Technical Report
May 2020

Real-time High-Quality Rendering of Non-Rotating Black Holes

Eric Bruneton

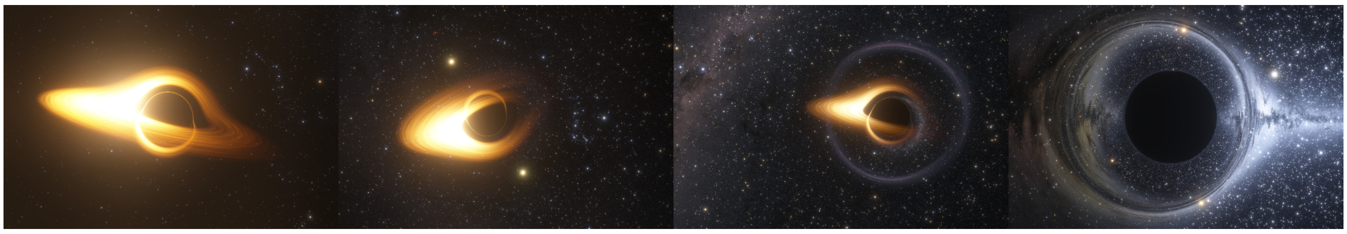

Figure 1: Some results obtained with our method. *Left*: distorted images of an accretion disc due to gravitational light bending, with relativistic Doppler and beaming effects. *Middle*: gravitational lensing creates several amplified images of each punctual star and creates Einstein rings. *Right*: near the speed of light, light is amplified and blue-shifted ahead, and is reduced and red-shifted behind.

Abstract

We propose a real-time method to render high-quality images of a non-rotating black hole with an accretion disc and background stars. Our method is based on beam tracing, but uses precomputed tables to find the intersections of each curved light beam with the scene in constant time per pixel. It also uses a specific texture filtering scheme to integrate the contribution of the light sources to each beam. Our method is simple to implement and achieves high frame rates.

1. Introduction

Black holes are strange objects which recently got a lot of public exposure with the Interstellar movie [JvTFT15], the detection of gravitational waves from merging black holes [AAA+16], and the first image of a black hole [EHT19]. A real-time, high-quality visualization of a black hole could help the public in getting an intuitive "understanding" of their properties, for instance in planetariums or in 3D astronomy software. It could also be useful in space games. In this context, we propose a real-time high-quality rendering method for non-rotating black holes, with 2 contributions: a precomputation method for constant time beam tracing, and a texture filtering scheme to compute the contribution of the light sources to each beam.

We present the related work in Section 2, our model in Section 3 and its implementation in Section 4. We conclude with a discussion of our results, limitations and future work in Sections 5 and 6.

2. Related work

Black hole visualization has a long history starting with [Lum79], and summarized in [Lum19]. Offline rendering methods generally use beam tracing in curved space-time, support rotating black holes and produce very high-quality images [Ham14, Ria14, JvTFT15]. However, they are complex to implement and are not interactive (*e.g.* an IMAX Interstellar frame requires at least 30 minutes with 10 cores and the renderer has 40kLoC [JvTFT15]). Physically accurate general relativistic magnetohydrodynamics simulations of accretion discs are even more complex and require supercomputers [LHT+18].

Our work is more related to interactive visualization methods, usually restricted to non-rotating black holes to reduce the complexity of the problem. [MW10] render about 120000 background stars around a black hole, whose apparent positions, colors and intensities change due to the gravitational effects. Each star is rendered with a point primitive, and its projection(s) on screen are found in constant time by using a precomputed 4096×4096 lookup table. [MB11] render a torus and a background night sky texture for an observer orbiting a black hole, with a ray-tracing method. Ray intersections with the scene are found in constant time thanks to lookup tables precomputed with a parallelized code (for a fixed orbit radius). [MF12] use ray-tracing to render an accretion disc around a black hole. Ray intersections are found in constant time by using an analytic expression involving the Jacobi-sn function (evaluated with arithmetic-geometric, complex number series).

In comparison, our method uses only two small tables (512×512 and 64×32) which are very fast to precompute. They are used to find, in constant time, the intersection(s) of curved light beams with the accretion disc and millions of background stars (stored in a cubemap with a specific filtering scheme).

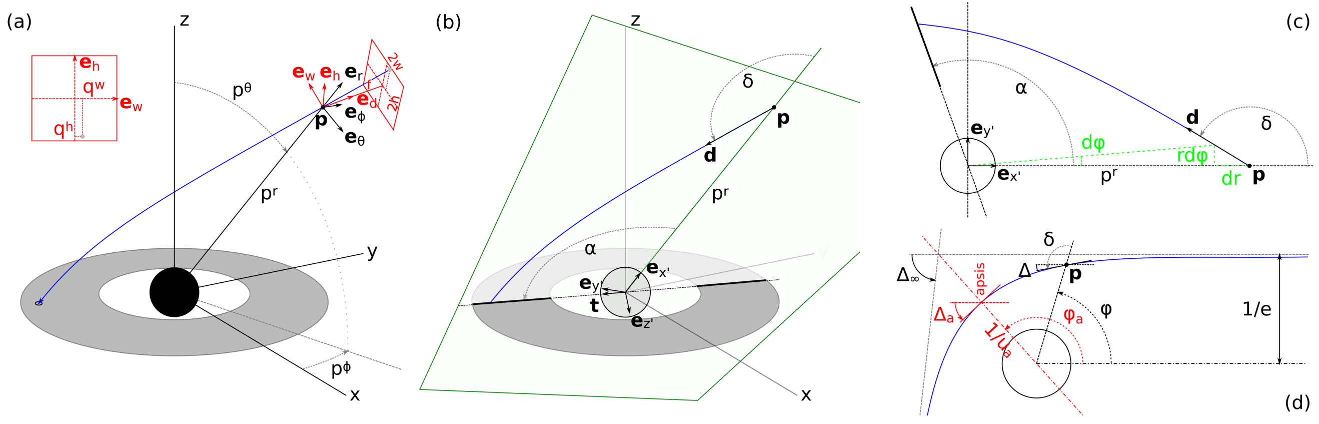

Figure 2: Notations. *(a)* the camera reference frame and image plane (in red) and a curved light ray (in blue) intersecting the accretion disc. *(b)* in the plane containing the light ray, the initial ray angle is noted δ and the accretion disc inclination α. *(c)* δ verifies $\tan\delta = r\mathrm{d}\varphi/\mathrm{d}r$ (in green), *i.e.* $\delta = \pi - \arctan 2(u, \dot{u})$. *(d)* the deflection Δ verifies $\Delta = \varphi + \delta - \pi$. The ray is symmetric around the axis through its apsis (in red).

3. Model

Our goal is to render a non-rotating black hole with an accretion disc and background stars, illustrating the effects of gravitation on light. Simulating a realistic accretion disc is *not* a goal: we thus use a basic, infinitely thin disc model instead. However, we want to get real-time *and* high-quality images, which is not easy:

- a simple ray-marching algorithm can render a sky map texture distorted by a black hole in real-time, but not with a high quality (*e.g.* stars become curved segments instead of staying punctual),
- conversely, offline beam-tracing methods produce high-quality images but are not real-time [JvTFT15].

To this end, we propose a "precomputed beam tracing" method: for each pixel, we initialize a light beam, compute its intersections with the scene using precomputed tables, and then the light received from the intersected objects. These 3 steps are explained below, after a very short introduction to the Schwarzschild metric.

3.1. Schwarzschild metric

The space-time geometry around a non-rotating black hole can be described with the Schwarzschild metric. In units such that the radius of the black hole's event horizon and the speed of light are 1, this metric is

$$\mathrm{d}\tau^2 = \left(1-\frac{1}{r}\right)\mathrm{d}t^2 - \left(1-\frac{1}{r}\right)^{-1}\mathrm{d}r^2 - r^2\mathrm{d}\theta^2 - r^2\sin^2\theta\mathrm{d}\phi^2 \quad (1)$$

where τ is the proper time and (t, r, θ, ϕ) are the Schwarzschild coordinates [Wei72]. r, θ, ϕ are (pseudo-)spherical coordinates, and in the following we also use the corresponding (pseudo-)Cartesian coordinates x, y, z, as well as the inverse radius $u \triangleq 1/r$. In particular, the Cartesian coordinates of an orthonormal basis $\vec{e}_t, \vec{e}_r, \vec{e}_\theta, \vec{e}_\phi$ for a static observer at (t, r, θ, ϕ) are, respectively (see Fig. 2a)

$$\frac{1}{\sqrt{1-u}}\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \sqrt{1-u}\begin{bmatrix}0\\\sin\theta\cos\phi\\\sin\theta\sin\phi\\\cos\theta\end{bmatrix}, \begin{bmatrix}0\\\cos\theta\cos\phi\\\cos\theta\sin\phi\\-\sin\theta\end{bmatrix}, \begin{bmatrix}0\\-\sin\phi\\\cos\phi\\0\end{bmatrix} \quad (2)$$

3.2. Beam initialization

The first step of our method is to compute, for each pixel, the initial direction of the corresponding light beam. As [MW10], and in order to simplify the next steps, we take advantage of some symmetries to reduce this direction to a single angle δ, as shown below.

Let $p = (p^t, p^r, p^\theta, p^\phi)$ be the camera position in Schwarzschild coordinates, and Λ the Lorentz transformation [Wei72] specifying the camera orientation and velocity with respect to a static observer at p. An orthonormal basis for the camera is thus $\vec{e}_\tau, \vec{e}_w, \vec{e}_h, \vec{e}_d$ (see Fig. 2a), given by

$$\vec{e}_i = \Lambda_i{}^j\vec{e}_j, \quad i \in \{\tau, w, h, d\}, \quad j \in \{t, r, \theta, \phi\} \quad (3)$$

For a pinhole camera with focal length f, the initial beam direction $\mathbf{d}$ for a pixel with screen coordinates q^w, q^h is thus (see Fig. 2a)

$$\mathbf{d} = -\sqrt{(q^w)^2 + (q^h)^2 + f^2}\,\mathbf{e}_\tau + q^w\mathbf{e}_w + q^h\mathbf{e}_h - f\mathbf{e}_d \quad (4)$$

where $\mathbf{e}$ denotes the spatial part of a 4-vector $\vec{e}$.

We now take advantage of the spherical symmetry of the metric, and of the fact that its geodesics are planar [Wei72], to reduce $\mathbf{d}$ to a single angle. Let $(t, r, \vartheta, \varphi)$ be *rotated* Schwarzschild coordinates such that the beam's axial ray is contained in the equatorial plane $\vartheta = \pi/2$. They can be defined as the (pseudo-)spherical coordinates corresponding to the following new orthonormal basis vectors (for the Euclidean metric – see Fig. 2b):

$$\mathbf{e}_{x'} \triangleq \frac{\mathbf{p}}{p^r} \quad \mathbf{e}_{y'} \triangleq \frac{\mathbf{e}_{z'} \wedge \mathbf{e}_{x'}}{\|\mathbf{e}_{z'} \wedge \mathbf{e}_{x'}\|} \quad \mathbf{e}_{z'} \triangleq \frac{\mathbf{e}_{x'} \wedge \mathbf{d}}{\|\mathbf{e}_{x'} \wedge \mathbf{d}\|} \quad (5)$$

In these rotated coordinates the metric (1) keeps the same form and the light beam starts from $(p^t, p^r, \pi/2, 0)$ with an initial angle $\delta \triangleq \arccos(\mathbf{e}_{x'} \cdot \mathbf{d}/\|\mathbf{d}\|)$ from the x' axis (see Fig. 2b).

Finally, note that in the $\vartheta = \pi/2$ plane the accretion disc becomes two line segments at angles α and $\alpha + \pi$ from the x' axis, with

$$\alpha = \arccos(\mathbf{e}_{x'} \cdot \mathbf{t}) \quad \mathbf{t} \triangleq \pm\mathbf{e}_z \wedge \mathbf{e}_{z'}/\|\mathbf{e}_z \wedge \mathbf{e}_{z'}\| \quad (6)$$

and where the sign is chosen such that $\mathbf{t} \cdot \mathbf{e}_{y'} \geq 0$ (see Fig. 2b).

3.3. Beam tracing

The second step of our method is to compute the beam intersections with the scene, and the light emitted there. For this we first need to determine the geodesic followed by the beam's axial ray.

For light rays $d\tau = 0$, and there exist curvilinear coordinates σ, defined up to an affine transform, such that $(1-u)dt/d\sigma$ and $r^2\sin^2\vartheta d\varphi/d\sigma$ are constant along the ray [Wei72]. We can thus choose σ such that

$$(1-u)\frac{dt}{d\sigma} = e \quad \text{and} \quad \sin^2\vartheta\frac{d\varphi}{d\sigma} = u^2 \tag{7}$$

where e is a constant of motion. By substituting this in (1) with $d\tau = 0$ and $\vartheta = \pi/2$ we get the geodesic equation

$$\dot{u}^2 \triangleq \left(\frac{du}{d\varphi}\right)^2 = e^2 - u^2(1-u) \quad \Rightarrow \quad \ddot{u} = \frac{3}{2}u^2 - u \tag{8}$$

Integrating this numerically at each pixel, with a high precision, would be too slow (see Section 5). Alternatively, the analytic solution for $u(\varphi)$, using the Jacobi-sn function (not available on GPU), could be implemented with numerical series [MF12]. However, we also need the retarded time (to animate the accretion disc) and the light ray deflection (for the stars). To compute all this easily and efficiently, we use instead two small precomputed tables. We explain below how we precompute and use these tables to find the beam intersections, thanks to some ray properties that we present first.

3.3.1. Ray properties

Light rays can be divided in 3 types. If e^2 is larger than the maximum $\mu \triangleq 4/27$ of $u^2(1-u)$ over $[0,1]$, reached at the *photon sphere* $u = 2/3$, then (8) shows that all values of u are possible. The light ray thus comes from infinity into the black hole, or vice-versa. Otherwise, some values around $2/3$ are excluded. The ray either stays in the (empty) region $u > 2/3$, or comes from infinity, reaches an apsis $u_a < 2/3$, and goes back to infinity. In the later case u_a is given by setting $\dot{u} = 0$ in (8):

$$u_a = \frac{1}{3} + \frac{2}{3}\sin\left(\frac{1}{3}\arcsin\left(\frac{2e^2}{\mu} - 1\right)\right) \tag{9}$$

and the light ray is unchanged by the reflection $\varphi \to 2\varphi_a - \varphi$ (see Fig. 2d). In any case, $\varphi \to -\varphi$ changes a solution of (8) into another, with $e, \sigma, \dot{u}$ and α changed into their opposite and δ into $\pi - \delta$.

3.3.2. Precomputations and beam tracing: background stars

For background stars we first compute the beam's escape angle, and then sum the light emitted by all the stars in the beam's footprint on the celestial sphere, around this escape direction.

Escape angle Let δ' be the beam's escape angle (or ∞ if it falls into the black hole), measured from the x' axis. For efficient rendering, δ' could be precomputed for all initial conditions p^r, δ. But this would yield an $O(n^3)$ algorithm. Instead, we precompute the deflection Δ of rays coming from infinity (see Fig. 2d) in a $\mathbb{D}(e,u)$ table, for all $e \geq 0$ and $u < 1$ or $u \leq u_a$ (depending on e and taking advantage of the above symmetries). This gives a trivial $O(n^2)$ algorithm (see Algorithm 1 – we use the Euler method but Runge-Kutta or other methods are possible too). At runtime, we compute δ' as $\delta + \Delta$ or $\delta + \Delta_\infty - \Delta = \delta + 2\Delta_a - \Delta$, depending on the ray direction (see Fig. 2d and Algorithm 1).

```
procedure PRECOMPUTE($\varepsilon$)
 for all $e \geq 0$
  $t \leftarrow 0,\ u \leftarrow 0,\ \dot{u} \leftarrow e,\ \varphi \leftarrow 0,\ d\varphi \leftarrow \varepsilon$
  while $u < 1$ and ($\dot{u} \geq 0$ or $\varphi < \pi$)
   if $\dot{u} \geq 0$  then $\mathbb{D}(e,u) \leftarrow [t,\ \Delta = \varphi - \arctan2(u,\dot{u})]$
   if $\varphi < \pi$  then $\mathbb{U}(e,\varphi) \leftarrow [t,\ u]$
   if $u > 0$  then $t \leftarrow t + e\,d\varphi/(u^2 - u^3)$
   $\dot{u} \leftarrow \dot{u} + (3u^2/2 - u)d\varphi,\ u \leftarrow u + \dot{u}d\varphi,\ \varphi \leftarrow \varphi + d\varphi$

procedure TRACERAY($p^r, \delta, \alpha, u_{ic}, u_{oc}$)
 $u \leftarrow 1/p^r,\ \dot{u} \leftarrow -u\cot\delta,\ e^2 \leftarrow \dot{u}^2 + u^2(1-u)$
 if $e^2 < \mu$ and $u > 2/3$  then return $(\infty, \emptyset)$
 $s \leftarrow \mathrm{sign}(\dot{u}),\ [t,\Delta] \leftarrow \mathbb{D}(e,u),\ [t_a,\Delta_a] \leftarrow \mathbb{D}(e,u_a)$
 $\varphi \leftarrow \Delta + (s = 1\ ?\ \pi - \delta\ :\ \delta) + s\alpha,\ \varphi_a \leftarrow \Delta_a + \pi/2$
 $\varphi_0 \leftarrow \varphi \mod \pi,\ [t_0,u_0] \leftarrow \mathbb{U}(e,\varphi_0), I \leftarrow \emptyset$
 if $\varphi_0 < \varphi_a$ and $u_{oc} \leq u_0 \leq u_{ic}$ and $\mathrm{sign}(u_0 - u) = s$  then
  $I \leftarrow I \cup [s(t_0 - t),\ u_0,\ \alpha + \varphi - \varphi_0]$
 if $e^2 < \mu$ and $s = 1$  then
  $\varphi \leftarrow 2\varphi_a - \varphi,\ \varphi_1 \leftarrow \varphi \mod \pi,\ [t_1,u_1] \leftarrow \mathbb{U}(e,\varphi_1)$
  if $\varphi_1 < \varphi_a$ and $u_{oc} \leq u_1 \leq u_{ic}$  then
   $I \leftarrow I \cup [2t_a - t - t_1,\ u_1,\ \alpha + \varphi - \varphi_1]$
 if $\dot{u} > 0$  then $\Delta \leftarrow (e^2 < \mu\ ?\ 2\Delta_a - \Delta\ :\ \infty)$
 return $(\delta' = \delta + \Delta, I)$
```

Algorithm 1: PRECOMPUTE is based on (7), (8) and the properties illustrated in Fig. 2d. TRACERAY uses the properties and symmetries presented in Section 3.3.1.

In practice $\mathbb{D}(e,u)$ is defined only in a subset $\mathcal{D}$ of $[0,\infty[\times[0,1[$, diverges at $(\sqrt{\mu}^-, u_a)$ and $(\sqrt{\mu}^+, 1)$, and varies rapidly around $u = 2/3$ (because rays make more and more turns near the photon sphere before falling or escaping). For good precision we thus map $\mathcal{D}$ non-linearly into a square $[0,1]^2$ domain, designed to get more samples in these regions (see Appendix A).

Emitted light It now remains to compute the light emitted from all the stars in the beam's footprint on the celestial sphere, around $\mathbf{d}' = \cos\delta'\mathbf{e}'_x + \sin\delta'\mathbf{e}'_y$. For extended sources such as nebulae or galaxies, we can simply take advantage of anisotropic texture filtering, by storing these sources in a cube map. For punctual stars, however, this would yield unrealistically stretched star images. Our solution is to use a manually filtered cube map (see Fig. 3):

- Each texture element (or *texel*) stores the color and position (in the texel) of at most one star. A color *sum* (and not average) and luminosity-weighted position average is used for mip-mapping.
- We compute the beam's footprint in the cube map by using screen space partial derivatives (implemented with finite differences by the rendering pipeline). To avoid discontinuities at cube edges, we compute the partial derivatives $\partial_w\mathbf{d}'$ and $\partial_h\mathbf{d}'$ of $\mathbf{d}'$, and then compute the derivatives of the cube map face texture coordinates U, V analytically from them (*e.g.* for the +z face $U = d'^x/d'^z$, $\partial_w U = (\partial_w d'^x - U\partial_w d'^z)/d'^z$).
- We compute a mipmap level from the size of the footprint, fetch all the texels at this level in the footprint, and accumulate the col-

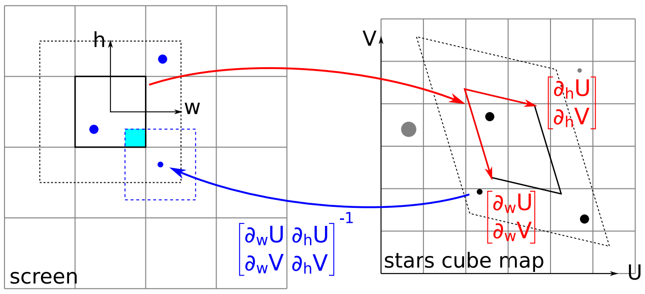
h
w
screen
V
U
stars cube map
$\begin{bmatrix}\partial_h U\\ \partial_h V\end{bmatrix}$
$\begin{bmatrix}\partial_w U\\ \partial_w V\end{bmatrix}$
$\begin{bmatrix}\partial_w U & \partial_h U\\ \partial_w V & \partial_h V\end{bmatrix}^{-1}$

Figure 3: Filtering. We compute the emitted light for a pixel by summing the light from the stars in its extended footprint (dashed parallelogram, computed with screen space partial derivatives) in a stars map, weighted by their pixel overlap area (cyan).

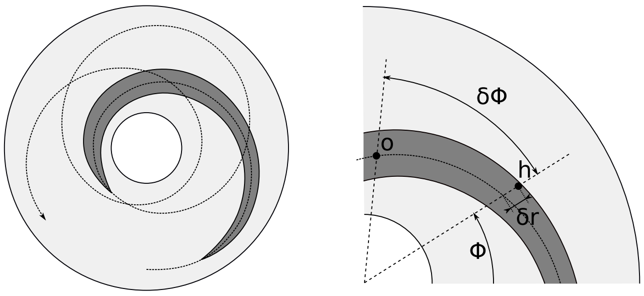

Figure 4: Accretion disc. We compute the density with a sum of linear particles moving along precessing orbits (left), whose density at a point h depends on its "distance" $d(\delta r, \delta\phi)$ from the particle center o (see Appendix B).

ors of the corresponding stars. For anti-aliasing, and to conserve the total intensity, we view each star as a 1×1 area and multiply its intensity with the area of its intersection with the considered screen pixel (*i.e.* with $f(w)f(h)$, where $f(x) = \max(1-|x|, 0)$ and(w,h) is the star's subpixel coordinates – the pixel domain being $[-\frac{1}{2}, \frac{1}{2}]^2$). Note that this requires to consider an extended footprint (see Fig. 3). In our implementation, we select the mipmap level so that it is at most 9×9 texels.

Note that this method approximates quadrilateral footprints with parallelograms, and does not use interpolation across mipmap levels. This would be easy to fix, but is not really necessary since our method already gives very good results.

3.3.3. Precomputations and beam tracing: accretion disc

As for stars, we compute the beam intersection(s) with the accretion disc by using a precomputed table. We then compute the light emitted there by using a simple procedurally animated disc model.

Intersections Let r_{ic} and r_{oc} be the inner and outer radius of the disc (with $r_{ic} \geq 3$, the innermost stable circular orbit [Las16]). Since the intersections can only occur at $\varphi = \alpha + m\pi$ (see Fig. 2b), we only need the function $u(\varphi)$ to check if there exist m such that $u_{oc} \triangleq r_{oc}^{-1} \leq u(\alpha + m\pi) \leq u_{ic} \triangleq r_{ic}^{-1} \leq 1/3$. For this we precompute $u(e,\varphi)$, *for light rays coming from infinity*, in a $\mathbb{U}(e,\varphi)$ table. At runtime, $u(\alpha + m\pi)$ can then be computed with $\mathbb{U}(e, \varphi_p + \alpha + m\pi)$, where φ_p is the camera position (which can be obtained from the deflection $\Delta_p = \mathbb{D}(e, 1/p^r)$ since $\Delta = \varphi + \delta - \pi$ – see Fig. 2d).

Note that we don't need $\mathbb{U}$ for all φ: we can stop when $u \geq 1/3$ since no intersection can occur between this point and the apsis, if any (and the rest can be deduced by symmetry). In practice, this means that we only need $\mathbb{U}(e,\varphi)$ for $0 \leq \varphi < \pi$, which has two consequences:

- we don't need to evaluate $\mathbb{U}(e, \varphi_p + \alpha + m\pi)$ for all m: in fact we only need $\mathbb{U}(e, (\varphi_p + \alpha) \mod \pi)$,
- there can be at most two intersections: one on each symmetric part of the ray (if it does not fall into the black hole).

The algorithms to precompute $\mathbb{U}(e,\varphi)$ and to find the accretion disc intersections (u_0, φ_0), (u_1, φ_1) follow from these properties, and those of Section 3.3.1, and are shown in Algorithm 1. As for $\mathbb{D}$, we map $\mathbb{U}$'s domain non-linearly into $[0,1]^2$ to get good precision in large gradient areas (see Appendix A).

Finally, note that for an animated disc we also need to compute the *retarded time* between the intersections and the camera. For this we also precompute and store t – using $dt/d\varphi = e/(u^2 - u^3)$, from (7) – in $\mathbb{D}$ and $\mathbb{U}$. This allows the computation of the retarted times t_0, t_1 at the disc intersections, as shown in Algorithm 1.

Emitted light It now remains to compute the light emitted by the accretion disc at the intersection points. For this we use the light emitted by a black body at temperature $T(u)$, times the disc density. We use $T^4(u) \propto u^3(1 - \sqrt{3u})$ [Las16], and compute the density with a sum of procedural particles moving along quasi-circular precessing orbits (see Fig. 4 and Appendix B).

3.4. Shading

Due to gravitational effects, the light received at the camera is different from the emitted light, computed above. We present these effects below, and explain how we compute them.

3.4.1. Gravitational lensing effects

Due to gravitational lensing, the light emitted by a punctual star is received amplified by a factor Ω/Ω', where Ω (resp. Ω') is the beam's solid angle at the camera (resp. emitter) [VE99]. We compute it by using the screen space partial derivatives of the beam directions at the camera and at the emitter:

$$\frac{\Omega}{\Omega'} = \frac{||\partial_w \mathbf{q} \wedge \partial_h \mathbf{q}||}{||\partial_w \mathbf{d}' \wedge \partial_h \mathbf{d}'||} \tag{10}$$

where $\mathbf{q}$ is the normalized $[q^w, q^h, -f]^\top$ vector. Note that this does not apply to area light sources, because the beam's subtended area varies in inverse proportion.

3.4.2. Doppler and beaming effects

Due to gravitational and relativistic time dilation and length contraction effects, the frequency ν of the received light differs from the emitted frequency ν'. The ratio is given by [PHL17]

$$\frac{\nu}{\nu'} = \frac{g(\vec{k}, \vec{l})}{g'(\vec{k}', \vec{l}')} = \frac{g_{tt}k^t l^t + g_{rr}k^r l^r + g_{\theta\theta}k^\theta l^\theta + g_{\phi\phi}k^\phi l^\phi}{g'_{tt}k'^t l'^t + g'_{rr}k'^r l'^r + g'_{\theta\theta}k'^\theta l'^\theta + g'_{\phi\phi}k'^\phi l'^\phi} \tag{11}$$

where g_{jj} are the metric coefficients at the receiver, $\vec{k}$ is the 4-velocity of the receiver, $\vec{l}$ is the tangent 4-vector $\mathrm{d}s/\mathrm{d}\sigma$, at the receiver, of the light ray curve $s(\sigma)$, and g'_{jj}, $\vec{k}'$ and $\vec{l}'$ are the corresponding emitter quantities. We thus need $\vec{k}$, $\vec{k}'$, $\vec{l}$, and $\vec{l}'$.

In *rotated* Schwarzschild coordinates, $\vec{l}$ and $\vec{l}'$ are given by $[\mathrm{d}t/\mathrm{d}\sigma, \mathrm{d}r/\mathrm{d}\sigma, \mathrm{d}\vartheta/\mathrm{d}\sigma, \mathrm{d}\varphi/\mathrm{d}\sigma]^\top$. Using (7), this gives

$$\vec{l} = \left[\frac{e}{1-u}, -\dot{u}, 0, u^2\right]^\top \quad \vec{l}' = \left[\frac{e}{1-u'}, -\dot{u}', 0, u'^2\right]^\top \tag{12}$$

where e is the *negative* root of (8) – for the actual light rays $\mathrm{d}\varphi/\mathrm{d}t$ is negative, unlike in the previous section where rays were traced backward. In *non-rotated* Schwarzschild coordinates, we have

$$\vec{k}' = \left[\sqrt{\frac{2}{2-3u'}}, 0, 0, \sqrt{\frac{u'^3}{2-3u'}}\right]^\top \tag{13}$$

for the accretion disc (if we assume a circular motion) [PHL17], $\vec{k}' = [1,0,0,0]^\top$ for static stars, and $\vec{k} = \vec{e}_\tau$ for the camera. Finally, to compute $g(\vec{k},\vec{l})$ and $g(\vec{k}',\vec{l}')$, we need the corresponding *rotated* coordinates: k^t and k^r are unchanged, k^ϑ is not needed since $l^\vartheta = 0$ and $k^\varphi = \vec{k}\cdot\vec{\partial}_\varphi/r^2 = u\mathbf{k}\cdot\mathbf{e}_{y'}$ (and similarly for $\vec{k}'$). For instance, for the accretion disc, we get $k'^\varphi = k'^\phi \mathbf{e}_z \cdot \mathbf{e}_{z'}$ and

$$g(\vec{k}',\vec{l}') = e\sqrt{\frac{2}{2-3u'}} - \sqrt{\frac{u'^3}{2-3u'}}\mathbf{e}_z\cdot\mathbf{e}_{z'} \tag{14}$$

The above Doppler effect has an associated *beaming* effect: the received intensity differs from the emitted one because, from Liouville's theorem, $I(\nu)/\nu^3$ is invariant [MTW73]. In terms of wavelength $\lambda \triangleq \nu^{-1}$, and with $I(\lambda)\mathrm{d}\lambda = I(\nu)\mathrm{d}\nu$, this gives $I(\lambda) = (\lambda'/\lambda)^5 I(\lambda')$. For black bodies the two effects result in a temperature shift $T = (\nu/\nu')T'$. For other light sources however, that we want to support, the result is more complex. We thus precompute it in a 3D texture $\mathbb{C}(xy, D)$, for each chromaticity xy and Doppler factor $D \triangleq \nu/\nu'$.

To this end we need to choose a spectrum for each chromaticity, among the infinite number of possible spectrums. For simplicity and to get black body spectrums for black body colors, we use spectrums of the form $I(\lambda') = B_T(\lambda')(1 - a_1 A_1(\lambda') - a_2 A_2(\lambda'))$, where B_T is the black body spectrum for temperature T, T is the correlated color temperature, and A_1 and A_2 are two fixed absorption spectrums. A linear system gives a_1 and a_2 from the xy chromaticity, and the Doppler and beaming effects give CIE XYZ colors that we precompute with

$$\mathbb{C}(xy, D) = D^5 \frac{\int I(D\lambda)\,[\bar{x}(\lambda), \bar{y}(\lambda), \bar{z}(\lambda)]^\top\,\mathrm{d}\lambda}{\int I(\lambda)(\bar{x}(\lambda) + \bar{y}(\lambda) + \bar{z}(\lambda))\,\mathrm{d}\lambda} \tag{15}$$

where $\bar{x}$, $\bar{y}$ and $\bar{z}$ are the CIE color matching functions. At runtime, the emitted XYZ color computed in Section 3.3 is transformed into the received color $(X+Y+Z)\mathbb{C}(xy, \nu/\nu')$ with (11).

3.4.3. Lens glare effects

Due to light scattering and diffraction inside the eye, haloes appear around very bright light sources, which would otherwise be hard to distinguish from fainter sources. For this reason we apply a bloom shader effect on the final image, before tone-mapping. We use a series of small support filter kernels on mipmaps of the full image, approximating a point spread function from [SIS+95], but more precise methods are possible too [HESL11].

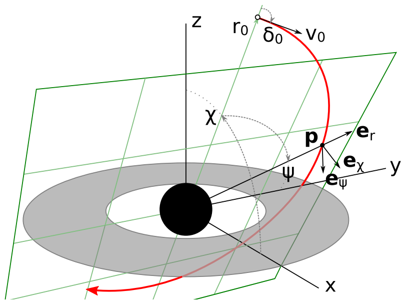

Figure 5: Camera orbit. The orbit, in red, is specified by an inclination χ and the initial conditions r_0, δ_0 and v_0 (see Appendix C).

4. Implementation

We implemented our method in C++ for the precomputations, and WebGL 2 for the rendering. The full source code and an online demo are available at `https://github.com/ebruneton/black_hole_shader`. The demo simulates a static or freely falling observer (see Fig. 5) and allows the user to set various parameters (black hole mass, disc temperature and density, camera orbit, etc).

We precompute $\mathbb{D}(e,u)$ and $\mathbb{U}(e,\varphi)$ in 512×512 and 64×32 RG32F textures, respectively. PRECOMPUTE takes about 11 seconds on a 3.2GHz Intel Core i5-6500 CPU, with $\varepsilon = 10^{-5}$, and unit tests show that TRACERAY results δ', u_0 and u_1 are within 10^{-3} of reference values computed without intermediate textures. We precompute $\mathbb{C}(xy,D)$ in a $64\times32\times64$ RGB32F texture, in about 7 seconds.

We also precompute two $6\times2048\times2048$ RGB9E5 cubemaps for the area and punctual light sources, from the Gaia DR2 [Gai18] and Tycho 2 [HFM+00] star catalogs. This requires downloading and processing 550GB of compressed data, which can take a day, and yields ≈ 3.6 million punctual light sources. In our implementation we don't store a sub-texel position for each star: instead, we use a hash of its color to compute a pseudo-random position.

5. Results and discussion

Some results obtained with our method are shown in Fig. 1. They are rendered in Full HD at about 150 fps on an NVidia GeForce GTX 960 (see Table 1).

The benefit of our precomputed tables can be measured by replacing TRACERAY with a ray-marching method integrating (8) numerically (and keeping everything else unchanged). To get the same performance as with the precomputed tables, only 25 integration steps at most can be used, and stars end up at several degrees from their correct positions. To get (almost) the same precision, up to 1000 integration steps must be used, and the framerate drops to about 45 fps (see method M3 in Table 1).

View	M1	M2	M3	Stars	Disc	Bloom
1	150	64	43	2.99	0.82	1.11
2	160	59	45	2.64	0.78	1.15
3	153	67	45	2.99	0.69	1.10
4	114	64	41	4.14	1.87	1.08

Table 1: The framerate obtained with our method (M1), with stars rendered with [MW10] (M2), and with TRACERAY replaced with ray-marching (M3), on the views shown in Fig.1 and numbered from left to right (1920 × 1080p, NVidia GeForce GTX 960). The other columns give the time used per frame (in milliseconds), to render the stars, the accretion disc and the bloom effect with our method.

The benefit of our custom texture filtering method to render the stars can be measured by replacing it with the method from [MW10] (and keeping everything else unchanged). Each of the 3.6 million stars is then rendered with n 2×2 anti-aliased point primitives (we used $n=2$ in our tests). The point position is computed with a lookup in a 4096×4096 precomputed texture. This gives about 65 fps (see method M2 in Table 1). The main bottleneck is due to the fact that, inside the region of Einstein rings, many stars project into the same pixel, leading to a lot of overdraw.

A limitation of our method is that views from inside the horizon are not supported. Indeed, since no observer can remain static in this region, we can no longer specify the camera and initialize light beams by using a reference static observer. Also the Schwarzschild metric diverges at the horizon. However, by using different coordinates, as in [MW10], we believe that our method can be extended to support this case.

Another limitation is that motion blur, which is necessary for very high quality animations, is not supported. Also, because of the approximations in our custom texture filtering method, in some cases a few stars flicker when the camera moves. Fixing both quality issues might be easier by extending [MW10] rather than by extending our method, at the price of decreased performance.

Finally, another limitation of our method, and of [MW10] as well, is that rotating black holes are not supported. Because they are only axially symmetric, the "inverse ray tracing" approach of [MW10] would probably be hard to generalize to this case. Our precomputed ray tracing method, on the other hand, could in principle be generalized to 4D tables containing the deflected direction and the accretion disc intersections of any ray (specified with 2 position and 2 direction parameters). In practice however, obtaining precise 4D tables of reasonable size might be hard.

6. Conclusion

We have presented a beam tracing method relying on small precomputed textures to render real-time high-quality images of non rotating black holes. Our method is simple to implement and achieves high frame rates. Extending it to views from inside the horizon, and to rotating black holes, if this is possible, is left as future work.

Appendix A: Texture mappings

We store $\mathbb{D}(e,u)$ at texel coordinates

$$\left[\frac{1}{2}-\sqrt{-\log(1-e^2/\mu)/50},\ 1-\sqrt{1-u/u_a}\right]\ \text{if}\ e^2<\mu$$

$$\left[\frac{1}{2}+\sqrt{-\log(1-\mu/e^2)/50},\ \frac{\sqrt{2/3}\pm\sqrt{|u-2/3|}}{\sqrt{2/3}+\sqrt{1/3}}\right]\ \text{otherwise}$$

where $\pm$ is the sign of $u-2/3$, and $\mathbb{U}(e,\varphi)$ at texel coordinates

$$\left[\frac{1}{1+6e^2},\ \frac{\varphi}{3}\frac{1+6e^3}{1+e^2}\right]$$

Appendix B: Disc particles

The orbit of a point particle in the accretion disc is given by

$$u=u_1+(u_2-u_1)\mathrm{sn}^2\left(\frac{\phi}{2}\sqrt{u_3-u_1},\kappa\right),\quad \kappa=\sqrt{\frac{u_2-u_1}{u_3-u_1}}$$

where sn is the Jacobi-sn function, $u_1\le u_2\le 1/3$, and $u_3=1-u_1-u_2$ [Dar59]. For quasi-circular orbits this can be approximated with

$$\begin{aligned}u(t)&\approx u_1+(u_2-u_1)\sin^2\left(\frac{\pi}{4K}\phi(t)\sqrt{u_3-u_1}\right)\\ \phi(t)&\approx\sqrt{\frac{\bar{u}^3}{2}}\,t+\phi_0,\quad K=\int_0^1\frac{\mathrm{d}x}{\sqrt{(1-x^2)(1-\kappa^2x^2)}}\end{aligned}\tag{16}$$

where $\bar{u}=(u_1+u_2)/2$ since, for circular orbits, (13) gives $\mathrm{d}\phi/\mathrm{d}t=\sqrt{u^3/2}$. For a linear particle parameterized by $a\in[0,2\pi[$, the position $u_a(t),\phi_a(t)$ of a point a is obtained by replacing $\phi(t)$ with $\phi_a(t)=a+\phi(t)$ in (16). Thus, given a ray hit point h^t,h^r,h^ϕ, we compute the parameter a of the "nearest" particle point with $a=h^\phi-\phi(h^t)\mod 2\pi$. We then compute the "distance" between h and the linear particle center (at $a=\pi$) with $d^2=(a/\pi-1)^2+(h^r-1/u_a(h^t))^2$. We finally compute the particle density at h with a smoothly decreasing function of d.

Appendix C: Camera orbit

Position The camera position is specified by its polar coordinates (r,ψ) in an orbital plane with inclination χ (see Fig. 5). In Schwarzschild coordinates adapted to this orbital plane the camera 4-velocity is $\vec{k}_c=[\frac{\mathrm{d}t}{\mathrm{d}\tau},\frac{\mathrm{d}r}{\mathrm{d}\tau},0,\frac{\mathrm{d}\psi}{\mathrm{d}\tau}]^\top=[\frac{e}{1-u},\frac{\mathrm{d}r}{\mathrm{d}\tau},0,lu^2]^\top$, where e and l are two constants of motion. Substituting this in (1) gives

$$\left(\frac{\mathrm{d}r}{\mathrm{d}\tau}\right)^2=e^2+l^2u^3-l^2u^2+u-1\Rightarrow\frac{\mathrm{d}^2r}{\mathrm{d}\tau^2}=\frac{2l^2u^3-3l^2u^4-u^2}{2}$$

We use these relations to update the coordinates (t,r,ψ) at each proper time step $\mathrm{d}\tau$. The corresponding Cartesian coordinates are

$$r\begin{bmatrix}\cos\chi\cos\psi\\ \sin\psi\\ \sin\chi\cos\psi\end{bmatrix}=p^r\begin{bmatrix}\sin p^\theta\cos p^\phi\\ \sin p^\theta\sin p^\phi\\ \cos p^\theta\end{bmatrix}$$

from which we deduce the Schwarzschild coordinates $p^t=t$, $p^r=r$, $p^\theta=\arccos(\cos\psi\sin\chi)$ and $p^\phi=\arctan2(\sin\psi,\cos\chi\cos\psi)$.

The above relations require the constants of motion e and l. We compute them from the initial position, direction and speed, noted

$r_0 = 1/u_0$, δ_0, and v_0 (see Fig. 5). We get $e^2 = (1-u_0)/(1-v_0^2)$ from the Lorentz factor $\gamma = g(\vec{k}_c, \vec{k}_s) = e/\sqrt{1-u} = 1/\sqrt{1-v^2}$, where $\vec{k}_s = [1/\sqrt{1-u}, 0, 0, 0]^\top$ is the 4-velocity of a static observer. Finally, using $\tan\delta = r\mathrm{d}\psi/\mathrm{d}r$ and the above equations, we get $l^2 = (e^2 + u_0 - 1)/(u_0^2(1 - u_0 + \cot^2\delta_0))$.

Lorentz transform We compute the Lorentz transform Λ from the static observer basis $\vec{e}_t, \vec{e}_r, \vec{e}_\theta, \vec{e}_\phi$ to the camera basis $\vec{e}_\tau, \vec{e}_w, \vec{e}_h, \vec{e}_d$ by using the intermediate orthonormal basis $\vec{e}_t, \vec{e}_r, \vec{e}_\chi, \vec{e}_\psi$ where $\mathbf{e}_\chi$ is the orbital plane's normal (see Fig. 5), as follows.

Let ${R_k}^j$ be the rotation matrix from $\vec{e}_t, \vec{e}_r, \vec{e}_\theta, \vec{e}_\phi$ to $\vec{e}_t, \vec{e}_r, \vec{e}_\chi, \vec{e}_\psi$: $\vec{e}_k = {R_k}^j \vec{e}_j$, $k \in \{t, r, \chi, \psi\}$, $j \in \{t, r, \theta, \phi\}$. Its lower right block is

$$\begin{bmatrix} \mathbf{e}_\chi \cdot \mathbf{e}_\theta & \mathbf{e}_\chi \cdot \mathbf{e}_\phi \\ -\mathbf{e}_\chi \cdot \mathbf{e}_\phi & \mathbf{e}_\chi \cdot \mathbf{e}_\theta \end{bmatrix} \text{ with } \begin{array}{l} \mathbf{e}_\chi \cdot \mathbf{e}_\theta = \sin\chi \cos p^\theta \cos p^\phi + \cos\chi \sin p^\theta \\ \mathbf{e}_\chi \cdot \mathbf{e}_\phi = \sin\chi \sin p^\phi \end{array}$$

In the $\vec{e}_t, \vec{e}_r, \vec{e}_\chi, \vec{e}_\psi$ basis the camera 4-velocity and speed are:

$$\vec{k}_c = \left[\sqrt{1-u}\frac{\mathrm{d}t}{\mathrm{d}\tau}, \frac{1}{\sqrt{1-u}}\frac{\mathrm{d}r}{\mathrm{d}\tau}, 0, \frac{1}{u}\frac{\mathrm{d}\psi}{\mathrm{d}\tau} \right]^\top$$

$$\mathbf{v} = \left[\frac{1}{1-u}\frac{\mathrm{d}r}{\mathrm{d}\tau} \Big/ \frac{\mathrm{d}t}{\mathrm{d}\tau}, 0, \frac{1}{u\sqrt{1-u}}\frac{\mathrm{d}\psi}{\mathrm{d}\tau} \Big/ \frac{\mathrm{d}t}{\mathrm{d}\tau} \right]^\top$$

A reference frame for the camera is thus $\vec{e}_{k'} \triangleq B(\mathbf{v})_{k'}{}^k \vec{e}_k$, $k' \in \{t', r', \chi', \psi'\}$, where $B(\mathbf{v})$ is a Lorentz boost: $B(\mathbf{v})_{k'}{}^k = B(-\mathbf{v})^{k'}{}_k$, $\mathrm{d}p^{k'} = B(\mathbf{v})^{k'}{}_k \mathrm{d}p^k$ [Wei72].

Finally, let ${O_i}^{k'}$ be a user specified rotation matrix. We then compute Λ with ${\Lambda_i}^j = {O_i}^{k'} B(\mathbf{v})_{k'}{}^k {R_k}^j$. Note that this procedure assumes that the camera orientation is actively controlled, *i.e.* is not freely evolving as a gyroscope would be.

References

[AAA+16] B.P Abbott, R. Abbott, Thomas Abbott, Matthew Abernathy, Fausto Acernese, K. Ackley, C. Adams, Teneisha Adams, Paolo Addesso, R.X Adhikari, Vaishali Adya, C. Affeldt, M. Agathos, Kazuhiro Agatsuma, Nishu Aggarwal, Odylio Aguiar, L. Aiello, Anirban Ain, P. Ajith, and John Zweizig. Observation of gravitational waves from a binary black hole merger. *Physical Review Letters*, 116, 2 2016.

[Dar59] Charles Galton Darwin. The gravity field of a particle. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 249(1257):180–194, 1959.

[EHT19] EHT Collaboration et al. First M87 event horizon telescope results. I. The shadow of the supermassive black hole. *The Astrophysical Journal Letters*, 875:1, 2019.

[Gai18] Gaia Collaboration et al. Gaia Data Release 2. Summary of the contents and survey properties. *Astronomy and Astrophysics*, 616:A1, 8 2018.

[Ham14] Andrew Hamilton. Black hole flight simulator. 2014.

[HESL11] Matthias B. Hullin, Elmar Eisemann, Hans-Peter Seidel, and Sungkil Lee. Physically-based real-time lens flare rendering. *ACM Trans. Graph. (Proc. SIGGRAPH 2011)*, 30(4):108:1–108:9, 2011.

[HFM+00] E. Høg, C. Fabricius, V. V. Makarov, S. Urban, T. Corbin, G. Wycoff, U. Bastian, P. Schwekendiek, and A. Wicenec. The Tycho-2 catalogue of the 2.5 million brightest stars. *Astronomy and Astrophysics*, 355:L27–L30, March 2000.

[JvTFT15] Oliver James, Eugénie von Tunzelmann, Paul Franklin, and Kip S Thorne. Gravitational lensing by spinning black holes in astrophysics, and in the movie Interstellar. *Classical and Quantum Gravity*, 32(6):065001, 2 2015.

[Las16] Jean-Pierre Lasota. Black hole accretion discs. In Cosimo Bambi, editor, *Astrophysics of Black Holes: From Fundamental Aspects to Latest Developments*, pages 1–60. Springer Berlin Heidelberg, 2016.

[LHT+18] M. Liska, C. Hesp, A. Tchekhovskoy, A. Ingram, M. van der Klis, and S. Markoff. Formation of precessing jets by tilted black hole discs in 3D general relativistic MHD simulations. *Monthly Notices of the Royal Astronomical Society*, 474(1):L81–L85, 2 2018.

[Lum79] J. P. Luminet. Image of a spherical black hole with thin accretion disk. *Astronomy and Astrophysics*, 75:228–235, 5 1979.

[Lum19] Jean-Pierre Luminet. An illustrated history of black hole imaging : personal recollections (1972-2002). 3 2019.

[MB11] Thomas Müller and Sebastian Boblest. Visualizing circular motion around a Schwarzschild black hole. *American Journal of Physics*, 79:63–73, 1 2011.

[MF12] Thomas Müller and Jörg Frauendiener. Interactive visualization of a thin disc around a Schwarzschild black hole. *European Journal of Physics*, 33(4):955–963, 5 2012.

[MTW73] C. W. Misner, K. S. Thorne, and J. A. Wheeler. *Gravitation*. 1973.

[MW10] Thomas Müller and Daniel Weiskopf. Distortion of the stellar sky by a Schwarzschild black hole. *American Journal of Physics*, 78:204–214, 1 2010.

[PHL17] Dennis Philipp, Eva Hackmann, and Claus Laemmerzahl. Redshift and frequency comparison in Schwarzschild spacetime. 11 2017.

[Ria14] Alain Riazuelo. Simulation of starlight lensed by a camera orbiting a Schwarzschild black hole. 2014.

[SIS+95] Greg Spencer, Taligent Inc, Peter Shirley, Kurt Zimmerman, and Donald Greenberg. Physically-based glare effects for digital images. *Computer Graphics (Proc. SIGGRAPH 1995)*, 29:325–334, 08 1995.

[VE99] Kumar Virbhadra and George Ellis. Schwarzschild black hole lensing. *Physical Review D*, 62, 04 1999.

[Wei72] S. Weinberg. *Gravitation and cosmology: principles and applications of the general theory of relativity*. Wiley, 1972.