Optical Imaging as a Linear System

Robert Jacobs
Department of Electrical and Systems Engineering
Washington University in St. Louis
St. Louis, MO, United States
robert.jacobs@wustl.edu

Max Cusick
Department of Electrical and Systems Engineering
Washington University in St. Louis
St. Louis, MO, United States
c.marguerite@wustl.edu

Ben Ko
Department of Electrical and Systems Engineering
Washington University in St. Louis
St. Louis, MO, United States
ben.k@wustl.edu

***Abstract* – This case study focused on ray tracing and imaging systems. We looked at how rays travel and interact with lenses as well as how to focus and create sharp images. We developed an imaging system based on various transformation matrices and tested it with a wide variety of values. We then applied the system to lightField.mat to create a sharp image out of a mystery object.**

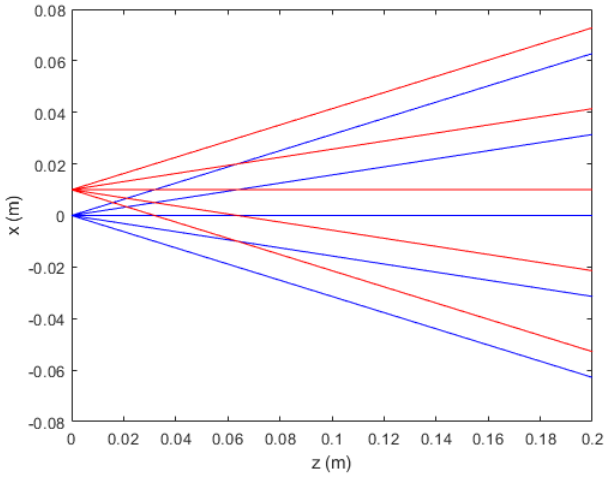
I. Introduction

In this case study, we learned how rays travel and interact with lenses and objects. We also learned the basics of how imaging systems are made and function. Our work centered around making a basic imaging system with one lens and using the lightField.mat file to supply rays to manipulate and show. Our imaging system started with just following the paths of some generated rays and expanded into following the rays from a start position, through a lens, and to their convergence point. It upheld Equation 11 to make clear images where all rays converged at a point after going through the lens. We then manipulated the "M" matrices to create a system to sharpen the rays in lightfield.mat.

II. Methods

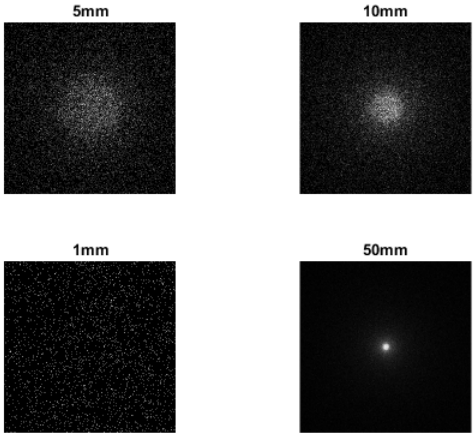
A. *Exploring raytracing*

Part 1.3.1 started with us generating a ray diagram to show how rays originating at various points traveled along the xz axis. This was done in the file exploreRayTracing.m. To make the rays we set the initial x and z values then made two vectors to hold the angles we wanted our arrays to travel along. The path the arrays traveled was then modified by taking the appropriate calculations based off the matrix M_b and plotting the values found for rays_in and rays_out. This process was done twice, once for the rays starting at (0,0,0) and once for the rays starting at (10,0,0).



In part 1.3.2 we are tasked with using rays2img() to render images of the provided rays. We found that it was not possible to see the object that generates the rays. All we could see is a blurry sphere that looks like an image of the moon. We then iterated through different sensor width values. None unveiled the actual identity of the object, and this is because increasing or decreasing the sensor width only either zooms in or out. It does not sharpen. Next, we tried increasing the number of sensor pixels, this too was not fruitful, but we did find that a 500x500 pixel size was a good baseline for the proceeding questions. We cannot recover a sharp image with changing pixel size alone since all it does is set the size of the output. Next, we used equation 7 and M_d to propagate the rays with varying values of d. We were not able to discern a physical object from any of our values. After propagation, the larger the d value (distance) the more scattered the rays are in

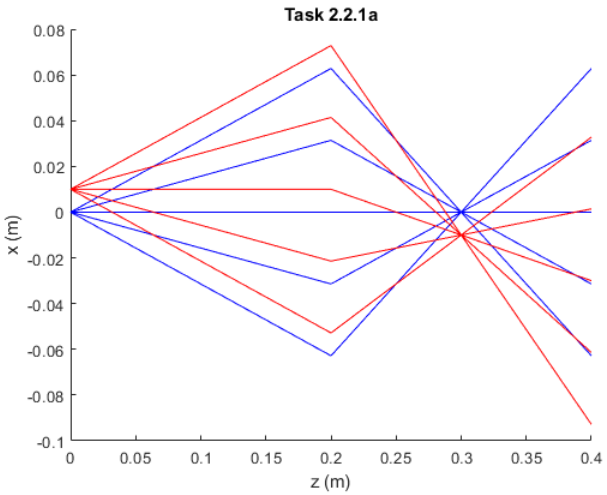
the image.



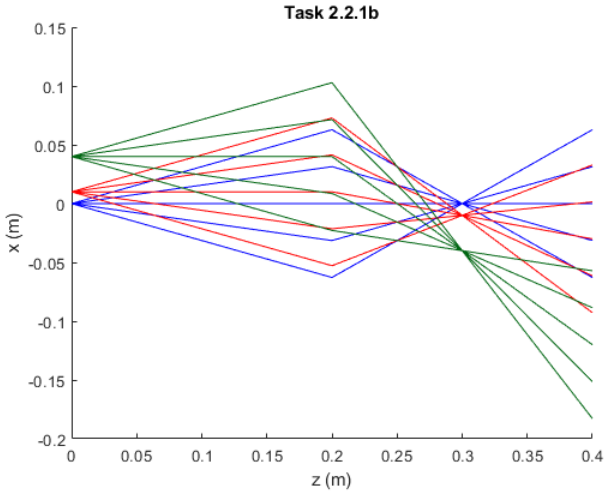
For tasks 1.3.2d and 1.3.3, we found that by solely using propagation matrix M_{d2} we are only letting the rays of light travel in the free space for a distance of d. So, changing d alone will never sharpen the image to the point where we can discern the source of light.

B. Visualizing imaging using ray tracing

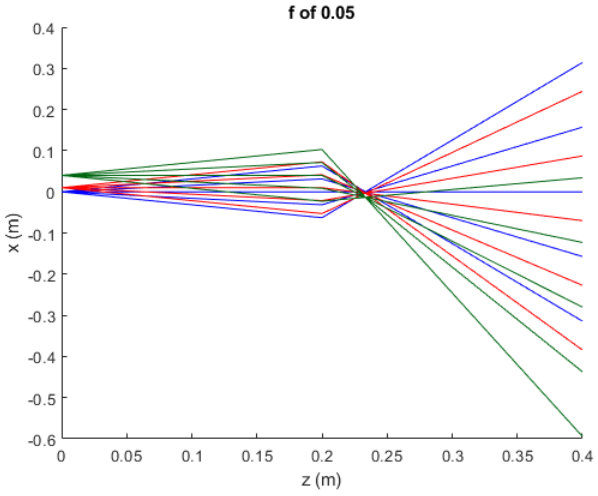
Part 2.1 expanded on the rays initially made in part 1 and added a lens. This was done in the rayTracingWithLens.m file. The rays were generated similarly to part 1.3.1, with the added matrices M_2 and M_f to store the d2 and f transformation matrices. The rays were calculated through three phases. An initial phase exactly like part 1.3.1, a lens phase to calculate the change in angle due to the lens, and a final phase to calculate the new positions of the rays after the lens was applied. These rays were then graphed with a slight modification to ray_z to account for the extra ray transformations. This general plotting strategy was followed for all of 2.2.1.

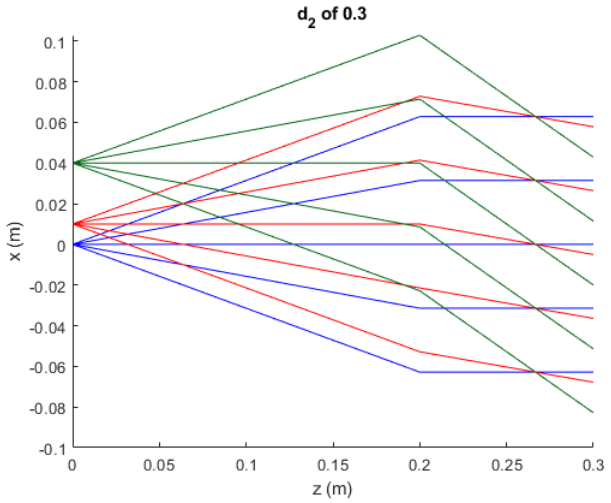


For Task 2.2 1b, a third origin was added at x = .04. This led to rays which were initially higher up on the plot than the previously generated rays and lower on the plot after the lens was applied. Though the x values for the rays were different they still converged in the same z plane as the other rays. This result is expected because the new ray has the same z plane as the others and passes through the same lens just at a different part.



For Task 2.2 1c, the system of three separate origins from part 2.2.1b was modified to take in a variety of values of d2 and f. It was seen that the systems still formed an image for f values of .1 or less since the rays converged. The system did not form an image for all tested d2 values since some rays did not converge after passing through the lens.





For Task 2.2.1d, the equation is still shown to hold up since the values that break the relationship do not form an image. Both f and d2 values outside of what is expected by equation 11 don't converge after passing through the lens so they cannot show an image.

For part 2.2.2.a, we used the equation below to help make the M matrix:

$M_{d2} * M_f * M_{d1}$ = matrix M

$$M = \begin{pmatrix} -\frac{d_2}{d_1} & 0 & 0 & 0 \\ \frac{-d_2-d_1}{d_1 d_2} & -\frac{d_1}{d_2} & 0 & 0 \\ 0 & 0 & -\frac{d_2}{d_1} & 0 \\ 0 & 0 & \frac{-d_2-d_1}{d_1 d_2} & -\frac{d_1}{d_2} \end{pmatrix}$$

For part 2.2.2.b, we applied the M matrix to see the relationship between the input and output rays when it was used.

$$\begin{bmatrix} x_2 \\ \theta_{x2} \\ y_2 \\ \theta_{y2} \end{bmatrix} = \begin{bmatrix} -\frac{x_1 \cdot d_2}{d_1} \\ \frac{x_1(-d_2-d_1) - \theta_{x_1} d_1^2}{d_1 d_2} \\ -\frac{y_1 \cdot d_2}{d_1} \\ \frac{y_1(-d_2-d_1) - \theta_{y_1} d_1^2}{d_1 d_2} \end{bmatrix}$$

The interesting relationship between the input ray and output is that locations of output ray could be rewritten in terms of d1 and d2 as seen in the vector above.

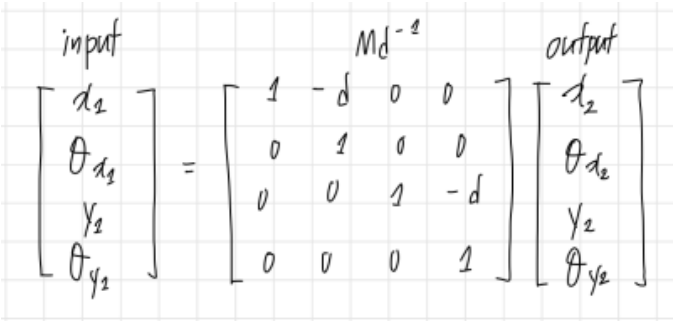
For part 2.2.2.c, ray-transfer matrix M refers to a single optical system that composes of first propagation, a convergent lens, and second propagation after rays go through a lens.

The "job" of the given imaging system is to first "gather" light rays passing through the system using the convergent lens and have them converge at a set focal point. By adjusting focal length, using the relationship between focal length and propagation constants, we can adjust the sharpness of the image rather than letting the rays to infinitely propagate through free space like Part 1.

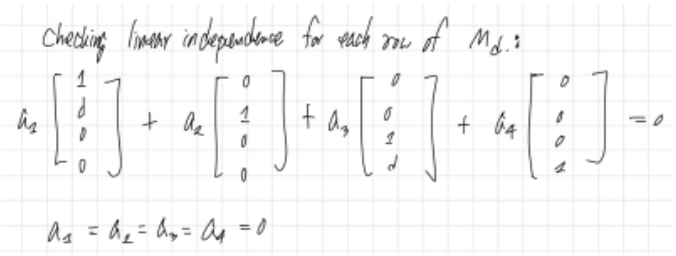
For part 2.2.3, we used the propagation matrix M_d and the lens matrix M_l similarly to how they were used in 1.1 and 2.1. The main difference being that for this question we lacked a d_1. Because of this, we put the rays through the M_{d2} and M_l matrices while changing d2 and f until we got a clear image. Our final f value was .1 and our final d2 value was .2. After entering the rays through our imaging system and using rays2img() to display the final image, we found that the image was of a virus. The virus is about .002mm, and possibly a coronavirus though that is hard to concretely determine since there are other viruses of similar shape and size.

C. Toward computational imaging

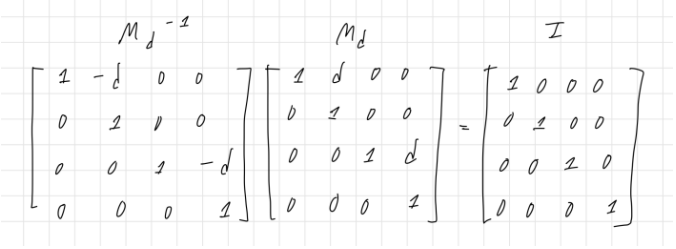
Given how optical imaging could be modeled as linear system with input rays multiplied by propagation matrix, the following relationship would stand if the inverse of propagation matrix exists.



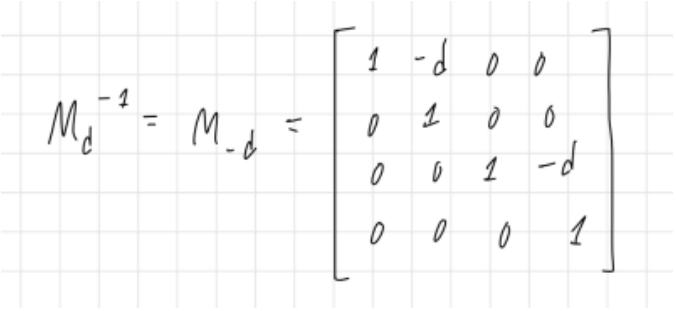
For part 3.2.1, we found that since matrix M is a 4 x 4 square matrix and the columns and rows are linearly independent, matrix M is invertible, and therefore, does exist.



The following shows the derivation of matrix inverse using characteristics of inverse matrices. If we multiply original propagation matrix with matrix inverse, we will get a 4x4 identity matrix.



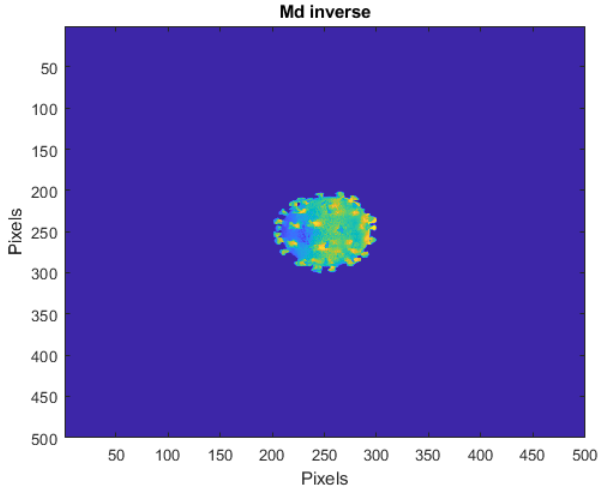
For part 3.2.2, we found that both and have a same mathematical expression as the following:



As the equation shows, they are equivalent, and the equivalence does make a physical sense.

The physical meaning of M_d^{-1} is that it is a "tracing" matrix for input rays when multiplied by output ray vector. This could be interpreted as regular propagation matrix, however, where the destination of the rays would now be a starting point and vice versa. Therefore, two matrices bear same physical meaning that they are used to find the position and angle of the original ray.

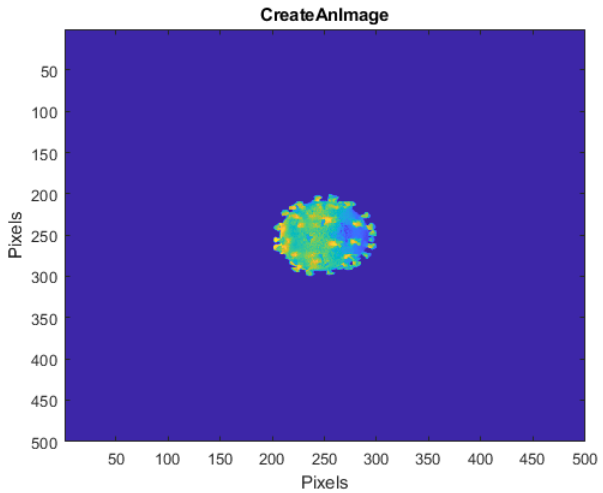
For part 3.2.3, we implemented the matrix inverse Md to using a d value of 0.2. This generated the sharp image shown below. We noticed a cool relationship between this version of the virus and the one created in part 2. The virus is flipped: one can see the blue shadow is now on the left side whereas it was previously on the right side.



For Task 3.2 Question 3, we used d value of 0.2

III. Results and Discussion

Our imaging system was able to create a sharp image from lightField.mat. We were able to successfully develop a system that could create a sharp and focused image without knowing the value of d1. The resulting image is below:



IV. Conclusion

In conclusion we successfully made an imaging system and uncovered the mystery object in lightField.mat. Our system was successfully able to make a sharp image after some adjustment to d_2 and f. In creating our system we found that the relationship between focal length and distances to hold up, and that input rays could be written as matrix-matrix multiplication with inverse propagation matrix and output ray. If we were to expand on this project in the future we may want to make a system which could work on moving images or adjust d_2 and f automatically without human verification of sharpness. This would be closer to what modern ray tracing is improving on right now and would enhance the capabilities of our system. Overall, we have made a system which works as intended and is able to sharpen images when given a set of rays.