UNIVERSITY OF
TORONTO

Department of Computer Science

CSC2503H: Foundations of Computer Vision

Assignment #2:

Camera Obscura & Robust Estimation

Sanjif Rajaratnam

999091986
rajara24

Course Instructor: Dr. Kyros Kutulakos
November 16th, 2016

Table of Contents

Table of Figures iii

1.0 Camera Obscura 1

1.1 Basic Operation 1

1.2 Different Pin-hole Shapes & Sizes 3

1.3 Modeling Lens-Less Image Formation 6

2.0 Robust Estimation 8

2.1 Noise Model 8

2.2 LS Estimator 9

2.3 Circle Proposals 11

2.4 Circle Selection 12

2.5 Robust Fitting 13

2.5.1 IRLS Algorithm 15

2.5.2 Selecting the GM Estimator 15

2.6 Model Update 16

2.7 Brief Evaluation 17

2.7.1 Overlapping Circles 19

2.7.2 Missing Circle in Cluster 19

3.0 References 20

Table of Figures

Figure 1: Pin-hole Camera 1

Figure 2: Lassonde Mining Building Sign 2

Figure 3: Scene captured by Pin-hole Camera 3

Figure 4: Disc-Shaped Pin-hole 4

Figure 5: Squared-Shaped Pin-hole 4

Figure 6: Cross-Shaped Pin-hole 5

Figure 7: Hexagonal-Shaped Pin-hole 5

Figure 8: 2D Lens Model 6

Figure 9: Sample robust estimation with various σg values 16

Figure 10: Example of imperfect circle missing lots of edgels 17

Figure 11: Results from robust estimation 18

Figure 12: Overlapping circles 18

Figure 13: Missing circle in cluster 19

1.0 Camera Obscura

1.1 Basic Operation

The pin-hole camera in this section was made using an old cardboard box. A hole was cut on one side and a piece of a paper was placed on the opposite side. A picture of the pin-hole camera can be seen in **Figure 1**. The top of the box was closed, and covered with a jacket to prevent light from entering through the top. A camera was placed inside the box, and set to take a long exposure shot with 30 seconds of exposure time for the pin hole, and 8 seconds of exposure time for the larger holes. There was a delay set for the camera to open so that the box could be closed and covered. The scene that was photographed is the bicycle stands in front of the sign of the Lassonde Mining Building at the University of Toronto. The sign can be seen in **Figure 2**.

Figure 1: Pin-hole Camera

Figure 2: Lassonde Mining Building Sign

The procedure for shooting an image was as follows: (1) the pin-hole camera was setup so that the pin-hole was orientated parallel to the scene it was shooting; (2) a camera was placed inside under the pin-hole and set to long exposure to capture the shot; (3) the box was closed and covered; and (4) the box remained closed and covered until the capture completed its exposure shot so only light entering the box is from the pin-hole.

The image captured by the pin-hole camera can be seen in **Figure 3**. The image shown is the raw image that has been linearly scaled to maximize brightness and clarity. The image was projected upside down, and very clear. This is as expected, as a pin-hole camera will cause the incident light to get projected upside down due to light only being able to travel straight. An

ideal pin-hole is infinitesimally small and, therefore, a point in space projects to one point in the image plane so the image, as expected, is in focus.

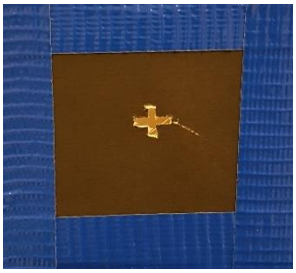
Figure 3: Scene captured by Pin-hole Camera

1.2 Different Pin-hole Shapes & Sizes

The same scene from **Figure 2** was shot using the various pin-hole shapes. The camera was setup in the same way as well. **Figure 4** shows the disc pin-hole and the image that resulted from that pin-hole. **Figure 5**, **Figure 6**, and **Figure 7** show the square-shaped, cross-shaped, and hexagonal-shaped pin-holes, and their resultant projected images, respectively.

Figure 4: Disc-Shaped Pin-hole

Figure 5: Squared-Shaped Pin-hole



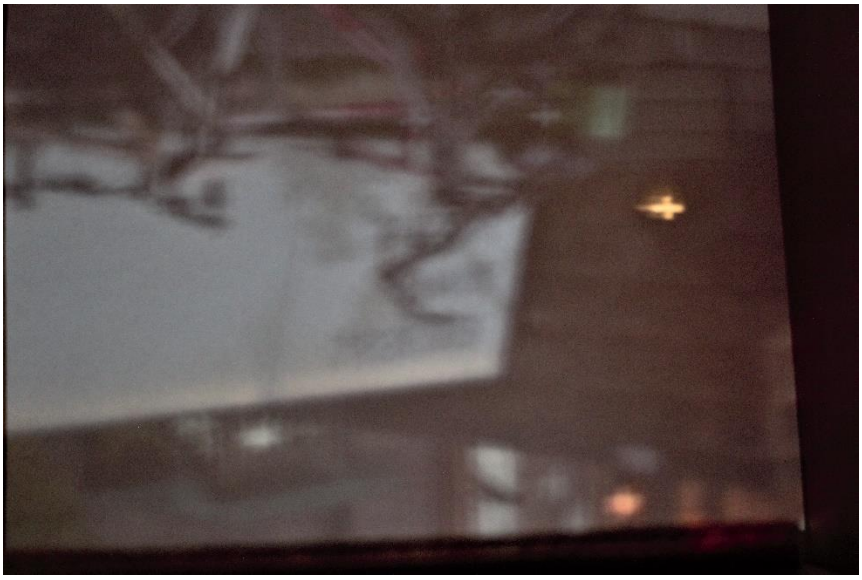
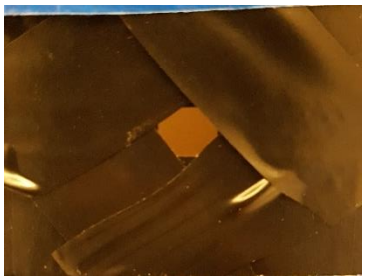


Figure 6: Cross-Shaped Pin-hole



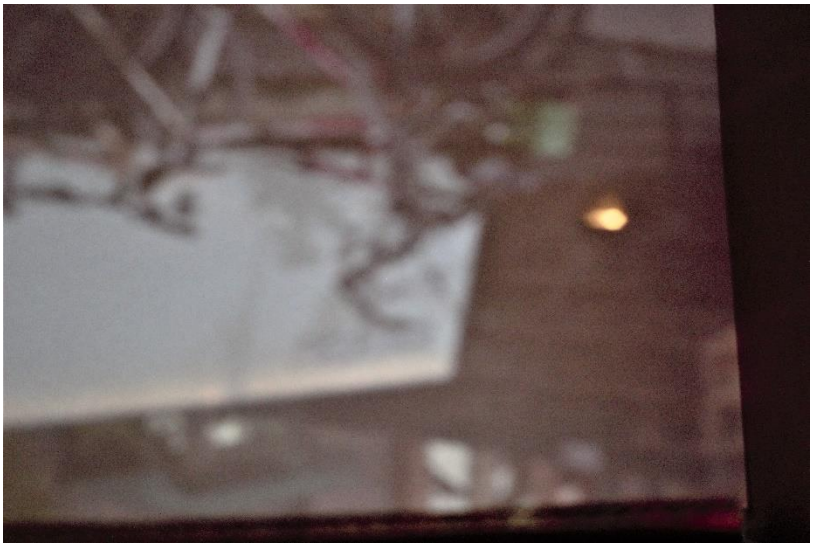


Figure 7: Hexagonal-Shaped Pin-hole

These images were taken with 8 seconds of exposure time because the larger holes allowed more light to enter the box. The image becomes saturated more quickly. The disc pin-hole resulted in the sharpest image of all the shapes. All the images had a halo from a light source in the image that showed the shape of the blur circle. It is seen that the blur circle is in the shape of the pin-hole. The disc shaped pin-hole resulted in blurring circles for incident lights. The

square pin-hole resulted in blurring diamonds because of how it was oriented. It blurred the most along the central x and y axis, with the corners the least blurred. The cross pin-hole resulted in a blurring cross with most of the blurring within the cross-shape projection. The hexagonal pin-hole was the most blurred of them all. This could be because it was the largest of the pin-holes sizes.

1.3 Modeling Lens-Less Image Formation

The "ideal" image of the scene can be modeled by a pin-hole camera image with A = 1 in the following equation:

$$I_{pin\_hole}(x, y) = A \cdot I(x, y) = I(x, y)$$

The pin-hole has the lowest intensity values because it has the smallest size (infinitesimally small). It allows the least light to reach the sensor plane. The different shapes can be modelled by using the assumption that they are created by several discrete pinholes. The model can be built from the assumption that each shape can be discretized in the x-direction using the model for the y seen in **Figure 8**.

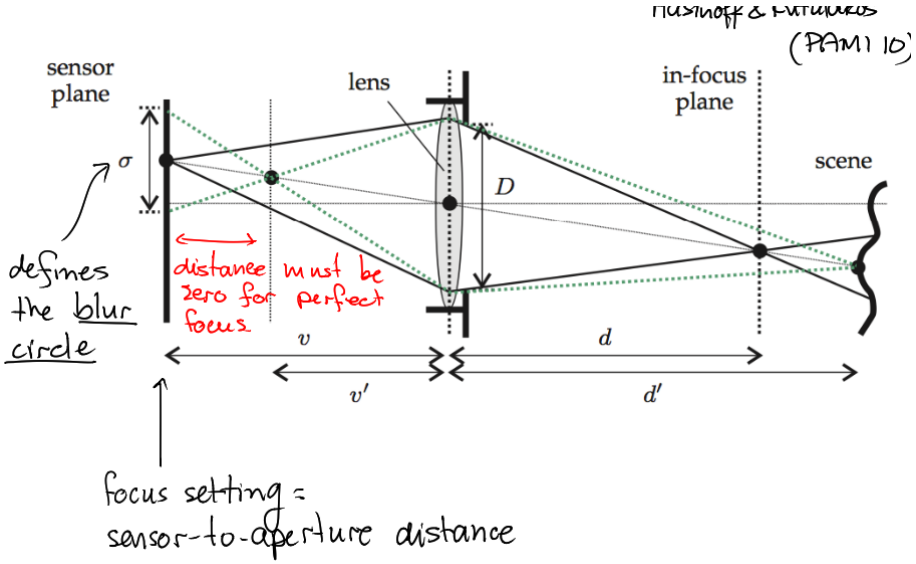


Figure 8: 2D Lens Model

The equation for the disc can be obtained by multiplying the intensity by the area of the circle:

$$A = \pi r^2 \quad \rightarrow \quad I_{disc}(x, y) = \pi r^2 \cdot I(x, y)$$

Where r is the radius of the disc.

Similarly, for a box:

$$A = a^2 \rightarrow I_{box}(x, y) = a^2 \cdot I(x, y)$$

Where a is the side length of the box.

Similarly, for a cross:

$$A = a^2 + 4ab \rightarrow I_{cross}(x, y) = (a^2 + 4ab) \cdot I(x, y)$$

Where a is the short side length, and b is the long side length.

Similarly, for a hexagon:

$$A = \frac{3\sqrt{3}}{2} a^2 \rightarrow I_{hexagon}(x, y) = \left(\frac{3\sqrt{3}}{2} a^2\right) \cdot I(x, y)$$

Where a is the side length of the hexagon.

It is evident that the area results in a higher intensity image. For example, the cross is thin at the ends and wide at the middle. This means that light at the end reflect shorter distances (and shorter angles) and create smaller blur patches and light along the central axis will reflect greater distances (with larger angles). The Lambertian assumption is required because perfect specular reflection is required at the pin-hole interface for the model to flipped and projected upside-down. The scene's planarity and parallelism to the front plane is required because this eliminates the foreshortening effect for the intensity.

2.0 Robust Estimation

2.1 Noise Model

Firstly, it is known that the measured points (x_k) are equal to the true point ($\hat{\vec{x}}_k$) plus a Gaussian noise ($\vec{m}$). The noise can be modelled with the independent, isotropic Gaussian noise model: $\vec{m} = \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. Next, the assumption is made that the model parameters (circle center $\vec{x}_c$, radius, r, and subsequently the true points on the circle $\hat{\vec{x}}_k$) are set values independent of noise and have no variance. Their distributions can be seen in **Equation 1**.

$$\vec{x}_c \sim \mathcal{N}(\vec{x}_c, 0), \qquad r \sim \mathcal{N}(r, 0) \quad \rightarrow \quad \hat{\vec{x}}_k \sim \mathcal{N}\left(\hat{\vec{x}}_k, 0\right) \qquad (1)$$

Then, from *Equation 3* of the manual, it is seen that $\vec{x}_k$ is equal the sum of two Gaussian distributions. The sum of two Gaussian distributions results in a Gaussian distribution with a mean equal to the sum of the means, and a variance equal to the sum of the variances **[1]**. Therefore, the distribution for $\vec{x}_k$ can be seen in **Equation 2**.

$$\vec{m} \sim \mathcal{N}(0, \sigma \boldsymbol{I}) \; + \; \hat{\vec{x}}_k \sim \mathcal{N}\left(\hat{\vec{x}}_k, 0\right) \; \rightarrow \quad \vec{x}_k \sim \mathcal{N}\left(\hat{\vec{x}}, \sigma^2 \mathbf{I}\right) \quad (2)$$

Then, from *Equation 2* of the manual, $\vec{d}_k$ can be set equal to $\vec{x}_k - \vec{x}_c$. This is also a sum of two normally distributed Gaussian distributions. Therefore, the distribution for $\vec{d}_k$ can be derived in the same way as $\vec{x}_k$, and it can be seen in **Equation 3**. The distribution for $\vec{d}_k$ can then be separated into its parametric equations as seen in **Equation 4**.

$$\vec{x} \sim \mathcal{N}\left(\hat{\vec{x}}_k, \sigma^2 \boldsymbol{I}\right) \; + \; (-\vec{x}_c) \sim \mathcal{N}(-\vec{x}_c, 0) \; \rightarrow \vec{d}_k \sim \mathcal{N}\left(\hat{\vec{x}}_k - \vec{x}_c, \sigma^2 \boldsymbol{I}\right) \quad (3)$$

$$\vec{d}_k = \vec{x}_k - \vec{x}_c = \begin{bmatrix} d_{k,x} \\ d_{k,y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \hat{x}_{k,x} \\ \hat{x}_{k,y} \end{bmatrix} - \begin{bmatrix} x_{c,x} \\ x_{c,y} \end{bmatrix}, \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}\right) \quad \rightarrow$$

$$\begin{matrix} d_{k,x} \sim \mathcal{N}(\hat{x}_{k,x} - x_{c,x}, \sigma^2) \\ d_{k,y} \sim \mathcal{N}(\hat{x}_{k,y} - x_{c,y}, \sigma^2) \end{matrix} \qquad (4)$$

Now, $\left|\left|\vec{d}_k\right|\right|^2$ can be found by summing its squared parameters as seen in **Equation 5.** The sum of squares of 2 independent standard normal distributions results in a chi-squared distribution with 2 degrees of freedom (χ_2^2) **[2]**.

$$\left|\left|\vec{d}_k\right|\right|^2 = d_{k,x}^2 + d_{k,y}^2 \quad (5)$$

The r is another squared normal distribution with 0 variance. The resultant e_k function can therefore be modelled by a chi-squared distribution with 3 degrees of freedom (χ_3^2). From this formulation, it is seen that the circle center, $\vec{x}_c$, affects the mean, and the radius, r, affects the variance of the e_k distribution.

2.2 LS Estimator

The Kullback-Leibler (KL) divergence from Q to P (**Equation 6**) can be used to model the difference between the probabilities of the chi-squared distribution (Q) derived in the **Section 2.1** and the Gaussian distribution approximation (P) for the noise **[3]**.

$$D_{KL}(P||Q) = \sum_i \Pr(i) \log \frac{P(i)}{Q(i)} \quad (6)$$

$$e_k \sim \mathcal{N}(0, \sigma^2) \quad (7)$$

Now, continuing with the assumption the e_k can be modeled with a Gaussian Normal distribution. The objective is to maximize the likelihood of getting the sample, given the model. This is equal to maximizing the probability of getting the sample, given the model:

$$\mathcal{O}(\mu, \sigma, \vec{a}, b) = \arg\max_{\mu,\sigma,\vec{a},b} Pr(\vec{x}_1, \dots, \vec{x}_k | \mu, \sigma, \vec{a}, b) \quad (8)$$

But, it is known that $\{\vec{x}_k\}_{k=1}^N$ are independent. Therefore **Equation 8** can be re-written as:

$$\Pr(\vec{x}_1, \dots, \vec{x}_k | \mu, \sigma^2, \vec{a}, b) = \prod_{k=1}^{N} Pr(\vec{x}_k | \mu, \sigma, \vec{a}, b) \quad (9)$$

The probability of getting the sample, given the model is equivalent to the probability of getting the model, given the model, and the noise, given the model because the sample is equal to model plus the noise. Also, it can logically be concluded that the probability of getting the model, given the model is equal to 1. Therefore, the overall probability of getting the sample, given the model is equivalent to getting the noise, given the model. Since the samples are independent, the noise for those samples are also independent. It is also given the mean for the noise is 0. This all can be seen in **Equation 10**.

$$\Pr(\vec{x}_1, \dots, \vec{x}_k|\mu, \sigma, \vec{a}, b) = \Pr\left(\hat{\vec{x}}_1, \dots, \hat{\vec{x}}_k\middle|\mu, \sigma, \vec{a}, b\right) * \Pr(e_1, \dots, e_k|\mu, \sigma, \vec{a}, b)$$

$$= \Pr(e1, \dots, e_k|\mu, \sigma, \vec{a}, b) = \prod_{k=1}^{N} \Pr(e_k|\sigma, \vec{a}, b) \qquad (10)$$

Now subbing in the normal distribution:

$$\arg\max_{\sigma,\vec{a},b} \Pr(\vec{x}_1, \dots, \vec{x}_k|\sigma, \vec{a}, b) = \prod_{k=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{e_k^2}{2\sigma^2}} \quad (11)$$

Maximizing the probability of something occurring is equivalent to minimizing the negative log likelihood. **Equation 12** shows the negative log likelihood with some logarithm rules applied:

$$\mathcal{L}(\sigma, \vec{a}, b) = -\ln(1) + N\ln\left(\sigma\sqrt{2\pi}\right) + \sum_{k=1}^{N} \ln(e) * \left(\frac{e_k^2}{2\sigma^2}\right) = Nln\left(\sigma\sqrt{2\pi}\right) + \sum_{k=1}^{N} \frac{e_k^2}{2\sigma^2} \quad (12)$$

Now, subbing in e_k, we get:

$$\mathcal{L}(\sigma, \vec{a}, b) = Nln\left(\sigma\sqrt{2\pi}\right) + \sum_{k=1}^{N} \frac{\left(\vec{a}^T\vec{x}_k + \vec{b} + \vec{x}_k^T\vec{x}_k\right)^2}{2\sigma^2} \quad (13)$$

Here, it is evident by comparing this formulation to the one in the notes that the Maximum Likelihood (ML) estimation for this problem is not Least Squares (LS) because the variance is inversely dependent on b^2 which means its also inversely dependent on r^2.

The LS model parameters can be derived by making the variance term constant and differentiating the likelihood function with respect to $\vec{a}$ and b, respectively. These equations can be seen in **Equation 14**, and **15**, respectively. These equations should be solved iteratively to get the least squares approximation for this model.

$$\frac{\partial \mathcal{L}}{\partial \vec{a}} \rightarrow \vec{a}_{LS} = -\sum_{K=1}^{N} \frac{b + \vec{x}_k^T \vec{x}_k}{\vec{x}_k^T} \qquad (14)$$

$$\frac{\partial \mathcal{L}}{\partial b} \rightarrow b_{LS} = \sum_{k=1}^{N} -\vec{x}_k^T \vec{x}_k - \vec{a}^T \vec{x}_k \qquad (15)$$

2.3 Circle Proposals

The circle proposals were selected using the following approach:

1. *Randomly choose an edgel ($\vec{x}_a$) where $\vec{x}_a \in [1, K]$*
2. *Then check every 10 indices from a for a point ($\vec{x}_b$) within a defined distance.* Every 10 indices were checked so the data points could be quickly scavenged for data points. The minimum threshold was set to 5 pixels to avoid points next to each other that have nearly parallel lines. The maximum threshold was set to 40 pixels to try and get only points on the same circle. The distance was calculated using the Euclidean distance formula. These numbers were chosen via trial and error.
3. *Given two points on a circle ($\vec{x}_a, \vec{x}_b$), and their normals ($\vec{n}_a, \vec{n}_b$), they are on the same circle if they share a common r and $\vec{x}_c$.* The equations below were used to get the radius in both the i and j directions:

$$\begin{bmatrix} x_{c,i} \\ x_{c,j} \end{bmatrix} = \begin{bmatrix} x_{a,i} \\ x_{a,j} \end{bmatrix} + r \begin{bmatrix} n_{a,i} \\ n_{a,j} \end{bmatrix} = \begin{bmatrix} x_{b,i} \\ x_{b,j} \end{bmatrix} + r \begin{bmatrix} n_{b,i} \\ n_{b,j} \end{bmatrix} \rightarrow \begin{matrix} r_i = \dfrac{x_{b,i} - x_{a,i}}{n_{a,i} - n_{b,i}} \\ r_j = \dfrac{x_{b,j} - x_{a,j}}{n_{a,j} - n_{b,j}} \end{matrix} \qquad (16)$$

4. *Throw away guess if the resultant answers are too far apart ($abs(r_i - r_j) > r\_tol$) or if the radius falls outside the radius acceptance band ($r_{min} < r_{i,j} < r_{max}$).* The tolerance was included since their radius might not be the same due to noise. The constant r_{max} was set to half the x-range and y-range for the given set since the largest possible circle should have a radius equal to half the window. The constant r_min is picked such that really small circles are eliminated.
5. r_i and r_j *are averaged to get* r_{avg}.
6. r_{avg} *is used to calculate the centers for both the circles using the same formula seen in* ***Equation 16****.*
7. *Finally, the center point distances for both points are computed to ensure that it is within a set tolerance to account for noise*
8. *Finally, the averaged circle radii and centers are returned*

This approach was used because it was a quick and effective way to generate proposals. It could find at least one circle with very few iterations. It also provided a good estimate for the circle radii and centers. This approach was chosen for its simplicity. The most intensively formula used is this method is the distance equation.

Another way to generate a proposal would have been to generate random samples than evaluate random samples using the Hough voting method. Finally, the samples with high enough votes are selected. This approach is much more simple and only involved basic geometry.

2.4 Circle Selection

The main circle of all the proposals was selected by applying the robust estimation algorithm for 1 iteration to all the proposals. The circle with the lowest difference between the model parameters $\vec{\theta}$ vs $\vec{\theta}^{i-1}$ was chosen to be the best circle. The derivation for the robust estimation algorithm can be seen in **Section 2.5**. This method required only n runs, where n is equal to the number of guess circles. It was mainly chosen because it wasn't too computationally expensive

and it provided the circle that is likely to be most quickly fit during robust estimation. Time lost here can be saved in the later step since the ideal circle for robust estimation was chosen.

Another method that could have been used here instead is the Hough voting method. The proposed circles could have been evaluated using the voting method to determine which circle had the highest votes. That circle would then be chosen as the best circle.

2.5 Robust Fitting

The goal is to minimize the objective ($\mathcal{O}$) function with respect to the model parameters ($\vec{a}, b$). We let $\vec{\theta}$ represent the model parameters: $\begin{bmatrix}\vec{a}\\ b\end{bmatrix}$. The objective function can be seen in **Equation 17**. The objective function is minimized by setting the derivative equal to 0. The derivative is found using chain rule. The derivation can be seen in **Equation 18**.

$$\mathcal{O}(\vec{\theta}) = \sum_{k=1}^{N} \rho(e_k) \quad (17)$$

$$\frac{\partial \mathcal{O}}{\partial \vec{\theta}} = \sum_{k=1}^{N} 2w_k e_k \frac{\partial e_k}{\partial \vec{\theta}} = \sum_{k=1}^{N} \frac{\partial \rho}{\partial e_k}\frac{\partial e_k}{\partial \vec{\theta}} \quad (18)$$

From these two equations, it is seen that:

$$w_k = \frac{1}{e_k}\frac{\partial \rho}{\partial e_k} \quad (19)$$

First, the GM, $\rho(e_k, \sigma_G)$ equation (*Equation 1* of the manual), is differentiated with respect to $\vec{\theta}$ using the chain rule with: $u = e_k^2 \rightarrow \frac{du}{de_k} = 2e_k$

$$\frac{\partial \rho}{\partial e_k} = \frac{\partial \rho}{\partial u}\frac{du}{de_k} = \frac{\sigma_G^2}{(u+\sigma_G^2)^2} \cdot (2e_k) = \frac{2e_k\sigma_G^2}{(e_k^2+\sigma_G^2)^2} \quad (20)$$

Subbing **Equation 20** into **Equation 19** yields the weight equation (w_k):

$$w_k = \frac{2\sigma_G^2}{(e_k^2 + \sigma_G^2)^2} \qquad (21)$$

Finally, the e_k formula (*Equation 2* of the manual) can be rewritten in terms of $\vec{\theta}$:

$$e_k = \vec{a}^T\vec{x}_k + b + \vec{x}_k^T\vec{x}_k = \vec{x}_k^T\vec{a} + b + \vec{x}_k^T\vec{x}_k = [\vec{x}_k^T \quad 1]\begin{bmatrix}\vec{a}\\ b\end{bmatrix} + \vec{x}_k^T\vec{x}_k = [\vec{x}_k^T \quad 1]\vec{\theta} + \vec{x}_k^T\vec{x}_k \quad (22)$$

Now, differentiating e_k with respect to $\vec{\theta}$ yields:

$$\frac{\partial e_k}{\partial \vec{\theta}} = [\vec{x}_k^T \quad 1] \qquad (23)$$

Now substituting **Equation 21,** and **Equation 23** yields:

$$\frac{\partial \mathcal{O}}{\partial \vec{\theta}} = \sum_{k=1}^{N} 2w_k\big[\,[\vec{x}_k^T \quad 1]\vec{\theta} + \vec{x}_k^T\vec{x}_k\big][\vec{x}_k^T \quad 1] \qquad (24)$$

Now setting the derivative equal to 0 and separating it yields:

$$\sum_{k=1}^{N} w_k[x_k^T \quad 1]\vec{\theta}[x_k^T \quad 1] = -\sum_{k=1}^{N} w_k\vec{x}_k^T\vec{x}_k[\vec{x}_k^T \quad 1]$$

$$\sum_{k=1}^{N}[\vec{x}_k^T \quad 1]^T w_k[\vec{x}_k^T \quad 1]\vec{\theta} = -\sum_{k=1}^{N}[\vec{x}_k^T \quad 1]^T w_k\vec{x}_k^T\vec{x}_k \qquad (25)$$

Now converting this to matrix notation yields:

$$X^TWX\vec{\theta} = -X^TWX_{sq} \quad (26)$$

Where X is a matrix of size Nx3, W is a square matrix of size NxN with w_k along its diagonal, X_{sq} is a matrix of size Nx1. N represents the number of edgels. The matrix definitions can be seen below:

$$X = \begin{bmatrix} \vec{x}_1^T & 1 \\ \vec{x}_2^T & 1 \\ \vdots & \vdots \\ \vec{x}_k^T & 1 \end{bmatrix}, \quad W = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & w_k \end{bmatrix}, \quad X_{sq} = \begin{bmatrix} ||\vec{x}_1||^2 \\ ||\vec{x}_2||^2 \\ \vdots \\ ||\vec{x}_k||^2 \end{bmatrix}, \quad \vec{\theta} = \begin{bmatrix} a_x \\ a_y \\ b \end{bmatrix} \qquad (27)$$

After solving for $\vec{\theta}$ the model parameters can be found using the following formulae:

$$\vec{x}_c = -\frac{\vec{a}}{2} \rightarrow \begin{bmatrix} x_{c,x} \\ x_{c,y} \end{bmatrix} = -\frac{1}{2}\begin{bmatrix} a_x \\ a_y \end{bmatrix} \qquad (28)$$

$$r = \sqrt{\vec{x}_c^T \vec{x}_c - b} \qquad (29)$$

2.5.1 IRLS Algorithm

The IRLS algorithm works as follow:

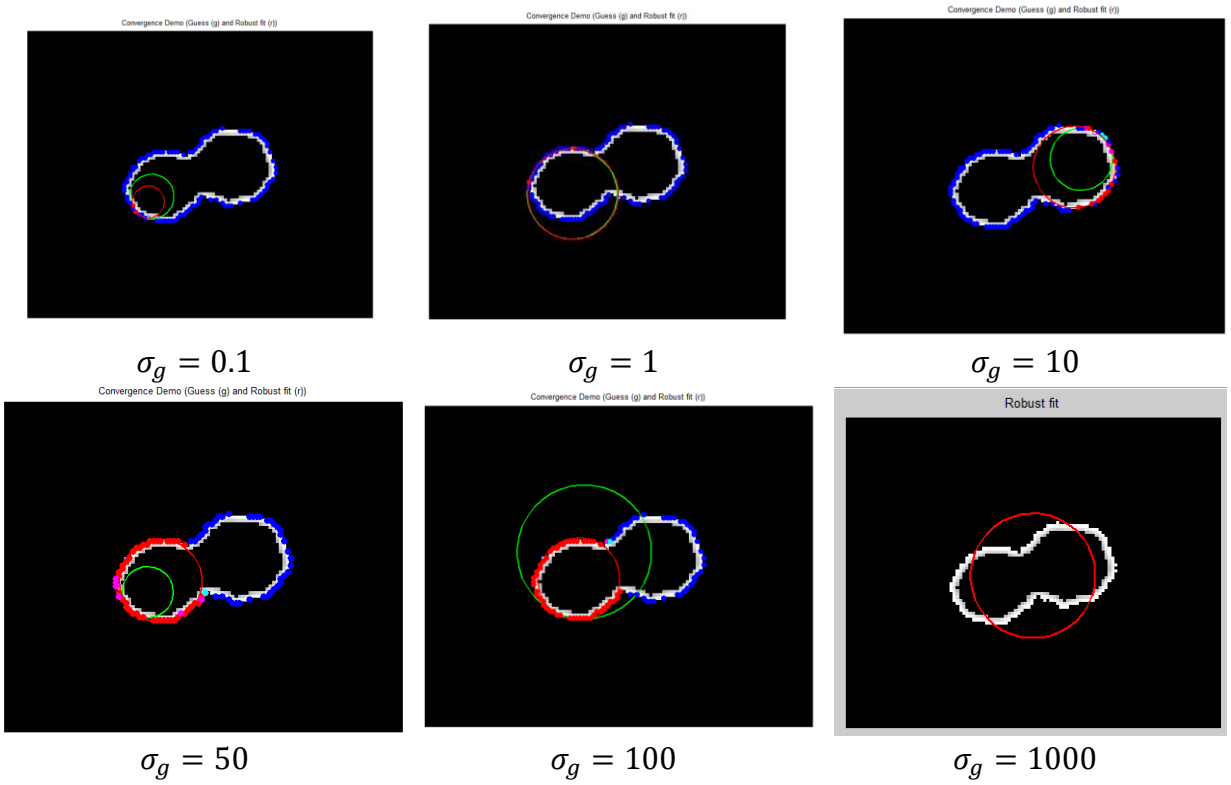
1. First the best circle chosen in **Section 2.4** is used as the initial guess:
$$\vec{\theta}^0 = \begin{bmatrix} \vec{a}^0 \\ b^0 \end{bmatrix}$$
2. The weights are computed via **Equation 21**
$$w_k^i = w\left(x_k^T \vec{\theta}^{i-1}\right)$$
3. Solve **Equation 26** to get line parameters $\vec{\theta}^i$
4. Repeat until convergence
$$\left(\vec{\theta}^i - \vec{\theta}^{i-1}\right) \sim 0$$

2.5.2 Selecting the GM Estimator

One way that could be used to select σ_g is to work on a small subset of your main set. You can test various σ_g values to determine which one would work the best.

The σ_g term controls the weight applied to each edgel. For low σ_g, the weight placed on every edgel is very low, so it fails to accurately detect a cell. For high σ_g, the weight placed on every edgel is very high, so it mistakenly models two cells as one. Increasing σ_g smooths the objective function, so there are fewer local minimum and, therefore, fewer solutions to the problem. No

circle was detected for $\sigma_g = 0.1$ & 1. One of the circles was detected for $\sigma_g = 10$. Two circles were correctly detected for $\sigma_g = 50$ & 100. Both the circles were detected as one circle for $\sigma_g = 1000$. In this case, the best σ_g value would be around 50 to 100.



$\sigma_g = 0.1$ $\sigma_g = 1$ $\sigma_g = 10$

$\sigma_g = 50$ $\sigma_g = 100$ $\sigma_g = 1000$

Figure 9: Sample robust estimation with various σ_g values

2.6 Model Update

To determine if a circle should be kept or not, the first metric that is used is a threshold metric. A perfect circle will have edgels all around its circumference with a weight of 1 so the sum of the weights of the circle would equal to $2\pi r$ but, since there is noise, this is not the case. In this case, circles could be overlapped by 1 or more circles so not all circles have edgels all along their circumference. Also, a circle could be cropped by the edges of the screen. A circle could have about half its edgels as seen in **Figure 10**. Also, as seen in **Figure 10**, a circle could be imperfect so some edgels can have weights equal to less than 1. To account for both these cases, the threshold was set such that a circle must have a sum of edgel weights greater than 30% of the total threshold as seen in **Equation 30**.

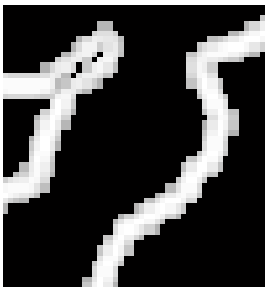


Figure 10: Example of imperfect circle missing lots of edgels

$$isCircleThreshold = 0.30 * (2\pi r) \qquad (30)$$

After the first circle is found, edgels are removed from the dataset. This means that entire sides of a circle could be removed. To account for this, we must first check to see if there are intersections between the current circle and circles already removed from the dataset. A circle intersects another circle if the distance between their centres is less than the sum of their radii.

For the second circle, onwards, edgels are removed from the dataset, so entire sides of a circle could be removed. In the case of no intersection, **Equation 30** will be used again. In the case of intersection(s), a modified version of **Equation 30** is used to account for the removal. The angle of intersection is found and that angle is removed from the max potential value, i.e., the max weight this circle can have is equal to $(2\pi - \phi)r$ where ϕ is the angle of intersection. The angle of intersection is the angle between the lines formed by the radius and the points of the intersection of the circles. The modified threshold formula can be seen in **Equation 31**. This equation is given a lower threshold percentage because it was found through testing that it performed better with a lower threshold.

$$isCircleThresholdwIntersection = 0.25 * (2\pi - \phi) * r \qquad (31)$$

2.7 Brief Evaluation

The results of the robust estimation code can be seen in **Figure 11**.

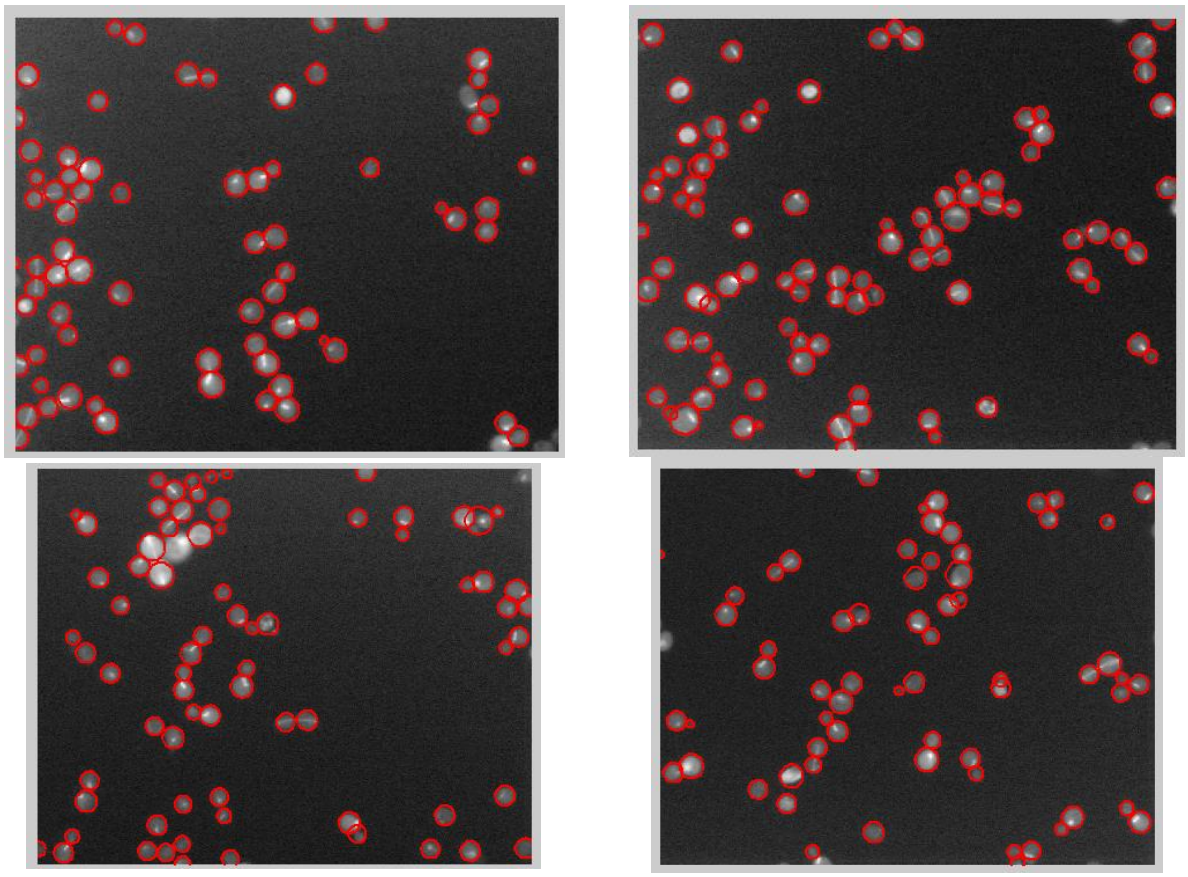


Figure 11: Results from robust estimation

The robust estimator worked well for the given data set. It captures almost all the circles including some of the expected outliers. The algorithm seemed to suffer for blocks of cells in a cluster. There were two major outliers among the rest that stood out: (1) overlapping circles (**Figure 12**), and (2) missing circles within a cluster (**Figure 13**).

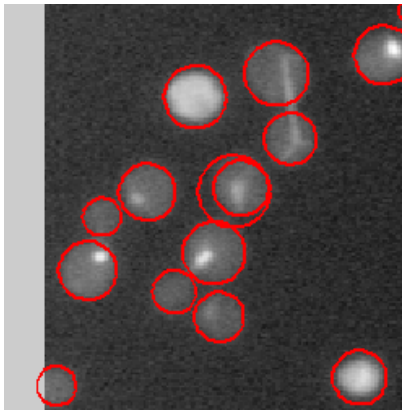


Figure 12: Overlapping circles

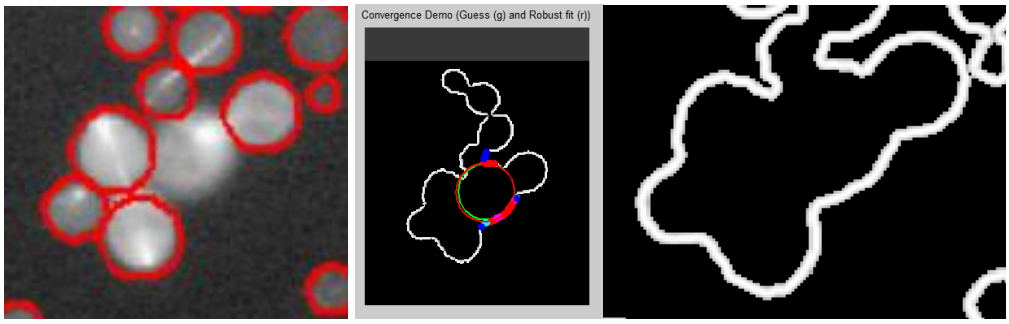


Figure 13: Missing circle in cluster

2.7.1 Overlapping Circles

Two cells should not overlap each other like this. One cell in this image is contained within the other cell. A possible cause of this is a smaller circle got fit with enough weightage in a large circle. Enough edgels still exists after the small circles edgels were removed to fix this, there are two possible method. The simple approach is to add code to the 'isGoodCircle.m' code to not place circles that more than halfway into an already placed cell. The better method is to add back the edgels removed by both circles and draw the larger circle first. Then fit that circle and remove the edgels associated with it. Then an attempt can be made to fit the smaller circle, and it can be placed or eliminated depending on how it fairs within 'isGoodCircle.m'.

2.7.2 Missing Circle in Cluster

This is another common occurrence that generally happens because of the how a good circle is selected. The circle gets found early on with 'getProposals.m' but it is eliminated by 'bestProposal.m' because every other circle found usually gets better results from GM. This is because it has very few edgels on the cluster as evident in **Figure 12**. This can be fixed via two potential methods. The first method would be to increase the number of rounds. With the current algorithm, this circle is likely to be selected last so it has a better chance of being found and fit in later rounds of guessing and fitting. The second method would be to analyze clusters with enough remaining edgels that are near each other after robust estimation is complete. A final circle(s) can be placed to include the remaining edgels that are close together, and the attempt can be made to fit the circle on these edgels with possible. Invalid results would then get rejected via 'isGoodCircle.m'.

3.0 References

[1] https://en.wikipedia.org/wiki/Sum_of_normally_distributed_random_variables

[2] https://en.wikipedia.org/wiki/Chi-squared_distribution

[3] https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence