Joel Cheverie
1002924393
CSC 2503: Assignment 2

A.1.

Figure A.1: Left: The scene in front of my camera obscura, Top: My Setup with a DSLR camera, the light source and camera obscura box, Right: My camera obscura

I followed the steps outlined in this video to make my camera obscura:
https://www.youtube.com/watch?v=Y0wenfVfHuo

It comprises a cardboard exterior which has been taped shut to keep light from entering the obscura from the side. On one end I have a glass lens to let light in, which is incased in a square hole. At the other end it is open to allow for viewing. Inside the carboard tube I have another cardboard tube with a piece of tracing paper at the end. This is the camera plane, where images are projected. I can slide the inner tube farther and closer to the aperture of my camera obscura to focus light as needed. Using aluminum foil I am able to construct different shaped pinholes at the aperture end of my setup. I am using a lightbulb in a lamp, which I turn on in a completely dark room as the scene in my setup.

Figure A.2: Results of the small pinhole aperture. Note that the lightbulb orientation is flipped, it is oriented downwards, not upwards.

A.2.

Figure A3: Larger Disk

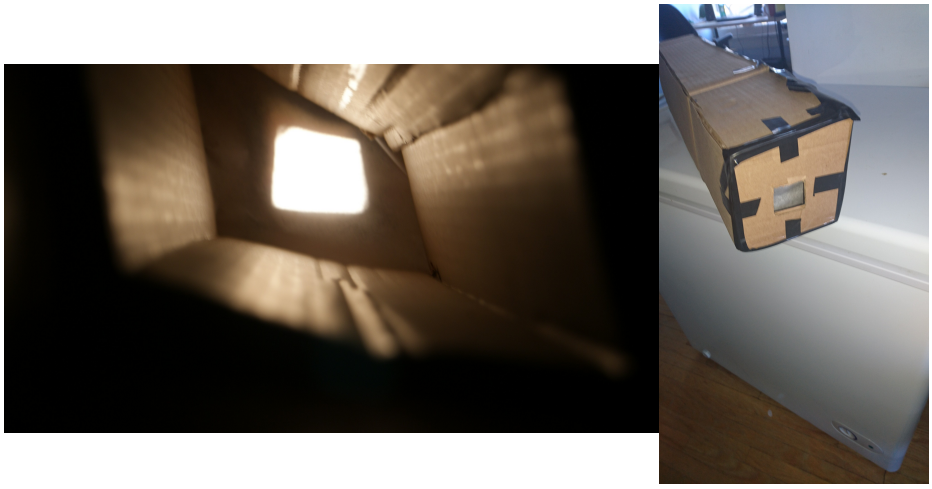


Figure A4: Square Hole (2cm by 2cm)

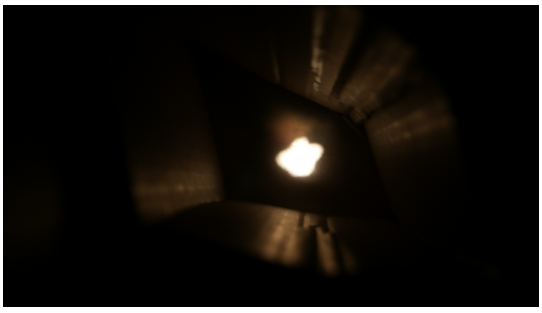


Figure A5: Cross-Shaped Hole (2cm x 2cm)

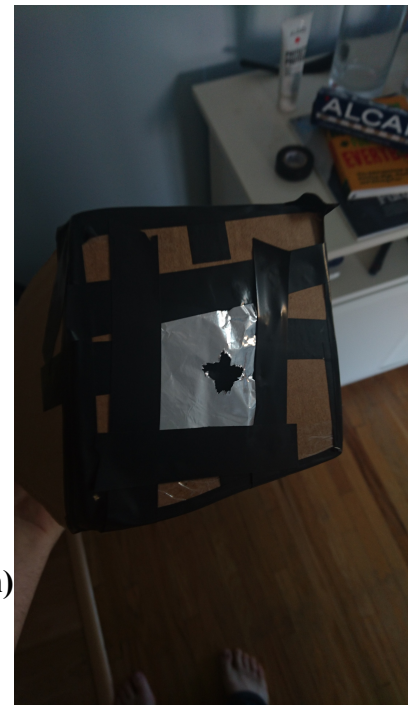


Figure A6: Hexagon Shaped Hole (0.5 cm each side)

Note: My scene did not change at all between changing pinholes. I kept it fixed, so I did not include redundant images of the scene. The camera obscura, lamp and everything else were always placed in the same location in my bathroom.

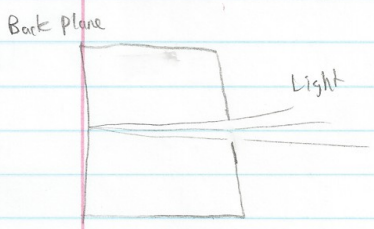
A.3: Initially one may want to approach this problem from a radiometry perspective. Due to foreshortening, the nature of pinhole size and other factors however, this is not feasible.

Instead, I will approach this problem from a perspective of convolution and filtering.

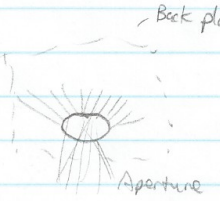
Let $I(x,y)$ be the ideal image. This is a continuous function defined on the sensor plane. The Lambertian assumption, as well as the scene's planarity are crucial here, as without this assumption, light that enters the camera obscura is dependent on the camera's position relative to the scene. The Lambertian assumption of the surface allows the brightness of the scene (and the consequential light rays) to be assumed invariant to camera angle.

The parallelism and planarity of the scene to the camera also simplifies the derivation, as we can then assume light rays are entering the camera directly.

We have the following scenario as a result: (no need to worry about angle or refraction/reflection)



In 3D:



We notice however that as the pinhole expands, the image becomes blurry. Consequently, one can think of the shape S of the pinhole as representing a filter f_s in a convolution with $I(x,y)$. f_s here has values dependent on the shape S.

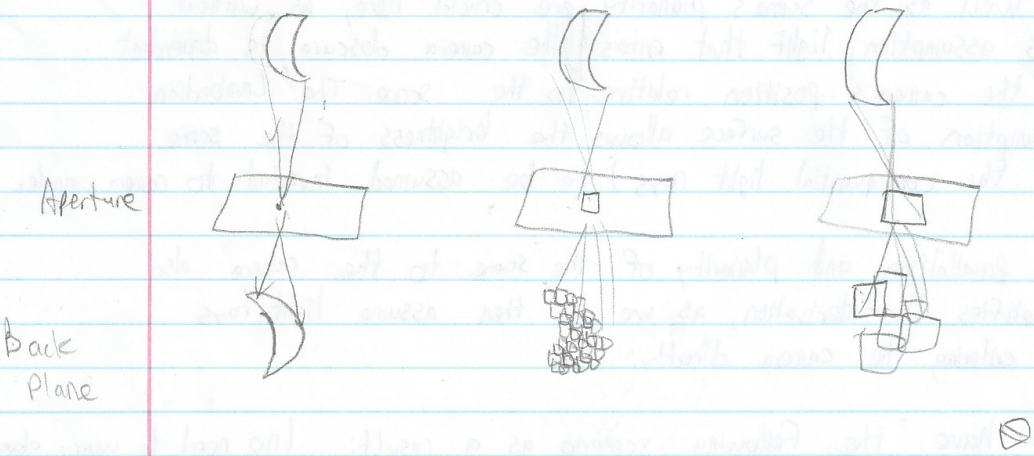
In other words, $I_s(x,y) = \iint f_s(u,v)\, I(x-u, y-v)\, du\, dv$

$= f_s * I(x,y)$

Intuitively, what this means is that $I_S(x,y)$ looks like a few superimposed copies of the shape S that make up a blurry representation of $I(x,y)$.

In my results above, this was difficult to realize, but if you look closely enough, you will see many copies of the pinhole shape S making up the ideal image $I(x,y)$ in a blurry way.

A visual representation:



B.1.

Let $\vec{x}_k = \hat{\vec{x}}_k + m$, then

$$e_k = \|\hat{\vec{x}}_k + m - \vec{x}_c\|^2 - r^2$$
$$= \hat{\vec{x}}_k^T \hat{\vec{x}}_k + 2\hat{\vec{x}}_k \vec{m} - 2\hat{\vec{x}}_k \vec{x}_c - 2\vec{m}\vec{x}_c + \vec{x}_c^T \vec{x}_c + \vec{m}^T \vec{m} - r^2$$
$$= \|\hat{\vec{x}}_k - \vec{x}_c\|^2 + 2\hat{\vec{x}}_k \vec{m} - 2\vec{m}^T \vec{x}_c + \vec{m}^T \vec{m}$$

But as $\hat{\vec{x}}_k$ is the true position, $\|\hat{\vec{x}}_k - \vec{x}_c\|^2 = r^2$, so the above simplifies to:

$$= r^2 + 2\hat{\vec{x}}_k \vec{m} - 2\vec{m}^T \vec{x}_c + \vec{m}^T \vec{m} - r^2$$
$$= 2\hat{\vec{x}}_k \vec{m} - 2\vec{m}\vec{x}_c + \vec{m}^T \vec{m}$$
$$= 2(\hat{\vec{x}}_k - \vec{x}_c)\vec{m} + \vec{m}^T \vec{m}$$

Now $\|\hat{\vec{x}}_k - \vec{x}_c\| = r$, which implies that the first term is going to have its mean and variance scaled by r.

So, as $\vec{m} \sim N(0, \sigma^2 I)$, we get that the first term is a Normal Distribution (as multiplying a Normal distribution by a constant is still normal). In this case, we get that

$2(\hat{\vec{x}}_k - \vec{x}_c)\vec{m} \sim N(0, 4r^2)$ as multiplying a Gaussian Distribution by a scalar value multiplies the variance by a squared amount, and the mean being 0 is unaffected.

As for the second term, it is a Chi-Square Distribution with 2 degrees of freedom. To see this, note that $\vec{m} = [m_1, m_2]$, where each component is normally distributed. As $\vec{m}^T \vec{m} = m_1 m_1 + m_2 m_2$, we have by definition a Chi-Square distribution with 2 degrees of freedom as this is the sum of 2 independent Normal distributions, respective to each coordinate.

One must be careful however. The Chi-Square distribution is defined for sums of squares of Normal Distributions with unit variance. We do not have that here as $\vec{m} \sim N(0, \sigma^2 I)$.

Hence, if we have

$$X_i \sim N(\mu_i, \sigma^2 I) \Rightarrow \sum_{i=1}^{2} \frac{X_i^2}{\sigma^2} \sim \chi_2^2$$

That means $\sum_{i=1}^{2} X_i^2 \sim \sigma^2 \chi_2^2$

So we have a Chi-Square distribution that has been scaled by σ^2.

So, $e_K \sim N(0, 4r^2) + \sigma^2 \chi_2^2$, it depends on r, but not on $\vec{x}_c$.

Note: The mean of $\sigma^2 \chi_2^2$ is $2\sigma^2$ and the variance of $\sigma^2 \chi_2^2$ is $4\sigma^4$ (as by definition without scaling it is k and $2k$ respectively, where $k=2$ here). ☒

B.2. Now in order to consider the parameter σ_N^2 of the assumed Gaussian noise, we can turn to KL Divergence.

The Normal portion of the true distribution will only differ in terms of variance from this assumption, but the Chi-Square will differ more significantly. Consequently, one needs to compute the KL Divergence for that distribution.

Note: A Chi-Square with 2 degrees of freedom is defined to have the following distribution:

$$f_{\chi_2^2} = \begin{cases} \frac{1}{2} e^{-x/2} & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Ours is scaled however, so in our case,

$$f_{\sigma_c^2 \chi_2^2} = \begin{cases} \frac{1}{2\sigma_c^2} e^{-x/2\sigma_c^2} & x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

This follows as if the PDF of a distribution Z, say has a specific form, then $F_{cz}(c \cdot z)$ is found by:

$Pr\{c \cdot z < c\} = Pr\{z < z/c\} = F_z(z/c)$, so

$\frac{d}{dz}[F_z(z/c)] = \frac{1}{c} f_z(z/c)$

Now, I will take $e^K = X$ to simplify my derivation.
Then, if $p(t) = f_{\sigma_c^2 \chi_2^2}$ (σ_c^2 is the σ^2 for this distribution)

and $q(t) = N(0, \sigma_N^2) = \frac{1}{\sqrt{2\pi}\sigma_N} e^{\frac{X^2}{2\sigma_N^2}}$ (σ_N^2 is the σ^2 for this distribution)

The KL divergence is defined to be:

$\int_{-\infty}^{\infty} p(t) \ln\left(\frac{p(t)}{q(t)}\right) dx \quad (*)$

But as $p(t) = 0$ for $x < 0$, $(*)$ becomes:

$$\int_0^{\infty} p(t) \ln\left(\frac{p(t)}{q(t)}\right) dx = \int_0^{\infty} \frac{1}{2\sigma_c^2} e^{-x/2\sigma_c^2} \ln\left(\frac{\frac{1}{2\sigma_c^2} e^{-x/2\sigma_c^2}}{\frac{1}{\sqrt{2\pi}\sigma_N} e^{-x^2/2\sigma_N^2}}\right) dx$$

$$= \frac{1}{2\sigma_c^2} \int_0^{\infty} e^{-x/2\sigma_c^2} \left[\ln\left(\frac{1}{2\sigma_c^2}\right) + \ln\left(e^{-(x/2\sigma_c^2)}\right) - \ln\left(\frac{1}{\sqrt{2\pi}\sigma_N}\right) - \ln\left(e^{-x^2/2\sigma_N^2}\right)\right] dx$$

(as $\log(ab) = \log(a) + \log(b)$ and $\log(a/b) = \log(a) - \log(b)$)

$$= \frac{1}{2\sigma_c^2} \int_0^{\infty} e^{-x/2\sigma_c^2} \left[\ln\left(\frac{1}{2\sigma_c^2}\right) - \left(x/2\sigma_c^2\right) - \ln\left(\frac{1}{\sqrt{2\pi}\sigma_N}\right) + \frac{x^2}{2\sigma_N^2}\right] dx$$

(as $\ln(e^x) = x$)

We can break this integral into parts.

$$= \frac{1}{2\sigma_c^2} \int_0^{\infty} e^{-x/2\sigma_c^2} \ln\left(\frac{1}{2\sigma_c^2}\right) - e^{-x/2\sigma_c^2} \left(x/2\sigma_c^2\right) - e^{-x/2\sigma_c^2} \ln\left(\frac{1}{\sqrt{2\pi}\sigma_N}\right) + e^{-x/2\sigma_c^2} \left(x^2/2\sigma_N^2\right) dx$$

$$= \frac{1}{2\sigma_c^2} \left[\int_0^{\infty} e^{-x/2\sigma_c^2} \ln\left(\frac{1}{2\sigma_c^2}\right) dx - \int_0^{\infty} e^{-x/2\sigma_c^2} \left(x/2\sigma_c^2\right) dx - \int_0^{\infty} e^{-x/2\sigma_c^2} \ln\left(\frac{1}{\sqrt{2\pi}\sigma_N}\right) dx \right.$$
$$\left. + \int_0^{\infty} e^{-x/2\sigma_c^2} \left(x^2/2\sigma_N^2\right) dx\right]$$

So there are 4 integrals to solve. I will do that now.

$$A: \int_0^\infty e^{-(x/2\sigma_c^2)} \ln\left(\frac{1}{2\sigma_c^2}\right) dx$$

$$= \ln\left(\frac{1}{2\sigma_c^2}\right) \int_0^\infty e^{-(x/2\sigma_c^2)}\, dx$$

$$= \ln\left(\frac{1}{2\sigma_c^2}\right) \cdot -2\sigma_c^2 e^{-(x/2\sigma_c^2)} \Big|_0^\infty$$

$$= \lim_{t\to\infty} \ln\left(\frac{1}{2\sigma_c^2}\right) \cdot -2\sigma_c^2 e^{-(t/2\sigma_c^2)} + 1 - \left(\ln\left(\frac{1}{2\sigma_c^2}\right) \cdot e^{-(0)} (-2\sigma_c^2)\right)$$

$$= 0 + 2\sigma_c^2 \ln\left(\frac{1}{2\sigma_c^2}\right)$$

$$B: \int_0^\infty e^{-(x/2\sigma_c^2)} \left(x/2\sigma_c^2\right) dx = \frac{1}{2\sigma_c^2} \underbrace{\int_0^\infty e^{-(x/2\sigma_c^2)} x\, dx}_{\Delta}$$

One can integrate by parts.

$\int u\,dv = uv - \int v\,du$

$u = x \quad dv = e^{-(x/2\sigma_c^2)}$

$du = dx \quad v = -2\sigma_c^2 e^{-(x/2\sigma_c^2)}$

$$\text{So } (\Delta) = -2x\sigma_c^2 e^{-(x/2\sigma_c^2)} + 2\sigma_c^2 \int_0^\infty e^{-(x/2\sigma_c^2)}\, dx$$

$$= -2x\sigma_c^2 e^{-(x/2\sigma_c^2)} \Big|_0^\infty + 2\sigma_c^2 \left[-2\sigma_c^2 e^{-(x/2\sigma_c^2)} \Big|_0^\infty\right]$$

$$= \lim_{t\to\infty} -2t\sigma_c^2 e^{-(t/2\sigma_c^2)} + 2(0) + 2\sigma_c^2 \left[-2\sigma_c^2 e^{-(t/2\sigma_c^2)} + 2\sigma_c^2 e^{(0)/2\sigma_c^2}\right]$$

(the $2(0)$ term and $-2\sigma_c^2 e^{-(t/2\sigma_c^2)}$ term are 0)

Note: $\lim_{x\to\infty} \frac{x}{e^x} \overset{H}{=} \frac{1}{e^x} = 0$ (by L'Hôpital's Rule), so the first term above is 0.

$\Delta = 4\sigma_c^4$

So $B = \frac{1}{2\sigma_c^2} \cdot 4\sigma_c^4 = 2\sigma_c^2$

C: $\int_0^\infty e^{-x/2\sigma_c^2} \ln\left(\frac{1}{\sqrt{2\pi}\sigma_N}\right) dx$

$= \ln\left(\frac{1}{\sqrt{2\pi}\sigma_N}\right)\left[-2\sigma_c^2 e^{-(x/2\sigma_c^2)}\Big|_0^\infty\right]$

$= \ln\left(\frac{1}{\sqrt{2\pi}\sigma_N}\right)\left[\lim_{t\to\infty} -2\sigma_c^2 e^{-(t/2\sigma_c^2)} + 2\sigma_c^2 e^{-(0)}\right]$

$= \ln\left(\frac{1}{\sqrt{2\pi}\sigma_N}\right)\left[2\sigma_c^2\right]$

D: $\frac{1}{2\sigma_N^2}\int_0^\infty e^{-(x/2\sigma_c^2)} x^2 dx$

Integrate by parts. $\int u dv = uv - \int v du$

$\frac{1}{2\sigma_N^2}\left[\int u dv\right]$ (□)

$u = x^2 \quad dv = e^{-(x/2\sigma_c^2)} dx$

$du = 2x dx \quad v = -2\sigma_c^2 e^{-(x/2\sigma_c^2)}$

So, $(□) = \frac{1}{2\sigma_N^2}\left[-2x^2\sigma_c^2 e^{-(x/2\sigma_c^2)} + \int_0^\infty 4x\sigma_c^2 e^{-(x/2\sigma_c^2)} dx\right]$

($\int u dv$ = ☆)

Integrate by parts again to obtain (☆)

(☆) $= \int u dv = uv - \int v du$

$u = x \quad dv = 4\sigma_c^2 e^{-(x/2\sigma_c^2)} dx$

$du = dx \quad v = -8\sigma_c^4 e^{-(x/2\sigma_c^2)}$

So (☆) $= -8x\sigma_c^4 e^{-(x/2\sigma_c^2)} + \int_0^\infty 8\sigma_c^4 e^{-(x/2\sigma_c^2)} dx$

$= -8x\sigma_c^4 e^{-(x/2\sigma_c^2)} - 16\sigma_c^6 e^{-(x/2\sigma_c^2)}$

Then (□) $= \frac{1}{2\sigma_N^2}\left[-2x^2\sigma_c^2 e^{-(x/2\sigma_c^2)} - 8x\sigma_c^4 e^{-(x/2\sigma_c^2)} - 16\sigma_c^6 e^{-(x/2\sigma_c^2)}\right]\Big|_0^\infty$

$= \lim_{t\to\infty} \frac{1}{2\sigma_N^2}\left[-2t^2\sigma_c^2 e^{-(t/2\sigma_c^2)} - 8t\sigma_c^4 e^{-(t/2\sigma_c^2)} - 16\sigma_c^6 e^{-(t/2\sigma_c^2)}\right]$

(0 by l'Hôpital's Rule; 0)

$-\frac{1}{2\sigma_N^2}\left[0 - 0 - 16\sigma_c^6\right] = 8\frac{\sigma_c^6}{\sigma_N^2}$

Finally, putting it all together,

$(*) = \frac{1}{2\sigma_c^2}[A - B - C + D]$

$= \frac{1}{2\sigma_c^2}\left[2\sigma_c^2 \ln\left(\frac{1}{2\sigma_c^2}\right) - 2\sigma_c^2 - \ln\left(\frac{1}{\sqrt{2\pi}\sigma_N}\right)[2\sigma_c^2] + 8\frac{\sigma_c^6}{\sigma_N^2}\right]$

$= \ln\left(\frac{1}{2\sigma_c^2}\right) - 1 - \ln\left(\frac{1}{\sqrt{2\pi}\sigma_N}\right)[\quad] + 4\frac{\sigma_c^4}{\sigma_N^2}$

To find the value of σ_N^2, we can take the derivative of $(*)$ w.r.t. σ_N^2.

As $\ln\left(\frac{1}{\sqrt{2\pi}\sigma_N}\right) = -\frac{1}{2}\ln\left(\frac{1}{2\pi\sigma_N^2}\right)$

$\frac{\partial(*)}{\partial\sigma_N^2} = 0 - 0 - 0 + \frac{\partial}{\partial\sigma_N^2}\left[\frac{1}{2}\ln\left(\frac{1}{2\pi\sigma_N^2}\right)\right] + \frac{\partial}{\partial\sigma_N^2}\left[\frac{4\sigma_c^4}{\sigma_N^2}\right] = 0$

$\Leftrightarrow \frac{1}{2\left(\frac{1}{2\pi\sigma_N^2}\right)}\left(\frac{1}{2\pi}\right) - \frac{4\sigma_c^4}{\sigma_N^4} = 0$

$\Leftrightarrow \frac{1}{2\sigma_N^2} = \frac{4\sigma_c^4}{\sigma_N^4}$

$\Leftrightarrow \frac{\sigma_N^2}{2} = 4\sigma_c^4 \quad \Leftrightarrow \quad \sigma_N^2 = 8\sigma_c^4$

So the value of σ_N^2 is simply $2(2\sigma_c^2)^2$, as this maximizes the likelihood of $(*)$.

One must also consider the normal component of the true distribution of e_K however. To do this, note that the sum of normal distributions is normal, and follows:

$X \sim N(\mu_1, \sigma_1^2)$, $Y \sim N(\mu_2, \sigma_2^2)$

$Z = X + Y \Rightarrow Z \sim N(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$

So by factoring in the original normal portion's distribution, $e_N \sim N(0, 4r^2)$, we get that this distribution should have a variance of

$\sigma_N^2 = 8\sigma_c^4 + 4r^2 = 4(2\sigma_c^4 + r^2)$, and mean 0.

Now, to find a least square estimator, we can take note of the i.i.d. measurements and noting that $e_k \sim N(0, \sigma_N^2 I)$, we can then note that the joint probability function is

$$p(e_1, e_2, \ldots, e_k | 0, \sigma_N^2) = \prod_{k=1}^{K} p(e_k | 0, \sigma_N^2)$$

$$= \prod_{k=1}^{K} \frac{1}{\sqrt{2\pi}\sigma_N} e^{-\frac{e_k^2}{2\sigma_N^2}}$$

This is simply the likelihood function. Maximizing the likelihood here is equivalent to minimizing the negative log likelihood (as log is a monotonic function). So

$$-\ln\left(p(e_1, \ldots, e_k | 0, \sigma_N^2)\right) = -\ln\left(\prod_{k=1}^{K} \frac{1}{\sqrt{2\pi}\sigma_N} e^{-\frac{e_k^2}{2\sigma_N^2}}\right)$$

$$= -\sum_{k=1}^{K} \ln\left(\frac{1}{\sqrt{2\pi}\sigma_N} e^{-\frac{e_k^2}{2\sigma_N^2}}\right) \quad \text{(by the property of logs that } \log(ab) = \log(a) + \log(b)\text{)}$$

$$= -\sum_{k=1}^{K} \ln\left(\frac{1}{\sqrt{2\pi}\sigma_N}\right) + \frac{e_k^2}{2\sigma_N^2} \quad (\text{as } \ln(e^x) = x)$$

$$= \frac{K}{2}\ln\left(\frac{1}{2\pi\sigma_N^2}\right) + \sum_{k=1}^{K} \frac{e_k^2}{2\sigma_N^2} \quad (\text{as } \ln(x^b) = b\ln(x))$$

In order to maximize the likelihood, we take the derivative w.r.t. the parameters of our function e_k, in this case $\vec{a}$ and b, and set it to 0. Define $Q = [\vec{a}\ b]^T$, then

$$0 = \frac{-\partial \ln(p(e_1, \ldots, e_k | 0, \sigma_N^2)}{\partial Q} = \frac{\partial}{\partial Q}\sum_{k=1}^{K} \frac{e_k^2}{2\sigma_N^2} = \sum_{k=1}^{K} \frac{\partial}{\partial Q} \frac{e_k^2}{2\sigma_N^2} \quad (*)$$

As $e_k = \vec{a}^T \vec{x}_k + b + \vec{x}_k^T \vec{x}_k$, note that by reparametrizing

$$e_k = [\vec{x}_k\ 1][\vec{a}\ b]^T + \vec{x}_k^T \vec{x}_k = [\vec{x}_k\ 1]Q + \vec{x}_k^T \vec{x}_k$$

So, $(*) = \frac{1}{\sigma_N^2}\sum_{k=1}^{K} \frac{\partial}{\partial Q}\left([\vec{x}_k\ 1]Q + \vec{x}_k^T \vec{x}_k\right)^2 = 0$

$$0 = \sum_{k=1}^{K} \left([\vec{x}_k\ 1]Q + \vec{x}_k^T \vec{x}_k\right)(2)\left([\vec{x}_k\ 1]\right)$$

So, $0 = \sum_{k=1}^{K} ([\vec{x}_k\ 1]Q + \vec{x}_k^T\vec{x}_k)[\vec{x}_k\ 1]$

But that implies $\sum_{k=1}^{K} [\vec{x}_k\ 1]Q[\vec{x}_k\ 1] = -\sum_{k=1}^{K} \vec{x}_k^T\vec{x}_k[\vec{x}_k\ 1]$

$$\sum_{k=1}^{K} [\vec{x}_k\ 1]^T[\vec{x}_k\ 1]Q = -\sum_{k=1}^{K} [\vec{x}_k\ 1]\vec{x}_k^T\vec{x}_k$$

So we can define a matrix X such that X is $K \times 3$, where each row is $[x_{Kx}\ x_{Ky}\ 1]$, similarly define a matrix Y that is $K \times 1$ so that each row is $\vec{x}_k^T\vec{x}_k$, then we have a system as follows:

$$X^TXQ = -X^TY$$

$$\Leftrightarrow \hat{Q} = -(X^TX)^{-1}(X^TY) \quad \text{(as } X^TX \text{ is } K \times K\text{, it is clearly invertible).}$$

Solving the above equation gives the least squares estimate for $\hat{a}$ and $\hat{b}$, which are the optimal parameters for e_k.

B 3. Circle Proposals

This is my algorithm for choosing circle proposals:

A) Compute the bounding box of the edgels. This is done by finding the minimum and maximum x & y coordinates of the edgel positions. Using this bounding box, compute the diagonal across the bounding box. No proposed circle can have a radius larger than this. Store the value as max_r.

B) Choose two edgels at random from the edgel data p.

C) Using some elementary 2D geometry, I determine if the normals of these two edgels intersect. More specifically, as the equation of a line can be seen as:

$y = mx + b$

We have two lines:

$y\_1 = m\_1*x\_1 + b\_1$
$y\_2 = m\_2*x\_2 + b\_2$

I take the two normals of the random edgels and get their slopes as follows:

slope_1 = normals(random_edgel_index_1, 2) / normals(random_edgel_index_2, 1);
slope_2 = normals(random_edgel_index_2, 2)/ normals(random_edgel_index_2, 2);

I then check if slope_1 == slope_2. If this is the case, the lines are parallel, and I repeat step B and try to find different random edgels to compare.

D) If the slopes are not parallel, I calculate where they intersect. This can be done through some simple algebra:

The x coordinate of intersection: $x\_1 = x\_2$

This means
$y\_1 = m\_1 * x\_1 + b\_1$
and
$y\_2 = m\_2* x\_1 + b\_2$

Rearranging, we get that

$x\_cross = (b\_2 – b\_1) / (m\_1 – m\_2)$

Hence:

$y\_cross = m\_1 * x\_cross + b\_1$

So the coordinates of intersection are x_cross and y_cross.

E) Next I verify that the intersection of the two lines occurs inside the circumference of the proposed circle. This can be done by seeing if the intersection point lies in the direction of both normals. If it does not, go back to step B.

F) Using the point where the lines cross and the actual edgel positions, I next calculate the distance between the normal intersections and the edgel positions. These are the two proposed radii. If either one of these propsed radii is larger than max_r, go back to Step B.

G) It is also important that the two radii are similar in value, as in a given circle the radius should be the same from any edgel position. I compute the ratio of both (radius_1 /radius_2 and radius_2/radius_1) and ensure they fall within a threshold of 1.4. If either ratio is larger than this threshold, go back to step B.

H) Take x_c to be the coordinates (x_cross, y_cross) and the value of r to be the mean of (radius_1 and radius_2).

I chose this algorithm over something more complex like a Hough accumulator as it is not difficult to implement, nor is it computationally intensive. The major gain with this algorithm is in space complexity in particular (it is only O(n) in space complexity) and it is O(n^2) in time complexity. This follows as it only involves some elementary 2D geometry to generate the necessary P proposals. By considering parallel lines, the location of the intersection of edgel normals and the ratios of the computed radii, I actually remove bad proposals every time this algorithm is run. We want circles where the radius is the same from every point on its circumference to its centre after all, and this algorithm ensures that these are produced.

B 4. Circle Selection

With K circle proposals, I select the one with the least error. More specifically, I use the Geman-Mclure (p(e,sigma_g) = e^2 / (sigma_g^2 + e^2)) estimator to evalued the error for a given circle relative to its edgels. The following is done:

A) Loop through the proposed circles and compute p(e_k, sigma_g) for a given edgel e_k. Sum this value up across the edgels for the proposed circle.

B) Select the circle which has the smallest value of p(e,sigma_g) as the best circle proposal.

I choose the Geman-Mclure estimator as it is robust against outliers. It also makes sense to choose the circle with the smallest error as the best circle as in this case, that means it best fits the set of edgels.

B 5. Robust Fitting

B. 5. We want to minimize the objective function, for the circle parameters

$\vec{a} = -2\vec{x}_c$

$b = \vec{x}_c^T \vec{x}_c - r^2$

So $O(\vec{a}, b) = \sum_k \rho(e_k(\vec{a}, b), \sigma)$

where $\rho(e_k, \sigma) = e_k^2 / \sigma^2 + e_k^2$

with $e_k = \vec{a}^T \vec{x}_k + b + \vec{x}_k^T \vec{x}_k$

To optimize $O(\vec{a}, b)$ for both $\vec{a}$ and b, take $Q = [\vec{a}\ b]^T$.

Then one has the following optimization:

$$0 = \frac{\partial O(\vec{a}, b)}{\partial Q} = \sum_k \frac{\partial \rho(e_k(\vec{a}, b), \sigma)}{\partial Q} = \sum_k \frac{\partial \rho(e_k(\vec{a}, b))}{\partial e_k} \frac{\partial e_k}{\partial Q}$$ (through Chain Rule)

To begin, $\frac{\partial \rho(e_k, \sigma_g)}{\partial e_k} = \frac{\partial}{\partial e_k} \frac{e_k^2}{\sigma_g^2 + e_k^2}$

$$= 2 e_k \cdot \left(\frac{\sigma_g}{\sigma_g^2 + e_k^2}\right)^2$$

Let $P(e_k) = \left(\frac{\sigma_g}{(\sigma_g^2 + e_k^2)}\right)^2$

So that $\frac{\partial \rho(e_k, \sigma_g)}{\partial e_k} = 2 \cdot e_k \cdot P(e_k)$

Now we can compute $\frac{\partial e_k}{\partial Q}$

As $e_k = \vec{a}^T \vec{x}_k + b + \vec{x}_k^T \vec{x}_k$, we can reparametrize so that

$e_k = [\vec{x}_k\ 1]\, Q + \vec{x}_k^T \vec{x}_k$

So then $\frac{\partial e_k}{\partial Q} = [\vec{x}_k \; 1]$

Plugging this back into $0 = \sum_k \frac{\partial p(e_k(\vec{a}, b), \sigma_g}{\partial e_k} \frac{\partial e_k}{\partial Q}$, we have

$$0 = \sum_k e_k \cdot P(e_k) \cdot [\vec{x}_k \; 1]$$

$$\Leftrightarrow \quad 0 = \sum_k P(e_k) \cdot ([\vec{x}_k \; 1] Q + \vec{x}_k^T \vec{x}_k) \cdot [\vec{x}_k \; 1]$$

(as the dot product is commutative and $e_k = [\vec{x}_k \; 1] Q + \vec{x}_k^T \vec{x}_k$

$$\Leftrightarrow \sum_k P(e_k) \cdot [\vec{x}_k \; 1] Q \cdot [\vec{x}_k \; 1] = -\sum_k P(e_k) \cdot \vec{x}_k^T \vec{x}_k \cdot [\vec{x}_k \; 1]$$

Now this can be rewritten in matrix form so that

$P \subseteq \mathbb{R}^{k \times k}$ contains all $P(e_k)$ along its diagonal, $k \in [1, k]$

$X \subseteq \mathbb{R}^{k \times 3}$ where each row is $[\vec{x}_k \; 1] = [x_k \; x_{k+1} \; 1]$

and $Y \subseteq \mathbb{R}^{k \times 1}$ is a matrix with $[\vec{x}_k^T \vec{x}_k]$ in each row.

The above equation then becomes:

$$(X^T P X) Q = -X^T P Y$$ (note that some rearranging is done so the dimensions line up).

So solving for $Q \subseteq \mathbb{R}^{3 \times 1}$, we obtain values for $\vec{a}$ and b. The first two rows are components of $\vec{a}$ and the third row is b. Using the least squares method we can solve for $\vec{a}$ and b so that

$$\vec{x}_c = -\vec{a}/2 \qquad r = \sqrt{\vec{x}_c^T \vec{x}_c - b}$$

Here is my algorithm:

A) Use the initial parameters:
a= -2 * x0;
b= x0 * x0’ - initr^2;

to form the matrices X,Y, and P defined above.

B) Solve the WLS problem to get circle parameters.

Compute the following to obtain values for Q:

Q = (X’ * P * X)^-1 * (-X’ * P * Y);

C) Use the parameters obtained from Q to set the parameters of a and b in step A.

D) I loop over steps A-C up to 50 times. At each iteration, I check if the sum of parameter values in Q has changed from the previous iteration. To check for convergence, I see if the sum of the parameter values is less than 0.05. If it has converged or 50 iterations are reached I return the values of:

x0 = a/(-2);

r = sqrt(dot(x0,x0’)-b);

where a and b are the parameters estimated during the last iteration.

To find a suitable value of sigma_g, I experiment with different values and compare how it fits circles for different values. Specifically, I try with sigma_g = 1, 5, 50, 500

Here are the results with a given cell in the first image:

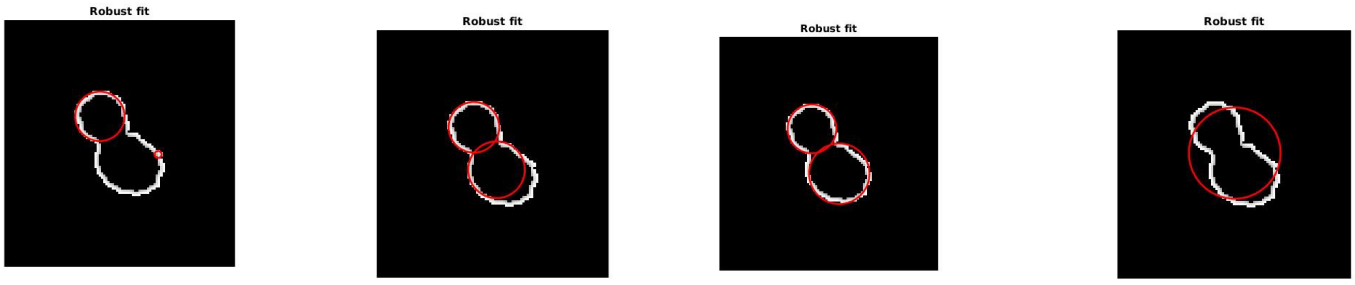


Figure B.1: Plots of a Cell for Sigma_G = 1, 5, 50 and 500 respectively.

With smaller sigma_g, it appears that smaller circles are generated. This makes sense as with sigma_g quite small, the GM-error function becomes 1, so points that are far away don't contribute as much to the error. When sigma_g is very large, the fitting will attempt to cover more points, as the variance is very large. It seems that sigma_g between 5 and 50 gives the best results. After narrowing it down further, I found that the value of sigma_g=40 gave me the best results.

B 6. Model Update

In order to decide if a robust fitted circle is good, I do the following:

If nFound == 0 (so the first circle to be found using the current set of edgels) I compare the sum of weights across the edgels to a threshold. I found that a weightThreshold of 0.25 times the circumference of the circle worked well. It makes sense to use the weights in evaluating the initial circles as these have the most edgels for calculating weights.If the sum of the weights is less than the threshold, then I say that the circle is not good.

Otherwise, if nFound >0, I then check if the circle overlaps with circles that were previously found. If the distance between the center points of the two circles is less than the difference of their radii squared, I reject the new circle as that implies that one circle contains the other. This follows as the furthest points between a circle and a point would be on a line from the point, through the center of the circle. So a circle B is contained in a circle A iff circle A contains the furthest point, which implies:

r_A >= sqrt((x_A – x_B) ^2 + (y_A – y_B)^2) + r_B
Or that (r_A-r_B)^2 >= (x_A – x_B) ^2 + (y_A – y_B)^2 *(see reference 1)*

I then check if the distance between the two circles is less than the sum of their radii squared. This implies that the two circles are near each other and could in fact intersect. I do allow for some intersection to occur (as we do want to detect buds of cells), but after testing, I find that by ensuring that the intersecting angle is within 3* pi, small buds are retained. In the case of multiple circles intersecting, I compute the sum of intersecting angles and again compare it to the threshold of 3 * pi. This ensures that the circles do not intersect too much, while retaining buds.

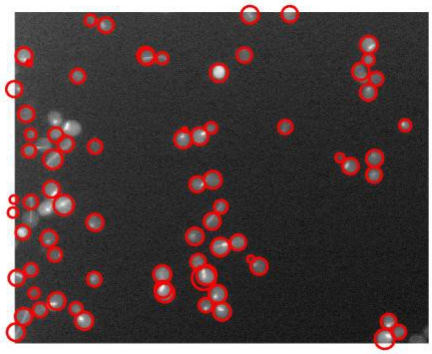
In order to calculate the intersecting angle, I use these formulas:

x = (distance^2 – r^2 + R^2)/ 2*d, where r is the radius of one circle and R^2 is the radius of another.

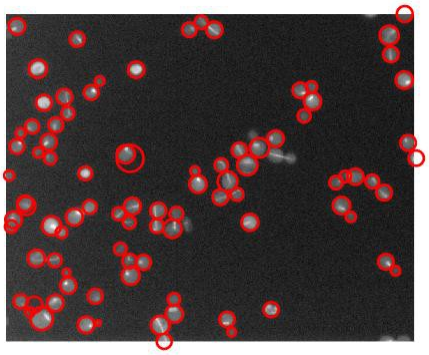
Then the intersecting angle is 2*arcsin(x/r).

(I obtained these formulas from reference 2).

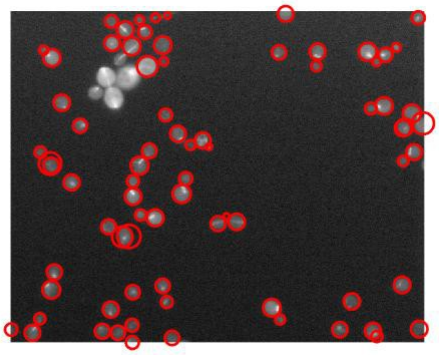
B 7. Evaluation



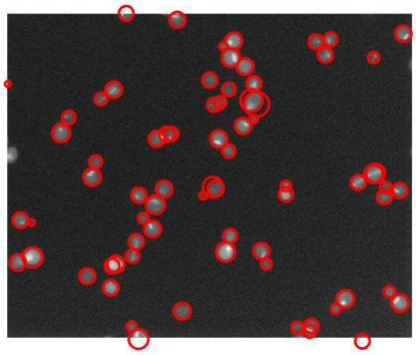
Cell Image 1



Cell Image 2



Cell Image 3



Cell Image 4

Figure B2: Cell Images

Interestingly, my algorithm is fairly adept at finding small buds and significantly cropped cells near the edge of the image. It has issues with eliptical shapes (particularly this one below) as well as large clusters of cells (such as those in images 1 and 3). I suspect these problems are due to my choice of a GM error estimator as well as the parameters I am using to penalize intersections and summed angles. For the former problem, potentially switching to a least squares estimator could help alleviate the issue, although at the expense of noiser or worsened performance. I could also modify my estimation to account for ellipses instead of simply circles as well, but that would require solving for futher parameters to fit the data to an elliptical model. For the second problem of clusters, I suspect that my angle penalization may be too strict. More specifically, in large clusters there are likely to be several circles that intersect and my penalization may be too conservative to account for all of these. I have

experimented with a smaller angleThreshold criteria however, and I found the results tend to deteriorate (there are many circles with significant overlap if I lower the angle threshold to even 4*pi), so I will keep my conservative settings in order to reduce overlaps, at the expense of large cluster accuracy.

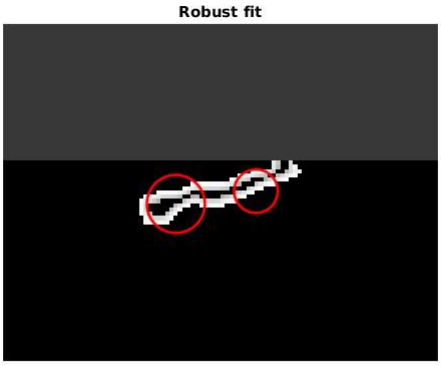


Figure B3: The Robust Estimator Has Issues With Eliptical Shapes

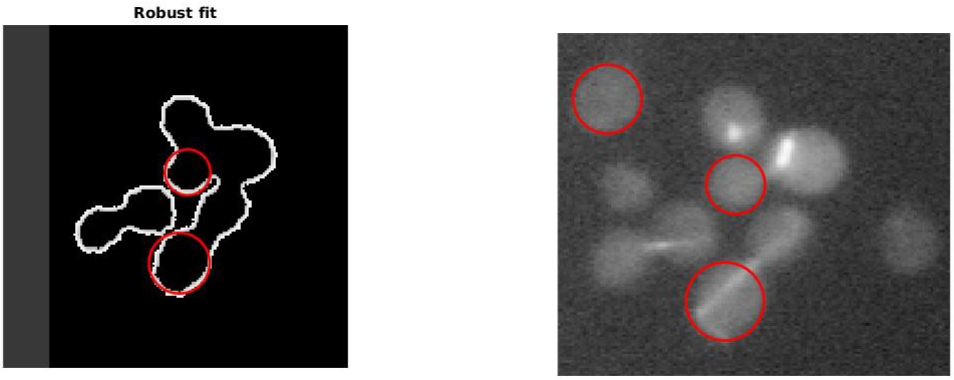


Figure B4:Two Views of a Cluster of Cells. My Angle Threshold Prevents Too Much Overlap in Proposals

References

1: http://math.stackexchange.com/questions/275514/two-circles-overlap

2: http://www.had2know.com/academics/intersection-angle-two-circles.html