

A Polarizing Filter Function for Real-Time Rendering

Viktor Enfeldt

Blekinge Institute of Technology

Prashant Goswami

Blekinge Institute of Technology

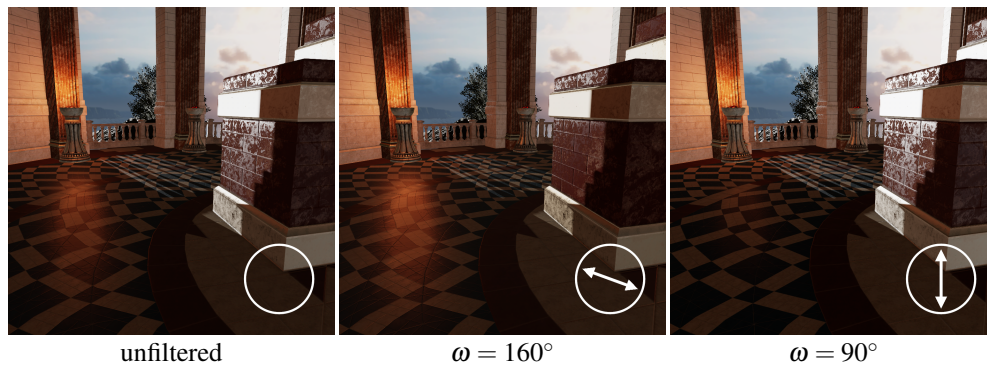


Figure 1. Our polarizing filter function is capable of simulating the reflection-altering abilities of real polarizing filters. From left to right: the *Sun Temple* scene rendered without a polarizing filter, with a polarizing filter oriented to reduce reflections in the low wall, and with a polarizing filter oriented to reduce reflections in the floor.

Abstract

We present a function that can be used in conventional non-polarizing renderers to simulate the reflection-altering visual effects of real polarizing filters, without having to replace the existing light and surface representations. The relevant Stokes-Mueller polarization calculations are simplified so that neither Stokes vectors nor Mueller matrices are needed in the finished implementation. Our function approximates the surface's complex refractive index with its specular color, and the accuracy of this approximation is demonstrated with some common conductor materials; no approximation needs to be made for dielectric materials. As our function only affects specularly reflected light, it cannot simulate all the visual effects produced by real polarizing filters, only the reflection-reducing ones. We show the visual effects of our filter function and measure its execution time in a real-time rendered application. The function's correctness is verified by comparing it with a filter implemented in a polarizing offline renderer. HLSL source code is provided for the real-time implementation.

1. Introduction

When placed in front of a camera lens, a polarizing filter can have a significant visual impact on the light that reaches the image sensor: it can reduce glare and reflections, make glass and water appear more see-through, make foliage appear more vibrant, and darken the sky. As these filters are used by many photographers, a real-time simulation of them for computer graphics would likely be a welcome addition to the photo modes that have become commonplace in modern video games. Such an implementation might also be useful in driving simulators, as it would reduce glare from reflections in the road and other vehicles just as polarized sunglasses do. The effects of a polarizing filter are difficult to replicate once a photo has been taken, as their simulation requires information about the incoming light's polarization state. Therefore, a computer graphics simulation of the effects needs to simulate this polarization information as well.

Polarization itself has no significant impact on how our eyes perceive light. It has therefore, to the best of the authors' knowledge, not yet been included in the lighting equations of any commercially used real-time renderer. However, it has previously been modeled in some research-focused ray-traced and beam-traced renderers [Wolff and Kurlander 1990; Weidlich and Wilkie 2008; Vlker and Hamann 2013; Nimier-David et al. 2019], and several mathematical models have been developed over the years to analyze and describe polarized light and its interactions with various materials and optical filters [Chipman 1995].

In this paper, we use the Stokes-Mueller calculus to develop a polarizing filter function that can be used to control specular reflections from direct and image-based light sources in conventional non-polarizing real-time renderers. Our function requires minimal changes to existing rendering implementations and relies on the material's specular color instead of its complex index of refraction (IOR), as the former information is commonly available in real-time renderers while the latter is not.

2. Theory

Some background information about Fresnel reflectance, specular color, polarizing filters, and the Stokes-Mueller polarization calculus is needed to understand how we arrived at our polarizing filter function.

2.1. Fresnel Reflectance

The Fresnel equations are a central part of realistic physically-based rendering (PBR), as they model both how reflections appear more mirror-like at grazing angles as well as the distinct visual appearance of conductors (i.e., metals). They describe the ratio of incoming light that is reflected in the angle of incidence $\theta \in [0^\circ, 90^\circ)$, and they are split into two polarized parts: F_\perp is linearly polarized perpendicular to the plane of

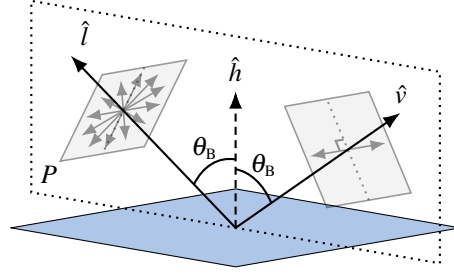


Figure 2. Light from the light source direction \hat{l} that is reflected towards the view direction \hat{v} in a dielectric surface at Brewster's angle θ_B becomes completely linearly polarized (in an angle perpendicular to the plane of incidence P). The halfway vector \hat{h} is the normalized vector at the half angle between \hat{l} and \hat{v} ; it can be thought of as the normal vector of the participating surface microfacets.

incidence and F_{\parallel} is linearly polarized parallel to the plane of incidence. They are both defined using the surface's complex relative IOR, $\eta = n + i\kappa$ as

$$\begin{aligned} F_{\perp}(\eta, \theta) &= \frac{a^2 + b^2 - 2a \cos \theta + \cos^2 \theta}{a^2 + b^2 + 2a \cos \theta + \cos^2 \theta}, \\ F_{\parallel}(\eta, \theta) &= \frac{a^2 + b^2 - 2a \sin \theta \tan \theta + \sin^2 \theta \tan^2 \theta}{a^2 + b^2 + 2a \sin \theta \tan \theta + \sin^2 \theta \tan^2 \theta} F_{\perp}(\theta, \eta), \end{aligned} \quad (1)$$

with a and b defined by the equations

$$\begin{aligned} 2a^2 &= \sqrt{(n^2 - \kappa^2 - \sin^2 \theta)^2 + 4n^2 \kappa^2} + (n^2 - \kappa^2 - \sin^2 \theta), \\ 2b^2 &= \sqrt{(n^2 - \kappa^2 - \sin^2 \theta)^2 + 4n^2 \kappa^2} - (n^2 - \kappa^2 - \sin^2 \theta), \end{aligned} \quad (2)$$

where n is the real refractive index and κ is the extinction coefficient [Wilkie and Weidlich 2012]. For dielectric (i.e., non-metallic) materials, $n > 1$ and $\kappa = 0$.

Brewster's angle. When light is specularly reflected in a dielectric surface at the incident angle θ_B , known as Brewster's angle, F_{\parallel} becomes zero and all of the reflected light becomes linearly polarized in the direction perpendicular to the plane of incidence. A visualization of this interaction is shown in Figure 2. It is due to this phenomena that a polarizing filter, which blocks light that is linearly polarized perpendicular to the filter's alignment, can be used to filter out¹ reflections in dielectric surfaces.

2.2. Specular Color and Schlick's Approximation

In real-time renderers, the Fresnel reflectance behavior of a material is typically defined by its specular color R_0 (i.e., the intensity-reflection coefficient at normal incidence)








































¹or highlight, depending on the alignment of the filter

instead of by its complex refractive index. If the reflecting surface is surrounded by air, R_0 can be calculated from the wavelength-dependent complex IOR [Bennett 1995]:

$$R_0 = \frac{(n-1)^2 + \kappa^2}{(n+1)^2 + \kappa^2}. \quad (3)$$

Refractive index and specular color values of some common conductor and dielectric materials are listed in Section 2.2. Dielectrics all have $\kappa = 0$ and, with the exception of diamond, $n = 1.5 \pm 0.2$ throughout the visible spectrum. Conductors have much more varied n and κ values, both from material to material and throughout the visible spectrum for the same material. Since the real refractive indexes are spectrally varying and can not be modeled in a conventional RGB renderer, the approximated values listed here will introduce some error into our method.

Schlick's approximation. As only the intensity of reflected light tends to be of interest in conventional renderers, the Fresnel functions used only need to calculate the average value of the terms F_{\perp} and F_{\parallel} from Equation (1). A commonly used function for this

Material	n [RGB]	κ [RGB]	R_0 [RGB]
Aluminum	1.346, 0.965, 0.617 	7.475, 6.400, 5.303 	0.91, 0.91, 0.92 
Brass	0.444, 0.527, 1.094 	3.695, 2.765, 1.829 	0.89, 0.79, 0.43 
Copper	0.271, 0.677, 1.316 	3.609, 2.625, 2.292 	0.93, 0.72, 0.50 
Gold	0.183, 0.421, 1.373 	3.424, 2.346, 1.770 	0.94, 0.78, 0.37 
Iron	2.911, 2.950, 2.585 	3.089, 2.932, 2.767 	0.53, 0.51, 0.50 
Lead	1.910, 1.830, 1.440 	3.510, 3.400, 3.180 	0.63, 0.63, 0.64 
Platinum	2.376, 2.085, 1.845 	4.266, 3.715, 3.137 	0.68, 0.64, 0.59 
Silver	0.159, 0.145, 0.135 	3.929, 3.190, 2.381 	0.96, 0.95, 0.92 
Titanium	2.741, 2.542, 2.267 	3.814, 3.435, 3.039 	0.62, 0.58, 0.54 
Diamond	2.409, 2.423, 2.539 	0.000, 0.000, 0.000	0.17, 0.17, 0.19 
Glass*	1.521, 1.525, 1.532 	0.000, 0.000, 0.000	0.04, 0.04, 0.04 
Ice	1.308, 1.311, 1.316 	0.000, 0.000, 0.000	0.02, 0.02, 0.02 
Plastic†	1.579, 1.589, 1.608 	0.000, 0.000, 0.000	0.05, 0.05, 0.05 
Quartz	1.457, 1.460, 1.466 	0.000, 0.000, 0.000	0.03, 0.04, 0.04 
Water	1.331, 1.333, 1.337 	0.000, 0.000, 0.000	0.02, 0.02, 0.02 

* Soda-lime glass – Clear.

† PC – Polycarbonate.

Table 1. The complex IOR and specular color of some common materials. Refractive index values are taken from <https://refractiveindex.info/>'s [Polyanskiy (n.d.)] “selected data for 3D artists” section, with red, green, and blue defined as 650 nm, 550 nm, and 450 nm wavelengths, respectively. Subsequent graphs and renders use these material values unless stated otherwise.

calculation is the following approximation by Schlick [1994]:

$$F = \frac{F_{\perp} + F_{\parallel}}{2} \approx F_{\text{Schlick}}(R_0, \theta) = R_0 + (1 - R_0)(1 - \cos \theta)^5. \quad (4)$$

2.3. Polarizing Filters

Linear polarizing filters only let light waves through if they are linearly polarized in the same orientation as the filter. They can, therefore, be used to filter out specularly reflected light from dielectric surfaces; how much of the reflected light is filtered out depends on how close the incident angle is to Brewster's angle.

Malus' law. The intensity I that passes through a perfect linear polarizing filter is

$$I = I_0 \cos^2 \varphi,$$

where I_0 is the intensity of the incoming linearly polarized light and φ is the angle between the filter's polarization orientation and the incoming light's angle of polarization. As unpolarized light consists of light that is randomly polarized in all angles and the average value of $\cos^2 \varphi$ is $1/2$, a polarizing filter will, on average, block half of the incoming light.

2.4. The Stokes-Mueller Calculus

Several mathematical models of polarized light and its interactions with various media have been developed over the years. To develop our polarizing filter function, we have chosen to work with the Stokes-Mueller calculus.

Only the parts necessary to understand the development of our polarizing filter function will be covered in this article. For a more comprehensive introduction to polarization and the Stokes-Mueller calculus, we refer the reader to literature on optics, such as Shurcliff's book [1962]. For an implementation-focused overview of polarization rendering, see the course notes by Wilkie and Weidlich [2012].

2.4.1. Stokes Vectors

The Stokes parameters describe the polarization state of a quasi-monochromatic² light ray. The parameters consist of four real numbers in the form of a Stokes vector:

$$\vec{S} = \begin{bmatrix} I \\ Q \\ U \\ V \end{bmatrix} = \begin{bmatrix} \leftrightarrow + \updownarrow \\ \leftrightarrow - \updownarrow \\ \nearrow - \nwarrow \\ \odot - \ominus \end{bmatrix}, \quad (5)$$

²i.e., light which has a very narrow wavelength spectrum

where $I \in [0, \infty)$ is identical to the intensity value used for RGB color components in conventional renderers, $Q \in [-I, I]$ is the difference between the amplitude of polarization in the horizontal and vertical axis, $U \in [-I, I]$ is the difference between the amplitude of polarization in the two vertical axes, and $V \in [-I, I]$ describes the handedness of the elliptical polarization. The arrow notation in Equation (5) is a visualization of how these terms are calculated (from the perspective of an observer looking towards the light source). For linearly polarized light, the combination of Q and U defines the angle of linear polarization with respect to a reference vector in the Stokes vector's local coordinate system.

Degree of polarization (DOP). The ratio of polarized light to unpolarized light is described by the degree of polarization. It is calculated from the Stokes parameters as

$$p = \frac{\sqrt{Q^2 + U^2 + V^2}}{I}, \quad (6)$$

with $p = 1$ describing completely polarized light and $p = 0$ describing completely unpolarized light. With this definition, it follows that completely unpolarized quasi-monochromatic light is represented by the Stokes vector $\vec{S} = (I, 0, 0, 0)$.

2.4.2. Mueller Matrices

Mueller matrices are real-valued four-by-four matrices that can describe how various optical filters and surfaces affect the polarization and intensity of light [Chipman 1995]. The resulting Stokes vector from light interacting with a surface or optical filter, described by the Mueller matrix \mathbf{M} , is calculated as

$$\vec{S}' = \mathbf{M}\vec{S}.$$

For the purposes of this article, only the Mueller matrices for Fresnel reflectance and linear polarizing filters are of interest.

Fresnel reflectance. The Mueller matrix for Fresnel reflectance is defined as

$$\mathbf{M}_{\text{Fresnel}} = \begin{bmatrix} A & B & 0 & 0 \\ B & A & 0 & 0 \\ 0 & 0 & C & S \\ 0 & 0 & -S & C \end{bmatrix},$$

with

$$\begin{aligned} A &= \frac{F_{\perp} + F_{\parallel}}{2}, & C &= \cos(\delta) \sqrt{F_{\perp} F_{\parallel}}, \\ B &= \frac{F_{\perp} - F_{\parallel}}{2}, & S &= \sin(\delta) \sqrt{F_{\perp} F_{\parallel}}, \end{aligned} \quad (7)$$

where F_{\perp} and F_{\parallel} are the Fresnel terms from Equation (1), and δ is the difference in phase shift between the light that is polarized along the perpendicular and parallel axes. Note that A here is identical to the averaged Fresnel function F from Equation (3).

The formulas listed in [Section 2.4.2](#) and [Equations \(7\) and \(8\)](#) are from Wilkie and Weidlich [2012]. Formulas for δ are omitted from this paper, as the terms C and S have no effect on the unpolarized Stokes vectors used in our method.

Linear polarizing filters. A perfect horizontal linear polarizing filter that, from the perspective of an observer looking towards the light source, has been rotated counter-clockwise by an angle φ is described by the Mueller matrix

$$\mathbf{M}_{\text{pf}\varphi} = \frac{1}{2} \begin{bmatrix} 1 & \cos(2\varphi) & \sin(2\varphi) & 0 \\ \cos(2\varphi) & \cos^2(2\varphi) & \cos(2\varphi)\sin(2\varphi) & 0 \\ \sin(2\varphi) & \cos(2\varphi)\sin(2\varphi) & \sin^2(2\varphi) & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad (8)$$

3. Our Method

Although we utilize the Stokes-Mueller calculus to develop our approximate polarizing filter function, called Ω , no Stokes vectors or Mueller matrices are needed in the implementation of Ω thanks to simplifications permitted by the following assumptions:

- A1. *All incident light is unpolarized.* Although the Sun is generally a source of unpolarized light [Goldstein 2011], the blue of a clear sky, and some light sources, such as LCD panels, produce various degrees of linearly polarized light. The polarizing filter's impact on reflections from polarized light sources and previous reflections is not accurately simulated by our method, as we only simulate the closest reflection point's polarization contribution.
- A2. *Only specularly reflected light can be polarized.* Just as in Wolff and Kurlander's early work on polarization rendering [1990], we assume that all diffusely reflected light is completely unpolarized. Light from diffuse reflections in thin materials can show a notable degree of polarization [Collin et al. 2014], which might have a noticeable impact on the filtered appearance of painted conductor surfaces and coated materials; however, the visual impact this diffusely reflected light has on polarizing filters can generally be assumed to be much less significant than the impact from polarized light caused by specular reflections.
- A3. *The specular material color is available.* To accurately model how polarizing filters affect reflected light from conductor surfaces, information about both terms of the material's per-color channel complex IOR would be needed. As this information is rarely available in renderers, our approximate implementation instead relies on the more commonly available specular color R_0 .

3.1. Developing the Polarizing Filter Function

Our approximate polarizing filter function Ω is the result of combining these five steps:

1. Calculate the result of a Fresnel reflection with unpolarized incident light.
2. Apply a polarizing filter to the resulting Stokes vector to get an expression of how the filter affects the light's intensity.
3. Convert the result of the previous step to a function that can be used on light which has already been affected by a Fresnel factor (e.g., image-based lighting which includes a pre-computed Fresnel term in one of the split-sum textures [Karis 2013]). The resulting function Ω_{IOR} is our exact IOR-based polarizing filter function.
4. In order to simplify things further in Step 5, introduce a function ψ that represents the degree of horizontal linear polarization.
5. Approximate ψ with a function called $\tilde{\psi}$ that uses the specular color R_0 instead of the complex refractive index η in its calculations.

3.2. Step 1: Unpolarized Light After Fresnel Reflectance

Multiplying a Stokes vector representing unpolarized light of the wavelength λ with the Mueller matrix for Fresnel reflectance from Section 2.4.2 produces the Stokes vector

$$\vec{S}'_{\lambda} = \mathbf{M}_{\text{Fresnel}} \vec{S}_{\lambda} = \begin{bmatrix} A & B & 0 & 0 \\ B & A & 0 & 0 \\ 0 & 0 & C & S \\ 0 & 0 & -S & C \end{bmatrix} \begin{bmatrix} I_{\lambda} \\ 0 \\ 0 \\ 0 \end{bmatrix} = I_{\lambda} \begin{bmatrix} A \\ B \\ 0 \\ 0 \end{bmatrix}. \quad (9)$$

Thus, the C and S terms from Equation (7) do not need to be calculated when unpolarized incident light is reflected in a surface, as they do not affect the reflected result. As per Equation (6), the resulting Stokes vector's degree of polarization is equal to $\frac{B}{A}$.

3.3. Step 2: Filtering the Result With a Polarizing Filter

Next, we apply a linear polarizing filter $\mathbf{M}_{\text{pf}\phi}$ from Equation (8) to the reflected result from Equation (9), divide the result with the incident light's intensity I_{λ} , and double the intensity in accordance with Malus' law to match the intensity of the light that is not processed with the filter function (e.g., diffuse reflections). This results in the

Stokes vector

$$\vec{S}_\lambda'' = \frac{2\mathbf{M}_{\text{pf}\varphi}\vec{S}_\lambda'}{I_\lambda} = \begin{bmatrix} A + B\cos(2\varphi) \\ A\cos(2\varphi) + B\cos^2(2\varphi) \\ A\sin(2\varphi) + B\cos(2\varphi)\sin(2\varphi) \\ 0 \end{bmatrix},$$

which can be seen as the polarizing factor of the reflected light that passed through a polarizing filter oriented at an angle $\varphi + 90^\circ$ relative to the plane of incidence.

As we are only interested in how the polarizing filter affects the intensity of the light, we discard all but the first parameter and express it as a Fresnel factor with an included polarizing filter:

$$F_\varphi = A + B\cos(2\varphi).$$

3.4. Step 3: The Exact Polarizing Filter Function Ω_{IOR}

Although we can now describe how polarizing filters affect the intensity of light without having to use Stokes vectors and Mueller matrices, we would prefer to have a polarizing filter factor $\Omega \in [0, 2]$ that can be applied to light which has already been multiplied with a Fresnel term, as that would be more widely applicable.

We recall from Equation (7) that A is equivalent to the exact Fresnel factor from Equation (4); therefore, assuming that the light has already been multiplied with the Fresnel factor F , the exact IOR-based polarizing filter factor is defined as

$$\Omega_{\text{IOR}}(\eta, \theta, \varphi) = \frac{F_\varphi}{F} = \frac{A + B\cos(2\varphi)}{A} = \frac{B}{A}\cos(2\varphi) + 1.$$

3.5. Step 4: The Horizontal Linear Polarizing Factor ψ

To be able to simplify this further, we define a function $\psi(\eta, \theta) = \frac{B}{A}$ that is equal to the DOP of the unfiltered reflected light in Step 1. It can be thought of as the linear polarizing factor of the surface at the given incident angle.

By substituting A and B with their definitions from Equation (7) and, subsequently, F_\parallel and F_\perp with their definitions from Equation (1), the polarizing factor simplifies to

$$\psi(\eta, \theta) = \frac{\sqrt{2} \sqrt{\sqrt{(n^2 - \kappa^2 - \sin^2 \theta)^2 + 4n^2 \kappa^2} + n^2 - \kappa^2 - \sin^2 \theta} \cos \theta \sin^2 \theta}{\sqrt{(n^2 - \kappa^2 - \sin^2 \theta)^2 + 4n^2 \kappa^2} \cos^2 \theta + \sin^4 \theta},$$

which can be expressed in a more compact form as

$$\psi(\eta, \theta) = \frac{g \cos \theta \sin^2 \theta}{h \cos^2 \theta + \sin^4 \theta}, \quad (10)$$

where

$$\begin{aligned} g &= \sqrt{2}2a = \sqrt{2}\sqrt{h+c}, \\ h &= a^2 + b^2 = \sqrt{c^2 + 4n^2\kappa^2}, \\ c &= n^2 - \kappa^2 - \sin^2 \theta. \end{aligned} \quad (11)$$

Note that a and b here are the same as the a and b terms used in Equations (1) and (2).

The shape of ψ . From the graphs in Figure 3, it is apparent that dielectric materials can polarize light more strongly than conductors can, with the reflected light becoming completely linearly polarized when $\psi = 1$ (which occurs at Brewster's angle).

Although reflections in conductor surfaces never become completely linearly polarized and can thus never be fully blocked by a polarizing filter, there is a notable variance among conductors in how polarized the reflected light becomes. Reflections in silver surfaces (the pink lines in the top graph) remain almost completely unpolarized at all incident angles, while reflections in iron surfaces (the dark gray lines in the bottom graph) can reach a DOP of over 50% at incident angles around 70°. For some conductors (brass, copper, gold), there is also a large variance in how polarized the reflected light's three color components become, with blue becoming the most polarized of the three and red the least polarized.

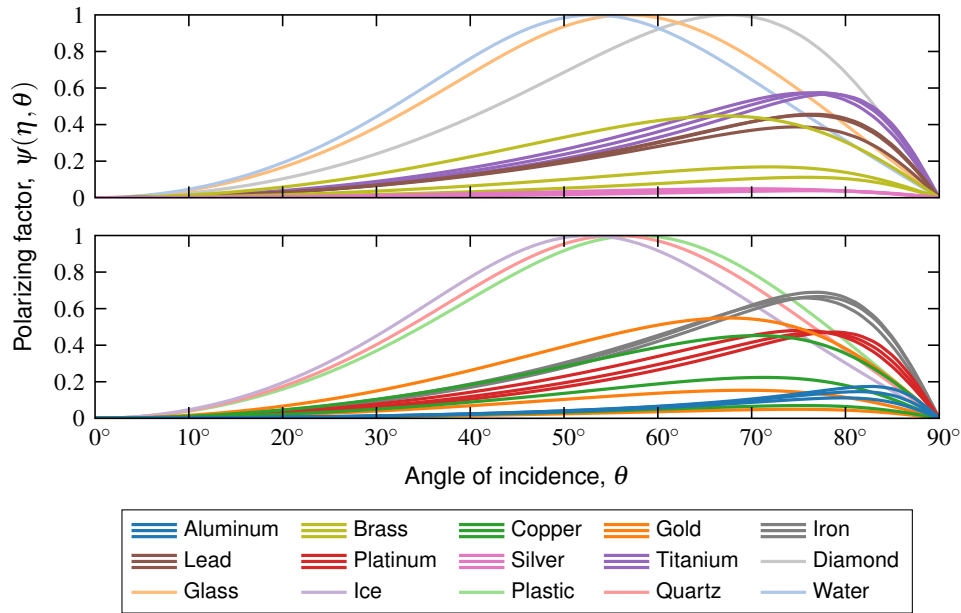


Figure 3. The shape of the polarizing factor ψ for some common materials. Individual lines for the red, green, and blue results are shown for conductors, as their polarizing factor can vary significantly throughout the visible spectrum.

3.6. Step 5: The Approximate Polarizing Factor $\tilde{\psi}$

If the values of the complex IOR are known, then the exact polarizing filter function

$$\Omega_{\text{IOR}}(\eta, \theta, \varphi) = \psi(\eta, \theta) \cos(2\varphi) + 1 \quad (12)$$

can be used. However, as complex IOR information is rarely available in real-time renderers, a polarizing filter function based on the specular color R_0 would be preferable. To accomplish this, we replace ψ with two separate functions: one exact polarizing factor ψ_D , for dielectric surfaces; and one approximate polarizing factor $\tilde{\psi}_C$, for conductor surfaces.

Dielectrics. For dielectric surfaces, we use the correct function ψ with the specular color R_0 as the input parameter instead of η ; since $n > 1$ and $\kappa = 0$ for dielectrics, the polarizing factor ψ from Section 3.5 can be simplified to

$$\psi_D(R_0, \theta) = \frac{2\sqrt{n^2 - \sin^2 \theta} \cos \theta \sin^2 \theta}{(n^2 - \sin^2 \theta) \cos^2 \theta + \sin^4 \theta}, \quad (13)$$

where, as per the definition of R_0 in Equation (3), n is given by the equation

$$n = \frac{1 + \sqrt{R_0}}{1 - \sqrt{R_0}}. \quad (14)$$

Some renderers (e.g., Unreal Engine 4 [Karis 2013]) hard-code the R_0 values of dielectrics to 0.04 (i.e., $n = 1.5$), so in those cases, n in Equation (13) can be replaced by the constant value 1.5. A HLSL implementation of the dielectric polarization function ψ_D , which uses Equation (14) to calculate n , is shown in Listing 1.

```

1 // R0: specular color, ct: cos(theta), st2: sin^2(theta)
2 float psiDielectricExact(float R0, float ct, float st2) {
3     float n = (1.0 + sqrt(R0))/(1.0 - sqrt(R0));
4     float c = n*n - st2;
5
6     float psi = (2.0*sqrt(c)*ct*st2)/(c*ct*ct + st2*st2);
7     return saturate(psi);
8 }

```

Listing 1. HLSL implementation of the polarizing factor for dielectrics.

Conductors. For conductor surfaces, we unfortunately cannot extract the exact values of n and κ from the specular color and have to resort to an approximate function called $\tilde{\psi}_C$. We note from Equation (11) that only the g and h terms in the equation are dependent on η ; and we rewrite the exact function ψ from Equation (10) with the terms $\beta = \frac{h}{g}$ and $\gamma = \frac{1}{g}$:

$$\psi_C(\eta, \theta) = \frac{\cos \theta \sin^2 \theta}{\left(\frac{h}{g}\right) \cos^2 \theta + \left(\frac{1}{g}\right) \sin^4 \theta} = \frac{\cos \theta \sin^2 \theta}{\beta \cos^2 \theta + \gamma \sin^4 \theta}.$$

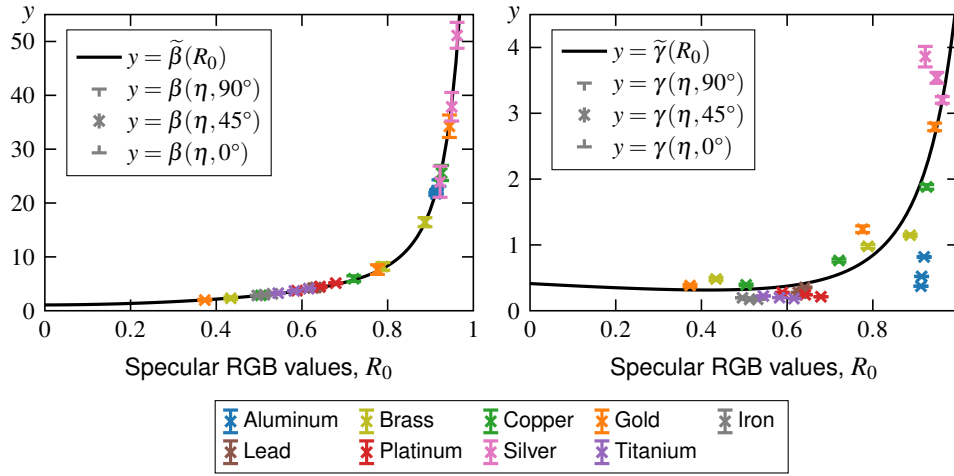


Figure 4. The approximation functions $\tilde{\beta}$ and $\tilde{\gamma}$, shown behind the correct β and γ values of some conductors' red, green, and blue wavelengths. The correct values are plotted for the incident angles $\theta = 90^\circ, 45^\circ$, and 0° .

The correct values of β and γ for some common conductors are plotted against those materials' specular colors R_0 in Figure 4. From the correct values we develop two approximate functions, $\tilde{\beta}$ and $\tilde{\gamma}$, that attempt to map the specular color values to the correct values of β and γ :

$$\beta(\eta, \theta) = \frac{h}{g} \approx \tilde{\beta}(R_0) = \frac{0.1}{(1.095 - R_0)^3} + 5.4R_0^2 + 1, \quad (15)$$

$$\gamma(\eta, \theta) = \frac{1}{g} \approx \tilde{\gamma}(R_0) = \frac{0.16R_0^2}{(1.18 - R_0)^2} + 0.35(1.18 - R_0). \quad (16)$$

As can be seen in Figure 4, there is a clear pattern to β and our approximation $\tilde{\beta}$ follows it fairly well. Our approximation $\tilde{\gamma}$, on the other hand, does not follow the correct values as closely, as it does not appear to be possible to accurately map R_0 to γ without having more information about the material. Our approximations do not take the incident angle θ into account, as changes in θ have a negligible impact on the values of both β and γ .

With these two functions, the approximation $\tilde{\psi}_C$ of the exact conductor polarizing factor from Equation (15) is written as

$$\psi_C(\eta, \theta) \approx \tilde{\psi}_C(R_0, \theta) = \frac{\cos \theta \sin^2 \theta}{\tilde{\beta} \cos^2 \theta + \tilde{\gamma} \sin^4 \theta}.$$

A HLSL implementation of $\tilde{\psi}_C$ is shown in Listing 2.

```

1 // R0: specular color, ct: cos(theta), st2: sin^2(theta)
2 float3 psiConductorApprox(float3 R0, float ct, float st2) {
3     float3 Rb = float3(1.095) - R0;
4     float3 Rg = float3(1.18) - R0;
5     float3 beta = float3(0.1)/(Rb*Rb*Rb) + 5.4*R0*R0 + float3(1.0);
6     float3 gamma = (0.16*R0*R0)/(Rg*Rg) + 0.35*Rg;
7
8     float3 psi = (ct*st2)/(beta*ct*ct + gamma*st2*st2);
9     return saturate(psi);
10 }

```

Listing 2. HLSL implementation of the approximate polarizing filter function for conductors.

Combining the two functions. Finally, the approximate polarization function $\tilde{\psi}$ is defined as either ψ_D or $\tilde{\psi}_C$ depending on the material's metalness value at the reflection point:

$$\tilde{\psi}(R_0, \theta) = \begin{cases} \tilde{\psi}_C(R_0, \theta) & \text{if metalness} > 0.5 \\ \psi_D(R_0, \theta), & \text{otherwise.} \end{cases}$$

The metalness value is almost always a binary value (0.0 for dielectrics, 1.0 for conductors); however, values in between are sometimes used when there is a thin layer of dielectric material (such as dirt) on top of a conductor surface. As those in-between values are rare, we choose to use a step function instead of an interpolation function to avoid redundant calculations.

The shape of $\tilde{\psi}$. The two top graphs in Figure 5 show the shape of our approximate polarizing factor, and the two bottom graphs show how much it differs from the exact polarizing factor shown in Figure 3. For dielectrics, the lines exactly match the correct values as no approximations had to be made for them. For conductors, the values match the correct function closely up until around $\theta = 60^\circ$ where they begin to diverge. At incident angles around 80° , most conductors (aluminum, iron, lead, platinum, and titanium) become slightly less polarized by our approximation than by the correct function; however, for some color channels of brass, copper, and gold the polarizing effect is instead stronger with our approximation than with the correct function. The consistently low polarizing factor of silver (the pink lines) remains mostly unaffected by our approximation.

3.7. Our Polarizing Filter Function Ω

With the approximate function $\tilde{\psi}$ now defined, the polarizing filter function Ω can finally be expressed using the specular color R_0 instead of the complex IOR η :

$$\Omega_{\text{IOR}}(\eta, \theta, \varphi) \approx \Omega(R_0, \theta, \varphi) = \tilde{\psi}(R_0, \theta) \cos(2\varphi) + 1. \quad (17)$$

This approximate polarizing filter function Ω is compatible with existing approximate Fresnel functions (e.g., Schlick's approximation), it does not require information about the complex IOR, and it does not require the inclusion of any Stokes vectors or Mueller matrices in the light transport calculations. For dielectric materials, its output

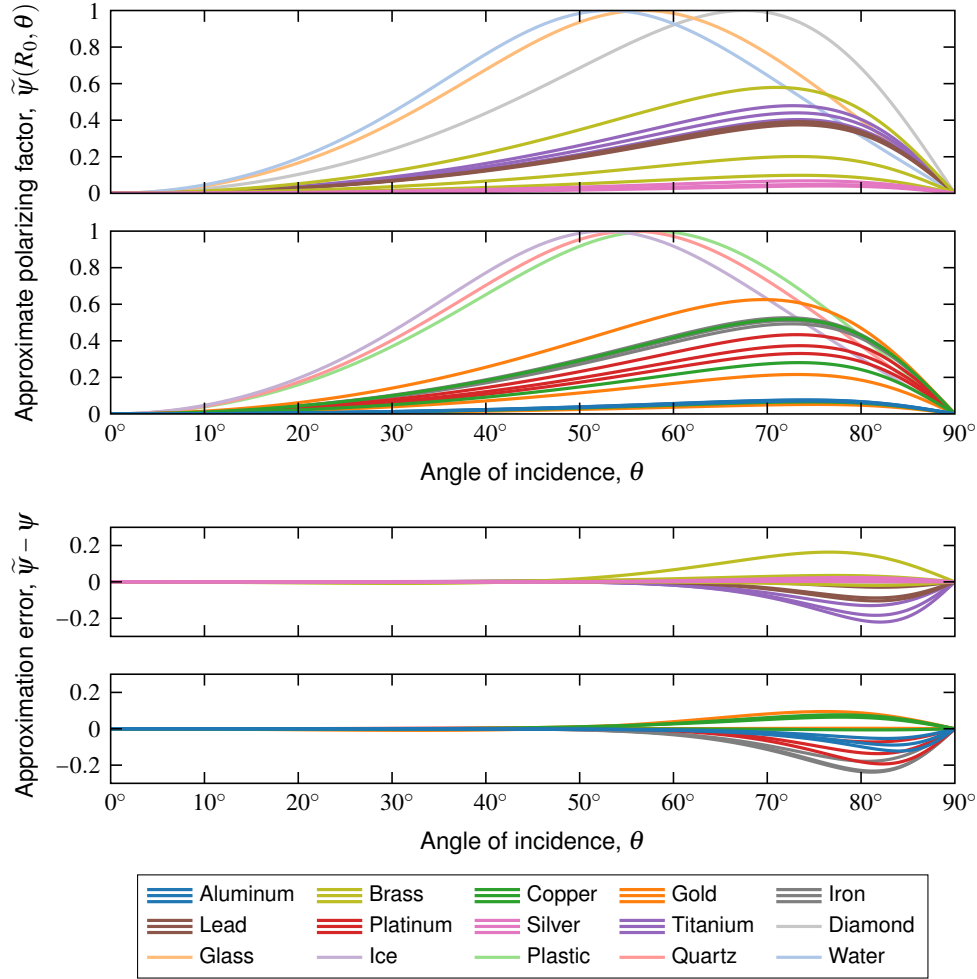


Figure 5. The shape of the approximate polarizing factor $\tilde{\psi}$ and the approximation error, which is calculated as the signed difference between the the approximate polarizing factor $\tilde{\psi}$ (shown above) and the exact polarizing factor ψ (shown in Figure 3).

is identical to Ω_{IOR} if matching specular color and IOR values are used. A HLSL implementation of Ω is shown in Listing 3.

```

1 float3 polarizingFilter(ShadingData sd, LightSample ls, float3 cameraX) {
2     float3 H = normalize(sd.V + ls.L);
3     float angle = calcRelativeAngle(cameraX, H, sd.V);
4     float st = length(cross(ls.L, H)); // sin(theta)
5     float st2 = st*st;                // sin^2(theta)
6
7     float3 psi;
8     if (sd.metalness > 0.5) {
9         psi = psiConductorApprox(sd.specular, ls.LdotH, st2);
10    } else {
11        psi = float3(psiDielectricExact(sd.specular.r, ls.LdotH, st2));
12    }
13
14    return (cos(2.0*angle)*psi + float3(1.0));
15 }

```

Listing 3. HLSL implementation of the polarizing filter function Ω .

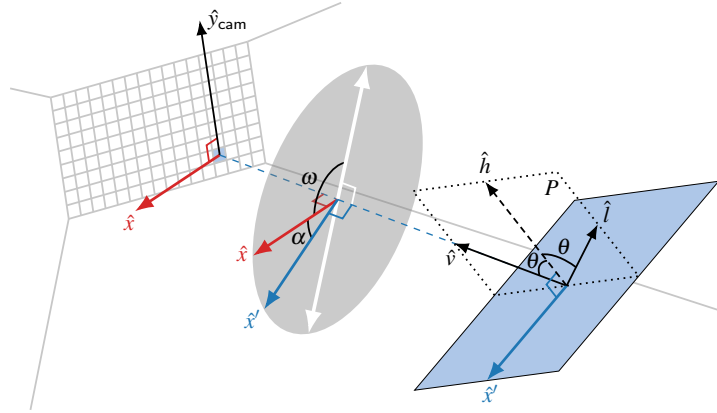


Figure 6. The relation between the vectors and angles used in our calculations. The white double-ended arrow represents the orientation of the linear polarizing filter, \hat{v}_{cam} is the camera's up vector, and P is the plane of incidence (which contains \hat{v} , \hat{h} , and \hat{l}).

The adjusted filter rotation angle φ . The polarizing filter function Ω depends on a rotation angle φ that expresses the alignment of the filter in relation to the reflection surface (more specifically, in relation to a vector orthogonal to the plane of incidence). The angle is calculated as $\varphi = \omega + \alpha$, where ω is the counterclockwise orientation angle of the polarizing filter in relation to the viewer's horizontal axis, and α is analogous to the reference frames that are used to keep track of a Stokes vectors' local coordinate systems in polarizing renderers. The adjustment angle α is calculated as

$$\alpha = \text{atan2}(\hat{v} \cdot (\hat{x}' \times \hat{x}), \hat{x}' \cdot \hat{x}), \quad (18)$$

where \hat{x} is the viewer's reference vector, \hat{x}' is the surface's reference vector, and $\text{atan2}(y, x) \in [-\pi, \pi]$ computes a signed angle in radians between the positive x -axis and the ray to the point (x, y) .

A visualization of the angles α and ω , as well as the vectors involved in Equation (18), is shown in Figure 6, and our HLSL implementation of φ is shown in Listing 4. As a consequence of the viewer's reference vector \hat{x} being defined to be orthogonal to the view direction, the simulated polarizing filter is not perfectly flat, as real polarizing filters typically are, it is instead convex in a shape that allows all incoming light to reach it at normal incidence.

```

1 float calcRelativeAngle(float3 cameraX, float3 H, float3 V) {
2     float3 surfaceX = normalize(cross(H, V));
3     float dotX = dot(surfaceX, cameraX);
4     float detX = dot(V, cross(surfaceX, cameraX));
5
6     return gPolarizingFilterAngle + atan2(detX, dotX);
7 }

```

Listing 4. Calculating the relative polarizing filter angle.

4. Results

To evaluate our polarizing filter function Ω , we implemented it in a modified version of the Falcor rendering framework's *ForwardRenderer* sample [Benty et al. 2019]. Our polarizing filter is applied to the specular term of reflected light contribution from both direct lights³ and image-based lights (IBL): Falcor's specular term for direct lights uses a microfacet-based Cook-Torrance BRDF with Schlick's Fresnel approximation, Smith's geometric term, and the GGX normal distribution function; Falcor's specular term for IBL uses Karis' [2013] split-sum approximation. The modifications that were made to the material evaluation functions are shown in Listing 5.

```
1 // cameraX the same for all light source and is calculated as
2 // cameraX = normalize(cross(cameraUp, sd.V))
3 ShadingResult evalMaterial(ShadingData sd, Light light, float3 cameraX) {
4     .
5     .
14     if (gEnablePolarizingFilter) {
15         sr.specular *= polarizingFilter(sd, ls, cameraX);
16     }
17     sr.color.rgb += sr.specular;
18     .
21     return sr;
22 }
```

Listing 5. Excerpts from the modified `evalMaterial` shading function which is executed once per light source. The parameter `cameraX` represents the \hat{x} vector from Figure 6.

Although our function was implemented and tested in a forward renderer, it is fully compatible with deferred renderers as well (if the same surface and light source information is made available in the deferred renderer's lighting pass).

4.1. Visual Impact of the Polarizing Filter

The example images in Figures 1 and 7 show the impact that our polarizing filter function can have when used in the *Sun Temple* [Epic Games 2017] and *Bistro Exterior* [Amazon Lumberyard 2017] test scenes.

The top row in Figure 7 shows how a polarizing filter can make vegetation appear more green by altering the brightness of the white specular reflections from sunlight: in the middle image the filter is oriented to reduce the brightness of reflected light in the bush and the tree, while in the right image it is oriented to increase it. Notice also the significant visual impact this has on the red awning over the window.

To highlight the filter's ability to reduce, or increase, the relative brightness of reflections based on the surface orientation, a sphere made of opaque black glass was added to the scene in the bottom row of Figure 7. With the filter oriented at 122° , the reflections near the top and bottom of the sphere and in the table are reduced while reflections near the left and right sides of the sphere and the window are brightened. With the filter oriented at 0° , the effect is instead reversed, and most of the reflected light from the window is blocked while the reflection in the table is brightened.

³i.e., point and directional lights

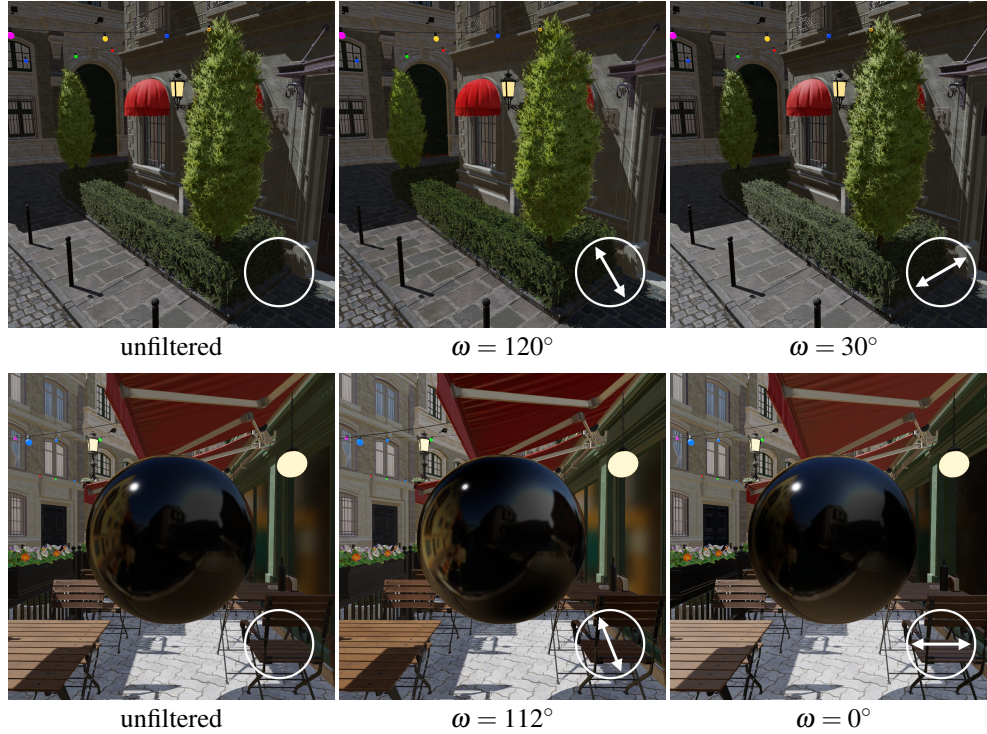


Figure 7. Two areas of the *Bistro Exterior* scene rendered without a polarizing filter (left column) and with one (middle and right columns). The white double-ended arrow represents the orientation of the polarizing filter. Notice the visual impact the filter has on the vegetation and the red awning in the top row, and on the sphere and the table in the bottom row.

The three images in [Figure 1](#) similarly demonstrate the orientation-dependent effects of the filter. Compared to the unfiltered image, the reflections in the floor (from an orange point light) become brighter and darker in the middle and right images, respectively. Likewise, reflections in the low wall on the right (from a directional white light) become darker and brighter in middle and right images, respectively.

A video demonstration of how enabling and rotating the filter affects these test scenes is included in the [supplemental material](#).

4.2. Comparison With the Exact Filter Function

In [Figure 8](#), the effect our approximate polarizing filter function Ω , from [Equation \(17\)](#), has on spheres of various materials is compared to the effect the exact IOR-based filter function Ω_{IOR} , from [Equation \(12\)](#), has on them. The rightmost column shows the RGB difference between the two functions (exaggerated by a factor of ten) and corresponds to the approximation error shown in [Figure 5](#).

The top and bottom sides of the opaque glass sphere become noticeably darkened by the filter (and the left and right side noticeably brightened). As no approximations

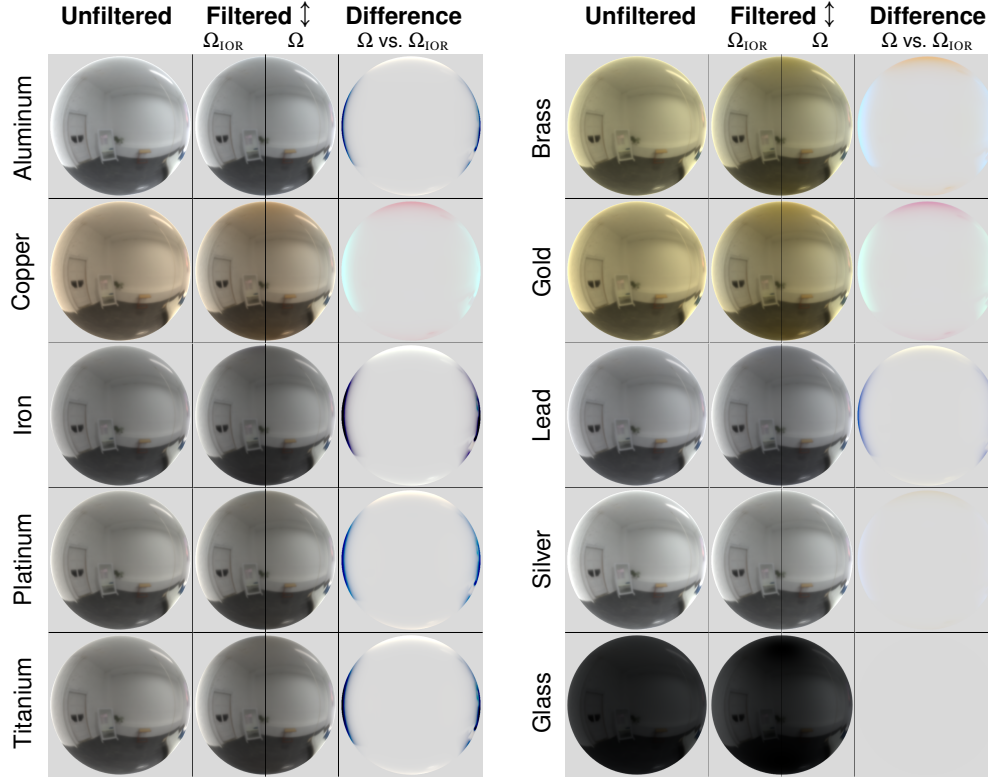


Figure 8. A comparison between the exact IOR-based polarizing filter function Ω_{IOR} and our approximate specular color-based polarizing function Ω when using a vertically aligned filter. The linear roughness is set to 0.08 for all materials and an environment map is the only light source. The exaggerated difference (rightmost column) is caused by the approximation error shown in Figure 5: it is calculated in linear color space as $\mathbf{c}_{\text{gray}} + 10 \times (\mathbf{c}_{\Omega} - \mathbf{c}_{\Omega_{\text{IOR}}})$, where \mathbf{c}_{Ω} is our approximate filtered result, and $\mathbf{c}_{\Omega_{\text{IOR}}}$ is the exact filtered result.

had to be made for ψ_D , our approximate filter function produces identical results as the exact filter function when used on glass or any other dielectric material (visualized by the completely gray difference image in the bottom right).

For conductors, the effect of a polarizing filter are much more subtle, and sometimes mostly noticeable by its color-tinting effects (e.g., compare the top and bottom of the filtered images for brass and gold with their unfiltered versions). For silver, the polarizing factor is very small for all angles of incidence, and our approximation is able to match this behavior well. However, as was also shown in Figure 5, our approximate filter function is not able to fully match the exact function for the other conductor materials. For aluminum, iron, lead, platinum, and titanium, the brightening effect (top and bottom sides) and darkening effect (left and right sides) of the filter are more significant with our approximation than with the exact filter function.

Our approximation also results in a slight colored tint for some materials when compared with the exact version. Copper and gold both produce a slightly more pink

filtered result at near-grazing angles when the plane of incidence is parallel to the filter’s alignment (top and bottom sides), and slightly more cyan results when the plane of incidence is orthogonal to the filter’s alignment (left and right sides).

4.3. Performance

Performance tests were run at a resolution of 1920×1080 px with an Nvidia GeForce RTX 2070 Super 8GB GPU and an AMD Ryzen 5 1600 CPU.

The unmodified *ForwardRenderer* sample was used as the baseline and the performance impact was calculated as the average increase in the GPU-side computation time of the lighting pass (as reported by the Falcor application). Each test recorded the execution time of 5 000 consecutive frames as the camera moved along the scene’s predefined camera path with a fixed time step. The averaged results of these measurements, as well as the average frame rates, are shown in Table 2. It should be noted that the execution time measurements are more representative of the filter’s performance impact than is the frame rate, as the impact on the frame rate strongly depends on how time-consuming the rest of the rendering computations are.

As the filter needs to be individually applied to each light source’s specular contribution, the total performance impact varies depending on how many light sources are used. Our tests found that for the filter’s per-frame performance impact was less than 0.15 ms for each additional direct light source in the scene.

Test scene	Dir. lights	IBL	Baseline		With filter		Difference	
			[FPS]	[ms]	[FPS]	[ms]	[FPS]	[ms]
<i>Bistro Exterior</i>	1	Yes	88.1	1.40	87.6	1.66	−0.5	+0.26
<i>Bistro Exterior</i>	1	No	88.9	1.09	88.4	1.19	−0.5	+0.10
<i>Sun Temple</i>	14	Yes	139.1	1.99	112.9	3.76	−26.1	+1.77
<i>Sun Temple</i>	9	Yes	151.0	1.40	130.7	2.48	−20.3	+1.08
<i>Sun Temple</i>	4	Yes	161.7	0.96	152.3	1.33	−9.4	+0.37
<i>Sun Temple</i>	1	Yes	165.9	0.77	163.8	0.87	−2.1	+0.10
<i>Sun Temple</i>	1	No	174.2	0.57	172.5	0.63	−1.7	+0.06

Table 2. Average frame rate of the application and average GPU-side execution time of the lighting pass. All tests were run without anti-aliasing and with Falcor’s “Specialize Material Shaders” option enabled.

4.4. Comparison With a Reference Polarizing Renderer

The correctness of the aforementioned filter functions was validated by implementing them in Mitsuba 2 [Nimier-David et al. 2019] and visually comparing them with a polarizing filter in Mitsuba’s Stokes-Mueller-based polarizing renderer. A comparison of the three implementations is shown in Figure 9. The scene consists of a perfectly

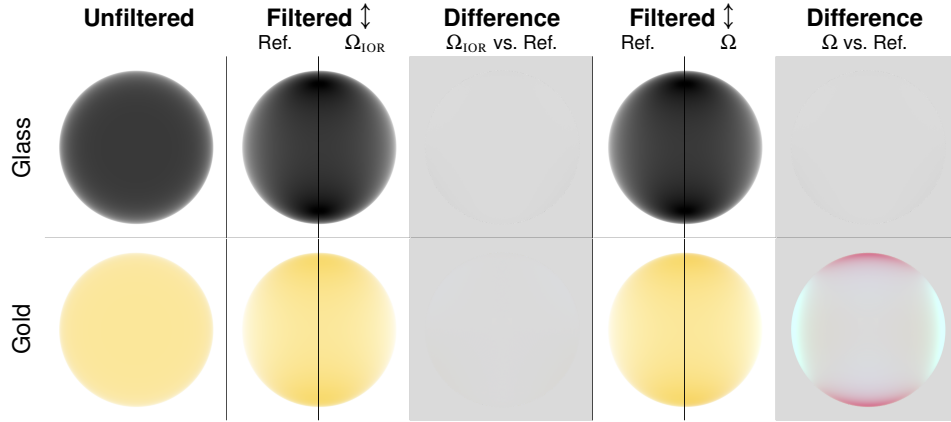


Figure 9. A comparison between a reference polarizing renderer implementation in Mitsuba 2 [Nimier-David et al. 2019] and the exact and approximate polarizing filter functions presented in this paper. The difference images are calculated as in Figure 8 with differences exaggerated by a factor of 10. The reference render had its scene brightness doubled to match the brightness of the other images. Note that the materials here use Mitsuba’s default values and not the ones specified in Section 2.2.

smooth opaque sphere surrounded by white light. In the reference renders there is also a flat polarizing filter located between the viewer and the sphere.

As is evident by the completely gray difference images in the first row of Figure 9, our exact and approximate polarizing filter implementations produce the same result as Mitsuba’s Stokes-Mueller polarization implementation when viewing a dielectric sphere through a polarizing filter. For conductors, the bottom difference image in the third column shows that our correct IOR-based filter function Ω_{IOR} matches the reference implementation, while the bottom-right difference image shows that there is a slight difference between our approximate filter implementation and the reference version (just as there was between the approximate and exact version in Figure 8).

5. Limitations

As our polarizing filter is limited by the assumptions listed in Section 3, it is unable to simulate all of the visual effects of a real polarizing filter.

Multiple-bounce reflections. We only account for the closest reflection point and assume that all incident light is unpolarized; therefore, if a scene is viewed through a mirror, then our filter function will only affect the reflection in the mirror and not any of the previous reflections in objects that are viewed in the mirror. However, as mirrors generally do not alter the polarization state of light (they just reflect it), our function could likely be adapted for use with perfect mirrors by incorporating an additional adjusted rotation angle into the calculations.

Refractions. Our function does not affect refracted light from bodies of water and other transparent surfaces. It might be possible to formulate a similar approximate polarizing filter function for refracted light; however, the assumption that all incoming light is unpolarized is not as applicable for natural light in water as it is for natural light in air. Another methodology than the one used in this paper might therefore be needed to handle refractions.

Diffuse reflections and layered materials. Our approach, just as the one used in Wolff and Kurlander’s polarization ray tracer [1990], assumes that all diffuse light is unpolarized; however, when a surface is covered with a thin layer of a material (e.g., paint on aluminum) then diffuse reflections can produce non-trivial amounts of polarization [Collin et al. 2014]. Polarimetric BRDF (pBRDF) models capable of modeling polarization effects from diffuse reflections have been proposed [Baek et al. 2018]; however, it is unclear if they could be used to develop a polarizing filter function for diffuse reflections in non-polarizing renderers with conventional BRDF models.

Similarly, our polarizing function was developed with the assumption that all surfaces consist of a single thick layer of the same material; it has not been adapted to, or tested on, any surfaces modeled with layered BRDFs.

Skylight polarization. Due to atmospheric scattering, the blue light in the daytime has a noticeable polarization pattern. As a consequence of this, polarizing filters can be used to darken blue skies in photographs. Our polarizing filter function does not model this effect; however, a skylight polarization factor ψ_s could likely be incorporated as a special skybox texture based on an analytical model of skylight polarization, such as the one by Wilkie et al. [2004].

6. Conclusion

We have presented a polarizing filter function that can be incorporated in real-time renderers, such as the ones used in games, without requiring any changes or additions to existing representations of light and surfaces. Our function is based on the Stokes-Mueller polarization calculus and can simulate the orientation-dependent brightening and darkening effects that polarizing filters have on specular reflections. In dielectric surfaces, it does so with physically-based calculations (provided the material’s specular color is accurate), and in conductor surfaces, it does so with two specular color-based approximations. We have demonstrated these effects in a real-time renderer and shown the accuracy of our approximations for some common conductor materials.

The performance impact of our method is low enough that it can be used in a real-time rendered application. Developing computationally cheaper approximations of both our dielectric and conductor functions would further improve the performance.

Although our polarizing filter function is limited to only simulating the filter's effects on specular reflections, it is likely possible to expand our model to also incorporate the filter's effects on skylight, real-time reflections in perfect mirrors, refractions, and diffuse reflections. Despite these shortcomings, our function exhibits the most recognizable effect of real polarizing filters (i.e., the selective reduction of reflected light), and we believe that it is the first implementation of a polarizing filter in any conventional non-polarizing renderer.

References

- AMAZON LUMBERYARD, 2017. Amazon Lumberyard Bistro, Open Research Content Archive (ORCA), July. URL: <http://developer.nvidia.com/orca/amazon-lumberyard-bistro>. 74
- BAEK, S.-H., JEON, D. S., TONG, X., AND KIM, M. H. 2018. Simultaneous acquisition of polarimetric SVBRDF and normals. *ACM Transactions on Graphics* 37, 6 (Dec.), 268:1–268:15. URL: <http://doi.org/10.1145/3272127.3275018>. 79
- BENNETT, J. M. 1995. Polarization. In *Handbook of Optics*, M. Bass, Ed., 2nd ed., vol. 1. McGraw-Hill, New York, NY, 5.1–5.30. 62
- BENTY, N., YAO, K.-H., FOLEY, T., OAKES, M., LAVALLE, C., AND WYMAN, C., 2019. The Falcor rendering framework, Mar. version 3.2.2. URL: <https://github.com/NVIDIAGameWorks/Falcor>. 74
- CHIPMAN, R. A. 1995. Polarimetry. In *Handbook of Optics*, M. Bass, Ed., 2nd ed., vol. 2. McGraw-Hill, New York, NY, 22.21–22.35. 60, 64
- COLLIN, C., PATTANAIK, S., LIKAMWA, P., AND BOUATOUCH, K. 2014. Computation of polarized subsurface BRDF for rendering. In *Proceedings of Graphics Interface*, GI 2014. Canadian Human-Computer Communications Society, Toronto, Ontario, Canada, 201–208. URL: <https://graphicsinterface.org/proceedings/gi2014/gi2014-26/>. 65, 79
- EPIC GAMES, 2017. Unreal Engine Sun Temple, Open Research Content Archive (ORCA), Oct. URL: <https://developer.nvidia.com/ue4-sun-temple>. 74
- GOLDSTEIN, D. H. 2011. *Polarized Light*, 3rd ed. CRC Press, Boca Raton, FL. 65
- KARIS, B. 2013. Real shading in Unreal Engine 4. In *ACM SIGGRAPH 2013 Courses*, SIGGRAPH '13. Association for Computing Machinery, New York, NY, USA, July. URL: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.372.5001>. 66, 69, 74
- NIMIER-DAVID, M., VICINI, D., ZELTNER, T., AND JAKOB, W. 2019. Mitsuba 2: A retargetable forward and inverse renderer. *ACM Transactions on Graphics* 38, 6 (Nov.), 203:1–203:17. URL: <https://doi.org/10.1145/3355089.3356498>. 60, 77, 78
- POLYANSKIY, M. N., (n.d.). Refractive index database. URL: <https://refractiveindex.info>. 62

- SCHLICK, C. 1994. An inexpensive BRDF model for physically-based rendering. *Computer Graphics Forum* 13, 3 (Nov.), 233–246. URL: <https://doi.org/10.1111/1467-8659.1330233>. 63
- SHURCLIFF, W. A. 1962. *Polarized Light: Production and Use*. Harvard University Press, Cambridge, MA. 63
- VLKER, M., AND HAMANN, B. 2013. Real-time rendering of cut diamonds. Technical report, University of California, Davis, CA. URL: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.409.2851>. 60
- WEIDLICH, A., AND WILKIE, A. 2008. Realistic rendering of birefringency in uniaxial crystals. *ACM Transactions on Graphics* 27, 1 (Mar.), 6:1–6:12. URL: <https://doi.org/10.1145/1330511.1330517>. 60
- WILKIE, A., AND WEIDLICH, A. 2012. Polarised light in computer graphics. In *SIGGRAPH Asia 2012 Courses*, SA '12. Association for Computing Machinery, New York, NY, USA, Nov. URL: <https://doi.org/10.1145/2407783.2407791>. 61, 63, 65
- WILKIE, A., ULBRICHT, C., TOBLER, R. F., ZOTTI, G., AND PURGATHOFER, W. 2004. An analytical model for skylight polarisation. In *Eurographics Workshop on Rendering*, A. Keller and H. W. Jensen, Eds., EGSR'04. Eurographics Association, Aire-la-Ville, Switzerland, 387397. URL: <http://doi.org/10.2312/EGWR/EGSR04/387-397>. 79
- WOLFF, L. B., AND KURLANDER, D. J. 1990. Ray tracing with polarization parameters. *IEEE Computer Graphics and Applications* 10, 6 (Nov.), 44–55. URL: <https://doi.org/10.1109/38.62695>. 60, 65, 79

Index of Supplemental Materials

- <http://jcgt.org/published/0010/02/03/PolarizingFilterFunctions.hlsl>: Shader source code of the functions needed to implement the polarizing filter. The complete source code of the demo application is available at: <https://github.com/viktor4006094/PolarizingFilter>.
- <http://jcgt.org/published/0010/02/03/DemoVideo.mp4>: A video showing what using and rotating the filter looks like.
- <http://jcgt.org/published/0010/02/03/jpegScreenshots.zip>: Rendered spheres of various materials and full rendered scenes with varying polarization.

Author Contact Information

Viktor Enfeldt
Blekinge Institute of Technology
Department of Computer Science
SE-371 79 Karlskrona, Sweden
viktor.enfeldt@gmail.com

Prashant Goswami
Blekinge Institute of Technology
Department of Computer Science
SE-371 79 Karlskrona, Sweden
prashant.goswami@bth.se

Viktor Enfeldt and Prashant Goswami, A Polarizing Filter Function for Real-Time Rendering, *Journal of Computer Graphics Techniques (JCGT)*, vol. 10, no. 2, 59–82, 2021
<http://jcgt.org/published/0010/02/03/>

Received: 2020-10-06

Recommended: 2020-11-30

Published: 2021-06-17

Corresponding Editor: Alexander Wilkie

Editor-in-Chief: Marc Olano

© 2021 Viktor Enfeldt and Prashant Goswami (the Authors).

The Authors provide this document (the Work) under the Creative Commons CC BY-ND 3.0 license available online at <http://creativecommons.org/licenses/by-nd/3.0/>. The Authors further grant permission for reuse of images and text from the first page of the Work, provided that the reuse is for the purpose of promoting and/or summarizing the Work in scholarly venues and that any reuse is accompanied by a scientific citation to the Work.

