CSC420— Introduction to Image Understanding, Fall 2011

Assignment 2: Estimation of Multiple Models

Due: 3:00pm, Tues., Nov. 1 (at the start of the class)
This assignment is worth 20 marks towards your grade in this course.

What to hand in. Write a short report addressing each of the itemized questions below. We will be marking both the ocntent and style of your written reports. You can assume that the marker knows the context of the questions, so do not spend time repeating material in this hand-out, or in class notes. Package the report as a pdf file, A2Report.pdf, and your Matlab solution code as, A2Matlab.zip. Submit both these files to CDF, as for assignment 1.

Robustly Fitting Circles. A geneticist wishes to automatically count cells in microscope images, such as the one in Fig. 1a, and take specific note of cells that are in various stages of splitting (i.e., those with cell "buds"). Here we consider an application of RANSAC that will get us started on a solution for the geneticist. We will approximate the shape of individual cells using circles (although their true shape may deviate from this).

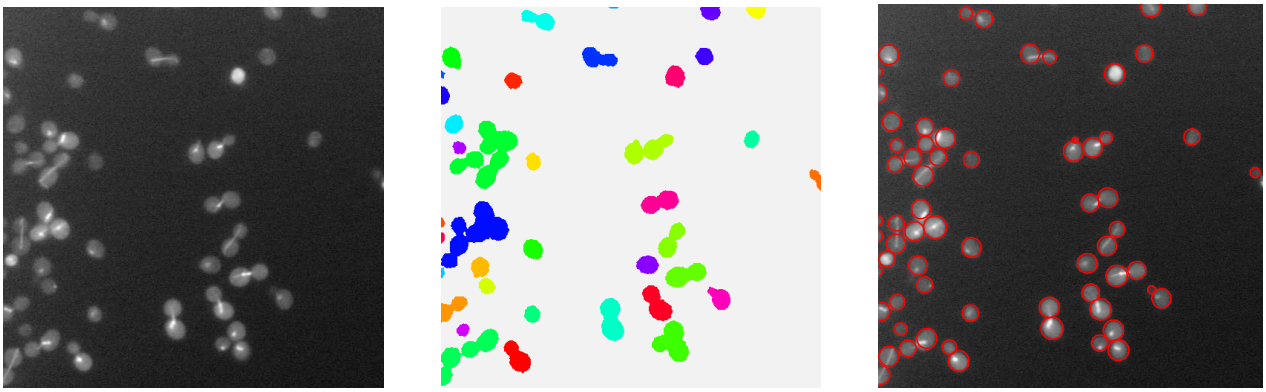

Figure 1: (a) Cell image; (b) Connected foreground components; (c) Fitted cells (view in colour).

The handout code `findCellScript.m` reads in a microscope image of cells (Fig. 1a). Separately, we have preprocessed these images for you, detected foreground cell pixels, and have labelled connected components of these foreground pixels (Fig. 1b). The connected component labels are read in by the handout code as a second image. A binary foreground image is generated (with white foreground pixels and black background pixels), and Canny edgels are computed for this foreground mask. The handout code then loops over regions around each connected component.

Your job is to fit circles to the edgels obtained from the foreground mask. The idea is that each fitted circle should correspond to one "cell," including any small circle for a cell bud just being born (see Fig. 1c). This is a vague definition of what constitutes a "cell" or a cell "bud". However, if you study Fig. 1c (perhaps by blowing it up in the electronic copy), it should be intuitively clear what is meant by these terms. Part of your job in this assignment is to decide on how to operationally define these intuitive notions.

In the following we denote a given edgel by $(\vec{x}_k, \vec{n}_k)$, where $\vec{x}_k$ is the image position and $\vec{n}_k$ is the edgel normal. The edgel normal points in the direction of increasing brightness of the foreground mask (i.e., towards the *inside* of the cells).

Because cells occur in close proximity, a set of edgels $\{(\vec{x}_k, \vec{n}_k)\}_{k=1}^{K}$ from any one region will often correspond to multiple (roughly circular) cells. When estimating the parameters of one circle (cell), we view the edgels

for any other cell as outliers.

To help direct your solution, a general strategy for fitting circles to this edgel data has been partially implemented in the handout code `findCellScript.m`. You only need to complete several M-functions, as described below. In general, you are allowed to make changes to the handout code if you wish.

1. **Circle proposals.** Your first job is to randomly select pairs of edgels and generate a proposal for a circle which passes nearby both edgels in the pair (if possible). Here you should use the edgel positions $\vec{x}_{k(j)}$ and the normals $\vec{n}_{k(j)}$ for the selected edgels $k(j)$, $j = 1, 2$.

   Complete `getProposals.m` (in the `circleFit` subdirectory) to implement your strategy for proposing possible circles. When finished this function should return a $P \times 3$ array `circles`, where each row of `circles` corresponds to the parameters (x_c, y_c, r) of a proposed circle. Here $\vec{x}_c = (x_c, y_c)^T$ denotes the image position of the center of the circle, and $r > 0$ denotes the radius of the circle. Your implementation of `getProposals` should attempt to produce up to `numGuesses` proposals.

   If you have trouble with this step, you can begin by writing `getProposals` so that it simply generates proposals from moused in points (say a center point plus one point on the circle). You won't get any marks for this, but it will help you get started on other parts of the problem.

   In your write up, clearly explain your approach for proposing circle parameters from two randomly selected edgels. What, if anything, do you need to assume about the data?

2. **Circle selection.** The next step in `findCellScript.m` is to select the best circle from the list of proposed circles. Implement your selection process in the function `bestProposal` so that it chooses a promising circle from the list of proposals. In your write up, clearly describe how you select the best circle (i.e., be specific about what you mean by "best"), and explain your reasons for this choice.

3. **IRLS.** Derive an IRLS algorithm for estimating the circle parameters $(\vec{x}_c, r)$ which (locally) minimize the objective function

$$\mathcal{O}(\vec{x}_c, r) = \sum_k w_k [e_k(\vec{x}_c, r)]^2, \quad \text{with} \tag{1}$$

$$e_k(\vec{x}_c, r) = (\vec{x}_k - \vec{x}_c)^T \vec{u}_k - r. \tag{2}$$

   Here $\vec{u}_k \equiv (\vec{x}_k - \vec{x}_c)/||\vec{x}_k - \vec{x}_c||$, which is a unit vector in the direction of $\vec{x}_k$ from the circle center $\vec{x}_c$. As is the class notes, use binary weights w_k. In order to minimize $\mathcal{O}(\vec{x}_c, r)$ in (1), calculate the normal equations. In this calculation ignore the dependence of $\vec{u}_k$ on $\vec{x}_c$ and treat the weights w_k as constant.

   Given these normal equations, develop an IRLS algorithm which uses the solution at the previous iteration to re-evaluate both $\vec{u}_k$ and the weights, and then solves the resulting normal equations for updated values $\vec{x}_c$ and r. At each stage the binary weights can be chosen in the manner described in the lecture notes on RANSAC.

   Complete the function `fitCircleRobust` to implement this IRLS algorithm. In order to test your code, you can initially set `demoRobustConv` to `true`. This displays how your code behaves for each initial guess provided by `getProposals` in step (2) above.

   In your write-up, include the derivation of the these normal equations for $\vec{x}_c$ and r. Clearly describe your IRLS algorithm, including how you chose an appropriate value for the RANSAC parameter ϵ, and how you defined the binary weights w_k.

   Before continuing on, set `demoRobustConv` back to `false`.

4. **Model update.** The "model" for each region is defined to be a set of circles which are meant to be in one-to-one correspondence with cells and buds. The plan is to build up such models incrementally, one circle at a time. In particular, given the robustly fit circle from step 3, decide if it should be kept in the model (this decision should be implemented in the function `isGoodCircle`). If it is decided that the circle should be added to the current model, then greedily remove all edgels from the data which correspond to

the current circle (you decide how to determine this). Steps 1-4 are then repeated to fit additional circles and possibly add them to the model. This completes the cell finder.

In your write up, describe on how your function `isGoodCircle` decides to keep a new circle. Moreover, explain precisely which edgels you remove from the data. Finally, explain why you made these choices.

5. **Evaluation.** Your completed code should now provide a very reasonable first cut at solving our geneticist's problem. It should be expected to miss cells that have been significantly cropped due to the side of the images. Also, small buds on some cells might be missed (see Fig. 1). Other than these two "failure modes", your algorithm should work very well. What other situations are observed to cause your algorithm to fail to find a plausible set of circles? Briefly describe what you might do to try to fix these remaining problems.