CSC420: Assignment 2

Rohan Chandra

g9chandr

996274142

Circle proposals:

In general, the algorithm randomly selects a pair of edgels and calculates the intersection point of their normals . This intersection point is chosen as the center of the proposed circle and the radius is set to the distance between the intersection point and one of the selected edgels.

The intersection is calculated as:

Where P_i is the edgel point,d_i is a unit vector in the direction of the edgel normal, and L_i is a scaling parameter

$P_1 + L_1 d_1 = P_i$

$P_2 + L_2 d_2 = P_i$

$P_1 + L_1 d_1 = P_2 + L_2 d_2$

$L_1 d_1 - L_2 d_2 = P_2 - P_1$

$[d_1, -d_2][L_1, L_2]^t = [P_2 - P_1]$

and then solving for $[L_1 L_2] = [d_1 - d_2] \setminus [P_2 - P_1]$

The algorithm detects erroneous proposals in the following ways:

The algorithm discards edgels pairs who have normals that do not intersect, are near parallel, or occur on top of each other, based on the conditioning of the matrix $[d_1, -d_2]$ (I.e using the conditioning number to check if the matrix is near singular). If either of the scaling parameters are negative, then the intersection occurs outside of the circle on which the corresponding edgel lies, as the edgel normals points towards the center of the edgel's circle. Additionally if the size of the scaling parameters are disproportionate comparatively to each other then it indicates that the intersection point is closer to one edgel than the other, contradicting the intention to find a pair of edgels that lie on the radius of the same circle. Finally, if the edgels are relatively close together, and there is a relatively large number of edgels remaining, it is likely, for a cell that is not a bud, that the intersection point will occur far outside of the circle and the proposed circle will be erroneous.

If it is the case that no such valid pair of edgels are randomly selected, according to the aforementioned criteria, the proposal loop will stop attempting to generate proposals after $||edgels||^2$, which in general should be enough to test most combinations of remaining edgels.

In this method, I assume that at least one of the edgels pairs I select lie on the same circle, and thus will generate an intersection point that roughly corresponds to the center of their circle.

Circle selection:

The best proposal algorithm works by rewarding for the number of points on the radius and penalizing for the number of points that occur within the radius and the number of points that have normals that deviate largely from the direction of the proposed circle's center. These score values are normalized by the size of the proposed circle's radius, to encourage smaller circles, and the proposed circle with the highest score is considered the best proposal.

To determine the points that occur on the radius, I calculate the distance between each edgel and the circle's center. If the distance is within some threshold factor above and below the radius of the proposed circle I consider it to be a valid point on the radius of the proposed circle. Conversely, a penalty is awarded for every point whose distance to the center is less than some threshold fraction of the proposed circle's radius, as a correct circle should not contain edgels, namely points that are meant to lie on the edge of the true circle. Finally, a smaller penalty is given to points that lie on the radius but have normals that deviate largely from the proposed circle's center. This helps prevent choosing circles that have many points on the radius and some points within them, when a proposal circles a mass of conjoined cells. Finally the overall score is divided by the radius of the proposed circle, which encourages smaller circles with less points inside of its radius rather than very large circles who either contain a sparse set of points within its radius or so many points on the radius that the penalty of inner points is largely irrelevant. In certain circumstances, the normalization also forces the algorithm to choose circles that represent the bud first before the connecting cell, which avoids errors that may occur if too large a circle was fitted to the connecting cell to allow for a bud.

Initially I attempted to fit a Gaussian distribution to the radius and penalize by a Gaussian fit to the center of the proposed circle, however I had issues with the inner Gaussian as I either was penalizing far to harshly for points towards the center and not enough as they approached the radius or was distributing the penalty across the points to strongly. Attempting to remove the normalization of the Gaussian produced results that were far more skewed towards very large circles that had only a few points on the edge of their radius and a few points that were around the midpoint of the radius. I found that I had far better results using the above counting method rather than a series of Gaussian.

IRLS:

I derived the normal equations as follows:

The derivation for $dO(X_c,r)/dX_c$:

$$O(X_c,r) = \text{sum}(1,k):w_k[e_k(X_c,r)]^2$$

$$dO(X_c,r)/dX_c = -2*\text{sum}(1,k):w_k[e_k(X_c,r)]\ d/dX_c[e_k(X_c,r)]$$

Then $d/dX_c[e_k(X_c,r)]$ (I.e direvative of e_k with respect to X_c)

$$d/dX_c[e_k(X_c,r)] = d/dX_c[(X_k - X_c)^t u_k - r]$$

$$= -u_k$$

Substituting this into the previous equation:

$$dO(X_c,r)/dX_c = -2*\text{sum}(1,k):\ w_k[e_k(X_c,r)]^2 u_k$$

$$dO(X_c,r)/dX_c = -2*\text{sum}(1,k):\ w_k\,(X_k - X_c)^t u_K^2 - r*u_k$$

$$dO(X_c,r)/dX_c = -2*u_k*w_k*\text{sum}(1,k):(X_k - X_c)^t u_K - r$$

The partial derivative with respect to r , $dO(X_c,r)/dr$ is:

$$O(X_c,r) = \text{sum}(1,k):w_k[e_k(X_c,r)]^2$$

$$dO(X_c,r)/dr = -2*\text{sum}(1,k):w_k[e_k(X_c,r)]\ d/dr[e_k(X_c,r)]$$

Then $d/d_r[e_k(X_c,r)]$ (I.e direvative of e_k with respect to r)

$$d/d_r[e_k(X_c,r)] = d/r[(X_k - X_c)^t u_k - r]$$

$$= -1$$

Substituting this into the previous equation:

$$dO(X_c,r)/dr = -2*\text{sum}(1,k):\ w_k[e_k(X_c,r)]^2 u_k$$

$$dO(X_c,r)/dr = 2*\text{sum}(1,k):\ w_k\,(X_k - X_c)^t u_K - r$$

$$dO(X_c,r)/dr = 2*w_k:\text{sum}(1,k):\ (X_k - X_c)^t u_K - r$$

Thus, the partial derivatives are

$$dO(X_c,r)/dX_c = -2*u_k*w_k*\text{sum}(1,k):\ (X_k - X_c)^t u_K - r$$

$$dO(X_c,r)/dr = 2*w_k*\text{sum}(1,k):\ (X_k - X_c)^t u_K - r$$

since, X_c is a function of of x and y, it itself has the partial derivatives

$$dO(X_{cx},r)/dX_{cx} = -2*u_{kx}*w_{kx}*\text{sum}(1,k):\ (X_{kx} - X_{cx})^t u_{Kx} - r$$

$$dO(X_{cy},r)/dX_{cy} = -2*u_{ky}*w_{ky}*\text{sum}(1,k):\ (X_{ky} - X_{cy})^t u_{Ky} - r$$

Then, equating the partial derivatives to 0, the normal equations are found as follow:

As the partial derivatives for r and X_c differ by a scalar factor, when treating uk as a constant. Since the partial derivatives are equated to 0, we can divide through by the scalar factors in both cases and both derivatives have the form

$$\text{sum}(1,k):\ (X_k - X_c)u_K - r = 0$$

To solve this, first split X_k into its two dimensions:

$$\text{sum}(1,k):\ ([X_{kx} - X_{cx}\,;\, X_{ky} - X_{cy}]^t)[u_{Kx}, u_{Ky}] - r = 0$$

$$\text{sum}(1,k):\ ([X_{kx} - X_{cx}\,;\, X_{ky} - X_{cy}]^t)[u_{Kx}, u_{Ky}] - r = 0$$

$$\text{sum}(1,k):\ X_{kx}\,u_{Kx} - X_{cx}\,u_{Kx} + X_{ky}\,u_{Ky} - X_{cy}\,u_{Ky} - r = 0$$

$$\text{sum}(1,k): X_{cx}\,u_{Kx} + X_{cy}\,u_{Ky} + r = \text{sum}(1,k):\ X_{kx}\,u_{Kx} + X_{ky}\,u_{Ky}$$

Thus, we solve for a specific k as

$$X_{cx}\,u_{Kx} + X_{cy}\,u_{Ky} + r = X_{kx}\,u_{Kx} + X_{ky}\,u_{Ky}$$

$$(u_{Kx}; u_{Ky}\,;\, 1) \cdot (X_{cx}\,;\, X_{cy}\,;\, r) = (X_{kx}\,u_{Kx} + X_{ky}\,u_{Ky})$$

$$(X_{cx}\,;\, X_{cy}\,;\, r) = (u_{Kx}; u_{Ky}\,;\, 1) \backslash (X_{kx}\,u_{Kx} + X_{ky}\,u_{Ky})$$

Hence, the whole system can be solved as

$$\begin{vmatrix} u_{1x} & u_{1y} & 1 \\ u_{2x} & u_{2y} & 1 \\ \vdots & & \\ u_{kx} & u_{ky} & 1 \end{vmatrix} * \begin{vmatrix} X_{cx} \\ X_{cy} \\ r \end{vmatrix} = \begin{vmatrix} X_{1x}\,u_{1x} + X_{1y}\,u_{1y} \\ X_{2x}\,u_{2x} + X_{2y}\,u_{2y} \\ \vdots \\ X_{kx}\,u_{kx} + X_{ky}\,u_{ky} \end{vmatrix}$$

The IRLS algorithm I implemented iteratively chooses a set of inlier, solving the aforementioned normal equations using the system of equations described previously using these inlier as the set of points X_k. Since u_k is treated as a constant, the values for the Xc generated in the previous iterations (using the proposed circles original center on the first iteration) are used to generate a value for u_k to be used on the current iteration. The values for the fitted circle's center coordinates and its radius are then updated with the solution from the system of equations formed by the normal equations.

A point is determined to be an inlier by checking its distance to the center of the circle to the radius and confirming if it is within some threshold range of the radius. The set of inlier is further culled by removing those points whose normals deviate from a direction vector from the point to the center by a significant degree, I.e those edgels who likely do not correspond to the proposed center.

Rather than using a ransac parameter epsilon to determine inliers that are within some epsilon of the fitted circles radius, the algorithm selects all points whose distance to the fitted center is less than the radius + 2*sigmaErr. This helped the algorithm when fitting a circle to a cell that was in the process of cell division but had yet to undergo cytokinses and thus appeared more elliptical than circular. In these cases a larger circle would be fit to the ellipse and the radius wouldn't necessary contain all of the edgels on its boundary, but rather would contain some within its diameter.
The binary weight for each point was set to false if the point occurred outside of the radius + 2*sigmaErr, and true otherwise.

Model Update

The algorithm begins by rejecting circles made up of a very small number of points as these tend to produce erroneous results and may prevent a more valid circle from later being generated else where. The algorithm then calculates the intersection of the proposed circle with existing circles and rejects it if contains or is contained by an existing circle or if intersects a circle of comparatively radius by some non trivial amount.

The intersection is calculated by first determining the distance between the two circles, if the distance-r1-r2 is greater than 0 than the circles do not intersect. Else the distance from the center of one circle to the intersection point of the two can be solved in the standard way, as seen here http://mathworld.wolfram.com/Circle-CircleIntersection.html . If the two circles intersect such that the distance from the intersection point to the center is greater than some threshold value applied to the radius, the algorithm considers the circle for rejection. If the proposed circle is a small fraction of the radius of the existing circle it exists, it is strongly possible that it is intersecting a bud and is not discarded, otherwise, if the radii are comparable, the proposed circle is discarded.

If a circle is deemed valid, all of the edgels that were selected as inlier to it, as described in the IRLS section, are removed from the data set, as they correspond to the radius of a valid circle or are contained within it. If the circle is not deemed valid, all the edgels are returned to the set of edgels to be considered for future circles.

Evaluation:

The following are the failure models that are not cells that are significantly cropped or small buds.

1) If the cell is an elliptical cell that is close to dividing, the algorithm incorrectly decides that the circles have not intersected far enough into each other as to be considered an invalid intersection. Because the intersection condition is a function of the distance the radius of the new circle passes into the existing one it is difficult to prevent this occurrence without also removing the algorithm's ability to test for buds. However, this might be solvable by observing that buds have a very distinct edgel pattern, particularly that they are usually a semi circle of edgels occurring off the side of an existing circle. Trying to detect this pattern and then creating a different set of validation rules for such a pattern would allow this case to be handled without hampering bud detection.

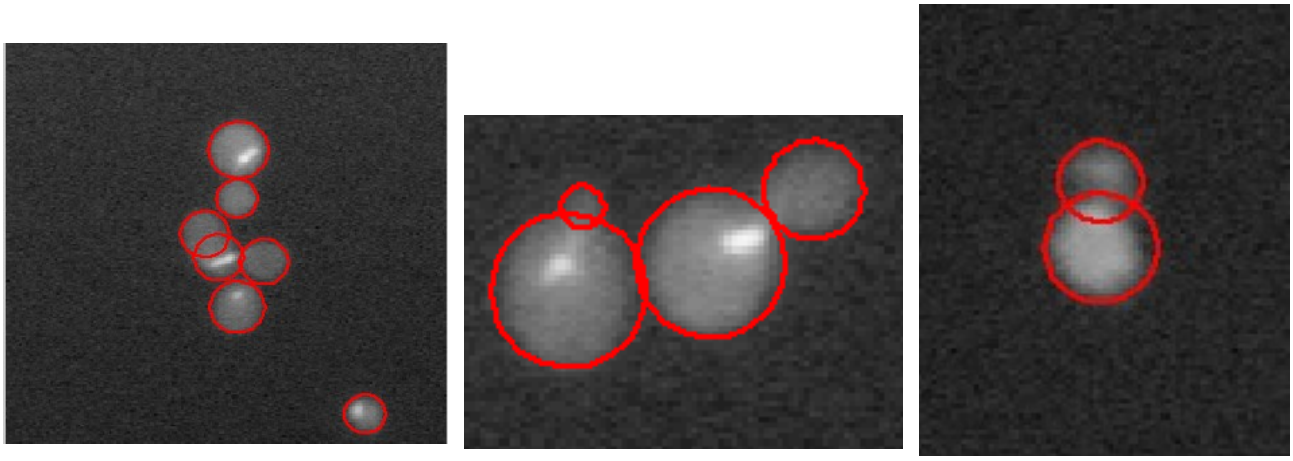

figure 1: an example of an incorrectly allowed circle intersection and two examples of the bud case

2. The algorithm occasionally fits two small of a circle to an elongated cell. This is a result of the algorithm strongly favoring smaller cells in the best proposal section. The edgels that are outside of the proposed circle but on the elongated cell are too far away to be considered inlier during the IRLS process and the fitting increasingly fits to the edgels in the top section with each iteration. The algorithm might be able to avoid this if the best proposals did not give such high scores to smaller circles, but this would cause reduced accuracy in fitting circles to cell groups consisting of several intersecting cells. However, this only occurs occasionally, if the algorithm was performed several times and the circle fit to cells compared across instances, it may be possible to correct for this error by either averaging each iteration.

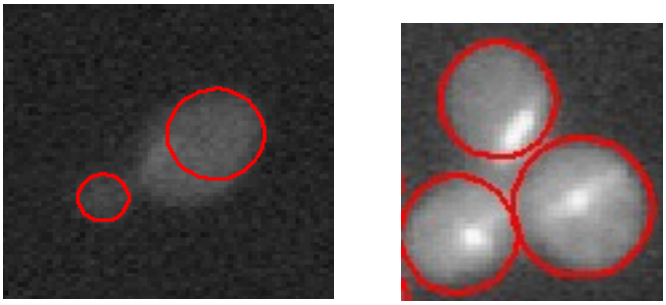

Figure 2: an elongated cell with too small of a circle fitted to it and a group of cells made up of multiple intersecting cells.

3)

When trying to fit a circle to the narrow line like structural made up of many cells, the proposed circle encompass all of the edgels and has a center that occurs off of the image. This error occurs due to fact that the edgels are all very close together and the circles that would have to be fitted to them would be very small. If the maximum distance between every edgel was calculated, an upper bound could be fixed on the possible radius of any circle fitted to the edgels could be determined. The algorithm could then compare the length and width of the entire set of edgels to determine that it is much larger in one dimension then the other to realize that a single circle should not fit the edgels. Then a series of circles with radius less than the maximum previously computed could be fitted to the string of small cells in order to better approximate it.

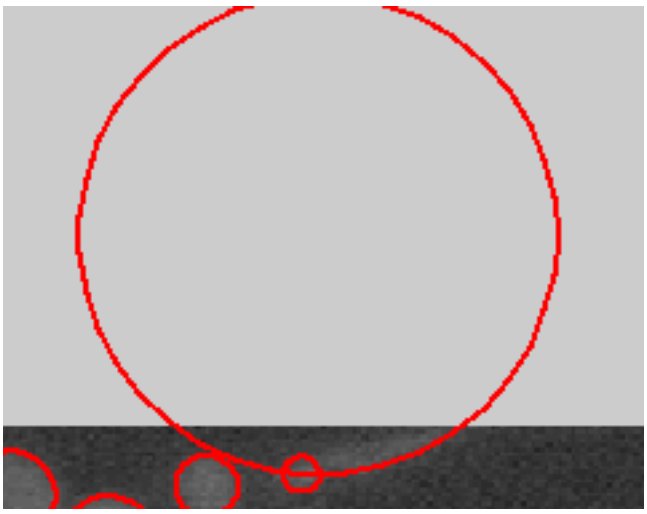

Figure 3: the string of small cells that cause a large erroneous circle to be fit to them.

4) Occasionally this specific structure of cells is missed. This occurs due to the fact that there are a small number of edgels on the circle to evaluate, as it is largely contained by other cells on each side. If the other circles are evaluated and set first, then there is only one side of edgels remaining, and the proposals they generate will tend to intersect existing cells. This issue arises depending on the order in which the cells are evaluated and occurs infrequently, so an averaging over a number of runs of the algorithm could be used to resolve this as discussed previously.

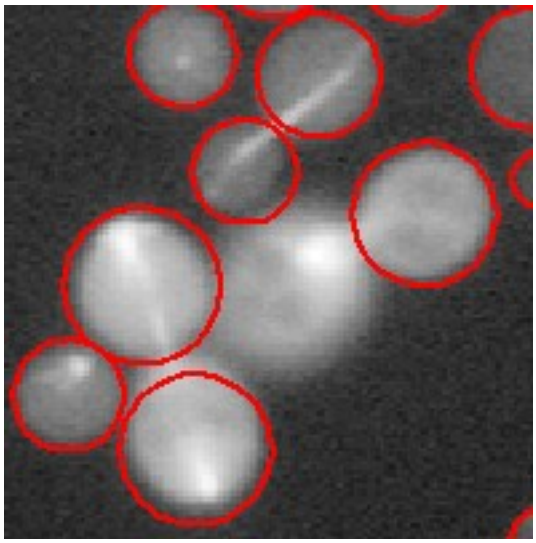

Figure 4: a cell who that is surronded by other cells and is ignored by the algorithm.