Projet 4GMM
Cell segmentation in biological image

Nicolas Dublé, Aimée Simcic–Mori, Emna Jaoua
Tutors: Pierre Armand Weiss[1], Renaud Morin[2]

May 2020

[1] *CNRS researcher at Institut de Mathématiques de Toulouse (UMR 5219)*
[2] *École nationale supérieure d'électrotechnique, d'électronique, d'informatique, d'hydraulique et des télécommunications (ENSEEIHT), Toulouse, France*

Introduction

(–¿https://www.overleaf.com/read/ktckvyhvckjz lien de l'overleaf explication comment écrire une bibliographie)
In the past few years, humans have made a considerable progress in medical research thanks to new technologies. Today we can observe how drugs work when applied to cells and we can also detect tumours thanks to cell segmentation algorithms. Cell segmentation consists in the detection of the number and the shape of cells in biological images. However, every observation is made on two-dimensional (2D) images, and rarely on three-dimensional (3D) images. Indeed, passing from a 2D image to a 3D image drastically increases the number of pixels that the algorithm has to analyse. In other words, we are faced with two problems : the memory constraint of the computer and the very long computing time. The aim of this project is to create a cell segmentation algorithm for 2D images by ourselves in a cost efficient way. In section 1 we will present two methods of cell segmentation, which the second one is the improved version of the first one. Both methods consist on minimization problem that we solve with image processing techniques, and are thought in limited memory space. Section 2 shows results obtained with both methods and we make some comments on the observations. Then in section 3 we analyse the pros and cons of each method. Finally, in section 4 we discuss about perspectives and ideas for cell segmentation algorithm for 3D images.

1 Methods

In this section, for each method we present the theoretical part of our problem, in other word its mathematical background.

1.1 Method 1

We begin by conceptualizing with a quite basic first method.

1.1.1 Construction of the problem

Let u be a $m = n \times n$ 2D biological image containing n cells, with $n \in \mathbb{N}$. (see Figure 1)

Figure 1: The 2D biological image u that we are going to use in this report

Let $D \in \mathbb{R}^{m \times (p \times m)}$ represents a dictionary that stocks p possibilities of shape of cells in m possibilities of position. We consider that a cell is mathematically represented by its tilt angle, its diameter and its position (the center of the cell). Then each column of the dictionary is dedicated to one type of shape (tilt angle and diameter) of cell. However one cell can be on any pixel of a m size image. Thus, we model each type of shape of cell that can be on the m possibilities of position by a vector $D_i \in \mathbb{R}^{p \times m}$ with $1 \leq i \leq p$. Each vector contains in order information about the tilt angle, the diameter, and the position. The information about the position is coded by a vector which all of his coefficients are null, except only one that will be 1 which corresponds to the pixel where the cell is located. (see Figure 2)
The matrix D will be used to define a vector $x \in \mathbb{R}^m$ which is the solution of the following equation:

$$Dx = u. \tag{1}$$

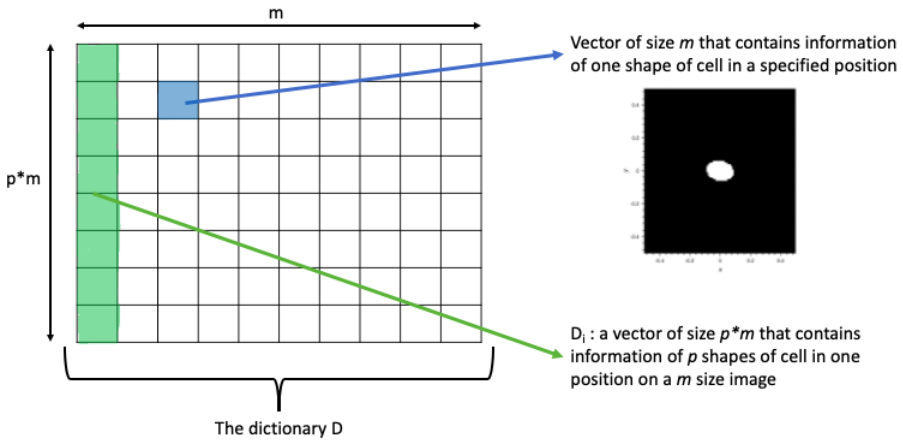

Figure 2: Schema of the dictionary D

The vector x represents the weight, i.e. the intensity of the luminosity (because cells do not appear with the same intensity of luminosity on real images, see figure 1), of each cell's image D_i with $1 \leq i \leq p \times m$. We want x to be parsimonious because we want to reproduce the initial image with the least patterns possible. In fact, (1) has difficult constraints because it is impossible to reproduce exactly u with $D \times x$. That is why we will approach the solution within the meaning of the norm 2.
Thus, this equation is rewritten into the following minimization problem with constraints below:

$$\inf_{\substack{\|x\| \leq \gamma \\ x \geq 0}} \frac{1}{2} \|Dx - u\|^2 \tag{2}$$

where γ is a maximum threshold which imposes x to be parsimonious. More precisely it fixes the maximum of the intensity of the cells. Why do we need a maximum threshold ? Because when two cells are overlapped, the overlapped part appears with a very high intensity compared to the intensity of one simple cell, thus our algorithm may make a mistake by counting only one cell instead of counting 2 overlapped cells.
We then convert the constrained problem (2) into the equivalent unconstrained problem (3) given below:

$$\inf_{x \geq 0} \frac{1}{2} \|Dx - u\|^2 + \lambda \|x\| \tag{3}$$

where λ is a regulation parameter and must be positive. Here, the parameter λ must be small enough in order to make x parsimonious.
We will define the values of γ and λ later, on part 3 of the report.

1.1.2 Computing tools

We first introduce an algorithm that will help us to solve the optimization problem with constraints (2).

Theorem 1 (Frank-Wolfe Algorithm (Conditional Gradient))**.** *Let D be a compact and convex set. Be $f : D \to \mathbb{R}$ a convex differentiable real-valued function. The Frank-Wolfe Algorithm solves the optimization problem:*

$$\min_{x \in D} f(x) \tag{4}$$

The algorithm is:
Initialization : *$k = 0$ and $x_0 \in D$*
Step 1: *Direction finding subproblem: $\min_{s \in D} s^T f(x_k)$ such as $s_k = \text{argmin}_{s \in D} \langle \nabla f(x_k), s \rangle$*
Step 2: *Step size determination: Set τ such as*

$$\sum_{k=1}^{+\infty} \tau_k = +\infty \quad \text{and} \quad \sum_{k=1}^{+\infty} \tau_k^2 < +\infty$$

subject to $0 \leq \tau \leq 1$.
Step 3: *Update: Let $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \tau(\mathbf{s}_k - \mathbf{x}_k)$, let $k \leftarrow k + 1$ and go to Step 1.*

[Wikipedia contributors, 2020]

In this algorithm we can see that we need a dictionary and its conjugate transpose. To do so, we need to determine the gradient of the problem's function.

Definition 2. *Let $A \in M_{n,m}(\mathbb{R})$. Its transpose matrix is denoted $A^T \in M_{m,n}(\mathbb{R})$.*

Now, we apply this algorithm to our problem.

Theorem 3 (Gradient)**.** *Be $D \in \mathbb{R}^{d*n}, u \in \mathbb{R}^n$, $f(x) = \frac{1}{2} \|Dx - u\|_{L^2}^2$.*
Its gradient is denoted

$$\nabla f(x) = D^T(Dx - u) \tag{5}$$

Proof. $f(x + h) = \frac{1}{2} \|D(x + h) - u\|_{L^2}^2$
$f(x + h) = \frac{1}{2} \langle D(x + h) - u, D(x + h) - u \rangle$
$f(x + h) = \frac{1}{2} \langle (Dx - u) + Dh, (Dx - u) + Dh \rangle$
$f(x + h) = \frac{1}{2} (\langle Dx - u, Dx - u \rangle + 2\langle Dx - u, Dh \rangle + \langle Dh, Dh \rangle)$
$f(x + h) = f(x) + \langle D^T(Dx - u), h \rangle + o(\|h\|^2)$ □

Remark. *We can observe that the expression of the gradient is very explicit. It is the product between the transpose matrix of the dictionary and the problem's formula. In this way, the problem's formula reflects the error of the approximation of the image Dx compared to the real image u. As a result, the gradient represents the scalar product between the error and each cell of the dictionary that is added. Our aim is to maximise this scalar product.*

Therefore, we need to create the dictionary in order to implement the Frank-Wolfe algorithm.

1.1.3 Construction of the dictionary

As mentioned above, D is a collection of m vectors that contain information about cells. Each component of a vector $D_i \in \mathbb{R}^{p\times m}$ represents an image of a cell with a defined tilt angle, diameter and position. These parameters take values between 0 and 90°for the tilt, 10 and 16 pixels for the width and the length (diameter). The number of values taken for each parameter has to be adjusted to be large enough to solve the problem (minimizing the error $Dx - u$) but without expanding the problem too much which would make computation time too slow. Therefore, the purpose of the dictionary D is to discretizased the given image u. However, to code such dictionary that stocks $p \times m$ possibilities of shape of cell with m possibilities of position for each shape of cell will take too much space in the computer's memory, and its computation will be very time consuming.

Here is a solution. We will not express the dictionary directly, but Dx will be constructed implicitly by convolutions of each type of cell by passing through the Fourier domain.

Let us introduce the Fourier transform of a 2D signal defined over a discrete finite grid of size $M \times N$, in other words the discrete two-dimensional Fourier transform of an image array of size $M \times N$.

Definition 4. *The 2D discrete Fourier transform of a discrete function* $f : \mathbb{Z} \to \mathbb{C}$ *is*

$$\hat{f}[u,v] = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f[m,n]e^{-i2\pi(\frac{u}{M}m+\frac{v}{N}n)} \tag{6}$$

And its Fourier inverse transform is

$$f[m,n] = \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} \hat{f}[u,v]e^{i2\pi(\frac{u}{M}m+\frac{v}{N}n)} \tag{7}$$

Definition 5. *The expression of the circular convolution of two two-dimensional discrete functions* f *and* g *is*

$$f * g[x,y] = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f[m,n]g[x-m,y-n] \tag{8}$$

We also use the theorem given below to construct the dictionary.

Theorem 6 (Convolution theorem)**.**

$$f * g[x,y] \iff \hat{f}[u,v]\hat{g}[u,v]$$

$$f[x,y]g[x,y] \iff \hat{f} * \hat{g}[u,v]$$

IMAGE DU CODE ?
+ Meilleure explication de comment on obtient un dictionaire de manière implicite sans trop prendre de mémoire

1.1.4 Construction of the conjugate transpose of the dictionary

Now that we have constructed our dictionary D, we turn to constructing its conjugate transpose, because we also need it for the Frank-Wolfe Algorithm. Here is some definitions of notations we use.

Definition 7. *The adjoint of a function f is denoted f^*.*

Definition 8 (Hermitian symmetry)**.** *A function staisfies the Hermitian symmetry if*

$$f(x) = f^*(-x).$$

For the construction of $D*$, we encounter the same problem as for the construction of the dictionary D. We cannot code $D*$ explicitly since D cannot be explicit. To do so, we use the convolution theorem to implicitly code D^*.

Let $D = [D_1, ..., D_p]$ be our dictionary where $D_i \in \mathbb{R}^{n*n}$ is a convolution, i.e. $D_i u = h_i * u$

Thus, we have : $D \begin{pmatrix} x_1 \\ . \\ . \\ . \\ x_p \end{pmatrix} = \sum_{i=1}^{p} h_i * x_i$

and therefore our problem is to express $D^* = \begin{pmatrix} D_1^* \\ . \\ . \\ . \\ D_p^* \end{pmatrix}$.

If we remember that a convolution is diagonalized by the Fourier transform, i.e. $D_i = F^* \Sigma_i F$ with $\Sigma_i = diag(\hat{h_i})$ where $\hat{h_i} = F h_i$, then we have $D^* = F^* \Sigma_i^* F$, where Σ_i^* is the matrix Σ_i which we conjugated the diagonal elements, i.e. $\Sigma_i^* = diag\left(\widehat{\bar{h}}\right)$.

+ meilleure explication de D transposée
Now that we have every elements needed to apply the Frank-Wolfe algorithm, we turn to the application and the observation of our obtained results.
+ ajouter plus de choses à la conclusion ?

1.2 Method 2: Improved method

In order to improve the results we had with the first method, and to resolve the major problem we encountered : counting more cells then they really are (cf. Results), we will introduce new parameters to construct a new problem.

1.2.1 Construction of the problem

In this section, we keep almost the same construction of the problem as for the Method 1, so we keep the same dictionary D and the same gradient-descent algorithm. However, we will consider a biological image as the sum of two images: the background and an image of the cells. Hence, we introduce in this part the matrix P, such that Py is a sum of polynomial functions.
Therefore, the equation we have to minimize at each iteration is, for a fixed x :

$$y_n = \underset{x\in\mathbb{R}^3}{\arg\min}\ \frac{1}{2}\|Dx + Py - u\|_2^2 + \alpha\|x\|_1$$

Which is equivalent to the least square problem:

$$\begin{aligned} y &= \underset{y\in\mathbb{R}^3}{\arg\min}\ \|Dx + Py - u\|_2^2 \\ &= \underset{y\in\mathbb{R}^3}{\arg\min}\ \|Py - (u - Dx)\|_2^2 \\ &= \underset{y\in\mathbb{R}^3}{\arg\min}\ \|P^*Py - P^*(u - Dx)\|_2^2 \\ &= (P^*P)^{-1}\, P^*(u - Dx) \end{aligned}$$

Then, we obtain y_n that depends on x_n. As above, we use the Franck-Wolf algorithm to determinate $x_(n+1)$, this time using y_n and x_n in this iteration.

2 Results

In this section we show two types of results. The first one is made on a synthetic cell image and the second one on a real cell image.

2.1 Application on synthetic cell image

Let us begin with the synthetic cell image. First of all, we generate a synthetic cell image randomly with the dictionary.

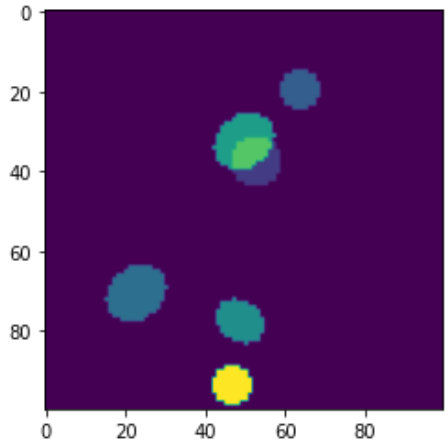

Figure 3: Synthetic cell image u used for experimentation

In this image, we randomly generate a random quantity of cells with a random shape. We can count with our own eyes 6 cells. If our algorithm works, we should find the same number of cell in the obtained segmented cell image. See Figure 4 for the obtained segmented cell image with our code.
Observations:
Number of iterations:
The algorithm counts 6 cells with very high coefficients and 135 cells with very infinitesimal coefficients.
Convergence ?

Comments:
As its appears, we obtain exactly the same image as the synthetic base image, differences cannot be seen with our own eyes. However, if we compare the number of cells counted by our algorithm to the number of cells we count by ourselves on the base image, we can say that the algorithm see details invisible to the naked eye.

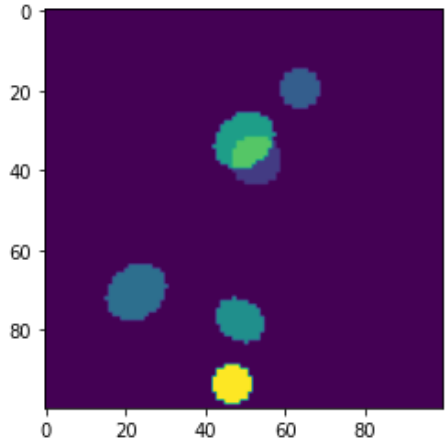

Figure 4: Reconstructed image Dx obtained with Figure 3

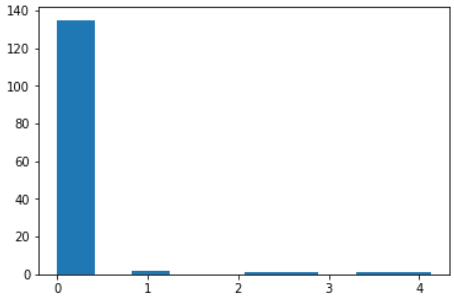

Figure 5: Histogram of weight of segmented cells

2.2 Application on real cell image

Now let us look at the result we obtain on real cell image. Here is the cell image u we used for our experimentation (see Figure 3). We can count with our own eyes that there are 14 cells more or less on this picture.
This is the image we obtain with the Frank-Wolfe algorithm (see Figure 4).

Observations:
Number of iterations:
Number of cells counted:
Convergence ?

Comments:
We can see that the shape of cells is similar in the both figure. Our algorithm manages to recognize where are cells in the picture u. However, the number of

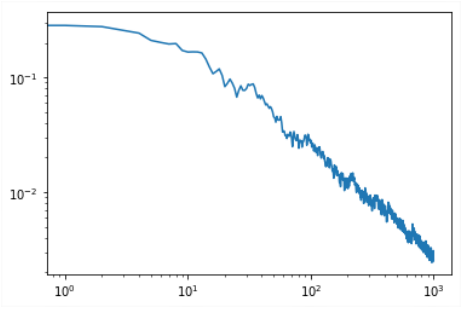

Figure 6: Rate of converge with the norm $\mathcal{L}^2$

Figure 7: Cell image u used for experimentation

counted cells is way too great than expected.

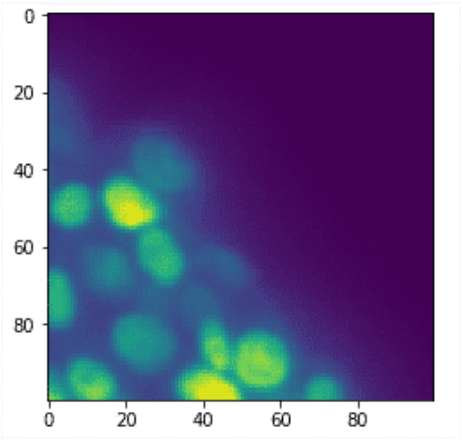

Figure 8: Segmented cell image Dx obtained with Figure 5

3 Analysis

In this section, we presents the strengths and the weaknesses of each method from results we obtained in the previous section.

3.1 The pros and cons of Method 1

3.1.1 Strengths

As we mentioned in the previous part, our first version of the algorithm is efficient on synthetic images.

3.1.2 Weaknesses

We encountered one major problem. The algorithm that we coded works well on synthetic cell images, since the cells have a perfect elliptical shape the algorithm can detect them very easily. However, when we apply it to real cell images, it counts more cells than there are in it. Our algorithm can only recognize ellipses, whereas in real life, cells do not have a perfect elliptical shape. As a result, to tend to be more similar to the real picture, the algorithm adds more and more segmented cells to form the shape of one real cell.
A way to get around this problem is to add another variable y that regulates the background of the reconstructed image. Then the algorithm will not add more and more segmented cells indefinitely but will add some background. This is actually the Method 2.

3.2 The pros and cons of Method 2

3.2.1 Strengths

3.2.2 Weaknesses

4 Discussion

In this part we
Segmenting cells is a difficult task since each cell does not have the same shape. There are many others possibilities to implement a cell segmentation algorithm. For example, we could use thin-plate spline instead of our variable y to regulate the background. The thin-plate spline is the two-dimensional analog of the cubic spline in one dimension. It has the form of

$$\Phi(x) = \|x\|_2^2 \log(\|x\|_2). \tag{9}$$

However, the use of thin-plate spline is mathematically very technical, so very complex to implement.
The ultimo aim of our program would be to improve it in order to apply on three-dimensional (3D) cell images. Indeed, 3D images are more useful for medical research since 3D cell representation most closely reflects the development of a cell in a human body.

References

[Wikipedia contributors, 2020] Wikipedia contributors (2020). Frank–wolfe algorithm — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Frank%E2%80%93Wolfe_algorithm&oldid=951919410.