6.837 Depth of Field Simulation with Ray Tracing

Andy Kuang
Massachusetts Institute of Technology
akuang@mit.edu

Ka Wai (Joanne) Lee
Massachusetts Institute of Technology
kwlee@mit.edu

December 6, 2018

1 Motivation

With the increased prevalence of social media and better cameras, there has been increasing demand for various cool photo effects. One desired photo effect is called depth of field, which is when a photo has high sharpness on one part of the photo, while having blurring effects on the rest of the image. This effect is especially good for portrait photos because it creates the illusion that the sharp object is the "center of attention". The effect is also desired because it is often more photo-realistic and resembles that of an SLR camera. In this project, we will attempt to simulate this effect with a ray tracing system.

To capture this effect, we must first understand the phenomenon from an optics perspective. When taking a photo, the shape of the lens causes the light rays to change angles in such a way that only objects of a specific distance away are sharp, while the rest of the image is blurry. We can see a diagram of this in Figure 1.

2 Background Work

There are many software programs that perform optics simulation on a lens level, such as Zemax, Code V, or Oslo. These programs all simulate light rays and how each individual lens affects the angle of each light ray. However, the degree of complexity in these software programs is incredibly high and hence almost impossible to replicate for a final project for a semester-length class. Therefore, we have focused on simpler simulations.

In computer graphics, prior work to simulate depth of field has been done with a modification to the classic ray tracing algorithm. Skee Lee [1] has demonstrated an "averaging" approach, in which a square bracket with a set aperture is placed with each image plane pixel as the center, and the color of the pixel is the average of all the colors of the depth of field (DOF) rays shot through the squares in the bracket.

For a given pixel on the image plane, the focal point $\mathbf{p}$ can be calculated as follows [1]:

$$\mathbf{p} = \mathbf{e} + \frac{(di + f)dp}{di}\mathbf{d}$$

where $\mathbf{e}$ is the location of the eye, $\mathbf{d}$ is the direction of the eye, di is the distance from the eye to the image plane, f is the focal length, and dp is the distance from the eye to the pixel in question.

3 Approach

3.1 Depth of field

We start with the ray tracer provided in problem set 4 of 6.837.

We adopt the basic approach as outlined in [1]. The user supplies a specific focal length that they wish to use. At this focal length, we define a focal plane, placed between the image plane and the scene to be rendered.

For each pixel in the image plane, we perform the following steps:

1. Compute the focal point P on the focal plane by shooting a ray from the eye through the image pixel to the focal plane, and find the intersection.

2. Define a square bracket around the pixel, with a given aperture. Shoot depth of field rays from each square in this square bracket through the focal point, and average the colors that result from the hits with the scene. The color to shade at the image pixel is this averaged color.

3.2 Jittering

The basic depth of field implementation accomplishes our basic goal; it focuses on certain aspects of the scene and blurs out others. However, the resulting photo is not as realistic as it could be. The reason for this non-realism is that we shoot depth-of-field rays through the center of each square of the square bracket. This causes undesirable grids to form in the background, as a result of our equally spaced sampling points.

Therefore for enhanced realism, we added jittering to our depth of field rays. Instead of shooting depth of field rays through the center of each square in the square bracket, we adopt a stratified sampling approach and slightly offset the origin of the ray in the x, y, z directions respectively by a small random amount. This added jittering yields a more realistic result without sacrificing sharpness in desired regions.

3.3 Axial chromatic aberration

In addition to basic depth-of-field, we also wanted to simulate a phenomenon known as axial chromatic aberration. So far in our depth of field implementation, we assumed that we are working with a perfect lens system, where light of all wavelengths focus onto one single point.

In real life, light of different wavelength have different indices of refraction. This means that when they pass through the lens, they exit with slightly different angles. This causes different wavelengths of the incoming light to have varying focal points - a phenomenon known as axial chromatic aberration. As shown in Figure 3, the blue component has the shortest focal length, while the red component has the longest.

To simulate axial chromatic aberration, we use the following algorithm. For each image pixel, we compute the DOF-traced color at three separate focal lengths. We then sum the blue component of color of the shortest focal length, the green component of the color of the second shortest focal length, and the red component of the color of the longest focal length.

To get the specific focal lengths for our computation, we first assume that we are working with a single lens, so that the focal length and the index of refraction has a straightforward inverse relationship. We find the different indices of refraction that red, green, and blue light have with glass, which are as follows:

- Red: 1.50917
- Green: 1.51534
- Blue: 1.51690

We let the focal length of the green ray be the user-specified focal length, and then we calculate

$$f_{color} = f_{green} * \frac{i_{green}}{i_{color}}$$

where f_{color} is the focal length of a specific color and i_{color} is the index of refraction of the color.

Later, we find that because the difference in indices is so small, only very far away objects will have obvious chromatic aberration effects. To exaggerate these effects, for some of our results, we amplified the difference in indices of refraction between the colors.

4 Results

4.1 Depth of field

Figure 4 shows basic depth-of-field renderings of different images for various focal lengths. This accomplishes the desired task of focusing in on parts of the scene and blurring out others. As mentioned before, due to the lack of jittering, there are undesirable grids that form in the blurry background (most noticeable for bunny results), due to our equally spaced sampling points.

4.2 Depth of field with jittering

Figure 5 shows a comparison between unjittered and jittered images of the bunny. This jittering removes the undesirable grid in the background, yielding a more realistic photograph.

4.3 Axial chromatic aberration

The effects of chromatic aberration only show up on very far away objects, which is hard to simulate because we have to make sure the camera can still see the object when we are very far away. We can see some of the effects near the edges in Figure 6. After we exaggerated the difference in indices of refraction, we see a much more obvious effect, which is shown in Figure 7.

5 Conclusion

We have simulated depth of field by extending upon the basic ray tracer algorithm. We started off with the basic approach as outlined in [1], which is an "averaging approach" that involved computing focal points and averaging the results of depth-of-field rays shot through square brackets of a certain aperture. This approach worked well but created undesirable grids in the background, which we resolved by jittering our depth of field rays. Finally to take into account lens systems in real life, we simulated the effects of axial chromatic aberration, where the different wavelengths of the incoming light result in different focal points.

There are many more lens effects we could have simulated, such as transverse chromatic aberration, spherical aberration, and radial distortion. These are different effects that lenses have on light rays, which all seem fun to implement. A larger project could be to extend our ray tracer to take as input a 3D model of a lens and simulate how the specific geometry of the lens would affect the photo taken.

6 References

[1] Lee, Skeel (2007, March). CG:Skeelogy - Depth Of Field Using Raytracing. Retrieved December, 2018, from http://cg.skeelogy.com/depth-of-fieldusing-raytracing/

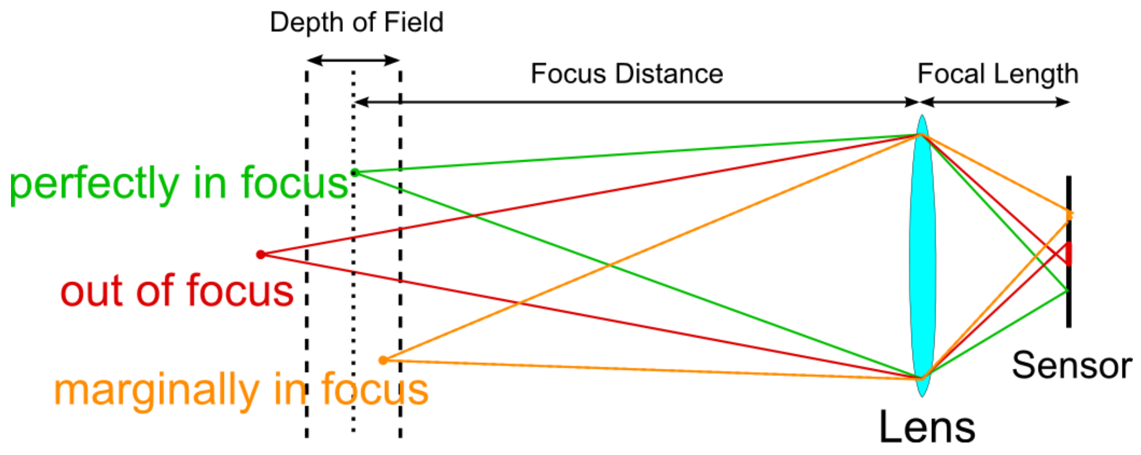


Figure 1: A diagram of the depth of field effect. Because the lens is slightly arced, the light rays originating from the left hit the sensor differently depending on where the point of origin is.

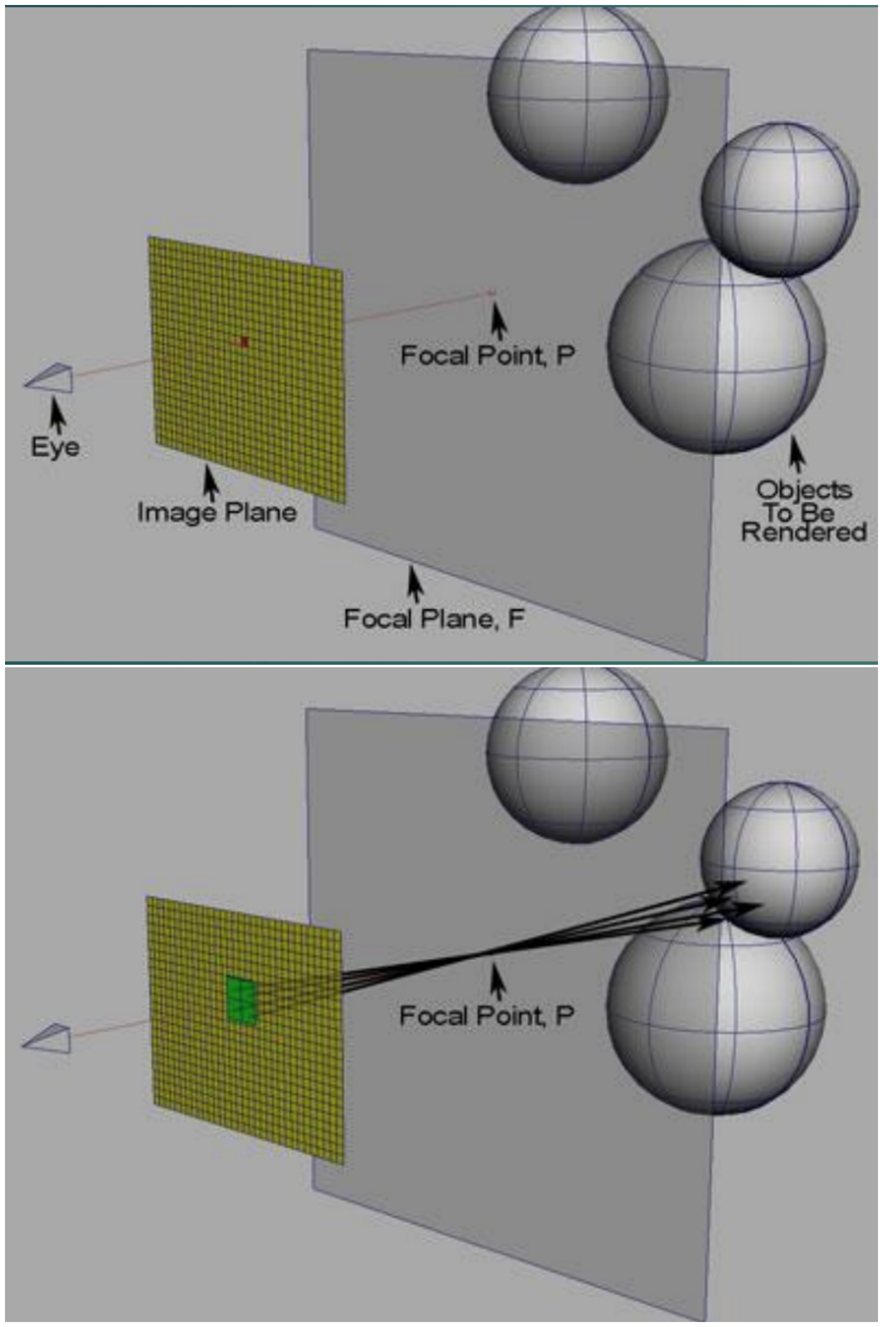


Figure 2: Our depth of field implementation. For each pixel on the image plane, we construct a ray from the eye to the image plane pixel, and we intersect it with the focal plane to get the focal point. We then choose several points surrounding the image plane pixel in a square bracket of a certain aperture, and for each point, we construct depth of field rays through the focal point. The color of the pixel is the average of the hit colors of the depth of field rays [1].

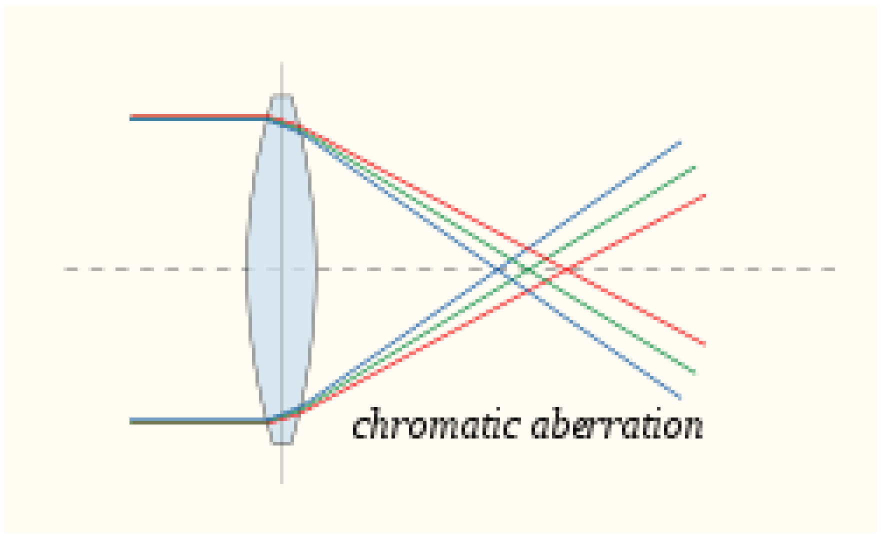


Figure 3: Because of the curvature of the lens, light bends as it exits the lens. Light of different wavelengths have different indices of refraction, so they bend at different angles. This causes the effect that light of different colors have very slightly different focal lengths. As shown, blue has the shortest focal length, while red has the longest.

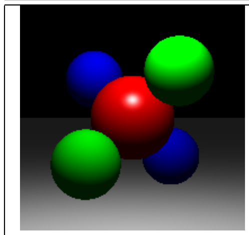
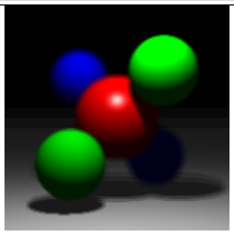
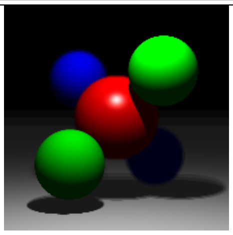
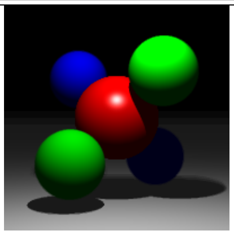
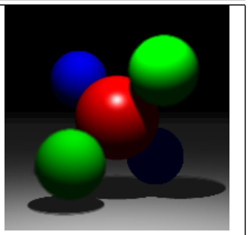
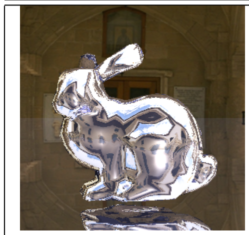
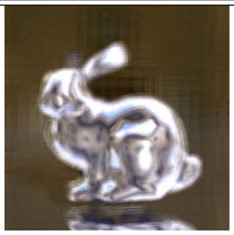
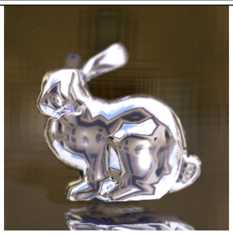
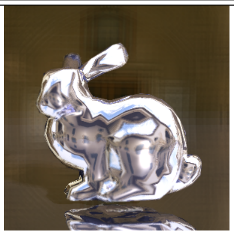
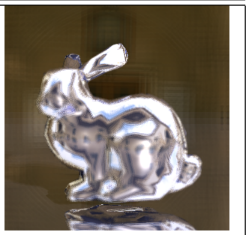
source	4	5	6	7
				
				

Figure 4: Depth of Field Results (Unjittered). The leftmost column denotes the source image, and the other images are the results at various focal lengths. We can see that the focus shifts to farther and farther away as we move right.

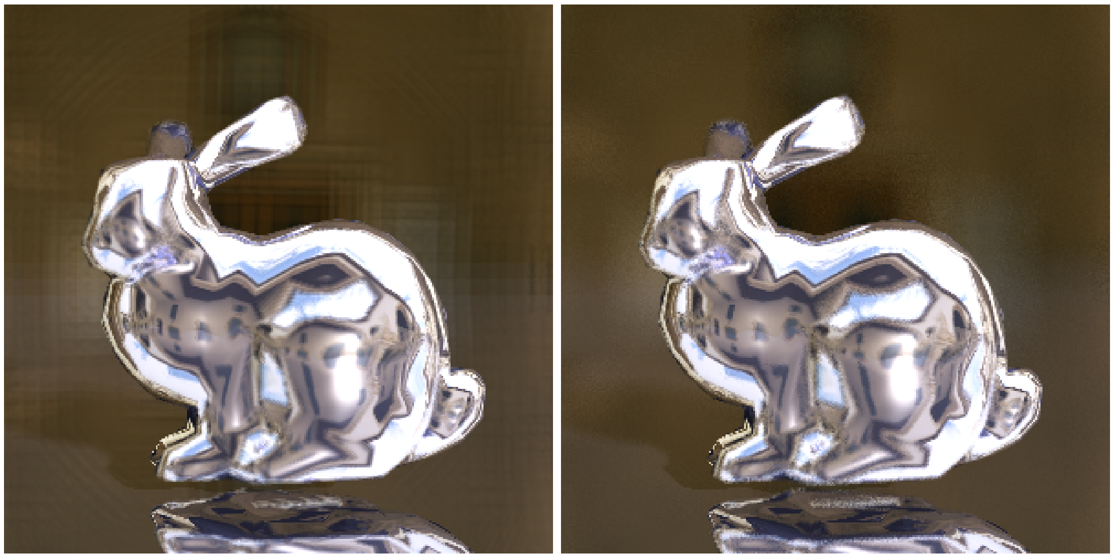


Figure 5: On the left, we have the original unjittered image. We can see in the background that there are undesirable grids that form as a result of our equally spaced sampling points. On the right, we added jittering, which removes the gridded background.

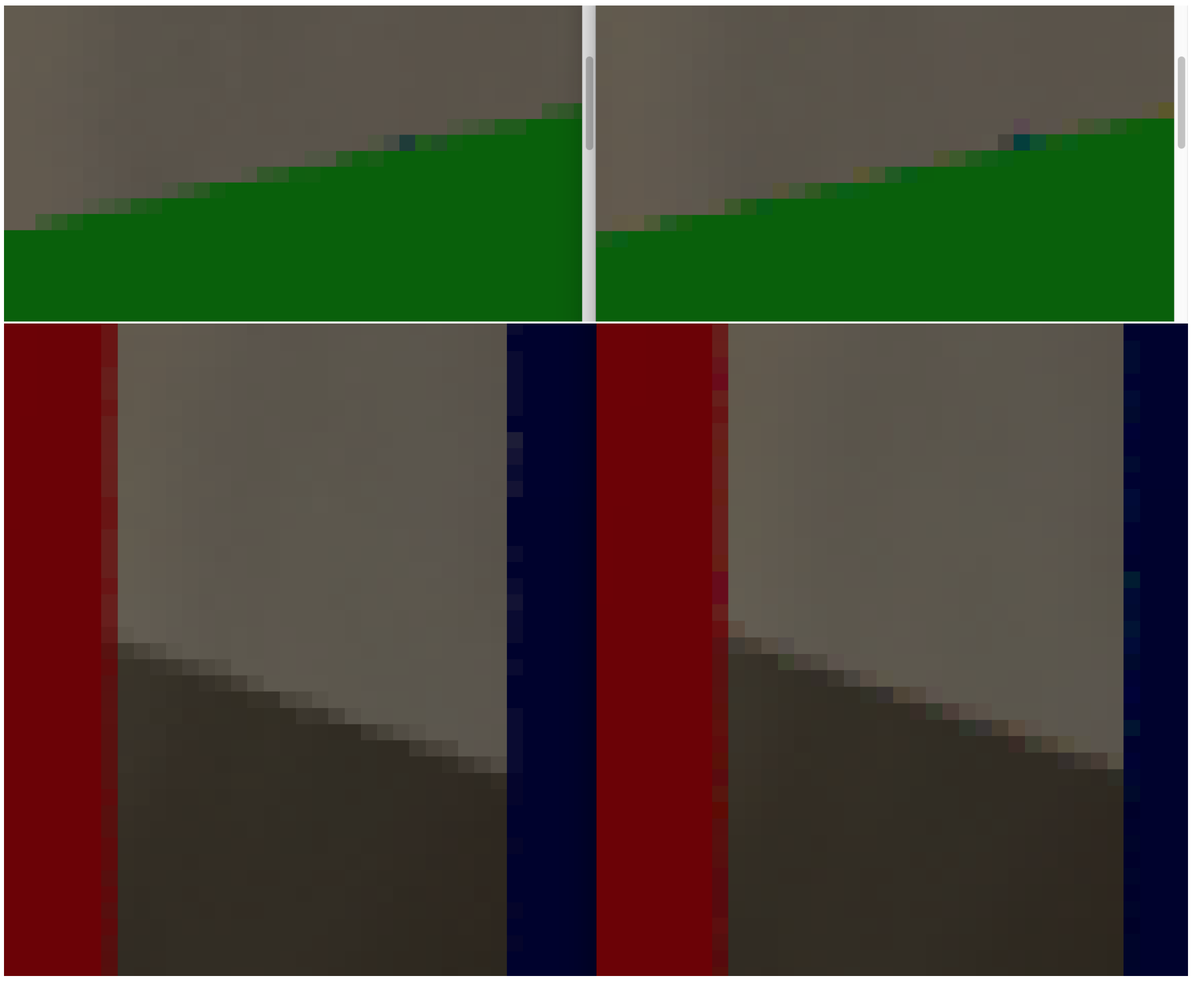


Figure 6: We can see two examples of chromatic aberration near the edges of these photos. On the left are images that do not exhibit the aberration, whereas on the right we see the effect. On the top right, we see examples of yellow dots which should not appear on a green edge. On the bottom right, we see "rainbows" where the shadows intersect with lighted regions. The objects are approximately 40 units away in the simulation.

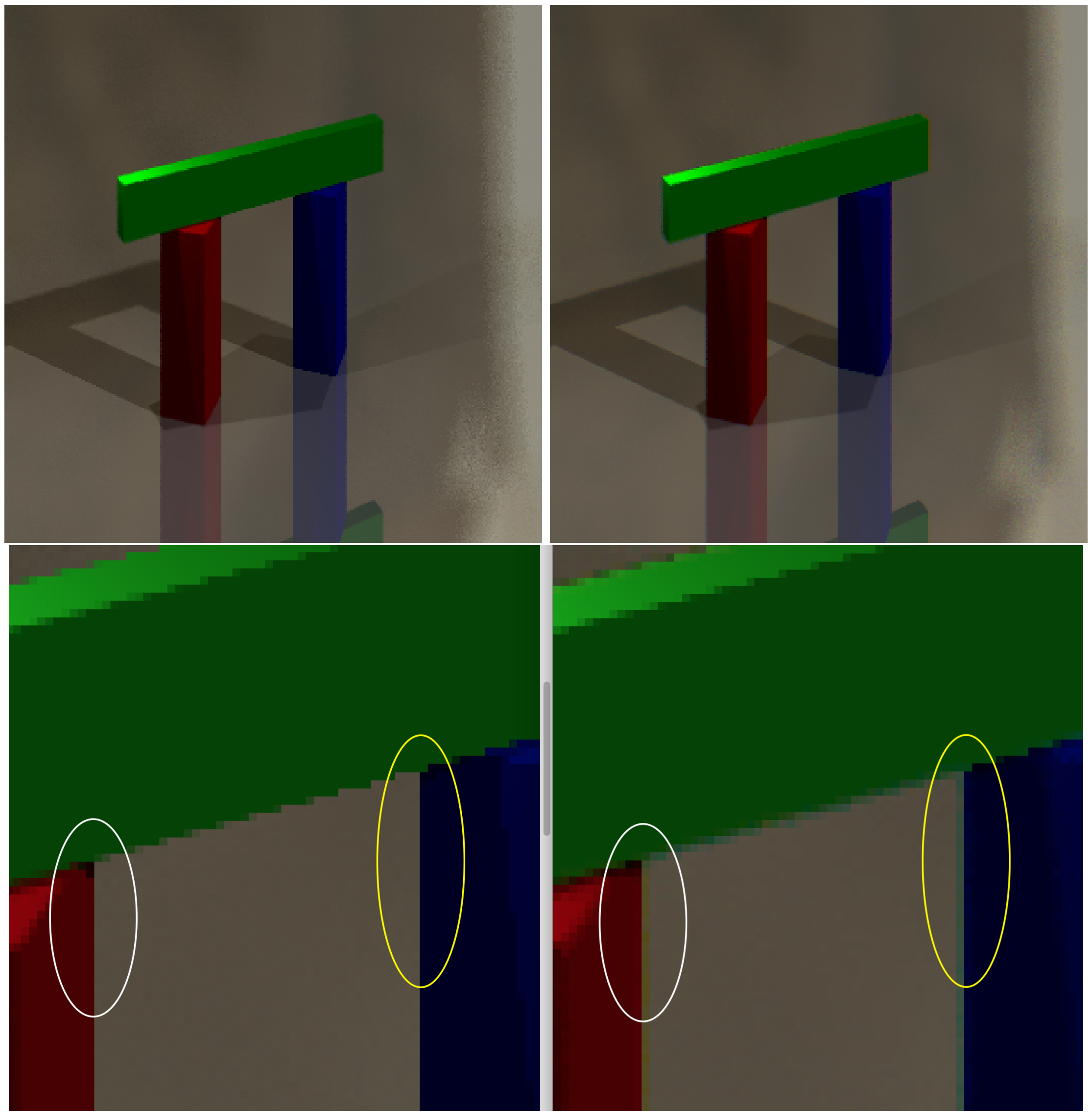


Figure 7: We have a normal DOF image on the top left, and one with chromatic aberration effects on the top right. The bottom images are zoomed-in versions of the top. In this example, we exaggerated the difference in indices of refraction between the different wavelengths of light. We can see effects of chromatic aberration very clearly, near the edges of objects where there is sharp contrast.