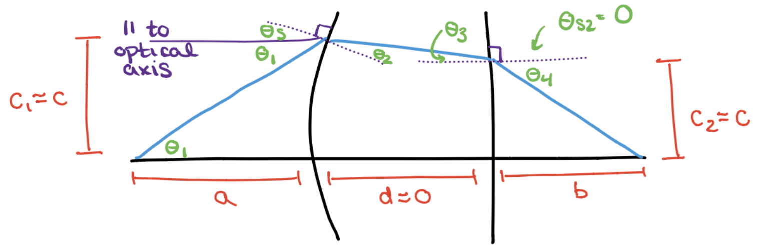
6.869 Pset 2

Nada Hussein

March 2021

Problem 1

a) We can draw a new system for this lens, as shown below:



Most of this scenario does not change from the old one. Our first equation involving θ_1 stays the same, and we can use the small angle approximation to say that $\theta_1 = c/a$. Second, our equation for θ_2 also stays the same: $n\theta_2 = \theta_1 + \theta_S$, as this does not yet concern the back of the lens. However, the θ_3 equation does change. In the initial case, we had $2\theta_S = \theta_2 + \theta_3$. However, we cannot use $2\theta_S$ because the angles of the curves are no longer the same. Since the back of the lens is flat, we know that its angle of curvature is 0. Therefore, $2\theta_S$ becomes θ_S, and we end up with $\theta_S = \theta_2 + \theta_3$. The $\theta_S + \theta_4$ also changes – since θ_S is 0 on this side of the lens, we end up with simply $n\theta_3 = \theta_4$. Finally, the θ_4 equation does not change, and we are left with $\theta_4 = c/b$. Below is a table of all of these equations:

Angle	Description	Relation	Reason
θ_1	initial angle from optical axis	$\theta_1 = c/a$	small angle approx.
θ_2	angle of refracted ray wrt front surface normal	n$\theta_2 = \theta_1 + \theta_S$	Snell's law, small angle approx.
θ_4	angle of refracted ray wrt back surface normal	$\theta_S = \theta_2 + \theta_3$	symmetry of lens, thin lens approx.
$\theta_S(0) + \theta_4$	angle of ray exiting lens wrt back surface normal	n$\theta_3 = \theta_4$	Snell's law, small angle approx.
θ_4	final angle from optical axis	$\theta_4 = c/b$	small angle approx.

b) We can cascade the constraints we came up with in part a to find a relationship for θ_S as follows:

$$\theta_1 = \frac{c}{a}$$

$$n\theta_2 = \theta_1 + \theta_S => \theta_2 = \frac{1}{n}(\frac{c}{a} + \theta_S)$$

$$\theta_S = \theta_2 + \theta_3 => \theta_S = \frac{1}{n}(\frac{c}{a} + \theta_S) + \theta_3 => \theta_3 = \theta_S(1 - \frac{1}{n}) - \frac{c}{an}$$

$$n\theta_3 = \theta_4 => \theta_4 = n[\theta_S(1 - \frac{1}{n}) - \frac{c}{an}] => \theta_4 = \theta_S(n-1) - \frac{c}{a}$$

$$\theta_4 = \frac{c}{b} => \theta_S(n-1) - \frac{c}{a} = \frac{c}{b}$$

$$\theta_S = \frac{c}{n-1}(\frac{1}{a} + \frac{1}{b})$$

Once we have this, we also know that $\theta_S = \frac{c}{R}$. Plugging this in, we get:

$$\frac{c}{R} = \frac{c}{n-1}(\frac{1}{a} + \frac{1}{b})$$

$$\frac{1}{R} = \frac{1}{n-1}(\frac{1}{a} + \frac{1}{b})$$

We also know that through the Lensmakers' equation, $\frac{1}{a} + \frac{1}{b} = \frac{1}{f}$. Plugging this in, we end up with:

$$\frac{1}{R} = \frac{1}{n-1}\frac{1}{f}$$

$$f = \frac{R}{n-1}$$

which gives us our focal length f.

Problem 2

a) We can first solve for Z_c by way of comparing similar triangles. Our larger triangle has height Z_c and width d. The smaller triangle can be created with along the edge of the two sensors' triangles, creating a triangle with height $Z_c - f$ and width $d + u_p - u_c$. Using these to create a ratio equation, we get:

$$\frac{d}{Z_c} = \frac{d + u_p - u_c}{Z_c - f}$$

This leads us to a final answer:

$$Z_c = \frac{fd}{u_c - u_p}$$

From here, we have equations relating x to X and y to Y:

$$u_c = fX_c/Z_c => X_c = \frac{u_c Z_c}{f} => X_c = \frac{u_c d}{u_c - u_p}$$

$$v_c = fY_c/Z_c => Y_c = \frac{v_c Z_c}{f} => Y_c = \frac{v_c d}{u_c - u_p}$$

b) To create T1, we were relating $(u_p, v_p, 1/Z_p)$ to (X_p, Y_p, Z_p). Using the equations from above, I knew that $X_p = \frac{u_c Z_p}{f}$ and $Y_p = \frac{v_c Z_p}{f}$. Thus, The matrix we needed was:

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & f \\ 0 & 0 & f & 0 \end{bmatrix}$$

If we plug this in, we get

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & f \\ 0 & 0 & f & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1/Z_p \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ f \\ f/Z_p \end{bmatrix} = \begin{bmatrix} uZ_p/f \\ vZ_p/f \\ Z_p \\ 1 \end{bmatrix}$$

To create T2, we needed to convert (X_p, Y_p, Z_p) to (X_c, Y_c, Z_c) This was a simple matrix – we know that the only difference between the two camera frames is a translation by d in the X direction. Thus:

$$T_2 = \begin{bmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

This leads to

$$\begin{bmatrix}1 & 0 & 0 & d\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}X_p\\Y_p\\Z_p\\1\end{bmatrix}=\begin{bmatrix}X_p+d\\Y_p\\Z_p\\1\end{bmatrix}=\begin{bmatrix}X_c\\Y_c\\Z_c\\1\end{bmatrix}$$

Finally, T3 was a conversion from (X_c, Y_c, Z_c) to $(u_c, v_c, 1)$ This is the reverse operation from T1. So we get the following:

$$T_3=\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1/f & 0\end{bmatrix}$$

This leads to

$$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1/f & 0\end{bmatrix}\begin{bmatrix}X_c\\Y_c\\Z_c\\1\end{bmatrix}=\begin{bmatrix}X_c\\Y_c\\Z_p/f\end{bmatrix}=\begin{bmatrix}X_cf/Z_p\\Y_cf/Z_p\\1\end{bmatrix}$$

This aligns with our equations relating the world to the image.

Finally, to get T, we must multiply all of these matrices together in reverse order to maintain the multiplication order. Thus, we get:

$$T=T_3T_2T_1=\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1/f & 0\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & d\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & f\\0 & 0 & f & 0\end{bmatrix}=\begin{bmatrix}1 & 0 & df & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\end{bmatrix}$$

c) When we change the scan path to left to right, the laser path stops wiggling.Given the T we got from part b, we can write the mapping from one camera image to the other as:

$$\begin{bmatrix}1 & 0 & df & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}u_p\\v_p\\1/Z_p\\1\end{bmatrix}=\begin{bmatrix}u_c\\v_c\\1\end{bmatrix}=\begin{bmatrix}u_p+df/Z_p\\v_p\\1\end{bmatrix}$$

This shows that the v_c maps directly v_p, whereas u_c maps to u_p+df/Z_p, meaning that the u_c is dependent on the depth of the real world, Z_p. This means that the translation of u_p to u_c is variable depending on the real world depth of that point. Meanwhile, there is no translation of v_c to v_p, which is why when we go from left to right the laser path stays straight, but when we go from top to bottom the laser wiggles due to the dependency of u_c on the depth of the point the last is focused on.

d) I used my equation from part a to compute Z_c:

$$Z_c = \frac{fd}{u_c - u_p}$$

Using this in my function, I got the following result:

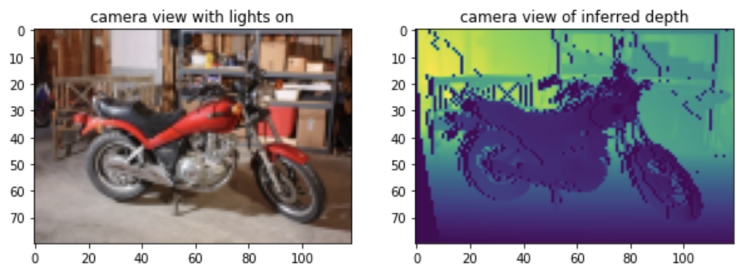


Figure 1: Inferred Depth Result

This did a good job of inferring the depth of the motorcycle and its surroundings. However, it does come out looking much more pixelated due to the resolution of the projector. This is why we end up seeing a more grainy version of the motorcycle in the depth map. Furthermore, we also are unable to use this to infer depth for occlusions. This is why we also see some darker areas in the background – these are occluded spaces that our function was unable to infer depth for.

e) After illuminating the scene with the stripe pattern, the result looks like the following:

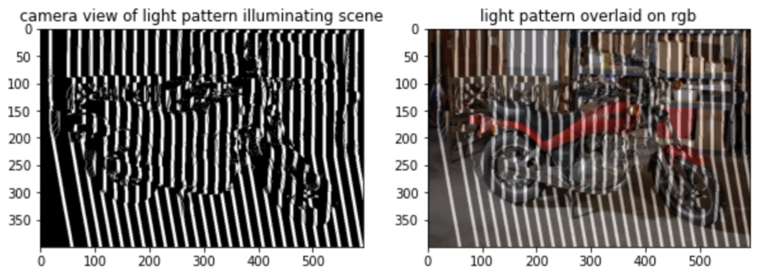


Figure 2: Stripe Pattern Illumination Result

For an alternative pattern of light, I decided to project a checkerboard pattern onto the image. This ended up looking like the following:

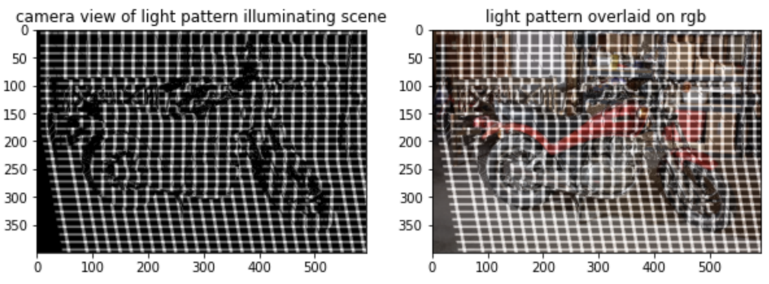


Figure 3: Checkerboard Pattern Illumination Result

For this pattern, we ended up with some occlusions. We can tell that around the outline of the motorcycle, we are unable to see what is behind it, meaning that we have occlusions around the motorcycle. We can especially see this next to the front wheel.

f) For my initial strategy, we know that only x changes between the two images. Thus, our light pattern only needs to vary over x as this is all we need to write the mapping between the images. Thus, initially I created an illumination where the leftmost column of pixels had an illumination of 0, and every column after that had an illumination incremented by 1 from the last column. To write F from this, I then just mapped the y points to the old y points, and the x points were mapped to the result Lc, as this new illumination directly mapped to the new x's since it reflected where the illuminations moved to from the original.

Below is the result of this implementation:

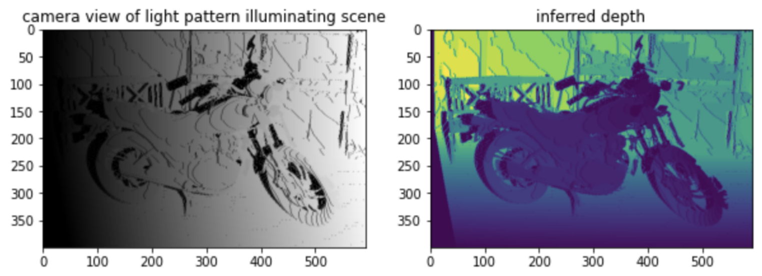


Figure 4: Incremental Pattern Depth Map

A more complicated implementation that would work is to increment every pixel of illumination – i.e. incrementing over the row, and then wrapping around to the next column such that the top left corner is 0, and the bottom right corner is the highest value. Based on this, we can use the dimensions of the image to look at Lc and find the mapping original pixel based on the value at Lc (this value will map to the old coordinate). From here, we can assign x by assigning x_p based on where it used to be given the value of Lc at the (x_c, y_c) point.

g) If we use the same illumination pattern and F as the last part, we do not end up with a correct inferred depth. This is due to the fact that the Lambertian rendering changes the values of xy_p due to the reflections. Thus, since we are not preserving the values we are no longer able to construct the depth map as easily since simply mapping the same value won't work anymore.

To change this, we now need a way to normalize the xy illumations. I did this by setting one of the RGB channels to the same illumination as before, but I then made a second channel all 1s. This was a normalizing factor such that the ratio for each pixel would be preserved. Thus, I then created F by way of mapping the ratio of the incremental filter over the one that previously used to be all 1s. Because the Lambertian reflectance affects each channel the same way, this led to the ratios being preserved, so that I could use a similar mapping as used in part f, on the ratios instead of just a single value.

Below is the result of this implementation:

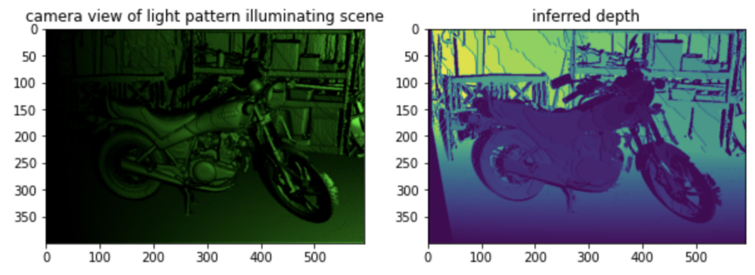


Figure 5: Incremental RGB Pattern Depth Map