Image processing and analysis of cell migration

Nicolás González Muñoz

December, 2017

Introduction

Cell's movement analysis has very important applications in biology and medicine, but actually this analysis isn't automatic. The objective of this work is to create an algorithm that can do this analysis automatically, so it will require less effort and time. For this we pass through two steps. The first is the image segmentation. The second, the image analysis.

Definitions and suppositions

We can consider the matrix $A_i \in \mathcal{M}_{n\times m}(\mathbb{R})$, that corresponds to the *ith* frame of a video with k frames, where n and m are the pixel width and tall of the frame, respectively.
We will consider that each image contains only cells and tiny stains. Also we will suppose that the pixel value of background and cells are constant, except that little stains in the images, and the background covers, at least, 90% of the image area.

Image segmentation

The objective of this section is to get matrices of 0 and 1, where the 1 entries, that will be cells, have a neighbourhood of 0 entries, that will be background.
First we will consider $\bar{x}$ and σ, the mean and standard deviation of the pixel values in A_1, respectively. The image have, at least, 90% background pixels, so the mean should be very close of them.

We can affirm that background pixels are in the interval $[\bar{x} - 0.7\sigma, \bar{x} + 0.7\sigma] = I$, and it doesn't contain cell pixels.
We define the function $f : \mathcal{M}_{n\times m}(\mathbb{R}) \to \mathcal{M}_{n\times m}(\mathbb{R})$, that takes each entry of a matrix A and assign it to 1 if it isn't in I, or 0 if it do.

$$f\begin{pmatrix} \mathbf{0.8} & 0.15 & 0.15 & 0.15 & 0.15 \\ 0.15 & 0.15 & 0.15 & 0.5 & 0.15 \\ 0.15 & 0.15 & \mathbf{0.45} & 0.15 & 0.15 \\ 0.15 & 0.15 & 0.15 & 0.15 & 0.5 \\ 0.15 & 0.15 & 0.15 & 0.15 & 0.15 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\bar{x} = 0.188;\ \sigma \approx 0.138;\ I \approx [0.0914; 0.2846]$$

Then, we can define the morphological matrix $M \in \mathcal{M}_{n\times m}(\mathbb{Z})$ of a binary image how the matrix that maintains the 0 entries and assigns an integer number to a 1 entry and their neighbourhood of 1 entries.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 & 0 \\ 3 & 0 & 2 & 0 & 5 \\ 0 & 4 & 0 & 5 & 5 \end{pmatrix}$$

So, we can define the function $f_2 : \mathcal{M}_{n\times m} \to \mathcal{M}_{n\times m}$ that give us the morphological matrix associated to a matrix A, that is:

$$f_2(a_{i,j}) = \begin{cases} 0 \text{ if } a_{i,j} = 0 \\ \max_{\substack{i'<i \\ j'<j}} a_{i',j'} + 1 \text{ if } a_{i,j\pm1} = 1 \text{ and } a_{i\pm1,j} = 1 \end{cases}$$

We can consider then a function $g : \mathcal{M}_{n\times m}(\mathbb{R}) \to \mathcal{M}_{n\times m}(\mathbb{R})$, that maintains the 0 entries from de matrix, eliminates only one pixel in a morphological neighbourhood, and assigns the other entries to 1. That is:

$$g(a_{i,j}) = \begin{cases} 0 \text{ if } a_{i,j} = 0 \text{ or } (a_{i,j} = n \\ \text{and } \max_{\substack{i'<i \\ j'<j}} a_{i',j'} = n-1 \text{ and} \\ (a_{i\pm1,j} = n \text{ or } a_{i,j\pm1} = n)) \\ 1 \text{ in other case} \end{cases}$$

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 2 & 2 & 0 \\ 0 & 2 & 2 & 0 & 0 \\ 3 & 0 & 2 & 0 & 5 \\ 0 & 4 & 0 & 5 & 5 \end{pmatrix} \to \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 2 & 0 \\ 0 & 2 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 5 & 5 \end{pmatrix} \to \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

So we can subtract the matrix obtained by this operations to the binary image's matrix to obtain a new matrix with segregated points, so it will be easier to analyse.

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Image analysis

Once we get segmented images, we want to find some parameters from there. This parameters are:

- Euclidean distance: The distance on the x axis travelled from the initial point to final frame
- Total distance
- Mean velocity
- Persistence: Euclidean distance divided by total distance
- Directionality: Cells that are in a range of 60 degrees in all their trajectory

First, to follow a cell we need to consider consecutive frames, so we will suppose that the coordinates corresponding to the studied cell in the image $i + 1$ are the nearest non-zero point to their last position, in the image i.
However, the matrices doesn't have only cells. They also have some pixels that correspond to stains in the background, so we need to ensure that we are following a cell, and not a background pixel.

For our purpose, we will define a function $h_1 : \mathbb{N}^2 \times \mathcal{M}_{n\times m}(\mathbb{R}) \to \mathbb{R}^2$, that takes as argument the coordinates of a point in a image and returns the nearest non-zero entry in the image's matrix. If there are more than 1 minimum points, we will chose the one that has the minimum coordinates.

$$h((a,b),(a_{i,j})) = argmin\{i+j : (i,j) \in argmin\{i-j : (i,j) \in argmin\{(i-a)^2+(j-b)^2 : a_{i,j} \neq 0\}\}\}$$

$$h_1\left((3,3), \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{0} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix}\right) = (1,1)$$

Then, we can define other function
$h_2 : \mathbb{N}^2 \times (\mathcal{M}_{n\times m}(\mathbb{R}))^k \to (\mathbb{R}^2)^k$, where

$$h_2((i,j),(M_1,\ldots,M_k)) = (h_1((i,j),M_1), h_1(h_1((i,j),M_1),M_2),\ldots)$$

That is, h_2 is the function that follows a cell along all the video.
In the case that most of the time the point doesn't move between consecutive frames, we will consider this as a background point, so we aren't going to consider that cases.
Once we get the trajectory of the cells we are interested in, we can do the usual analysis of distance, velocity and other parameters.

Results

In the reality, the programming of this algorithm isn't very complicated, so we programmed that to see and compare the results with real videos of cells.
There are the results of the image filtering for a particular image.

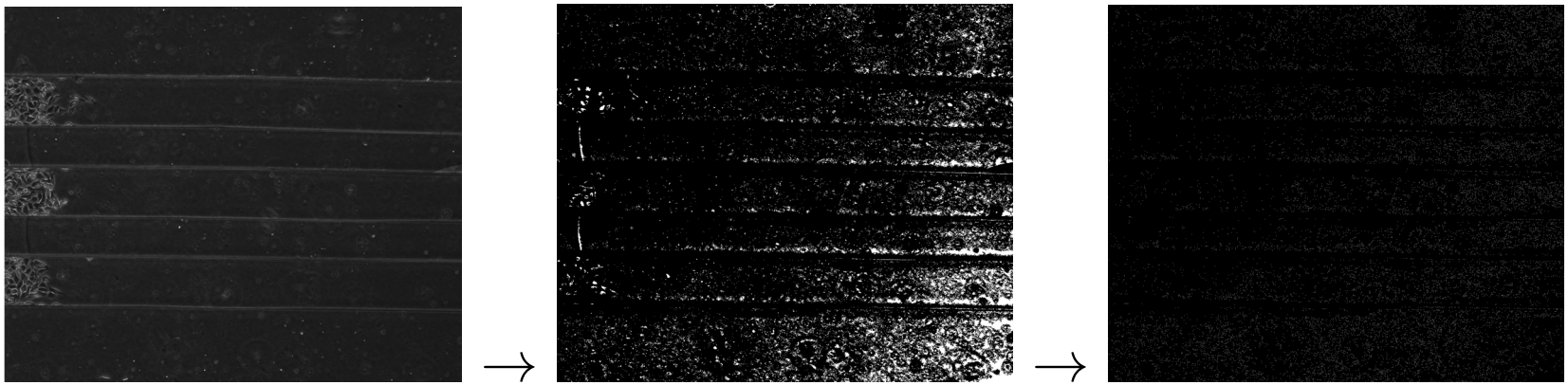

Figure: Principal steps of image segmentation

For the image analysis, there are general results for 9 cells, 3 for each channel.

- Mean euclidean distance: -1.4306[μm]
- Mean trajectory length: 183.697[μm]
- Mean final distance: 13.4247[μm]
- Mean velocity: 0.172485[$\mu m/min$]
- Mean rapidity: 0.0126054[$\mu m/min$]
- Mean persistence: 0.00922666
- Directionality: 0%

The time required by this program to analyse this image is around 12 minutes.

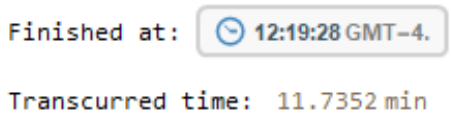

Figure: The reduction of time costs are very notorious

For the image analysis, there are the graphics of the displacement of 9 cells, 3 for each channel and centered in the origin.

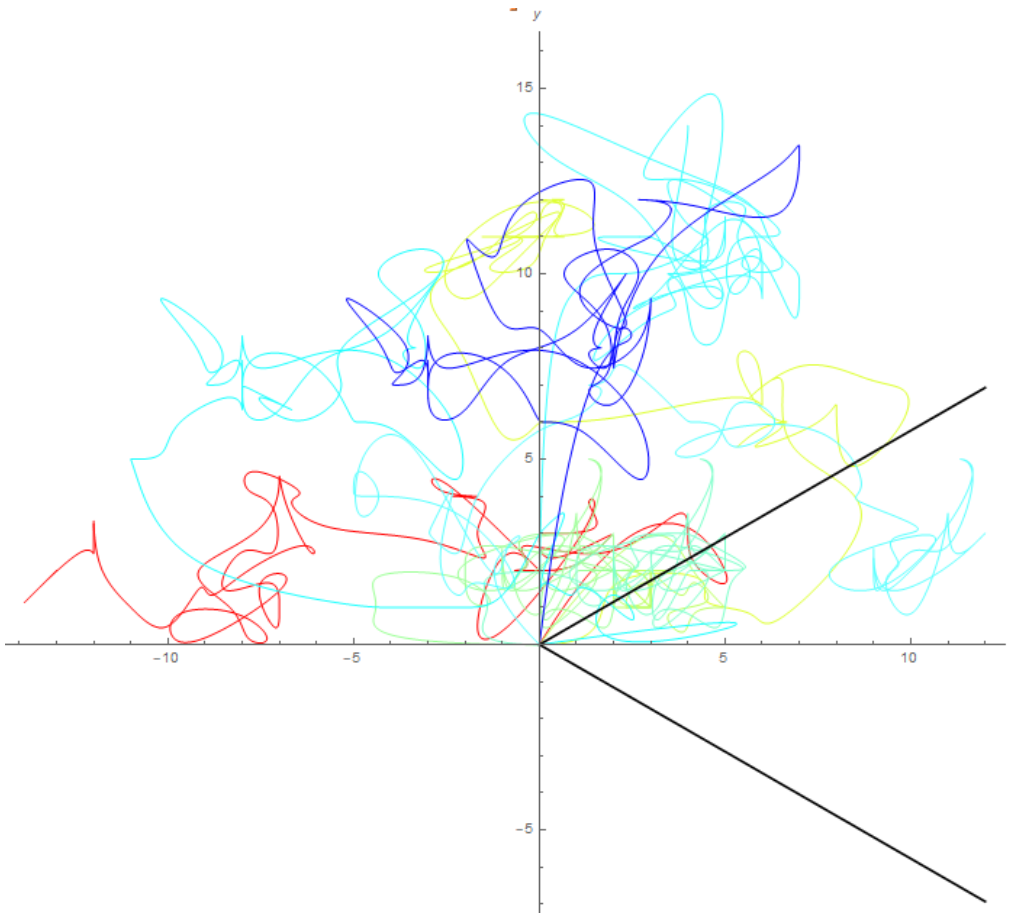

Figure: The graphic shows a spline approximation of first degree of the cells trajectory

Conclusions

This algorithm works very well and is very quick in comparison with the manual analysis, but sometimes it can miss information by deleting cells that doesn't contrast too much with the background. However, the information that it loses is small, and it can be controlled by certain constants