ESE 105 Case Study 4: Imaging and Ray Tracing

DUE: Friday, December 6, 2019 at 5 PM

In this case study you will use the skills and methods you have learned in linear algebra and MATLAB to simulate how an optical imaging system, such as a digital camera, a microscope, or your eye, forms images in free space. The methodology you will use is called *ray tracing*.

Ray optics is the simplest theory of light and comprises a set of geometrical rules that describe how rays travel. Therefore, ray optics is also called *geometrical optics*. These geometrical rules are simple to implement in the language of linear algebra.

Your goals are threefold:

1. Simulate how a single lens bends rays. Calculate where images form and how they are magnified/shrunk as a function of the position of an object relative to the lens.
2. Analyze how an imaging system's aperture affects its depth of field.
3. Explore an application of computational imaging: digitally refocus an image without using a physical lens.

1 Part 1: Ray optics

1.1 Postulates of ray optics

1. Light travels in the form of rays. They are emitted by light sources and can be observed when they reach an optical detector.
2. An optical medium is characterized by its refractive index $n \geq 1$, defined as the ratio of the speed of light in free space c_0 to that in the medium c:

$$n = \frac{c_0}{c}. \tag{1}$$

Therefore, the time light takes to travel a distance d in a medium n is given by

$$t = \frac{d}{c} = \frac{nd}{c_0}. \tag{2}$$

3. Fermat's Principle: *Light rays travel along the path of least time.* This concept can be used to derive Snell's law (Fig. 1) at the interface between two materials:

$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \tag{3}$$

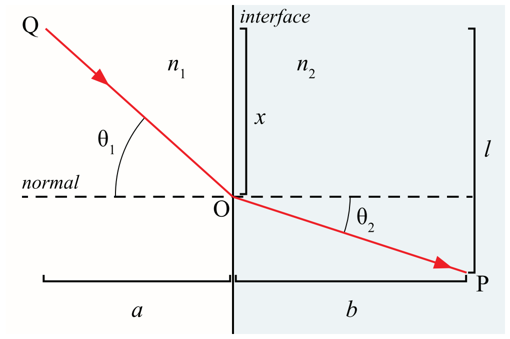

Figure 1: Diagram to derive Snell's law, https://commons.wikimedia.org/wiki/File:Snells_law_Diagram_B_vector.svg

1.2 Matrix optics

A ray is described by its position y and its angle θ with respect to the primary axis z of travel, called the optical axis. To simplify the situation, let's first assume that the rays travel within a single 2D plane, the yz plane.

For *paraxial rays*, that is, rays that travel mostly parallel to the optical axis such that $\sin\theta \approx \theta$, the position and angle of a ray before and after an optical system (Fig. 2) can be described by a 2×2 matrix called the ray-transfer matrix $\boldsymbol{M}$ with entries A, B, C, and D.

$$\begin{bmatrix} y_2 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} y_1 \\ \theta_1 \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} y_1 \\ \theta_1 \end{bmatrix} \tag{4}$$

The ray-transfer matrix $\boldsymbol{M}$ completely characterizes any optical system.

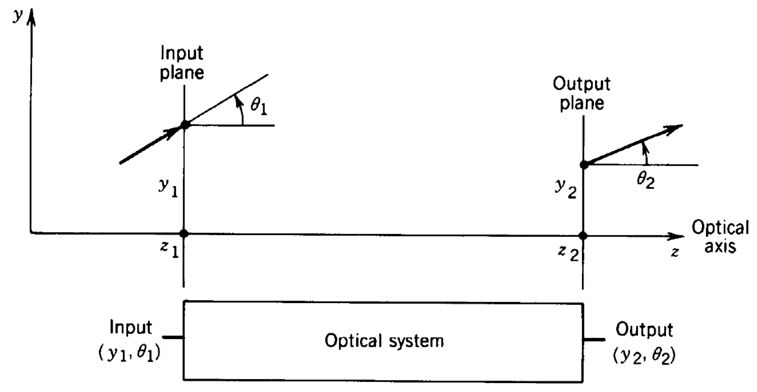

Figure 2: A ray enters an optical system at position y_1 and angle θ_1 and leaves the system at position y_2 and angle θ_2. Figure from [1].

1.3 Free-space propagation

Rays travel in straight lines. Therefore, after the ray traverses a distance d along the optical axis (Fig. 3), we know

$$y_2 = y_1 + d\tan(\theta_1) \approx y_1 + d\theta_1 \tag{5}$$
$$\theta_2 = \theta_1. \tag{6}$$

Therefore, the corresponding ray-transfer matrix is given by

$$\boldsymbol{M}_d = \begin{bmatrix} 1 & d \\ 0 & 1 \end{bmatrix}. \tag{7}$$

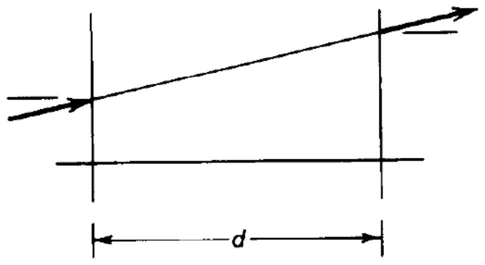

Figure 3: Free-space propagation. Figure from [1].

1.4 Transmission through a thin lens

For a thin lens (Fig. 4), we assume that a ray exits at the same position as it enters. A lens's focal length f is defined such that for a series of incoming rays traveling parallel to the optical axis ($\theta_1 = 0$), the exit angle θ_2 of each ray is given by

$$\theta_2 = -\frac{y}{f}, \tag{8}$$

that is, the rays converge at a distance f behind the lens, called its focal point. If $\theta_1 \neq 0$, then $\theta_2 = -y/f + \theta_1$. We therefore have

$$\boldsymbol{M}_f = \begin{bmatrix} 1 & 0 \\ -1/f & 1 \end{bmatrix}. \tag{9}$$

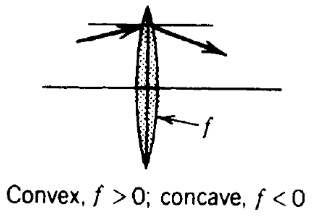

Figure 4: Propagation through a thin lens. Figure from [1].

1.5 Matrices of cascaded optical components

A series of optical elements with ray transfer matrices $\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_N$ (Fig. 5) is equivalent to a single optical component with ray-transfer matrix $\boldsymbol{M}$ given by

$$\boldsymbol{M} = \boldsymbol{M}_N \cdots \boldsymbol{M}_2 \boldsymbol{M}_1. \tag{10}$$

Note the order of operations: rays are first transformed by the first optical component $\boldsymbol{M}_1$, then by $\boldsymbol{M}_2$, and so on.

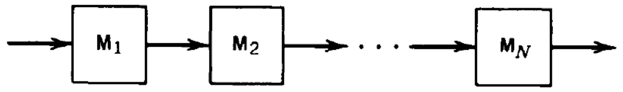

Figure 5: A series of optical components. Figure from [1].

1.6 Task

Complete the function `[y_out,theta_out] = simRayProp(M,y_in,theta_in)` provided in `simRayProp.m`. This function should implement Eq. 4, given the following parameters as inputs:

- `M`: a 2×2 matrix representing the optical component or imaging system
- `y_in`: a $1 \times N$ vector representing y_1, the locations of N rays before the optical component $\boldsymbol{M}$
- `theta_in`: a $1 \times N$ vector representing θ_1, the angle of travel of N rays before the optical component $\boldsymbol{M}$

Your function should provide the following outputs:

- `y_out`: a $1 \times N$ vector representing y_2, the locations of N rays after the optical component $\boldsymbol{M}$
- `theta_out`: a $1 \times N$ vector representing θ_2, the angle of travel of N rays before the optical component $\boldsymbol{M}$

2 Part 2: A simple imaging system

Consider a single thin lens as shown in Fig. 6. All rays emitted from point P_1 on an object at a distance z_1 in front of the lens converge to a *single* point P_2 at a distance z_2 behind the lens. Therefore point P_2 is an image of the object point P_1. This imaging property will occur for any point P_1 that is within the xy plane located at a distance z_1 in front of the lens; that is, there is a *one-to-one mapping* between any pair of points at distances z_1 and z_2 from the lens.

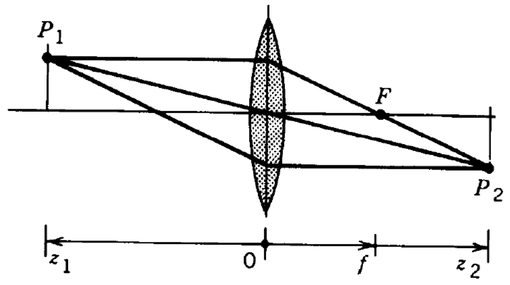

Figure 6: Image formation by a thin lens. The object distance z_1, image distance z_2, and lens focal length f are all positive as depicted. Figure from [1].

We can model the propagation of rays through this imaging system by using the product of three ray transfer matrices:

$$\begin{bmatrix} y_2 \\ \theta_2 \end{bmatrix} = \boldsymbol{M}_{z2}\boldsymbol{M}_f\boldsymbol{M}_{z1}\begin{bmatrix} y_1 \\ \theta_1 \end{bmatrix} \tag{11}$$

where

$$\boldsymbol{M}_{z2} = \begin{bmatrix} 1 & z_2 \\ 0 & 1 \end{bmatrix}, \quad \boldsymbol{M}_f = \begin{bmatrix} 1 & 0 \\ -1/f & 1 \end{bmatrix}, \quad \boldsymbol{M}_{z1} = \begin{bmatrix} 1 & z_1 \\ 0 & 1 \end{bmatrix}. \tag{12}$$

Therefore, the optical system $\boldsymbol{M} = \boldsymbol{M}_{z2}\boldsymbol{M}_f\boldsymbol{M}_{z1}$ is an imaging system if $y_2 = my_1$ for any value of θ_1. We call the scaling constant m the magnification of the imaging system.

2.1 Tasks

We would like to use MATLAB (and the code you wrote in section 1.6) to simulate the imaging properties of a single lens imaging system. We choose the lens focal length f to be 100 mm.

Hint: Given an object position z_1 and a lens focal length f, you can use Eq. 11 to solve for a unique $\boldsymbol{M}_{z2}$ that makes the total system $\boldsymbol{M} = \boldsymbol{M}_{z2}\boldsymbol{M}_f\boldsymbol{M}_{z1}$ obey the laws for an *imaging system*.

1. Suppose an object is located at a distance $z_1 = 250$ mm in front of the lens. Using Eq. 11, simulate the propagation of three special rays from a single point P_1 on the object to its image point P_2:

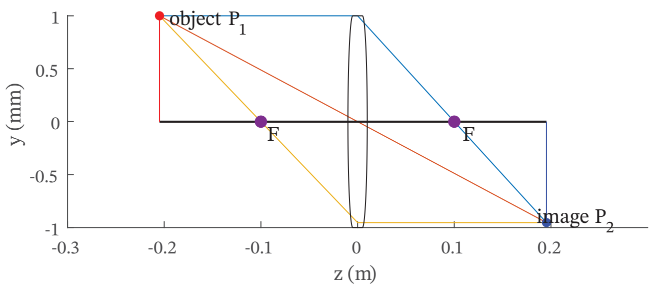

Figure 7: Image formation by a thin lens, simulated in MATLAB. Your 2D ray diagrams should resemble this figure.

(a) A ray from P_1 traveling parallel to the optical axis ($\theta_1 = 0$), similar to the blue ray in Fig. 7

(b) A ray traveling from P_1 through the center of the lens, similar to the red ray in Fig. 7

(c) A ray traveling from P_1 through the front focal point F located a distance f in front of the lens, similar to the gold ray in Fig. 7

Create a 2D ray diagram (see example in Fig. 7) that shows how these rays travel from the object point P_1 to the image point P_2. Be sure to show:

- The path of each ray
- The object point P_1, image point P_2, the front focal point F located a distance f in front of the lens, and the back focal point F located a distance f behind the lens
- The location of the lens
- The optical axis

2. Create a second 2D ray diagram, similar to that above, for an object located at $z_1 = 45$ mm in front of the lens.

3. For a range of object distances z_1 from 1 mm to 1 m, compute the locations of the corresponding images z_2.

   - Plot z_2 versus z_1.
   - Overlay the theoretical curve given by the imaging equation:

$$z_2 = \left(\frac{1}{f} - \frac{1}{z_1}\right)^{-1} \tag{13}$$

   - Does your MATLAB simulation agree with the theory?
   - A "real" image is defined as one whose position $z_2 > 0$, while a "virtual" image is defined as one whose position $z_2 < 0$. Does this definition make sense? Why?

4. For the same range of object distances above, compute the magnification m of each image, given by

$$m = \frac{y_2}{y_1}, \tag{14}$$

where y_1 is the height of object point P_1 and y_2 is the height of the image point P_2.

   - Plot your computed magnification m versus z_1.

- Overlay theoretical magnification curve given by:

$$m = \left(1 - \frac{z_1}{f}\right)^{-1}. \tag{15}$$

- Does your MATLAB simulation agree with the theory?
- An "upright" image is defined as one whose magnification $m > 0$, while an "inverted" image is defined as one whose $m < 0$. Based on your simulations, what is the relationship between real, virtual, upright, and inverted images?

3 Part 3: Depth of field

An imaging system's depth of field (DOF) is the distance around the focal plane where objects appear "acceptably sharp" in an image. As you found in Part 1, an optical imaging system can precisely focus on only one plane at a time. However, the decrease in sharpness is gradual, so that within the DOF, the blurriness within an image is usually imperceptible. If an object lies outside the DOF, then it will be noticably blurry.

In some cases, it may be desirable to have all objects within an image be sharp, and therefore a large DOF is appropriate. In other cases, a small DOF may be more effective, emphasizing the subject while deemphasizing the background behind the subject.

For a given object and optical system, the DOF is usually controlled by the lens aperture diameter. This diameter is quantified by the imaging system's f-number, denoted by "$f/\#$", where

$$\# = \frac{f}{D}, \tag{16}$$

f is the focal length of the imaging system, and D is the diameter of the aperture (see Fig. 9). For example, a lens with an f-number of 1 (the same as "f/1") means $f = D$, or its focal length is equal to its diameter. Therefore, larger f-numbers correspond to smaller apertures.

3.1 Tasks

An object that is out of focus will be blurred on the image sensor; that is, the rays originating from point P_1 will no longer map to a single point P_2 in the image plane but a cluster of points. We define a circle of confusion as the smallest possible circle of radius R such that it encompasses all of the rays originating from P_1.

Suppose we have a single lens ($f = 200$ mm) imaging system as shown in Fig. 9. A camera sensor is placed at a distance z_2 behind the lens such that it images an object located at a distance $z_1 = 2$ m in front of the lens.

1. For an out-of-focus object placed at a position 4 m in front of the lens, plot the radius R of the circle of confusion versus f-number for f-numbers ranging from 1.4 to 22.
   Hint: In order to model different aperture sizes, use different values of θ_1 for rays originating from P_1, such that these rays intersect the lens at a height $y = f/(2(fNumber))$, as shown in Fig. 8.

2. A cartoon turkey of width 200 mm lies 2 m in front of the lens, and a cartoon snow globe of width 200 mm is placed 4 m in front of the lens, as shown in Fig. 9.

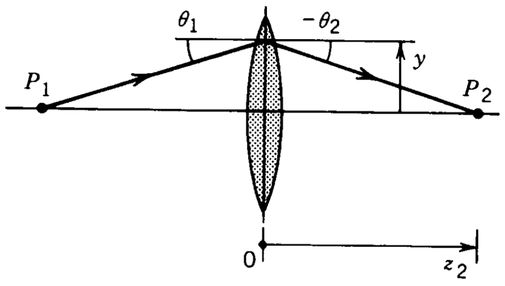

Figure 8: Ray bending by a thin lens. The image distance z_2 and lens focal length f are positive as depicted. Figure from [1].

(a) Choose two f-numbers such that one corresponds to a large DOF (both the in-focus and background objects are sharp) and one corresponds to a small DOF (only the in-focus object is sharp).

(b) Simulate the images created by these two imaging systems captured by a square 25 mm imaging sensor, similar to Fig. 10.

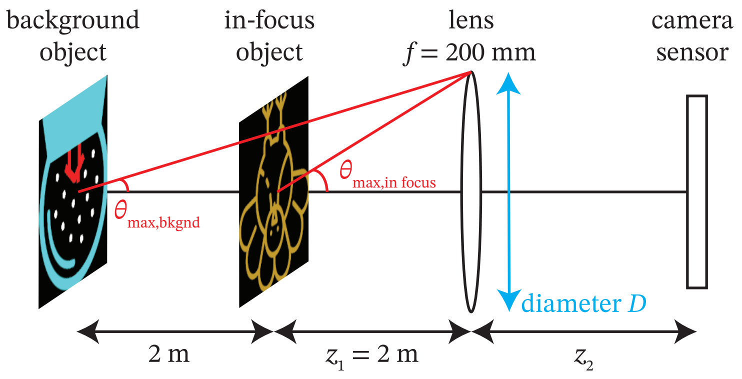

Figure 9: Imaging system for part 3. The sizes of the objects and the distances z_1 and z_2 are not shown to scale. Images of these objects are adapted from https://commons.wikimedia.org/wiki/File:Turkey_(example).svg and https://commons.wikimedia.org/wiki/File:Snow_Globe_-_The_Noun_Project.svg.

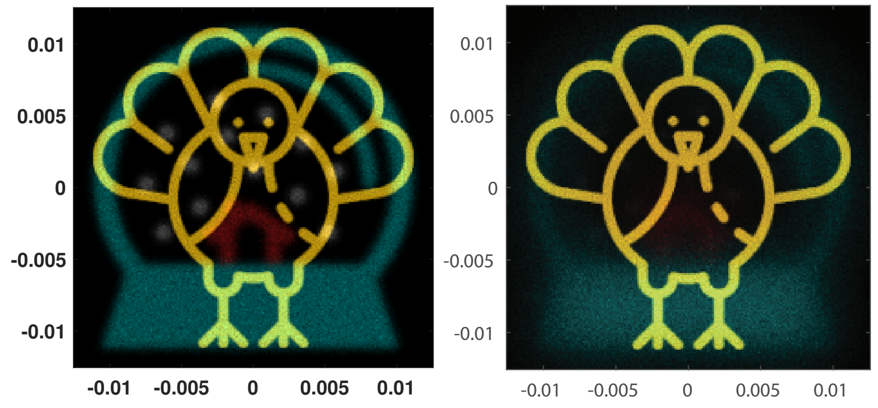

Figure 10: Example images produced by an imaging system with (left) a large DOF and (right) a small DOF. Axes labels are in units of meters.

You will use the provided function `img2rays(img,width,numRays,maxRayAngle)` to convert the objects to rays and the function `rays2img(x_in,y_in,color,width,Npixels)` to convert the rays back into an image, along with the function `simRayProp(M,y_in,theta_in)` you wrote in Section 1.6.

`function [x_out,y_out,theta_x_out,theta_y_out,color] = img2rays(img,width,numRays,maxRayAngle)` accepts the following inputs:

- `img`: An $H \times W \times 3$ matrix representing an image to be converted into rays. The third dimension represents the colors red, green, and blue respectively.
- `width`: A scalar that represents the total width of the image in meters.

- `numRays`: A scalar that specifies how many rays to generate. A good ballpark number of rays to create realistic images is 1 million.
- `maxRayAngle`: A scalar that specifies the maximum ray angle θ (in radians) to be used for the fan of rays generated from each point on the object.

`function [x_out,y_out,theta_x_out,theta_y_out,color] = img2rays(img,width,numRays,maxRayAngle)` generates the following outputs:

- `x_out`: An $1 \times numRays$ vector representing the x position of each ray.
- `y_out`: An $1 \times numRays$ vector representing the y position of each ray.
- `theta_x_out`: An $1 \times numRays$ vector representing the θ_x (in radians) propagation direction in the xz plane of each ray.
- `theta_y_out`: An $1 \times numRays$ vector representing the θ_y (in radians) propagation direction in the yz plane of each ray.
- `color`: An $3 \times numRays$ matrix representing the RGB color of each ray.

`function [img,x,y] = rays2img(x_in,y_in,color,width,Npixels)` accepts the following inputs:

- `x_in`: A $1 \times N$ vector representing the x position of each ray.
- `y_in`: A $1 \times N$ vector representing the y position of each ray.
- `color`: An $3 \times N$ matrix representing the RGB color of each ray.
- `width`: A scalar that specifies the total width of the image sensor in meters.
- `Npixels`: A scalar that specifies the number of pixels along one side of the image sensor.

`function [img,x,y] = rays2img(x_in,y_in,color,width,Npixels)` generates the following outputs:

- `img`: An $Npixels \times Npixels \times 3$ matrix representing an image captured by an image sensor of width $width$ with a total number of pixels $Npixels^2$.
- `x`: A 1×2 vector that specifies the x position of the left and right edges of the imaging sensor in meters.
- `y`: A 1×2 vector that specifies the y position of the bottom and top edges of the imaging sensor in meters.

4 Part 4: Computational imaging challenge (light field imaging)

You are given a simulated dataset containing the locations and propagation directions for $N = 7.5 \times 10^6$ rays, stored in `lightField.mat`. (Be sure to use a broadband internet connection to download the case study .zip file; the light field dataset alone is ~100 MB!) This file contains:

- `ray_color`: a $3 \times N$ matrix describing the RGB color of N rays
- `ray_theta_x`: a $1 \times N$ vector describing the propagation direction (in radians) in the xz plane of N rays

- `ray_theta_y`: a $1 \times N$ vector describing the propagation direction (in radians) in the yz plane of N rays
- `ray_x`: a $1 \times N$ matrix describing the x position of N rays
- `ray_y`: a $1 \times N$ matrix describing the y position of N rays

Using the provided function `rays2img()` on this dataset, we obtain the image in Fig. 11, which isn't very interesting. However, performing the correct transformation(s) $\boldsymbol{M}$ on this dataset should enable you to find 4 "hidden" images. Skeleton code to load and visualize the rays in the dataset is located in `visualizeLightField.m`.

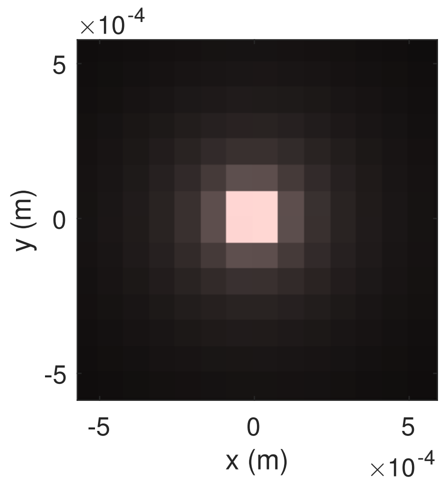

Figure 11: Image created from the light field dataset `lightField.mat` using the function `rays2img()`. A movie showing 3 of the 4 images contained in the dataset is on Canvas in the Case Study 4 module.

For your report, perform the following tasks and answer the following questions, using supporting equations and/or figures when possible.

1. Describe a methodology to find the 4 "hidden" images in the dataset.
   - How do you know that you have successfully found an image? What mathematical quantity or metric can you use to conclude you have found an image?
   - How well does your proposed metric work for finding the images?
2. Plot pictures of each image in the dataset.
3. Traditional cameras cannot refocus images after they are taken, since their images only capture ray positions (i.e., only `ray_x` and `ray_y` and not `ray_theta_x` and `ray_theta_y` in this dataset). Describe, in terms of what you have learned in this class, why cameras cannot create sharp images without physically changing their attached lens(es), i.e., refocusing their lens(es).
4. Speculate on the possibility of building a camera that does not need any lenses, i.e., a lens-less camera. What data would this camera need to capture?

5 Tips

- Use matrix and vector operations within your code where possible. In particular, the function `simRayProp` should use vector math to compute new ray positions and angles.
  Hint: It is possible to implement Eq. 4 using one line of code for *any* number of rays contained in `y_in` and `theta_in`!

- You have complete freedom to implement code for all other parts of the case study as you wish.
- For the 3D simulations in parts 3 and 4, we will assume that the ray positions and angles in the xz plane are independent of the ray positions and angles in the yz plane. Therefore, you can compute the ray propagations separately using the same ray-transfer matrix $\boldsymbol{M}$:

$$\begin{bmatrix} x_2 \\ \theta_{x2} \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} x_1 \\ \theta_{x1} \end{bmatrix} \tag{17}$$

$$\begin{bmatrix} y_2 \\ \theta_{y2} \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} y_1 \\ \theta_{y1} \end{bmatrix} \tag{18}$$

  Food for thought: Is this assumption reasonable?
- To create 2D ray diagrams, the MATLAB functions `plot()`, `scatter()`, `rectangle()`, and `text()` may be useful.
  In particular, for $M \times N$ matrices x and y, the `plot(x,y)` function will simply plot each column of x against each column of y, with connecting lines between each point (x_{ij}, y_{ij}) for a fixed j.

6 What to turn in

1. Any MATLAB `.m` files you write or modify, including `simRayProp.m`
2. PDFs or Word DOCs for each of the MATLAB files above. Use the `publish()` command similar to what you've produced for previous MATLAB homeworks.
3. A 2-4 page report, answering each question within the case study and documenting your design choices. *Use the provided Word template.* Make sure that your report clearly states/presents:
   (a) Answers to each question within the case study
   (b) Justification to your answers using calculations or plots from your MATLAB code when possible
   (c) Design choices that you've made
4. A signed version of the provided honor code

7 Rubric

- Correctness of MATLAB code - 40%
  - Implementation of ray-transfer matrix to transform input rays to output rays
  - Calculation and plotting of image positions, image magnifications, and the radius of the circle of confusion
  - Plotting of 2D ray diagrams, images showing the depth of field of the imaging system, and images in the light field dataset
- Presentation - 20%
  - Plots are easy to read and interpret, with appropriate font sizes, line widths, axis labels, etc.
  - Report should be well-organized, concise, and clearly written.

- Programming style - 20%
- Study design - 20%
  - Strategy for choosing the f-numbers of the imaging system for large and small depths of field
  - Strategy for finding images in the light field dataset

References

[1] B. E. A. Saleh, *Fundamentals of photonics*, 2nd ed., ser. Wiley series in pure and applied optics. Hoboken, N.J.: Wiley-Interscience, 2007.