CSC 2503 Assignment 1

Wuqi Li
1000292033

Question 1 Noise Model

As shown in the question, the noise distribution is Gaussian Distribution $\vec{m} \sim \mathcal{N}(0, \sigma^2\mathbf{I})$. Also, for a single edge position, the true edge position measurement $\hat{x_k}$ can be expressed in Gaussian distribution i.e $\hat{\vec{x_k}}, \sim \mathcal{N}(\hat{\vec{x_k}}, 0)$.

Therefore, by knowing $\vec{x_k} = \hat{\vec{x_k}} + \vec{m}$, we have $\vec{x_k} \sim \mathcal{N}(\hat{x_k}, \sigma^2\mathbf{I}))$

In the handout we have equation (2) , $e_k = \vec{a}^T\vec{x_k} + \vec{b} + \vec{x_k}^T\vec{x_k}$, it is clear to see that e_k is constructed by a normal distributed variable $\vec{x_k}$, a constant $\vec{b}$ and a chi-squared [1] distributed variable $\vec{x_k}^T\vec{x_k}$ with freedom 1.

For each e_k , $\vec{a}^T$ and $\vec{b}$ are constants since the center point is fixed when we are calculating the error. Thus, the variance can only determined by the normal distributed variable $\vec{x_k}$ and the chi-squared distributed variable $\vec{x_k}^T\vec{x_k}$. Therefore, $\vec{a}^T$ and $\vec{b}$ cannot affect the variance

From the equation (2) we can see that the magnitude of e_k is highly depended on the magnitude of $\vec{a}^T$ and $\vec{b}$, therefore, $\vec{a}^T$ and $\vec{b}$ will affect the Mean of e_k.

$\vec{a}^T$ and $\vec{b}$ are transform of $\vec{c}$ and r , the change of $\vec{c}$ and r will affect $\vec{a}^T$ and $\vec{b}$, and result in a change of Mean of e_k. Thus, $\vec{c}$ and r will affect the mean of e_k.

Question 2 LS Estimator

Question 3 Circle Proposals

Algorithm:

The basic idea of this algorithm is grouping point which belongs to same circle, and find the corresponding circle for each group. In order to do that, the points are firstly ordered by its position, therefore, looping the points is in the same order as travelling through the edge. While traveling the points, the dot product of current point and the next point will determine if two points are in same group. If two points belong to same circle, the angle between the normal vector will be small, and the dot product of two normal vectors will close to one.

Step1 index the points along the edge

(1) Random pick a point **p** in dataset and remove it from dataset D.
(2) Create a new dataset D' and put p into it
(3) Find the nearest point **q** from **p,** remove **q** from dataset **D** and put **q** to the new dataset **D'**.
(4) Set **p** to **q.**
(5) Repeat (3) - (4) until the dataset D is empty.

Now we have a new dataset D' and the points are ordered by location (go through the dataset equivalent to traveling along the edge.)

Step2 travel the edge and grouping points

(1) Get first point **p** and get the next point **p+1.** If the dot product of **normal of p** and the **normal of p+1** is greater than some threshold, the group **p** and **p +1**.
(2) Set p to p +1.
(3) Repeat (1) (2) until **p+1** does not exist.

Now we have a set up groups, which all the points in the same group are most likely belongs to same circle.

Step3 find center and radius

For each group, find a circle corresponding to the group, the center the mean of all the points, and the radius is the average distance from the center to all edge points.

Correctness

This algorithm is efficient since it can accurately find the circle which most of the points are included in the dataset (Image 2.1 left). Also, it can provide an estimated center for the circle which only have a small amount of points in the edge (Image 2.1 mid), and the center will be

correct during the robust fitting. Also, it perfectly prevent the case that the circle that is formed by the outside of the edge since the first normal vector is pointing inward and the also points will compare the dot product with previous point.

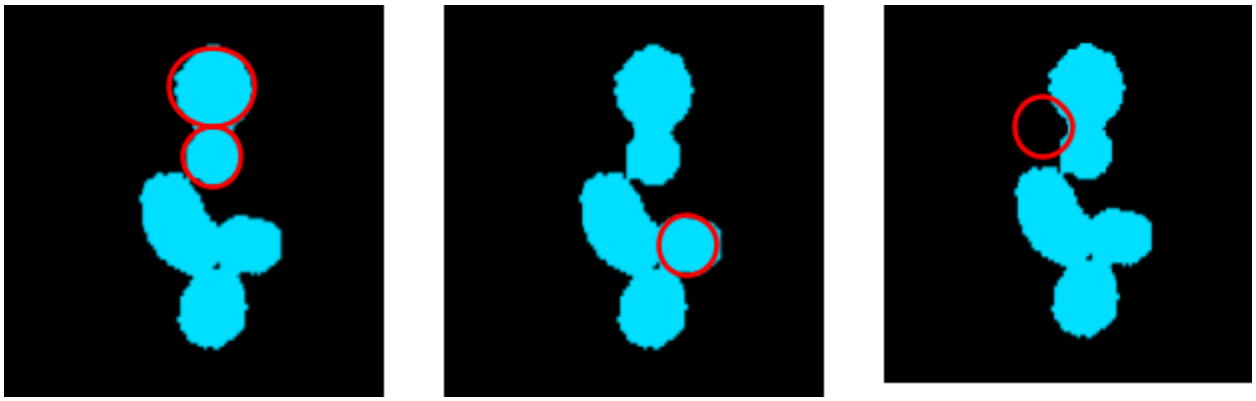


Image 2.1

Q4 Circle Selection

The best circle should fit its edge points as close as possible. Therefore, we should consider the total squared function. $e_k = (x_k - x_c)^2 - r^2$ and $O' = \sum_k e_k$.

However, the O will be highly affected by the points which are not belongs to this circle. O can be very large even if it fit the points very well locally. Therefore, we need to weaken the error which are very large. (If e_k is very large, it is most likely belongs to other circle, and we do not need to consider it.). GM estimator is considered for estimation the total error of the selected circle. $\rho(e) = \frac{e^2}{\sigma^2 + e^2}$ and $O(e) = \sum_k p(e)$, and we select the circle which have minimized O(e) as the best circle.

Q5 Robust Fitting

Step 1

As shown in the question $O(\vec{a}, b) = \sum \rho(e_k(\vec{a}, b), \sigma_g)$, we know O is a function $\vec{a}, b$, for simplification, we use $\vec{c}$ to represent $\vec{a}, b$ where $\vec{c} = [\vec{a}, b]$ with dimension 1x3.

=> $O(\vec{c}) = \sum \rho(e_k(\vec{c}), \sigma_g)$

In order to construct WLS, we use $w_k = \frac{1}{e_k}\frac{\partial \rho}{\partial e_k}$, and $O(\vec{c}) = \sum w_k e_k^2$.

Step2

From the handout we know

$$\rho(e, \sigma_g) = \frac{e^2}{\sigma_g^2 + e^2}$$

Get the derivative of rho respect to e, we have

$$\frac{\partial \rho}{\partial e} = \frac{2\sigma_g^2 e}{\sigma_g^2 + e^2} \qquad w_k = \frac{2\sigma_g^2}{(\sigma_g^2 + e^2)^2}$$

Step3

From previous section, we know $e_k = \vec{a}^T \vec{x_k} + \vec{b} + \vec{x_k}^T \vec{x_k}$, therefore, we have

$$e_k = c^T y_k + x_k^T x_k$$

Step4

From step1 and step3, we have

$O(\vec{c}) = \sum w_k (c^T y_k + x_k^T x_k)^2$ $\frac{\partial O(\vec{c})}{\partial \vec{c}} = \sum 2 y_k^T w_k (c^T y_k + x_k^T x_k)$ as a result of derivative.

By letting the derivative to be 0. We have

$$\sum y_k^T w_k c^T y_k = \sum y_k^T w_k x_k^T x_k$$

Construct matrix by the sum equation, we have

$Y^T W Y c = Y^T W X'$ where X' is row wise norm-2 square

Y is matrix that have all the k points and one extra column that filled by ones.
W is matrix whose diagonal filled by w_k
X' is the original dataset which has k elements

Deriving x_r and r

$\vec{c} = [\vec{a}, b]$

From question 2, we know that $\vec{a} \equiv -2\vec{x}_c$ and $b \equiv \vec{x_c}^T \vec{x}_c - r^2$, it is easy to see that

$$\vec{x_c} = -\frac{1}{2}\vec{a}$$

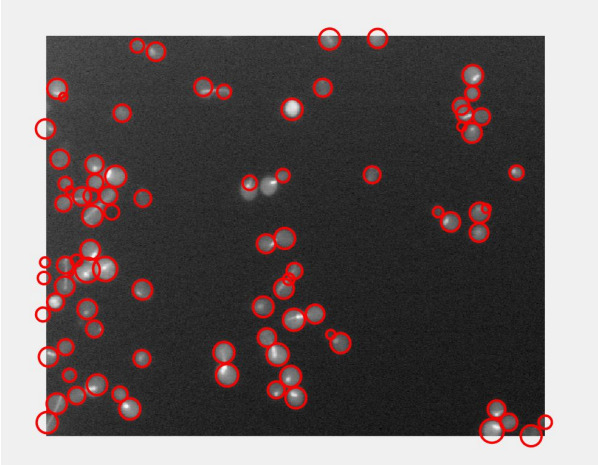
$$r = \sqrt{\vec{x_c^T}\vec{x_c} - b}$$

Varying sigma

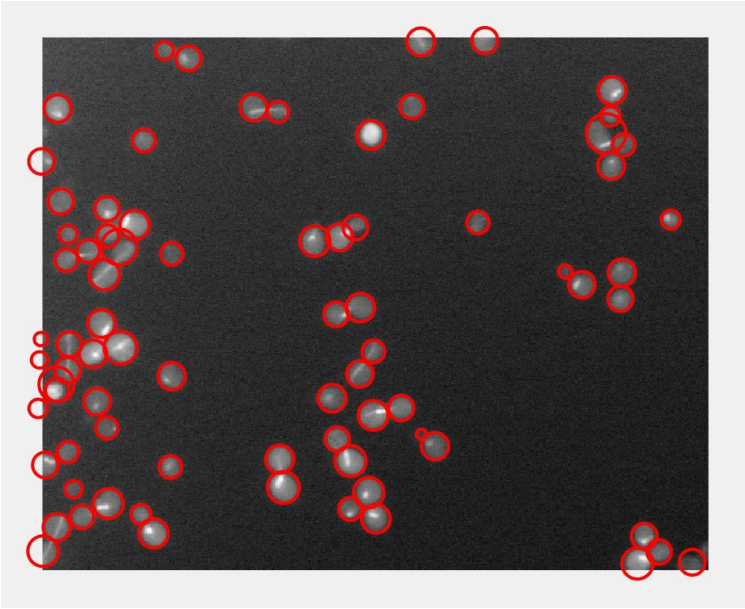
Different sigma will result to different sizes and locations of circle.

For small sigma, the error of points from far away will affect less to the total error since the w_k will converge to 0 faster as error increase. Therefore, there exist some small circles that only fit some small subset of points, and the radius are normally small.

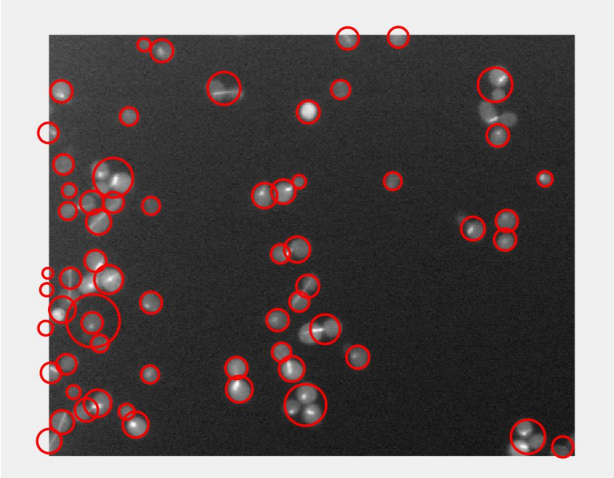
For larger sigma, the error from far have more impact to the error, therefore, it result to the existence of circles have large radius.



sigma30



Sigma 100



Sigma 200

6. Modal Update

In order to select a good circle, there are two factors need to be considered.

(1) The circle should be fit very well (error to be small enough).

(2) The circle should not cross with other circles

(1) For point one, we need evaluate W. after have an array of w. We need the average of w to be smaller than a factor.

As discussed above, $w_k = \frac{2\sigma_g^2}{(\sigma_g^2 + e^2)^2}$.

If error very small, $w_k = \frac{2}{(\sigma_g)^2}$

If the error is very large $w_k = 0$

Therefore, we want the mean of w_k to be greater than a factor. The factor should be varying with sigmaG. If it is too small, which means a lot of points are far away. It may result in a bad circle.

(2) For point two, we need to look at each existing circle, and find out the distance between the new circle and the existing circle. If the distance is smaller than the sum of radiuses times a factor, then it is crossing other circles too much. It is a bad circle.

7. Brief Evaluation

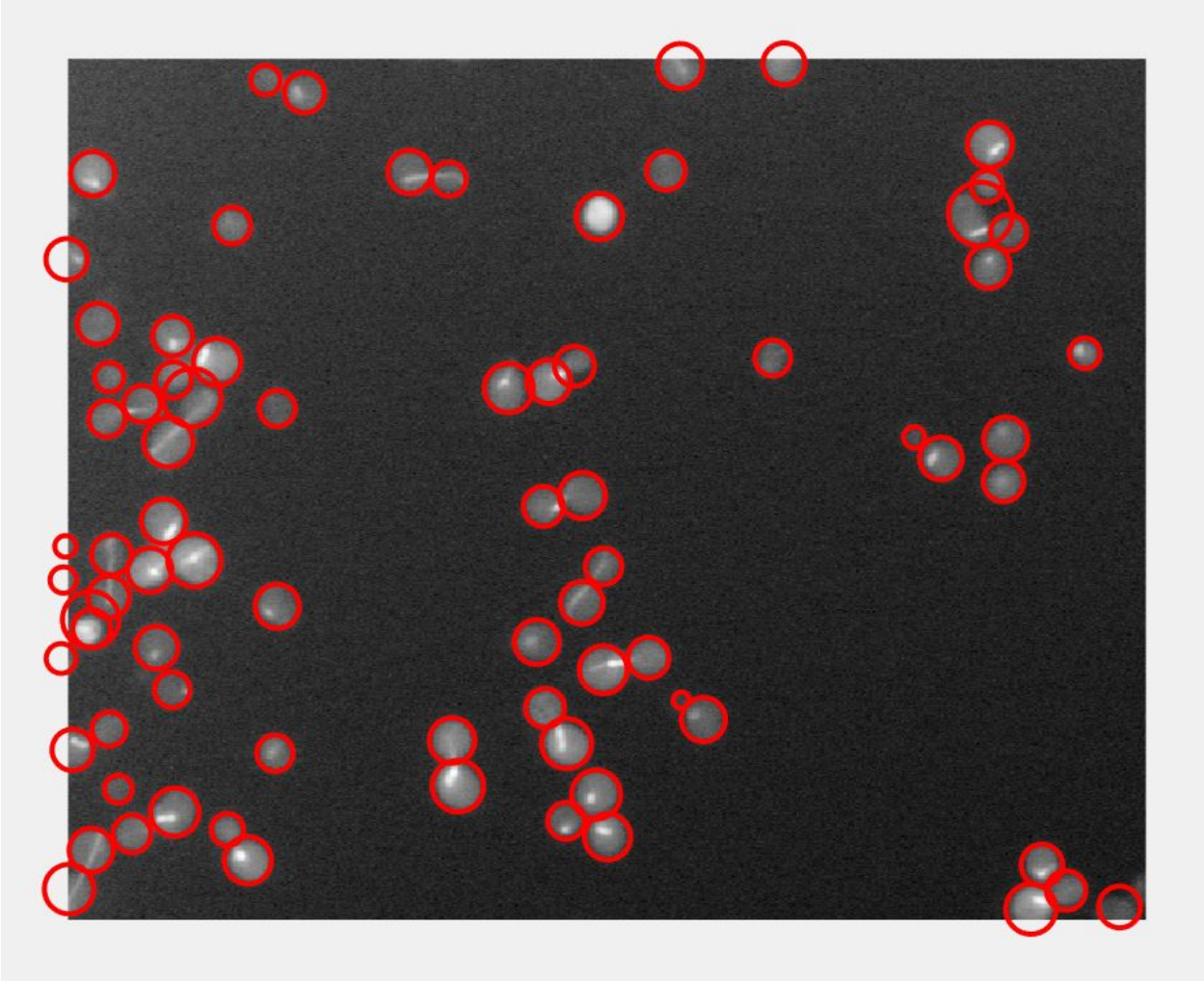
The algorithm works surprisingly good for the test cases. It even caught all the boundary points. For the inner points, it has overlapped circles at one positions. The cause for the failure is constructed by isGoodCircle function.

The overlapped circle should be eliminated by isGoodCircle by setting false, because the distance between two center is much smaller than the sum of radius.

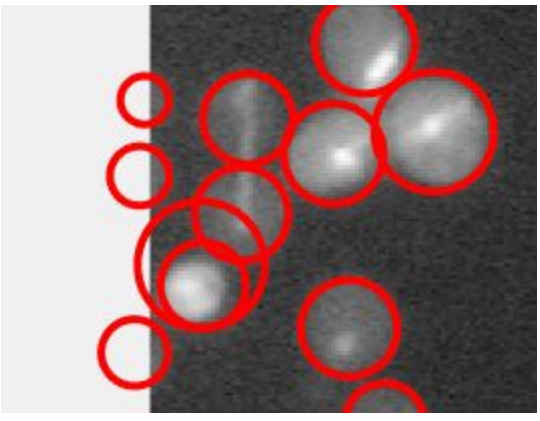
```
if distance < (r + radius) * factor
    goodCircle = false;
    return
end
```

However there exist a factor which need to multiply the sum of radius. If the factor is too large, it will be too strict, and circles which are close to any other circle will be eliminated. If the factor is too small, the result will be too lose and there will be many more overlappings.

Future fix, find a way to check the percentage of overlapping in isGoodCircle function, if the percentage is greater than certain threshold, the new circle should be eliminated.



Test result with sigma 100



Failure position

References

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