Letendre Paul *Mau Adrien* *Dreyfus Virgile*

Compte-rendu de projet

Breaking the diffraction limit by deconvolution of images acquired by photothermal microscopy using gold nanoparticles

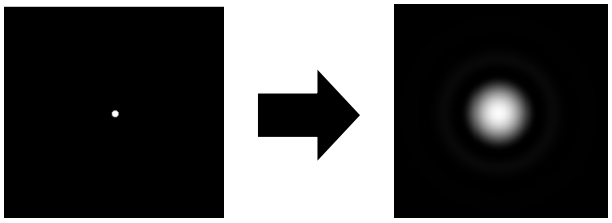
Sommaire

1- Contexte et objectifs attendus

De nos jours, les microscopes optiques ont atteint une qualité suffisante pour que l'on puisse considérer qu'ils sont dénués d'aberrations. Ainsi, leur résolution est limitée seulement par la diffraction, qui est une limite physique infranchissable pour tout système imageant.

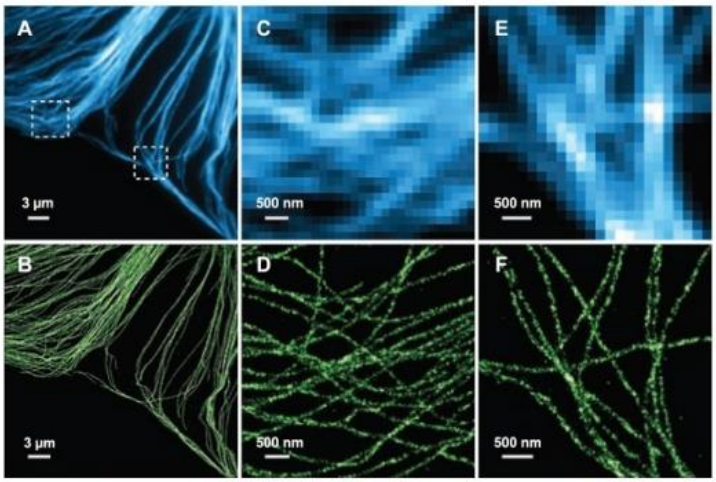
Mais dans le but de voir toujours plus petit, différentes techniques (physiques ou numériques) ont été mises au point pour dépasser cette limite. On les appelle techniques de super-résolution.

Parmi ces méthodes, il existe un sous-ensemble qu'on appelle « super-résolution par localisation de molécules individuelles », qui consiste à imager des particules de taille nanométrique placées dans l'échantillon que l'on étudie. Lorsqu'elles sont isolées, ces particules de taille bien inférieure à la réponse impulsionnelle (PSF) du microscope forment une image correspond à cette PSF.



Etalement d'un point source par une PSF

On peut alors approximer ces images par un modèle connu de la PSF ce qui permet d'estimer le centre de chacune de ces tâches images et donc la position des différentes particules. Le fait de connaitre les positions des particules permet alors de reconstruire une image.



Images limitées par diffraction (en bleu) et images super-résolues (en vert)

La nature de la particule choisie, liée à la modalité d'imagerie du microscope, comme marqueur influe fortement sur l'efficacité de ces méthodes.

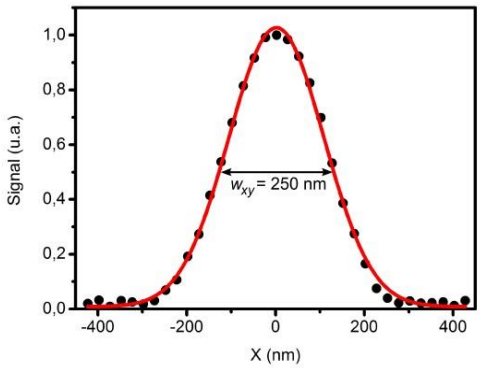
La microscopie de fluorescence est une première modalité très utilisée en combinaison avec des méthodes de super-résolution. Elle se base sur la détection optique de particules fluorescentes de taille nanométrique, ce qui, en vertu de l'explication précédente, résulte en des tâches de diffraction. Différentes méthodes développées au cours des dernières années permettent d'imager ces molécules de manière individuelle et ainsi de faire de la super-résolution.

Néanmoins, les particules fluorescentes ont leurs défauts. En effet, elles souffrent par exemple d'une durée d'émission limitée (photoblanchiement) ou d'un effet de clignotement (blinking) pouvant biaiser les mesures.

La microscopie photothermique est une seconde modalité d'imagerie qui se base sur l'effet photothermique, par exemple dans des nanobilles d'or. Cet effet se manifeste par l'absorption d'une onde incidente par la bille, ce qui conduit à son échauffement ainsi que l'échauffement du voisinage de la bille. L'échauffement induit une variation d'indice du milieu contenant la bille, c'est cette variation d'indice que l'on détecte lorsqu'on fait de la microscopie photothermique.

Une nanobille d'or d'un diamètre donné entraine la détection d'un signal de forme connu – en l'occurrence une gaussienne. Cette gaussienne est de taille largement supérieure à celle de la nanobille (10nm pour la particule, environ 250nm pour la gaussienne), mais comme expliqué précédemment, on peut remonter à une localisation précise de la particule à l'aide de modèle gaussien connu.

Les nanoparticules d'or ont l'avantage de ne pas présenter les caractéristiques de photoblanchiement et de clignotement des fluorophores, le signal détecté est donc très stable (temporellement et spatialement), ce qui favorise l'utilisation de méthodes de super-résolution en imagerie photothermique.



Réponse d'une bille d'or

Notre projet s'inscrit dans cette volonté d'utiliser la microscopie photothermique afin d'obtenir des images super-résolues. En effet, notre objectif est de réaliser un programme de super-résolution destiné à améliorer les images acquises par deux chercheurs du LP2N, L. Cognet et B. Lounis. Il s'agit également de faciliter le traitement de ces images afin de présenter correctement leurs informations. Ainsi, nous avons pour but final de développer une interface graphique ergonomique permettant de réaliser des opérations de super-résolution et de faciliter le post-traitement des images.

2- Etat de l'art

Afin de choisir la meilleure façon de mener à bien notre projet, nous commençons par établir un état de l'art. Celui-ci s'oriente selon deux grands axes. D'une part il nous faut étudier les diverses démarches algorithmiques possibles pour les différentes étapes de traitement des images de microscopie photothermique (super-résolution et post-traitement). D'autre part il nous faut choisir le langage et le logiciel nous permettant d'implémenter ces algorithmes et de développer une interface graphique adaptée aux utilisateurs finaux.

2- 1. Algorithmes pour le traitement des images

Comme expliqué précédemment, la super-résolution peut être obtenue par localisation de molécules individuelles et l'algorithme de fit multi-gaussien nous a été fourni par les encadrants du projet. Mais la super-résolution peut être également obtenue par des méthodes de déconvolution.

Après avoir rappelé le principe de la déconvolution, nous présentons les différentes catégories d'algorithmes tout en comparant leurs avantages et inconvénients à nos exigences et contraintes.

La déconvolution est une méthode utilisée afin de restaurer un signal, dégradé par l'instrument de mesure lors de l'acquisition (le signal mesuré résulte d'une convolution entre le signal à mesurer et la réponse impulsionnelle du système).

Dans le contexte de l'imagerie photothermique, il faut prendre en compte la globalité du dispositif expérimental pour préciser à quoi correspond la réponse impulsionnelle du système. En effet, ce n'est pas la PSF du système optique qui est à l'origine du flou de l'image qui est acquise pixel par pixel. C'est l'effet photothermique et sa détection qui transforment un « point » d'une dizaine de nanomètres en une tâche de plusieurs centaines de nanomètres.

Pour comprendre les démarches de résolution du problème, il faut retenir que la déconvolution appartient à la classe des problèmes mal posés au sens de Hadamard. En effet :

-la solution peut ne pas exister

- elle n'est pas nécessairement unique

- elle peut être instable (une faible erreur initiale sur les données peut entrainer une forte erreur sur le signal reconstruit)

Ceci est intimement lié au fait que l'on a récupéré une information incomplète lors de la mesure par un système réel i.e non parfait. Ainsi, il est important de régulariser le problème, c'est-à-dire d'apporter de l'information a priori via notre connaissance du système physique de mesure. Nous verrons plus en détail quelques méthodes de régularisation du problème ci-dessous.

Pour présenter les différentes méthodes algorithmiques, distinguons dans un premier temps les algorithmes utilisant une connaissance a priori du système d'acquisition et ceux fonctionnant « à l'aveugle ». Ces derniers se proposent d'estimer à la fois l'image à reconstruire et le modèle d'acquisition. Certes très intéressants dans le domaine de l'imagerie satellitaire, nous les écartons d'office car nous pouvons estimer la réponse de notre système de mesure et ainsi utiliser des algorithmes plus simples.

Nous pouvons ensuite faire la distinction entre les traitements par filtrage (itératifs ou non) et les méthodes stochastiques (Richardson-Lucy). Bon nombre de techniques de filtrage se basent sur une minimisation du critère des moindres carrés $||Hx - y||$ où H représente la réponse du système, x le signal cherché et y le signal mesuré. Mais la matrice H est souvent mal conditionnée ou inversible ce qui peut conduire à un trop grand nombre de solutions. Une façon de régler ce problème consiste à utiliser une régularisation de Tikhonov i.e l'ajout d'un terme au critère à minimiser : $||Hx - y|| + ||\Gamma\, x||$. L'ajout de ce terme permet de sélectionner des solutions pertinentes selon la connaissance du problème physique. Par exemple, il peut permettre de lisser le signal ou de le confiner autour d'une forme connue a priori. Certains algorithmes se basent de plus sur une optimisation sous contraintes – afin d'imposer des valeurs prédéterminées à certains pixels.

Nous décidons de nous orienter vers les méthodes de filtrage linéaire qui d'une part sont plus efficaces en termes de temps de calcul, et qui d'autre part sont davantage inscrites dans le cadre de nos connaissances mathématiques.

Le dispositif utilisé est capable d'acquérir des images selon d'autre modalités que le photothermique. Il serait donc intéressant de pouvoir coupler ces informations afin de réaliser une image multimodale, notamment afin de replacer une image photothermique déconvoluée dans une image, plus large, acquise en lumière blanche.

Le problème de recalage d'image (« image registration » en anglais) peut essentiellement être abordé par deux types de méthodes automatiques.

Les premières se basent sur des mesures de similarité de la répartition d'intensité dans les images à superposer. Elles ne sont pas adaptées à notre situation car nos images à superposées ne sont pas acquises selon la même modalité et sont donc difficilement corrélables.

Une seconde catégorie d'algorithmes se base sur une approche géométrique. L'objectif étant d'extraire des primitives géométriques (coins, points saillants, contours…) communes aux deux images afin de les recaler. Nos images n'étant pas composées de nombreux objets aux caractéristiques géométriques différentes, ces méthodes ne sont pas non plus adaptées à notre problème.

Nous ne pouvons donc proposer qu'une méthode manuelle de recalage d'image dans laquelle l'utilisateur doit sélectionner des points communs dans les deux images pour que le programme puisque calculer les déformations et déplacements nécessaires à la superposition des deux images.

2- 2. Choix du langage et du logiciel pour développer le GUI

En parallèle du choix des méthodes algorithmiques il nous faut également déterminer dans quel langage et avec quel logiciel nous pouvons programmer l'interface graphique souhaitée.

Compte tenu du cadre du projet et de nos compétences, trois solutions potentielles s'offrent à nous. La première consiste à coder l'interface en C++ afin d'obtenir les meilleurs résultats en termes d'efficacité de calcul. Bien que des bibliothèques de fonctions pour le traitement d'images (par exemple OpenCV) soient disponibles et que certains logiciels aident à concevoir une interface graphique en C++, nous abandonnons tôt cette solution car elle présente une complexité bien supérieure au développement d'une interface sur MatLab. En effet, le choix de réaliser le programme sur MatLab est une option intéressante car de nombreuses fonctions de traitement des images sont disponibles mais surtout grâce à l'outil GUIDE qui permet de concevoir facilement et rapidement des GUI. Les inconvénients de MatLab sont sa moindre efficacité en termes de calculs numériques et la nécessité d'utiliser des boîtes à outils supplémentaires pour certains traitements spécifiques. Une troisième option se présentant à nous est de développer le GUI grâce au logiciel ImageJ – utilisé par les encadrants du projet pour réaliser diverses opérations de post-traitement d'images. Nous avons particulièrement cherché à exploiter les possibilités de développement de fonctionnalités modulaires à ajouter à l'interface de base sous forme de macro. Mais Java étant le langage principal pour développer des GUI compatibles avec ImageJ, nous avons abandonné cette solution après avoir essayé de prendre en main rapidement ce nouveau langage que nous ne connaissions pas.

Au final, nous optons pour un développement du GUI avec l'outil GUIDE de MatLab – celui-ci permettant de réaliser l'interface en partie grâce à du « drag and drop » - tout en ayant trouvé un moyen de transférer les images traitées dans une fenêtre ImageJ afin de réaliser certains post traitement si nécessaire.

3- Solutions proposées

Nous présentons ici notre démarche pour l'implémentation de l'algorithme de déconvolution et de l'interface graphique.

3- 1. Déconvolution

Les tests

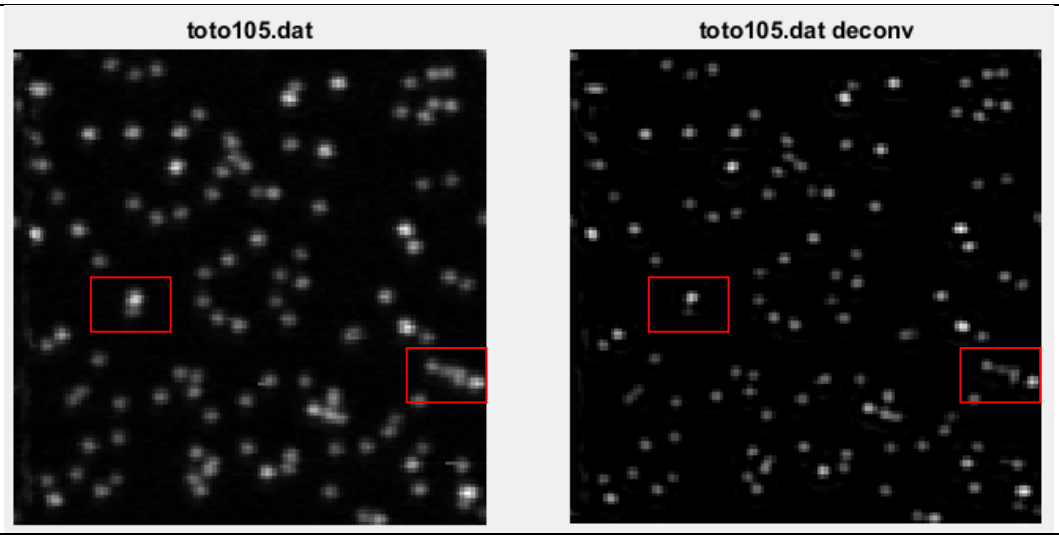
Lorsqu'on image des nanoparticules, on sait que si cette particule est isolée, c'est-à-dire suffisamment loin de ses voisines, on peut retrouver son centre de manière relativement précise. Par contre, l'analyse devient autrement plus compliquée si elles sont proches et que leurs réponses impulsionnelles se recouvrent. Pour cela, un but de la déconvolution est permettre de résoudre, de séparer deux particules proches. Parce que l'acquisition d'une image est longue et que le temps d'acquisition augmente avec la taille et la résolution de l'image, il est également nécessaire de rechercher jusqu'à quelle résolutions ces méthodes sont efficaces, en les testant sur des images à différentes résolutions.

Les résultats

La déconvolution est assurée par les fonctions filtreWiener et filtreWienerAuto, la deuxième fonction ne se démarquant de la première que par le fait qu'elle propose le choix automatique d'un paramètre de la première fonction.

Les images sur lesquelles ont été testées les fonctions de déconvolution ont des échantillonnages allant de pixels de 50 nanomètres à des pixels de 200 nanomètres.

Ci-dessous est présenté le résultat de la déconvolution pour une image avec un pas de 100 nanomètres.

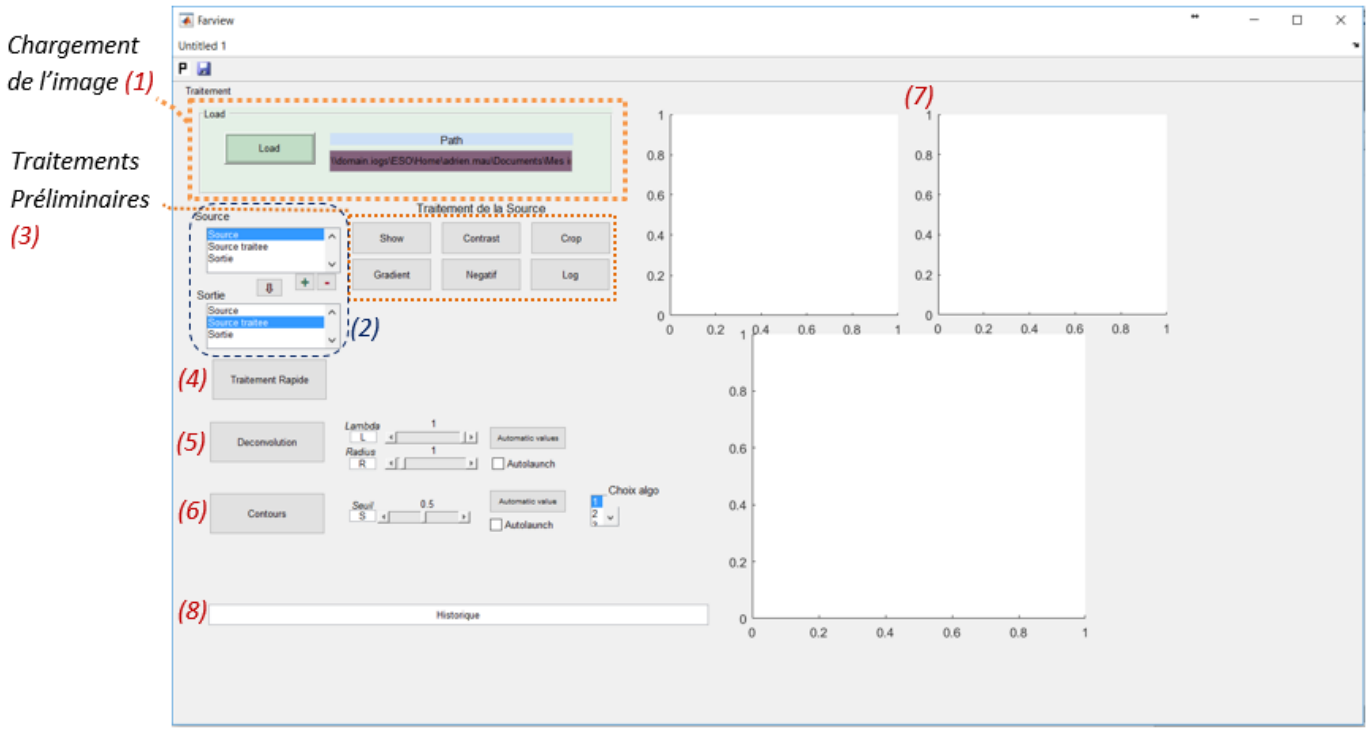


A gauche, une image obtenue par microscopie photothermique, échantillonée par pas de 100 nm
A droite, la même image déconvoluée. On peut apercevoir certaines zones (encadrées en rouge) où la déconvolution a aidé à résoudre deux gaussiennes voisines

La déconvolution commence à présenter des résultats pour des pas d'acquisition aux alentours de 100 nanomètres. Les résultats de la déconvolution peuvent également être traités pour faciliter l'analyse.

3- 2. Conception du GUI

Créer avec l'outil Guide de Matlab, il peut être compilé pour s'exécuter de façon autonome.



(1) Chargement :

Le bouton **Load** ouvre le dernier dossier ouvert, et permet de charger des images classique (jpg, png …) ou en .dat (utilise trackread)

L'image se charge dans l'emplacement sélectionné sur la liste 'Source'

(2) Listes :

Source :

Sélectionnez ici l'image de départ utilisée pour les traitements

Sortie :

Sélectionnez ici l'image de sortie d'un traitement.

Autres boutons :

 Remplace l'image sélectionnée en sortie par l'image sélectionnée en source.

 Ajoute une image à la liste possible, ou supprime l'image actuelle. (agis dans les deux listes)

(3) Traitements préliminaires :

/!\ seul le bouton show ne crée pas une image en sortie

Contrast : utilise imcontrast() pour changer le contraste entre deux valeurs.

Crop : L'utilisateur choisit (clic souris et glissement) la nouvelle image, tirée de l'ancienne.

Gradient : applique un gradient

Négatif : inverse les couleurs (1-img)

Log : Applique un logarithme log(1+img), pratique pour contraster l'image et mieux voir les détails.

(4) Traitement rapide :

A IMPLEMENTER

Sensé automatiquement faire à la fois la déconvolution préliminaire et la localisation et caractérisation des gaussiennes.

(5) Déconvolution et paramètres :

La déconvolution s'effectue à l'aide d'un filtre de Wiener paramétré. Lambda caractérise le bruit et est généralement optimal vers 0.01. R est la variance moyenne des gaussiennes que l'on recherche.

Ces valeurs peuvent être rentrées à la main, au glissement du slider, ou recherchées automatiquement.

Si le bouton **Autolaunch** est coché, un changement de valeur lance automatiquement l'algorithme, sinon il faut appuyer sur le bouton **Déconvolution**.

(6) Contours et paramètres :

Détecte le nombre et les caractéristiques (position, amplitude, variance) des gaussiennes de l'image.

Cette valeur peut être rentrée à la main, au glissement du slider, ou recherchée automatiquement.

Si le bouton **Autolaunch** est coché, un changement de valeur lance automatiquement l'algorithme, sinon il faut appuyer sur le bouton **Contours**

Choix de l'algorithme, avec n le nombre de gaussiennes à trouver :

1/ Fit directement sur n gaussiennes (long)

2/ Fit des gaussiennes séparément, sur n zones, mais position approximative.

3/ Fit des gaussiennes séparément, sur n zones avec marqogauss. Plus précis.

4/ Findpeak (à implémenter)

On renvoit ensuite les histogrammes en intensité et en rayon des gaussiennes.

Le paramètre des gaussiennes est enregistré dans le dossier, avec la date correspondante.

(7) Affichage :

L'image de gauche affiche l'image en Source, et l'image de droite l'image en Sortie.

L'image du bas sert à la superposition d'images.

(8) Historique

Encore à améliorer

Enregistre les opérations réalisées sur l'image.

(Bonus) Touches clavier :

Afin de pouvoir rapidement observer les nuances des images sans modifier leurs valeurs, on peut appuyer sur les touches **C** et **L**. Celles-ci respectivement contrastent ou affichent le logarithme des images *log(1+img) /log(2)* en entrée et en source.

Si on veut afficher de nouveau les images normales, il suffit d'appuyer sur **N**.

Note :

Les boutons de sauvegarde n'étant pas encore implémentées, on peut sauvegarder des images en l'affichant (**Show**) puis en utilisant l'utilitaire de Matlab.

Fonctions utilisées :

Farview.fig
Farview.m
RngaussRI.m
approxR.m
calcR.m
contours.m
contoursp.m
filtreWiener.m
filtreWienerAuto2.m
findpeak.m (bientôt)
fit_ngaussRI.m
fitngauss.m
fitopt.m
gaussian.m
marqogauss.m
trackread.m

Matlab montre les fonctions utilisées par le programme avec:

[fList,pList] = matlab.codetools.requiredFilesAndProducts('Farview.m')

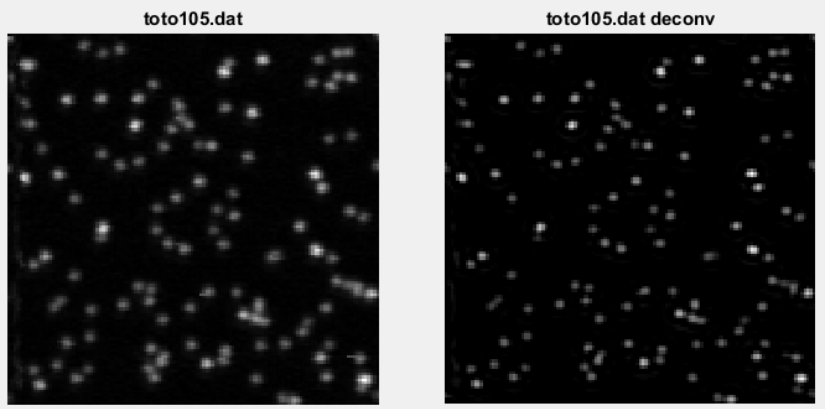
Ce qu'il reste à faire :

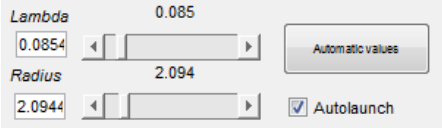
-Enregistrement plus simple des images et des paramètres obtenus.
-On peut imaginer rajouter quelques manipulations d'images : étirement, rotation…
-Superposition des images et déplacement.
-Améliorer les logs

Ce qui est fait :

¤ Le chargement d'image de différents formats, les traitements initiaux et l'affichage des images est effectué. On peut facilement choisir quelle image passe en entrée et en sortie de chaque traitement et remplacer, ajouter ou supprimer des images.

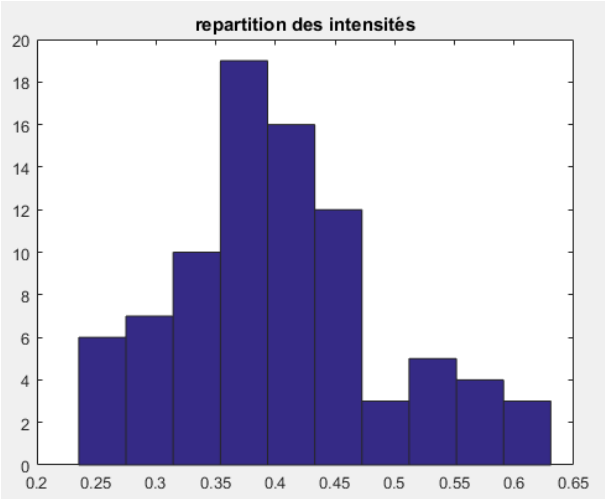
¤ La Déconvolution est au point mais peut encore être améliorée avec par exemple du suréchantillonnage.
On peut voir le résultat de façon quasi instantané et manipuler facilement les paramètres impliqués grâce au slider.

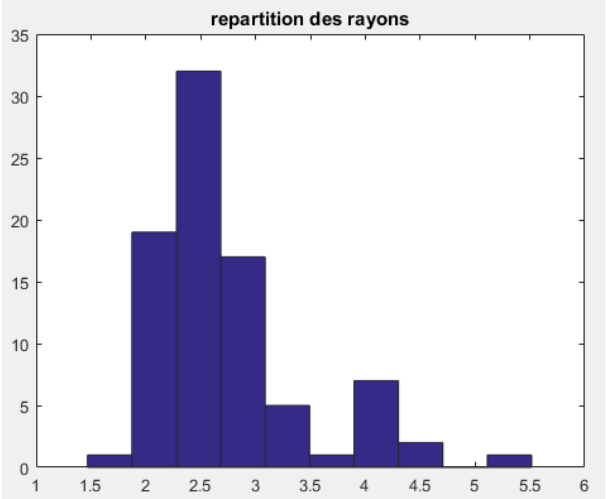




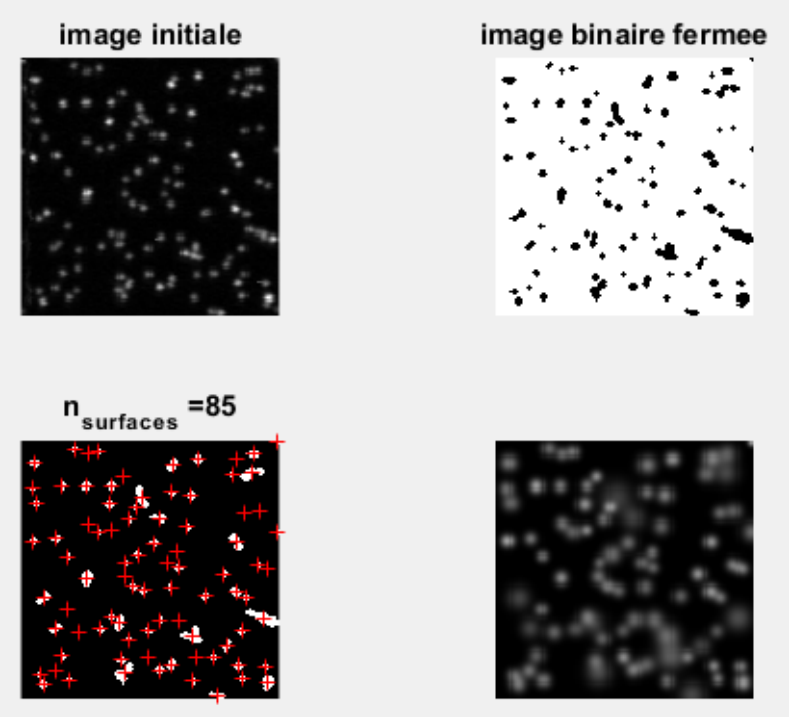
¤ L'algorithme de fit (bouton **Contours**) est aussi facilement accessible, et renvoit les histogrammes en rayon et en intensité des gaussiennes.
Le paramètre p est enregistré dans le dossier principal.

Exemple d'histogrammes obtenus :





L'algorithme de contour affiche aussi son cheminement, afin que l'utilisateur puisse –ou non- valider le résultat qu'il obtient :



On remarque que l'algorithme n'est pas très robuste quand deux gaussiennes sont très proches.

4- Organisation du projet

Déroulement des semaines

La répartition des tâches effectuées selon les semaines de projet est présentée ci-dessous.

01/11/2016	07/11/2016	14/11/2016	21/11/2016
Découverte du sujet			
Recherches sur Java et ImageJ	Recherche sur Java et ImageJ	Recherche sur Java et ImageJ	
	Recherche de librairies C++ ou	Tests création scripts Java	
	Matlab existantes	et plugins ImageJ	
	Recherche gui Matlab	Recherche gui Matlab	Choix Matlab
			Mise en place GitHub
			Tests fonctions déconvolution de Matlab
			Acquisition de fonctions et images tests
			Implémentation de fonction de détection de contours
			Début du programme

21/11/2016	28/11/2016	05/12/2016	12/12/2016
Choix Matlab			
Mise en place GitHub			
Tests fonctions déconvolution de Matlab	Recherche, implementation et tests de fonctions de déconvolution	Recherche, implementation et tests de fonctions de déconvolution	
Acquisition de fonctions et images tests	Familiarisation avec les fonctions		
Implémentation de fonction de détection de contours	Implémentation du fit gaussien	fit gaussien	
Début du programme	Réalisation programme	Réalisation programme	Finalisation logiciel
		Incorporation déconvolution et fit gaussien dans programme	Ajout de fonctionnalités
			Préparation présentation

Légende :		Adrien
		Virgile
		Paul

Nous avons segmenté le travail entre les trois membres du groupe. Adrien s'occupait principalement du développement de l'interface graphique, Paul de l'implémentation de la déconvolution et Virgile des outils d'affichage et de post traitement.

Annexe : Bibliothèque de fonctions

Nous présentons ci-dessous l'ensemble des fonctions utilisées par notre programme.

Manipulation d'images :

- **trackread** (file) : charge l'image au format .dat
  - file : nom du fichier binaire à ouvrir ('toto.dat')
- **imdata**(numéro) : charge l'image toto*numéro*.dat
- **imshow2** (image) : identique à imshow mais affiche l'image sur une plage [min,max], en l'étirant
- **imshowf** (image,n) **:** affiche séparément l'amplitude et la phase d'une image dans Fourier

Interface utilisateur :

- **Farview** () : Programme Graphique permet à l'utilisateur de manipuler et d'étudier les images sans passer par les commandes Matlab.

Analyse d'image :

- **is_clear**(img,method,param): caractérise la netteté d'une image par différents algorithmes (sélectionnés avec method).
- **marqogauss**(p0,img,dimg,Mopt) : effectue un fit gaussien avec une erreur quadratique moyenne et l'algorithme de Levenberg/Marquard.
- **fitopt** (OptIn,ConfIn,TrackIn) : définit les options pour marqogauss.
- **fitngauss**(img,seuil,algo,doplot) : Fit des gaussiennes présentes sur une image,

  1 -utilise un seuil pour détecter des zones et le nombre de gaussiennes.

  Calcule leurs positions et leurs rayons approx.

  2- Applique un algorithme:

algo=1 Fit directement sur n gaussienne

algo=2 Fit de 1 gaussienne sur n zones. Les positions seront assez approximatives car juste trouvees avec les barycentres.

algo=3 Fit de 1 gaussienne sur n zones avec marqogauss /on sort alors aussi p

- **fit_ngaussRI**(img,barycentres) : Fonction adaptée au cas d'une image contenant n gaussiennes à fiter, dont on connait le nombre n et les positions et dont on veut trouver le rayon.
- **RngaussRI**(data,p,x,y,barycentres) ! calcule la différence entre les données Data et un fit avec le paramètre p : *permet d'utiliser lsqnonlin de Matlab*
- **findpeak**(image,fOpt) : Trouve les pics dans l'image sélectionnée et les caractérise (position, largeur, intensité…)
- **contours**(img,s) : renvoie les barycentres et les rayons de 'cercles' présents sur une image. s est un seuil qui peut être spécifié, ou trouvé automatiquement par le programme.
- **approxR**(img,fast) : renvoie le rayon moyen de figures présentes sur une image

Traitement d'image :

- **passe_hg(img,fradius,power)** : Passe bas ou passe haut fréquentiel :
  Multiplication terme à terme par un masque hypergaussien dans l'espace de Fourier
  img: image en nuance de gris/ 2D
  fradius : facteur pour le rayon de l'hypergaussienne (voir calcul du rayon)
  fradius>0 => passe bas (TF * masque)
  fradius<0 => passe haut (TF * 1-masque)
  power: hypergaussian power

- **passe_hg(img,fradius,power)**
- **filtreWienerAuto2 :** (Data,RI,D,nbIterations): renvoie un paramètre mu proche de l'optimal pour la déconvolution de Wiener
- **filtreWiener**(Data,RI,D,mu) : Effectue le filtre de Wiener sur l'image Data …

Arborescence actuelle du programme