Ω is replaced by $N(s)$, to prevent calculations on values with near-zero spatial weights.

4.5. Line Intersection

The diffuse and specular components are estimated between two neighbouring pixels using a plane-intersection function by Nayar et al. (Nayar et al., 1993). However, this operation is not robust to camera sensor noise, and they introduce an angular threshold to reduce the impact of noisy values. This works when there is only impulse noise for a small amount of pixels, but cannot compensate for strong sensor noise throughout the image. Our method replaces their plane-intersection function with a line-intersection function, which is less susceptible to sensor noise.

The intersection of two lines with two or more dimensions can be defined as a linear vector operation (see Figure 4.7. For a line A, any point A_p on that line can be described by $A_p = A_0 + a * A_{dir}$ with exactly one

value for a. A_0 can be chosen as any point on the line, but changing it also changes the value of a for all A_p. When two lines $A_p = A_0 + a * A_{dir}$ and $B_p = B_0 + b * B_{dir}$ intersect each other, their coordinates are equal:

$$A_0 + a * A_{dir} = B_0 + b * B_{dir} \tag{4.21}$$

This gives an equation that can be solved uniquely for (a, b), where either a or b can be used to calculate the coordinates of the intersection point. Two non-parallel lines are only guaranteed to intersect if there are two dimensions. In the case of three dimensions, the two lines are unlikely to intersect due to the extra dimension of freedom. With four or more dimensions, the likelihood of an intersection point becomes increasingly lower. Rather than searching for an intersection point, we instead calculate the closest points between the two lines, which is the pair of points on the lines where there is only an orthogonal component to both lines between the points. This orthogonal component is the cross product, denoted with the × symbol, of the line directions of A and B, A_{dir} and B_{dir} respectively:

$$C_{dir} = A_{dir} \times B_{dir} \tag{4.22}$$

The closest point between A and B can then be described by:

$$A_0 + a * A_{dir} = B_0 + b * B_{dir} + c * C_{dir} \tag{4.23}$$

which can be rewritten in vector-matrix form:

$$\begin{bmatrix} A_{dir} & -B_{dir} & -C_{dir} \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = [-A_0 + B_0] \tag{4.24}$$

This is a linear system that can be solved for (a, b, c) using an existing solver. Our implementation uses Cramer's Rule to find a, b and c in parallel without requiring any conditionals. The point on line A can then be found at $A_0 + a * A_{dir}$, and the point on line B is at $B_0 + b * B_{dir}$. When lines A and B are parallel, there is no solution. This case is handled by the valid() check in the functional bilateral filter.

4.6. Multi-Scale Optimization Iterations

The values k calculated by the tile-based estimate are a starting point, and are downscaled to a low resolution amplitude weight map. A scaling level i has amplitude weights k_i. Level 0 and map k_0 are the highest level with highest resolution map k, counting down into negative integers to lower resolution maps. k_i is downscaled to k_{i-1} by applying bilinear downscaling. k_0 is downscaled to k_{-1}, which is downscaled to k_{-2}, etc., until the lowest level is calculated. The total amount of levels depends on image size and runtime constraints. Starting from the lowest level, the weight map k_i is improved by applying local optimization criteria, explained in this section. Edge-stopping is implemented through the functional bilateral, explained in Section 4.4. After multiple iterations of applying the optimization criteria, the weight map k_i is upscaled to k_{i+1}. This process is repeated for each subsequent level, improving the weight map at level i and then upscaling to level $i+1$. Once k_0 has been improved following the local optimization criteria, the process stops, and the final weight map k can be read from k_0.

Amplitude Weight Bounds Before iterating to the locally optimal amplitude weights, it is useful to set an upper and lower bound on the amplitude weights to ensure that one noisy pixel cannot drastically influence the results. The most conservative lower bound on an amplitude weight k is $k_{min} = 1$, which means that the specularity must be least equal to the polarized amplitude: $I_s \geq I_{amp}$, $I_s = k * I_{amp}$, therefore $k \geq 1$. The lower bound value 1 can be increased by calculating local correlation and taking the maximum of the local correlation and 1. This local correlation is calculated using the functional bilateral filter, using the following function κ_{corr}:

$$\kappa_{corr}(p, q) = \mathbf{proj}(I_{mean}(p) - I_{mean}(q), I_{amp}(p) - I_{amp}(q)) \tag{4.25}$$

using the same $\mathbf{proj}(A, B)$ function as Section 4.3.

$k = \kappa_{corr}(p, q)$ minimizes the Euclidean distance between $I_{mean} - k * I_{amp}$ for pixels p and q. This would be a direct solution to the intrinsic image decomposition, but usually gives a lower k than the ground truth due to noise in both I_{mean} and I_{amp}. Therefore the value found is only used as an input to find the minimum amplitude weight k_{min}.

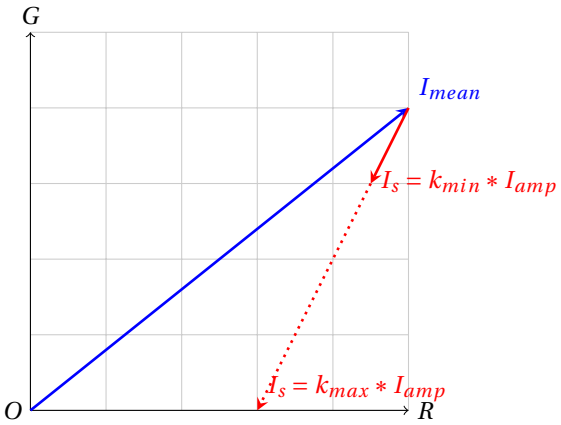

Figure 4.8: RG color space intersection between OR and $I_s = k * I_{amp}$. The value for I_s is somewhere on the dotted line segment. The final value for k_{max} is determined by projecting separately on all three color planes in RGB color space, and taking the lowest k_{max} value of the three.

The function **corr** is then defined as the functional bilateral using $\kappa = \kappa_{corr}$, with the weights being determined by I_{mean}. The new lower bound is the maximum of the previous lower bound, which was 1, and the local correlation:

$$k_{min}(x,y) = \max(1, \textbf{corr}(x,y)) \tag{4.26}$$

This only holds if **corr**(x,y) is defined such that the local correlation of I_{amp} with I_{mean} is only calculated on one region with the same material characteristics on an object, and not across objects or across surfaces of different materials. The edge-stopping performance of the functional bilateral filter used is crucial for the accuracy of Equation 4.26. If the correlation becomes higher than ground truth due to the inclusion of edge gradient steps, the lower bound becomes too high, resulting in artifacts around edges. These artifacts cannot be removed by the local optimization step because it respects the lower bound, which is itself causing the artifacts.

For the maximum amplitude weight, the theory from Nayar et al., 1993 is used, which states that the maximum limit is set by the point where one or more color bands of the diffuse image would become negative (see Figure 4.8). This is equal to a projection onto either the RG, RB or GB color plane. The maximum amplitude weight m^c for color band c is the scale between I_{mean} and I_{amp} in that color band:

$$\begin{aligned} m^c &= I^c_{mean} / I^c_{amp} \\ &\text{with } c \in \{R,G,B\} \end{aligned} \tag{4.27}$$

$$k_{max} = min(m^R, m^G, m^B) \tag{4.28}$$

Both bounds k_{min} and k_{max} are pre-calculated for the local optimization step.

Local Optimization At a level i with weight map k_i, the local optimization has a wide information propagation step based on the Fresnel ratio assumption and a local diffuse estimation step based on the diffuse color vector direction constancy assumption.

The wide information propagation step runs a cross bilateral filter with the current estimate k as the input image and I_d as the weight image, to blur towards edges. This is based on the assumption that the Fresnel ratio $I_{s,perp}/I_{s,par}$, and therefore the amplitude weight k is locally similar. Wolff (L. Wolff, 1989) has shown that this gives good results for dielectric and metallic materials.

The local diffuse estimation step brings $I_d = I - I_s = I - k * I_{amp}$ of nearby pixels together. This is based on the assumption that one object has one diffuse color D, which is scaled by a varying multiplier m based on the amount of irradiance from white light sources and the local surface direction changes across the object. Some objects are inhomogeneous, breaking this assumption. However, these objects can be processed as if the homogeneous object patches are objects themselves. $D = I_d/m$ is diffused using the line intersection algorithm from Section 4.5 as the κ function for the functional bilateral filter. Define $\begin{bmatrix} a & b & c \end{bmatrix}^T = \text{li}(A_0, A_{dir}, B_0, B_{dir})$ with **li** a line intersection function implementing the theory from Section 4.5, (a,b) the coefficients to find

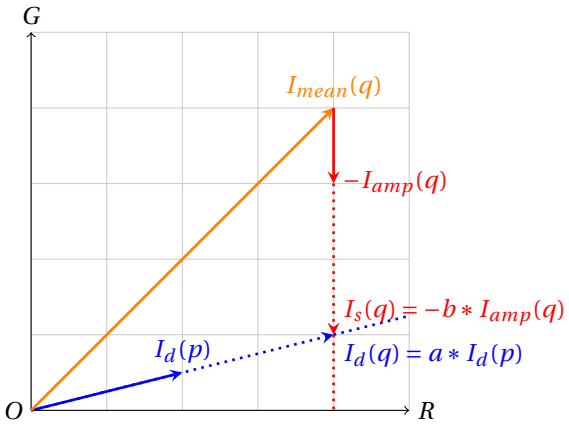

Figure 4.9: *RG* color space estimation of $I_d(q)$ given $I_d(p)$, by using the matrix-based line intersection from Section 4.5 to find the two coefficients (a, b). The important coefficient is b, which becomes the new amplitude weight estimate $k(q)$ for pixel q.

points P and Q on lines A and B (see Figure 4.9) and c the length of the cross product between the two lines if they do not intersect. Then κ is implemented as:

$$\kappa_{li}(p,q) = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \cdot \mathbf{li}(O, I_d(p), I_{mean}(q), I_{amp}(q)) \tag{4.29}$$

The three-column matrix is inserted before the function because only b is necessary to find the next value for k, as the other two coefficients provide information about the length of the diffuse vector and the size of the cross product between the two lines, which are both not required.

For each neighbouring pixel p, an RGB line from the origin O to p's current I_d estimate is intersected with the RGB line that q's I_d has to lie on, which is the line extending from $I_{mean}(q)$ extending in the direction of the polarization amplitude $I_{amp}(q)$. It has to lie on this line because of the assumption that $I_s = k * I_{amp}$, and therefore $I_d = I_{mean} - k * I_{amp}$ which is precisely this line. In a noise-free environment the two lines should always intersect, but noise can cause a slight offset, which is why the line intersection function li is implemented according to Section 4.5 with an additional cross product to account for the offset.

These two steps are repeatedly intertwined as an edge preserving low-pass filter in two separate dimensions. The amount of iterations is chosen to be $2 * (1 - i)$ where i is the level in the downscaling pyramid. Both steps respect the bounds k_{min} and k_{max}, and will discard values in the functional and cross bilateral that fall outside the bounds. The points with a high absolute polarization I_{pol} will serve as a local upper bound on k, as k_{max} at those points will be close to 1, causing the diffusion of k to bring all local values closer to 1. The points with a low absolute polarization I_{pol} will be an anchor for the diffuse component, as even with a high k the estimate for I_d will remain stable.

There is one additional cross bilateral filter applied as a post-processing step after the repeated execution of the wide information propagation step and the diffuse intersection step on a pyramid level. k is once again used as the input image, but I_{mean} instead of I_d is used as the weight image, to propagate information across regions with a similar total radiance. k on any local patch that visually looks like one object is blurred into one object by this step. It also ignores the bound k_{min}, to reduce the minor artifacts around edges where the bilateral filter for k_{min} assigned a higher weight across an edge than intended, causing a higher k_{min} than the ground truth. This step is ran twice on each pyramid level.

5

Implementation Details

The algorithm was implemented in MATLAB® 2022, with the source code publicly available at https://github.com/amirzaidi/Polarization.

5.1. Image Scaling

In Sections 4.3 and 4.6, the input image is assumed to be divisible by a large power of two. When this is not the case, the input image is first extended until it is divisible by this factor:

$$Mirr(x) = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{otherwise} \end{cases} \tag{5.1}$$

$$I'(x,y) = I(Mirr(x), Mirr(y)) \tag{5.2}$$

If $I(x,y)$ denotes the image intensity at pixel coordinates (x,y), $I'(x,y)$ is calculated for an extended range of coordinates compared to the input image. The padding is chosen to be a symmetric edge-mirror padding due to it looking more natural than image repetition or edge repetition, and it having a value distribution that resembles the region along the edge rather than only the edge itself. However, if the image would be cropped after the execution of the algorithm, an edge repetition padding should not significantly influence the result. If the input image is 14-by-14, $I(0,0)$ and $I(13,13)$ would be the original corners. The extended corners are $I'(-1,-1)$, $I'(14,14)$. This extended image is generated for both I_{mean} and I_{amp}, which are subsequently updated to a zero-indexed shift of the 16-by-16 I'_{mean} and I'_{amp}, ranging from $I'(0,0)$ to $I'(15,15)$.

The downscaling operation is preceded with a Gaussian blur $G(\sigma)$ to prevent the formation of moiré patterns, while the upscaling operation is succeeded by the same Gaussian blur to reduce the blockiness of the result:

$$G(\sigma) = G([-N \text{ to } N], \sigma)^T \circledast G([-N \text{ to } N], \sigma) \circledast I \tag{5.3}$$

$$G(x,\sigma) = \frac{1}{2\pi\sigma^2} * e^{-x^2/(2\sigma^2)} \tag{5.4}$$

with N chosen to be approximately $3 * \sigma$. The Gaussian blur suppresses the high-frequency components, which reduces aliasing from the bilinear scaling.

The blur is applied in two steps using Equation 5.3 with $\sigma = 1.36$ in first the horizontal, and then the vertical direction. This value for σ was chosen as a trade-off between the retention of frequencies lower than a quarter of the sampling rate, while attenuating the frequencies higher than a quarter of the sampling rate (see Figure 5.1). Sampling is performed by averaging tiles of 2-by-2 when downscaling, and duplicating each pixel into a 2-by-2 tile when upscaling.

One possible issue with this approach is that checkerboard pattern textures with high frequency lose their pixel variance entirely, becoming one constant color. We did not notice this during our testing, as such a pattern is not common in real-world scenes, but could exist due to aliasing of highly detailed textures.

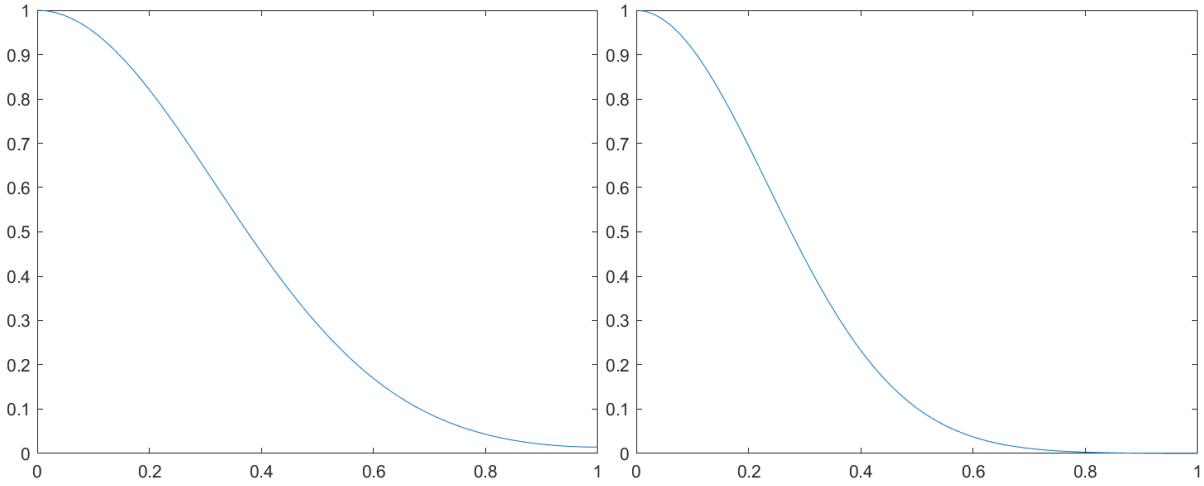

Figure 5.1: Gaussian blur frequency response for $\sigma = 1.00$ (left) and $\sigma = 1.36$ (right). X-axis is the relative frequency, with 1.0 being half the sampling frequency of the image. Y-axis is the relative amplitude after applying the gaussian blur. A higher σ value more strongly suppresses high frequencies at the cost of losing the middle frequencies.

5.2. Parameters

There are multiple parameters that were empirically chosen. First, there is the number of pyramid levels in Section 4.6. We chose a total of 4 levels for input images of size approximately 3000-by-2000, because choosing a higher number of level did not improve results. Second, we use $N = 16$ for our tile-based estimate in Section 4.3 due to the input images being of the aforementioned size. If the size of the images would be doubled in both dimensions, N should also be doubled to encapsulate the same relative object patch size. Third, four separate calls to the functional bilateral filter are made, which all have their own spatial sigma σ_f and tonal sigma σ_g values:

1. The minimum bound in Section 4.6 is calculated once in each pyramid level with the same σ values on each level. $\sigma_f = 0.33$ and $\sigma_g = 0.10$ were chosen to keep the calculation very localized. This prevents random correlations between pixels far apart from setting a high k_{min} and pushing k higher than intended.
2. The line intersection in Section 4.6 is also calculated once in each pyramid level with the same σ values. $\sigma_f = 0.66$ and $\sigma_g = 0.02$ were chosen to set a strong tonal boundary on which pixels are allowed to recalculate k between them. If σ_g is significantly increased, edge boundaries are not respected creating extreme outliers of k along edges. σ_f might be increased, but is kept low for performance reasons.
3. For the cross bilateral filter intertwined with the line intersection in Section 4.6 we used $\sigma_f = 2.00$ and $\sigma_g = 0.05$ to allow for rapid wide-area diffusion of k without significantly crossing edge boundaries.
4. For the final cross bilateral filter using I_{mean} rather than the estimate for I_d as its weight input image, we used $\sigma_f = 2.00$ and $\sigma_f = 0.01$, because this is intended as a blur on color patches in I_{mean} which is a ground truth measurement of what the human eye would see, unlike color patches in I_d which is only an estimate of the diffuse component. If there is even a small tonal difference in I_{mean}, this bilateral filter should not apply any diffusion. This step is ran twice at the end of each pyramid level in our implementation.

These parameters are input image size independent, because the same algorithmic steps are applied on each level of the Gaussian pyramid. If the image resolution is doubled, the optimization step at previously the highest level map k_0 is shifted once to map k_{-1}, while k_0 becomes a new optimization step at the doubled image resolution, to improve the highest frequencies of the result.

6

Evaluation

Before providing experimental results, we provide an overview of the analysis criteria used in the works by Nayar et al. (Nayar et al., 1993). We then apply these criteria to samples collected in both an outdoors and indoors environment.

6.1. Metrics

Nayar et al. analysed the performance of their algorithm through its effectiveness in removing specular highlights and retaining textures. They note there can be artifacts due to multiple factors, such as the polarization of diffuse light at edges of objects and the clipping of highlight values. Our analysis will first focus on the quality of the specular highlight removal and how well the algorithm handles the artifacts, followed by a quantitative analysis of the runtime performance for various image sizes.

6.2. Sample Collection

The two sample sets 'zoom' and 'inside' were captured with a Canon EOS 4D Mark II in CR2 RAW mode. The sample set 'potdng' was captured by a OnePlus 7 Pro in DNG RAW mode. Each sample set consists of four images, each captured with a 45° difference in polarization filter angle. CR2 images were converted to uncompressed 16-bit sRGB TIFF files using Darktable with all post-processing disabled. DNG images were converted to sRGB JPEG files using DNG Processor.

6.3. Qualitative Analysis

Both outdoor and indoor images were tested to analyse the overall quality of the decomposition in challenging circumstances.

Outdoor Images Outside, the direct reflection of the sun on reflective dielectric materials creates the ideal testing conditions for highlight removal. In the first outdoor sample 'zoom' (see Figure 6.1), there are small highlights on the rocks and leaves. There are two lamp posts with a curved top, creating small regions that show that correlation-based decomposition can result in a smooth diffuse image.

In the second outdoor sample 'potdng' (see Figure 6.2), there is a large rectangular highlight on the side of the raised tile in the middle of the image. This highlight is perfectly decomposed into its diffuse and specular components, revealing the blue diffuse color of the tile. One apparent problem is the movement of the leaves between measurements due to wind, causing edges of leaves on the palm tree in the back on the right and the withered plant on the left to be falsely detected as areas with high polarization. Solutions to this problem are discussed in Section 8.2. Another problem is the lack of diffusion through the sky, creating a bright halo around the branches in the specular image. This could be solved by having multiple iterations of the entire pyramid, or by running more iterations of the individual steps inside the pyramid. However, this increases the runtime significantly, and is therefore only an option when quality is preferred over speed.

Indoor Images Indoors during daytime, the amount of specular light is lower due to the lighting being mostly scattered bounces of outside light sources, which prevents objects from having a significant incoming

Figure 6.1: Diffuse-specular decomposition of sample set 'zoom'. The top two images show I_{mean} (left) and I_{amp} (right). The next three rows have the tile-based estimate on the left, and the final decomposition on the right. Those rows show I_d, I_s and k in order from top to bottom. k is rescaled from $[0,4]$ into the range $[0,1]$ for visibility.

Figure 6.2: Diffuse-specular decomposition of sample set 'potdng'. The top two images show I_{mean} (left) and I_{amp} (right). The next three rows have the tile-based estimate on the left, and the final decomposition on the right. Those rows show I_d, I_s and k in order from top to bottom. k is rescaled from $[0,4]$ into the range $[0,1]$ for visibility. Sample set taken with a mobile phone rather than a professional camera.

Sample	Size	Run 1	Run 2	Run 3
Potdng	1976x1476	22.470s	21.748s	22.022s
Zoom	1861x2801	36.347s	35.940s	36.156s
Inside	1861x2801	35.867s	35.909s	35.980s

Table 6.1: Runtime of our implementation.

light direction. However, in our indoor sample 'inside' (see Figure 6.3), we found that mirror-like reflections are a special case of specularities, and are therefore separated by correlation-based decomposition. The specular image reveals the entirety of the reflection, while the diffuse image has the reflection removed except for some remaining noise.

The pot in the middle still has one highlight in the top right of the diffuse image, caused by the curving geometry breaking the assumption of a constant Fresnel ratio across an object's surface. The tile-based estimate does properly remove the entire highlight, but the blurring of k across the highlight reduces the quality of the decomposition of the highlight.

On the left of the pot, there is also a blue remnant on the diffuse image after the removal of the highlight. This is caused by the quality of the polarized filter used to take the samples. The filter itself has an imperfect response curve for polarized light, most noticeably letting through more blue light than other wavelengths, creating a color imbalance on the polarization cosine. One possible solution to this problem is to assume all specular light is entirely white, and projecting I_s onto the white color vector. However, that would require all light sources to be white balanced, and would prevent color reflections from being decomposed, reducing the quality of the overall decomposition.

Finally, there is one brown square on the wall on the top left of the diffuse image. This is caused by clipping of some of the input images, resulting in an impossible to solve local balance between the diffuse color and the weight map. This could be solved by HDR bracketing to increase the dynamic range, or using a lower exposure on a high bit depth camera.

6.4. Quantitative Analysis

On an Intel Core i5-6600K and NVIDIA GeForce GTX 970, our implementation of the algorithm takes approximately 30 seconds for the full decomposition including all IO operations and UI handling (Table 6.1). The implementation vectorizes each calculation step such that there are no iterations over individual pixels. Additionally, Cramer's Rule is optimized to only calculate the necessary determinants once. Nevertheless, kernel-based parallelization over pixels should be significantly faster as there are less synchronization points and the pipeline can be optimized for the hardware. A native implementation in OpenGL, OpenCL or Vulkan could improve the runtime by multiple orders of magnitude. For example, one cross/joint bilateral filter on a 3000-by-2000 RGB image takes approximately 1 second in our implementation, while the same filter in OpenGL takes only 100 milliseconds. If all operations gain a similar speedup, the runtime could be reduced by a factor of 10.

Figure 6.3: Diffuse-specular decomposition of sample set 'inside'. The top two images show I_{mean} (left) and I_{amp} (right). The next three rows have the tile-based estimate on the left, and the final decomposition on the right. Those rows show I_d, I_s and k in order from top to bottom. k is rescaled from $[0,4]$ into the range $[0,1]$ for visibility.

7

Applications

The decomposed diffuse and specular images can be used for direct scene editing without prerequisite knowledge of the scene. Textures and highlights can be adjusted, and tonemapping algorithms have more information for HDR-to-SDR image compression.

7.1. Texture Modification

In RGB color space an object with a constant texture color can be represented as:

$$\forall p : (p \in object) \leftrightarrow T(p) = \begin{bmatrix} R & G & B \end{bmatrix}' \tag{7.1}$$

where T denotes the texture color at a given pixel. The diffuse color is:

$$I_d(p) = m * T(p) \tag{7.2}$$

where m is a multiplier determined by one or more white light sources' angle of incidence relative to the object. Modifying the diffuse color directly would remove m from the diffuse image. To prevent this loss of lighting information, the color vectors are normalized by their Euclidean length to separate m:

$$I_{d,norm}(p) = \frac{I_d(p)}{|I_d(p)|} \tag{7.3}$$

$$|I_{d,norm}(p)| = |k * T(p)| = k * |T(p)| \tag{7.4}$$

$$I_{d,norm}(p) = \frac{k * T(p)}{k * |T(p)|} = \frac{T(p)}{|T(p)|} \tag{7.5}$$

Equation 7.5 shows that if an entire object has the same texture color $T(p)$, it also has the same normalized diffuse color $I_{d,norm}(p)$. The normalized diffuse color can first be used for object classification using a L2 norm in RGB color space. Second, the normalized diffuse colors can be modified to another color $I'_{d,norm}(p)$. The shading information is contained in the denominator of the first formula: $|I_d(p)|$. Multiplying this value back in to restore the shading information gives a new diffuse color with accurate lighting:

$$I'_d(p) = I'_{d,norm}(p) * |I_d(p)| \tag{7.6}$$

where $I'_{d,norm}(p)$ is a forged texture to map onto the object. Finally, the new image can be constructed by adding the original specular component:

$$I'(p) = I'_d(p) + I_s(p) \tag{7.7}$$

This was tested on one flower pot in 'potdng' (see Figure 7.1). The classification based on the normalized diffuse color is imperfect, leaving a trail of pink pixels alongside the bottom left edge of the flower pot. Even so, the texture replacement looks realistic, and the specular component being added makes the right side of the edited pot look as shiny as the original. One apparent shortcoming is the lack of adjustment of the red glow on the green pot, caused by the pink diffuse light of the pink pot being emitted to and reflected by the green pot. The model used is too simple to account for such reflections between objects.

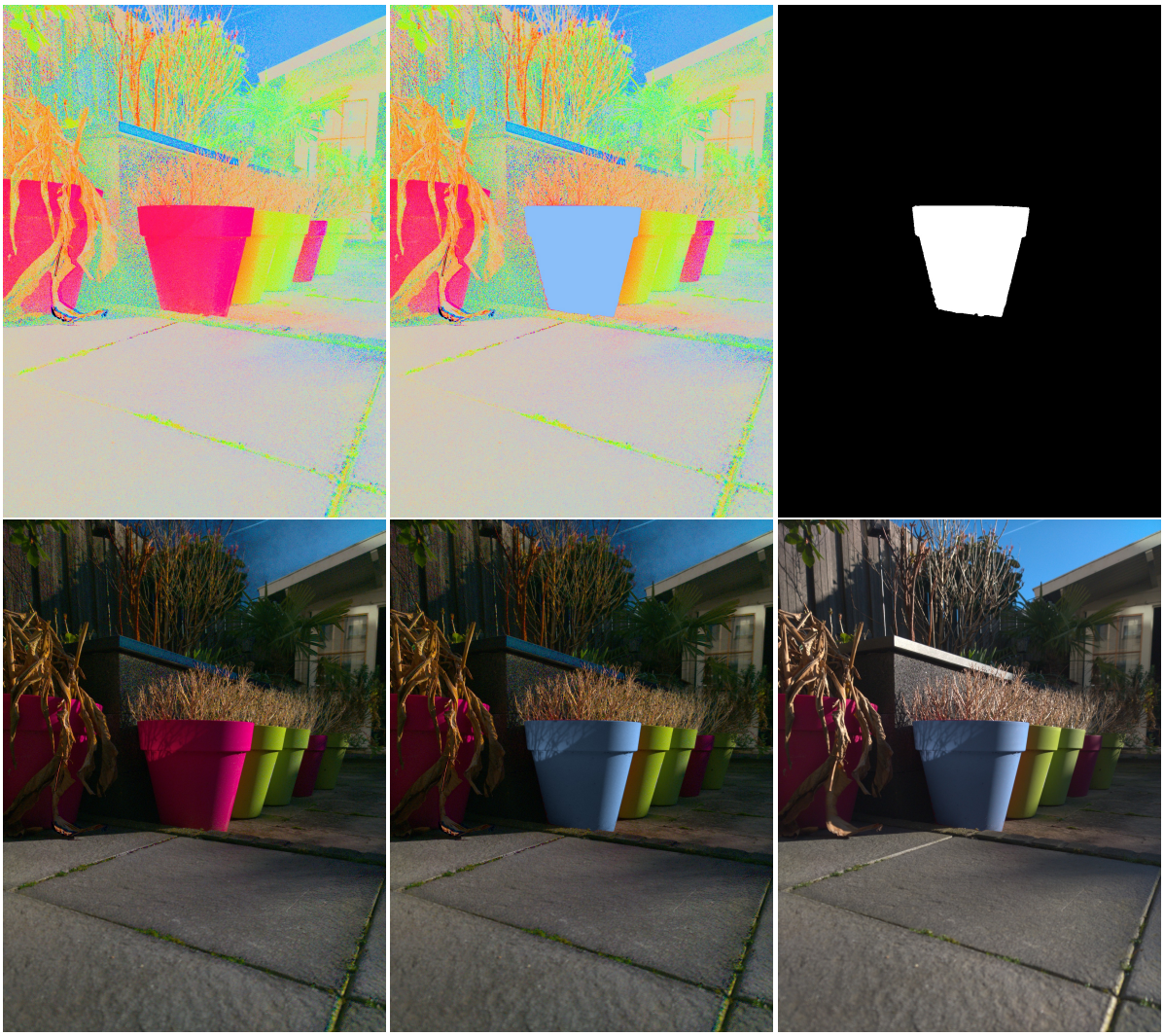

Figure 7.1: Replacing the normalized diffuse color of the centered pot in sample 'potdng'. Top left shows $I_{d,norm}$, top middle $I'_{d,norm}$, and top right the binary map to segment the centered pot using a color classifier. Bottom left shows I_d, bottom middle I'_d which is calculated through the color adjustment in normalized diffuse color space, and bottom right the final image I'.

7.2. Highlight Gloss Remapping

Highlights are by definition specular, and can therefore be edited by modifying only the specular image. The Blinn-Phong model for 3D rendering can be applied to the specular image to realistically adjust the shininess. To calculate an object's radiance, the Blinn-Phong model sums three separate components I_a, I_d and I_s, each of which estimates a specific real-world lighting phenomenon. The first component is the ambient illumination, which is caused by the scattering of all light from the environment excluding light sources onto an object. This results in one constant color across the entire object independent of the orientation of that object. The second component is the diffuse component, which is caused by the object's body reflectance of one or more incoming light sources. The third component is the specular component, which is caused by the object's surface reflection of the same light sources as the diffuse component. In the Blinn-Phong model's specular component, the shininess is determined by the power n:

$$I_s = k_s * L \tag{7.8}$$

$$k_s = (\hat{N} \cdot \hat{H})^n \tag{7.9}$$

where L is the color and intensity of the irradiant light. To change the shininess from n to n', the specular image should be raised to the power n'/n:

$$(k_s)^{n'/n} = ((\hat{N} \cdot \hat{H})^n)^{n'/n} \tag{7.10}$$

$$(k_s)^{n'/n} = (\hat{N} \cdot \hat{H})^{n*n'/n} \tag{7.11}$$

$$(k_s)^{n'/n} = (\hat{N} \cdot \hat{H})^{n'} \tag{7.12}$$

Because L is approximately a constant vector, modifying I_s has the same effect as modifying k_s directly, except for a scaling factor $L^{n'/n}$. The remaining problem is then estimating L to compensate for it. In our implementation we estimate L at any pixel as the average of the specular component across a large patch around that pixel. Then the specular image can be adjusted:

$$I'_s = (\frac{I_s}{|L|})^{n'/n} * |L| \tag{7.13}$$

giving the same result as Equation 7.12. This was tested on 'potdng' (see Figure 7.2), making all objects appear shinier.

7.3. Intelligent Tonemapping

Tonemapping is the process of changing the luminosity and chromaticity of an image with the intent of maintaining its original look while changing the way it is displayed. Images from digital cameras are often tonemapped due to the small dynamic range of monitors compared to the dynamic range of real world scenes. When a daylight scene is displayed on a monitor without tonemapping, there is low contrast in small scale features. Tonemapping can bring out the small scale features by compressing the large scale features. This commonly has the effect of making the lighting look washed out.

One way to overcome this problem would be to know what areas are shadows and have no direct illumination, and which areas are directly lit and should be kept bright. Using the diffuse-specular decomposition, I_d can be separately tonemapped such that specularities are not adjusted. This still brings out the small-scale details without creating a washed out look due to compressed specularities. To showcase this effect, we implemented a simplified version of Durand and Dorsey's tonemapping algorithm (see Figure 7.3). The specular reflection of the sun is kept bright, resulting in an image with more local contrast.

Figure 7.2: Comparing two highlight adjustments in sample ‘potdng’. Normal highlights on the left, highlights adjusted with $n'/n = 1.5$ in the middle, and highlights adjusted with $n'/n = 2.0$ on the right.

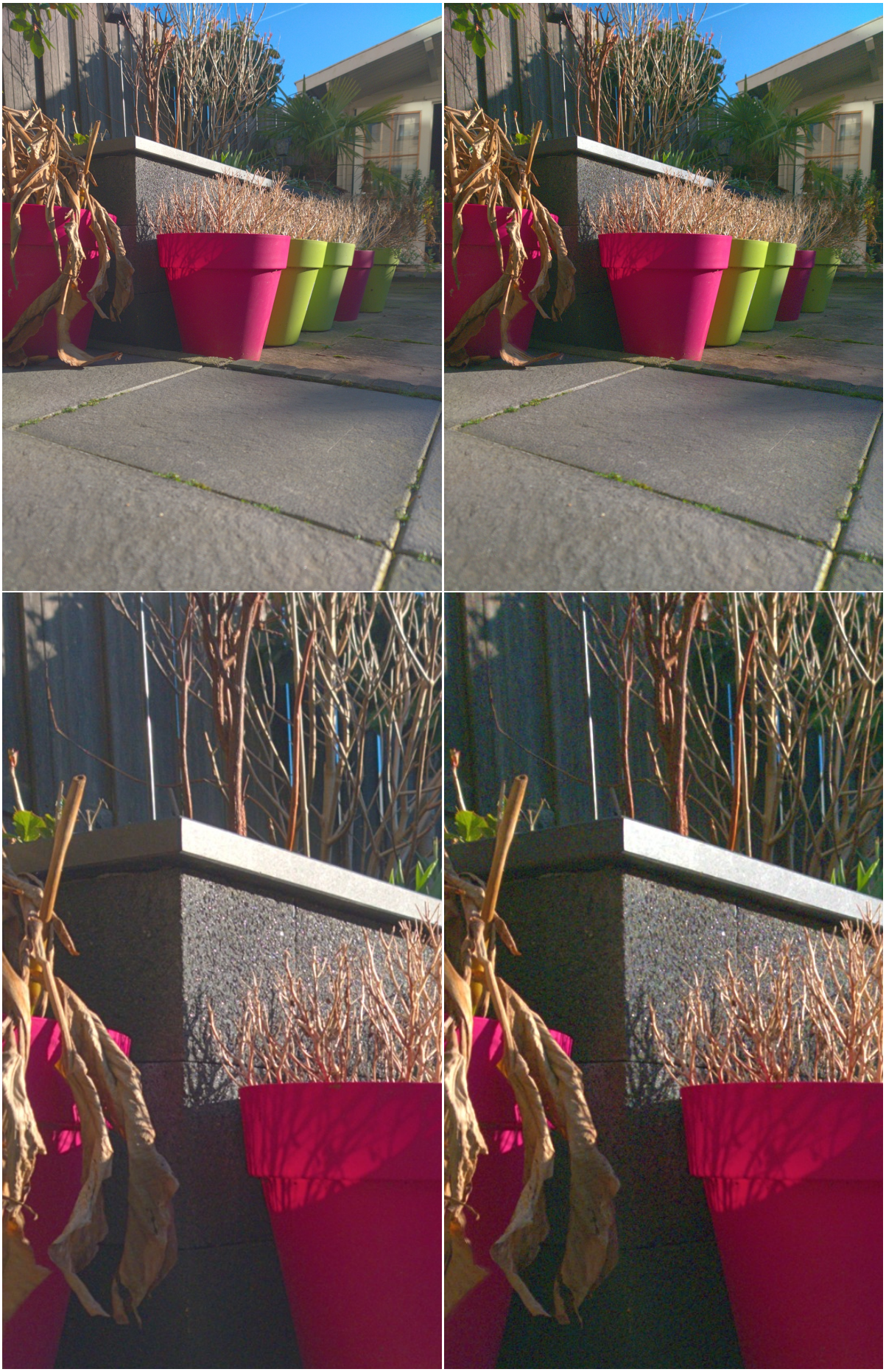

Figure 7.3: Comparing a normal tonemapping algorithm operating on I (left) and a modified tonemapping algorithm operating exclusively on the diffuse component I_d (right). The diffuse tonemapping algorithm appears less washed out due to maintaining the original uncompressed highlights and ignoring a small amount of polarized ambient light in the shadows.

8

Discussion

Despite the improvements regarding robustness and runtime, the results show that our approach is still imperfect. This section discusses both the contributions of this work, and the remaining challenges that could be resolved in future iterations.

8.1. Contributions

The proposed algorithm offers four key improvements over the work by Nayar et al (Nayar et al., 1993). First, the entire algorithm is massively parallelizable, without the requirement for an iteration over individual pixel coordinates in the implementation. Every operation is vectorized over the entire image, making an implementation on manycore processors or GPUs very performant. Second, the soft decision making from the functional bilateral filter is a strict improvement over hard decision making, as the filter itself can be replaced by a binary decision to revert to the hard decision version. Third, the line-intersection algorithm provides a more robust alternative to the plane-intersection algorithm due to the removal of any angular thresholds. Finally, the performance of the algorithm is similar for any resolution image, as the multi-scale representation using a Gaussian pyramid ensures that data diffusion happens at the same speed as long as the pyramid scale count is chosen to match with the image size. If the image size is twice as large in both dimensions, adding one pyramid level is enough to retain the information diffusion speed.

8.2. Future Work

One of the major problems induced by the sequential acquisition of sample sets was the temporal change in the scene between measurements due to the motion of the camera and objects and the changes in appearance from varying illumination intensity and direction. The motion can be compensated for using alignment. However, if the alignment is imperfect, edges are seen as highly polarized areas regardless of the actual amount of polarization on the edges, reducing the quality of the decomposition (see Figure 8.1). Recently, there have been developments towards a full-Stokes polarization camera (Rubin et al., 2019), that can directly provide the polarization cosine from one image, using special sensor patterns to measure different polarization orientations across the image. This would greatly increase the quality of the diffuse-specular decomposition, and allow for the post-processing of videos.

Another aspect that can be improved is the runtime. We used an interpreted programming language due to its ease of experimentation, but a native implementation would result in a much lower runtime. If the decomposition could run on video data, the diffuse component could be shown as a version of the video data with better visibility. This is useful when the visibility of the scene is poor due to highly directional light sources in dark areas. Furthermore, a diffuse video stream can be compressed better due to having less variance, giving smaller file sizes when the trade-off of losing the specular highlights is acceptable.

If the performance can be improved to real-time processing, the decomposition could be applied on a video stream as the video is being recorded. This would provide a viewfinder showing only the diffuse component of the video stream, which would be helpful for situations where the only source of light creates many strong specularities, such as in underground construction work with flashlights as the light sources. Further research towards optimization techniques and implementation in a native programming language is required to test the viability of such a real-time application.

Figure 8.1: I_{mean} (left) and I_{amp} (right) of sample set 'window', showing the result of the movement of clouds on the polarization cosine. The second row shows a close-up of the sky. The pattern in the sky seen in I_{amp} is not created by the polarization state, but only by the movement of the clouds, which creates temporal intensity differences. This is incorrectly interpreted by the polarization cosine calculation as polarization state differences.

9

Conclusion

The diffuse-specular decomposition is still a relevant operation for modern image processing pipelines. By taking four measurements of a scene through a linearly polarized filter, the polarization state can be calculated in the form of polarization cosines which can then be used as the basis for a decomposition algorithm. Our method uses two key assumptions about real world scenes, local diffuse color vector direction constancy and local Fresnel ratio constancy, to provide a robust decomposition for both indoor and outdoor scenes. The results show that the decorrelation approach used can accurately distinguish between diffuse and specular light in simple cases, and still provides a good estimation in the more difficult cases where there is camera sensor clipping or there are large reflections in the scene.

The decomposition can be applied to provide additional illumination information to multiple post-processing operations to improve their performance. Such post-processing operations include but are not limited to texture modification, glossiness remapping and tonemapping. While modifying a texture and remapping the glossiness of an object is much simpler with the additional information from the decomposition, the tonemapping algorithm we tested only marginally shows a difference between the basic version and the modified version using the decomposition result. Nevertheless, the results of the experiments on the three post-processing operations indicate that these operations can be either simplified or can have their quality improved with the additional information supplied by the diffuse and specular intrinsic images.

Bibliography

Phong, B. T. (1975). Illumination for computer generated pictures. *Commun. ACM*, *18*(6), 311–317. https://doi.org/10.1145/360825.360839

Blinn, J. F. (1977). Models of light reflection for computer synthesized pictures. *Proceedings of the 4th Annual Conference on Computer Graphics and Interactive Techniques*, 192–198. https://doi.org/10.1145/563858.563893

Shafer, S. A. (1985). Using color to separate reflection components. *Color Research & Application*, *10*(4), 210–218. https://doi.org/https://doi.org/10.1002/col.5080100409

Klinker, G. J., Shafer, S. A., & Kanade, T. (1988). The measurement of highlights in color images. *International Journal of Computer Vision*, *2*(1), 7–32. https://doi.org/10.1007/bf00836279

Wolff, L. (1989). Using polarization to separate reflection components. *Proceedings CVPR '89: IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 363–369. https://doi.org/10.1109/CVPR.1989.37873

Wolff, L. (1990). Polarization-based material classification from specular reflection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*(11), 1059–1071. https://doi.org/10.1109/34.61705

Nayar, S., Fang, X.-S., & Boult, T. (1993). Removal of specularities using color and polarization. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 583–590. https://doi.org/10.1109/CVPR.1993.341071

Bhat, D. N., & Nayar, S. K. (1994). Stereo in the presence of specular reflection. https://doi.org/10.7916/D8QR5577

Tomasi, C., & Manduchi, R. (1998). Bilateral filtering for gray and color images. *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, 839–846. https://doi.org/10.1109/ICCV.1998.710815

Wolff, L. B., Nayar, S. K., & Oren, M. (1998). *International Journal of Computer Vision*, *30*(1), 55–71. https://doi.org/10.1023/a:1008017513536

Schechner, Y., Narasimhan, S., & Nayar, S. (2001). Instant dehazing of images using polarization. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, *1*, I–I. https://doi.org/10.1109/CVPR.2001.990493

Durand, F., & Dorsey, J. (2002). Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. Graph.*, *21*(3), 257–266. https://doi.org/10.1145/566654.566574

Schechner, Y. Y., Narasimhan, S. G., & Nayar, S. K. (2003). Polarization-based vision through haze. *Appl. Opt.*, *42*(3), 511–525. https://doi.org/10.1364/AO.42.000511

Eisemann, E., & Durand, F. (2004). Flash photography enhancement via intrinsic relighting [publisher: ACM Press]. *ACM Transactions on Graphics (Proc. of SIGGRAPH)*, *23*. http://graphics.tudelft.nl/Publications-new/2004/ED04a

Petschnigg, G., Szeliski, R., Agrawala, M., Cohen, M., Hoppe, H., & Toyama, K. (2004). Digital photography with flash and no-flash image pairs. *ACM Trans. Graph.*, *23*(3), 664–672. https://doi.org/10.1145/1015706.1015777

Meylan, L., Daly, S., & Süsstrunk, S. (2007). Tone mapping for high-dynamic range displays. In B. E. Rogowitz, T. N. Pappas, & S. J. Daly (Eds.), *Human vision and electronic imaging xii* (pp. 370–381). SPIE. https://doi.org/10.1117/12.706472

Stroia-Williams, P., Hilton, A., & Grau, O. (2010). Reflectance transfer for material editing and relighting. https://doi.org/10.20385/1860-2037/7.2010.6

Takatani, T., Mukaigawa, Y., Matsushita, Y., & Yagi, Y. (2018). Decomposition of reflection and scattering by multiple-weighted measurements. *IPSJ Transactions on Computer Vision and Applications*, *10*. https://doi.org/10.1186/s41074-018-0049-4

Rubin, N. A., D'Aversa, G., Chevalier, P., Shi, Z., Chen, W. T., & Capasso, F. (2019). Matrix fourier optics enables a compact full-stokes polarization camera. *Science*, *365*(6448), eaax1839. https://doi.org/10.1126/science.aax1839