Dasnoy Damien 2013-2014

PROJ – H - 401

Université libre de Bruxelles

Détection de mutants dans une population de C-Elegans

Table des matières

Détection de mutants dans une population de C Elegans

1. Introduction

L'objectif du projet était de créer un système automatisé pour détecter dans une population de C Elegans des mutants particuliers sur base d'images de fluorescence. La molécule radioactive que l'on injecte aux vers s'attache à une protéine qui est utilisée par le ver dans un neurone particulier. Le but étant donc de voir quelles sont les mutations qui apportent un comportement différent pour cette protéine vis-à-vis de ce neurone.

Sur les images il faudra donc pouvoir mesurer l'intensité de la fluorescence, le nombre de blobs (taches de fluorescence le long du neurone), leur taille, la taille du neurone etc. Et par la suite récupérer ces informations pour pouvoir comparer la population entière afin de trouver les cas particuliers.

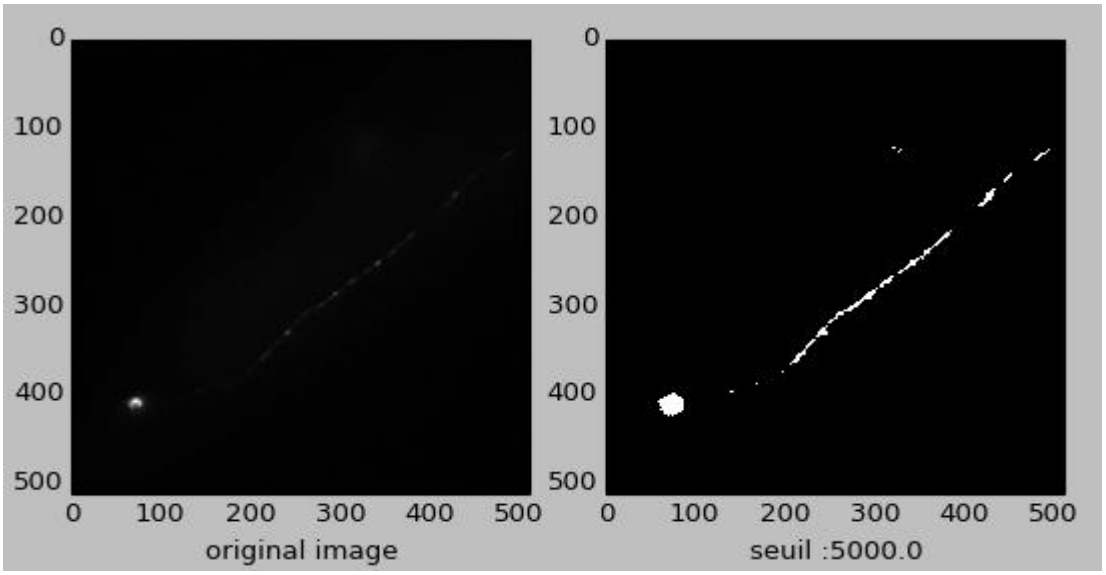
2. Traitement des images

a) Seuillage

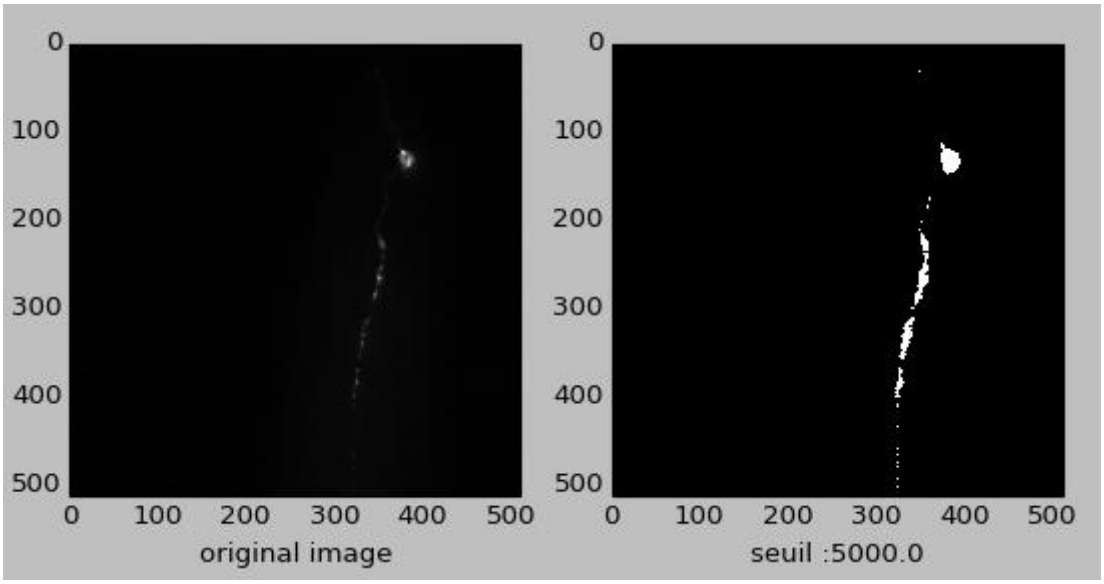
Pour pouvoir identifier les blobs comme des objets différents du fond, il faut commencer par seuiller l'image à l'aide de seuils binaires noir/blanc. Passer par une première étape de seuillage est bien souvent utilisé en imagerie quand il s'agit de séparer des objets. Dans un premier temps j'ai testé des fonctions de seuil qui déterminent la valeur de seuil utilisée de manière automatique en fonction de certains paramètres comme le seuil global d'Otsu ou local de Bernsen. Mais malgré la possibilité de faire varier certains paramètres, les résultats n'étaient pas satisfaisants. De plus, comme l'objectif est de comparer les images, utiliser un seuil automatique qui varie, même légèrement, pour chaque image risque fort de fausser les résultats. J'ai donc opté pour un seuillage global fixe où la valeur du seuil peut être donnée par l'utilisateur. Une valeur de seuil qui fonctionne bien et que j'ai utilisée pour la plupart de mes tests est 5000 (les images sont en tiff, les niveaux de gris sont codés sur 16 bits, cela fait 65 536 niveaux de gris)

Dans l'exemple 1 ci-dessous, le seuil permet de bien séparer le neurone du fond mais un peu de bruit persiste. Pour l'exemple 2 par contre, le seuil fonctionne très bien dés le départ car l'image est dépourvue de bruit au dessus de ce seuil.

Exemple 1



Exemple 2



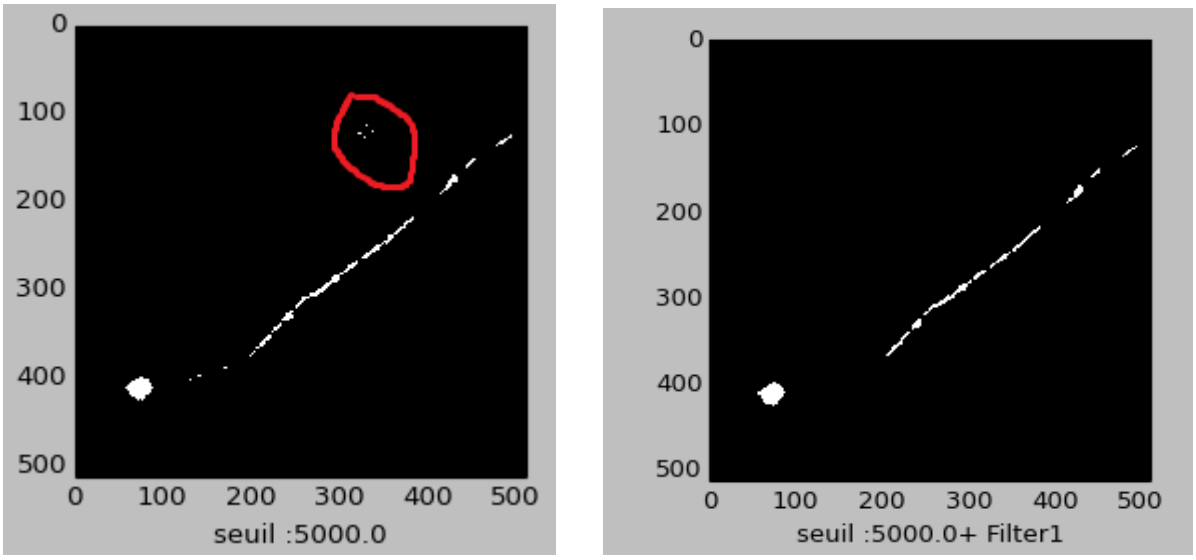
b) Filtrage

Pour certaines images, le seuillage ne suffit pas. Du bruit peut parfois se trouver au dessus du seuil et donc être considéré comme faisant partie du neurone. Le filtre est donc la pour essayer de retirer ce bruit. Le filtre à cet endroit là du traitement (il y a un deuxième filtre plus loin dans le traitement de l'image) est un filtre qui détecte et retire les petits objets (remove_small_objects). Il vient de la librairie morphology de skimage. Cela fonctionne bien pour enlever le bruit indésirable, mais cela enlève aussi des objets utiles appartenant au neurone, principalement aux extrémités ou la fluorescence se réduit souvent à quelques points isolés. A nouveau, il faut appliquer le même traitement à chaque image pour ne pas

fausser la comparaison. Il faut choisir entre garder du bruit et toutes les données utiles (après seuillage) ou éliminer le bruit et perdre une partie du neurone.

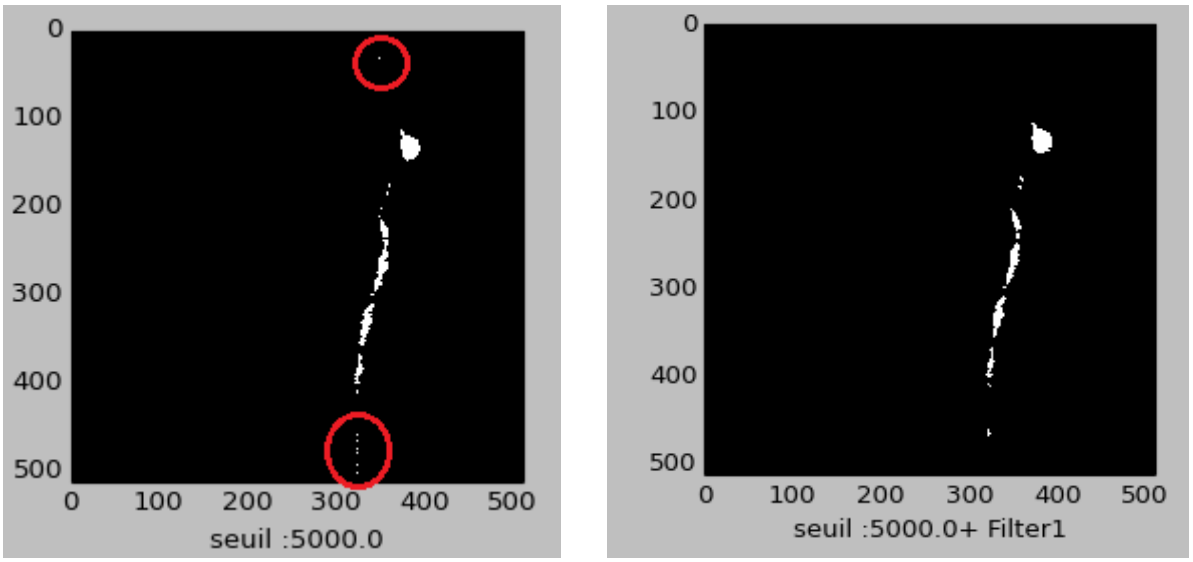
Dans le premier exemple ci-dessous, on utilise le filtre et on parvient à retirer le bruit indésirable en perdant également des petits objets dans l'axone.

Exemple 1



Tandis que dans le deuxième on perd l'information du bout de l'axone et de la dendrite alors qu'il n'y avait pas de bruit indésirable. L'utilisation du filtre n'était pas nécessaire.

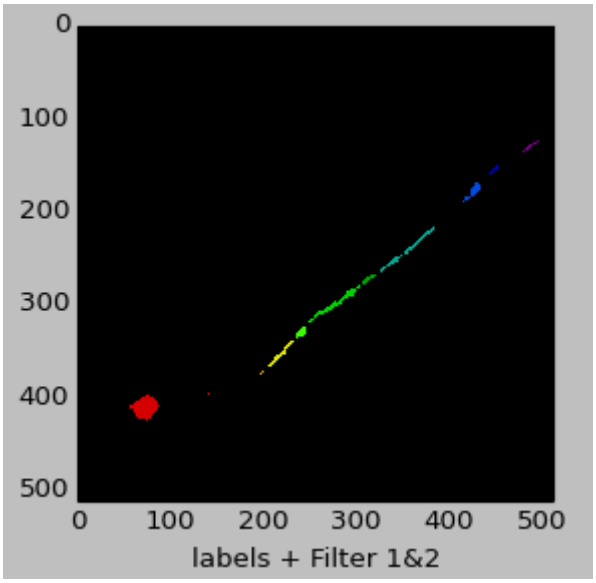
Exemple 2



c) Labellisation et deuxième filtrage

Une fois le neurone séparé du fond, on va vouloir détecter les positions des centres de masses des différents blobs qui le constituent pour pouvoir mesurer la fluorescence au bon endroit. Pour cela, on utilise une labellisation. C'est-à-dire on sépare les blobs en différents objets dont on peut mesurer les propriétés, notamment la taille et le centre de masse qui nous seront utiles. Pour cela on utilise d'abord la fonction *label* de la librairie skimage – morphology pour séparer les blobs. Et ensuite sur la liste des blobs on utilise la fonction *regionprops* de la librairie skimage – measure pour récupérer les tailles et centre de masse de chaque label.

Exemple de labellisation



Exemple de mesure des propriétés des labels

```
props = [{'Area': 7.0, 'Centroid': (32.285714285714285, 347.85714285714283),
'Label': 1}, {'Area': 534.0, 'Centroid': (132.84831460674158, 381.19662921348316),
'Label': 2}, {'Area': 1.0, 'Centroid': (149.0, 364.0), 'Label': 3}, … ]
```

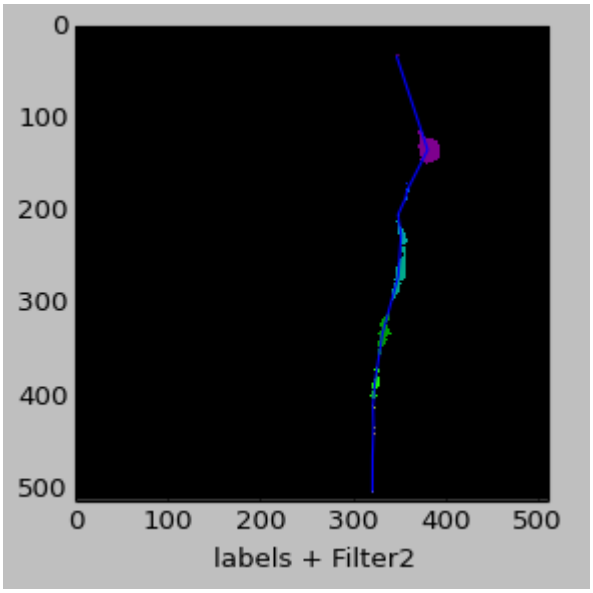
Une fois cela fait, l'utilisateur a le choix d'utiliser un deuxième filtre. Ce deuxième filtre a le même effet que le premier, mais plus loin dans le traitement de l'image. Le principe est toujours de retirer les petits objets, mais cette fois-ci on regarde la taille des objets dans les propriétés des labels mesurés (`'Area':` dans l'exemple ci-dessus). De cette façon, on pourra parfois enlever des petites régions ayant échappées au premier filtre. Ces régions peuvent parfois être gênantes car elles créent des zig-zag locaux dans notre « neurone-virtuel » décrit ci-après que l'on utilise pour mesurer la fluorescence. La taille de ces petites régions est fixée et n'est pas laissée au choix de l'utilisateur. Mais on pourrait implémenter cette possibilité par la suite.

d) Création du neurone-virtuel

Le prétraitement de l'image est terminé à ce stade. Il faut maintenant choisir où et comment mesurer la fluorescence.

Pour cela, on crée un neurone-virtuel qui est une liste de coordonnées comprenant les centres de masse des régions labellisées. (Créé sur base des `'Centroid':` de l'exemple précédent)

L'objectif est de pouvoir mesurer la fluorescence tout le long du neurone et pas uniquement sur les blobs qui ont résisté au prétraitement des images. On va donc relier ces centres de masse et ajouter dans le neurone-virtuel des points à intervalles réguliers entre ces centres. (L'intervalle est laissé au choix de l'utilisateur.)



Exemple graphique (en réalité le neurone-virtuel est une succession de points et non pas de segments de droites)

Si tout fonctionne bien, le neurone réel reste dans une bande d'une certaine largeur autour du neurone virtuel pour toutes les images. Si ce n'est pas le cas, une partie de la fluorescence ne sera pas mesurée. (Dans le cas où le seuillage et les filtres utilisés ont enlevé une partie faisant partie du neurone réel mais décentrée par rapport aux blobs adjacents)

Dans toutes les images testées, les changements de direction dans le neurone réel restent assez faibles (en tout cas de manière locale) et donc ce cas ne s'est pas présenté. Mais si ca devait arriver, cela peut être une cause de dysfonctionnement du programme.

3. Mesure de la fluorescence

Une fois le neurone-virtuel complet, on peut enfin passer aux mesures de fluorescence qui vont nous permettre de différencier et détecter les mutants.

La première idée sur laquelle je me suis penché consistait à mesurer la fluorescence dans des petits rectangles le long du neurone-virtuel. Ces petits rectangles auraient été centrés

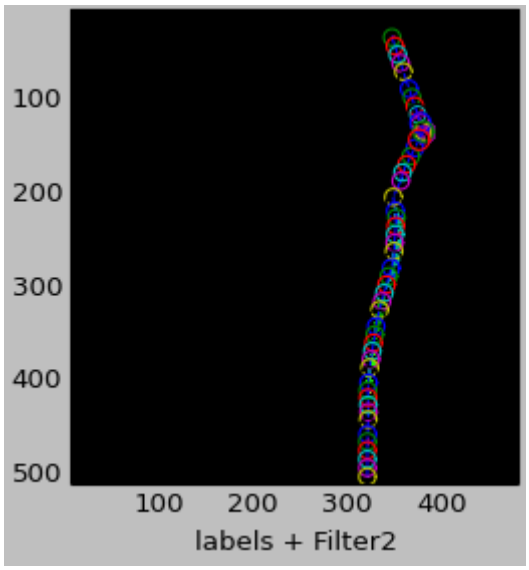
sur les points formant le neurone-virtuel. Mais cela amenait deux complications : la première était le fait qu'il est nécessaire de mesurer les angles entre chaque segment pour orienter les rectangles dans le bon sens en fonction de leur position sur le neurone-virtuel. La deuxième était de définir quels pixels faisaient partie de ces rectangles en fonction de ces angles et des dimensions des rectangles (nécessitant une fonction de rotation).

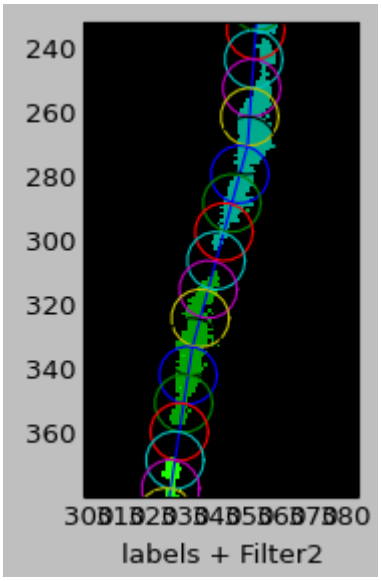
Pour éviter cela, mon choix s'est porté sur un modèle plus simple évitant ces problèmes : des cercles. De cette façon l'orientation n'est plus impliquée.

Sur chaque point du neurone-virtuel, on place un cercle. Le choix du diamètre des cercles est laissé à l'utilisateur. Du coup, l'utilisateur peut choisir dans quelle mesure les cercles se chevauchent. (Il peut choisir l'écart entre chaque point du neurone-virtuel et le diamètre des cercles de mesure.) Pour mesurer la fluorescence dans un cercle, on somme simplement les valeurs de niveaux de gris de chaque pixel faisant partie du cercle. Les étapes précédentes permettaient de définir les coordonnées du neurone-virtuel en utilisant des images traitées, mais on mesure bien la fluorescence sur les images originales.

Exemple pour tout un neurone et zoom sur la partie centrale de celui-ci

Distance entre les points : 10 (distance géométrique en pixels)
Diamètre des cercles : 18 (distance géométrique en pixels)

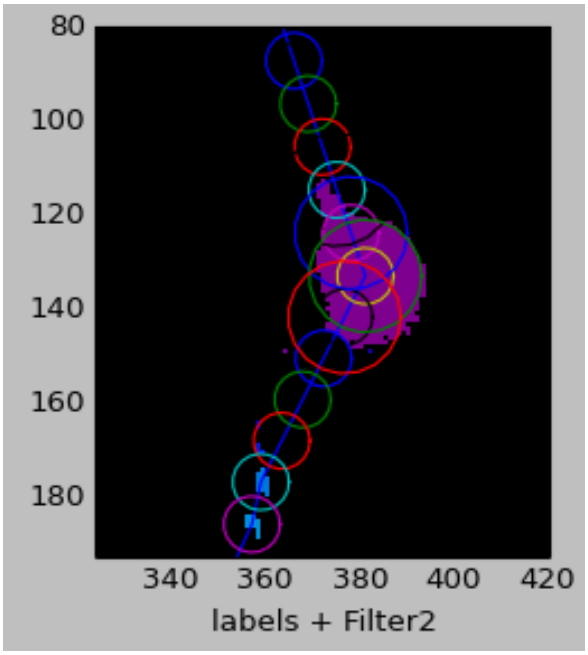




Une fois une première mesure effectuée, on trouve où se trouve le maximum de fluorescence. Cela permet de récupérer la position du soma du neurone dans la liste du neurone-virtuel car le soma est toujours le point à fluorescence maximale pour nos images. (Le soma est le nom donné au corps d'un neurone, le soma contient le noyau de la cellule nerveuse.) On récupère ensuite également les coordonnées des points adjacents au soma pour constituer la zone-soma formée des 3 points consécutifs (Le point central du Soma plus

un point de chaque côté). Dans cette zone, on remesure la fluorescence dans des cercles de diamètre plus grand que pour le reste du neurone.

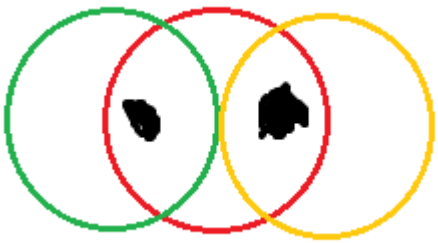
On peut voir dans l'exemple ci-dessous les 3 cercles de diamètre plus grand autour des 3 cercles de base ayant été détectés comme étant la zone-soma.



Les raisons de ce choix de différentes tailles de cercles de mesures sont multiples :

1. Garder les cercles les plus petits possibles tout en restant à une taille suffisante pour englober les blobs en entier permet de diminuer l'impact des pixels du background qui ont un niveau de gris entre 3900 et 4500 et qui seront sommés aussi lors de la mesure. En effet, la valeur de fluorescence mesurée dans un cercle où il n'y a aucune fluorescence n'est pas de 0 mais bien de +-4200 multiplié par le nombre de pixels dans le cercle. Donc adapter la taille des cercles à la taille des blobs, même de manière grossière en considérant uniquement 2 types de zones (soma et axone) améliore considérablement la manière dont on mesure la fluorescence.

2. Garder les cercles les plus petits possibles permet de diminuer le nombre de fois ou un pic de fluorescence sera détecté entre 2 blobs à cause d'un cercle qui englobe 2 bords de blobs ou 2 blobs entier comme expliqué sur le schéma ci-dessous.



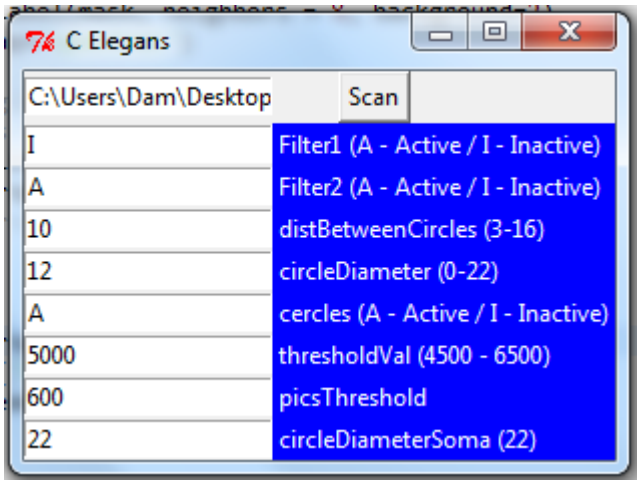
Dans le cas ci-contre, le maximum de fluo sera mesuré comme se trouvant au centre du cercle rouge et non pas comme 2 blobs bien distincts.

L'utilisation de cercles plus petits permet d'éviter cela.

Ici il n'y a pas de cercle qui contient les 2 blobs et ils seront bien considérés par le programme comme 2 blobs séparés.

4. Utilisation du programme et sauvegarde des résultats

Pour faciliter l'utilisation, j'ai implémenté une interface graphique à partir de laquelle on peut changer les paramètres et lancer l'analyse. Pour cela j'ai utilisé Tkinter, un outil d'interfaces en python. Une fois le programme lancé, cela ouvre la fenêtre suivante :

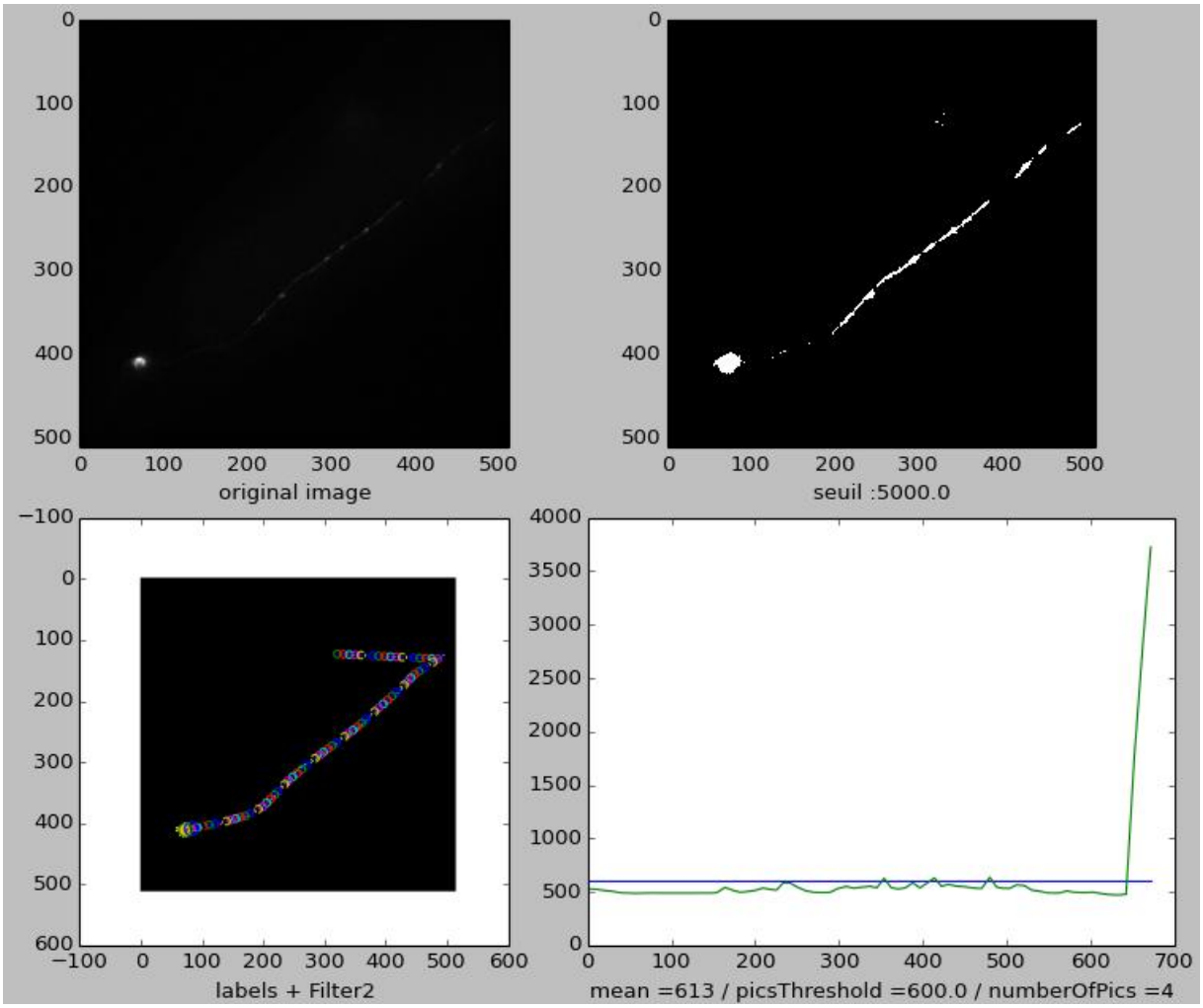


Le premier champ est destiné au path du dossier à scanner. Les champs suivants permettent à l'utilisateur de choisir les valeurs de paramètres qu'il souhaite et d'appliquer ou non les filtres.

Une fois que l'on appuie sur le bouton « Scan », l'analyse des images commence.

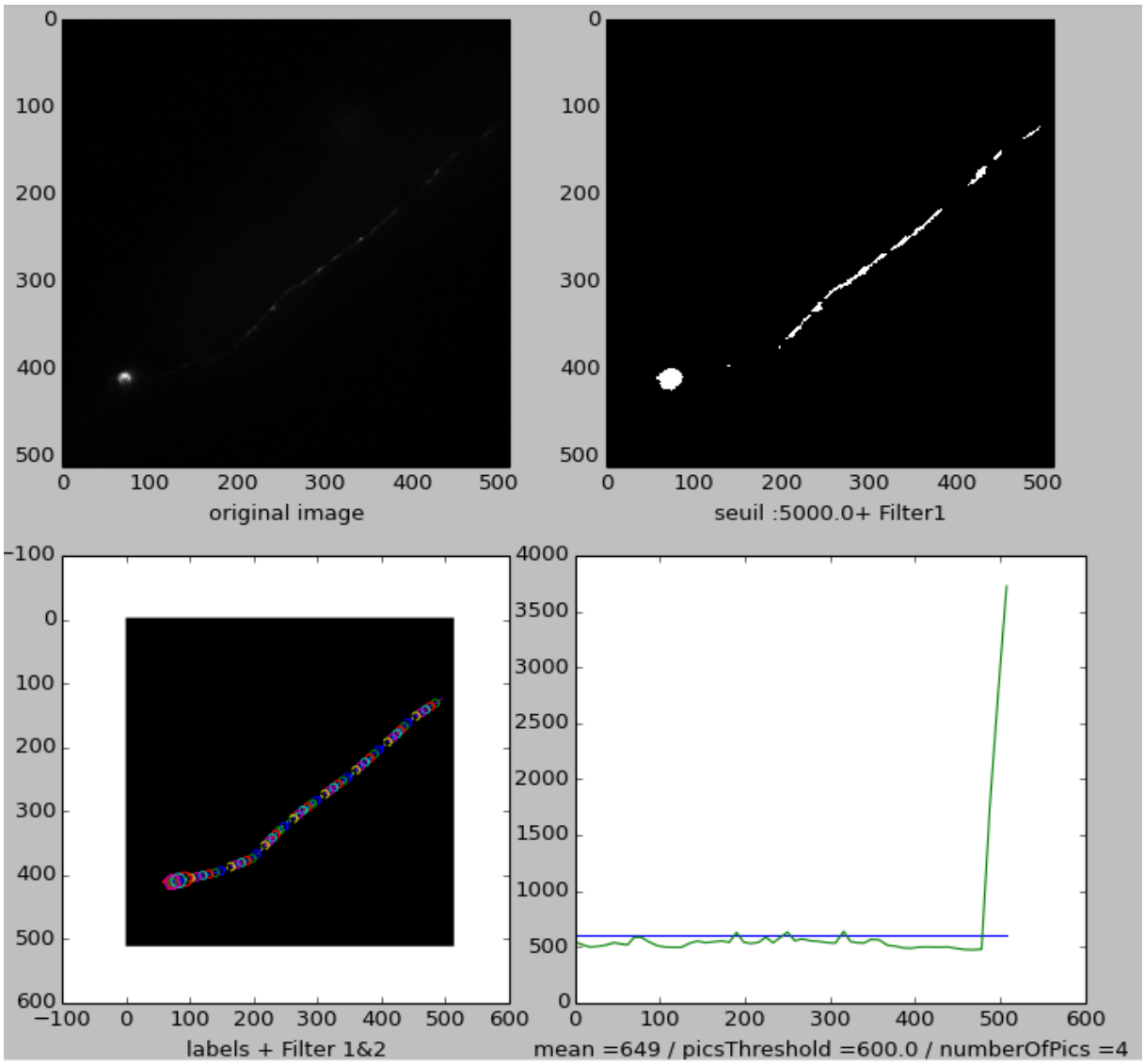
Pour chaque image du dossier, tout le traitement décrit précédemment va être appliqué et deux fichiers de résultats seront sauvegardés dans un autre dossier.

1. Une image séparée en 4 permettant de voir rapidement différentes choses :
   - si les mesures de fluorescence ont bien été prises au bon endroit
   - quels filtres ont été utilisés pour produire ces résultats
   - a quel niveau du traitement de l'image se situe le problème s'il y en a un
   - un graphique de la fluorescence le long du neurone
   - le nombre de pics au dessus d'un seuil défini par l'utilisateur
   - la moyenne du niveau de fluorescence le long du neurone



Dans l'exemple ci-dessus le filtre1 n'est pas utilisé, donc le bruit que l'on voit sur l'image seuillée en haut à droite a été considéré comme faisant partie du neurone et pris en compte dans les mesures.

Ci-dessous la même image mais avec le filtre1 utilisé. Le résultat est celui recherché.



2. Une ligne dans un fichier xls reprenant toutes des mesures utiles pour trouver des mutants particuliers ou pour faire des analyses statistiques sur la population des C Elegans. Ces mesures comprennent :

   - La moyenne dans tout le neurone
   - La moyenne sans le soma
   - Le nombre de pics au dessus d'un certain seuil
   - La longueur du neurone
   - Le nombre de pics absolus

Les valeurs des paramètres et les filtres utilisés sont également répertoriés dans le fichier xls.

	A	B	C	D	E	F	G	H
1		Filter1	Filter2	distBetweenCircles	circleDiameter	circleDiameterSoma	picsThreshold	thresholdVal
2		False	True	10	12	22	600	5000
3								
4	ID	mean	meanWithoutSoma	neuron lenght	max	numberOfPicsOverTreshold	numberOfPics	
5								
6	Acquired-15.tif	549	447	312	627	1	5	
7	Acquired-19.tif	509	474	1432	920	1	7	
8	Acquired-2.tif	567	481	559	952	2	7	
9	Acquired-21.tif	592	482	409	1019	2	4	
10	Acquired-24.tif	562	463	399	770	1	5	
11	Acquired-4.tif	694	495	485	1297	5	6	
12	Acquired-5.tif	642	488	390	864	2	7	
13	Acquired-6.tif	724	448	195	965	3	2	
14	Acquired-8.tif	609	464	542	926	2	8	
15	Acquired.tif	596	492	478	1099	2	6	
16								
17								
18								
19								

5. Organisation du code

Le code a proprement parlé est structuré en :

- 1 classe pour l'interface graphique
- 4 fonctions principales :
  - Prétraitement de l'image avant les mesures de fluorescence
  - Calculs des différentes propriétés utiles résultant de la mesure de fluorescence (moyenne, nombre de pics, …)
  - Une fonction qui sauvegarde les données du scan dans le fichier xls
  - Une fonction CElegans qui fait office de main et qui appelle les autres fonctions dans l'ordre voulu. (La fonction CElegans se lance lorsque l'on clique sur « Scan » dans l'interface)

- Différentes petites fonctions mathématiques (Somme de la valeur des pixels dans un cercle, calcul de distances, …)

Le code n'est pas séparé en plusieurs fichiers. Cela est simplement du au fait que j'ai essayé de faire quelque chose en plusieurs fichiers qui colle plus aux règles de bonne pratique de programmation, mais je ne parvenais pas à exécuter un autre fichier après le clic sur le bouton et à appeler les fonction ou classe des autres fichiers. J'ai donc abandonné et copié mes fonctions dans le fichier de l'interface, cela fonctionne très bien mais ce n'est pas de la « belle » programmation.

6. Conclusion

Le programme fonctionne et réalise assez bien ce qui était demandé mais on n'échappe pas à une vérification humaine image par image pour enlever les cas où du bruit n'a pas été éliminé. Comme bien souvent en imagerie, le problème principal est de séparer le bruit des objets d'intérêt. Il tourne assez vite également (moins d'une seconde par image). Il y a certainement moyen d'implémenter une étape de post-traitement qui tiendrait compte des angles, de la longueur totale du neurone-virtuel ou encore d'une comparaison entre le neurone-virtuel et une courbe lissée pour enlever les contributions du bruit. J'ai imaginé plusieurs méthodes pour optimiser cela mais je n'ai pas voulu passer trop de temps supplémentaire pour le faire.