PHYS 410/510 Fall 2024 Homework 5
Bret Brouse Jr. (10/30) (Late submission)

For Problem 3, I ended up breaking my work from HW4 in the process of answering it, so some of my time was rewriting *all* of that! I apologize for turning it in a day late!

Problem 1 (Signal to Noise Ratio in Images).

Solution. I'd rather do this mathematically so here we go:

The definition given of a signal to noise ratio is SRN $= \langle x \rangle / \sigma$ where $\langle x \rangle$ is the mean and σ is the usual standard deviation. Now let's multiple x by some scaling factor A. The standard deviation then becomes

$$\begin{aligned}
\sigma &= \sqrt{\langle (Ax - \langle Ax \rangle)^2 \rangle} \\
&= \sqrt{\langle (Ax - A\langle x \rangle)^2 \rangle} \\
&= \sqrt{\langle A^2 (x - \langle x \rangle)^2 \rangle} \\
&= \sqrt{A^2 \langle (x - \langle x \rangle)^2 \rangle} \\
&= A\sqrt{\langle (x - \langle x \rangle)^2 \rangle} \\
&= A\sigma
\end{aligned}$$

Substituting this back into the definition of the SNR, while noting that we also need to multple the top x by the same factor A we get

$$\begin{aligned}
\text{SNR} &= \frac{\langle Ax \rangle}{\sigma} \\
&= \frac{\langle Ax \rangle}{A\sigma} \\
&= \frac{A\langle x \rangle}{A\sigma} \\
&= \frac{\langle x \rangle}{\sigma}.
\end{aligned}$$

Thus the multiplication by some scale factor A does not matter.

Some brief notes: The mean of some value scaled up is the same as the scale factor times the mean. This follows directly from the definition of a mean (I think this goes without saying but wanted to be clear. If I need to prove this, let me know and I'll annotate.) Otherwise, everything follows from some basic algebra.

Problem 2 (Centroid warmup).

Solution.

```
#########################
# Problem 2a, HW5
n = 1000    # number of positions | 1000
x = collect(0:n-1); | ✓
I = 0.2 .* sqrt.(x) | 1000-element Vector{Float64}:
xc = sum(x .* I) / sum(I) | 599.7058418359127

# Problem 2b, HW5
w = rand.(Poisson(10), n)   # generate Poisson noise with mean 10 | 1000-element Vector{Int64}:
I_noise = 30 ./ (x .+ 20) .+ 0.05 .* w  # Intensity with noise | 1000-element Vector{Float64}:
xc_noise = sum(x .* I_noise) / sum(I_noise) | 446.9049367054943
```

Figure 1: Both centroid calculations from problem 2 where the centroids are located at about 599.7 and 446.9 for parts (i) and (ii) respectively. I should point out that in my code, the expected center is 4.0, which is where we see this histogram peaking. All of this could be shifted by a factor of N/2 to center things. (I added an image of my code and output to the end to show this.)

Problem 3 (Centroid localization).

Solution. (a):

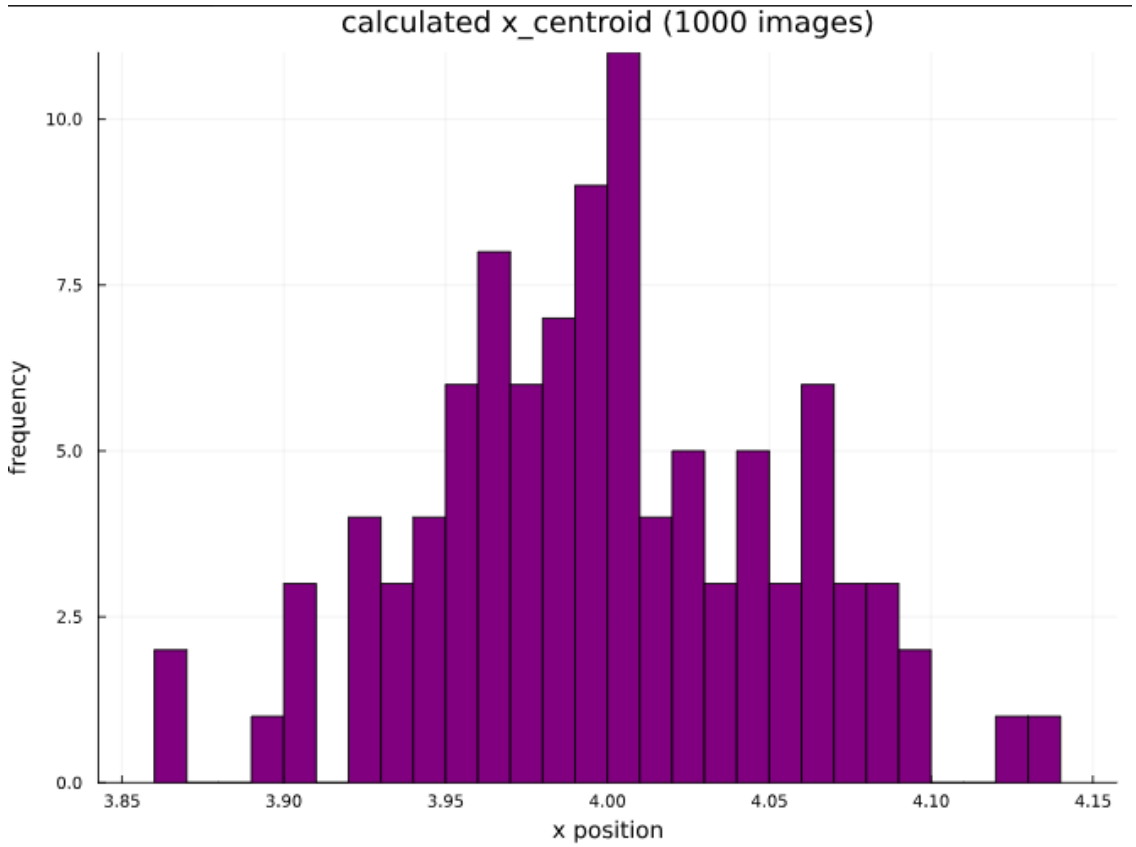


Figure 2: The calculated value of the centroid (in x) using 1000 images. I chose to leave it in pixels rather than real units. The RMS error was found to be about 0.0776 pixels.

(b):

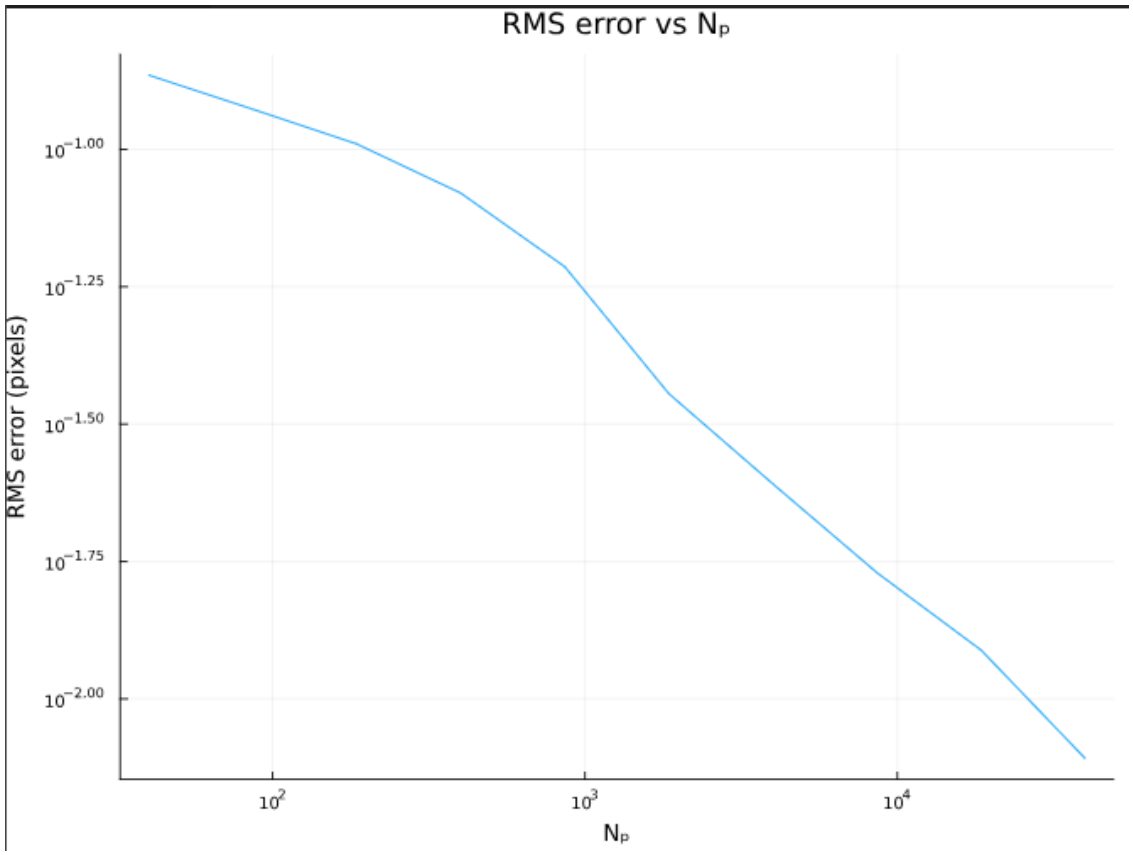
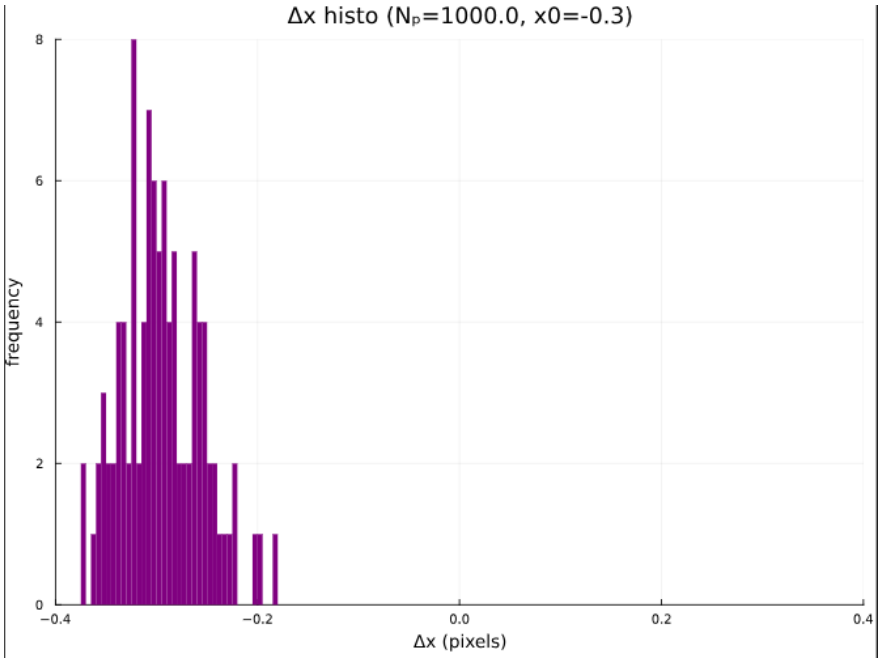
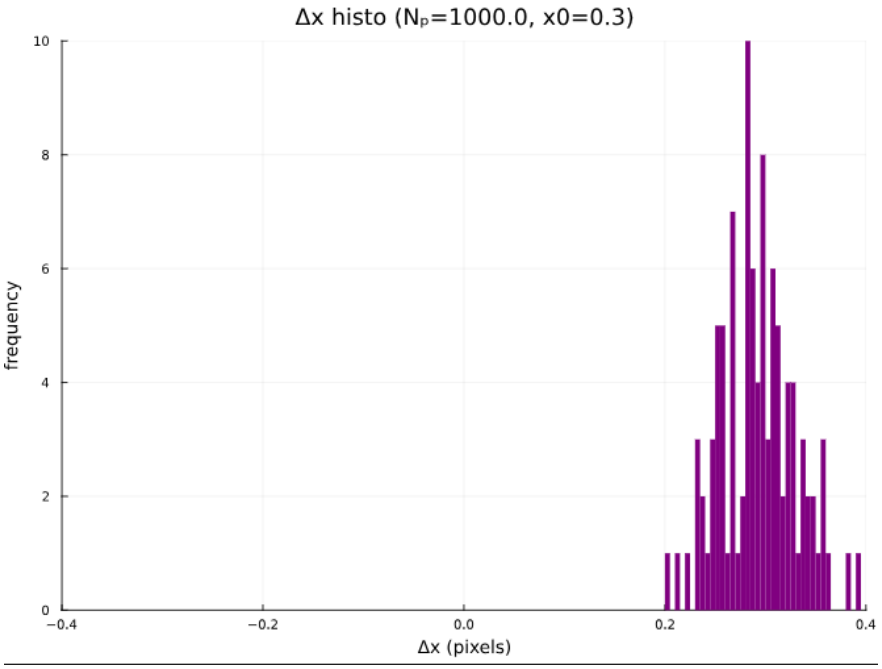


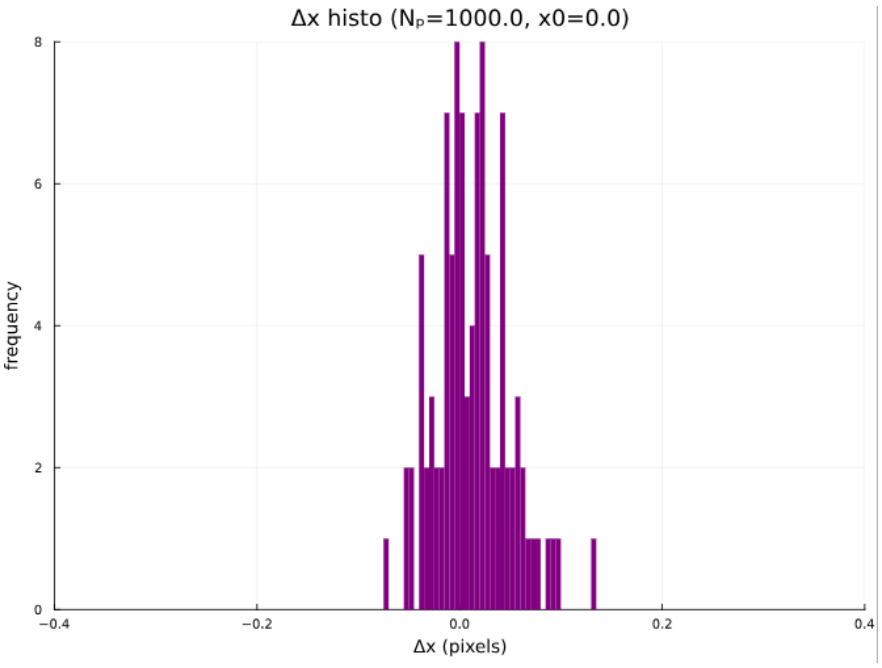
Figure 3: RMS error vs Number of Photons on a log-log scale. This is the expected result as it falls off by a power of 1/2, which we found in previous work to be the case. As N_p increases, we should see the error fall off by $1/\sqrt{N_p}$

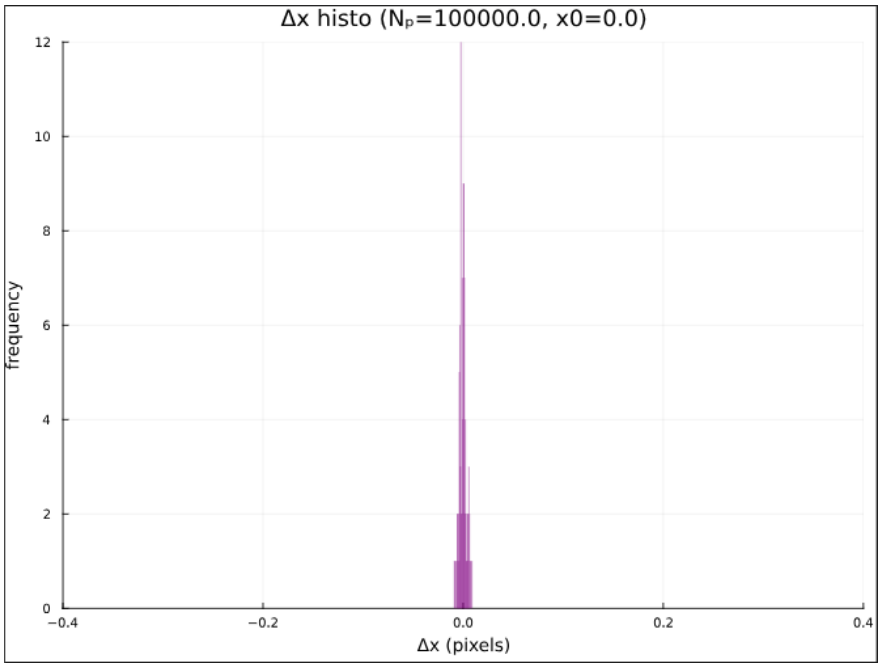
(c):

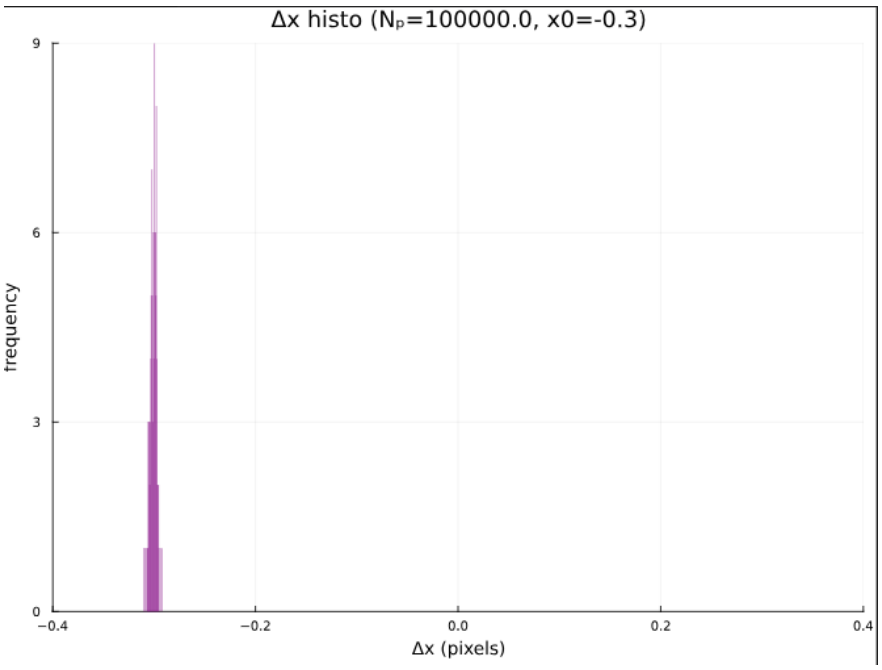


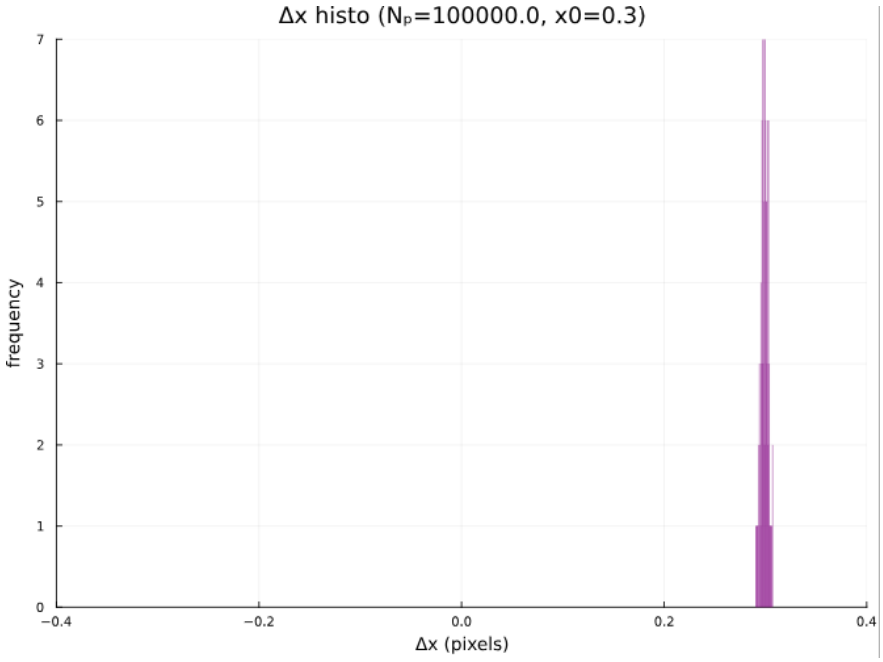
For the above figures, we can see that the centroid is an unbiased estimator since the spread of each of the histograms is roughly the same, regardless of the true value. We can also see that as N_p increases, we get less spread, which means we get closer to values falling on the true value.


Δx histo (Np=1000.0, x0=0.3)
frequency
Δx (pixels)
0
2
4
6
8
10
−0.4
−0.2
0.0
0.2
0.4


Δx histo (Np=1000.0, x0=0.0)
frequency
Δx (pixels)
0
2
4
6
8
−0.4
−0.2
0.0
0.2
0.4


Δx histo (Np=100000.0, x0=0.0)
frequency
Δx (pixels)
0
2
4
6
8
10
12
−0.4
−0.2
0.0
0.2
0.4


Δx histo (Np=100000.0, x0=-0.3)
frequency
Δx (pixels)
0
3
6
9
−0.4
−0.2
0.0
0.2
0.4


Δx histo (N$_p$=100000.0, x0=0.3)
frequency
Δx (pixels)
0
1
2
3
4
5
6
7
−0.4
−0.2
0.0
0.2
0.4

(d):

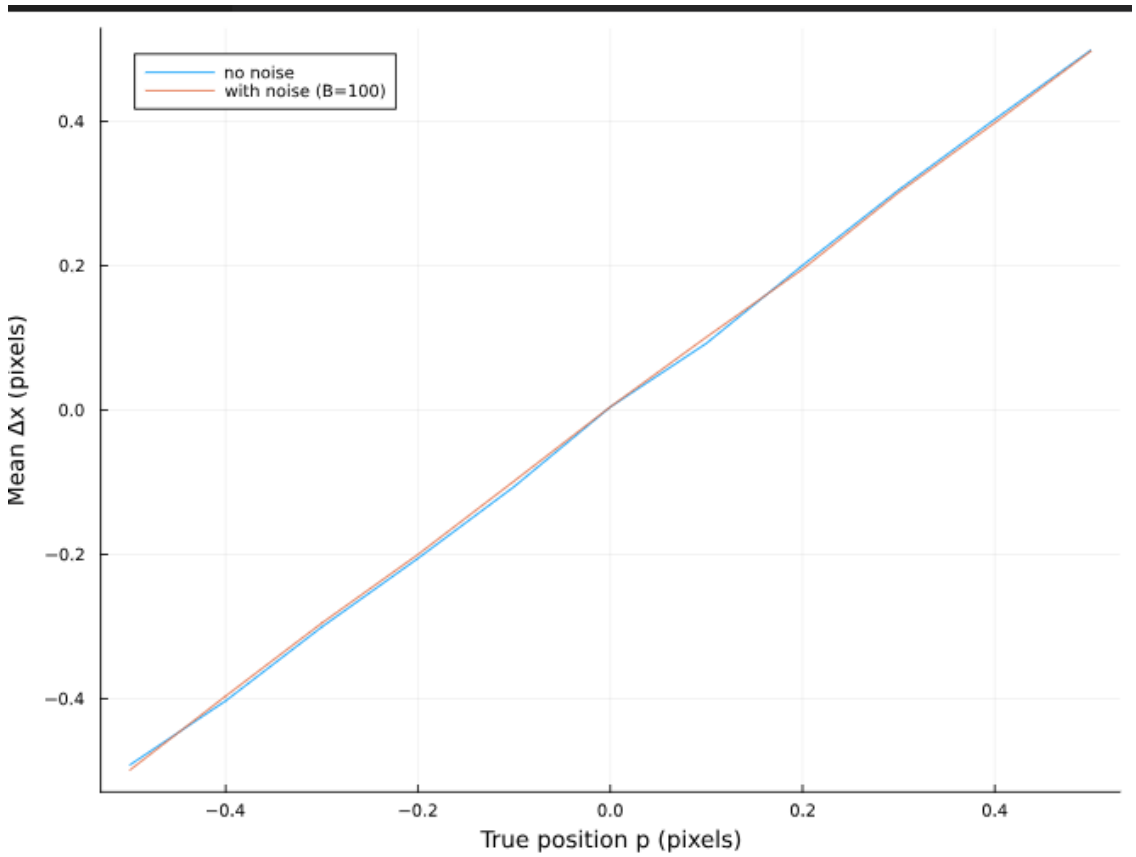


Figure 4: This is the one plot I'm not too sure on. I appears as if they line up pretty well which I would not expect when something is noisy. I suspect something is wrong with how I coded this, unless this is the surprise mentioned earlier in the assignment. Otherwise, I have a hunch there's some linear dependence on p, which was coded into what I wrote anyway so I would suspect linear dependence.

Problem 4 (Reading).

Solution. The first project topic that jumped out at me was segmentation. While I know our projects should not be research related, I read that as, 'You shouldn't be doing research as part of the assignment.' However, the reason I'd like to look into segmentation is to explore it as a means to eventually do lacunarity analysis. Lacunarity, simply put, can be described as the measure of 'gappiness' in an image and variations in those gaps. I looked into a paper called "A novel lacunarity estimation applied to SAR image segmentation" by Gan Du et al. This paper tests different lacunarity methods using SAR. They claim that it's a better method. What drew me the most to this was learning about SAR and how it can affect the image itself vs using other lacunarity methods. Regardless, this immediately corresponds to analyzing portions of an image, in a sense using a kernel of some sort.

The other project topic was image reconstruction, or if possible, image *deconstruction*. Regardless, I like the idea of taking a set of images that are slices and recreating a 3D model. These are useful for dosimetrists, a profession I considered at one point, but I would like to learn more about the process at hand. A paper titled 'Simulation of x-ray-induced acoustic imaging for absolute dosimetry: Accuracy of image reconstruction methods' by Forghani et al investigates multiple techniques to see what end up being more accurate. One thought I had regarding this and the course is performing the error analysis on these techniques. I'm wildy curious about how they determine if it's accurate or not, and at what level does this accuracy become acceptable for actual patients.

Problem 5 (problem 2 appendix image).

Solution.

```
N = 7;                  # camera pixels | ✓
λ = 0.510;              # wavelength in μm | ✓
NA = 0.9;               # numerical aperture | ✓
camera_scale = 0.1;     # camera scale (μm / px) | ✓
fine_scale = 0.01;      # fine grid scale (μm / px) | ✓
Nₚ = 500;               # number of photons | ✓
bg = 10;                # bg noise factor | ✓
xc, yc = 0.0, 0.0;      # offset in μm | ✓
M = 100;                # number of images to generate | ✓

centroids = [] | Any[]

# generate images and calculate centroids
for _ in 1:M
    img = psf(N, λ, NA, camera_scale, fine_scale, Nₚ, xc, yc, bg)
    x_cent, y_cent = calc_centroid(img)
    push!(centroids, (x_cent, y_cent))
end | ✓

# convert centroid list to arrays for RMS calc
x_cent_values = [c[1] for c in centroids] | 100-element Vector{Float64}:
y_cent_values = [c[2] for c in centroids] | 100-element Vector{Float64}:

# calculate RMS error based on center
x0, y0 = (N + 1) / 2, (N + 1) / 2 # Expected center | (4.0, 4.0)
rms_error = sqrt(mean((x_cent_values .- x0).^2 + (y_cent_values .- y0).^2)) | 0
```

Figure 5: Expected center is 4.0, which is what we see in the histogram above.