

Geometrical Optics of Gravitational Lensing: A Case Study in Black Holes

Kai Shinbrough

May 12, 2016

1 Introduction

1.1 Motivation: “Why Study Gravitational Lensing?”

This paper will explore the geometrical optics of gravitational lensing generally and investigate the specifics of gravitational lensing when applied to black holes.

Gravitational lensing is a useful tool in several respects. If gravitational lensing is observed and the relative distance between the lensing object and the source is known as well as the distance between the lensing object and Earth, calculations based on the observed gravitational lensing can be performed quickly to determine the mass distribution of the lensing object(s) [1]. Gravitational lensing is also useful as a means of observing magnified images of novel distant objects, as it is often the case that gravitationally lensed images appear several times the size of the original lensed object [6]. Further, gravitational lensing can be used to calculate several cosmological parameters, most notably the Hubble Constant and the Bias Parameter. Kayser & Refsdal have shown that the time delay between multiple lensed images of a quasar can be used to calculate the Hubble Constant [3], whereas Schneider, et al. have noted that weak gravitational lensing is one of the few techniques currently available for the investigation of the relationship between the statistical distribution of galaxies and dark matter, a key component in calculation of the Bias Parameter [6].

1.2 This Paper: “Why Black Holes?”

Black holes have garnered considerable media attention in the past year [4], and rightfully so. It is in accordance with this trend that this paper also centers around black hole study, but there are other, perhaps more pressing reasons for doing so as well. Given their axisymmetry and point-mass behavior, black holes serve as the simplest case to consider for gravitational lensing, effectively reducing understanding of the effect to careful geometric consideration. As such, centering this paper around black hole study should also serve to make gravitational lensing accessible to those without an advanced mathematical background—all that should be needed to understand this paper is a basic knowledge of trigonometry.

As alluded to above, this paper has two main sections: a general exploration of the geometrical optics of gravitational lensing—loosely, how we determine mass distributions given observed gravitational lensing, and how we accordingly model gravitational lensing arising from simple mass distributions (e.g. Black Holes)—and a more in depth investigation of the gravitational lensing of black holes, vis-à-vis a step-by-step overview of a program designed to model the view from the Event Horizon Telescope (EHT). The EHT is a telescope aimed at Sagittarius A*, the $\sim 4,000,000$ solar mass black hole at the center of the Milky Way, $\sim 25,000$ light years away from Earth [2].

2 Geometrical Optics of Gravitational Lensing

2.1 Determining mass distributions given observed lensing

Given observed gravitational lensing, determining the total mass of the lensing object is fairly straightforward, once one becomes familiar with the Einstein Deflection Law:

$$\hat{\alpha} = \frac{4GM_{\text{Total}}}{c^2\xi}. \tag{1}$$

Here I've stuck with the somewhat arcane historical formalism, where $\hat{\alpha}$ is the deflection angle, G is the familiar Newtonian gravitational constant, M_{Total} the total mass of the lensing object, c the speed of light, and ξ the impact parameter most will be familiar with from Rutherford Scattering (simply put, the distance of closest approach from the center of mass of the lensing object).

The Einstein Deflection Law is perhaps best understood through examination of Figure 1.

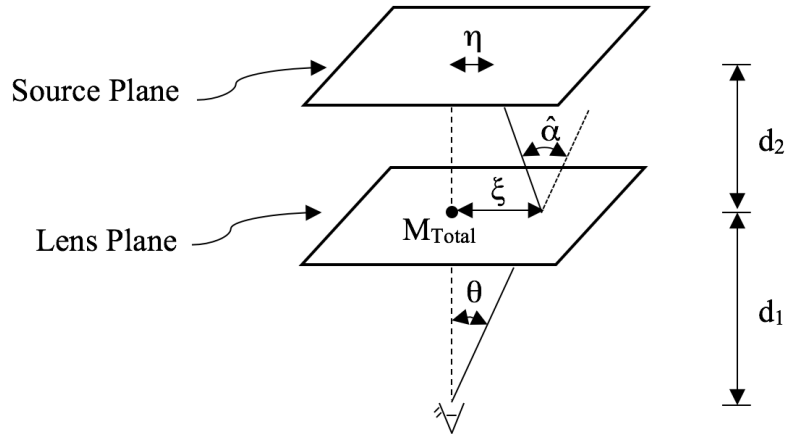


Figure 1: A visualization of the Einstein Deflection Law, where η is the distance from the center of the source plane corresponding to what appears at ξ in the lens plane, θ the angle between the observer's line of sight when looking at the center of mass of the lensing object versus at ξ , d_1 the distance between the observer and the lens plane, d_2 the distance between the lens plane and the source plane, and where $\hat{\alpha}$, M_{Total} , and ξ are defined as above. Image concept courtesy of Schneider, et al. [6]

Clearly this is an approximate relationship—in reality the light from the source plane would arc around the lensing object on its way to the observer instead of taking the two straight-line paths shown in Figure 1, but as long as $d_1, d_2 \gg \xi$ this approximation proves exceedingly good [6]. It is also worth noting that finding the center of mass of the lensing object is not always a straightforward task, and so neither is finding ξ , but for the case of a black hole this difficulty is avoided.

Schneider et al. have shown that in the case of multiple lensed images of the same source object, the images scale with lens mass as $\sqrt{M_{\text{Total}}}$ [6]. Thus, given multiple images occurring from gravitational lensing, we can measure the distance between them, and quickly arrive at a value for M_{Total} of the lensing object(s).

2.2 Modeling Gravitational Lensing arising from Simple Mass Distributions

Given what we now know about determining mass distributions of lensing objects given observed gravitational lensing, our task becomes to work backwards and model gravitational lensing given simple lensing mass distributions. What is referred to as simple here is any mass distribution displaying axisymmetry, be that a very well defined circular galaxy in line with the observer and source object, a massive lone neutron star, other axisymmetric mass distributions, or, paradigmatically, a black hole.

Since what is seen by the observer is the lens plane, the image seen at $\vec{\xi}$ arises from photons leaving from $\vec{\eta}$ in the source plane, where $\vec{\xi}$ and $\vec{\eta}$ are vectors with lengths ξ and η respectively, where $\vec{\xi}$ points from the center of the lens plane to the image observed and $\vec{\eta}$ points parallel to $\vec{\xi}$ but from the center of the source plane. From consideration of Figure 1 it can be seen that

$$d_2 \tan(\hat{\alpha} - \theta) = \xi - \eta, \quad (2)$$

where

$$\theta = \tan^{-1}\left(\frac{\xi}{d_1}\right). \quad (3)$$

Given our approximation that $d_1 \gg \xi$, we employ the small angle approximation $\tan(\theta) \approx \theta$, altering Equation (3) to:

$$\theta \approx \frac{\xi}{d_1}.$$

This in turn modifies Equation (2) such that

$$d_2 \tan\left(\hat{\alpha} - \frac{\xi}{d_1}\right) \approx \xi - \eta,$$

and given $d_1 \gg \xi$ it follows that $\hat{\alpha} \gg \frac{\xi}{d_1}$, and accordingly Equation (2) becomes:

$$d_2 \tan(\hat{\alpha}) \approx \xi - \eta. \quad (4)$$

With this simplified relationship, finding the image at $\vec{\xi}$ becomes a straightforward, two-part process: first find $\hat{\alpha}$ with Equation (1) for your given ξ , and second use Equation (4) to compute η and $\vec{\eta}$ – the image at $\vec{\xi}$ in the lens plane is simply the image at $\vec{\eta}$ in the source plane!

3 Black Hole Gravitational Lensing Program

We now turn to the second section of this paper: a more thorough investigation of gravitational lensing arising from black holes vis-à-vis exploration of a program designed to simulate it. The program takes exactly the approach outlined at the end of the previous section, with only a few caveats. Though hopefully comprehensive and thorough, the following is not intended to be a tutorial on how to write such a program (although it will surely serve as a helpful guide to anyone interested). Rather, it is intended only to serve as a helpful exercise in internalizing the geometrical optics of gravitational lensing.

The figures displaying the code for each of the following subsections are found at the end of this paper, under Lensing Program Code (Section 6).

3.1 Welcome to the Program

As any good program should, this program starts off with a greeting to the user and a brief explanation of what exactly it simulates, as well as some suggestions for what in-program values to use for optimal performance.

As can be seen in Figure 4 (in Section 6, Lensing Program Code, below), the first chunk of code in the program also initializes the program with the packages necessary for it to run, defines several familiar astronomical and physical constants, prompts the user to provide a picture to be ‘gravitationally lensed’, and computes the pixel dimensions of the picture provided by the user.

3.2 User-selected Parameters

The next part of the program asks for the user-selected parameters: the ‘physical’ parameters required for computing the gravitationally lensed picture. Here we ask for the mass of the black hole the user would like to superimpose on her picture, as well as distances d_1 and d_2 from Section 2.1.

Having everything we need from the user, we now move on to the tasks of calculating the ‘real’ Schwarzschild radius of the Black Hole and calculating what that Schwarzschild radius is in pixels, making use of a “real-to-pixel ratio” derived with a calculation of the ‘real’ height of the user’s picture. And because you cannot always rely on having a benevolent, careful user, this part of the program also includes a clause to prevent the input of a black hole mass and d_1 distance such that the user is within the Schwarzschild radius of the black hole.

The Schwarzschild radius of a black hole is calculated with the following equation:

$$R_{\text{Schwarzschild}} = \frac{2GM_{\text{BlackHole}}}{c^2},$$

which can be easily derived from Newtonian gravitation. Since such a derivation is not within the purview of this paper we refer the interested reader to a derivation in Schneider et al. [6].

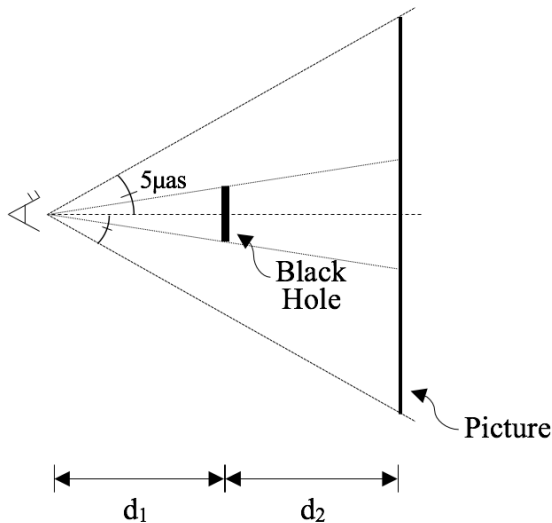


Figure 2: A visualization of the geometry that leads to a calculation of the picture height in meters.

The ‘real’ height of the user’s picture is calculated with the simple geometry of Figure 3, leading to a height h of

$$h = 2(d_1 + d_2) \tan(5\mu\text{as}), \tag{5}$$

where $5\mu\text{as}$ is half the angular resolution of the EHT [2]. We use half the angular resolution because the tangent is taken of the angle bisecting the angular resolution of the EHT (see Figure 2).

This value of the ‘real’ height of the picture proves exceedingly useful as we can use it for the basis of our “real-to-pixel ratio,” which is simply the picture’s calculated ‘real’ height divided by its height in

pixels. From this real-to-pixel ratio we can easily calculate our Schwarzschild radius in pixels, as shown in Figure 5.

3.3 Lensing the Picture

Once we've done the computationally-trivial welcoming of the user to our program, and finished up with the user-selected parameters, we move on to the physics of the program, i.e. lensing the image.

This is done by scanning across each pixel row and column of the original image (where the "original" image is the image as it was when we received it from the user) and reassigning every pixel a color based on the $\vec{\xi} - \vec{\eta}$ calculations discussed earlier.

Since we start our pixel scanning from the top-left corner of the image, we have to undergo the transformations

$$x = i - \frac{(\text{picture width})}{2} \tag{6}$$

$$y = \frac{(\text{picture height})}{2} - j, \tag{7}$$

to get from our scanning indices i and j to the Cartesian x - y coordinates of our pixel, where we've placed our origin at the center of the image.

We then parametrize our $\vec{\xi}$ into ξ_x and ξ_y , and calculate their respective 'real' lengths by dividing our x and y values by the real-to-pixel ratio calculated earlier.

Following the prescription of Section 2.1, we then calculate $\hat{\alpha}$ using Equation (1), and calculate parametrized η_x and η_y with Equation (4). We then convert our newfound η_x and η_y into pixels, again using our real-to-pixel ratio, and transform that value back into our scanning indices using our transformations (6) and (7). Then we reassign the color of our pixel to the color of the pixel at our newly calculated indices—and voilà! Our picture has been lensed.

3.4 Finishing Touches

All that remains to be done to finish the program is to draw the black hole (with the appropriate pixel Schwarzschild radius, calculated earlier) in the center of the image, display the final product, and give the proper accreditation. This is again computationally-trivial, but the program would not be complete without it.

3.5 Results

What follows is a comparison of the lensed images from the program detailed in this paper, and images produced with GravLens3, an iPhone App created by Rycoff [5].

In Figure 3, (a) shows an original image, (b) shows the corresponding 'lensed' image created by the program detailed in this paper, and (c) shows the image as 'lensed' by GravLens3. The black hole mass, and distances d_1 and d_2 necessary for this program were chosen to approximate the result of GravLens3. One notable difference between this program's images and those of GravLens3 is the quadrant-like behavior of our lensed images. This blemish on the lensed image could be ameliorated by using polar cylindrical coordinates, and parameterizing $\vec{\xi}$ into ξ_r and ξ_θ instead of into the Cartesian ξ_x and ξ_y . One clear advantage to the program described in this paper, however, is the variability of the black hole mass; GravLens3 allows

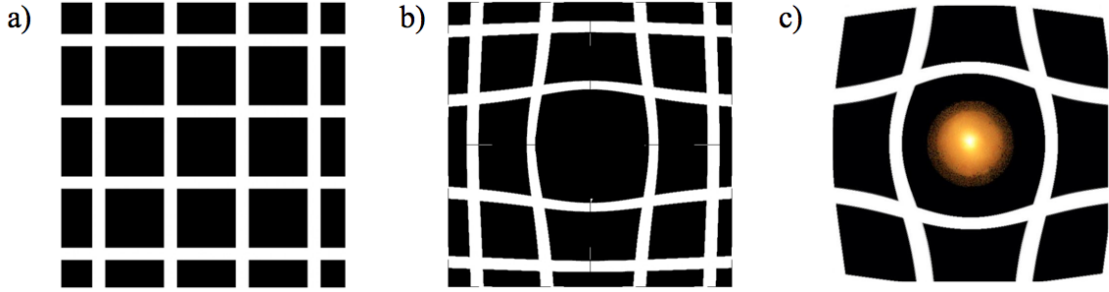


Figure 3: A comparison of (a) an original image, (b) the image lensed by the program detailed in this paper, and (c) the imaged as lensed by the iPhone App GravLens3

for only one lensing object mass (in actuality the lensing object is intended to be galaxy). This program also gives the user some context for gravitational lensing, namely that it could be viewed from the EHT, and additionally allows the user to see what happens to the lensing of their image when they change the black hole size, or distances d_1 and d_2 .

4 Further Study

This program serves as an educational tool for use in introductions to gravitational lensing, but as such could be improved. One obvious avenue for improvement is that outlined above: parameterizing $\vec{\xi}$ and $\vec{\eta}$ into polar cylindrical coordinates instead of Cartesian. An option to include multiple black holes at either fixed or mobile positions would also likely be very helpful for achieving a heightened intuitive understanding of gravitational lensing arising from black holes on the part of the user. And on a more topical note, creating a simulation of gravitational lensing arising from two orbiting black holes would undoubtedly be of popular interest.

5 Acknowledgments

The author would like to give a special thanks to Professor Cynthia Taylor of Oberlin College, without whose website and teaching [7] this program would never have come to fruition.

6 Lensing Program Code

```
import _imaging
import PIL
import picture2
#
#
# Defining constants
G = float(6.67*10-11) # In SI units
c = float(3*108) # In meters/sec
Msolar = float(1.988*1030) # In kg
#
# Getting file name, getting picture height/width, displaying picture:
print("Welcome to Kai Shinbrough's black hole gravitational lensing program!")
print("This program takes any .png image you care to upload, and superimposes a black hole on it, with mass and separation distances of your choosing!")
print("Note that this program simulates what you'd see out of the Event Horizon Telescope (EHT), which has an angular resolution of ~10 micro-arcseconds.")
print("Before you use this program, you might want to do some rough estimates of how massive your black hole should be, given the separation distances you choose.")
print("I recommend looking at your picture as if it were on the other side of the Milky Way galaxy, lensed by Sagittarius A*.")
print("Sagittarius A* is the ~1000000 solar mass black hole at the center of the Milky Way, ~25000 light years away from Earth.")
print
fileName = raw_input("Please enter the file name of the photo you'd like to upload:")
print
original = picture2.Picture(fileName)
pictureWidth = original.getWidth()
pictureHeight = original.getHeight()
print(pictureWidth,pictureHeight)
copy = picture2.Picture(fileName)
print("Ok, here is your lovely picture. Press enter to move on, when you're done admiring it.")
copy.display()
raw_input()
```

Figure 4: Welcome code; initializing the program with whatever packages are necessary, defining constants for later use, greeting the user, brief explanation of program, suggestions for what values to use later in the program, accepting an input picture, and getting information about that picture's height and width.

```

# Getting mass of black hole, computing Schwarzschild radius, getting distances, checking that the mass-distance combination doesn't imply you're inside the
# Schwarzschild radius,
# getting 'real' picture height (in lightyears), converting Schwarzschild radius from lightyears to pixels
massBH = float(raw_input("How massive of a black hole would you like to superimpose on your picture (in solar masses, please)?"))
print
realSchwarzschildRadius = float(((2.0*massBH*Msolr**G)/(c*c))/(9.46*10000000000000.0)) # Last term is m/ly
distance1 = float(raw_input("How far away from you should the black hole be, in lightyears?"))
print
test1 = 0
while test1==0:
    if distance1 < realSchwarzschildRadius:
        print("At that mass-distance combination you'd be within the Schwarzschild Radius of the black hole!")
        remedy = raw_input("Please enter another distance (in lightyears), or, if you'd like to change the mass of your black hole, type 'change mass'.")
        print()
        if remedy=="Change mass" or remedy=="change mass" or remedy=="Change Mass." or remedy=="Change mass." or remedy=="change mass." or remedy=="Change Mass.":
            massBH = float(raw_input("How massive of a black hole would you like to superimpose on your picture (in solar masses, please)?"))
            print
            realSchwarzschildRadius = float(((2.0*massBH*Msolr**G)/(c*c))/(9.46*10000000000000.0))
        else:
            distance1 = float(remedy)
    else:
        test1 = 1
distance2 = float(raw_input("How far behind the black hole should your picture be, in lightyears?"))
print
realPictureHeightSource = float(2.0*(distance1+distance2)*9.46*10000000000000.0*(0.0000000004848)) # In meters. Second term is tan(5microarcseconds),
# 5microarcseconds being approximately the angular resolution of the EHT (Event Horizon Telescope)
realPictureHeightLens = float(2.0*(distance1)*9.46*10000000000000.0*(0.0000000004848)) # In meters.
realToPixelRatioSource = float(realPictureHeightSource/pictureHeight) # meters/pixel
realToPixelRatioLens = float(realPictureHeightLens/pictureHeight)
pixelSchwarzschildRadius = float((realSchwarzschildRadius*(9.46*10000000000000.0))/realToPixelRatioLens)

```

Figure 5: Code responsible for taking in the user-selected parameters: the mass of our black hole and the distances d_1 and d_2 . This code also computes the Schwarzschild radius of the Black hole in meters as well as pixels, and the height of the user's picture in meters.

```

# Lensing the picture
for i in range(1, pictureWidth-1):
    for j in range(1, pictureHeight-1):
        x = int(i-(pictureWidth/2)) # x is horizontal pixel distance from center of picture
        y = int(-j+(pictureHeight/2)) # y is vertical pixel distance from center of picture
        xiX = float(realToPixelRatioLens*x) # In meters, has sign, in lens plane
        xiY = float(realToPixelRatioLens*y) # In meters, has sign, in lens plane
        if xiX==0 or xiY==0:
            copy.setPixelColor(i,j,original.getPixelRed(i,j),original.getPixelGreen(i,j),original.getPixelBlue(i,j))
        else:
            alpha = .000000001*4*G*massBH*Msolr/(c*c*((xiX**2+xiY**2)**(.5)))
            if xiX>0:
                etaX = xiX - .06*(distance2*(9.46*10000000000000.0)*(alpha+((alpha**3)/3)+(2*(alpha**5)/15)))
                # term involving alpha is Taylor expansion of tan(x) to 5th order
            if xiX<0:
                etaX = xiX + .06*(distance2*(9.46*10000000000000.0)*(alpha+((alpha**3)/3)+(2*(alpha**5)/15)))
            if xiY>0:
                etaY = xiY - .06*(distance2*(9.46*10000000000000.0)*(alpha+((alpha**3)/3)+(2*(alpha**5)/15)))
            if xiY<0:
                etaY = xiY + .06*(distance2*(9.46*10000000000000.0)*(alpha+((alpha**3)/3)+(2*(alpha**5)/15)))

            pixelEtaX = etaX/realToPixelRatioLens
            pixelEtaY = etaY/realToPixelRatioLens

            iEtaX = pixelEtaX+pictureWidth/2
            jEtaY = -pixelEtaY+pictureHeight/2

            if (iEtaX)>pictureWidth-1 or (iEtaX)<1 or (jEtaY)>pictureHeight-1 or (jEtaY)<1:
                copy.setPixelColor(i,j,original.getPixelRed(i,j),original.getPixelGreen(i,j),original.getPixelBlue(i,j))
            else:
                copy.setPixelColor(i,j,original.getPixelRed(int(iEtaX),int(jEtaY)),original.getPixelGreen(int(iEtaX),int(jEtaY)),original.getPixelBlue(int(iEtaX),int(jEtaY)))

```

Figure 6: The code for lensing the image. Two for-loops scan across each pixel row and column of the original image and reassign each pixel its new appropriate color, determined with the $\vec{\xi} - \vec{\eta}$ relationship developed earlier.

```
# Drawing the black hole
copy.drawCircleFill(pictureWidth/2,pictureHeight/2,pixelSchwarzschildRadius)
#
# Displaying the lensed picture
copy.display()
print("Here is your 'gravitationally lensed' picture! Thanks for using this program.")
print("*****")
print("This program was created in 2016 by Kai Shinbrough as a final project for Astronomy 302, a course at Oberlin College. ")
raw_input()
```

Figure 7: Code for drawing the black hole, displaying the final ‘lensed’ image, and crediting the programmer.

References Cited

- [1] Binney, James, and Michael Merrifield. *Galactic Astronomy*. (Princeton University Press, Princeton NJ, 1998).
- [2] “Event Horizon Telescope Specifications.” Event Horizon Telescope. http://www.eventhorizontelescope.org/technology/building_a_larger_array.html.
- [3] Kayser, R. and S. Refsdal. *The difference in light travel time between gravitational lens images* (Astronomy & Astrophysics, 1983)
- [4] Laser Interferometer Gravitational-Wave Observatory (LIGO) *LIGO News* (Caltech, MIT, NSF 2015-2016)
- [5] Rykoff, Eli. GravLens3. Apple Software. Version 3.2. GravLens3 in App Store. February 19, 2015. <http://rykoff.co/gravlens3/> and <https://itunes.apple.com/us/app/gravlens3/id318275930?mt=8>.
- [6] Schneider, P., C. S. Kochanek, and Joachim Wambsganss. *Gravitational lensing: Strong, Weak, and Micro*. (Springer, Berlin, 2006).
- [7] Taylor, Cynthia. CS 150 Fall 2014. Oberlin College. <https://cs.oberlin.edu/~ctaylor/150/>.

I affirm that I have adhered to the Honor Code on this assignment.

-Kai Shinbrough