Introduction to Data Mining Mouse Embryo Images

Jonas E. Malmsten

***Abstract*—Data mining is often associated with large databases containing numbers and statistical data, and how to make sense of it all. But similar techniques can also be applied to digital images, to categorize and identify objects within the images. This is an introduction to how some basic data mining techniques can be applied to digital images of mouse embryos captured using time-lapse microscopy. I use an image of a 2-cell mouse embryo, and extract coordinates for where each cell is located.**

***Index Terms*—Data mining, Embryo, Image Analysis, Mouse, Time-lapse microscopy**

I. Introduction

DATA MINING is the process of discovering patterns in large datasets so that new information about the data may be understood [1]. A digital image is a digital representation of an image, usually as a two-dimensional array of numbers – a dataset. Data mining on images involves finding patterns, such as lines, circles, or faces, within the image. A trivial task for humans when viewing an image but can be a daunting task for a computer, especially if the image contains lots of noise or is of poor quality. By noise I mean all data that does not describe what we are looking for. If we can eliminate that, then what we have left are relevant data that we need to make sense of.

In the next sections I will provide a brief introduction to Time Lapse Microscopy (TLM), and how images are represented digitally. Following that, I will describe several standard data mining techniques, and how they can be applied on TLM images of mouse embryos. The objective is to extract new information from these images. Specifically, I want to count the number of cells within a mouse embryo.

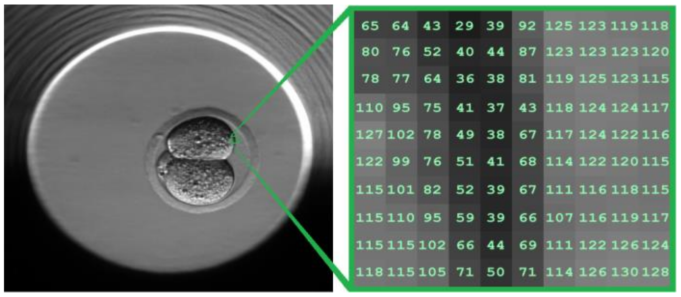
II. Time-Lapse Microscopy

Our setup is an incubator containing small dishes, called slides, each with up to 12 fertilized mouse embryos. It has a built-in microscope with an attached camera taking a picture of each embryo every 10 minutes, in 5 different focal planes. It takes about 3 days for a mouse embryo to grow from 1 cell to blastocyst (many cells), giving us 432 images for each embryo and focal plane. These images can be put in sequence to form a video showing a fast forwarded sequence of events. I will refer to each image in a sequence as a frame. In this study I will only make use of 2 images from one of these sequences. Namely, images 123 and 124 from focal plane 0 of slide D2015.01.12_S1216_I126. In other words, 1230 minutes, or 20.5 hours, after fertilization. The embryo consists of 2 cells at this point.

Submitted as partial completion of course DCS 861B Data Mining, summer 2016, at Seidenberg School of CSIS, Pace University, Pleasantville, New York.

III. Image Representation

Each image acquired from our TLM is 500 by 500 dots (pixels) grayscale, digitally represented as a 500x500 matrix, where each element is a number between 0 (black) and 255 (white), representing the brightness, or intensity, of each pixel (fig. 1).



65	64	43	29	39	92	125	123	119	118
80	76	52	40	44	87	123	123	123	120
78	77	64	36	38	81	119	125	123	115
110	95	75	41	37	43	118	124	124	117
127	102	78	49	38	67	117	124	122	116
122	99	76	51	41	68	114	122	120	115
115	101	82	52	39	67	111	116	118	115
115	110	95	59	39	66	107	116	119	117
115	115	102	66	44	69	111	122	126	124
118	115	105	71	50	71	114	126	130	128

Fig. 1. Left: Image of mouse embryo with two cells. Right: Visualization of digital image data, darker shades of gray are represented by lower numbers.

IV. Normalization

If an image is over-all very dark, or very bright, it is common to normalize the image to highlight as much detail as possible. Normalization is done by finding the intensities of the darkest pixel, P_{min}, and brightest pixel, P_{max}. These are then used to apply a linear transform (eq. 1) to every pixel, P, in the image, ensuring that the full contrast range of [0..255] is used within the image.

$$P_{new} = (P - P_{min}) * 255 / (P_{max} - P_{min})$$

Eq. 1: Image Normalization.

V. Background Subtraction

A common way to detect and locate moving objects in a video sequence is to use background subtraction [2]. Matrixes of two subsequent frames are subtracted from each other. Since intensity is a value between 0 and 255, the resulting intensity after subtraction can assume a value between -255 and 255. Because negative intensity values are not allowed, we will use the absolute value for each element in the resulting matrix. As it turns out, there is always some intracellular activity going on inside a live embryo. The area outside is fairly static and will be canceled out nicely (fig. 2). If the frames are very similar, the result after background subtraction may be very subtle. To enhance this, the result is always normalized.

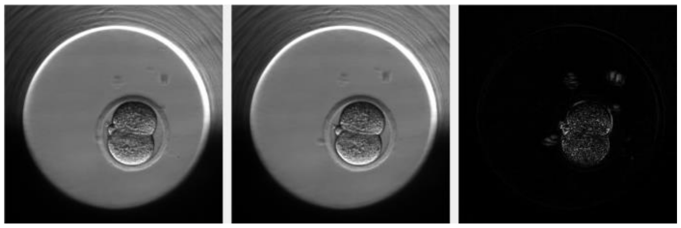


Fig. 2. Left and middle images are subsequent frames (123 and 124 from our sample data). Image on the right is the normalized result after subtracting the other two images.

VI. Region of Interest

Region of Interest (ROI) is an important term in image analysis. We want to remove as much noise as possible so we don't waste resources on analyzing data that is not useful. The result from background subtraction provides us with an easy way to pinpoint the location of the embryo, this is our ROI. We can use the formula from physics to calculate center of mass for a particle system (eq. 2) to give us the coordinates.

$$R = \frac{1}{M}\sum_{i=0}^{n} m_i r_i$$

Eq 2. R is a vector pointing to center of mass; M is total mass in system – sum of all intensities in the image; n is number of pixels in image – 500x500; r_i is a vector [x, y] pointing to a pixel in image, and m_i is the intensity of that pixel.

The shape of an embryo is always near circular, and has close to same radius for all mouse embryos. All we need is the center to know our ROI with a fixed radius. I've estimate the radius to be 90 pixels. Observing the resulting image in fig 2, we can see that some noise still remains above and to the left of the embryo. This will offset the center a little bit. One solution to this is to calculate center of mass twice, first time for the whole image to get an approximate center, and then again using only the pixels within a distance from the approximate center to refine the result. I've used 110 pixels for this distance. Once ROI has been established, we can use the coordinates in the original image for further analysis (fig. 3).

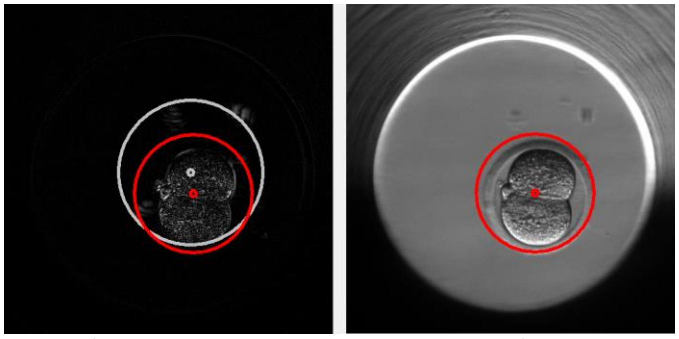


Fig. 3. Left: White circle is centered on centroid calculated from all pixels in the image. Red circle is centered on the centroid calculated from all pixels within the white circle. Right: Red circle applied using same coordinates on original image.

VII. Median Filter

Median filters [3] are used to remove high frequency noise from a signal. They are sometimes called salt-and-pepper filters for their effectiveness to eliminate scattered bright and dark spots, commonly introduced when transmitting a signal using an analog medium. For each pixel, it creates a list of intensities from pixels in a surrounding area and updates the intensity with the median from this list. Median is the value from the center of the list after sorting it in ascending, or descending, order. With some experimentation I found that using a circular area with a radius of 5 pixels produces a good result. I've also considered using a Gaussian filter, which is another common method for reducing high frequency noise. It produces a similar result, but median filter is better for preserving edges (fig. 4), which we will use later. A Gaussian filter is a convolution filter using a normal distributed kernel matrix. Convolution filters will be explained in next section about edge detection.

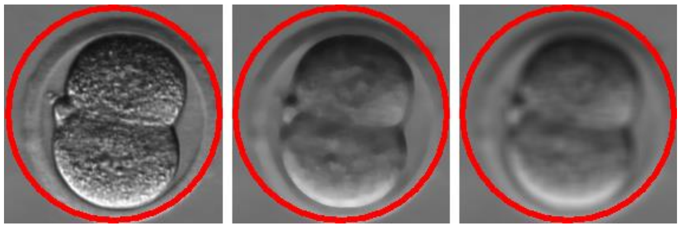


Fig. 4. Left: Original image zoomed in on ROI. Middle: Original image transformed using a median filter with 5-pixel radius. Right: Original image transformed using a Gaussian filter with 5-pixel radius.

Another possibility with median filter is to apply it over time axis, using the median for pixels over several frames. This will remove fast moving objects from images, something like the opposite of background subtraction.

VIII. Edge Detection and Convolution

There are many ways to detect edges within an image. They all somehow analyze the gradient of an image. Canny Edge Detector [4] is perhaps the most popular algorithm, but I will use another method based on a convolution filter [5]. Convolution filters have a wide range of uses in image analysis, edge detection is but one of them. The idea is to multiply each pixel, and its neighboring pixels, by a matrix called a kernel. Suppose our kernel matrix is:

$$K = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

It's not a standard matrix multiplication, but an element by element multiplication and addition. Consider the 2-dimensional image representation from Section III, described by a matrix M. The new intensity for each pixel after convolution filter with the kernel K would be:

$M_{new}[x, y] =$
K[0,0]*M[x-1, y-1] + K[1,0]*M[x,y-1] + K[2,0]*M[x+1,y-1] +
K[0,1]*M[x-1, y] + K[1,1]*M[x,y] + K[2,1]*M[x+1,y] +
K[0,2]*M[x-1, y+1] + K[1,2]*M[x,y+1] + K[2,2]*M[x+1,y+1]

The observant may notice that M[x-1, y] would end up outside the image if x=0. One can either add a small neutral border to the original image, or allow the resulting image to be slightly smaller to solve this problem. Since the border is not that interesting to us, I chose the latter.
Looking further at this kernel, it will amplify the middle pixel and decrease intensity by surrounding pixels. If all pixels are of same intensity, the result will be zero. The result will be highest when the middle pixel has higher intensity than the average of all surrounding pixels. Consider the analogy where the image describes a landscape height map where intensity is altitude. The filter will highlight edges of cliffs. On the opposite, corners of valleys will become negative values. Observing the image, because of how the light is projected, we are interested in both cliffs and valleys to get as many edges as possible. I will therefor use the absolute value for each pixel after convolution. To visualize the edge detection filter, we can display the new image M_{new}. But first we need to normalize it, because the convolution filter is not restricting intensities to the interval [0..255] (fig. 5).

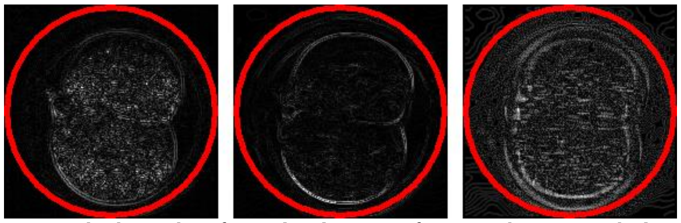


Fig. 5. Absolute values from edge detection after normalization, applied on images from fig 4. The middle image having been pre-processed using a median filter seemingly contains less noise.

IX. Thresholding and Clustering

I have found Circle Hough Transform, described in next section, to perform better on black and white images than grayscale. Thresholding is one of the simplest ways to convert a grayscale image into black and white. By experimentation I have determined 50 to be a suitable threshold; any intensity above this will be considered white. To enhance the image further, I will use clustering. The clustering algorithm is modified version of Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [6]. A white pixel is considered part of a cluster if there are at least 3 other white pixels within a radius of 2. If the total size of a cluster contains less than 20 pixels, I will consider it noise and remove it. After this is done, remaining clusters can be expanded using pixels around the edges with a lower threshold of 30. Cluster expansion is repeated until no more pixels are added (fig. 6).

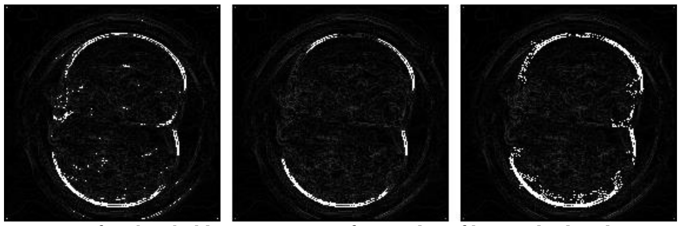


Fig. 6. Left: Thresholding on image after median filter and edge detection (middle from fig 5). Middle: Clustering applied on left image. Right: Expansion of clusters from middle image.

X. Circle Hough Transform

The naïve way to find a circle in an image would be to try all combinations. To determine if there is a circle with center at x, y, and radius r, we use the same algorithm as if we were to paint it. But instead of actually painting it, we read the intensity of every pixel that we would have painted. The sum of these intensities can be used as a measure of how likely there is a circle at the given coordinates and radius. To compare the likeliness of circles with different radius, we need to divide the sum by total number of pixels that would have been painted.
Trying all combinations for x, y, and r, would require a lot of computations. The Circle Hough Transform [7] is an algorithm to perform the same calculation with fewer operations. It uses that, after edge detection, there are probably many more dark than bright pixels, and we are only concerned with the bright ones.
Define an accumulator matrix with the same size as the image to be analyzed. For a given radius, and for every bright pixel in the original image, paint a circle at the corresponding coordinates in the accumulator matrix. But instead of painting the circle, use the intensity of the bright pixel, and add it in a circular shape in the accumulator matrix. The highest value in the accumulator matrix will be the most likely center of a circle with the given radius (fig. 7). To find circles with unknown radius, the process is repeated for every possible radius.

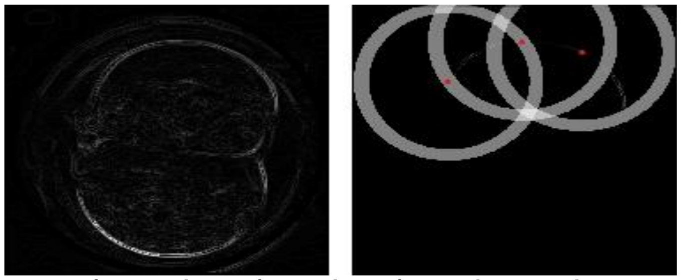


Fig. 7. Left: Original image for Hough transform. Right: Accumulator matrix with 3 circles painted, around centers marked in red, using addition. Notice how they amplify the center of the circle.

The right image from fig. 6 is suitable for Circle Hough Transform. I've applied it using a circle with a width of 2 pixels for each radii ranging between 20 pixels and image width divided by 2, increasing radius in steps of 2 pixels. The result is 37 images. These are then normalized together, such that the brightest point from the brightest image is used as P_{max} (eq. 2) for each image. After using a threshold of 190 the result is fairly conclusive; there are two circles with a radius of 55 pixels in the image (fig 8). There were other candidate circles with similar radii and position, but for illustration I selected the best looking ones. Their coordinates and radii can be used to paint them back on the original image (fig 9).

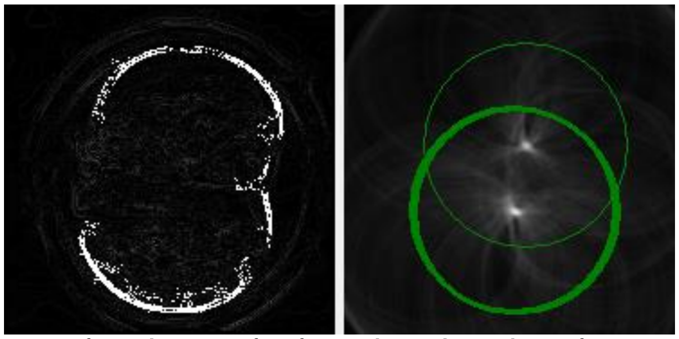


Fig 8. Left: Resulting image from fig 6. Right: Circle Hough Transform applied on left image using a radius of 55 pixels.

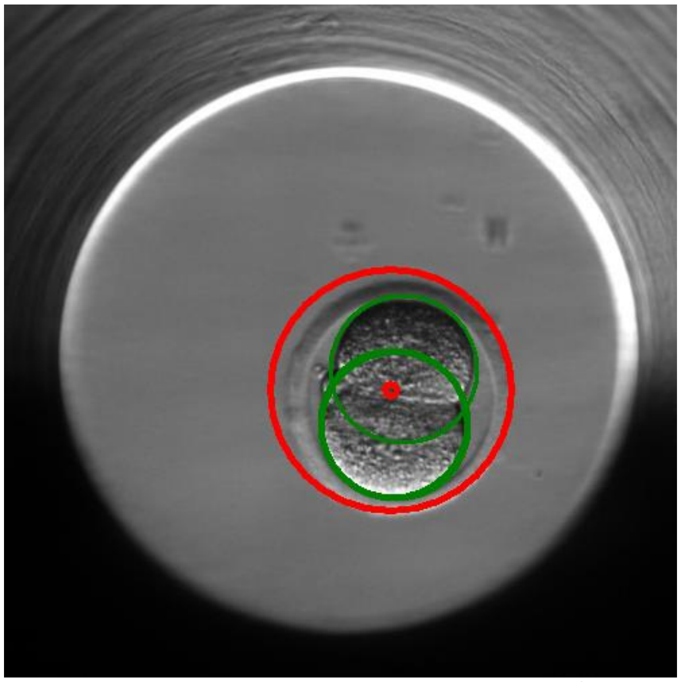


Fig 9. Red circle indicating ROI, green circles indicating final result produced by the various filters and transforms covered through-out this paper.

XI. Discussion and Future Work

This paper is an early proof of concept that it is possible to identify number of cells, and locate them, within embryos using classic data mining and image analysis techniques. There are many places where parameters have been adjusted to work specifically for the selected images. Future work may be to apply these techniques on other mouse embryo images and automate the finding of suitable parameters.

References

[1] https://en.wikipedia.org/wiki/Data_mining

[2] https://en.wikipedia.org/wiki/Background_subtraction

[3] https://en.wikipedia.org/wiki/Median_filter

[4] Canny, J. (1986). "A computational approach to edge detection." IEEE Transactions on pattern analysis and machine intelligence(6): 679-698.

[5] https://en.wikipedia.org/wiki/Kernel_(image_processing)

[6] Ester, M., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. Kdd.

[7] Ballard, D. H. (1981). "Generalizing the Hough transform to detect arbitrary shapes." Pattern recognition **13**(2): 111-122.