# Realistic image synthesis in presence of birefrigent media by backward ray tracing technique

Dmitry Zhdanov, Sergey Ershov, Lev Shapiro, Vadim Sokolov, Alexey Voloboy, et al.

Dmitry Zhdanov, Sergey Ershov, Lev Shapiro, Vadim Sokolov, Alexey Voloboy, Vladimir Galaktionov, Igor Potemin, "Realistic image synthesis in presence of birefrigent media by backward ray tracing technique," Proc. SPIE 10694, Computational Optics II, 106940D (28 May 2018); doi: 10.1117/12.2312675

SPIE.

Event: SPIE Optical Systems Design, 2018, Frankfurt, Germany

# Realistic image synthesis in presence of birefrigent media by backward ray tracing technique

Dmitry Zhdanov[a,b], Sergey Ershov[b], Leo Shapiro[b], Vadim Sokolov[b], Alexey Voloboy[b], Vladimir Galaktionov[b], and Igor Potemin[a,b]

[a]ITMO University, 49 Kronverksky Pr., St. Petersburg, 197101, Russia
[b]Keldysh Institute of Applied Mathematics, Russian Academy of Sciences, 125047 Miusskaya sq. 4, Moscow, Russia

## ABSTRACT

We describe an algorithm of tracing a backward (from camera) ray in a scene which contains birefrigent (uniaxial) media.

The physics of scattering of an electromagnetic wave by a boundary between two media is well known and is a base for ray tracing algorithms; but processing of a *backward* ray differs from scattering of a "natural" forward ray. Say, when a backward ray refracts by a boundary, besides the *energy* transfer coefficient like for a forward ray one must account for the luminance change due to *beam* divergence. We calculate this factor and prove it must be evaluated only for the first and the last media along the ray path while the contributions from the intermediate media mutually cancel.

In this paper we present a closed numerical method that allows to perform transformation of a *backward* ray on a boundary between two media either of which can be birefrigent. We hope it is more convenient and ready for usage in ray tracing engines that known publications.

Calculation utilizes Helmholtz reciprocity to calculate directions of scattered rays and their polarization (i.e. Mueller matrices) which is advantageous over a straightforward "reverse" of forward ray transformation.

The algorithm was integrated in the lighting simulation system *Lumicept* and allowed for an efficient calculation of images of scenes with crystal elements.

**Keywords:** Uniaxial crystal, backward ray tracing, polarization, reciprocity.

## NOTATIONS

$\boldsymbol{n}$ is the local normal (to a boundary between two media etc).

$\boldsymbol{a}$ is the optical axis; in case the medium is isotropic, any unit vector can be used.

$\eta_o, \eta_e$ ordinary and extraordinary refraction index

$\rho \equiv \eta_o/\eta_e$

$\lambda$ wavelength

$k_o$ ordinary wave number (unlike that of an extraordinary wave, it is independent of direction so is used as a property of the medium).

$\boldsymbol{s}$ is direction of the ray

**BRT** backward ray tracing, when the ray is emitted from a camera and then traverses scene, gathering luminance in the hit points; can be deterministic or stochastic (usually Monte-Carlo)

Send correspondence to Dmitry Zhdanov: E-mail: ddzhdanov@mail.ru

Computational Optics II, edited by Daniel G. Smith, Frank Wyrowski, Andreas Erdmann, Proc. of SPIE
Vol. 10694, 106940D · © 2018 SPIE · CCC code: 0277-786X/18/$18 · doi: 10.1117/12.2312675

**FRT** forward ray tracing, when the ray is emitted from a light source and then traverses scene, bringing illumination to the hit points; can be deterministic (very exotic, though) or stochastic (usually Monte-Carlo)

Subscripts "o" and "e" relate to the ordinary and extraordinary waves/rays.

## 1. INTRODUCTION

A powerful tool in computer graphics or optics simulation is *ray tracing* which is in fact a variety of rather different methods. There is a *stochastic ray tracing* (a Monte-Carlo or Metropolis-based etc) and there is a *deterministic* ray tracing. Also there is a *backward* ray tracing, when a ray emits from the *camera*, a *forward* ray tracing, when it is emitted from a light source. In principle, all the four combinations are possible, though a deterministic forward ray tracing is too exotic. The rest three are used rather widely; also there are various combinations of them.

Although *in nature* the eye does not really emit rays to sense environment, it does not mean that only forward ray tracing is physically correct. In fact all kinds of ray tracing are only *mathematical means* to solution of a self-consistent "global illumination problem", which are in a sense similar to solution of PDEs or integral equations with "stochastic trajectories". One can then find a solution in a single desired space-time point, emitting from it an ensemble of "trajectories" which go *to the past* until reach the "initial manifold" where they take (and "transfer" to the target point) the initial conditions. Or one can start those paths *from* the initial manifold to the "future", and when they reach the target time slice, they contribute to the accumulated averages. And both approaches are mathematically correct.

More or less the same is true for ray tracing. In the forward ray tracing, when rays are fired from the light source, they very rarely reach the "virtual eye" (or camera), and thus a "virtual photography" is badly inefficient. But if one wishes to calculate illuminance of a room walls by collecting all rays that hit it, that works well. The backward ray tracing, meanwhile, is good just for the virtual imaging. Since *all* paths "reach" the camera, the process is highly efficient. Deterministic backward ray tracing provides absolutely noise-free extremely detailed images.

Surely, the backward ray tracing is not an ideal tool; it has problems "on the opposite end" of the path. If they are stochastic, they hit small lights badly rarely. If they are deterministic, they do not provide an efficient convolution with a diffuse BDF.

Therefore, usually clean high-definition noise free images are calculated with a hybrid technique. First a stochastic forward ray tracing is performed which provides secondary illumination of the diffuse surfaces. Usually it is rather smooth and admits some kind of filtering which eliminates most of the noise. Then the backward pass is performed; now the rays are fired from the camera until they hit a diffuse surface (for the first time); there they take its luminance for both direct and indirect (secondary) illumination and "transfer" it to the camera image.

Before that first diffuse surface the camera ray may, however, face specular surfaces, e.g. window glasses, water surface etc. It is therefore necessary to simulate how a *backward* ray interacts with specular surface: how its direction changes and how the diffuse surface luminance changes when observed through a specular surface.

For isotropic media, such as glass, air, water etc. and in absence of polarization the solution is quite simple and more or less intuitively obvious. If polarization of light is taken into account, the more so in presence of anisotropic media, it sophisticates. One must realize that a backward ray must be treated differently than a forward one, because it is not simple a reversed direction but a different object. Suffice to say that for a forward ray we know its polarization parameters (Stokes vector) after emission; but a backward ray emitted from camera may *not* have it!

Such specular surface may also include anisotropic media e.g. uniaxial crystals. Such are gems, and here the "virtual imaging" is widely used for the, first, estimation of the raw stones, i.e. "looking inside" of them to detect inner defects and find the optimal cutting directions.[12] Second, to find the best-looking faceting for the given environment. Third, just for virtual presentation of the gems.[6]

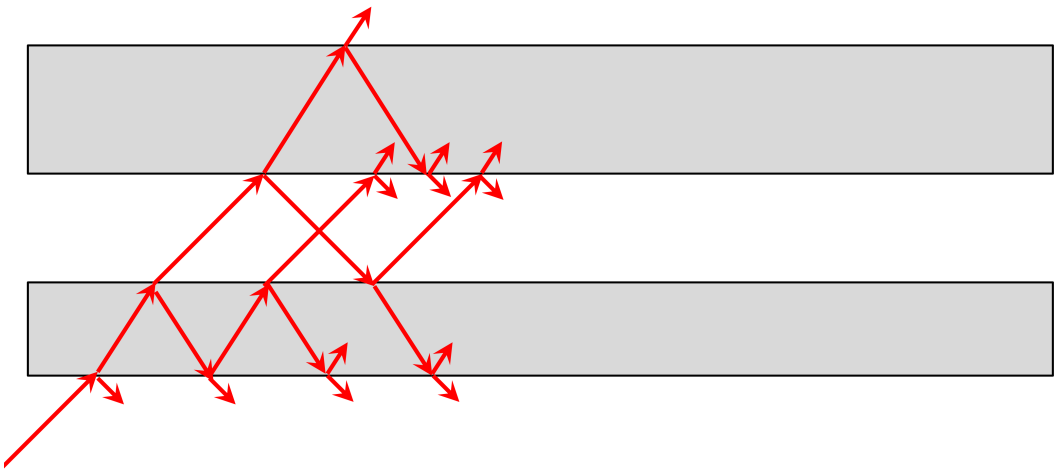

Figure 1. Backward deterministic ray tracing through specular surfaces. Shown are two glass plates and 4 first scattering order in them.

Besides these nice-looking expensive gems, there are also liquid crystals and polarization filters. The modern luminaires for e.g. flat display are sometimes sensitive to some subtle polarization effects, either utilizing them or suffering from them.[14]

In this paper we describe the algorithms used for calculation of interaction of a backward ray with a boundary between two media, one or both of which can be birefrigent. Although the physics[4,5,7] they are based upon is known for a long while, it can result in *different* numerical methods, which differ in both efficiency and so to say "readability". We make stress on the specifics of the backward ray and the possibility to "re-use" or extend the methods for forward rays. At last we attempted to provide a method that requires as little changes as possible for the extension the existing ray tracing simulator *Lumicept* as easily as possible. All what we needed was, first, replacement of the scalar ray color with a Mueller matrix for the backward ray and Stokes vector for the forward ray, and, second, some new mathematics under the same interface for birefrigent media.

The forward ray tracing method are used in the form described in Ref. 2,3; this approach is most similar to Ref. 13 and Ref. 11, differing from[8,9] in that no iterative calculations are used. Unlike Ref. 13, we include light incidence from anisotropic material thus being close to Ref. 11, but still operate Stokes formalism. As to the backward ray tracing, we are close to Ref. 11, but differ in the treatment of the "luminance correction factors" (change of a beam luminance after transmitting a specular surface), and also the algorithm details.

## 2. COMMON BACKWARD DETERMINISTIC RAY TRACING

Some scene surfaces can be seen (by camera) not directly, but only "through" specular surfaces. The simplest example is window glass, but also it can be a mirror, a prism, or even a lens. In these cases the camera ray has a specular path.

Most of specular surfaces can both reflect and refract ray. In case of probabilistic ray tracing, we always choose *one* event, but this is impossible for a deterministic ray tracing which must explore all possible ray paths one by one. In the simplest case of e.g. a glass surface there are two such transformed rays: one reflected and one transmitted. But if there is a *dispersion*, there are several refracted rays (one per wavelength channel).

The observed luminance in this case is a sum of luminance seen along these transformed rays; e.g. when we look from inside a room through a window glass, we see a sum of *reflection* of the room interior and *transmission* of the outdoor scene.

The specularly transformed ray (e.g. transmitted through window glass) may then hit another specular surface (e.g. the next window glass sheet), and "split" there further, as shown in Figure 1.

The usual technique of calculation of the luminance observed through this "tree of rays" is recursion. We start from the "tree root" i.e. the camera, and calculate luminance as a sum over all specular paths in the first

specular hit point. The luminance for that paths is in turn calculated as a sum over the ray *they* split in in the next hit point, and so on.

That is, in each hit point we call the recursive function like explained in Algorithm 1.

**Algorithm 1** Simplified pseudocode of calculation of luminance in the hit point

```
Color3d CalcRayLum(const Ray3d &ray)
{
  hit = TraceToSurface(ray);
  Color3d lum = hit.bdf * illumination;
  if (hit is specular)
  {
    TArray<Ray3d> newray;
    TArray<Mueller> newmueller;
    CalcOutRays(ray, newray, newmueller);
    for (i = 0; ...)
      lum += mueller[i] * CalcRayLum(newray[i]);
  }
  return lum;
}
```

This algorithm originated when there were no polarization and specular surfaces were either ideal mirrors or Fresnel boundaries of isotropic dielectrics. What, if any, must be changed when there is polarization and also anisotropic refraction?

## 3. BACKWARD RAY TRACING IN PRESENCE OF ANISOTROPIC MEDIA (AND POLARIZATION)

### 3.1 Transformation of backward ray

Let us begin with directions of rays and then come to their "amplitudes".

The usual approach in physics is that we consider a wave incident onto the surface and calculate directions and fields in the outgoing waves. In case of anisotropic refraction index, even their directions can depend on the *direction of field* in the incident wave.[4,7] Obviously, there is no field in a *backward* ray (suffice it to imagine which *field* it has just after emission from camera?!).

Therefore, while the transformation of a *forward* wave (or ray) is calculated straightforwardly from the basic physics, it can not be so for a *backward* ray. For the latter we must solve another problem: *find such a forward ray that the refracted ray for it coincides with the (negated) backward ray.* This would give us direction of the "refracted camera ray" (although one must realize it is merely a metaphor). The same is done for *reflection.*

For isotropic dielectrics, i.e. the Snell law, finding "direction that refracts into the camera ray" is very simple: the inverse Snell law is just the same function just for reversed refraction index. The (mirror) reflection direction is even simpler.

For a **boundary between two anisotropic media** the situation is rather similar. In that medium the *forward* ray (we search for) can be either *ordinary*, or *extraordinary*, or a *mixture* of both (when it is a sum of an ordinary and extraordinary *waves* with the same *ray* direction). Generally, the ordinary and extraordinary components refract (or reflect) into *different* directions, one for an ordinary and one for an extraordinary ray, so only *one* of them can match the backward ray, see Appendix A.

There are therefore $2 \times 2 = 4$ combinations, as shown in Table 1:

where, say, "$o \to e$" means that the forward ray is ***o***rdinary and the camera ray direction is matched by the ***e***xtraordinary ray. The direction of the backward ray can thus be matched by *four* forward directions.

Table 1. Possible combinations of polarization states for the initial and transformed *forward* ray.

| | |
|---|---|
| $o \to o$ | $e \to o$ |
| $o \to e$ | $e \to e$ |

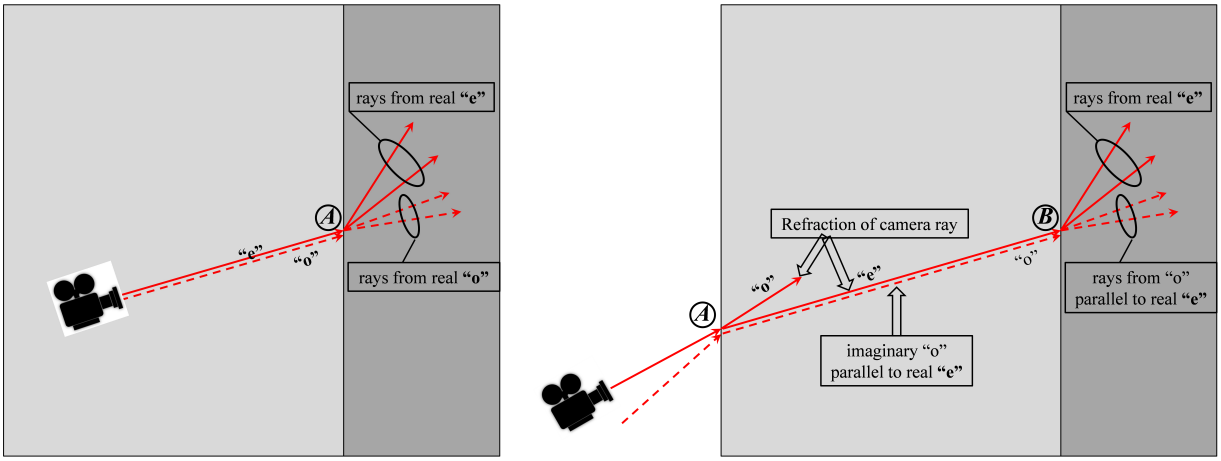


Figure 2. Reduction of ray splitting when camera is in air; two birefrigent media are shaded grey. **Left**: camera inside birefrigent medium; its ray initially does not have polarization type (=both) and thus refracts into *four* rays. **Right**: camera is in air; after entering a birefrigent medium its ray *has* a definite polarization type. So it refracts into *two* rays; two rays from *another* polarization type parallel to the camera ray are discarded because if traced *back* they would not return to camera.

A *mixture* forward ray splits into 4 generally different directions after refraction, so only *one* of them matches the backward ray. We then have the same four combinations as listed in Table 1, i.e. again there are the same four possible direction of the forward ray.

Notice that anyway the forward ray splits into several refracted rays, only one of which matches the backward ray direction. That is, when the found "real" i.e. forward ray hits the boundary, it splits into several ones of which all but one do *not* go towards camera. Therefore they are not used in *this* calculation (i.e. in this hit point).

We conclude that on a boundary between two anisotropic media there are 8 (four refracted and four reflected) forward rays that scatter into the (reversed) backward ray direction. The "ray tree" therefore grows very quickly as compared to the isotropic case, when the backward ray splits into only two. With just 3 crystal plates i.e. 6 boundaries we would need to explore $8^6 = 262144$ paths for each camera ray i.e. each pixel. This is very expensive, but happily this number can be usually much reduced.

Indeed, usually the camera in an *isotropic* medium (air or alike), see Figure 2(right). Then camera ray hits a crystal boundary for the first time in the point $A$, and here we have only 2 (not 4) forward ray (from the crystal) which would refract into it. This is because in air the variants "$o \to o$" and "$o \to e$ " , "$e \to e$" and "$e \to o$" coincide. According to Algorithm 1, we explore both starting e.g. with the "ordinary" forward ray. We trace it to the next boundary i.e. point $B$, and there we would apply the splitting procedure which yields 4 forward rays that would refract into the it (and 4 which would reflect). But of these 4 variants two are such that the forward ray after refraction becomes *extraordinary*. They have proper direction, but after reaching the point $A$ they would refract into direction different from camera ray, see Figure 2(right):

Therefore, in the point $B$ we must retain only *two* (of 4) forward ray directions: those which refract into the ray whose polarization type (ordinary or extraordinary) matches that of the backward ray. The same is true for reflection; so the backward ray actually splits then into "only" four, not eight, "predecessors" and the ray tree grows slower.

Should we follow Algorithm 1 formally, we would have 4 refracted rays but their Mueller matrices, when multiplied by current one, give 0 (because of the wrong polarization state). So this is merely an optimization, not a principal change of the method.

Similarly, if a (camera) ray goes from an anisotropic medium into an isotropic one, there are twice less refracted rays. If we know polarization state of the initial ray, there is a single refracted ray (in the isotropic medium). If polarization state of the initial ray is unknown (or is a mixture), then there are two refracted rays.

### 3.2 Transfer factors

These are the values the ray *energy* (or its "polarized" form = Stokes vector) "before" the surface must be multiplied by to get the one beyond it. In the depolarized isotropic case this is the simple scalar of reflection or refraction coefficient. In a polarized case it becomes a Mueller matrix.

These factors are calculated from the Fresnel coefficients, which in turn are the amplitudes of the field in the reflection (or refraction) of the forward wave with unit field.[3] So, formally, we should need to compute them for the "forward problem" for the four transformed rays and the calculation becomes fourfold longer than in the forward ray tracing. Naturally, we retain only the coefficients which relate to the refraction (or reflection) directions that match the backward ray; the rest are discarded (the same process as with the ray directions).

Happily, this "brute force" approach is not needed and we can reduce calculations amount utilizing Helmholtz reciprocity.[10] This is like when calculating directions of the transformed backward ray where we apply the *forward* ray transform law to the *backward* ray direction.

Here we also calculate the transfer factors, i.e. the four Mueller matrices, using the *forward* formulae.[3] Unlike the depolarized case, here the transfer factors for the backward ray are not just the same; they relate as

$$\begin{aligned} M^{(B)}_{i,j} &= M^{(F)}_{j,i} \times \xi_i \xi_j \\ \xi_k &\equiv \begin{cases} 1, & k \neq 3 \\ -1, & k = 3 \end{cases} \end{aligned} \tag{1}$$

(we number Stokes vector components 0 thru 3). That is, the matrix is transposed and then some its elements are negated. This relation originated from another light scattering problem[1] but apparently it also is valid for a Fresnel boundary. One must be anyway cautious with reciprocity in polarized case: it is not trivial, not even not all cases were proven, see e.g. a comprehensive study in.[10]

Therefore, we actually can calculate all the four transfer factors at once, applying the *forward* law to the (reversed) *backward* ray.

The Algorithm 1 just follows this way and it calculates them *together* with the ray directions. Meanwhile, they can be separated, i.e. we can calculate the ray directions first and transfer factors after that (with no overhead). Which is more advantageous?

#### 3.2.1 Transfer coefficients before tracing

Here we utilize reciprocity and thus calculate transfer factors for all the four reflected and refracted rays at once.

#### 3.2.2 Transfer coefficients after tracing

Here we trace the transformed rays and get their luminance and only after that calculate transfer factors for those directions which *did bring* luminance. Indeed, some rays may be absorbed or go away, so actually we do not need transfer factor for them.

The benefit is therefore achieved if we calculate transfer factors for some of the transformed (forward) rays only. This is really possible; but one can calculate the Fresnel coefficients only for *all* the rays this forward ray refracts and reflects into. In our case there are 4 of them, while only one of them (which corresponds to the backward ray direction) is used.

So if only one of the transformed rays brings luminance (the rest being "lost"), we calculate four transfer factors. This, however, is exactly the same amount of calculations as when we use reciprocity and get all the four transfer coefficients before tracing the rays. If two transformed rays bring luminance, we calculate transfer factors for both of them, and this is already *twice more expensive* than in the previous method with reciprocity.

Therefore, it is advantageous to calculate the Mueller matrices for the four outgoing rays *before* tracing them through the scene. We can combine it with calculation of the directions of rays in one call (of a routine for the *forward* ray transform!), as shown in Algorithm 1.

### 3.3 Selection of outgoing rays using Mueller matrices

As said above, frequently (it is always when the camera is in air) the backward ray transforms into four, not eight rays. This is because we need only those of them which reflect (or refract) into a ray with polarization type matching that of the backward ray. In other words, when calculating transform of the backward ray we take *one* row of the Table 1.

To do so we need to know polarization type of the ray (ordinary, extraordinary or mixed — in the latter case both rows are processed). In principle the ray data structure may hold this flag, but happily even that is not needed and the ray data used in isotropic case is enough.

This latter contains the Mueller matrix (instead of the scalar "ray color" for depolarized case), and it is enough. Our main recursive function from Algorithm 2 then receives it as an argument:

**Algorithm 2** Simplified pseudocode of calculation of luminance in the hit point

```
Color3d CalcRayLum(const Ray3d &ray, const Mueller &mueller)
{
  hit = TraceToSurface(ray);
  Color3d lum = hit.bdf * illumination;
  if (hit is specular)
  {
    TArray<Ray3d> newray;
    TArray<Mueller> newmueller;
    CalcOutRays(ray, mueller, newray, newmueller);
    for (i = 0;...)
      lum += mueller[i] * CalcRayLum(newray[i], newmueller[i]);
  }
  return lum;
}
```

The method `CalcOutRays()` now knows the Mueller matrix of the initial ray, and it retains only those transformed rays whose polarization state is not "killed" by that matrix. To do so, we first check the action of that Mueller matrix on the polarization state for the ordinary ray (in the medium the ray comes from!), then on that for the extraordinary one. If the first result is not 0, the backward ray contained the ordinary component; if the second is not 0, it contained the extraordinary component; and if both are not 0, it contains a mixture of both components. Correspondingly, when calculating ray directions we then explore the top, bottom or both rows of the Table 1, respectively.

## 4. LUMINANCE THROUGH A SPECULAR BOUNDARY

The Algorithm 1 (or 2) works correctly in case camera and luminance source (diffuse surfaces, SEO etc) are in the same media. That is, the ends of ray trajectory must be in the media with the same refraction indices.

In case refraction in the ray ends is different, the luminance calculated with this method would wrong. This effect is well known even for a depolarized isotropic case: the luminance (or radiance) if observed from a Fresnel boundary scales by squared refraction index times. This is because a light beam changes its angular divergence $d^2\omega$ when going through a refractive boundary.

So, in depolarized isotropic case the ray luminance calculated in Algorithm 2 must have been scaled by the squared relative refraction index. If the ray starts in medium 0 and then enters medium 1, then medium 2, ..., and eventually takes the diffuse surface luminance when in medium $N$, then at the $i$-th boundary the luminance must have been multiplied by $(\eta_i/\eta_{i+1})^2$ where $\eta_i$ is refraction index of the $i$-th medium. For the full ray path this totals to

$$\left(\frac{\eta_0}{\eta_1}\right)^2 \left(\frac{\eta_1}{\eta_2}\right)^2 \times \cdots \times \left(\frac{\eta_{N-1}}{\eta_N}\right)^2 = \left(\frac{\eta_0}{\eta_N}\right)^2$$

Therefore, only the first and the last refraction index are included, the intermediate ones mutually cancel. In deterministic backward ray tracing one thus can apply the own $(\eta_i/\eta_{i+1})^2$ factor for each refraction event, or use the ratio of the first and the last only. Usually the last way is used.

Mirror reflection by a Fresnel boundary between two isotropic media does not include any luminance scale (i.e. this factor equals 1).

This easily extends to an anisotropic media.

Let a beam with luminance $L_{in}$ and direction $d\boldsymbol{s}_{in}$ and divergence $d^2\boldsymbol{s}_{in}$ goes from a medium with refraction $(\eta_{o,in}, \eta_{e,in})$ into that with refraction $(\eta_{o,out}, \eta_{e,out})$ where it has direction $\boldsymbol{s}_{out}$. The power flow incident on the surface the incident beam is

$$dP_{in} = L_{in} \left|(\boldsymbol{s}_{in} \cdot \boldsymbol{n})\right| d^2\boldsymbol{s}_{in}$$

The power flow which goes away the surface in the transmitted beam is similarly

$$dP_{out} = L_{out} \left|(\boldsymbol{s}_{out} \cdot \boldsymbol{n})\right| d^2\boldsymbol{s}_{out}$$

The "transfer coefficient" $t$ is just the fraction of the transmitted *energy*, so

$$dP_{out} = t dP_{in}$$

Combining, we have

$$L_{out} = t \frac{\left|(\boldsymbol{s}_{in} \cdot \boldsymbol{n})\right|}{\left|(\boldsymbol{s}_{out} \cdot \boldsymbol{n})\right|} \frac{d^2\boldsymbol{s}_{in}}{d^2\boldsymbol{s}_{out}} \times L_{in}$$

That is, the *luminance* (or radiance) transfer coefficient is

$$T = t \frac{\left|(\boldsymbol{s}_{in} \cdot n)\right|}{\left|(\boldsymbol{s}_{out} \cdot n)\right|} \Big/ \left|\frac{\partial(\boldsymbol{s}_{out})}{\partial(\boldsymbol{s}_{in})}\right|$$

where $\left|\frac{\partial(\boldsymbol{s}_{out})}{\partial(\boldsymbol{s}_{in})}\right|$ is the Jacobian of the ray direction transform $\boldsymbol{s}_{in} \mapsto \boldsymbol{s}_{out}$. Notice here the "in" and "out" relate to the *forward* beam direction (i.e. from light to camera).

The "polarized luminance" is just the usual depolarized (scalar) luminance times normalized Stokes vector of the beam and the "energy transfer coefficient" $t$ is a Mueller matrix. From the above definition we see that the "luminance correction factor" is scalar:

$$\frac{\left|(\boldsymbol{s}_{in} \cdot \boldsymbol{n})\right|}{\left|(\boldsymbol{s}_{out} \cdot \boldsymbol{n})\right|} \Big/ \left|\frac{\partial(\boldsymbol{s}_{out})}{\partial(\boldsymbol{s}_{in})}\right|$$

Substituting (8) for $\left|\frac{\partial(\boldsymbol{s}_{out})}{\partial(\boldsymbol{s}_{in})}\right|$ we see that the luminance correction factor for a single boundary is

$$\left(\frac{\eta_{o;out}}{\eta_{o;in}}\right)^2 \left(\frac{1 + (\rho_{in}^2 - 1)(\boldsymbol{s}_{in} \cdot \boldsymbol{a}_{in})^2}{1 + (\rho_{out}^2 - 1)(\boldsymbol{s}_{out} \cdot \boldsymbol{a}_{out})^2}\right)^2$$

which covers all polarization state combinations if for an ordinary ray $\rho := 1$.

This value has the same remarkable property as in the isotropic case: this factor depends only on the properties of the medium (refraction and optical axis), not on the geometry of ray hit, i.e. not on the normal. Therefore for the whole camera ray path (from medium 0 to medium $N$) the product of these factors:

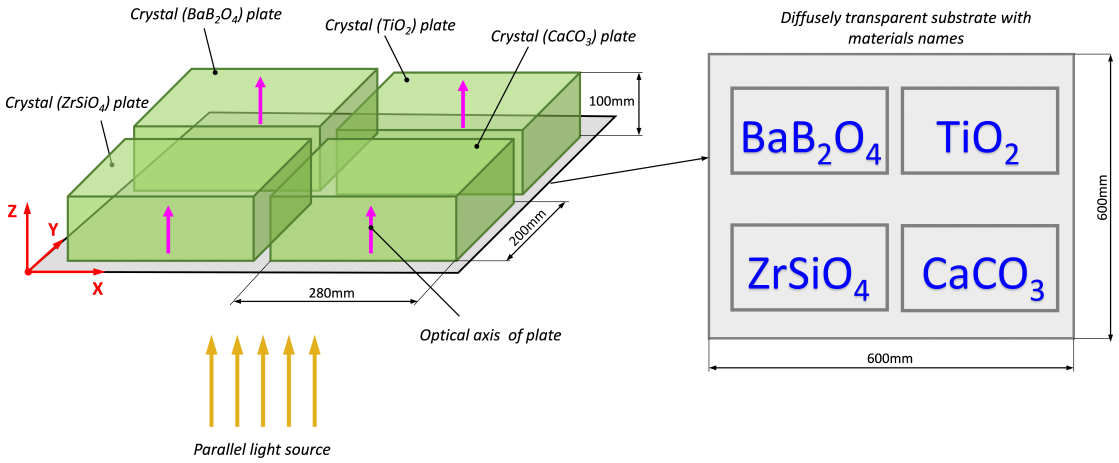


Figure 3. Example scene of four crystal plates laid on a paper sheet.

Table 2. Materials of the crystal plates.

| Plate name | Ordinary refraction | Extraordinary refraction |
|---|---|---|
| $BaB_2O_4$ | 1.6776 | 1.5534 |
| $TiO_2$ | 2.616 | 2.903 |
| $CaCO_3$ | 1.96 | 2.015 |
| $ZrSiO_4$ | 1.658 | 1.486 |

$$\prod_{i=0}^{N}\left(\frac{\eta_{o;i}}{\eta_{o;i+1}}\right)^2\left(\frac{1+(\rho_{i+1}^2-1)(\boldsymbol{s}_{i+1}\cdot\boldsymbol{a}_{i+1})^2}{1+(\rho_i^2-1)(\boldsymbol{s}_i\cdot\boldsymbol{a}_i)^2}\right)^2=\left(\frac{\eta_{o;0}}{\eta_{o;N}}\right)^2\left(\frac{1+(\rho_N^2-1)(\boldsymbol{s}_N\cdot\boldsymbol{a}_N)^2}{1+(\rho_0^2-1)(\boldsymbol{s}_0\cdot\boldsymbol{a}_0)^2}\right)^2$$

the intermediate terms mutually cancel and only those for the first (camera's) and the last (light's) media remain.

## 5. SIMULATION EXAMPLE AND RESULTS

Both benchmarks below were calculated with deterministic backward ray tracing using *Lumicept* simulator, extended with the support of birefrigent media.

The first scene imitates a well-known demonstration of "image doubling" when looking through a crystal plate. There are four crystal plates of various substances laid on a textured paper sheet with the names of the substances printed beneath each plate, illuminated from below. All plates have size 280x200x100 mm and optical axis orthogonal to the paper sheet they lay upon. Geometry of the scene is shown in Figure 3 and refraction indices are presented in Table 2. Camera is in the point (−722.7821, −2390.363, 2815.808) mm, the view direction (for the image centre) being (0.18914, 0.63457, −0.74936).

The simulated image is shown in Figure 4:

The second test scene reproduces the one used as benchmark in Ref. 11. It consists of an irregular hexahedron plate laid on a paper sheet with a chess-board-like texture, as shown in Figure 5. The plate's ordinary refraction index is 1.7 and the extraordinary one is 1.85. The optical axis is $(-0.15213, 0.37939, -0.91264)$ in the coordinate system of Figure 5. Coordinates of its vertices are listed in Table 3. Camera is in the point (57.187, 67.23533, 321.254) mm, the view direction (for the image centre) being (−0.074335, −0.088964, −0.99326).

The simulated image is shown in Figure 6.

## ACKNOWLEDGMENTS

The work was partially supported by RFBR Grants No. 16-01-00552 and 18-01-00569.

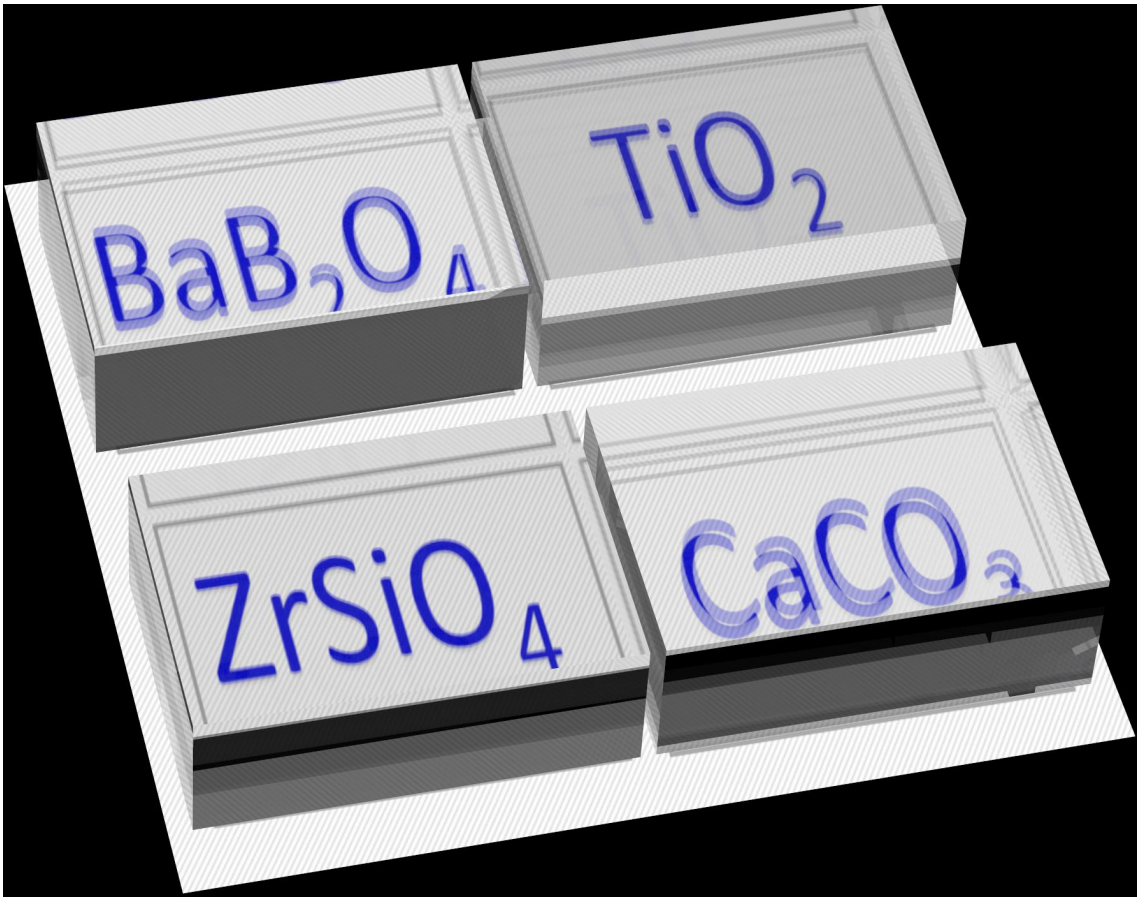


Figure 4. Simulated camera image of the scene from Figure 3.

Table 3. Coordinates of the plate vertices.

| **Vertex** | $x$ | $y$ | $z$ |
|---|---|---|---|
| 1 | 57.32966 | 18.31204 | 0 |
| 2 | 63.71413 | 46.35459 | 0 |
| 3 | 23.622 | 53.04367 | 0 |
| 4 | 14.82445 | 25.12837 | 0 |
| 5 | 47.94061 | 27.4328 | 16.91231 |
| 6 | 54.34407 | 55.74911 | 16.97624 |
| 7 | 14.22843 | 61.34301 | 14.59365 |
| 8 | 5.493791 | 33.10582 | 14.39393 |

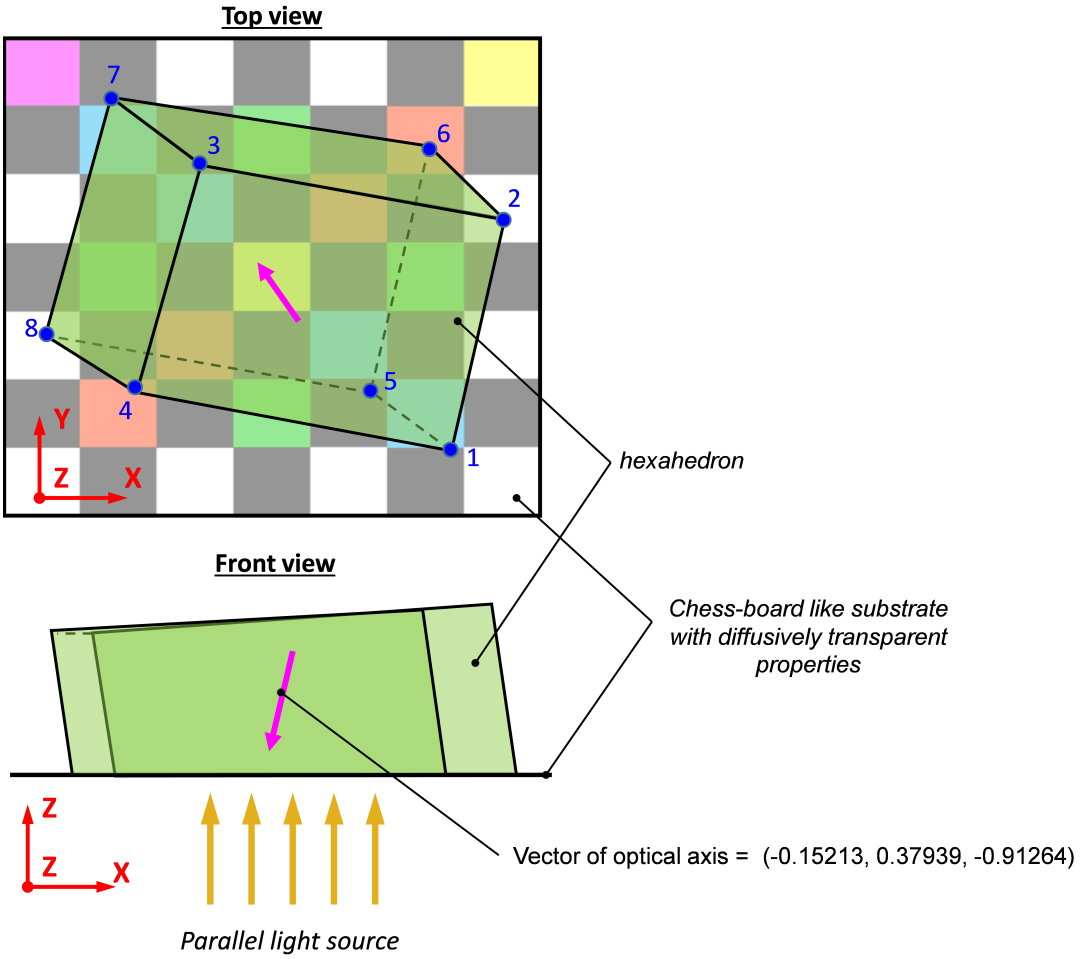


Figure 5. Example scene of a crystal prism laid on a paper sheet with chessboard-like texture.

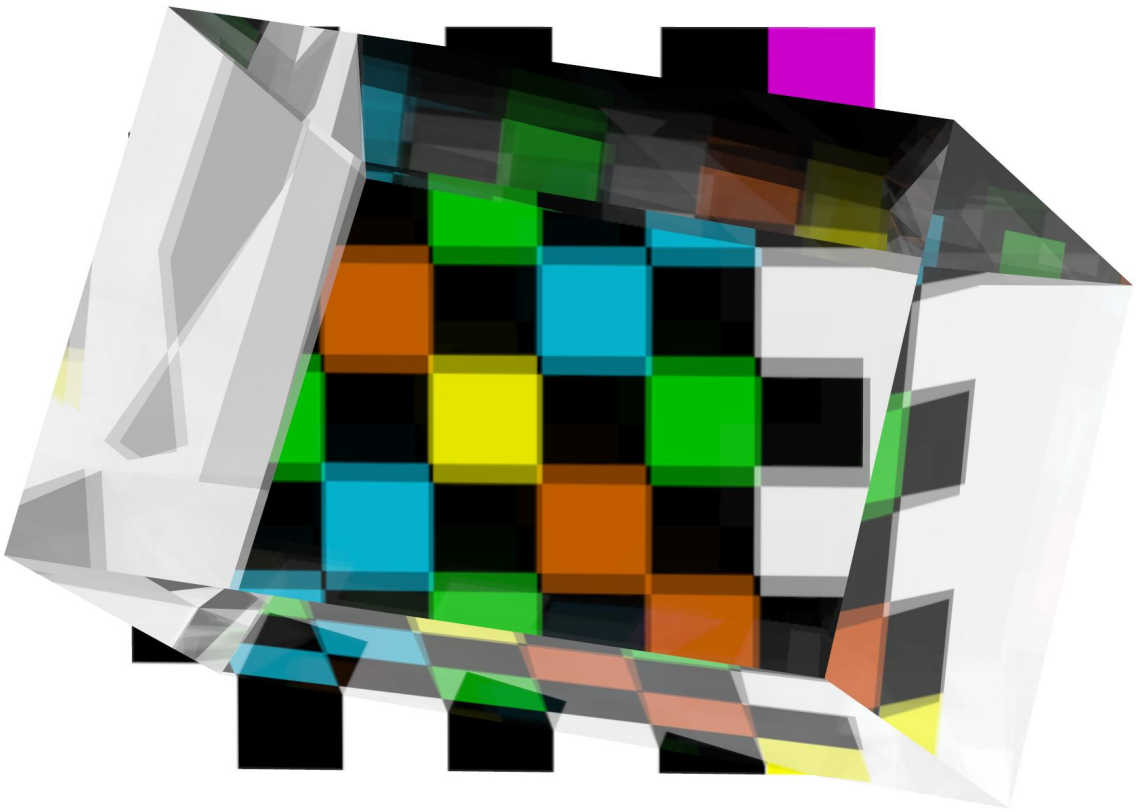

Figure 6. Simulated camera image of the scene from Figure 5.

## APPENDIX A. FORWARD RAY THAT WOULD TRANSFORM INTO THE GIVEN BACKWARD

In terms of physics (or the forward ray tracing) we are now given the *outgoing* ray direction and must find that of the *incident* wave. If polarization type of that outgoing ray is known, we do calculations *for it only*; otherwise they are repeated *for both types*, see Figure 2.

The calculation is much similar to that for a forward ray tracing and it is based on the very basic law: the tangent component of wave vector is the same for $\boldsymbol{s}_{in}$ and $\boldsymbol{s}_{out}$.

In both FRT and BRT the calculation scheme is the same:

1. Calculate the tangent wave vector $\boldsymbol{k}_\tau$ from the "incident" (known) ray. In FRT this is the forward ray from light source. In BRT this is the backward ray from camera. The result depends on the polarization state of the "incident" ray.

2. Calculate the full wave vector of the "outgoing" (sought-for; unknown) ray. In FRT it is the ray from the surface towards camera; in BRT it is the ray from the surface towards light source. Its tangent component equals $\boldsymbol{k}_\tau$ found above, so we only calculate its normal component. The result depends on the polarization state of the "outgoing" ray; if it is unknown in advance, we just do for all possible types.

3. From the wave vector (of the outgoing ray), found above, calculate direction of ray.

The mathematical method used was originally designed for the forward ray tracing,[3] but its first step, quite trivial in FRT, makes more sense for BRT. This is because BRT has *luminance correction factor*, absent in FRT, and we calculate these factors just from the properties of "the ray to tangent wave vector" transform. So we shall explain it here.

### A.1 Tangent wave vector from ray direction

This transformation is quite simple, but since it is done frequently here and there in simulation, one needs a simple and efficient form of it. Also we shall use the exact form of this transformation to calculate Jacobian later, see Appendix B.1.

For the *ordinary* case when the ray and wave vectors are parallel,

$$\boldsymbol{k}_\tau = (\boldsymbol{s} - (\boldsymbol{s}\cdot\boldsymbol{n})\boldsymbol{n})\, k_o$$

For the *extraordinary* case, the calculation is a bit more complicated. From the well-known "wave ellipsoid"[7]

$$(1-\rho^2)(\boldsymbol{k}_e\cdot\boldsymbol{a})^2 + \rho^2(\boldsymbol{k}_e\cdot\boldsymbol{k}_e) = k_o^2, \tag{2}$$

it follows that the wave number is

$$k_e = \frac{k_o}{\sqrt{(1-\rho^2)(\boldsymbol{\kappa}\cdot\boldsymbol{a})^2+\rho^2}}$$

where

$$\boldsymbol{\kappa} \equiv \boldsymbol{k}_e / |k_e|$$

is the *unit* (normalized) vector of wave direction. Also, the tangent component of the wave vector can be written as

$$\boldsymbol{k}_\tau = (\boldsymbol{\kappa} - (\boldsymbol{\kappa}\cdot\boldsymbol{n})\boldsymbol{n})k_e = \frac{(\boldsymbol{\kappa} - (\boldsymbol{\kappa}\cdot\boldsymbol{n})\boldsymbol{n})k_o}{\sqrt{(1-\rho^2)(\boldsymbol{\kappa}\cdot\boldsymbol{a})^2+\rho^2}} \tag{3}$$

On the other hand, the ray-wave transformation (see Eq. (4) from Ref. 2,3) gives for the normalized wave vector:

$$\boldsymbol{\kappa} = \frac{\boldsymbol{s} + (\rho^2-1)(\boldsymbol{a}\cdot\boldsymbol{s})\boldsymbol{a}}{|\boldsymbol{s} + (\rho^2-1)(\boldsymbol{a}\cdot\boldsymbol{s})\boldsymbol{a}|} = \frac{\boldsymbol{s} + (\rho^2-1)(\boldsymbol{a}\cdot\boldsymbol{s})\boldsymbol{a}}{\sqrt{1+(\rho^4-1)(\boldsymbol{a}\cdot\boldsymbol{s})^2}}$$

and substituting it into the above Eq. (3) yields

$$\begin{aligned}\boldsymbol{k}_\tau &= \frac{k_o}{\rho}\sqrt{\frac{1+(\rho^4-1)(\boldsymbol{a}\cdot\boldsymbol{s})^2}{1+(\rho^2-1)(\boldsymbol{a}\cdot\boldsymbol{s})^2}}(\boldsymbol{\kappa} - (\boldsymbol{\kappa}\cdot\boldsymbol{n})\boldsymbol{n}) \\ &= \frac{k_o}{\rho}\frac{\boldsymbol{s} - (\boldsymbol{s}\cdot\boldsymbol{n})\boldsymbol{n} + (\rho^2-1)(\boldsymbol{a}\cdot\boldsymbol{s})(\boldsymbol{a} - (\boldsymbol{a}\cdot\boldsymbol{n})\boldsymbol{n})}{\sqrt{1+(\rho^2-1)(\boldsymbol{a}\cdot\boldsymbol{s})}}\end{aligned} \tag{4}$$

Notice this also covers the ordinary case if assume $\rho = 1$ for an ordinary ray.

### A.2 Full wave vector from its tangent component

While the tangent wave vector component was found from the known, "incident" ray, the normal component is to be calculated. This depends on the "desired" polarization state of the scattered wave.

If the "scattered" ray is *ordinary*, this is trivial since the ordinary wave number is fixed:

$$k_n = \pm\sqrt{k_o^2 - |\boldsymbol{k}_\tau|^2}$$

The sign is determined by the boundary side the ray goes to (notice that for the ordinary ray directions of the wave and the ray vector coincide), i.e. whether it is refraction or transmission.

If the "scattered" ray is *extraordinary*, the wave vector obeys Eq. (2), and substituting in in the already *known* tangent component we have the same quadratic equation as in FRT[3]

$$((1-\rho^2)(\boldsymbol{n}\cdot\boldsymbol{a})^2 + \rho^2)k_n^2 + 2(1-\rho^2)(\boldsymbol{k}_\tau\cdot\boldsymbol{a})(\boldsymbol{n}\cdot\boldsymbol{a})k_n = k_o^2 - \rho^2(\boldsymbol{k}_\tau\cdot\boldsymbol{k}_\tau) - (1-\rho^2)(\boldsymbol{k}_\tau\cdot\boldsymbol{a})^2$$

This quadratic equation has two roots

$$k_n = \frac{(\rho^2-1)(\boldsymbol{a}\cdot\boldsymbol{n})(\boldsymbol{a}\cdot\boldsymbol{k}_\tau) \pm \sqrt{\rho^2(k_o^2 - |\boldsymbol{k}_\tau|^2) - (\rho^2-1)(k_o^2(\boldsymbol{a}\cdot\boldsymbol{n})^2 + \rho^2(|\boldsymbol{k}_\tau|^2(1-(\boldsymbol{a}\cdot\boldsymbol{n})^2) - (\boldsymbol{a}\cdot\boldsymbol{k}_\tau)^2)}}{\rho^2 - (\rho^2-1)(\boldsymbol{a}\cdot\boldsymbol{n})^2} \tag{5}$$

and we select the one whose *ray* goes *away* from the boundary — again like in FRT, see Eq. (9) of Ref. 2,3 i.e. "+" is for reflection and "–" is for refraction.

Notice this also covers the ordinary case if assume $\rho = 1$ for an ordinary ray.

The ray vector is given by Eq. (8) of Ref. 2,3:

$$\boldsymbol{s} = \frac{\boldsymbol{k} + (\rho^{-2}-1)(\boldsymbol{a}\cdot\boldsymbol{k})\boldsymbol{a}}{|\boldsymbol{k} + (\rho^{-2}-1)(\boldsymbol{a}\cdot\boldsymbol{k})\boldsymbol{a}|};$$

substituting in it $\boldsymbol{k} = \boldsymbol{k}_\tau + \boldsymbol{n}k_n$ with $k_n$ from Eq. (5) one has

$$\boldsymbol{s} = \frac{\boldsymbol{k_\tau} + (\rho^{-2}-1)(\boldsymbol{a}\cdot\boldsymbol{k}_\tau)\boldsymbol{a} + k_n(\boldsymbol{n} + (\rho^{-2}-1)(\boldsymbol{a}\cdot\boldsymbol{n})\boldsymbol{a})}{|\boldsymbol{k_\tau} + (\rho^{-2}-1)(\boldsymbol{a}\cdot\boldsymbol{k}_\tau)\boldsymbol{a} + k_n(\boldsymbol{n} + (\rho^{-2}-1)(\boldsymbol{a}\cdot\boldsymbol{n})\boldsymbol{a})|} \tag{6}$$

Again, this also covers the ordinary case if assume $\rho = 1$ for an ordinary ray.

## APPENDIX B. JACOBIAN OF RAY DIRECTION TRANSFORMATION

This transformation is the composition of the transformation (4) with transformation (5)+(6). Instead of the straightforward calculation of the Jacobian, we shall proceed from the obvious fact that the second transform is just the *inverse* of the first one, just with another parameters of the medium.

In other words, we note that the tangent wave vector for both the "incident" and "outgoing" rays is the same. That is, if we write (4) as

$$\boldsymbol{k}_\tau = F_{\boldsymbol{n};\boldsymbol{a};\eta}(\boldsymbol{s}) \tag{7}$$

(where particular form of $F$ depends on the polarization type) then

$$F_{n_{out};\boldsymbol{a}_{out};\eta_{out}}(\boldsymbol{s}_{out}) = \boldsymbol{k}_\tau = F_{n_{in};\boldsymbol{a}_{in};\eta_{in}}(\boldsymbol{s}_{in})$$

and therefore

$$\left|\frac{\partial(F_{n_{out};\boldsymbol{a}_{out};\eta_{out}}(\boldsymbol{s}))}{\partial\boldsymbol{s}}\right| d^2\boldsymbol{s}_{out} = d^2\boldsymbol{k}_\tau = \left|\frac{\partial(F_{n_{in};\boldsymbol{a}_{in};\eta_{in}}(\boldsymbol{s}))}{\partial\boldsymbol{s}}\right| d^2\boldsymbol{s}_{in}$$

from what it follows that

$$d^2\boldsymbol{s}_{out} = \frac{\left|\frac{\partial(F_{n_{in};\boldsymbol{a}_{in};\eta_{in}}(\boldsymbol{s}))}{\partial \boldsymbol{s}}\right|}{\left|\frac{\partial(F_{n_{out};\boldsymbol{a}_{out};\eta_{out}}(\boldsymbol{s}))}{\partial \boldsymbol{s}}\right|} d^2\boldsymbol{s}_{in}$$

which is nothing but "derivative of an implicit function".

Jacobian of the transformation of the ray into the tangent wave vector is derived in Appendix B.1. It can be written in the form (14) for any polarization state, if assume $\rho = 1$ for an ordinary ray. Substituting that for here we eventually arrive at

$$d^2\boldsymbol{s}_{out} = \left(\frac{\eta_{o;out}}{\eta_{o;in}}\right)^2 \left(\frac{1+(\rho_{in}^2-1)(\boldsymbol{s}_{in}\cdot\boldsymbol{a}_{in})^2}{1+(\rho_{out}^2-1)(\boldsymbol{s}_{out}\cdot\boldsymbol{a}_{out})^2}\right)^2 \frac{|(\boldsymbol{s}_{out}\cdot n)|}{|(\boldsymbol{s}_{in}\cdot n)|} d^2\boldsymbol{s}_{in} \tag{8}$$

## B.1 Jacobian of the ray to the tangent wave vector transform (7)

### B.1.1 The ordinary polarization state

Now the ray and wave vectors are parallel, so

$$\boldsymbol{k}_\tau = (\boldsymbol{s} - (\boldsymbol{s}\cdot\boldsymbol{n})\boldsymbol{n})\, k_o$$

and the differential area $d^2\boldsymbol{k}_\tau/k_o$ is just the projection of the differential area on the unit sphere:

$$d^2\boldsymbol{k}_\tau = k_o^2|(\boldsymbol{s}\cdot\boldsymbol{n})|d^2\boldsymbol{s} \tag{9}$$

### B.1.2 The extraordinary polarization state

Differentiating (4) gives

$$d\boldsymbol{k}_\tau = \frac{k_o}{\rho}\frac{d\boldsymbol{s} - (d\boldsymbol{s}\cdot\boldsymbol{n})\boldsymbol{n} + (\rho^2-1)(\boldsymbol{a}\cdot d\boldsymbol{s})\boldsymbol{a}_\tau}{\sqrt{A}} - \frac{k_o}{\rho}\frac{\boldsymbol{s}_\tau + (\rho^2-1)(\boldsymbol{a}\cdot\boldsymbol{s})\boldsymbol{a}_\tau}{A^{3/2}}(\rho^2-1)(\boldsymbol{a}\cdot\boldsymbol{s})(\boldsymbol{a}\cdot d\boldsymbol{s}) \tag{10}$$

where

$$\begin{aligned}\boldsymbol{s}_\tau &\equiv \boldsymbol{s} - (\boldsymbol{s}\cdot\boldsymbol{n})\boldsymbol{n}\\ \boldsymbol{a}_\tau &\equiv \boldsymbol{a} - (\boldsymbol{a}\cdot\boldsymbol{n})\boldsymbol{n}\end{aligned}$$

are the tangent components, and

$$A \equiv 1 + (\rho^2-1)(\boldsymbol{a}\cdot\boldsymbol{s})^2$$

Since a variation of a *unit* vector is orthogonal to that vector, $d\boldsymbol{s}$ is an effectively 2D and so can be written as a sum

$$d\boldsymbol{s} = \delta_1\boldsymbol{n}' + \delta_2\boldsymbol{a}'$$

of two vectors orthogonal to $\boldsymbol{s}$:

$$\begin{aligned}\boldsymbol{a}' &\equiv \boldsymbol{a} - (\boldsymbol{s}\cdot\boldsymbol{a})\boldsymbol{s}\\ \boldsymbol{n}' &\equiv \boldsymbol{n} - (\boldsymbol{s}\cdot\boldsymbol{n})\boldsymbol{s}\end{aligned}$$

Substituting this into (10) one obtains after some simple transformations that

$$d\boldsymbol{k}_\tau = \frac{\frac{k_o}{\rho}}{A^{3/2}}\left((\alpha_1\delta_1 + \alpha_2\delta_2)\boldsymbol{a}_\tau - (\beta_1\delta_1 + \beta_2\delta_2)\boldsymbol{s}_\tau\right) \tag{11}$$

where

$$\begin{aligned}
\alpha_1 &= (\rho^2 - 1)\left((\boldsymbol{a}\cdot\boldsymbol{n}) - (\boldsymbol{a}\cdot\boldsymbol{s})(\boldsymbol{s}\cdot\boldsymbol{n})\right) \\
\alpha_2 &= \rho^2 \\
\beta_1 &= (\boldsymbol{s}\cdot\boldsymbol{n}) + (\rho^2 - 1)(\boldsymbol{a}\cdot\boldsymbol{s})(\boldsymbol{a}\cdot\boldsymbol{n}) \\
\beta_2 &= \rho^2(\boldsymbol{a}\cdot\boldsymbol{s})
\end{aligned} \tag{12}$$

For the effectively 2D vector $d\boldsymbol{s}$ the area element $d^2\boldsymbol{s}$ is the area of a parallelogram spanned by two *different* variations $d\boldsymbol{s}$ and $d\boldsymbol{s}'$:

$$d^2\boldsymbol{s} = \sqrt{|d\boldsymbol{s}|^2|d\boldsymbol{s}'|^2 - (d\boldsymbol{s}\cdot d\boldsymbol{s}')^2}$$

Correspondingly, $d^2\boldsymbol{k}_\tau$ is the area of a parallelogram spanned by the *responses* to these two $d\boldsymbol{s}$ and $d\boldsymbol{s}'$, respectively, i.e

$$d^2\boldsymbol{k}_\tau = \sqrt{|d\boldsymbol{k}_\tau|^2|d\boldsymbol{k}'_\tau|^2 - (d\boldsymbol{k}_\tau\cdot d\boldsymbol{k}'_\tau)^2}$$

We can choose *any* not parallel $d\boldsymbol{s}$ and $d\boldsymbol{s}'$ because all this is merely a means to calculate the determinant of the linear operator (10), e.g.

$$\begin{aligned}
d\boldsymbol{s} &= \delta_1\boldsymbol{n}' \\
d\boldsymbol{s}' &= \delta_2\boldsymbol{a}'
\end{aligned}$$

For them, the parallelogram area is

$$d^2\boldsymbol{s} = |\delta_1\delta_2|\sqrt{1 - (\boldsymbol{s}\cdot\boldsymbol{a})^2 - (\boldsymbol{s}\cdot\boldsymbol{n})^2 + (\boldsymbol{a}\cdot\boldsymbol{n})^2 + 2(\boldsymbol{s}\cdot\boldsymbol{a})(\boldsymbol{a}\cdot\boldsymbol{n})(\boldsymbol{s}\cdot\boldsymbol{n})} \tag{13}$$

Then, according to (11) the responses to these $d\boldsymbol{s}$ and $d\boldsymbol{s}'$ are

$$\begin{aligned}
d\boldsymbol{k}_\tau &= \frac{\frac{k_o}{\rho}}{A^{3/2}}\delta_1(\alpha_1\boldsymbol{a}_\tau - \beta_1\boldsymbol{s}_\tau) \\
d\boldsymbol{k}'_\tau &= \frac{\frac{k_o}{\rho}}{A^{3/2}}\delta_2(\alpha_2\boldsymbol{a}_\tau - \beta_2\boldsymbol{s}_\tau)
\end{aligned}$$

and for them the parallelogram area is

$$d^2\boldsymbol{k}_\tau = \left(\frac{\frac{k_o}{\rho}}{A^{3/2}}\right)^2|\delta_1\delta_2||\alpha_1\beta_2 - \alpha_2\beta_1|\sqrt{1 - (\boldsymbol{s}\cdot\boldsymbol{n})^2 - (\boldsymbol{a}\cdot\boldsymbol{n})^2 - (\boldsymbol{s}\cdot\boldsymbol{a})^2 + 2(\boldsymbol{a}\cdot\boldsymbol{n})(\boldsymbol{s}\cdot\boldsymbol{n})(\boldsymbol{s}\cdot\boldsymbol{a})}$$

Combining with Eq. (13) we obtain

$$d^2\boldsymbol{k}_\tau = \left(\frac{\frac{k_o}{\rho}}{A^{3/2}}\right)^2 |\alpha_1\beta_2 - \alpha_2\beta_1| d^2\boldsymbol{s}$$

which substituting Eq. (12) yields

$$d^2\boldsymbol{k}_\tau \quad = \quad \left(\frac{k_o}{1+(\rho^2-1)(\boldsymbol{a}\cdot\boldsymbol{s})^2}\right)^2 |(\boldsymbol{s}\cdot\boldsymbol{n})|\, d^2\boldsymbol{s} \tag{14}$$

Notice this also covers the ordinary case (Eq. (9)) if assume $\rho = 1$ for an ordinary ray.

## REFERENCES


[1] S. Chandrasekhar. *Radiative Transfer*. Dover books on physics and engineering. Clarendon Press, 1950. 3.2

[2] S.V. Ershov, A.A. Garbul, S.G. Pozdnyakov, V.G. Sokolov, and A.G. Voloboy. Ray tracing in presence of birefrigent media. In *Proceedings of 26-th International Conference on Computer Graphics and Vision, September 19-23, 2016.*, pages 221–226, Nizhny Novgorod, 2016. 1, A.1, A.2

[3] S.V. Ershov, A.A. Garbul, S.G. Pozdnyakov, V.G. Sokolov, and A.G. Voloboy. Lighting simulation in anisotropic media. *Mathematica Montisnigri*, XXXIX:42–56, 2017. 1, 3.2, A, A.1, A.2, A.2

[4] F.I. Fedorov. *Optics of anisotropic media (in Russian)*. Editorial URSS, Moscow, 2004. 1, 3.1

[5] V.K. Ignatovich F.V. Ignatovich. Optics of anisotropic media. *Physics-Uspekhi*, 55(7):709–720, 2012. 1

[6] Stephane Guy and Cyril Soler. Graphics gems revisited: Fast and physically-based rendering of gemstones. *ACM Trans. Graph.*, 23(3):231–238, August 2004. 1

[7] Pitaevskii L. P. Landau L. D., Lifshitz E. M. *Electrodynamics of Continuous Media*. Butterworth-Heinemann, Oxford, 2004. 1, 3.1, A.1

[8] Stephen C. McClain, Lloyd W. Hillman, and Russell A. Chipman. Polarization ray tracing in anisotropic optically active media. i. algorithms. *J. Opt. Soc. Am. A*, 10(11):2371–2382, Nov 1993. 1

[9] Stephen C. McClain, Lloyd W. Hillman, and Russell A. Chipman. Polarization ray tracing in anisotropic optically active media. ii. theory and physics. *J. Opt. Soc. Am. A*, 10(11):2383–2393, Nov 1993. 1

[10] R J Potton. Reciprocity in optics. *Reports on Progress in Physics*, 67(5):717–754, 2004. 3.2, 3.2

[11] D.S. Kozlov V.A. Debelov. A local model of light interaction with isotropic and uniaxial transparent media. *Vestnik NGU. Informational technologies series*, 10:5–23, 2012. 1, 5

[12] A.V. Ignatenko V.V. Afanasiev, A.G. Voloboy. Automation of processing of transparent crystals. In NA, editor, *Proceedings of the 15th international conference CAD/CAM/PDM-2015*, pages 151–155, 2015. 1

[13] Andrea Weidlich and Alexander Wilkie. Realistic rendering of birefringency in uniaxial crystals. *ACM Trans. Graph.*, 27(1):6:1–6:12, March 2008. 1

[14] Dmitry D. Zhdanov, Vadim G. Sokolov, Igor S. Potemin, Alexey G. Voloboy, Vladimir A. Galaktionov, and Nikolay Kirilov. Modeling and computer design of liquid crystal display backlight with light polarization film. *Optical Review*, 21(5):642–650, Sep 2014. 1