Computational Photography 6

Brandon Hung

December 11, 2020

I want to preface this by saying I am sorry for doing poorly on this assignment. My time was swamped for the last two weeks between this, two final projects, grad school applications, and especially difficult weekly assignments from other courses. While I know that doesn't excuse my poor performance, I just wanted to let you know that I did my best with the limited time I have — but I cannot afford to work on this any longer, as I have a paper to write and the final project to wrap up. I will try to explain my likely sources of issue in the write-up.

1 Implementing structured-light triangulation

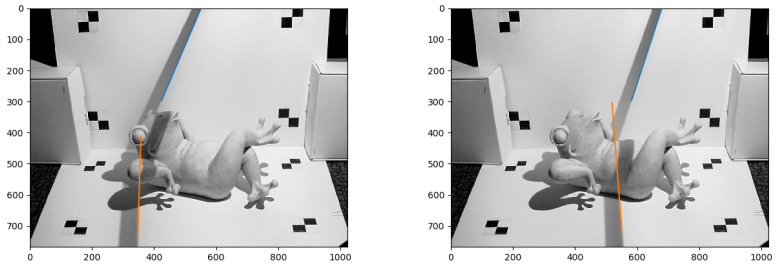
1.1 Video Processing

Spatial shadow edge localization

I used frames 64 to 127 for the shadows, and tuned the cropped area to look for shadows. There were quite a few issues with noise, where random pixels in completely dark regions would have a zero crossing in the $\Delta I(x, y, t)$, so I attempted to do some Gaussian blurring to reduce those zero crossings. While that mostly got rid of the problem, I had the issue of zero-crossings being detected in the shadow of the frog. This initially caused outliers to skew the lines; after cropping them out, I had one more problem. At frame 85, the horizontal plane line I cropped is nearly vertical. This caused the line fit to the points to split the line in half instead of vertically interpolate them. I tried to expand the bounding box I used to crop the points to account for this, but I couldn't get anything reliable since the frog was so close by.

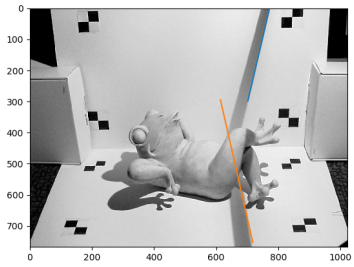
Here are the mostly good images I found of the spatial shadow lines, along with the bad one at frame 85:

Figure 1: Spatial shadow lines overlaid on image

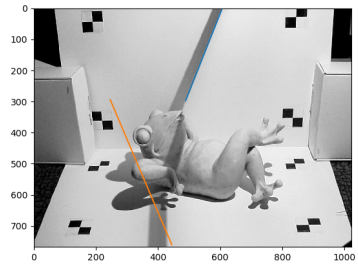


(a)

(b)



(c)



(d) Poorly fit line

The method I used to find zero-crossings was to take the difference image, find the signs of each pixel, find the first derivative of the image by convolving it with $[-1, 1]$, crop a vertical and a horizontal bounding box from the derivative, and find the pixels > 0 in each bounding box. Pixels > 0 corresponded to the right side of the shadow, so I used that as the shadow line.

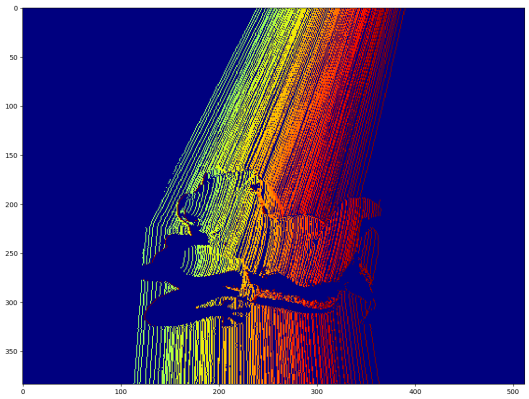
I believe this method might have contributed to some errors. Pixels that are similar to the shadow, but just slightly lighter than it, would show up as positive and had a potential to get counted. I mostly eliminated by removing pixels with a low contrast, but it still shows up (as will be apparent in the temporal shadow lines). I wasn't really sure how to do it differently, unfortunately, so I had to stick with it.

Temporal shadow edge localization

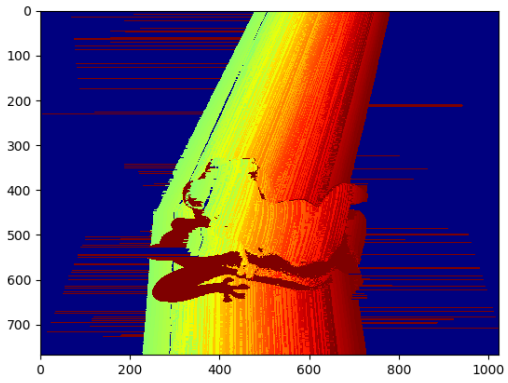
I approached the temporal shadow edge localization assigning pixels that had a zero-crossing the time associated with that frame. Since I started at frame 64, I bumped everything up to start from 64. Unfortunately, the same problem occurred where noisy pixel patches that went above and below the shadow

intensity would register as shadow crossings. While I mostly got rid of them, you see some (particularly around the frog's shadow) in the overlaid image. In addition, there are some hanging around in the image that strongly affect the interpolated image.

Figure 2: Temporal shadow edges



(a) Non-interpolated temporal shadows



(b) Interpolated temporal shadows

I interpolated the shadows by looking at each column in a row, checking if its value (representing time) was not zero, and interpolating that value of t until I hit the next non-zero column in that row. Unfortunately, you can see that this method caused the noisy pixels having a much larger influence on the

image through the streaks along the image. I'm honestly not sure how or if this really affected my reconstruction results, given the reconstructions I used them for didn't differ by a lot. Finally, it should be noted that I only went from 64 to 127 without discretizing them, resulting in the short range of colors.

1.2 Intrinsic and Extrinsic Calibration

Calibration of ground planes

$$t_h = \begin{bmatrix} -321.21 \\ 207.28 \\ 1560.15 \end{bmatrix}, R_h = \begin{bmatrix} 0.99 & 0.056 & -0.017 \\ -0.012 & -0.46 & -0.89 \\ -0.127 & 0.89 & -0.46 \end{bmatrix}$$

$$t_v = \begin{bmatrix} -320.35 \\ -62.44 \\ 1877.40 \end{bmatrix}, R_v = \begin{bmatrix} 1.0 & -0.026 & -0.017 \\ -0.014 & -0.88 & 0.48 \\ -0.028 & -0.48 & -0.88 \end{bmatrix}$$

(rounded to two sig figs).

Calibration of shadow lines

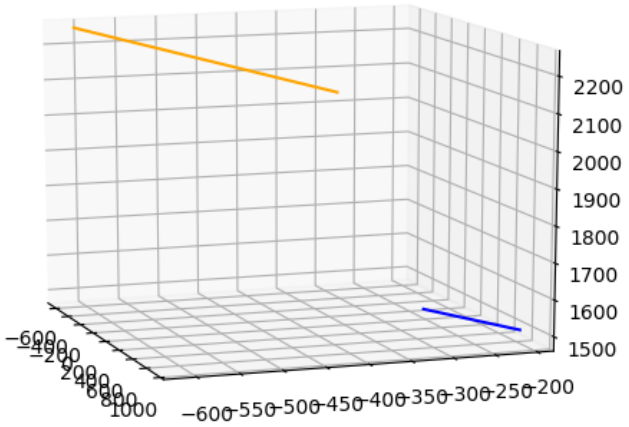
The process I used to calibrate the shadow lines was as follows:

1. Find two y values on the horizontal shadow line and the vertical shadow line. For vertical, I used 0 and the split between the scene planes. For horizontal, I used 768 (last row) and the split between scenes.
2. Use the shadow equations in that frame to find the corresponding x values.
3. Compute the ray going through each (x, y) point.
4. Transform the 3D points $p_{cam} = [x, y, 0]$ and corresponding ray into their counterparts in the horizontal and vertical planes. We do this by setting $p = R^T(p_{cam} - T)$, where R, T are the rotation and translation matrices of the either the horizontal or vertical frame.
5. Our new points are of the form $p = [x, y, z]$ with respect to their coordinate frame (vertical or horizontal plane). To find where the ray intersects our coordinate frame, we solve for $t = -\frac{z}{r_z}$.
6. Using $P = t * r + p$, we solve for P. This gives us a pixel $P_{plane} = [x, y, 0]$.
7. Finally, we convert P_{plane} back into camera coordinates by $R(P_{plane}) + T$, since rotation matrices are orthogonal so $RR^T = I$.

Now, the issue is that the lines don't look like they are in the right place. I don't understand why, but the lines appear to be incorrectly placed in space and don't lie on the right planes. I tried flipping the x and y coordinates, I tried transforming the points and rays using R instead of R^T, I tried translating the rays instead of leaving them untranslated, but none of that seemed to position

the lines correctly. I think it might be a combination of things, mainly incorrect coordinate transformations along with the wrong order of points that really make me wish we were working in homogeneous coordinates instead. As a result, we got strange lines like the following:

Figure 3: Shadow lines in 3D space

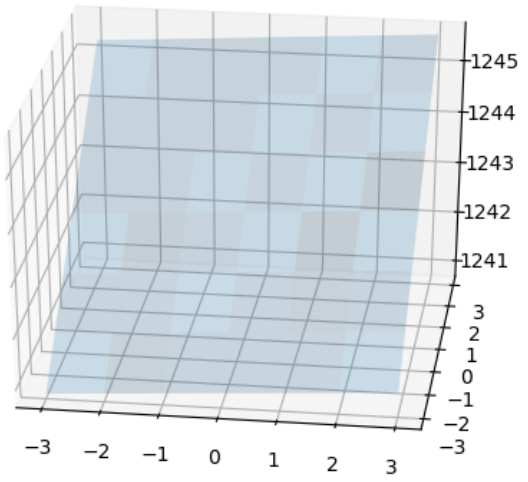


(a) The orange should be the vertical plane line, and the blue should be the horizontal line. But they don't line up with their planes.

Calibration of shadow planes

Following the process outlined in the write-up, I found $P_1, \hat{n}$ in the camera coordinate frame. Due to the weirdness of the lines, the shadow planes were likely kind of weird as well:

Figure 4: Shadow plane in 3D space



(a) The plane appears to be rotated 90 degrees about the x-axis of the camera frame, which is bizarre.

1.3 Reconstruction

Given the relation $(P - P_1) \cdot \hat{n} = 0$, which describes the shadow plane, and $p + r * t = P$ for some t, we want to find where $(p + rt - P_1) \cdot \hat{n} = 0$. This expands to

$$n_x(r_x t + p_x - P_{1,x}) + n_y(r_y t + p_y - P_{1,y}) + n_z(r_z t + p_z - P_{1,z}) = 0$$

which in turn simplifies to

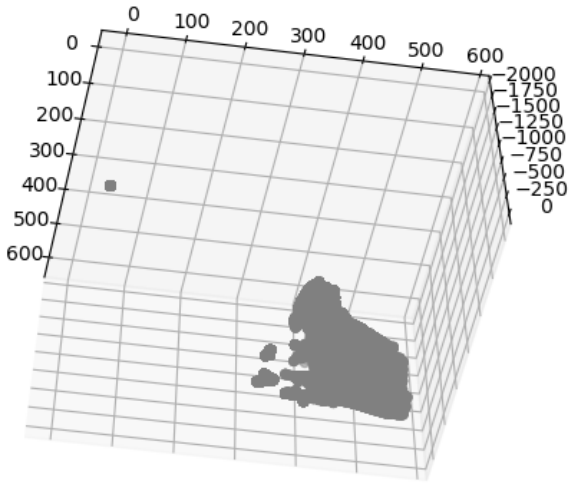
$$t = \frac{(p - P_1) \cdot \hat{n}}{r \cdot \hat{n}}$$

Solving for P, we find $P = p + t * r$ — which is the point in our camera frame.

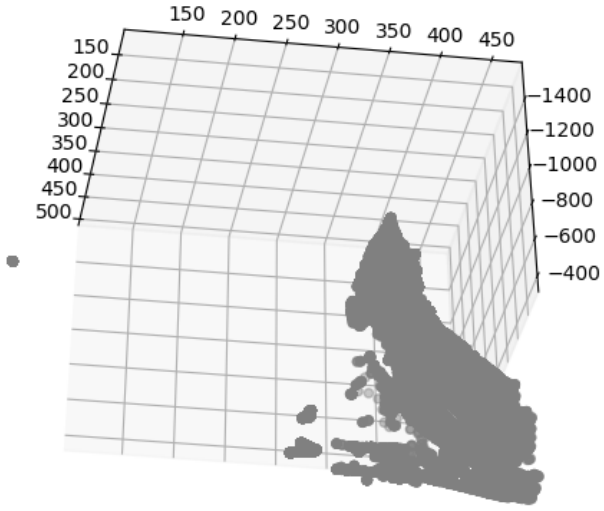
The final output was not great. My belief is that the shadow lines were in the wrong place, causing the planes and reconstruction to be in the wrong place as well. This caused a massive amount of distortion, which resulted in a pretty ugly frog. Some parts are distinguishable, but overall the dimensions seem really off. There were several outliers as well that contributed to making the reconstruction worse. I wish I had the time and energy to go back and redo

the lines, and I wish I knew why they didn't resemble figure 4 in the write-up. It doesn't help that matplotlib is basically impossible to orient properly in 3D, making it difficult to capture images of my reconstruction from the right angle.

Figure 5: Poor reconstruction

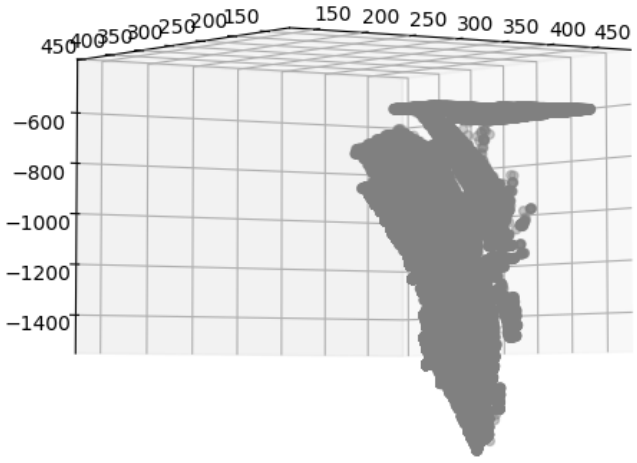


(a) Upper corner view of frog. The eye is slightly visible on the top left, along with the foot on the lower left of the frog.



(b) Same view, but closer. You can see the beginning of the elbow towards the middle-left of the frog, and the eye is a little more distinct

Figure 6: Rotated Reconstruction



(a) The frog is rotated by 90 degrees about the axis going into the page. As you can see, its not very good.

In the previous picture, the thigh of the frog is more visible but that weird horizontal blob is disconcerting. I suspect that's the shin of the frog being projected to a single plane; if so, that indicates that my shadow lines are really off.

2 Build your own scanner

Unfortunately, as I was unable to get the original reconstruction to work, I did not have time to even attempt this. It's especially frustrating since this seems like a very fun and cool project.

Conclusion

I really hate the end the last homework like this, but I really don't know what to do anymore. All I ask is that you take into consideration that I really gave it all I could. I simply have too much work to deal with at once, so I unfortunately need to stop here.