Visualizing Black Hole Geodesics With Ray Casting

William Luksha
August 11, 2023

Primary Advisor: Germano Iannacchione
Co-Advisor: William Sanguinet

Abstract

This MQP used FANTASY, a prebuilt symplectic geodesic integrator for the purposes of studying complex spacetimes, to compute the paths of light rays under Schwartzschild and Kerr metrics in order to ray cast renderings of black holes. It outlines the code built in order to interface with FANTASY and generate first-person POV renderings of these black holes using any environment image chosen. The resulting images are studied in order to show known phenomena such as Einstein rings to show the code's accuracy.

Contents

1 Introduction

What does a black hole look like? On April 10th, 2019 the EHT collaboration published a series of papers outlining their groundbreaking work imaging the M87 black hole [1]. They then went on to publish more results on May 12th, 2022 showing the results of their imaging Sagittarius A*, the black hole located at the center of the Milky Way galaxy [3].

Unlike conventional astronomy, these pictures involved much more than just pointing a large telescope at the sky. Using techniques such as "Very Long Baseline Interferometry" (VLBI) -which combined data from 11 satellite stations across the globe- as well as machine learning to synthesize an image from the raw data, EHT also had to use many complex simulations to ensure their final images were accurate. these simulations, GRMHDS (General Relativistic Magneto-Hydrodynamical Simulations) and specifically one named BHAC (Black Hole Accretion Code) were first used to create synthetic data for the machine learning to train with. This way, when the real data came in, it could quickly and accurately produce reliable images which would be true to the real black hole.

The motivation for this MQP comes from the simulations used in the EHT's endeavors. While reading about the simulations used was exciting, I wanted to have first-hand experience with them. Despite several problems arising in applying the open source BHAC code early on in the school year, and even attempting to create my own, I found myself exploring a geodesic integrator called FANTASY [2]. While not used by the EHT collaboration, it proved to be a very valuable tool in exploring more general spacetimes, as well as being more straightforward to apply. It is also a symplectic integrator, meaning that it is able to preserve quantities such as the Hamiltonian and keep them within a bounded error. This characteristic is very valuable in producing physically accurate trajectories. When exploring this code, I decided to create an interface with it which would allow me to study the visual phenomena produced by complex spacetime metrics.

The two main objectives of this MQP are as follows:

1. Create python code which takes any desired simulation setup, interfaces with FANTASY, and produces images from the simulation results.
2. Use these results to show and study phenomena such as gravitational lensing and Einstein rings.

This MQP will start with a background of the important physics, mathematics, and computational concepts needed to understand the extent of the work completed and to judge the completion of the previously listed goals. It will then continue into the methods section which outlines the tools I used, the code I wrote, and the methods employed by the code. The results section shows what was produced, and the analysis/discussion section will go over those results in more detail. Another less formal goal is to create code which can be built off of, and used for further studies into the nature of complex spacetime metrics.

2 Background

2.1 Spacetime and Geodesics

This section of the background will serve to give a basic introduction to special relativity, (SR), highlighting the topics important to this project. Although general relativity, (GR), is needed to fully understand the inner workings of the simulation, I will only expand into GR as is needed to the important concepts involved in this project. As such, the two fundamental concepts behind the simulations being performed is the notion of spacetime, and that of geodesics. These two define the physical framework of the space we are working in, and the nature of movement within that space, respectively.

2.1.1 Spacetime

Spacetime is the fusion of three spatial dimensions and one temporal dimension into a four-dimensional manifold. As such, coordinates in this space can be thought of as (t, x, y, z). However, to understand the nature of spacetime and special relativity, we must do away with the Newtonian-esque notions of time as a parameter, and simply think of it as another othogonal direction in which to travel through this new spacetime. With this set of coordinates we also define the formula for the **spacetime interval**:

$$(\Delta s)^2 = -(c\Delta t)^2 + (\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2$$

This interval is analogous to 'length' in Euclidean space we are used to working in. However it is also the basis for what we call **proper time**, 'τ'. This is defined as:

$$(\Delta \tau)^2 = -(\Delta s)^2$$

This quantity represents the *observed time of a subject traveling along the respective path associated with the given spacetime interval.*

This is all well and good, but what does it accomplish for us? Well at this point, we can handle movement through *flat spacetime*, but we also now have the framework required to understand how gravity affects particles by *curving* spacetime.

2.1.2 Geodesics

Geodesics are the shortest paths along curved surfaces. For a flat plane, this is a straight line. But for a surface such as a sphere, the shortest path becomes what's known as a 'great circle' (This is because it can be represented as the intersection of a sphere and a plane which is itself a circle). Its line element (the line element is the definition of a "straight line" on a curved surface) is defined as:

$$ds^2 = dr^2 + r^2 d\theta^2 + r^2 \sin^2(\theta) d\phi^2 \tag{1}$$

This is important to have because it tells us about the movement of particles in that spacetime. A secondary and equivalent definition of a geodesic is as follows: A geodesic is the path followed by an un-accelerated test particle moving through a spacetime. A test particle is simply a particle which does not influence the curvature of spacetime around it. This is a good approximation for relatively small objects, and an even better one for light which has no mass.

This definition of a metric being the straight line that an un-accelerated test particle follows makes it both a great tool for studying light rays around black holes, and a good generalized way of defining a curved spacetime.

A quick note on notation: The angles θ and ϕ are defined as ϕ being the *azimuthal angle* and θ being measured *from the z-axis.*

2.2 Black Holes

Given this discussion on geodesics, we can now move on to the mathematical description of different black holes (BH's). As an overview, a black hole is a region of spacetime which has been warped by gravity to such an extent that even light cannot escape it past a certain point. This limit is called the event horizon, and it has different mathematical definitions based on the type of black hole. The event horizon is what you see in depictions of black holes as the black circle surrounded by the glowing accretion disc. The shape and size of a black hole and its accretion disc is dependent on its mass, rotation, and electric charge. Two common black holes are the **Schwarzschild BH** and the **Kerr BH**. Each one and its associated metric are defined as follows:

2.2.1 Schwarzschild BH

The Schwarzschild BH is the simplest and most straightforward type, perfect for introducing the mathematics behind black holes. It has a definite size, with its event horizon being defined by the Schwarzschild radius, which is entirely dependent on the mass of the black hole. What makes this such a simple example is that it has zero electric charge, and zero angular momentum. This is why it is also known as a static black hole. Its solution to Einstein's field equations is spherically symmetric and its metric, 'g' takes the form:

$$g = (\Delta s)^2 = -c^2 d\tau^2 = -\left(1 - \frac{r_s}{r}\right) c^2 dt^2 + \left(1 - \frac{r_s}{r}\right)^{-1} dr^2 + r^2 g_\Omega \tag{2}$$

Where g_Ω is the metric on the two sphere: $g_\Omega = d\theta^2 + \sin^2(\theta)d\phi^2$ and r_s is the Schwartzschild radius: $r_s = \frac{2GM}{c^2}$

These equations are also often listed in what are called "natural units", where $c = G = 1$. This means the metric can be restated more simply, however, this may lead to confusion if I were to omit variables simply because they are multiplying by 1. Therefore in this report I will leave them in. However, when listed in the code, I omit these variables for simplicity's sake. Again for the sake of understanding, this metric can also be expressed in matrix form as:

$$g = \begin{bmatrix} -(1-\frac{r_s}{r})c^2 & 0 & 0 & 0 \\ 0 & (1-\frac{r_s}{r})^{-1} & 0 & 0 \\ 0 & 0 & r^2 & 0 \\ 0 & 0 & 0 & r^2\sin^2(\theta) \end{bmatrix}$$

2.2.2 Kerr BH

The Kerr BH is fairly simple, very much alike the Schwarzschild BH but instead it has a nonzero angular momentum. This makes many of the solutions around a Kerr BH extremely hard to find, but some solutions (namely those oriented within the plane of rotation) have been found exactly. The metric which describes the spacetime around a Kerr BH is as follows:

$$g = -\left(1 - \frac{r_s r}{\Sigma}\right)c^2dt^2 + \frac{\Sigma}{\Delta}dr^2 + \Sigma d\theta^2 + \left(r^2 + a^2 + \frac{r_s r a^2}{\Sigma}\sin^2(\theta)\right)\sin^2(\theta)d\phi^2 - \frac{2r_s r a\sin^2(\theta)}{\Sigma}cdtd\phi \tag{3}$$

This metric, however, is written in **Boyer-Lidquist coordinates** in order to be concise. This uses the oblate-spherical coordinates defined in cartesian coordinates as:

$$x = \sqrt{r^2 + a^2}\sin(\theta)\cos(\phi)$$

$$y = \sqrt{r^2 + a^2}\sin(\theta)\sin(\phi)$$

$$z = r\cos(\theta)$$

With the Σ, Δ, and a being defined as:

$$\Sigma = r^2 + a^2\cos^2(\theta)$$

$$\Delta = r^2 - r_s r + a^2$$

$$a = \frac{J}{Mc}$$

What makes these black holes so interesting is that they exhibit a phenomenon known as "frame dragging" by which objects within its gravitational influence will begin to rotate with the black hole. At close enough distances, all objects including light *must* rotate along with the BH simply because the spacetime in its vicinity has been warped and rotated along with the BH's rotation.

This metric in matrix form is:

$$g = \begin{bmatrix} -(1 - \frac{r_s r}{\Sigma})c^2 & 0 & 0 & -\frac{2r_s ra\sin^2(\theta)}{\Sigma}c \\ 0 & \frac{\Sigma}{\Delta} & 0 & 0 \\ 0 & 0 & \Sigma & 0 \\ -\frac{2r_s ra\sin^2(\theta)}{\Sigma}c & 0 & 0 & (r^2 + a^2 + \frac{r_s ra^2}{\Sigma}\sin^2(\theta))\sin^2(\theta) \end{bmatrix}$$

2.3 Ray Casting

Ray casting is a process which mimics the paths of light rays in such a way as to determine what an observer at a certain location would see. However, in order to simplify this task, only light rays which an observer would see are simulated. This is done by starting each ray at the observer's location and working backwards to determine where they end up. If one hits a red object, the pixel corresponding to that ray shows red. Each pixel in a rendered image corresponds to a simulated light ray (unless an interpolating method is used to cut down on computation time). How these rays are created starts with the camera object.

2.3.1 Camera Object

The camera object takes the place of the observer, and is what defines where light rays are originating from and which direction they are emanating in. The camera object defines the field of view (FOV) and aspect ratio of the desired render. These two parameters in conjunction with the resolution of the image are what determine the number and direction of light rays being simulated. The mathematics are as follows.

Given the desired resolution, 'Res', of an image (eg: $Res = W \times H = 1920 \times 1080[px]$), the aspect ratio, '$\alpha$' is determined by the ratio of the height and width:

$$\alpha = \frac{W}{H} = \frac{1920[px]}{1080[px]} = \frac{16}{9} = 16:9$$

This ratio is then used in conjunction with the FOV, (Φ), here defined as the horizontal angular range of the camera in a flat spacetime, to determine the vertical angular range, Θ of the camera:

$$\Theta = \frac{\Phi}{\alpha}$$

From this equation we can see that if an image is square, ie has an aspect ratio of 1:1 where $\alpha = 1$, then the equation reduces to $\Theta = \Phi$.

The horizontal and vertical ranges of the camera are then divided by the width and height, respectively, to find the separation angles, '$d\Phi$' and '$d\Theta$', between any two light rays in each dimension:

$$d\Phi = \frac{\Phi}{W}$$
$$d\Theta = \frac{\Theta}{H}$$

Lastly, the direction of each ray relative to the direction of the camera is defined with the coordinate pair, $(\theta_i^{rel}, \phi_i^{rel})$ which is generated by multiplying the ray's pixel coordinates, (x, y) by the separation angles, $(d\Theta, d\Phi)$, and adding the result to the direction of the camera.

For example, if the camera is at the origin in a spherical coordinate system facing $(\frac{\pi}{3}, \frac{\pi}{2})$, the aspect ratio is 16:9 with a resolution of 16x9, and the FOV is $\frac{\pi}{2}$, the pixel with coordinates (10, 5) would have the following initial direction:

$$\alpha = \frac{16}{9}$$

$$\Phi = \frac{\pi}{2}$$

$$\Theta = \frac{\Phi}{\alpha} \approx 1.025[rad] \approx \frac{\pi}{3}[rad]$$

$$d\Phi = \frac{\Phi}{W} = \frac{\frac{\pi}{2}}{16} = \frac{\pi}{32}$$

$$d\Theta = \frac{\Theta}{H} = \frac{\frac{\pi}{3}}{9} = \frac{\pi}{27}$$

Therefore the ray at coordinate (10, 5) would have the direction:

$$(\theta, \phi) = (\frac{\pi}{3} + (5 \times \frac{\pi}{27}), \frac{\pi}{2} + (10 \times \frac{\pi}{32})) = (\frac{14\pi}{27}, \frac{13\pi}{16})$$

Clarification: The spherical coordinates are listed as (θ, ϕ) which would translate to (y, x), but since the Cartesian coordinates are typically listed as (x, y), I keep them listed as such in this report. This just means the order the coordinates are listed in will flip depending on which type of coordinates are being used.

2.3.2 Collisions and Color Determination

After a ray's starting position and direction is calculated, it is then traced across the system until it intersects with an object. There are several operations which can now occur:

The color of the ray can inherit the color value of the object it intersects with. This is the simplest operation that can occur upon collision. This may be combined with an illumination value depending on how 'lit up' that part of the object is in the scene which would scale the color value of the object accordingly.

Another operation that can occur is a reflection. Given a certain value for the reflectiveness of the object's surface, the normal vector of the surface may be used to determine the direction that the light ray reflects to. This process may be repeated several times over to achieve a final color value derived from several factors.

Other operations may be performed based on refraction, diffusion, motion blur, ambient occlusion, and many other effects which are used to create photo-realistic images.

2.4 Simulations

Simulations such as these are done in an important way which combines all of the topics described in the background thus far; this method being *stepping*. This method typically takes in an initial state and a method for extending the simulation, in order to produce another state at a given time frame or step size away from the original.

In the case of ray casting in general, this process may be simplified greatly by calculating the equation of a line through space and mathematically determining its intersection points with objects, completely eliminating any stepping process, and then iterate again for a reflection as many times as desired. However, for more complex systems and simulations such as what is being done in this MQP, a full utilization of stepping is required.

An initial input is given for a light ray's position and momentum vector, and then a metric is defined. This metric tells the simulation what it means for a light ray to move a small distance, Δs, through spacetime (recall, the metric defines the motion of an un-accelerated particle in curved spacetime). The simulation then takes the initial conditions and metric, and given a determined step size, will output the next location of the light ray. This is continued N amount of times until the researcher is satisfied with the length of the simulation.

3 Methods

The breakdown of my work and method is as follows:

1. Create the camera object
2. Compute the initial position and trajectories of the light rays
3. Input each ray into the FANTASY simulation
4. Collect the outputs and reformat them
5. Analyze the output to determine a collision or the exit angle
6. Map the exit angles onto an image's colors
7. Assemble the colors to create a fully rendered image

Each step of the process is detailed in the preceding sections, splitting the tasks up into **Input**: (1,2,3), **FANTASY Output**: (4, 5), **Color Determination**: (5, 6), and **Image Generation**: (7).

3.1 Platforms / Software Used

For the simulations performed, I used python 3.8 in the Spyder 4.2.5 environment, a symplectic integrator called FANTASY created by Christian and Chan [2], and the following python modules:

Module	Object
numpy	
matplotlib.pyplot	
PIL	Image
datetime	datetime
time	
os	

3.2 Input

The task of creating the camera object is done by specifying the parameters "width", "height", and "Field of View":

```
FOV = np.pi/2

W = 1920

H = 1080

camera_angle = [np.pi/2, 0]
```

Which are used as described in the "Camera Object" section. These parameters are then input into the **ray_angles** function to create the camera object. This function outputs an array of shape $(H, W, 2)$ which describes the theta and phi angles of each ray corresponding to each pixel in the final image.

The program then iterates through each ray's angles, and uses the angles to specify the initial conditions of the ray for the simulation. At this stage several more parameters are also predefined to be inputted into the simulation:

```
N = 100

delta = 0.1

omega = 1

q0 = [0, 9, np.pi/2, -np.pi/2]

p0 = [-1, -1, angle[0], angle[1]]

a = 0.01

M = 1

order = 2
```

These describe, `N`, the number of iterations to perform, `delta`, the step size, `omega`, the interaction parameter between the two phase spaces, `q0`, the initial position, `p0`, the initial momentum, `a`, the length scale (as defined in the Kerr BH section), `M`, the BH mass, and `order`, the integration order of the simulation. As shown, the initial momentum is the only parameter which is changing between simulations with `angle[0]` and `angle[1]` being θ and ϕ for each ray.

3.3 FANTASY Output

The FANTASY program outputs an array of shape $(N, 4, 4)$ where the first dimension, 'N', is the number of time steps, the second dimension is the position and momentum coordinates in two different phase spaces, and the last dimension holds the time-space coordinates themselves. In other words, they are indexed by time, phase space + position/momentum, then coordinate component.

The python script I wrote then cuts and reshapes the output to only take what is desired. For example, one may want to look at only the position coordinates in the first phase space, so the modifications will give an array of shape (4, N). This is essentially a list containing four lists of length N which correspond to the four dimensions of the time-space coordinates. For future calculations, another array of shape (4, N) which is derived from the momentum in the first phase space is used for determining the exit trajectory.

3.4 Color Determination

The method for determining the colors of each ray in this simulation is based heavily on a *rendering environment*. This environment is an image which is used as a ray goes out towards infinity and does not collide with anything. Each pixel of the image is assigned a theta-phi angle range such that if a ray exits the system with a trajectory within that range, it will take that pixel's color. See figures and 2 for an illustrations of this process.

Figure 1 depicts the 'Local System' as the area within which the actual path the ray takes is simulated in. This is where the positional coordinates are relevant. The 'Distant System' is depicted as an interpolation of what the ray's path would look like from a distance infinitely far away. In this system, the ray's path is a straight line with definite angular coordinates. These angles are then used to determine its pixel's color.

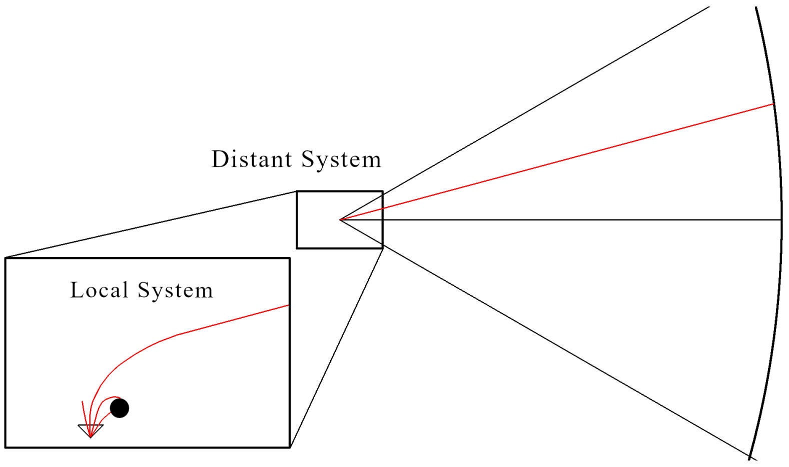

Figure 1: *Depiction of the 'local' and 'distant' systems. The path of the light ray is depicted in red with several smaller rays in the local system to show where the simulation is occurring.*

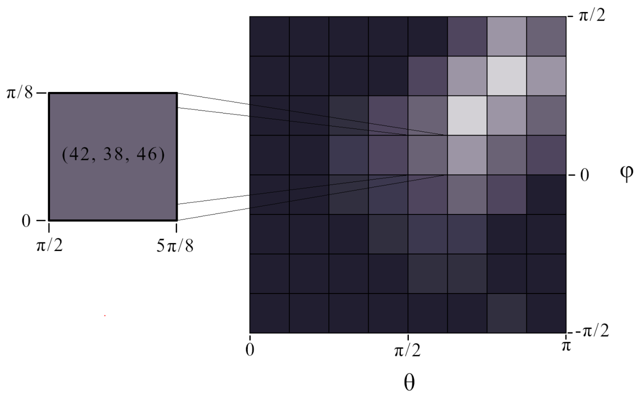

Figure 2: Illustration of the grid layout used in calculating the color value of a light ray. If a light ray's exit trajectory has angles within a certain range, the image pixel which is mapped to that range will imbue its color to that ray.

The color of a pixel that is displayed on screen is determined by the direction that the ray corresponding to that pixel is pointing when exiting the gravitational system. In the code, this is done using the ray's last momentum vector, and the function **image_map**.

There is first a check made for each ray's path through spacetime to determine If the path of the light ray ever meets or goes within the Schwartzschild radius or the given definition of the event horizon. If it does, then it makes the final color of the pixel black.

3.5 Image Generation

The **image_map** function takes the exit angles θ and ϕ taken from the ray's last momentum vector and returns a color. ϕ is first centered to the background by adding π, and then these angles undergo a modulo operation to ensure the angles' integer values are within $[0 \leq \phi \leq 2\pi]$, $[0 \leq \phi \leq \pi]$. Lastly, a pre-defined image is loaded in, and based on the size of the image a pixel is given an angle range of $(\Delta\theta, \Delta\phi)$ (In the code these are defined as `div_t`, `div_p`). The exit angles are then divided by $[\Delta\theta, \Delta\phi]$ and rounded to the nearest integer to get $[x, y]$ coordinates of the pixel that the light ray maps on to. This pixels' color value is then used to determine the color of that ray.

The **create_image** function then takes the colors in the form of a 1D list, and together with the predefined width and height values, creates an image of that size.

4 Results

4.1 Diagnostics

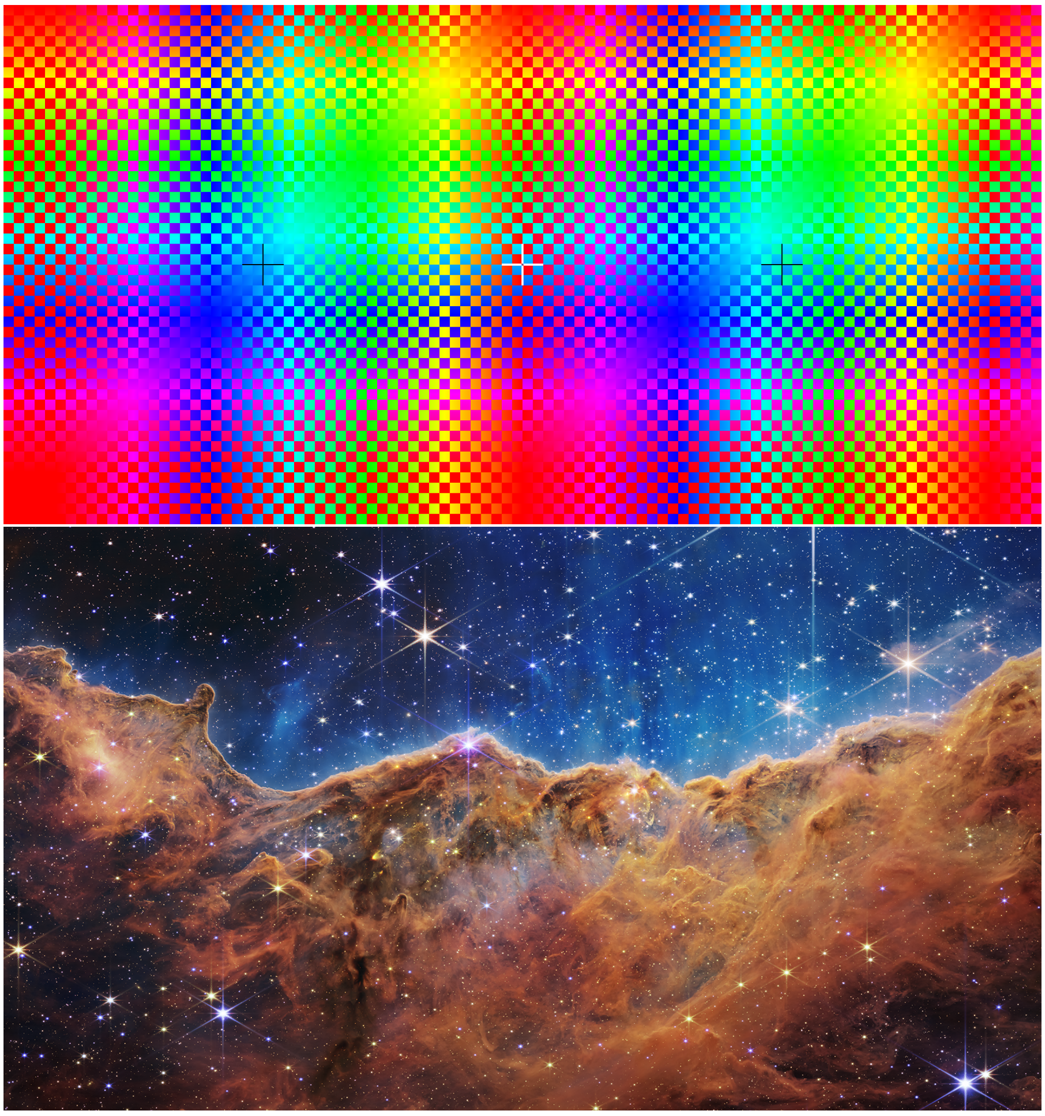

Figure 3: Diagnostic image being used to test the program (Top).
Example background for analyzing structures (Bottom).

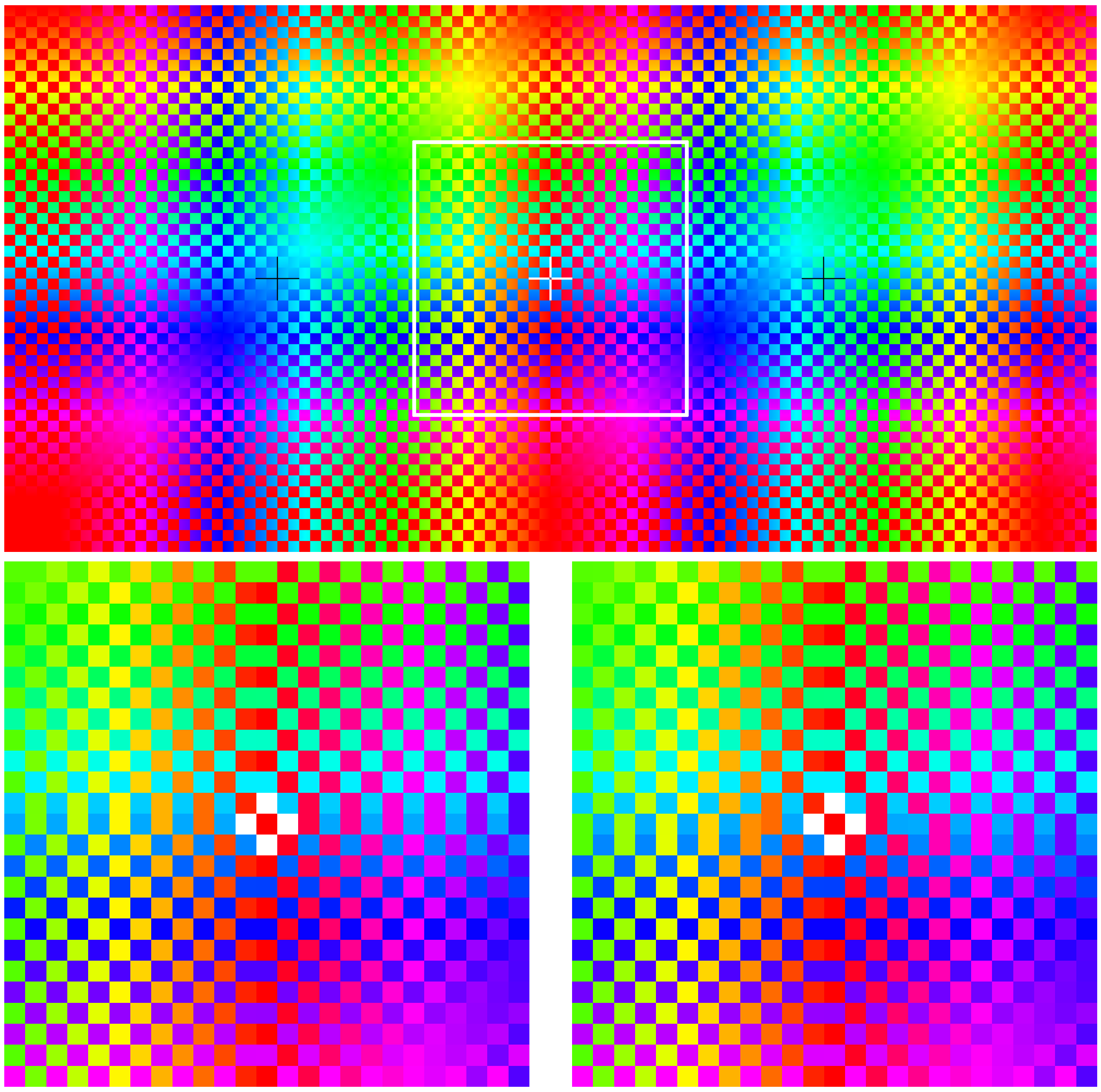
Figure 4: Diagnostic image with $[\pi/2 \times \pi/2]$ box overlay (Top), $[\pi/2 \times \pi/2]$ Control image (Left), and $[\pi/2 \times \pi/2]$ Rendered image for N=10 (Right).

4.2 First Renderings

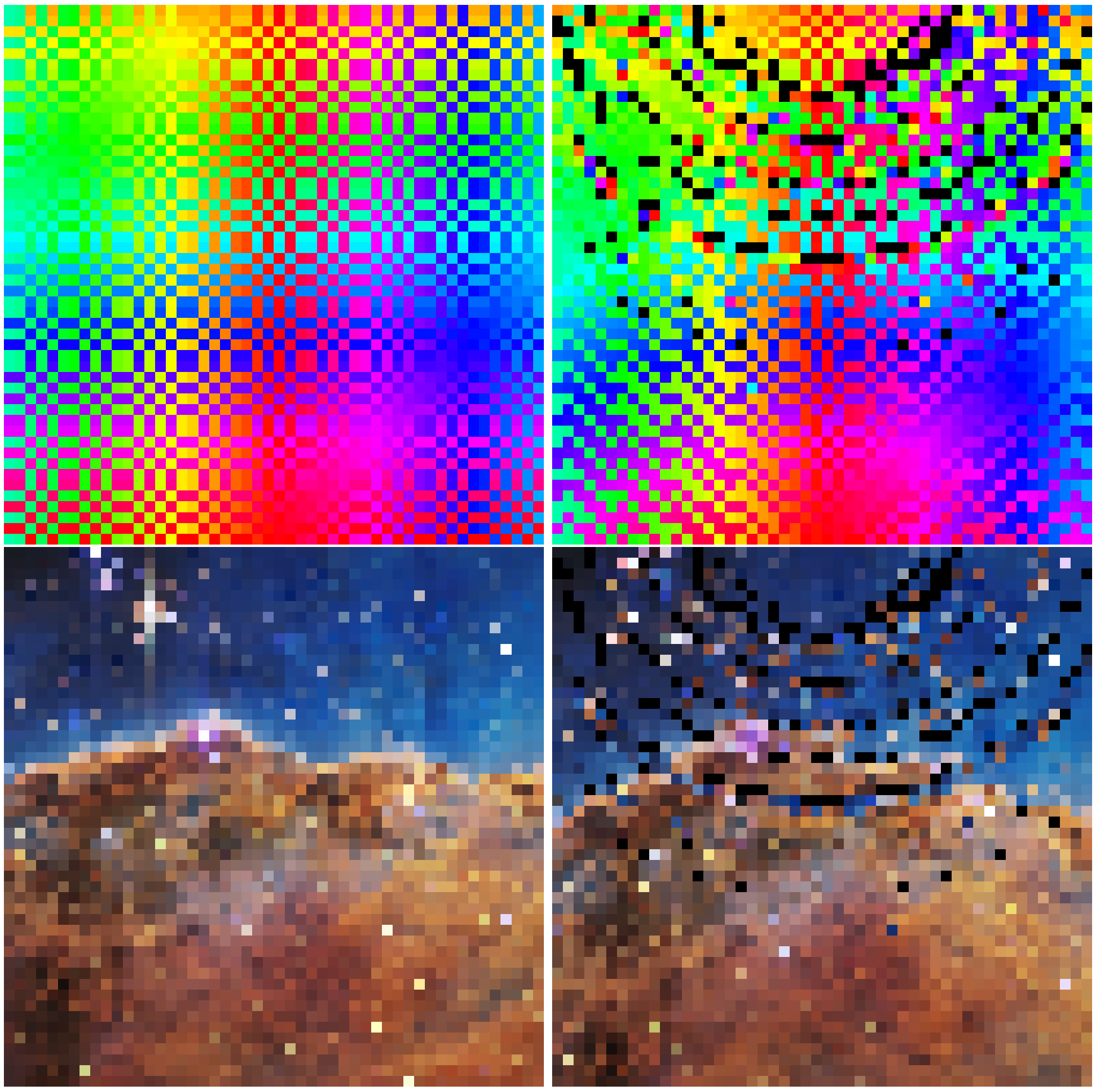

Figure 5: $[\pi/1.25 \times \pi/1.25]$ Control images (Left Column), and $[\pi/1.25 \times \pi/1.25]$ rendered images (Right Column). Parameters for both rows are: FOV = $\pi/1.25$, W = 50, H = 50, camera angle = $[\pi/2, 0]$, N = 52, delta = 0.1, omega = 10, q0 = $[0, 6, \pi/2, -\pi/2]$, p0 = [-1, -1, θ_i, ϕ_i], M = 1, a = 0.001, order = 2

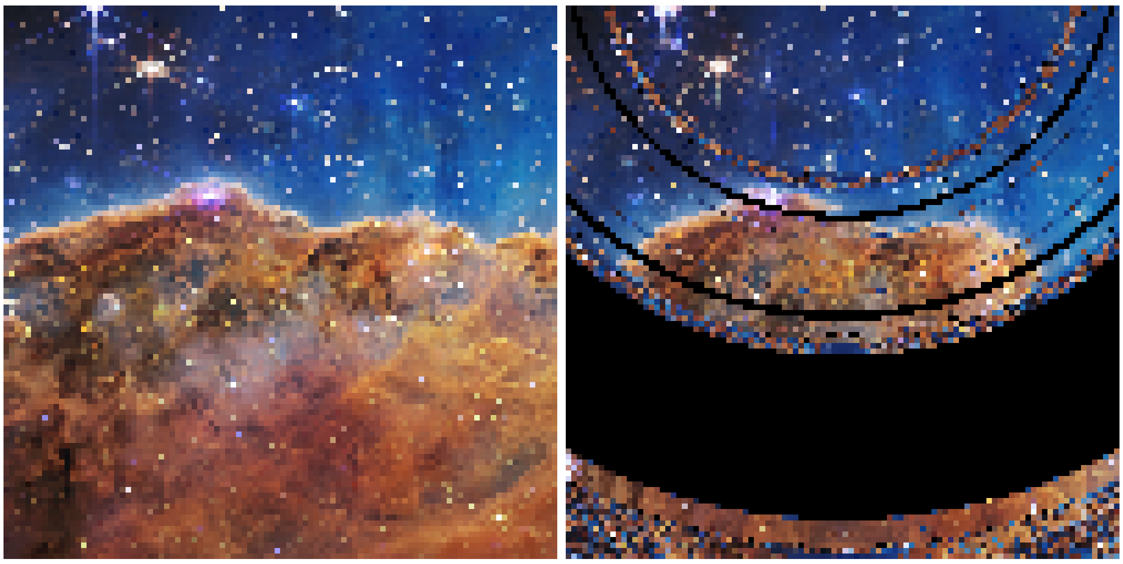

Figure 6: $[\pi/1.25 \times \pi/1.25]$ Control image (Left), and $[\pi/1.25 \times \pi/1.25]$ rendered image (Right). Parameters are: FOV = $\pi/1.25$, W = 100, H = 100, camera angle = $[\pi/2, 0]$, N = 50, delta = 0.5, omega = 10, q0 = $[0, 10, \pi/2, -\pi/2]$, p0 = $[-1, -1, \theta_i, \phi_i]$, M = 1, a = 0.001, order = 2

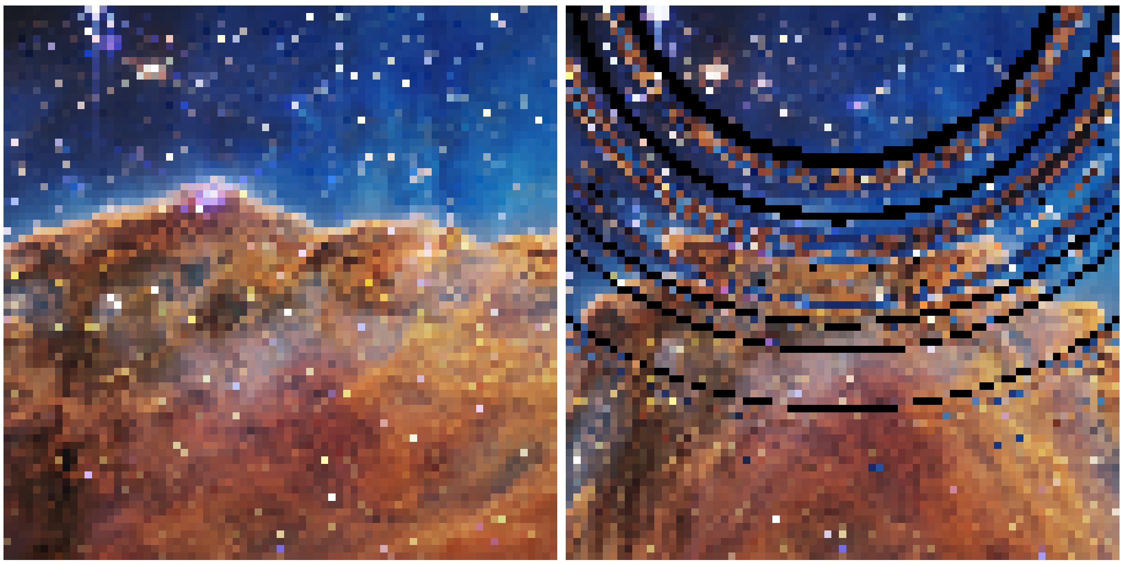

Figure 7: $[\pi/1.25 \times \pi/1.25]$ Control image (Left), and $[\pi/1.25 \times \pi/1.25]$ rendered image (Right). Parameters are: FOV = $\pi/1.25$, W = 75, H = 75, camera angle = $[\pi/2, 0]$, N = 50, delta = 0.4, omega = 10, q0 = $[0, 20, \pi/2, -\pi/2]$, p0 = $[-1, -1, \theta_i, \phi_i]$, M = 1, a = 0.001, order = 2

4.3 Einstein Rings

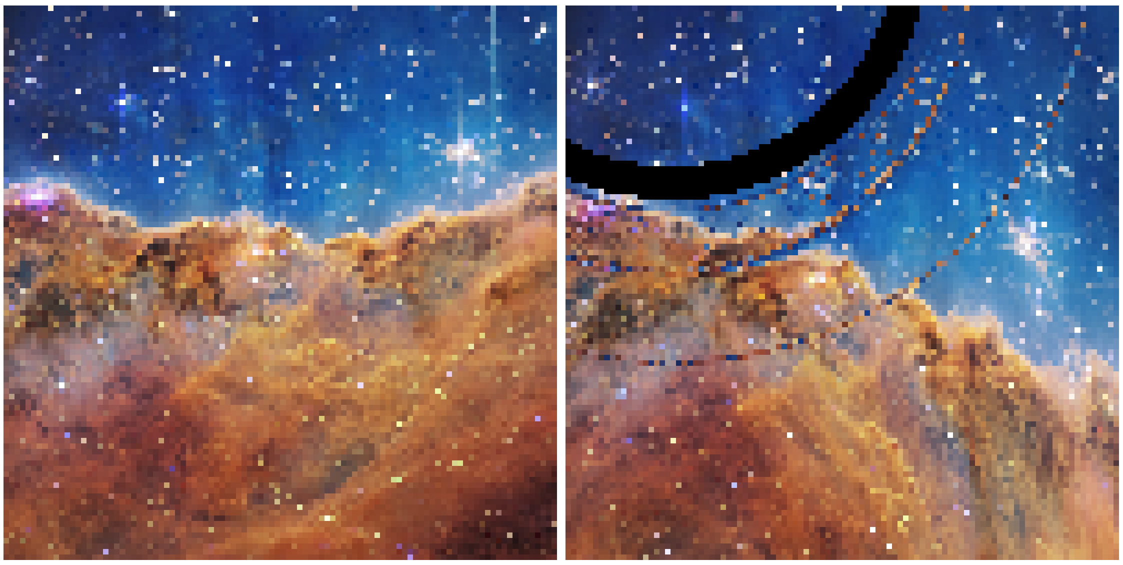

Figure 8: $[\pi/1.25 \times \pi/1.25]$ Control image (Left), and $[\pi/1.25 \times \pi/1.25]$ rendered image (Right). Parameters are: FOV = $\pi/1.25$, W = 100, H = 100, camera angle = $[\pi/2, \pi/4]$, N = 50, delta = 1, omega = 10, q0 = $[0, 50, \pi/2, -\pi/2]$, p0 = $[-1, -1, \theta_i, \phi_i]$, M = 1, a = 0.001, order = 2

5 Analysis/Discussion

The image in figure 3 was designed with a color gradient in vertical and horizontal directions, as well as with a grid showing straight lines along each axis in order to show that the rendered image had no warping and was oriented correctly. the colors themselves also overlap in such a way that it isn't difficult to tell what part of the image is being shown in the rendering.

Figure 4 shows the same diagnostic image with a $[\pi/2 \times \pi/2]$ white box where the rendered image is supposed to cover. There are two images below it: the left image is the control image, and the right image is the simulated image. The control is created by taking the initial angles from the `ray_angles` function and directly mapping them to the diagnostic image using the `image_map` function. The resulting image does not go through any simulation steps, and shows what an observer would see in a perfectly flat spacetime. As shown in the figure, this image aligns perfectly with what is expected. The rendered image goes through the simulation for 10 steps, with the camera placed very far from any curvature. Therefore we expect the warping to be very minimal, and the rendered image to look near identical to the control image. This is indeed what we see, including the white reticle in the center.

Figure 5 shows the first simulation under appreciable curvature. Here, the same simulation was performed twice with both the diagnostic and example background images for comparison. In the diagnostic set, we see that the bounds of the image don't change much if at all, but there is a clear warp in the grid structure. This is also the first simulation where light rays encountered the event horizon, indicated by the black pixels. The example image simulations concur with these observations, showing a warping of the edge of the nebula. Something interesting to note is that we can see in the example image the first instance of what may possibly be an Einstein ring. The outermost clear ring of black pixels cuts in to the nebula, but brings with it a clear ring of blue, indicating that the rays near this edge are being severely warped around the black hole. However, with such low resolution, and what appears to be a lot of noise, this is an optimistic conclusion.

In figures 6 and 7, we see much of the same as before, but in higher detail. This time we are seeing clear rings of brown and blue as the light rays curve around the black hole near the event horizon. What is becoming apparent, however, is a pattern emerging that is indicative of something wrong with the calculations. A black hole should appear to be a solid disc, with one or more Einstein rings forming outside of that disc. A good example of this is a simulated image created by Ute Kraus at the Institute of Physics, Universität Hildesheim, which is shown in figure 9.

In figure 8 however, we see something closer to what we are expecting to see. There is an obvious black circle (albeit not solid) with three clear rings around it. However, these rings are uncharacteristically sharp, and it brings into question whether they are truly arising from the spacetime geometry, or some error in the code related to the geometry.

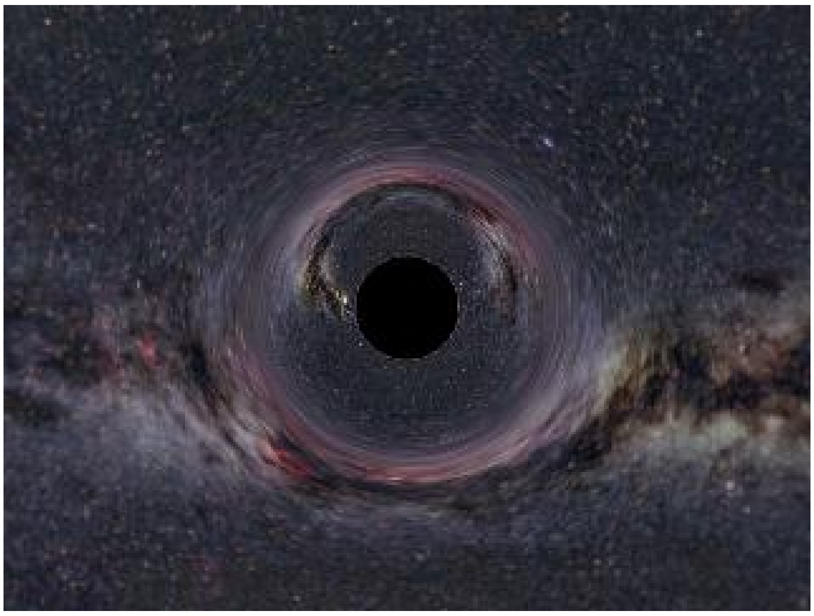

Figure 9: Computer simulated image of a black hole in front of the Milky Way galaxy, showing an Einstein ring. [4]

6 Conclusion

In total, evidence of gravitational lensing and Einstein rings were found, and a program was successfully written and implemented in conjunction with the FANTASY geodesic integrator in order to reproduce these phenomena. The user can produce first person POV renderings of black holes or of any spacetime metric with variable background images. Further research in this area would include creating much higher resolution (500×500 or larger) images to properly identify and confirm the existence of Einstein rings in the simulations. As of now, the noise produced in these low resolution results prevents a full analysis of the artifacts being shown.

7 References

References

[1] Kazunori Akiyama et al. “First M87 Event horizon telescope results. V. Physical origin of the Asymmetric Ring”. In: *The Astrophysical Journal* 875.1 (2019). DOI: `10.3847/2041-8213/ab0f43`.

[2] Pierre Christian and Chi-kwan Chan. “FANTASY: User-friendly Symplectic Geodesic Integrator for Arbitrary Metrics with Automatic Differentiation”. In: *The Astrophysical Journal* 909.1 (Mar. 2021), p. 67. DOI: `10.3847/1538-4357/abdc28`. URL: `https://dx.doi.org/10.3847/1538-4357/abdc28`.

[3] The Event Horizon Telescope Collaboration. “First Sagittarius A* Event Horizon Telescope Results. I. The Shadow of the Supermassive Black Hole in the Center of the Milky Way”. In: *The Astrophysical Journal Letters* 930.L12 (2022). URL: `https://doi.org/10.3847/2041-8213/ac6674`.

[4] Ute Kraus. *Black Hole in front of the Milky Way.* URL: `https://www.spacetimetravel.org/galerie`.