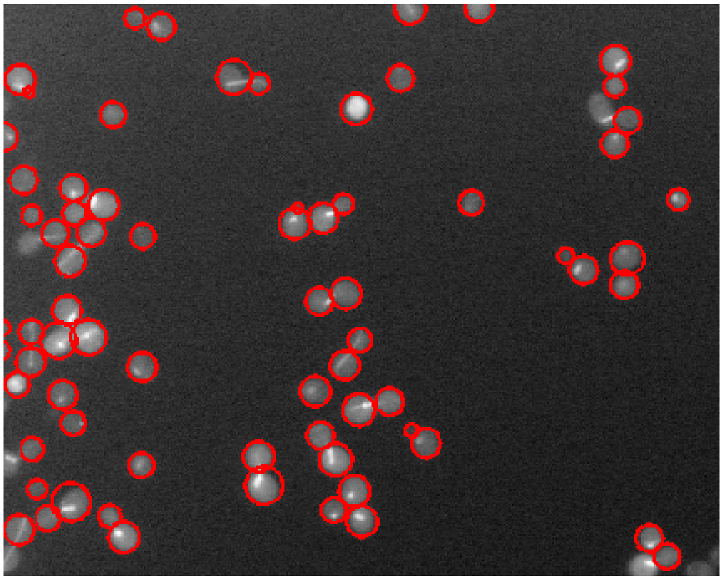
University of Toronto

Robust Estimation

CSC2503 Fall 2014
Foundations of Computer Vision

Alexander Hong
(c4hongal 997584706)



October 29, 2014

Contents

1 Noise Model **2**

2 Least-Squares Estimator **2**

3 Circle Proposals **3**

4 Circle Selection **4**

5 Robust Fitting **4**

5.1 Minimization of Objective Function 4

5.2 IRLS Algorithm 6

6 Model Update **7**

7 Evaluation **8**

1 Noise Model

Assuming $\vec{x}_k = \hat{\vec{x}}_k + \vec{m}$, we have error, e_k as

$$e_k = \vec{a}^T \vec{x}_k + b + \vec{x}_k^T \vec{x}_k \tag{1}$$
$$e_k = \vec{a}^T \hat{\vec{x}}_k + \vec{a}^T \vec{m} + \hat{\vec{x}}_k^T \hat{\vec{x}}_k + 2\hat{\vec{x}}_k^T \vec{m} + b + \vec{m}^T \vec{m} \tag{2}$$

The distribution of the errors e_k is a χ^2 (chi-squared) distribution with 2 degrees of freedom. The noise model shows that the circle centre, $\vec{x}_c$, is modified by the $\vec{m} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$ noise term (isotropic Gaussian) in $\vec{a}^T \vec{m}$. The radius between the circle centre and true circumference is also modified by the noise term in $2\hat{\vec{x}}_k^T \vec{m}$.

For further analysis as to why the distribution is χ^2, since $\vec{x}_k = \hat{\vec{x}}_k + \vec{m}$, the distribution of $\vec{x}_k$ is also an isotropic Gaussian. Thus, $\vec{d}_k = \vec{x}_k - \vec{x}_c$ is also an isotropic Gaussian with distribution $\vec{d}_k \sim \mathcal{N}(\hat{\vec{x}}_k - \vec{x}_c, \sigma^2 \mathbf{I})$. Since $\vec{d}_k = [d_{k1}\ d_{k2}]^T$, $\vec{x}_k = [x_{k1}\ x_{k2}]^T$, and $\vec{x}_c = [x_{c1}\ x_{c2}]^T$, we have $d_{k1} \sim \mathcal{N}(\hat{x}_{k1} - x_{c1}, \sigma^2)$ and $d_{k2} \sim \mathcal{N}(\hat{x}_{k2} - x_{c2}, \sigma^2)$. With $\|\vec{x}_k - \vec{x}_c\|^2 = \|\vec{d}_k\|^2 = d_{k1}^2 + d_{k2}^2$, we have the definition of χ^2 distribution with 2 degrees of freedom (i.e. a distribution of a sum of the squares of 2 independent standard normal random variables).

2 Least-Squares Estimator

Maximizing the likelihood of points on the true circumference, $P(\vec{x}_1, \vec{x}_2, ..., \vec{x}_K | a, b)$, is the same as minimizing likelihood of errors in $\vec{x}_k$. Suppose $\sigma_i^2 = \sigma_j^2\ \forall\ i, j \in \{1, 2, 3, ..., K\}$, then $e_k, k \in \{1, 2, 3, ..., K\}$ has the same Gaussian noise with variance σ^2. Since $e_k \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$,

$$\begin{aligned}
\text{Minimize: } P(e_1, e_2, ..., e_k | \mu = 0, \sigma^2) &= \prod_{k=1}^{K} P(e_k | 0, \sigma^2) \\
&= \prod_{k=1}^{K} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-e_k^2}{2\sigma^2}} \qquad (3) \\
\implies \text{Maximize: } -\ln P(e_1, e_2, ..., e_k | 0, \sigma^2) &= -\ln \prod_{k=1}^{K} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-e_k^2}{2\sigma^2}} \\
&= -\sum_{k=1}^{K} \ln \left(\frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-e_k^2}{2\sigma^2}} \right) \\
&= \frac{K}{2} \ln (2\pi) + \sum_{k=1}^{K} \left(\frac{e_k^2}{2\sigma^2} + \ln \sigma \right) \qquad (4)
\end{aligned}$$

Maximizing $-\ln P(e_1, e_2, ..., e_K | 0, \sigma^2)$ or minimizing $-\ln P(\vec{x}_1, \vec{x}_2, ..., \vec{x}_K | a, b)$ is the same as minimizing the sum of squares error function, $\sum e_k^2$, which is the least-squares estimator. To find a reasonable choice for the variance, σ^2, of the Gaussian noise, differentiate the negative log likelihood with respect to σ^2 to find the variance to maximize $-\ln P(e_1, e_2, ..., e_k|0, \sigma^2)$:

$$
\begin{aligned}
-\frac{\partial \ln P(e_1, e_2, ..., e_k|0, \sigma^2)}{\partial \sigma^2} &= 0 && (5)\\
\implies \sum_{k=1}^{K} \frac{\partial}{\partial \sigma^2}\left(\frac{e_k^2}{2\sigma^2} + \frac{1}{2}\ln \sigma^2\right) &= 0 && (6)\\
\implies \sum_{k=1}^{K}\left(\frac{-e_k^2}{2\sigma^4} + \frac{1}{2\sigma^2}\right) &= 0 \\
\implies \sum_{k=1}^{K}\left(\frac{-e_k^2}{2\sigma^4}\right) + \frac{K}{2\sigma^2} &= 0 && (7)\\
\implies \sigma_{ML}^2 &= \frac{1}{K}\sum_{k=1}^{K}(e_{k,ML})^2 && (8)
\end{aligned}
$$

where $e_{k,ML} = \vec{a}_{ML}^T\vec{x}_k + b_{ML} + \vec{x}_k^T\vec{x}_k$, is the maximized likelihood error.

3 Circle Proposals

To generate circle proposals, the following approach was taken:

1. Randomly choose 2 pair of edgels, $\vec{x}_\alpha$ and $\vec{x}_\beta$, where $\alpha, \beta \in [1, K], \alpha \neq \beta$.
2. Check if $\vec{n}_\alpha \nparallel \vec{n}_\beta$ to avoid selecting parallel lines. Parallel lines have no intersection point to propose as $\vec{x}_c$. If parallel lines have been selected, go back to Step 1.
3. Find the intersection point, $\vec{x}_c$, between $(\vec{x}_\alpha, \vec{n}_\alpha)$ and $(\vec{x}_\beta, \vec{n}_\beta)$. Check if the intersection point lies in the direction of the two normals. If the intersection point is not within the direction of the two normals, the proposed $\vec{x}_c$ is outside the circle; go back to Step 1.
4. Calculate the two distances (radii), r_α and r_β, between the intersection point, $\vec{x}_c$, and the edgel points, $\vec{x}_\alpha$ and $\vec{x}_\beta$. Check if the radius values are near each other (e.g. check if the their ratios ≤ 1.3). In addition, check if the radii are within a threshold, which is selected to be a reasonable upper limit cell size, in order to avoid proposals with too large of a radius. Moreover, check if the intersection points, $\vec{x}_c$, is located within the bounding box. If radius values are not near each other, does not satisfy the radius threshold, or the intersection point is out of bound, go back to Step 1.
5. Propose $\vec{x}_c$ as the intersection point between $(\vec{x}_\alpha, \vec{n}_\alpha)$ and $(\vec{x}_\beta, \vec{n}_\beta)$, and r as the average of r_α and r_β that have been obtained from previous steps.

This approach was used because it was a quick and effective way to generate $P, 0 \leq P \leq$ `numGuesses` of proposals. The algorithm's time complexity is $\mathcal{O}(n^2)$ and space complexity is $\mathcal{O}(n)$ to generate $P, 0 \leq P \leq$ `numGuesses`, proposals, so the algorithm is not computationally intensive. Choosing the radii to have values near each other is a simple method for choosing good proposals for the centre position of the circle as the radius is constant in a circle. The algorithm also filters out common bad proposals by checking for parallel lines, and the position of the centre of the circle with respect to the selected edgel pair. This approach was selected over other approaches due to its simplicity in both theory and implementation in code. In addition, one can get a fairly good set of proposals for circle selection, which is explained in the next section.

Another way to generate proposal, which is more computationally intensive, is to map out all the intersection points and find a point with high number of neighbours (other intersection points). In order to do this, one can identify an arbitrary radius for each intersection point as the viewing radius, and other intersection points within this circle are consider as neighbours. This finds a $\vec{x}_c$ within clusters. This method was not implemented because the approach used was simpler and worked fairly well. Usually, there is a tradeoff between performance and time. However, proposals do not have to be robust circle estimates.

4 Circle Selection

The best circle was selected using from the proposals generated in Section 3. In this case, "best" means the circle with the least-squared errors. The errors of each edgel for each proposal were calculated using the following equation:

$$e_k = \vec{a}^T \vec{x}_k + b + \vec{x}_k^T \vec{x}_k, k \in [1, K] \tag{9}$$

where $\vec{a} \equiv -2\vec{x}_c$ and $b \equiv \vec{x}_c^T \vec{x}_c - r^2$. The best circle was selected by choosing the proposal with the least-squared error (i.e. calculate $\sum e_k^2$ for each proposal and choose the proposal that has the minimum error as the best circle). This approach has many advantages over other approaches as it is straightforward and effective. Least-squared errors is a common approach to find a line of best fit to a set of data points. Likewise, this method is to find the proposal that fits best to a set of edgels.

5 Robust Fitting

5.1 Minimization of Objective Function

The goal is to minimize the objective function for circle parameters $(\vec{a}, b)$.

$$\begin{aligned} \mathcal{O}(\vec{a}, b) &= \sum_k \rho(e_k(\vec{a}, b), \sigma) && (10) \\ \rho(e_k, \sigma) &= \frac{e_k^2}{\sigma^2 + e_k^2} && (11) \\ & && (12) \end{aligned}$$

where e_k is defined in Equation (1). Let $\mathbf{p} \equiv \begin{bmatrix} \vec{a} \\ b \end{bmatrix}$ and rewrite Equation (1) as:

$$\begin{aligned} e_k &= \begin{bmatrix} \vec{x}_k & 1 \end{bmatrix} \begin{bmatrix} \vec{a} \\ b \end{bmatrix} + \vec{x}_k^T \vec{x}_k \\ &= \begin{bmatrix} \vec{x}_k & 1 \end{bmatrix} \mathbf{p} + \vec{x}_k^T \vec{x}_k & (13) \\ \implies \frac{\partial e_k}{\partial \mathbf{p}} &= \begin{bmatrix} \vec{x}_k & 1 \end{bmatrix} & (14) \end{aligned}$$

To minimize $\mathcal{O}$, differentiate it with respect to the circle parameters, $\mathbf{p}$, and set it to zero:

$$\frac{\partial \mathcal{O}}{\partial \mathbf{p}} = \frac{\partial \mathcal{O}}{\partial e_k^2} \frac{\partial e_k^2}{\partial \mathbf{p}} \tag{15}$$

To find $\frac{\partial \mathcal{O}}{\partial e_k^2}$:

$$\begin{aligned} \frac{\partial \mathcal{O}}{\partial e_k^2} &= \frac{\partial \sum_k \rho}{\partial e_k^2} \\ &= \sum_k \frac{\sigma^2}{\sigma^2 + e_k^2} & (16) \\ w_k &\equiv \frac{1}{e_k} \frac{\partial \rho}{\partial e_k} & (17) \\ &= \frac{1}{e_k} \frac{\partial \rho}{\partial e_k^2} \frac{\partial e_k^2}{\partial e_k} \\ &= 2 \frac{\partial \rho}{\partial e_k^2} \\ &= \frac{2\sigma^2}{\sigma^2 + e_k^2} & (18) \\ \implies \frac{\partial \mathcal{O}}{\partial e_k^2} &= 2 \sum_k w_k & (19) \end{aligned}$$

To find $\frac{\partial e_k^2}{\partial \mathbf{p}}$:

$$\begin{aligned} \frac{\partial e_k^2}{\partial \mathbf{p}} &= 2 e_k \frac{\partial e_k}{\partial \mathbf{p}} & (20) \\ &= 2 e_k \begin{bmatrix} \vec{x}_k & 1 \end{bmatrix} & (21) \end{aligned}$$

Thus, Equation (15) becomes:

$$\begin{aligned} \frac{\partial \mathcal{O}}{\partial \mathbf{p}} &= 2 \sum_k w_k \left(2 e_k \begin{bmatrix} \vec{x}_k & 1 \end{bmatrix} \right) \\ &= 4 \sum_k w_k \left(\begin{bmatrix} \vec{x}_k & 1 \end{bmatrix} \mathbf{p} + \vec{x}_k^T \vec{x}_k \right) \begin{bmatrix} \vec{x}_k & 1 \end{bmatrix} \\ &= 4 \sum_k w_k \begin{bmatrix} \vec{x}_k & 1 \end{bmatrix} \mathbf{p} \begin{bmatrix} \vec{x}_k & 1 \end{bmatrix} + 4 \sum_k w_k \vec{x}_k^T \vec{x}_k \begin{bmatrix} \vec{x}_k & 1 \end{bmatrix} & (22) \end{aligned}$$

Set $\frac{\partial \mathcal{O}}{\partial \mathbf{p}}$ to zero:

$$\frac{\partial \mathcal{O}}{\partial \mathbf{p}} = 0 \tag{23}$$

$$\Longrightarrow \sum_k w_k \begin{bmatrix} \vec{x}_k & 1 \end{bmatrix} \mathbf{p} \begin{bmatrix} \vec{x}_k & 1 \end{bmatrix} = -\sum_k w_k \vec{x}_k^T \vec{x}_k \begin{bmatrix} \vec{x}_k & 1 \end{bmatrix} \tag{24}$$

Define the following variables to rewrite Equation (24) in matrix notation:

$$\mathbf{X}^T \mathbf{W} \mathbf{X} \mathbf{p} = -\mathbf{X}^T \mathbf{W} \mathbf{X}_{SQR} \tag{25}$$

where $\mathbf{W} \subseteq \mathbb{R}^{K \times K}$ is square matrix containing all $w_k, k \in [1, K]$ on its diagonal, $\mathbf{X} \subseteq \mathbb{R}^{K \times 3}$, $\mathbf{X}_{SQR} \subseteq \mathbb{R}^{K \times 1}$, and K represents the number of edgels. The matrices are defined as the following:

$$\mathbf{W} = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_K \end{bmatrix} \tag{26}$$

$$\mathbf{X} = \begin{bmatrix} \vec{x}_1^T & 1 \\ \vec{x}_2^T & 1 \\ \vdots & \vdots \\ \vec{x}_K^T & 1 \end{bmatrix} \tag{27}$$

$$\mathbf{X}_{SQR} = \begin{bmatrix} \|\vec{x}_1\|^2 \\ \|\vec{x}_2\|^2 \\ \vdots \\ \|\vec{x}_K\|^2 \end{bmatrix} \tag{28}$$

$$\tag{29}$$

Solving for $\mathbf{p} \subseteq \mathbb{R}^{3 \times 1}$ yield values for both $\vec{a}$ and b, where the first two rows of $\mathbf{p}$ are the components of $\vec{a}$, and the last row of $\mathbf{p}$ is the value for b. After solving for $\vec{a}$ and b, the center of the robust circle and its radius are given by:

$$\vec{x}_c = -\frac{\vec{a}}{2} \tag{30}$$

$$r = \sqrt{\vec{x}_c^T \vec{x}_c - b} \tag{31}$$

5.2 IRLS Algorithm

The algorithm is as follows:

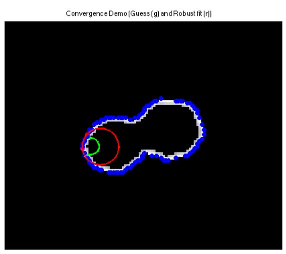
1. Starting with the initial guess generated from Section 3, compute the weight matrix, $\mathbf{W}$. In order to do this, the errors of each proposal must be calculated.

2. Solve the Weighted Least-Squares (WLS) problem to get circle parameters. This is detailed in Equation (25).

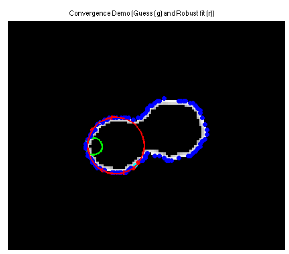
3. Check if the difference of the previous and current sum of squares of the circle parameters converged within a convergence threshold (i.e. check for changes in circle parameters every iteration). If it has not converge, reiterate step 1 with the generated circle parameters as the guess.

A suitable way for choosing σ_g is to start with σ_g^2 to be the variance of e_k, and try the robust estimator on a small sample. Next, calculate the variance of the e_k with the best fitting, and use it as σ_g for the next iteration.

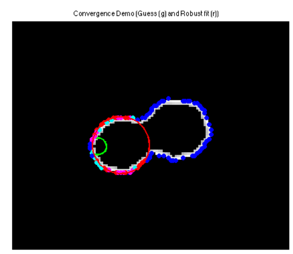
The following show how different values for σ_g affect the robust estimator. If σ_g is too small, the robust estimator fails due to lower weight values in each edgel. If σ_g is too large, the robust estimator will also fail because the high weight of the edgels will affect the circle fitting. For instance, in Figure 1, circle fitting with $\sigma_g = 500$ mistakens the two cells as one cell due the the high weight of the edgels.



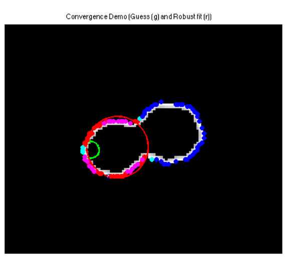
(a) $\sigma_g = 0.1$



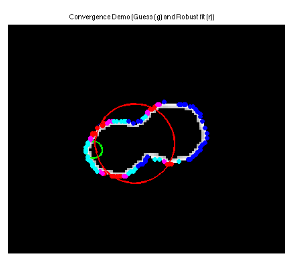
(b) $\sigma_g = 1$



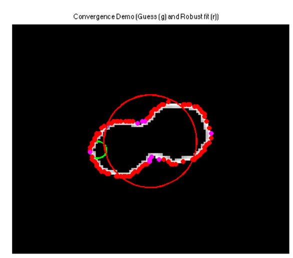
(c) $\sigma_g = 10$



(d) $\sigma_g = 50$



(e) $\sigma_g = 100$



(f) $\sigma_g = 500$

Figure 1: Circle fitting under different σ_g

6 Model Update

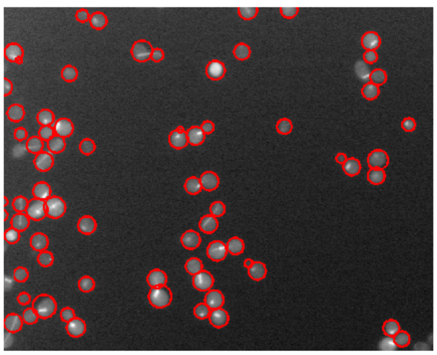
To decide if a circle should be kept, all intersected circles were found and compared with a threshold applied to the weights. First, intersected circles were found by investigating all circle estimates' radii and checking if the sum of the radii is less than the distance from one circle's center to another.

If a circle has no apparent intersection, the sum of weights of the circle was compared to a threshold ($T_1 = 0.3(2\pi r)$). If the sum of weights of the circle was greater, then it was a good circle; otherwise, it was bad. T_1 threshold was decided based on the ideal case where every edgel will contribute a maximum weight of 1.

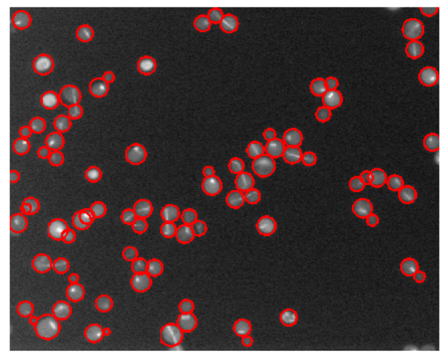
For circles that are intersecting, the angle of intersection, α, was calculated (i.e. the angle between the lines formed by the radius point and the circle intersection point of another circle). If there were more than 1 intersecting circle, the sum of the intersection angles was recorded instead. Finally, the sum of weights of the circle was compared to the threshold ($T_2 = 0.25(2\pi - \alpha)$). If the sum was greater, then it was a good circle; otherwise, it was bad. This method was used to better detect intersecting cells.

7 Evaluation

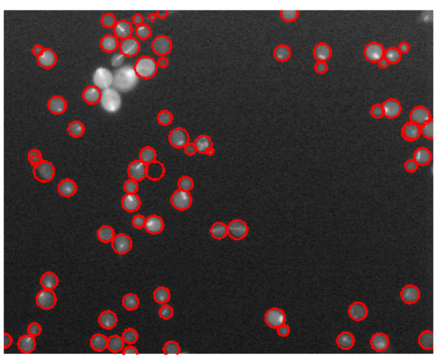
Figure 2 illustrates the results for the images provided.



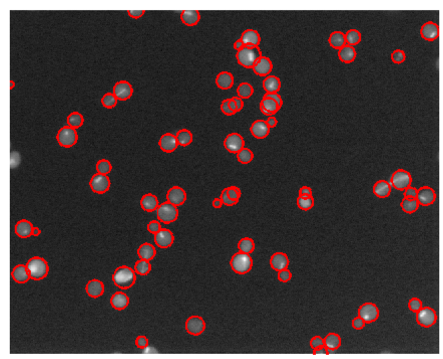
(a) Cell Image 1



(b) Cell Image 2



(c) Cell Image 3

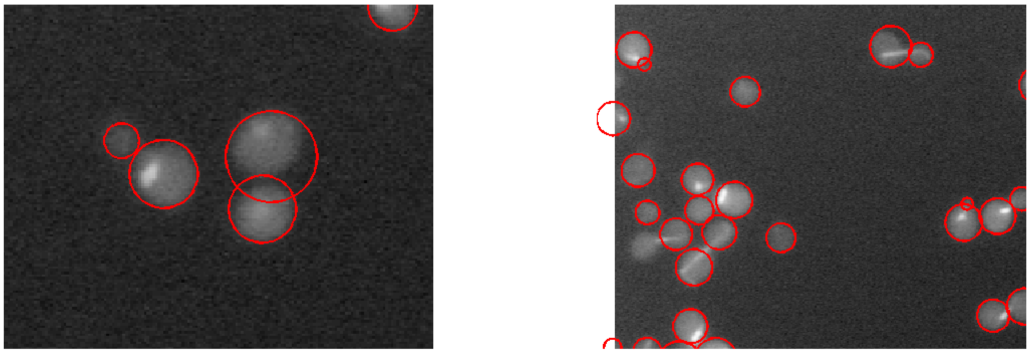


(d) Cell Image 4

Figure 2: Cell image with circle fitting

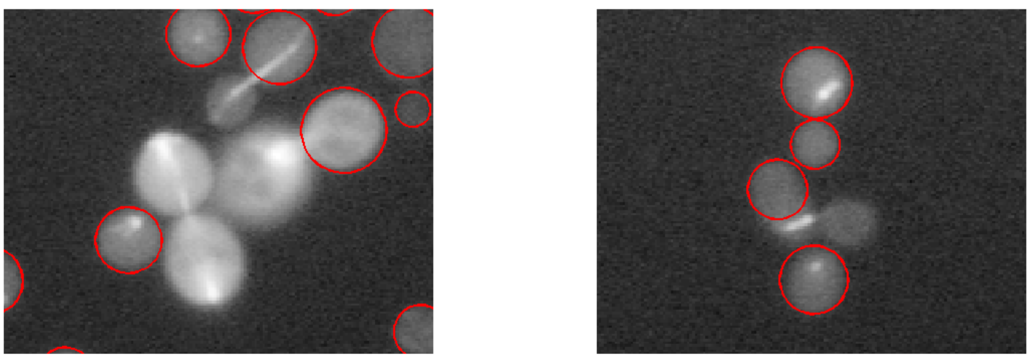
The algorithm is able to detect some small buds and cells that have been cropped out as shown in Figure 3. However, the algorithm fails sometimes when there are a cluster (closely linked) cells together in an area, as shown in Figure 4. The root of the problem is the selected circle is using least-squares to find the minimum error. Another way of selecting proposals is to rank the different candidates $(\vec{x}_c, r)$ in order of squared errors, instead of randomly choosing proposals as it was done in Section 3. These ranked candidates are the proposals.

For circle selection, instead of choosing the least-squares solution, one can choose randomly because the proposals are expected to be “best” relative to the other rejected candidates. This gives a stochastic approach which results in higher chance to detect linked clusters (i.e. intersecting cells).



(a) Small Bud (b) Cropped Cell

Figure 3: Magnified Cell image with circle fitting



(a) (b)

Figure 4: Magnified Cell image with failed circle fitting