Through the Wormhole: Using Ray Tracing to Visualize the Wormholes of Interstellar

by **Quinn White, Chase Hanson, and Lily Whitler**

April 14, 2020

* * * * * *

Problem

Using the equations from Einstein's theory of general relativity, we will be attempting to recreate the wormhole visuals found in the film *Interstellar*. To do this, we will be following the work done by physicist Kipp Thorne and his collaborators at the VFX studio *Double Negative* (James et al., 2015), where they describe how they built the visuals for the wormhole in the film. The necessary math can be summarized as needing to know the wormhole metric and the solutions to the geodesic equation. First, the metric for the wormhole we will use is written as[1]

$$ds^2 = -(1+2\Phi)dt^2 + d\ell^2 + r^2(d\theta^2 + \sin^2\phi) \tag{1}$$

The metric is crucial to general relativity, as it is what describes the geometry of our space. Following the metric, we turn to the geodesic equation.

$$\frac{d^2x^\alpha}{d\zeta^2} + \Gamma^\alpha_{\mu\nu}\frac{dx^\mu}{d\zeta}\frac{dx^\nu}{d\zeta} = 0 \tag{2}$$

For purposes of numerical calculations, we will use a rewritten version of this equation known as a superhamiltonian

$$H = \frac{1}{2}\left[-p_t^2 + p_\ell^2\frac{p\theta^2}{r(\ell)^2} + \frac{p_\phi^2}{r(\ell)^2\sin^2\theta}\right] \tag{3}$$

The superhamiltonian can be split into 5 separate differential equations that we will need to solve in order to describe the quantities necessary to recreate our desired image.

$$\begin{aligned}
\frac{d\ell}{dt} &= p_\ell && (4)\\
\frac{d\theta}{dt} &= \frac{p_\theta}{r^2} && (5)\\
\frac{d\phi}{dt} &= \frac{b}{r^2\sin^2\theta} && (6)\\
\frac{dp_\ell}{dt} &= B^2\frac{dr/d\ell}{r^3} && (7)\\
\frac{dp_\theta}{dt} &= \frac{b^2}{r^2}\frac{\cos\theta}{\sin^3\theta} && (8)\\
& && (9)
\end{aligned}$$

Approach

In order to reproduce visuals similar to those in the film and paper, we will be required to create a ray tracing program in Python. This will essentially allow us to follow the path of a light beam as it moves through curved space to our placed "camera". To accomplish this, we will follow the method described in

[1] All variable definitions and other equations of importance are presented in the *Interstellar* paper.

the appendix of James et al. (2015). This involves choosing a discrete set of initial camera positions, as well as establishing a set of coordinates that allows us to determine the direction in which the light propagates. Once we have these positions and coordinates determined, we can compute the other necessary initial conditions that will allow us to numerically integrate Equations (4)-(8) (likely with RK4).

Using the data points generated by the numerical integration, the next step will be to use an interpolator, likely from SciPy, on the data set to produce a mapping between all camera and celestial coordinates, allowing for the distortion of any input image. Which interpolator we will use is currently not known, though some preliminary research suggests that using the SciPy `CubicHermiteSpline` function is the best choice for RK4 (J. M., 2013), as it is consistent with both the value and the first derivative; although it is possible for interpolating functions to match higher-order derivatives, it is unnecessary in this case as Equations (4)-(8) only include first derivatives.

Finally, for image recreation, the current best option seems to be the Python Imaging Library *Pillow*. Within this package is an `Image` module, which has an `eval` function, which seems to allow us to apply our map to any original image. Alternatively, the scikit-image library allows for arbitrary image transformations via its `skimage.transform.warp` function, which could also be used to apply our mapping from camera to sky. This is currently a point of inexperience however, so it is possible this method or the functions we use will change.

Objectives

1. Establish initial camera position and compute other necessary initial conditions
2. Using our initial conditions and an RK4 integrator, numerically solve equations 4-9 to generate our map
3. Using our map and the python modules described above, transform input images to reflect the wormholes presence (see fig 7. of (James et al., 2015) for examples of desired effect).
4. Alter the properties of the wormhole to see how a change in its geometry impacts the resulting image.
5. (OPTIONAL OBJECTIVE) Recreate the Einstein rings described in (James et al., 2015)
6. (OPTIONAL OBJECTIVE) Do animated passes around and through the wormhole

References

(https://scicomp.stackexchange.com/users/127/j-m) J. M. Intermediate values (interpolation) after runge-kutta calculation. Computational Science Stack Exchange, 2013. URL https://scicomp.stackexchange.com/q/7363. URL:https://scicomp.stackexchange.com/q/7363 (version: 2013-05-24).

Oliver James, Eugénie von Tunzelmann, Paul Franklin, and Kip S. Thorne. Visualizing interstellar's wormhole. *American Journal of Physics*, 83(6):486–499, 2015. doi: 10.1119/1.4916949. URL https://doi.org/10.1119/1.4916949.