

General relativistic ray tracing

Computer simulations in physics

KÜRTI ZOLTÁN

2021.10.22.

Contents

| | | |
|----------|---|----------|
| 1 | Introduction | 3 |
| 2 | Theoretical background | 3 |
| 2.1 | Light rays | 3 |
| 2.2 | Observers in general relativity | 4 |
| 2.3 | The Schwarzschild black hole | 4 |
| 3 | Program description and results | 5 |
| 3.1 | General structure | 5 |
| 3.1.1 | Vierbein calculation | 6 |
| 3.1.2 | Initial condition generation | 6 |
| 3.1.3 | Differential equation solver | 7 |
| 3.1.4 | θ - ϕ checkerboard | 7 |
| 3.1.5 | Redshift | 7 |
| 4 | Conclusion | 8 |

1 Introduction

Modeling light transport in the framework of general relativity is an important topic in physics. Modern observatories allow multiple different measurements that can be compared to predictions of general relativity about light transport. [1, 2] These type of measurements are among the most important experimental confirmations of general relativity. Gravitational lensing can generally be divided into two categories, strong and weak gravitational lensing. In our work we focus on strong gravitational lensing using numerical methods. This category of lensing is important for understanding observations about distant massive compact objects like neutron stars and black holes. Resolution of light originating from the strongly lensing region of nearby supermassive black holes is now possible thanks to the Event Horizon Telescope collaboration.

2 Theoretical background

2.1 Light rays

We will use $(+, -, -, -)$ sign convention during this project. In the document Greek indices run from 0 to 3, latin indices in general run from 1 to 3 except when they index a vierbein vector in which case they run from 0 to 3, the 0th vector being timelike.

We modeled light in the geometric optics approximation, meaning light is described by light rays. These light rays in general relativity correspond to null geodesics.

The equation to be solved is therefore

$$\frac{d^2 x^\mu}{dt^2} + \Gamma^\mu_{\alpha\beta} \frac{dx^\alpha}{dt} \frac{dx^\beta}{dt} = 0, \quad (1)$$

with initial conditions $x^\mu(0) = x_0^\mu$ and $\frac{dx^\mu}{dt}(0) = v^\mu$ where $v^\mu v_\mu = 0$ in order for the geodesic to be null. In this case we want to integrate this equation backwards in time since the information we have is the direction and position of the ray when it hit the camera. From this information we have to determine the first intersection of the light ray of an object of interest that we want to visualize. This can be achieved with forward integration methods using $v_0^0 < 0$.

2.2 Observers in general relativity

Observers or cameras in this case can be described with a point p_0 in spacetime and a vierbein e_a^μ at that point p_0 . e_0^μ is a future timelike vector, corresponding to the 4-velocity of the observer. The rest of the four vierbein vectors represent the x y and z coordinates of the observer. For the camera x is the forward direction, y is the up direction and z is the right direction. These four vectors have to be orthogonal to each other, this way they describe a local coordinate system that is like the usual (t, x, y, z) coordinate system used in special relativity:

$$e_a^\mu e_{b\mu} = \eta_{ab} \quad (2)$$

where $\eta_{ab} = \text{diag}(1, -1, -1, -1)$ is the metric tensor from special relativity in (t, x, y, z) coordinates.

Int this local coordinate system the construction of a virtual screen is possible knowing the screen resolution and field of view. Choosing a pixel on this screen the corresponding light ray will have spatial velocity components equal to that of the pixel's, and a negative time component chosen such that the velocity vector is a past null vector. This velocity together with the position of the camera are the initial conditions of the geodesic equation. This formalism automatically includes all special relativistic effects on the directions of light rays, relativistic aberration is well illustrated by images generated based on this camera system.

2.3 The Schwarzschild black hole

Einstein's equations can be solved for the vacuum case assuming spherical and time translation symmetry. This solution is called the Schwarzschild solution. Many coordinate systems are used to describe this solution. In our choice the metric tensor is

$$ds^2 = \left(1 - \frac{2GM}{rc^2}\right) c^2 dt^2 - \frac{1}{1 - \frac{2GM}{rc^2}} dr^2 - r^2 d\theta^2 - r^2 \sin^2(\theta) d\phi^2. \quad (3)$$

This choice has the advantage that the formulas are easy to compute, everything is expressed in elementary functions and that for large values of r it is a good match for the usual spherical coordinates, which will be useful to approximate the intersection of light rays with the celestial sphere. The main disadvantage of this coordinate system is that it is singular at the event horizon, meaning that numeric solutions can't cross it and the

stepsize required to remain precise approach 0. This is not a problem if the camera stays outside the event horizon.

The non-zero components of the Christoffel-symbols up to symmetry in the last two indices are

$$\begin{aligned}
\Gamma_{rt}^t &= -\Gamma_{rr}^r = \frac{r_s}{2r(r-r_s)} \\
\Gamma_{tt}^r &= \frac{r_s(r-r_s)}{2r^3} \\
\Gamma_{\phi\phi}^r &= (r_s-r)\sin^2(\theta) \\
\Gamma_{\theta\theta}^r &= r_s-r \\
\Gamma_{r\theta}^\theta &= \Gamma_{r\phi}^\phi = \frac{1}{r} \\
\Gamma_{\phi\phi}^\theta &= -\sin(\theta)\cos(\theta) \\
\Gamma_{\theta\phi}^\phi &= \cot(\theta).
\end{aligned} \tag{4}$$

3 Program description and results

The program was written in C++, and uses only two external libraries, LibTIFF to write images to disc and OpenMP to parallelize for loops. The code can be downloaded from <https://github.com/KurtiZoltan/CompSim1>.

The program was designed to be easily extendable, any spacetime geometry can be represented just by implementing a child class of the pure virtual Spacetime class. An arbitrary number of displayable objects can be used. Their geometry is defined by scalar valued functions of the spacetime coordinates, negative values indicating the inside of the objects and zero the surface. The pixel color of a hit is calculated by another function that is a function of the spacetime coordinates along with the velocity vector of the light ray at the time of the hit. These two functions are provided by the user and added to an object of class Objects containing data relevant to intersection calculations.

3.1 General structure

The user has to provide an implementation of the abstract class Spacetime, provide the intersection and color functions corresponding to each object, give some initial data about the reference frame and camera. After these steps the program is ready to render the image seen by the camera.

3.1.1 Vierbein calculation

First based on the initial data provided to the reference frame, the proper vierbein corresponding to the observer is calculated. This requires the metric tensor with which using orthogonal projections the `lookAt` vector is projected with the `time` vector to obtain the `forward` vector. The `up` vector is both projected with the `time` and `forward` vectors to obtain the true `up` vector. Finally the four basis vectors are all projected with all three previous vectors, `time`, `forward`, `up`. The biggest norm out of theses is choosen for maximal numeric accuracy. This vector is parallel to the `right` direction. The sign is determined by the determinant of the four vectors, this ensures that the vierbein is right handed. As a final step the vectors are normalized, this completes the calculation of the vierbein representing the observer.

3.1.2 Initial condition generation

The next step is the determination of the initial conditions corresponding to the pixels of the camera. The initial position is dimply the position of the camera, while the initial velocity is calculated according to the following formula:

$$\text{dir} = \frac{\text{forward} + u \text{ up} + r \text{ right}}{\sqrt{1 + u^2 + r^2}} - \text{time}, \quad (5)$$

where u and r represent the pixel coordinates calculated based on the field of view and the resolution of the image,

$$\begin{aligned} r &= \tan\left(\frac{\text{FOV}}{2}\right) \left(\frac{2x}{\text{width} - 1} - 1\right) \\ u &= \tan\left(\frac{\text{FOV}}{2}\right) \left(\frac{2y}{\text{height} - 1} - 1\right) \frac{\text{height}}{\text{width}} \end{aligned} \quad (6)$$

This means that the time component of the light ray in the observer's reference frame is always 1, wich will be useful when calculating redshifts. The initial conditions are stored in an array. This makes it possible to move the initial condition data to the GPU in one step, allowing the use of massive parallelism provided by GPUs to integrate the equations of motion. However this is not done in the software, all calculations are done on the CPU.

3.1.3 Differential equation solver

The next step is to integrate the equations of motion for each initial condition, while checking for possible hits with the user provided objects. If a hit is found, the color is computed with the user provided color function corresponding to the hit and returned. The differential equation solver is capable of implementing a general forward Runge-Kutta method based on the A matrix, b and c vectors to solve the equation $y'(t) = f(t, y)$. The A matrix contains the information about how to calculate the y value for the current f evaluation based on the previous f estimates, the b vector contains the weights about how to combine the derivative estimates to give the final estimate used for the time step, and the c vector contains the t parameter increments to be used in each step evaluating f . The notation used for A b and c agrees with the notation used in [3].

3.1.4 θ - ϕ checkerboard

3.1.5 Redshift

The calculation of both gravitational and special relativistic red/blueshift is possible via comparing the time component of the velocity vector in the observer reference frame and in the reference frame corresponding to the object that is hit. These time components of velocity vectors are linearly proportional to the frequency of the light (they are just the ω in a local plane wave corresponding to the ray), therefore

$$\frac{f_{\text{observed}}}{f_{\text{source}}} = \frac{e_{o0}{}^{\mu} v_{o\mu}}{e_{s0}{}^{\mu} v_{s\mu}}. \quad (7)$$

$e_{o0}{}^{\mu}$ and $e_{s0}{}^{\mu}$ are the 0th vierbein vectors of the observer and source respectively, $v_o{}^{\mu}$ and $v_s{}^{\mu}$ are the velocity vectors of the light ray at the observer and source respectively, connected by parallel transport along the light ray. This does not mean any additional calculations, since the velocity vector of a geodesic is automatically parallel transported along the geodesic, so the velocity vector in the initial condition and at the last timestep of the integration are sufficient. If the initial condition generation algorithm guarantees the length of the time component in the observer's reference frame, redshift can be easily calculated just with the result of numerical integration and the 4-velocity vector of the source at the hit point.

4 Conclusion

References

- [1] Kevin P. Rauch and Roger D. Blandford. *Optical Caustics in a Kerr Spacetime and the Origin of Rapid X-Ray Variability in Active Galactic Nuclei*. *apj*, 421:46, 1994
- [2] Ed Fomalont, Sergei Kopeikin, Dayton Jones, Mareki Honma, and Oleg Titov. *Recent vlba/vera/ivs tests of general relativity*. *Proceedings of the International Astronomical Union*, 5(S261):291–295, 2009
- [3] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, USA, second edition, 1992