Red blood cell segmentation using guided contour tracing

Joost Vromen[1], Brendan McCane[2]

[1]Human Media Interaction
University of Twente, Enschede, The Netherlands
Phone: +31 53 489 3740 Fax: +31 53 489 3503
Email: j.a.f.vromen@student.utwente.nl

[2]Graphics and Vision Research Lab
University of Otago. Dunedin, New Zealand
Phone: +64 3 479-8488 Fax: +64 3 479-8529
Email: mccane@cs.otago.ac.nz

Presented at SIRC 2006 - The 18th Annual Colloquium of the Spatial Information Research Centre
University of Otago, Dunedin, New Zealand
November 6th-7th 2006

ABSTRACT

We present a model-based contour tracing approach to the problem of automatically segmenting a Scanning Electron Microscope image of red blood cells. These images characteristically have high numbers of overlapping cells and relatively smooth contours. We provide a brief look into what problems conventional algorithms encounter when attempting to segment these images, and go on to show that a model-based contour tracing approach attains high levels of accuracy and almost no false negatives.

Keywords and phrases: contour tracing, computer vision, image segmentation, red blood cells

1 INTRODUCTION

In the early 1980s, Dr L.O. Simpson(Simpson 1989), while researching the filterability of blood, hypothesised a link between red blood cell shape and the ease with which the blood could be filtered. Further research showed that the distribution of differently shaped red blood cells varied significantly in samples taken from people suffering from particular illnesses, such as ME(Myalgic Encephalomyelitis) and MS(Multiple Sclerosis).

Research in this area is held back because the task of visually identifying and counting enough red blood cells to form a reasonable estimate of their distribution is long and tedious. The aim of this project is to develop a method by which this process can be automated, or greatly simplified.

In his research, Simpson used images created by Scanning Electron Microscopy(SEM). Blood samples were first fixed immediately after drawing, and then gold-coated for better conductivity. This approach, while producing low-noise images, poses a significant problem. It is technically impossible to obtain pictures in which there is no overlapping of cells. An example of such a picture can be seen in figure 1[1].

The subject of this paper is to take on the first step of an automatic classifier: segmenting the source image into a number of different cells so we can obtain digital data for single cells, while circumventing the problem of cell overlap.

Overlapping cells, as shown in figure 1, cause imperfections in edges detected in a conventional way, such as when using Canny's edge detector(Canny 1986). Since the light cells are situated on a dark background, the change in brightness between a cell and the background will be much higher than the change in brightness between two cells. Consequently, edges separating cell from background are much clearer than those separating cell from cell, which leads conventional algorithms that use edge detectors and edge linkers to lump overlapping cells together and treat them as a single entity. Similarly, overlapping cells can lead to small gaps in detected cell edges, where there is next to no change in brightness. This further complicates finding the right edge to consider part of a cell contour.

[1]Note that the light gridlines visible on the image were added for the sake of manual counting. Before running the tracer algorithm, the lines are removedby interpolating between the adjacent lines.

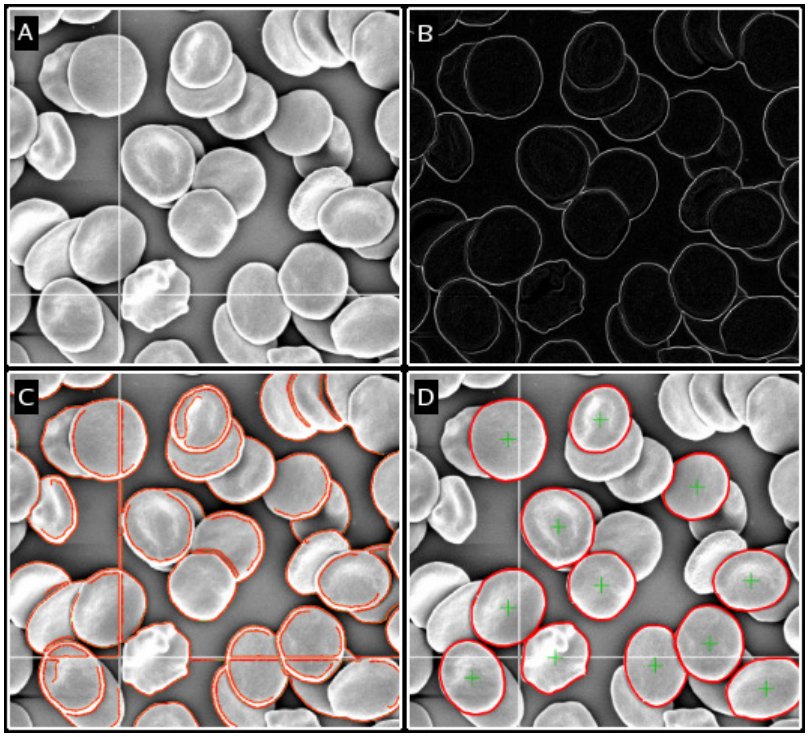

Figure 1: *A: Original image B: Gradient map C: Results of Canny's edge detector D: Results of guided trace*

The method presented in this paper avoids this problem by using a guided sequential trace of cell contours, based on information about changes in brightness in the source image. The incorporation of a model in the contour tracing process allows us to include *a priori* knowledge about contour shapes, instead of merely following the largest changes in brightness. For the purposes of this research, the algorithm will only look for fully visible or nearly fully visible cells, as these will be the easiest to classify in later stages of the project.

2 METHOD

There are two observations about the SEM images that we have to take into account:

- All images contain a large number of overlapping cells, and
- Some cells have non-elliptical boundaries

To overcome the first problem, we can force the contours to be smooth by using a model-based approach, such as matching ellipses to edge data. However, this approach will exclude the detection of the non-elliptic cells. Since these cells form an interesting category for detection, and our purpose is to identify and classify red blood cells, not being able to detect these "interesting" cells is a major drawback. We choose a middle-ground approach which ensures local smoothness, but not global shape.

The tracing algorithm can be divided into four steps:

- Calculate a gradient magnitude map
- Select starting points for traces
- Trace the contours
- Verify and cull the detected contours

This section describes each of those steps in turn.

2.1 Gradient map and starting point selection

Because SEM images are monochrome, we need not worry about colour information, only brightness. We use the two simple convolution operators $(-1, 1)$ and $(-1, 1)^{\mathrm{T}}$ to determine horizontal and vertical change in brightness. Early experimentation was done using the Prewitt operator instead, but the SEM images are crisp enough not to need the extra smoothing that using this operator provides. Convolution with these two operators defines a *gradient vector* for each pixel, indicating the direction of the largest change in brightness in that location.

In effect, the larger the gradient vector of a pixel, the higher the probability that it is in fact a pixel belonging to a cell edge. Therefore, we select the highest gradients as starting points for our traces. We cycle through each of them in turn, and start a trace in every point considered a valid starting point.

2.2 Tracing contours

Each trace is initiated by choosing an initial direction perpendicular to the gradient vector at its starting point. From this point on, the algorithm iteratively determines the next direction for the trace to take, until it either intersects itself or runs into some maximum contour length constraint. This maximum contour length should be larger than the circumference of the largest cell, or traces around large cells will be cut off before they complete. In our images, we set the constraint to 250 pixels.

Our task is to find the direction starting at the most recent point in the trace (called the *point of focus* for the remainder of this paper) that is most likely the direction of the next point along the cell contour. To accomodate both the need for a model-based approach to handle overlapping cells and the need for a flexible approach to ensure detection of non-elliptic cells, we need to take into account both gradient data around the point of focus and the history of the trace so far. To do this, we make both a history-based and a data-based prediction for each point in the trace.

The history-based prediction considers the last n points found in the current trace. This prediction is where we can add our *a priori* knowledge about cell contours. As can be seen in the pictures, most cells are roughly ellipse-shaped. While it can be tempting to model the contour as a best-fitting ellipse, early results showed that while ellipses give a good rough estimate of the entire contour, they lack the local accuracy needed to guide the trace around local deviations from a perfect ellipse. After some initial experimentation, we found that a simple second degree polynomial models local curves significantly better, and this is what we use in our current algorithm.

The polynomial is defined by:

$$x(t) = at^2 + bt + c$$

$$y(t) = et^2 + ft + g$$

We calculate the best fitting polynomial through a number of datapoints $\mathbf{x}_n$ according to the "Least Squares" technique, minimizing:

$$\sum_{k=1}^{n} ||\mathbf{x}_k - \mathbf{p}_k||^2$$

where n is the number of previously traced points considered and $\mathbf{p}_n$ contains the datapoints found for the polynomial. We can then predict a direction based on our model by taking the polynomial's derivative. This local curve will be very unstable and inaccurate during the first few steps of the trace, when only a few datapoints have been found. To overcome this, we use a linear model in the first ten steps of the trace.

The data-based prediction defines a set of unit vectors $\mathbf{u}_n$ with angles equally spaced around the point of focus. In the prediction step, we define:

$$\mathbf{u}_i = \{cos(i\alpha), sin(i\alpha)\}$$

In our algorithm, α is chosen to produce 60 different vectors.

We then calculate a confidence for each of the directions defined in $\mathbf{u}$, using the averaged gradient in that direction as a confidence measure, and calculate the set of predicted vectors $\mathbf{d_i}$:

$$d_i = \mathbf{u}_i * \frac{\sum_{t=1}^{l} G(\mathbf{f} + t\mathbf{v})}{l}$$

where l is the number of points to look ahead, $\mathbf{f}$ is the point of focus, and G is the bilinearly interpolated gradient map. The value of l can have a big impact on whether or not a contour is found. A value of l that is too low can prevent the algorithm from tracing across small low intensity gaps in the gradient image. If the value is too high, the directions chosen will be averaged over so many pixels that they bear little resemblance to the actual direction of the edge at that point. After some experimentation, we chose $l = 5$.

This produces a star-shaped collection of vectors, where the length of each vector indicates the average gradient in that direction, as shown in figure 2. Because the data prediction 'looks ahead' like this, the algorithm can continue the trace even if it encounters small low-intenstity gaps. These gaps frequently occur at junction points between cell edges.

Figure 2: *Visualisation of the evaluated direction vectors. The length of the vector indicates the average gradient in that direction.*

We then combine the two predictions by projecting the vectors from the data prediction onto the tangent found by the history prediction:

$$\mathbf{p}_i = c\mathbf{d}_i \cdot \mathbf{t}$$

where $\mathbf{t}$ is the tangent found by the history based prediction, and c introduces an *a priori* bias. This bias is used to ensure a tendency towards ellipse-like contours. If direction $\mathbf{d}_i$ points to the outside of the curve, c is 1. If the direction points inwards, c is 1.70. The direction with the maximum score in $\mathbf{p}_n$ is then selected, and the trace moves one pixel-length in that direction. After the trace intersects itself or becomes longer than a certain threshold, the contour is stored for post processing.

2.3 Verification and culling

Since our selection of starting points usually leads to more than one starting point per cell, it is inevitable that some contours will be found more than once. Some starting points can lead to "noise" contours, tracing around the outside of cells, or noise within cells, by mistake. This problem is solved in post processing by applying a culling algorithm. To check if a contour $\mathbf{c}_n$ has been found before, we calculate its centroid $\mathbf{a}$.

$$\mathbf{a} = \frac{\sum_{i=1}^{n} \mathbf{c}_i}{n}$$

If the euclidean distance between $\mathbf{a}$ and one of the previously found centroids is less than 30 pixels, $\mathbf{t}$ is culled. To check whether contour $\mathbf{t}$ is "noise", we use the method described by Fitzgibbon et al.(Fitzgibbon, Pilu & Fisher 1999) to find the parameters $\{a, b, c, e, f, g\}$ for the general conic equation $f(\mathbf{p}) = a\mathbf{p}_x^2 + b\mathbf{p}_x\mathbf{p}_y + c\mathbf{p}_y^2 + e\mathbf{p}_x + f\mathbf{p}_y + g = 0$ in such a way that the resulting conic is the ellipse most closely matching the datapoints of $\mathbf{t}$. We then calculate the *root mean square error* of $\mathbf{t}$ compared to that ellipse, using f as a measure of algebraic distance:

$$\mathrm{rmse}(\mathbf{t}) = \sqrt{\frac{\sum_{i=1}^{n} f(\mathbf{t}_i)^2}{n}}$$

If $\mathrm{rmse}(\mathbf{t})$ is higher than $1 * 10^5$, $\mathbf{t}$ is culled.

3 RESULTS

In evaluating the algorithm, we used a dataset provided by Dr Simpson containing 800 SEM images. These images measure 1024 by 768 pixels, grayscale. For a preliminary evaluation, we ran the algorithm on ten randomly selected images from the dataset. Detected contours were divided into three categories:

- Fully correct
- Minimally occluded
- Incorrect

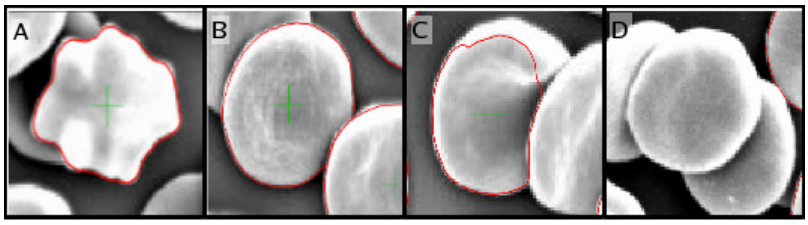

Figure 3: *A: Fully correct B: Minimally occluded C: Incorrect D: False negative*

The 'minimally occluded' category indicates a very slight overlap breaking the contour of this cell. Often, these cells are still correctly detected, as they are visually similar to non-occluded cells. It is reasonable to believe that they can be classified correctly, but because we cannot easily predict how much data the classifier algorithm will need, we categorise them seperately. Examples of each category can be seen in figure 3. We also counted the number of contours that were not found, but were nevertheless fully visible. These are false negatives.

The evaluation results can be found in table 3.

Dataset	Correct	Minimally Obscured	Incorrect	False Negative
A	112	9	1	2
B	81	8	6	0
C	77	11	5	2
D	91	16	12	0
E	93	9	0	0
F	72	11	7	0
G	106	13	9	0
H	129	10	1	0
I	106	11	3	0
J	89	12	5	4
Total	956	111	49	8

table 3: Results on ten randomly selected images

4 CONCLUSION

The results of our initial evaluation are promising, showing an 85% accuracy when merely considering the "perfect" contours, and 95% when also considering the contours that are minimally occluded. However, the usability of the minimally occluded contours will have to be tested in further stages of the project.

False negative rates are extremely low. The ellipse-based culling increases the risk of culling contours of non-elliptical cells, but no such case was found in the ten images evaluated. The false negatives that were found were mostly cells in the very high brightness ranges, with less contrast than the other cells.

The fact that this guided trace works so well is, for a large part, due to specific properties of our problem domain. The SEM produces low-noise, high-contrast images, and red blood cells shapes are smooth enough to have some benefit from a prediction using a local curve. The errors that do show up are often caused by noise within a cell. The trace "wants" to curve in towards the cell, and a noisy cell surface will create enough gradient information for it to do so. This kind of error makes up the majority of the incorrectly traced contours. Choosing the direction vector based on average gradient information over a number of steps in that direction causes a slight smoothing of the contour. Details smaller than the number of averaged steps may be lost.

References

Canny, J. F. (1986). "A Computational Approach to Edge Detection" *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **8**: 679–714.

Fitzgibbon, A., Pilu, M. & Fisher, R. B. (1999). "Direct Least Square Fitting of Ellipses" *IEEE Transactions on Pattern Analysis and Machine Intelligence*. **21**(5).

Simpson, L. (1989). "Blood from healthy animals and humans contains nondiscocytic erythrocytes" *British Journal of Haematology*. **73**: 561–564.