**June 2nd, 2006**

**[Cell segmentation](http://blogs.mathworks.com/steve/2006/06/02/cell-segmentation/" \o "Permanent Link to Cell segmentation)**

Blog reader Ramiro Massol asked for advice on segmenting his cell images, so I gave it a try. I'm not a microscopy expert, though, and I invite readers who have better suggestions than mine to add your comments below.

Let's take a look first to see what we have. I'm going to work with a cropped version of the [original](http://blogs.mathworks.com/images/steve/60/nuclei.png) so that the images aren't too big for the layout of this blog.

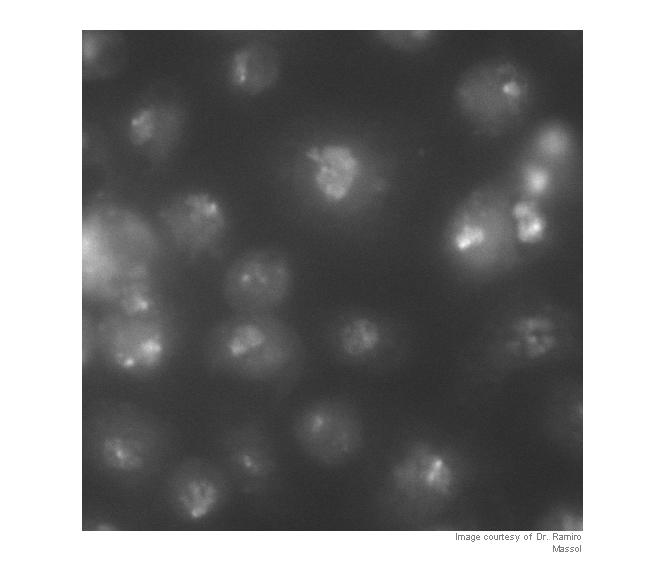
*Note: you can download the functions [imcredit](http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=9908&objectType=file) and [imoverlay](http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=10502&objectType=file) from MATLAB Central.*

I = imread('http://blogs.mathworks.com/images/steve/60/nuclei.png');

I\_cropped = I(400:900, 465:965);

imshow(I\_cropped)

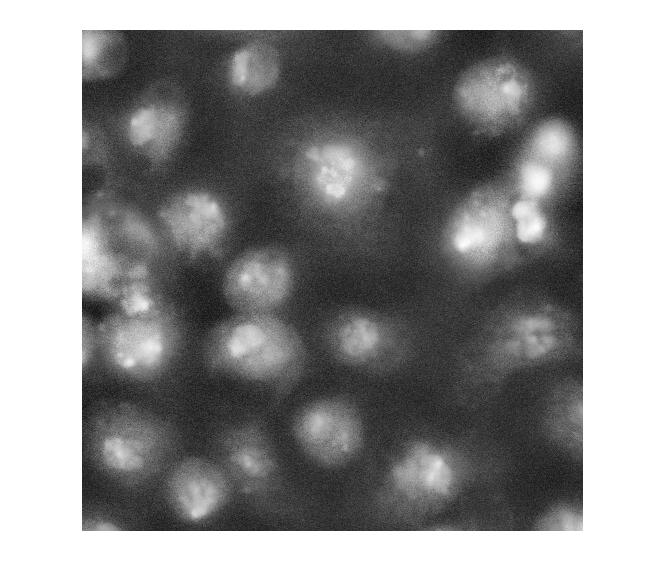
imcredit('Image courtesy of Dr. Ramiro Massol')



Strictly speaking, contrast adjustment isn't usually necessary for segmentation, but it can help the algorithm developer see and understand the image data better. This is a fairly low-contrast image, so I thought it might help. You can adjust the display contrast interactively with [imtool](http://www.mathworks.com/access/helpdesk/help/toolbox/images/imtool.html), or you can use an automatic method such as [adapthisteq](http://www.mathworks.com/access/helpdesk/help/toolbox/images/adapthisteq.html). adapthisteq implements a technique called *contrast-limited adaptive histogram equalization*, or CLAHE. (I always thought "CLAHE" sounded like it must be some Klingon delicacy.)

I\_eq = adapthisteq(I\_cropped);

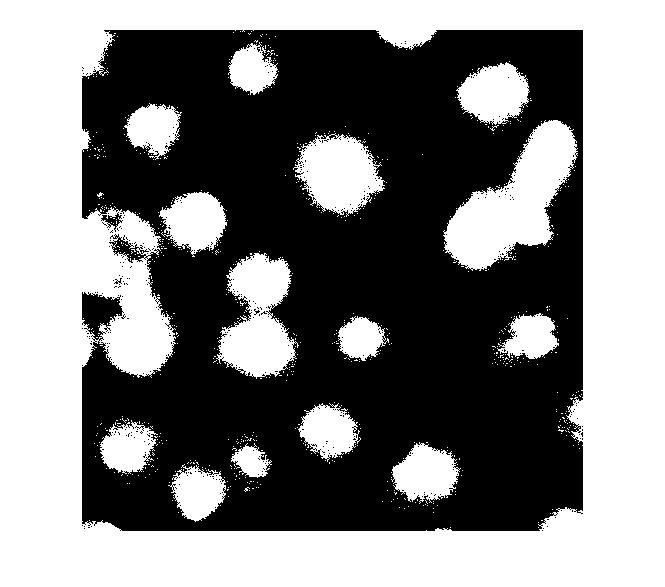
imshow(I\_eq)



So what happens if we just apply a threshold now?

bw = im2bw(I\_eq, graythresh(I\_eq));

imshow(bw)



Let's clean that up and then overlay the perimeter on the original image.

bw2 = imfill(bw,'holes');

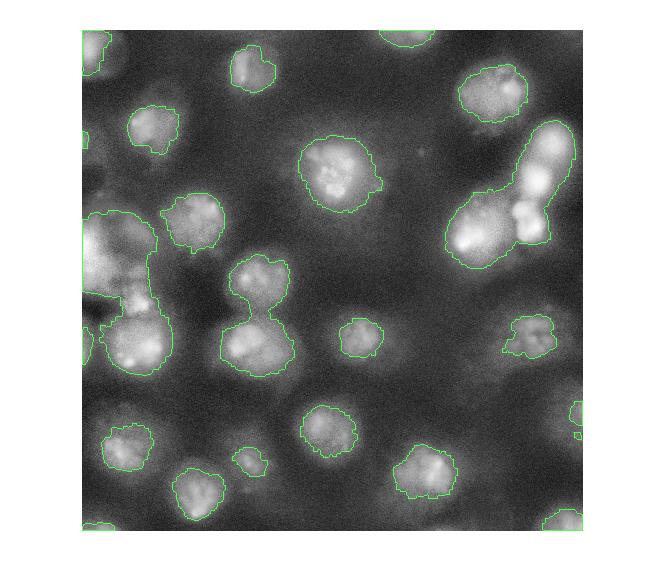
bw3 = imopen(bw2, ones(5,5));

bw4 = bwareaopen(bw3, 40);

bw4\_perim = bwperim(bw4);

overlay1 = imoverlay(I\_eq, bw4\_perim, [.3 1 .3]);

imshow(overlay1)



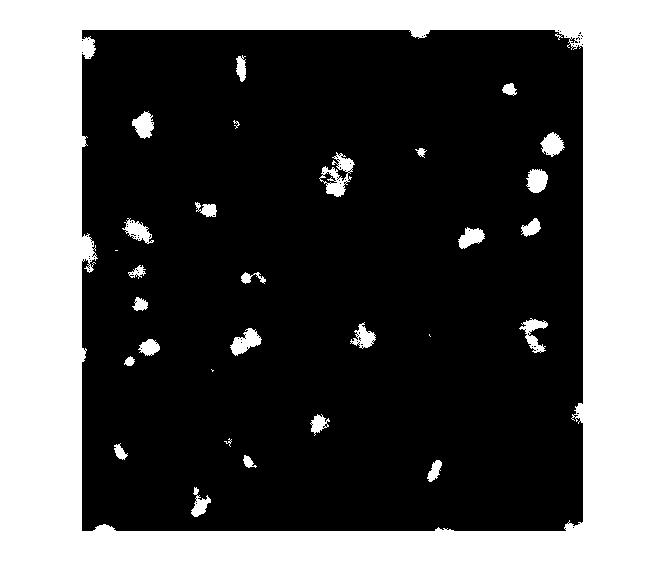
Now, I'm not familiar with these cell images, so I don't know exactly what I'm looking at. I assume some of these blobs need more help to be separated properly. One possible approach is called *marker-based watershed segmentation*. There's a [*demo*](http://www.mathworks.com/products/demos/image/watershed/ipexwatershed.html) of this idea on The MathWorks web site.

With this method, you have to find a way to "mark" at least a partial group of connected pixels inside each object to be segmented. You also have to mark the background.

Let's try to use the bright objects, which I assume are nuclei. The *extended maxima* operator can be used to identify groups of pixels that are significantly higher than their immediate surrounding.

mask\_em = imextendedmax(I\_eq, 30);

imshow(mask\_em)



Let's clean that up and then overlay it.

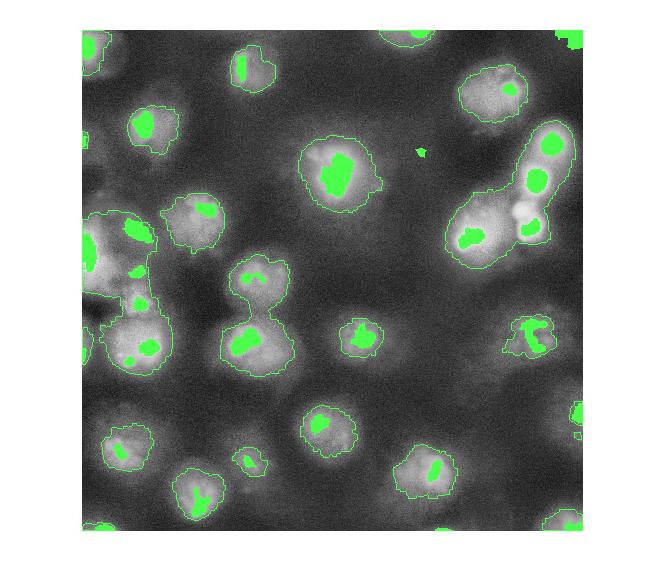
mask\_em = imclose(mask\_em, ones(5,5));

mask\_em = imfill(mask\_em, 'holes');

mask\_em = bwareaopen(mask\_em, 40);

overlay2 = imoverlay(I\_eq, bw4\_perim | mask\_em, [.3 1 .3]);

imshow(overlay2)



Next step: complement the image so that the peaks become valleys. We do this because we are about to apply the watershed transform, which identifies low points, not high points.

I\_eq\_c = imcomplement(I\_eq);

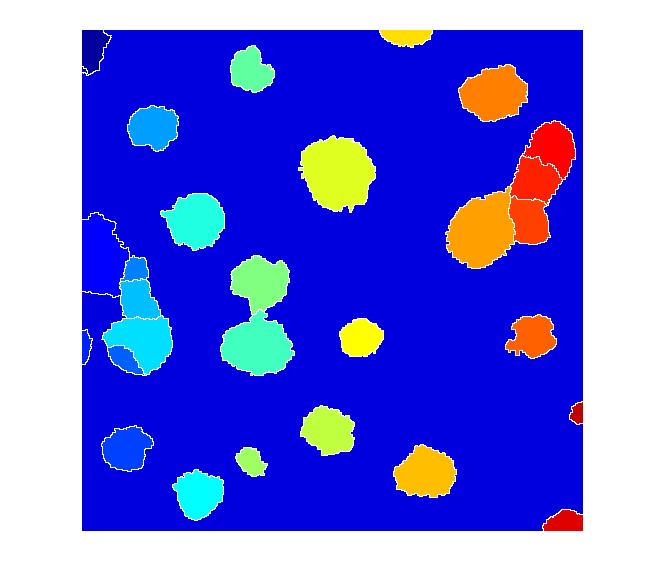
Next: modify the image so that the background pixels and the extended maxima pixels are forced to be the only local minima in the image.

I\_mod = imimposemin(I\_eq\_c, ~bw4 | mask\_em);

Now compute the watershed transform.

L = watershed(I\_mod);

imshow(label2rgb(L))



I don't know if this is a good segmentation result or not, but I hope some of the methods I've shown will give Dr. Massol some ideas to try.

Other readers ... if you have suggestions, I invite you to post your comments here.