PROJET MICROSCOPIE A FEUILLE DE LUMIERE - CONVOLUTION/DECONVOLUTION D'IMAGE MMNQ

Mathieu BOUYRIE
Antoine EDWELL

4 GMM - MMN

2 août 2011

Remerciements

Avant de commencer, nous tenons à remercier en premier lieu l'équipe pédagogique, Pierre WEISS et Aude RONDEPIERRE pour toute l'aide qu'ils nous ont apportée, dans un premier temps pour avancer sur le projet puis pour nous aider à étoffer le présent rapport.

Table des matières

Introduction

Le Cancéropôle de Toulouse

Un site Important

Le Cancéropôle de Toulouse est ouvert depuis 2009. Bien qu'encore inachevé, il représente aujourd'hui l'un des plus grands projets européens dans le traitement du cancer. Le site final devrait s'étendre sur 220 hectares. D'ici 10 ans, ce pôle devrait devenir l'un des leaders européens, voire internationaux, de la recherche contre le cancer. Le campus, in fine, se composera de cinq pôles :

- le centre de recherche publique : il accueillera les principales équipes de recherche de Toulouse telles que l'INSERM (Institut national de la santé et de la recherche médicale), le CNRS ou le Centre hospitalier de Rangueil.
- le centre de recherche privée : là résideront des entreprises telles que Sanofi, ou Pierre Fabre.
- Le Pôle Innovation et Valorisation : il inclut l'ITAV (voir plus loin) et une pépinière d'entreprises, en particulier des Start-up qui vont valoriser le pôle recherche du domaine.
- Le Pôle Formation : il fonctionnera en partenariat avec un certain nombre d'écoles de Haute-Garonne telles que l'Université Paul-Sabatier, l'école des Mines d'Albi-Carmaux, ou encore l'INSA de Toulouse.
- La clinique universitaire du Cancer : son ouverture est prévue pour 2013 ; elle fonctionnera en partenariat avec le CHU de Rangueil.

Gouvernance du Campus : l'Oncopôle de Toulouse

L'association Oncopôle est en charge de l'animation du Cancéropôle. Elle coordonne les différentes équipes, centres, et laboratoires du campus. Elle est aussi en charge de la promotion du cancéropôle

L'ITAV

L'ITAV (Institut des Technologies Avancées en sciences du Vivant) est un centre dédié à la recherche. Cet institut est organisé en Unités Mixtes de Service (UMS) qui regroupent plusieurs personnes employées par le CNRS, l'Université Paul-Sabatier ou enfin l'INSA de TOULOUSE. Ces unités travaillent dans trois domaines particuliers : la bionanotechnologie, la biochimie/ chimie de synthèse, et enfin l'Imagerie photonique.

Le Cancérôpole de Toulouse accueille aujourd'hui l'un des 10 prototypes mondiaux du SPIM (Selective Plane Illumination Microscop ou Microscope à Feuille de Lumière), un tout nouveau processus de microscopie, encore peu développé qui peut produire des images d'un échantillon en trois dimensions !

Le SPIM

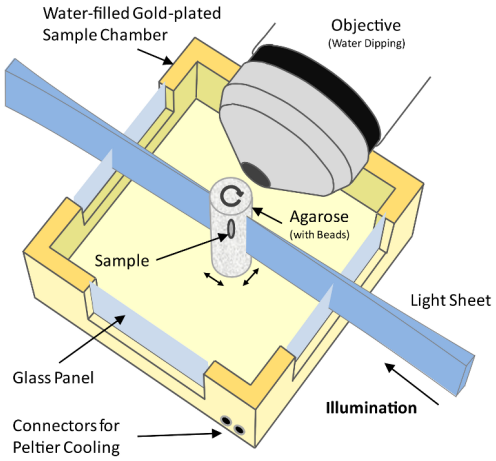
Fonctionnement

L'échantillon, traité par fluorescence, est plongé dans l'agarose, un polymère de très faible indice de réfraction, qui a la capacité de se solidifier.

Le faisceau lumineux passe d'abord à travers une lentille cylindrique, qui va avoir pour effet de l'« aplatir » selon dans une seule direction. Ensuite, il traverse l'échantillon qui va réémettre la lumière perpendiculairement au faisceau.

On peut donc voir d'où vient le terme « feuille de lumière » ; la lentille cylindrique fait converger les faisceaux selon une seule direction. En traversant l'échantillon, le faisceau s'apparente à une fine couche lumineuse.

Sur L'image ci dessous un schéma représentant une feuille de lumière :



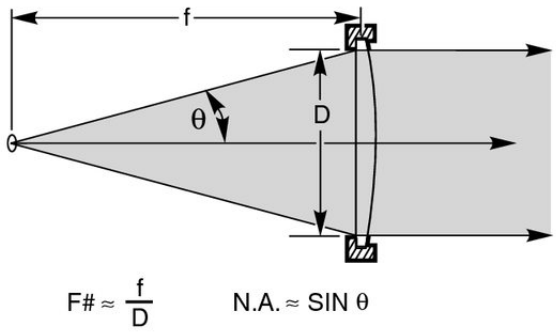
Application

Cette nouvelle méthode de microscopie peut nous permettre de mieux étudier la prolifération des cellules cancéreuses, phénomène encore très peu connu de nos jours...

Avantages de la microscopie à Feuille de Lumière

En microscopie classique, il existe une étroite relation entre ouverture numérique et profondeur de champ : quand l'un de ces paramètres croît, l'autre décroit. Ainsi, si on veut observer un large échantillon, on aura beaucoup plus de mal à le voir net sur toute sa profondeur.

Sur le dessin suivant, θ représente l'ouverture numérique :



Le SPIM évite ce problème dans la mesure où la lumière réémise l'est perpendiculairement à l'échantillon.

De plus, l'acquisition d'images de l'échantillon suivant plusieurs plans successifs peut permettre la création d'une reconstitution complète : l'échantillon pourra être observé en trois dimensions !

Inconvénient de la microscopie à Feuille de Lumière

D'une manière générale, l'épaisseur de la feuille de lumière est inversement proportionnelle à l'ouverture numérique. Il en est de même pour la taille de l'échantillon à observer. On peut donc déduire que ces deux derniers paramètres sont proportionnels. Or, bien sûr, l'objectif est de mesurer n'importe quelle taille d'échantillon (notamment de dimensions assez grandes) avec le plus de précision possible (c'est-à-dire une feuille de lumière la plus fine possible) : un compromis entre les 2 réglages va donc devoir être recherché !

Enfin, contrairement à la microscopie classique, la lumière traverse l'échantillon : celui-ci fait office de système optique. L'image à l'arrivée est donc floutée.

Problématique du projet

Le but de ce projet sera de corriger le dernier problème cité ci-dessus. On cherchera à déterminer la fonction de transfert de notre système optique pour ensuite déflouter notre image.

Ce rapport se composera de deux parties principales : dans la première sera simulé un floutage d'image. Dans la deuxième, sera déterminé un algorithme permettant de déflouter l'image. Enfin, pour terminer, nous analyserons les résultats obtenus.

Chapitre 1

Simulation : Floutage d'une Image

1.1 Floutage et Mathématiques

Dans la suite du rapport on optera pour les notations suivantes :
- $\mathcal{F}$ et $\mathcal{F}^{-1}$, respectivement la tranformée de Fourier et la transformée de Fourier inverse de $\mathcal{S}(\mathbb{R}^2)$ (ensemble des fonctions à décroissance rapide).
- FFT et $IFFT$ les opérateurs discrets associés à la tranformée de Fourier rapide et la transformée de Fourier rapide inverse.

1.1.1 Discrétisation du problème

Notre image étant pixelisée : nous introduisons les notations suivantes :
- n_x le nombre de pixels de notre image suivant l'axe des x.
- n_y le nombre de pixels suivant l'axe des y.
- $N = n_x * n_y$ le nombre de pixels de notre Image.

On ramène ainsi notre image à un vecteur de taille N. Les opérateurs FFT et $IFFT$ étant linéaires de $\mathbb{R}^N$ dans $\mathbb{R}^N$, on peut leur associer deux matrices F et F^{-1}.
Ainsi on peut établir une équivalence entre $FFT(IM)$ et Fu_{IM}, avec
- IM notre image en deux dimensions
- u_{IM} notre image ramenée sous forme de vecteur colonne.

1.1.2 Convolution et Transformée de Fourier en deux dimensions

La convolution de deux applications u et v en deux dimensions est définie par :

$$(u * v)(x, y) = \int_{\mathbb{R}^2} u(t, s)v(x - t, y - s)dsdt$$

Et La tranformée de Fourier en deux dimensions est définie par :

$$\mathcal{F}u(\lambda_1, \lambda_2) = \int_{\mathbb{R}^2} u(x_1, x_2)e^{2i\Pi(x_1\lambda_1 + x_2\lambda_2)}dx_1dx_2$$

On montre que $\mathcal{F}$ est une bijection de $\mathcal{S}(\mathbb{R}^2)$ dans $\mathcal{S}(\mathbb{R}^2)$ (fonctions à décroissance rapide), avec la relation d'équivalence d'égalité presque partout, ce qui nous permet de définir $\mathcal{F}^{-1}$. Comme dans le cas monodimensionnel on a la relation suivante :

$$\mathcal{F}(u * v) = \mathcal{F}u\mathcal{F}v$$

1.1.3 floutage et convolution

Le floutage d'une image se traduit mathématiquement par une convolution par un noyau h. Connaître ce noyau c'est pouvoir remonter à l'image d'origine. En effet, v est le résultat d'un floutage de u si et seulement si il existe h tel que $v = h * u$. Ainsi on a

$$\begin{aligned} v &= h * u \\ \mathcal{F}v &= \mathcal{F}(h * u) \\ \mathcal{F}v &= \mathcal{F}h\mathcal{F}u \\ u &= \mathcal{F}^{-1}\left(\frac{\mathcal{F}v}{\mathcal{F}h}\right) \end{aligned} \tag{1.1}$$

1.2 Noyau de Convolution et Phase

1.2.1 Relation entre le Noyau de Convolution et la Phase

En électromagnétisme, est appelé front d'onde une surface d'isovaleur de la phase. Physiquement, on observe que la déformation, le floutage de l'image provient de la déformation du front d'onde lumineux : si on suppose que la lumière réémise par l'échantillon possède, à l'origine, un front d'onde plan, en traversant l'échantillon ce front d'onde va se déformer. C'est ce phénomène qui déformera notre image. On voit donc très bien que le noyau de convolution dépend directement de la phase. La relation établie est la suivante :

$$h(\phi) = |\mathcal{F}^{-1}(pe^{i\phi})|$$

Avec p la fonction représentant la pupille du capteur d'image. (On prendra pour le reste du projet une indicatrice)

1.2.2 Les polynômes de Zernike

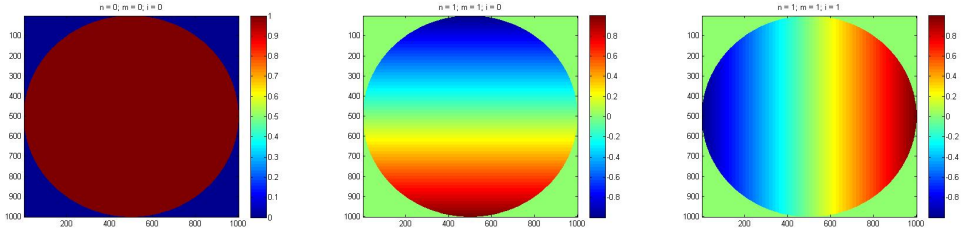
En réalité, la déformation du front d'onde n'est pas totalement inconnue, elle se décompose suivant plusieurs aberrations, c'est-à-dire déformations élémentaires. On appelle polynôme de Zernike tout élément de l'espace vectoriel engendré par ces aberrations. Les éléments de la base de cet espace dépendent de 3 entiers n,m et i. En coordonnées polaires, on a :

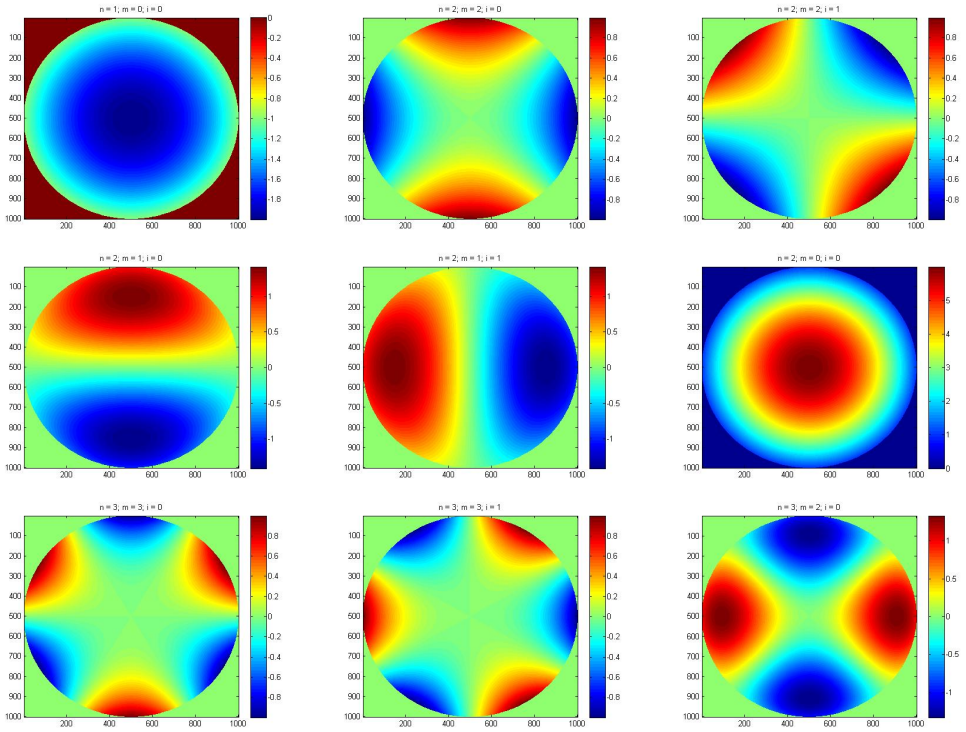
$$P_{n,m,i} = R_{n,m}(\rho)G_{i,m}(\psi)$$

avec

$$\begin{cases} R_{n,m}(\rho) = \sum_{s=0}^{n-m} (-1)^s C_{2n-m-s}^{n} C_{n}^{s} \rho^{2n-m-2s} \\ G_{i,m}(\psi) = \begin{cases} \cos(m\psi) \text{si i est pair} \\ \sin(m\psi) \text{si i est impair} \end{cases} \end{cases}$$

Voici les premiers éléments de la base de Zernike rangés par indice croissants

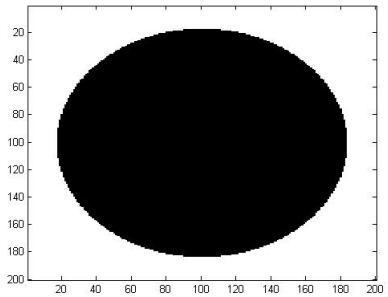




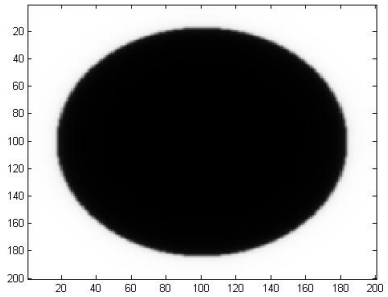
On va donc calculer la phase comme une combinaison linéaire de ces aberrations. Bien sûr, plus n et m sont grands, moins les aberrations sont probables. On va donc tronquer cette base en un nombre fini de polynômes.

1.2.3 Simulation

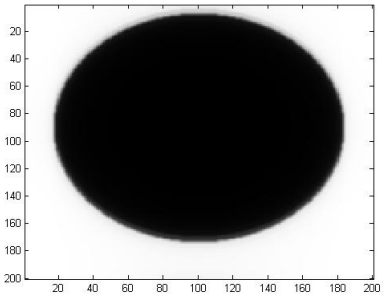
L'image sur laquelle on va effectuer nos opérations est une image de taille 100 par 100 que l'on va convoluer avec notre système. On peut observer le type de floutage que chaque élément génère. L'image nette se présente comme ceci :

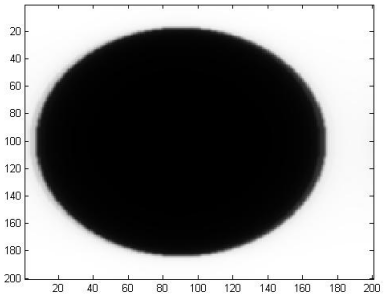


Le floutage avec le polynôme n°0 génère un flou uniforme. il caractérise par exemple un décalage entre le point d'acquision de l'image et le plan focal du système optique :

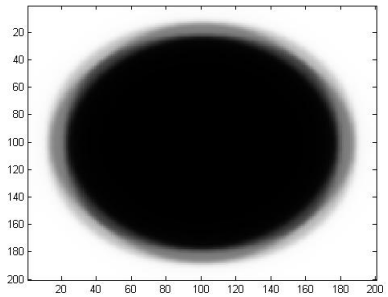


Le floutage avec les polynômes 1 et 2 génère respectivement un flou selon l'axe des y et des x. Ces deux polynômes représentent une inclinaison du plan d'acquisition de l'Image avec le plan focal du système optique :

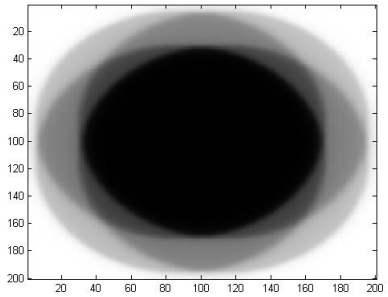




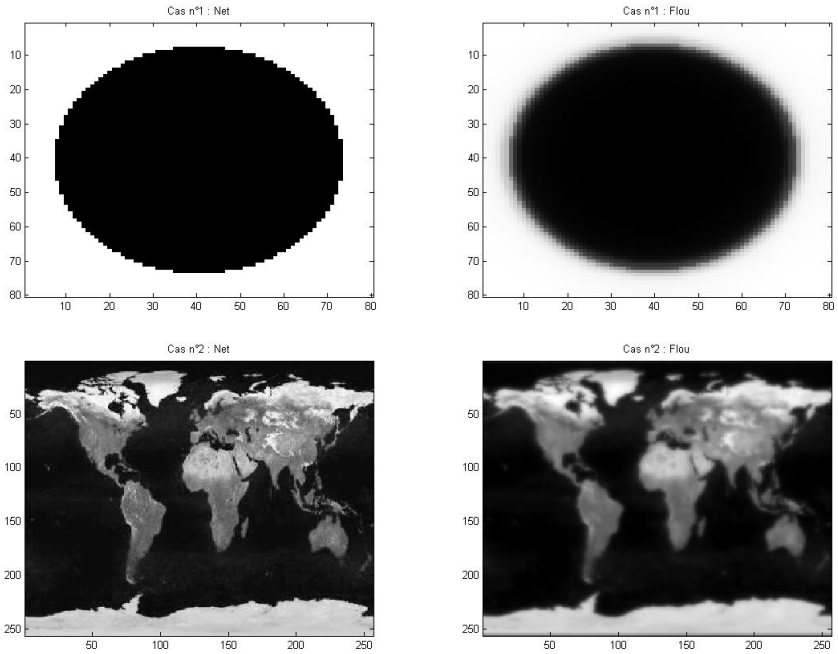
Le floutage avec le polynôme 3 génère un flou selon le sens radial. On retrouve ce polynôme par exemple dans le cas où la zone focale n'est pas plane.



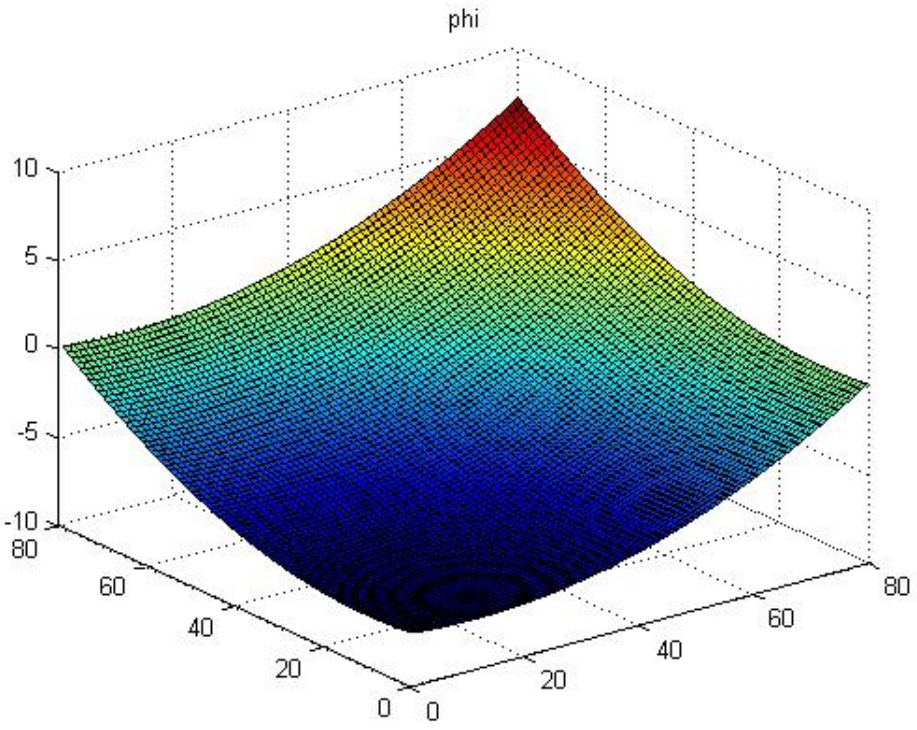
Le floutage avec le polynôme 4 ou 5 génère un flou astigmatique : l'image en plus d'être floutée est déformée.



Voyons maintenant un flou d'une image par un déphasage décomposé sur les 4 premiers coefficients de la base de Zernike (on prendra z égal à 5 sur chacune de ses composantes).



On voit bien qu'à l'oeil nu on ne peut pas décomposer les aberrations élémentaires (Il faut bien que notre algorithme soit utile !). Voici la phase pour le même z considéré :


phi
10
5
0
-5
-10
80
60
40
20
0
0
20
40
60
80

Chapitre 2

Calcul et Défloutage d'une Image

2.1 Positionnement du Problème

Pour obtenir un algorithme de défloutage, plutôt que calculer la fonction de transfert, on va s'intéresser la phase ϕ. Pour cela on utilise une méthode qui s'appelle la diversité de phase. Soit $u_f(\phi)$, notre image floue. la diversité de phase consiste en réalité à prendre plusieurs images : l'une prise normalement et les autres prises avec un dephasage constant et surtout connu, ϕ_t. Dans le cadre de notre projet, on va se contenter de deux images.On note alors u_1 l'image prise à déphasage ϕ et u2 celle prise à déphasage $\phi + \phi_t$. On veut donc résoudre

$$\left\{ \begin{array}{rcl} u_f(\phi) & = & u_1 \\ u_f(\phi + \phi_t) & = & u2 \end{array} \right.$$

Or le chapitre précédent nous dit que $u_f(\phi) = \mathcal{F}^{-1}(\mathcal{F}u\mathcal{F}h(\phi))$ On ramène donc notre problème à

$$\left\{ \begin{array}{rcl} \mathcal{F}^{-1}(\mathcal{F}u\mathcal{F}h(\phi)) & = & u_1 \\ \mathcal{F}^{-1}(\mathcal{F}u\mathcal{F}h(\phi + \phi_t)) & = & u2 \end{array} \right.$$

En discrétisant le problème on obtient (u,u_1,u_2 deviennent des vecteurs discrets de $\mathbb{R}^N$) :

$$\left\{ \begin{array}{rcl} F^{-1}(\mathrm{diag}(Fh(\phi))Fu) & = & u_1 \\ F^{-1}(\mathrm{diag}(Fh(\phi + \phi_t))Fu) & = & u2 \end{array} \right.$$

En notant $H_\phi = F^{-1}(\mathrm{diag}(Fh(\phi))F$ on a :

$$\left\{ \begin{array}{rcl} H_\phi u & = & u_1 \\ H_{\phi+\phi_t} u & = & u2 \end{array} \right.$$

Ce système étant sur déterminé on résoud un système des moindres carrés :

$$\underset{u,\phi}{\mathrm{argmin}} \left(\|H_\phi u - u_1\|_2^2 + \|H_{\phi+\phi_t} u - u_2\|_2^2 \right)$$

en ajoutant des termes de régularisation pour limiter le bruit

$$\underset{u,\phi}{\mathrm{argmin}} \left(\alpha\|H_\phi u - u_1\|_2^2 + \alpha\|H_{\phi+\phi_t} u - u_2\|_2^2 + \beta\|\nabla_d u\|_2^2 + \gamma\|\phi\|_2^2 \right)$$

Avec ∇_d le gradient spatial. Autrement dit $\nabla_d = \begin{pmatrix} \partial_x \\ \partial_y \end{pmatrix}$.

On note

$$J_u(\phi) = \left(\alpha\|H_\phi u - u_1\|_2^2 + \alpha\|H_{\phi+\phi_t} u - u_2\|_2^2 + \beta\|\nabla_d u\|_2^2 + \gamma\|\phi\|_2^2 \right)$$

Le problème devient donc

$$\underset{u,\phi}{\text{argmin}} J_u(\phi)$$

ce qui implique

$$\begin{cases} \partial_u J_u(\phi) = 0 \\ \partial_\phi J_u(\phi) = 0 \end{cases}$$

2.2 Calcul de u en fonction de ϕ

$$\begin{aligned} & \partial_u J_u(\phi) = 0 \\ \Rightarrow \quad & \partial_u \left(\alpha \|H_\phi u - u_1\|_2^2 + \alpha \|H_{\phi+\phi_t} u - u_2\|_2^2 + \beta \|\nabla_d u\|_2^2 + \gamma \|\phi\|_2^2 \right) = 0 \end{aligned}$$

Après plusieurs calculs on obtient (cf annexes) :

$$u(\phi) = F^{-1} D(\phi) F b(\phi)$$

Avec

$$\begin{cases} D(\phi) = \text{diag}(d(\phi)) \\ d(\phi), \text{ vecteur de } \mathbb{R}^N \text{ tel que } d(\phi)_i = \frac{1}{|Fh(\phi)_i\|^2 + |Fh(\phi+\phi_t)_i\|^2 + |Fh_{x_i}\|^2 + |Fh_{y_i}\|^2} \\ b(\phi) = H_\phi^T u_1 + H_{\phi+\phi_t}^T u_2 \\ h_x \text{ et } h_y, \text{ les masques de convolution associés à } \partial_x \text{ et } \partial_y \end{cases}$$

Le problème devient finalement,

$$\underset{\phi}{\text{argmin}} J(\phi)$$

Avec

$$\begin{aligned} J(\phi) &= J_{u(\phi)}(\phi) \\ &= \left(\frac{\alpha}{2} \|H_\phi u(\phi) - u_1\|_2^2 + \frac{\alpha}{2} \|H_{\phi+\phi_t} u(\phi) - u_2\|_2^2 + \frac{\beta}{2} \|\nabla_d(\phi)\|_2^2 + \frac{\gamma}{2} \|\phi\|_2^2 \right) \end{aligned}$$

On note enfin

$$\begin{cases} J_1(\phi) = \frac{1}{2} \|H_\phi u(\phi) - u_1\|_2^2 \\ J_1'(\phi) = \frac{1}{2} \|H_{\phi+\phi_t} u(\phi+\phi_t) - u_2\|_2^2 \\ J_2(\phi) = \frac{1}{2} \|\nabla_d u(\phi)\|_2^2 \\ J_3(\phi) = \frac{1}{2} \|\phi\|_2^2 \end{cases}$$

2.3 Calcul des Jacobiennes et des Gradients

Tous les calculs ici sont réalisés en annexe, seuls les résultats seront exposés.

2.3.1 Jacobiennes des fonctions intermédiaires

Jac_h

en notant $c(\phi) = \text{diag}(p) e^{i\phi}$ avec $\begin{cases} p, \text{ le vecteur discrétisé associé à pupille.} \\ e^{i\phi}, \text{ le vecteur discrétisé associé à la fonction } e^{i\phi} \end{cases}$

$$
\begin{aligned}
Jac_h(\phi) &= i\,\mathrm{diag}(\overline{F^{-1}c(\phi)})F^{-1}\,\mathrm{diag}(c(\phi)) - i\,\mathrm{diag}(F^{-1}c(\phi))\overline{F^{-1}}\,\mathrm{diag}(\overline{c(\phi)}) \\
&= 2\mathrm{Im}\left(\mathrm{diag}(F^{-1}c(\phi))\overline{F^{-1}}\,\mathrm{diag}(\overline{c(\phi)})\right) \\
Jac_h(\phi)^T &= i\,\mathrm{diag}(c(\phi))\overline{F}\,\mathrm{diag}(\overline{F^{-1}c(\phi)}) - i\,\mathrm{diag}(\overline{c(\phi)})F\,\mathrm{diag}(F^{-1}c(\phi)) \\
&= 2\mathrm{Im}\left(\mathrm{diag}(\overline{c(\phi)})F\,\mathrm{diag}(F^{-1}c(\phi))\right)
\end{aligned}
$$

Jac_b

$$
\begin{aligned}
Jac_b(\phi) &= F^{-1}\,\mathrm{diag}(Fu_1)\overline{FJac_h(\phi)} + F^{-1}\,\mathrm{diag}(Fu_2)\overline{FJac_h(\phi+\phi_t)} \\
Jac_b(\phi)^T &= \left(\overline{Jac_h(\phi)}^T F^{-1}\,\mathrm{diag}(Fu_1) + \overline{Jac_h(\phi+\phi_t)}^T F^{-1}\,\mathrm{diag}(Fu_2)\right)\overline{F}
\end{aligned}
$$

Jac_d

On note

$$
\begin{aligned}
L(\phi) &= \mathrm{diag}(Fh(\phi))\overline{FJac_h(\phi)} + \mathrm{diag}(\overline{Fh(\phi)})FJac_h(\phi) \\
&= 2Re\left(\mathrm{diag}(Fh(\phi))\ \overline{FJac_h(\phi)}\right) \\
L(\phi)^T &= \overline{Jac_h(\phi)}^T F^{-1}\,\mathrm{diag}(Fh(\phi)) + Jac_h(\phi)^T\overline{F^{-1}}\,\mathrm{diag}(\overline{Fh(\phi)}) \\
&= 2Re\left(\overline{Jac_h(\phi)}^T F^{-1}\,\mathrm{diag}(Fh(\phi))\right)
\end{aligned}
$$

Ainsi,

$$
\begin{aligned}
Jac_d(\phi) &= -D(\phi)^2\left(L(\phi) + L(\phi+\phi_t)\right) \\
Jac_d(\phi)^T &= -\left(L(\phi)^T + L(\phi+\phi_t)^T\right)D(\phi)^2
\end{aligned}
$$

Jac_u

$$
\begin{aligned}
Jac_u(\phi) &= F^{-1}\left(\mathrm{diag}(Fb(\phi))Jac_d(\phi) + D(\phi)FJac_b(\phi)\right) \\
Jac_u(\phi)^T &= \left(Jac_d(\phi)^T\,\mathrm{diag}(Fb(\phi)) + Jac_b(\phi)^T\overline{F^{-1}}D(\phi)\right)\overline{F}
\end{aligned}
$$

2.3.2 Gradients des Fonctions Coût

∇J_1

$$
\nabla J_1(\phi) = \left(Jac_u(\phi)^T\overline{F^{-1}}\,\mathrm{diag}(Fh(\phi) + Jac_h(\phi)^T\overline{F^{-1}}\,\mathrm{diag}(Fu(\phi)\right)\overline{F}\left(H_\phi u(\phi) - u_1\right)
$$

$\nabla J_1'$

$$
\nabla J_1'(\phi) = \left(Jac_u(\phi')^T\overline{F^{-1}}\,\mathrm{diag}(Fh(\phi') + Jac_h(\phi')^T\overline{F^{-1}}\,\mathrm{diag}(Fu(\phi')\right)\overline{F}\left(H_{\phi'}u(\phi') - u_2\right)
$$

Avec $\phi' = \phi + \phi_t$

∇J_2

$$\nabla J_2(\phi) = Jac_u(\phi)^T \nabla_d^T \nabla_d u(\phi)$$

Avec

$$\nabla_d^T \nabla_d = F^{-1} \left(\mathrm{diag}(Fh_x)\, \mathrm{diag}(\overline{Fh_x}) + \mathrm{diag}(Fh_y)\, \mathrm{diag}(\overline{Fh_y}) \right) F$$

∇J_3

$$\nabla J_3(\phi) = \phi$$

2.4 Algorithmes utilisés

Les algorithmes présentés dans ce rapport sont des algorithmes susceptibles de converger vers une solution. Cependant, on n'en pas la certitude, étant donné que le problème qui nous est présenté se situe en mileu non convexe.

2.4.1 préliminaires

Les algorithmes sont implémentés sous Matlab. Il faut savoir qu'aucune matrice de taille NxN (N représentant le nombre de pixels de l'image) n'est calculé expicitement dans ces programmes pour deux raisons : d'une part les stocker en mémoire prendrait beaucoup de RAM et ensuite les calculs seraient beaucoup trop longs. On réalise alors des programmes qui remplacent les multiplications matricielles. Pour cela on procède de la manière suivante :

- $\mathrm{diag}(x)y$ est remplacé par $x. * y$
- Fx et $F^{-1}x$ sont respectivement remplacés par $fft2(x)/sqrt(n)$ et $ifft2(x) * sqrt(n)$ (il faut savoir que la transformé de Fourier de Matlab n'est pas une isométrie, c'est pourquoi on la normalise).
- $\overline{F}x$ est remplacé par $conj(fft2(conj(x))/sqrt(n))$

Tous les programmes de calcul de gradients et de Jacobiennes sont donnés en annexe.

2.4.2 Algorithme de Descente de Gradient

Cet algorithme est celui qui a le plus de chance de converger. Le principe est le suivant : on descend suivant le gradient multiplié par un coefficient multiplicatif qui se règle au fur et à mesure des itérations. L'algorithme s'arrête quand notre ϕ stagne autour d'un point qui sera notre minimum.

2.4.3 Algorithme de Gauss/Newton

Cet Algorithme est le plus efficace : la convergence est plus rapide, et le résultat est en général plus précis. Mais contrairement à l'algorithme de descente de gradient, même en milieu non convexe, il ne converge pas toujours. Nous n'avons malheureusement pas eu le temps d'en finir le codage, mais ce sera fait d'ici peu de temps !
Le principe est utilisé est le suivant :

- On considère la fonction J comme la norme d'un vecteur : $J = \frac{1}{2}\|V\|_2^2$.

– le vecteur V est en fait la concaténation de V_1, V_1', V_2 et ϕ, avec

$$\begin{cases} V_1(\phi) = H_\phi u(\phi) - u_1 \\ V_1'(\phi) = H_{\phi+\phi_t} u(\phi+\phi_t) - u_2 \\ v_2(\phi) = \nabla_d u(\phi) \end{cases}$$

– On réalise un algorithme de Gauss-Newton sur V
– A chaque itération, on calcule h tel que l'approximation d'ordre 1 de $V(\phi + h)$ soit nul.

$$\begin{aligned} 0 &= V(phi) + Jac_V(\phi)h \\ Jac_V(\phi)h &= -V(\phi) \end{aligned}$$

– Ainsi à chaque système linéaire à résoudre, on utilise le gradient conjugué, nos matrices n'étant pas explicitement calculées.

2.5 Ajout des coefficients de Zernike

En réalité, l'optimisation que l'on désire effectuer est une optimisation sous contraintes : en effet ϕ appartient à l'espace de Zernike. Si on tronque la base des polynômes de Zernike à un certain rang n_z, il existe une matrice P_z telle que

$$\phi = P_z z$$

Avec ϕ le vecteur colonne représentant notre phase et z nos coefficients de Zernike. On se ramène donc au problème suivant :

$$\underset{z \in \mathbb{R}^{n_0}}{\operatorname{argmin}} J_z(z)$$

Avec,

$$J_z(z) = J(P_z z)$$

On montre facilement que

$$\nabla J_z(z) = P_z^T \nabla J(P_z z)$$

L'avantage de l'introduction des polynômes de Zernike est que l'on réduit notre espace de dimension N (nombre de pixels de l'image) à un espace de dimension n_z (on prend $n_z = 4$ dans les simulations).

Chapitre 3

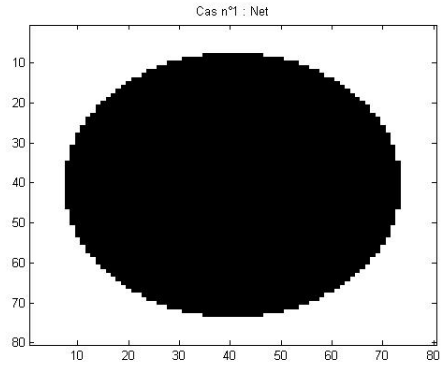
Résultats Obtenus

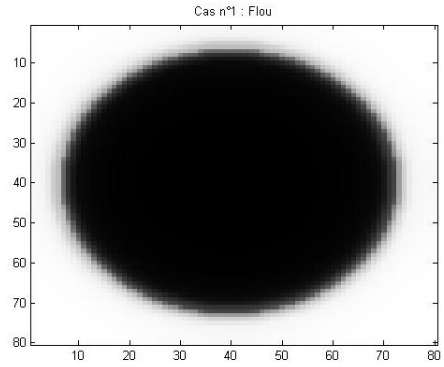
3.1 Un petit point sur nos coefficients α, β, et γ

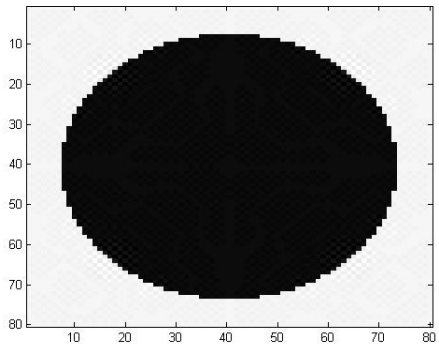
Après avoir arrangé maintes fois nos coefficients, nous nous sommes aperçus que les coefficients β et γ devaient être significativement très petits devant le α. En effet, si on veut par exemple trop régulariser le gradient, l'image obtenue devient encore plus floue que l'image acquise : ce n'est pas ce que l'on désire. Ce phénomène vient du fait que notre image présente des discontinuités (contours !). Ensuite si on veut trop minimier phi, la solution n'a plus de sens. Ces coefficients sont tout de même importants, car ils permettent de réguler la solution. un β nul annulerait la fonction d en certains points, et on ne pourrait pas obtenir de solution numérique. De plus, une contrainte sur ϕ est nécéssaire, sinon notre problème serait mal conditionné : un peu de bruit sur notre image floutée et le résultat s'écarte énormément de notre image de départ.

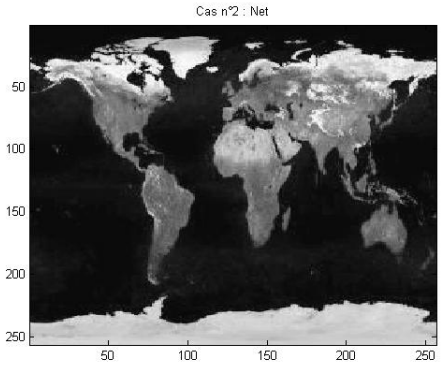
3.2 Test dans des conditions optimal

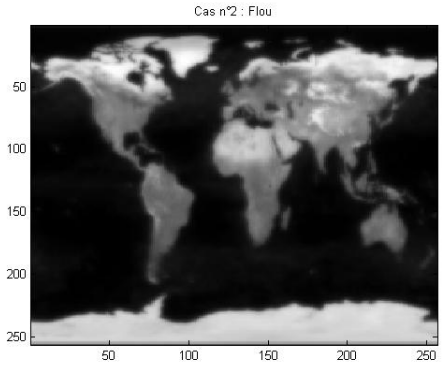
On prend notre point de départ de l'algorithme extrêment proche de notre solution minimum. On fait ensuite tendre β et γ vers 0. Etant donné que, lors de la simulation, l'image n'a pas été bruitée, on se ramène alors à un problème d'interpolation. On retombe donc sur notre image de départ. Ces conditions sur β et γ restent très restrictives, et dans le cas réel ne seront jamais rencontrées. Cependant notre algorithme prouve bien qu'il existe un réglage (α,β,γ) tel que le minimum de J sous ces paramètres soit proche de notre image réelle. En simulant par exemple notre disque et notre map-monde présentés dans la première partie, on obtient :
(dans l'ordre : l'image de départ, l'image floutée, et l'image défloutée)

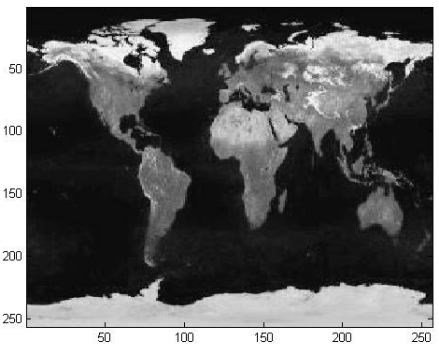












Les paramètres choisis ont été les suivants : $\alpha = 1, \beta = 10^{-}10, \gamma = 0$. En ce qui concerne le

ϕ, l'erreur relative entre la solution calculée et la solution exacte est de l'ordre de 10^{-7}.

La différence entre l'image nette et l'image défloutée vient plus de la déconvolution que du calcul d'optimisation : étant donné que Fh est nul en certain points, il est nécéssaire de lui rajouter un terme epsilon pour effectuer la déconvolution. (en notation matlab) :

$$u_{defloute} = F^{-1}(Fu_{floue} \,./\, (Fh + epsilon))$$

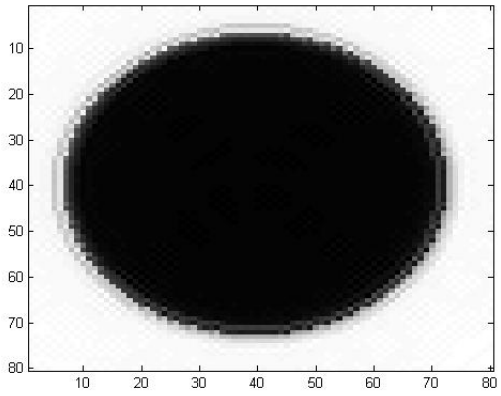
3.3 Problèmes rencontrés

3.3.1 Existence de plusieurs minima

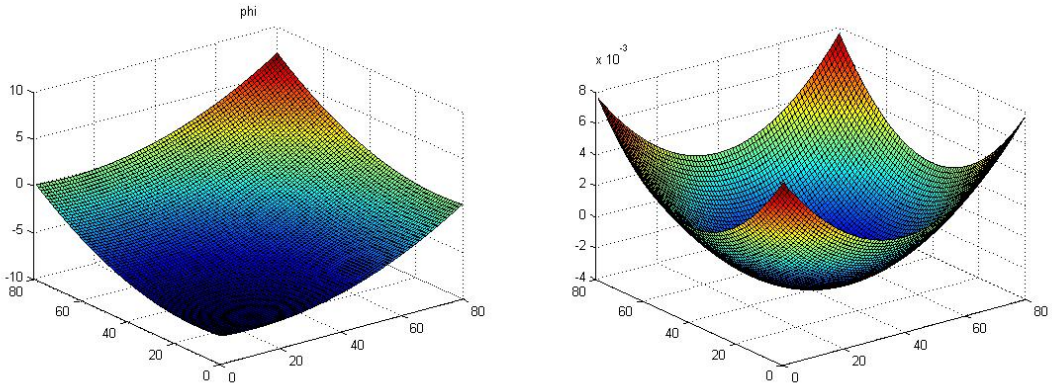
Si, en guise de point départ, on va choisir un point relativement éloigné de notre solution, l'algorithme va converger vers un autre point qui lui aussi sera un minimum local : notre fonction J n'est pas convexe ! Mieux encore, la solution la plus proche de notre phase réelle n'est pas le minimum global de la fonction J dans le cas optimal (c'est-à-dire $\gamma, \beta \to 0$: cas d'interpolation). Il est donc nécessaire de faire varier les coefficients, de telle manière que, pour une image donnée, on augmente les chances de tomber sur la solution désirée.

3.3.2 Convergence vers des solutions très éloignées de la solution théorique

Dans la plupart des cas, quand l'on fait varier les coefficients, notre l'algorithme ne converge pas vers la solution désirée, même en partant d'un point très proche de notre solution. Mais l'allure de l'image finale reste toujours semblable (d'un point de vue intuitif, non d'un point de vue numérique...). Voici son allure :



Comparons maintenant le ϕ calculé (à droite) avec le ϕ de base (à gauche) :



On voit bien que les deux ϕ sont radicalement différents tant sur l'ordre, que sur l'allure ! Il faut savoir aussi que le seul paramètre qui a changé par rapport au cas idéal est γ qui est passé de 0 à 10^{-10} ! Ceci nous fait prendre conscience de l'importance du jeu sur les coefficients.

3.3.3 Jeu sur les coefficients

Le jeu sur les coefficients est très délicat, le ϕ obtenu dans tous les cas, varie énormément si l'on change un paramètre. Nous n'avons malheureusement pas eu le temps de trouver les bons paramètres pour arriver à une image optimale.

Conclusion

Bilan du projet

Pour conlure, bien que nous ayons des résultats, en l'état actuel des choses, ce projet n'est pas directement utilisable. Nous avons montré expérimentalement qu'il est possible de trouver des coefficients α, β et γ, tels que l'on puisse déflouter notre image.

De plus, suite au temps passé à debogger tous nos programmes, nous pouvons affirmer avec certitude que tous nos programmes et nos calculs sont justes : ils constituent une bonne base pour une continuation probable du projet.

Cependant un certain travail reste à faire :
- Terminer l'algorithme de Gauss-Newton pour une convergence plus rapide et plus précise...
- Régler les fameux coefficients α, β et γ
- Gérer les problèmes de bruit, qui ont été pris en compte dans le modèle mais non testés (Il fallait déjà que notre programme marche dans le cas non bruité !)
- Tester notre programme en situation réelle !
- Implémenter le programme sur carte graphique

Je dirais donc que ce projet n'a été qu'une amorce parmi toute la recherche qu'il reste à faire. Il est par ailleurs assez frustrant de ne pas en arriver à bout. Un jour peut-être (et sûrement dans le cas de Mathieu !), nous auront la possibilité de continuer tout ce travail.

Enfin, je dirais que le modèle à partir duquel nous sommes partis possède une grosse lacune : pour pouvoir appliquer la transformée de Fourier, nous devons supposer que la fonction de transfert du système optique est invariante par translation (c'est-à-dire un échantillon homogène et isotrope). Or cette hypothèse reste physiquement forte, et la preuve en est que ce n'est pas réellement le cas. Donc en situation réelle, il y a beaucoup de chances que notre algorithme soit inefficace.

Bilan Personnel

Ce projet nous a permi de mettre un premier pas dans le monde de l'imagerie. De plus, c'est le premier projet dans le cadre scolaire, dans lequel on bénéficiait d'une certaine autonomie. Nous avons aussi dû combiner toutes nos connaissances mathématiques telles que l'algèbre ou l'analyse fonctionnelle, informatiques telles que la programmation ou l'analyse numérique et une bonne dose de patience, pour mettre un oeuvre une solution qui est acceptable. C'est pourquoi, nous avons trouvé ce projet réellement enrichissant et la plupart du temps motivant, bien que frustrant pour son manque de résultats concrets.

En effet, on a couru après le temps pour mener ce projet jusqu'à des résultats, et nous ne sommes pas arrivés à un programme directement exploitable : aurait-il fallut y consacrer le semestre entièrement ?

Le SPIM aura donc, indirectement, participé à nos premiers débuts dans la vie professionnelle : tant sur les connaissances pratiques qu'ils nous a apportées que sur la contrainte du temps : il représente ce qu'est un projet à part entière dans le monde professionnel.

Bibliographie

[1] Optical an Infrared Science Laboratory, Environmental Research Institue of Michigan
Richard G. PAXMAN, Timothy J. SCHULZ, James R FIENUP
Joint estimation of object by using phase diversity.
July 1992.

[2] Warren J. SMITH
Modern Optical Engineering.
Third Edition

[3] Belin Sup
Agnès MAUREL et Jean Marie MALBEC
Optique Géométrique.

[4] Montana State University, UCLA, and Wake Forest University
CurtisR. VOGEL, Tony CHAN, and Robert PLEMMONS
Fast ALgorithms for Phase Diversity-Based Blind Deconvolution .

[5] Pierre STROCK
Les Polynômes de Zernike.

ANNEXES

Calcul de u en fonction de phi

On veut calculer u en fonction de phi :
On a

$$\nabla_u J_u(\phi) = \alpha H_\phi^T (H_\phi u - u_1) + \alpha H_{\phi+\phi_t}^T (H_{\phi+\phi_t} u - u_2) + \beta \nabla_d^T \nabla_d u$$

On calcul u tel que $\nabla_u J_u(\phi) = 0$. On a donc (on pose $\lambda = \frac{\beta}{\alpha}$)

$$\begin{aligned} \alpha H_\phi^T (H_\phi u - u_1) + \alpha H_{\phi+\phi_t}^T (H_{\phi+\phi_t} u - u_2) + \beta \nabla_d^T \nabla_d u &= 0 \\ \left(H_\phi^T H_\phi + H_{\phi+\phi_t}^T H_{\phi+\phi_t} + \lambda \nabla_d^T \nabla_d \right) u &= H_\phi^T u_1 + H_{\phi+\phi_t}^T u_2 \end{aligned} \quad (1)$$

Or on sait que :
- $H_\phi = F^{-1} diag(Fh(\phi)) F$
- H_ϕ est réelle soit $H_\phi^T = H_\phi^*$

On en déduit que

$$\begin{aligned} H_\phi^T H_\phi &= H_\phi^* H_\phi \\ &= F * diag(\overline{Fh(\phi)}) F^{-1^*} F^{-1} diag(Fh(\phi)) F \\ &= F^{-1} diag(\overline{Fh(\phi)}) F F^{-1} diag(Fh(\phi)) F \\ &= F^{-1} diag(|Fh(\phi)|^2) F \\ H_{\phi+\phi_t}^T H_{\phi+\phi_t} &= F^{-1} diag(|Fh(\phi+\phi_t)|^2) F \end{aligned}$$

Avec $|Fh(\phi)|^2$ le vecteur de taille N contenant les carrés du module de chaque composante de $Fh(\phi)$. De plus $\nabla_d = \begin{pmatrix} \partial_x \\ \partial_y \end{pmatrix}$

Avec ∂_x et ∂_y des opérateurs matricielles reprèsentant la dérivée discrète en x et en y. On montre qu'il existe h_x et h_y tels que ∂_x soit une convolution par h_x et ∂_y soit une convolution par h_y.

$$h_x = \begin{bmatrix} 1 & 0 & \cdots & 0 & -1 \\ 0 & & \cdots & & 0 \\ \vdots & & & & \vdots \\ 0 & & \cdots & & 0 \end{bmatrix} \text{ et } h_y = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & & \cdots & 0 \\ -1 & 0 & \cdots & 0 \end{bmatrix}$$

On peut donc dire que :

$$\begin{cases} \partial_x = F^{-1} \operatorname{diag}(Fh_x) F \\ \partial_y = F^{-1} \operatorname{diag}(Fh_y) F \end{cases}$$

Soit (∂_x et ∂_y étant des matrices réelles) :

$$\begin{aligned} \nabla_d^T \nabla_d &= (\partial_x^T \ \ \partial_y^T) \begin{pmatrix} \partial_x \\ \partial_y \end{pmatrix} \\ &= \partial_x^T \partial_x + \partial_y^T \partial_y \\ &= \partial_x^* \partial_x + \partial_y^* \partial_y \\ &= F^* \operatorname{diag}(\overline{Fh_x}) F^{-1^*} F^{-1} \operatorname{diag}(Fh_x) F + F^* \operatorname{diag}(\overline{Fh_y}) F^{-1^*} F^{-1} \operatorname{diag}(Fh_y) F \\ &= F^{-1} \operatorname{diag}(|Fh_x|^2) F + F^{-1} \operatorname{diag}(|Fh_y|^2) F \\ &= F^{-1} \left(\operatorname{diag}(|Fh_x|^2) + \operatorname{diag}(|Fh_y|^2) \right) F \end{aligned}$$

Avec $|Fh_x|^2$ (resp.$|Fh_y|^2$) le vecteur de taille N contenant les carrés du module de chaque composante de Fh_x (resp. Fh_y).

En réinjectant le tout dans (1) : on obtient

$$\begin{aligned}
\left(H_\phi^T H_\phi + H_{\phi+\phi_t}^T H_{\phi+\phi_t} + \lambda \nabla_d^T \nabla_d\right) u &= H_\phi^T u_1 + H_{\phi+\phi_t}^T u_2 \\
F^{-1}\left(\mathrm{diag}(|Fh(\phi)|^2) + \mathrm{diag}(|Fh(\phi+\phi_t)|^2) + \lambda\,\mathrm{diag}(|Fh_x|^2) + \lambda\,\mathrm{diag}(|Fh_y|^2)\right) Fu &= H_\phi^T u_1 + H_{\phi+\phi_t}^T u_2 \\
F^{-1}\,\mathrm{diag}\left(|Fh(\phi)|^2) + |Fh(\phi+\phi_t)|^2 + \lambda|Fh_x|^2 + \lambda|Fh_y|^2\right) Fu &= H_\phi^T u_1 + H_{\phi+\phi_t}^T u_2
\end{aligned} \tag{2}$$

Enfin on a

$$u = F^{-1} d(\phi) F b(\phi) \tag{3}$$

avec

$$\begin{cases} d(\phi), \text{ le vecteur de taille N tel que } d(\phi)_i = \dfrac{1}{|(Fh(\phi))_i|^2 + |(Fh(\phi+\phi_t))_i|^2 + \lambda|(Fh_x)_i|^2 + \lambda|(Fh_y)_i|^2} \\ b(\phi) = H_\phi^T u_1 + H_{\phi+\phi_t}^T u_2 \end{cases}$$

Calculs des Jacobiennes et Gradients

Dans tous ces calculs, nous ferons référence à à la propriété suivante :

Propriété (P) 1. *Si a et b sont deux vecteur en dimension finie alors*

1. $\mathrm{diag}(b)a = \mathrm{diag}(a)b$

2. $\mathrm{diag}(\mathrm{diag}(a)b) = \mathrm{diag}(a)\,\mathrm{diag}(b)$

De plus F étant une isométrie on a les relation suivante :
- ${F^{-1}}^T = \overline{F}$
- $F^T = \overline{F^{-1}}$

1 Jacobiennes des fonctions intermédiaires

1.1 $Jac_h(\phi)$

Matriciellement,

$$h(\phi) = \mathrm{diag}(\overline{F^{-1}\,\mathrm{diag}(p)e^{i\phi}})F^{-1}\,\mathrm{diag}(p)e^{i\phi}$$

Avec

$$\begin{cases} p, \text{ le vecteur discrétisé associé à pupille.} \\ e^{i\phi}, \text{ le vecteur discrétisé associé à la fonction} e^{i\phi} \end{cases}$$

Ainsi,

$$\begin{aligned} h(\phi+\epsilon) &= \mathrm{diag}(\overline{F^{-1}\,\mathrm{diag}(p)e^{i\phi+i\epsilon}})F^{-1}\,\mathrm{diag}(p)e^{i\phi+i\epsilon} \\ &= h(\phi) + \mathrm{diag}(\overline{F^{-1}\,\mathrm{diag}(p)\,\mathrm{diag}(e^{i\phi})(1_{\mathbb{R}^N}+i\epsilon)})F^{-1}\,\mathrm{diag}(p)\,\mathrm{diag}(e^{i\phi})(1_{\mathbb{R}^N}+i\epsilon) + o(\epsilon) \end{aligned}$$

On note à présent $c(\phi) = \mathrm{diag}(p)e^{i\phi}$.
La Propriété (P) nous dit que $c(\phi) = \mathrm{diag}(p)\,\mathrm{diag}\left(e^{i\phi}\right)$ On a donc

$$\begin{aligned} h(\phi+\epsilon) &= h(\phi) + \mathrm{diag}(\overline{F^{-1}\,\mathrm{diag}(c(\phi))(1_{\mathbb{R}^N}+i\epsilon)})F^{-1}\,\mathrm{diag}(c(\phi))(1_{\mathbb{R}^N}+i\epsilon) + o(\epsilon) \\ &= h(\phi) + \mathrm{diag}(\overline{F^{-1}\,\mathrm{diag}(c(\phi))i\epsilon})F^{-1}c(\phi) + \mathrm{diag}(\overline{F^{-1}c(\phi)})F^{-1}\,\mathrm{diag}(c(\phi))i\epsilon + o(\epsilon) \end{aligned}$$

En utilisant une fois de plus la propriété (P), on a :

$$\begin{aligned} \mathrm{diag}(\overline{F^{-1}\,\mathrm{diag}(c(\phi))i\epsilon})F^{-1}c(\phi) &= \mathrm{diag}(F^{-1}c(\phi))\overline{F^{-1}\,\mathrm{diag}(c(\phi))i\epsilon} \\ &= \mathrm{diag}(F^{-1}c(\phi))\overline{F^{-1}\,\mathrm{diag}(c(\phi))}i\epsilon, \text{ car } \epsilon \text{ est réel.} \end{aligned}$$

Enfin, on obtient

$$\begin{aligned} h(\phi+\epsilon) &= h(\phi) + \left(\mathrm{diag}(F^{-1}c(\phi))\overline{F^{-1}\,\mathrm{diag}(c(\phi))i} + \mathrm{diag}(\overline{F^{-1}c(\phi)})F^{-1}\,\mathrm{diag}(c(\phi))i\right)\epsilon + o(\epsilon) \\ Jac_h(\phi) &= i\,\mathrm{diag}(\overline{F^{-1}c(\phi)})F^{-1}\,\mathrm{diag}(c(\phi)) - i\,\mathrm{diag}(F^{-1}c(\phi))\overline{F^{-1}}\,\mathrm{diag}(\overline{c(\phi)}) \\ &= 2\mathrm{Im}\left(\mathrm{diag}(F^{-1}c(\phi))\overline{F^{-1}}\,\mathrm{diag}(\overline{c(\phi)})\right) \\ Jac_h(\phi)^T &= i\,\mathrm{diag}(c(\phi))\overline{F}\,\mathrm{diag}(\overline{F^{-1}c(\phi)}) - i\,\mathrm{diag}(\overline{c(\phi)})F\,\mathrm{diag}(F^{-1}c(\phi)) \\ &= 2\mathrm{Im}\left(\mathrm{diag}(\overline{c(\phi)})F\,\mathrm{diag}(F^{-1}c(\phi))\right) \end{aligned}$$

1.2 $Jac_b(\phi)$

On rappelle que $b(\phi) = H(\phi)^T u_1 + H(\phi+\phi_t)^T u_2$.
Ensuite, on note $v_1(\phi) = H(\phi)^T u_1$ et $v_2(\phi) = H(\phi+\phi_t)^T u_2$.
Comme $H(\phi)$ est réelle on a $H(\phi)^T = H(\phi)^*$ et $H(\phi+\phi_t)^T = H(\phi+\phi_t)^*$

$$\begin{aligned}
v_1(\phi+\epsilon) &= H(\phi+\epsilon)^T u_1 \\
&= H(\phi+\epsilon)^* u_1 \\
&= (F^{-1}\,\mathrm{diag}(Fh(\phi+\epsilon))F)^* u_1 \\
&= F^{-1}\overline{\mathrm{diag}(Fh(\phi+\epsilon))}Fu_1 \\
&= F^{-1}\,\mathrm{diag}\left(\overline{Fh(\phi+\epsilon)}\right)Fu_1 \\
&= F^{-1}\,\mathrm{diag}\left(\overline{Fh(\phi)} + \overline{FJac_h(\phi)\epsilon}\right)Fu_1 \\
&= F^{-1}\,\mathrm{diag}\left(\overline{Fh(\phi)}\right)Fu_1 + F^{-1}\,\mathrm{diag}\left(\overline{FJac_h(\phi)\epsilon}\right)Fu_1 \\
&= H(\phi)^T u_1 + F^{-1}\,\mathrm{diag}\left(\overline{FJac_h(\phi)\epsilon}\right)Fu_1 \\
&= v_1(\phi) + F^{-1}\,\mathrm{diag}(\overline{FJac_h(\phi)}\ \epsilon)Fu_1
\end{aligned}$$

Et d'après la propriété (P),

$$\mathrm{diag}(\overline{FJac_h(\phi)}\ \epsilon)Fu_1 = \mathrm{diag}(Fu_1)\overline{FJac_h(\phi)}\ \epsilon$$

On en déduit que

$$v_1(\phi+\epsilon) = v_1(\phi) + F^{-1}\,\mathrm{diag}(Fu_1)\overline{FJac_h(\phi)}\ \epsilon$$

d'où

$$Jac_{v_1}(\phi) = F^{-1}\,\mathrm{diag}(Fu_1)\overline{FJac_h(\phi)}$$

de même

$$Jac_{v_2}(\phi) = F^{-1}\,\mathrm{diag}(Fu_2)\overline{FJac_h(\phi+\phi_t)}$$

Enfin,

$$\begin{aligned}
Jac_b(\phi) &= Jac_{v_1}(\phi) + Jac_{v_2}(\phi) \\
Jac_b(\phi) &= F^{-1}\,\mathrm{diag}(Fu_1)\overline{FJac_h(\phi)} + F^{-1}\,\mathrm{diag}(Fu_2)\overline{FJac_h(\phi+\phi_t)} \\
Jac_b(\phi)^T &= \left(\overline{Jac_h(\phi)}^T F^{-1}\,\mathrm{diag}(Fu_1) + \overline{Jac_h(\phi+\phi_t)}^T F^{-1}\,\mathrm{diag}(Fu_2)\right)\overline{F}
\end{aligned}$$

1.3 $Jac_d(\phi)$

1.3.1 Calcul de $\nabla(d_i)$

Soit :

$$D(\phi) = \frac{1}{|Fh(\phi)|^2 + |Fh(\phi+\phi_t)|^2 + |Fh_x|^2 + |Fh_y|^2}.$$

On note d le vecteur tel que $d_i = \frac{1}{|Fh(\phi)|_i^2 + |Fh(\phi+\phi_t)|_i^2 + |Fh_x|_i^2 + |Fh_y|_i^2}$
Ainsi $D = \mathrm{diag}(d)$ D'une manière générale on notera

$$[.,.]_{\mathbb{C}} : \mathbb{C}^k \times \mathbb{C} \to \mathbb{R}^k$$

$$x, y \to \frac{1}{2}(\overline{x}y + x\overline{y})$$

où $k = n$ ou 1

(Si $k = 1$ vous noterez qu'il s'agit du produit scalaire réel de $\mathbb{R}$ dans $\mathbb{C}$
Si x et y dépendent de ϕ on montre que $\partial_{\phi_j}[x, y]_{\mathbb{C}} = [\partial_{\phi_j} x, y]_{\mathbb{C}} + [x, \partial_{\phi_j} y]_{\mathbb{C}}$
Avec cette notation on a

$$\partial_{\phi_j} d_i = \frac{-2([\partial_{\phi_j}(Fh(\phi))_i, (Fh(\phi))_i]_{\mathbb{C}} + [\partial_{\phi_j}(Fh(\phi+\phi_t))_i, (Fh(\phi+\phi_t))_i]_{\mathbb{C}})}{\left(|Fh(\phi)|_i^2 + |Fh(\phi+\phi_t)|_i^2 + |Fh_x|_i^2 + |Fh_y|_i^2\right)^2} \quad (k=1)$$

on en deduit que

$$\nabla(d_i) = \frac{-2([\nabla(Fh(\phi))_i, (Fh(\phi))_i]_{\mathbb{C}} + [\nabla(Fh(\phi+\phi_t))_i, (Fh(\phi+\phi_t))_i]_{\mathbb{C}})}{\left(|Fh(\phi)|_i^2 + |Fh(\phi+\phi_t)|_i^2 + |Fh_x|_i^2 + |Fh_y|_i^2\right)^2} \quad (k=n)$$

Que l'on peut ecrire plus simplement

$$\nabla(d_i) = -2\ d_i^2([\nabla(Fh(\phi))_i, (Fh(\phi))_i]_{\mathbb{C}} + [\nabla(Fh(\phi+\phi_t))_i, (Fh(\phi+\phi_t))_i]_{\mathbb{C}})$$

1.3.2 Calcul de $Jac_d(\phi)$

$$\begin{aligned}
\langle \nabla(d_i)(\phi), \epsilon \rangle &= \nabla(d_i)(\phi)^T \epsilon \\
&= -2\ d_i^2([\nabla(Fh(\phi))_i, (Fh(\phi))_i]_{\mathbb{C}} + [\nabla(Fh(\phi+\phi_t))_i, (Fh(\phi+\phi_t))_i]_{\mathbb{C}})^T \epsilon \\
&= -2\ d_i^2(\frac{1}{2}(\overline{\nabla(Fh(\phi))_i} \times (Fh(\phi))_i + \nabla(Fh(\phi))_i \times \overline{(Fh(\phi))_i})^T \epsilon \\
&\qquad + (\text{même chose en} \phi + \phi_t)) \\
&= -d_i^2((\nabla\overline{(Fh(\phi))_i} \times (Fh(\phi))_i + \nabla(Fh(\phi))_i \times \overline{(Fh(\phi))_i})^T \epsilon \\
&\qquad + (\text{même chose en} \phi + \phi_t)) \\
&= -d_i^2((Fh(\phi))_i \sum_j \epsilon_j \overline{(\nabla(Fh(\phi))_i)_j} + \overline{(Fh(\phi))_i} \sum_j \epsilon_j (\nabla(Fh(\phi))_i)_j \\
&\qquad + (\text{même chose en} \phi + \phi_t)) \\
&= -d_i^2((Fh(\phi))_i \sum_j \epsilon_j \overline{(FJac_h(\phi))_{ji}} + \overline{(Fh(\phi))_i} \sum_j \epsilon_j (FJac_h(\phi))_{ji} \\
&\qquad + (\text{même chose en} \phi + \phi_t)) \\
&= -d_i^2((Fh(\phi))_i (\ \overline{FJac_h(\phi)}\epsilon)_i + \overline{(Fh(\phi))}_i (\ FJac_h(\phi)\epsilon)_i \\
&\qquad + (\text{même chose en} \phi + \phi_t))
\end{aligned}$$

De plus,

$$(Jac_d(\phi)\epsilon)_i = \langle \nabla(d_i)(\phi), \epsilon \rangle$$

On en deduit que

$$\begin{aligned}
Jac_d(\phi)\epsilon &= -\text{diag}(d)^2(\text{diag}(Fh(\phi))\ \overline{FJac_h(\phi)}\epsilon + \text{diag}(\overline{Fh(\phi)})\ FJac_h(\phi)\epsilon + \ (\text{même chose en} \phi + \phi_t)) \\
&= -D^2(\text{diag}(Fh(\phi))\ \overline{FJac_h(\phi)} + \text{diag}(\overline{Fh(\phi)})\ FJac_h(\phi) + \ (\text{même chose en} \phi + \phi_t))\ \epsilon
\end{aligned}$$

en notant

$$\begin{aligned} L(\phi) &= \mathrm{diag}(Fh(\phi))\overline{FJac_h(\phi)} + \mathrm{diag}(\overline{Fh(\phi)})FJac_h(\phi) \\ &= 2Re\left(\mathrm{diag}(Fh(\phi))\ \overline{FJac_h(\phi)}\right) \\ L(\phi)^T &= \overline{Jac_h(\phi)}^T F^{-1}\,\mathrm{diag}(Fh(\phi)) + Jac_h(\phi)^T\overline{F^{-1}}\,\mathrm{diag}(\overline{Fh(\phi)}) \\ &= 2Re\left(\overline{Jac_h(\phi)}^T F^{-1}\,\mathrm{diag}(Fh(\phi))\right) \end{aligned}$$

On a

$$\begin{aligned} Jac_d(\phi) &= -D(\phi^2)\left(L(\phi) + L(\phi+\phi_t)\right) \\ Jac_d(\phi)^T &= -\left(L(\phi)^T + L(\phi+\phi_t)^T\right) D(\phi^2) \end{aligned}$$

1.4 $Jac_u(\phi)$

1.4.1 Linéarisation de $D(\phi\ +\epsilon) - D(\phi)$

$$\begin{aligned} D(\phi+\epsilon) - D(\phi) &= \mathrm{diag}(d(\phi+\epsilon)) - \mathrm{diag}(d(\phi)) \\ &= \mathrm{diag}(d(\phi+\epsilon) - d(\phi)) \\ &= \mathrm{diag}(Jac_d(\phi)\epsilon + o(\epsilon)) \\ &= \mathrm{diag}(Jac_d(\phi)\epsilon) + o(\epsilon) \end{aligned}$$

On en déduit que

$$\begin{aligned} (D(\phi\ +\epsilon) - D(\phi)) &= \mathrm{diag}\left(Jac_d(\phi)\epsilon\right) + o(\epsilon) \\ &= \mathrm{diag}\left(-\mathrm{diag}(d)^2\left(L(\phi)\epsilon + L(\phi+\phi_t)\epsilon\right)\right) + o(\epsilon) \end{aligned}$$

La Propriété (P) nous donne :

$$\begin{aligned} \mathrm{diag}\left(Jac_d(\phi)\epsilon\right) &= -\mathrm{diag}(d)^2\left(\mathrm{diag}\left(L(\phi)\epsilon\right) + \mathrm{diag}\left(L(\phi+\phi_t)\epsilon\right)\right) \\ &= -D(\phi)^2\left(\mathrm{diag}\left(L(\phi)\epsilon\right) + \mathrm{diag}\left(L(\phi+\phi_t)\epsilon\right)\right) \end{aligned}$$

De même

$$\begin{aligned} \mathrm{diag}\left(L(\phi)\right) &= \mathrm{diag}\left(\mathrm{diag}(Fh(\phi))\overline{FJac_h(\phi)} + \mathrm{diag}(\overline{Fh(\phi)})FJac_h(\phi)\right) \\ &= \mathrm{diag}(Fh(\phi))\,\mathrm{diag}\left(\overline{FJac_h(\phi)}\right) + \mathrm{diag}(\overline{Fh(\phi)})\,\mathrm{diag}\left(FJac_h(\phi)\right) \end{aligned}$$

1.4.2 Calcul de $Jac_u(\phi)$

$$\begin{aligned} u(\phi+\epsilon) - u(\phi) &= F^{-1}D(\phi+\epsilon)Fb(\phi+\epsilon) - F^{-1}D(\phi)Fb(\phi) \\ &= F^{-1}D(\phi+\epsilon)F(b(\phi) + Jac_b(\phi)\epsilon + o(\epsilon)) - F^{-1}D(\phi)Fb(\phi) \\ &= F^{-1}(D(\phi+\epsilon) - D(\phi))Fb(\phi) + F^{-1}D(\phi+\epsilon)FJac_b(\phi)\epsilon + o(\epsilon) \end{aligned}$$

Et d'après le paragraphe pérécédent

$$\begin{aligned} u(\phi+\epsilon)-u(\phi) &= F^{-1}(\mathrm{diag}(Jac_d(\phi)\epsilon)+o(\epsilon))Fb(\phi)+F^{-1}D(\phi+\epsilon)FJac_b(\phi)\epsilon+o(\epsilon) \\ &= F^{-1}\,\mathrm{diag}(Jac_d(\phi)\epsilon)Fb(\phi)+F^{-1}o(\epsilon)Fb(\phi)+F^{-1}D(\phi+\epsilon)FJac_b(\phi)\epsilon+o(\epsilon) \\ &= F^{-1}\,\mathrm{diag}(Jac_d(\phi)\epsilon)Fb(\phi)+F^{-1}D(\phi+\epsilon)FJac_b(\phi)\epsilon+o(\epsilon) \end{aligned}$$

et d'après la propriété (P) $\mathrm{diag}(Jac_d(\phi)\epsilon)Fb(\phi)=\mathrm{diag}(Fb(\phi))Jac_d(\phi)\epsilon$

$$u(\phi+\epsilon)-u(\phi)=F^{-1}\,\mathrm{diag}(Fb(\phi))Jac_d(\phi)\epsilon+F^{-1}D(\phi+\epsilon)FJac_b(\phi)\epsilon+o(\epsilon)$$

Il ne reste plus qu'à linéariser $F^{-1}D(\phi+\epsilon)FJac_b(\phi)\epsilon$

$$\begin{aligned} F^{-1}D(\phi+\epsilon)FJac_b(\phi)\epsilon &= F^{-1}(D(\phi)+0(1))FJac_b(\phi)\epsilon \\ &= F^{-1}(D(\phi))FJac_b(\phi)\epsilon+F^{-1}0(1)FJac_b(\phi)\epsilon \\ &\qquad\qquad \text{et } F^{-1}0(1)FJac_b(\phi)\epsilon \text{ est de l'ordre de } o(\epsilon) \\ &= F^{-1}(D(\phi))FJac_b(\phi)\epsilon+o(\epsilon) \end{aligned}$$

On en deduit que

$$\begin{aligned} u(\phi+\epsilon)-u(\phi) &= F^{-1}\,\mathrm{diag}(Fb(\phi))Jac_d(\phi)\epsilon+F^{-1}D(\phi)FJac_b(\phi)\epsilon+o(\epsilon) \\ &= F^{-1}(\mathrm{diag}(Fb(\phi))Jac_d(\phi)+D(\phi)FJac_b(\phi))\epsilon+o(\epsilon) \end{aligned}$$

On extrait alors

$$\begin{aligned} Jac_u(\phi) &= F^{-1}\left(\mathrm{diag}(Fb(\phi))Jac_d(\phi)+D(\phi)FJac_b(\phi)\right) \\ Jac_u(\phi)^T &= \left(Jac_d(\phi)^T\,\mathrm{diag}(Fb(\phi))+Jac_b(\phi)^T\overline{F^{-1}}D(\phi)\right)\overline{F} \end{aligned}$$

2 Gradient des fonctions coûts

2.1 ∇J_1 et ∇J_1

on note $V_1(\phi)=H_\phi u(\phi)-u_1$ on rappelle que

$$\begin{cases} J_1(\phi)=\frac{1}{2}\|H_\phi u(\phi)-u_1\|_2^2 \\ J_1'(\phi)=\frac{1}{2}\|H_\phi u(\phi+\phi_t)-u_2\|_2^2 \\ H_\phi=F^{-1}\,\mathrm{diag}(Fh(\phi)F \end{cases}$$

on note $V_1(\phi)=H_\phi u(\phi)-u_1$

$$\begin{aligned} V_1(\phi+\epsilon) &= H_{\phi+\epsilon}u(\phi+\epsilon)-u_1 \\ V_1(\phi+\epsilon) &= F^{-1}\,\mathrm{diag}(Fh(\phi+\epsilon)Fu(\phi+\epsilon)-u_1 \\ V_1(\phi+\epsilon) &= F^{-1}\,\mathrm{diag}\left(Fh(\phi)+FJac_h(\phi)\epsilon\right)F\left(u(\phi)+Jac_u(\phi)\epsilon\right))-u_1+o(\epsilon) \\ V_1(\phi+\epsilon) &= V_1(\phi)+F^{-1}\,\mathrm{diag}\left(FJac_h(\phi)\epsilon\right)Fu(\phi)+F^{-1}\,\mathrm{diag}(Fh(\phi))FJac_u(\phi)\epsilon+o(\epsilon) \end{aligned}$$

En utilisant la Propriété (P) :

$$V_1(\phi+\epsilon) = V_1(\phi)+F^{-1}\left(\mathrm{diag}\left(Fu(\phi)\right)FJac_h(\phi)+\mathrm{diag}(Fh(\phi))FJac_u(\phi)\right)\epsilon+o(\epsilon)$$

Soit :

$$\begin{aligned} Jac_{V_1}(\phi) &= F^{-1}\left(\mathrm{diag}\left(Fu(\phi)\right)FJac_h(\phi) + \mathrm{diag}(Fh(\phi))FJac_u(\phi)\right) \\ Jac_{V_1}(\phi)^T &= \left(Jac_h(\phi)^T\overline{F^{-1}}\,\mathrm{diag}\left(Fu(\phi)\right) + Jac_u(\phi)^T\overline{F^{-1}}\,\mathrm{diag}(Fh(\phi))\right)\overline{F} \end{aligned}$$

$$\begin{aligned} J_1(\phi+\epsilon) &= \frac{1}{2}\|V_1(\phi+\epsilon)\|_2^2 \\ &= \frac{1}{2}\|V_1(\phi) + Jac_{V_1}(\phi)\epsilon)\|_2^2 + o(\epsilon) \\ &= \frac{1}{2}\|V_1(\phi)\|_2^2 + \langle Jac_{V_1}(\phi)\epsilon, V_1(\phi)\rangle \\ &= J_1(\phi) + \langle \epsilon, Jac_{V_1}(\phi)^T V_1(\phi)\rangle \end{aligned}$$

On en déduit que

$$\begin{aligned} \nabla J_1(\phi) &= Jac_{V_1}(\phi)^T V_1(\phi) \\ &= \left(Jac_h(\phi)^T\overline{F^{-1}}\,\mathrm{diag}\left(Fu(\phi)\right) + Jac_u(\phi)^T\overline{F^{-1}}\,\mathrm{diag}(Fh(\phi))\right)\overline{F}\left(H_\phi u(\phi) - u_1\right) \end{aligned}$$

De même ,

$$\nabla J_1'(\phi) = \left(Jac_h(\phi')^T\overline{F^{-1}}\,\mathrm{diag}\left(Fu(\phi')\right) + Jac_u(\phi')^T\overline{F^{-1}}\,\mathrm{diag}(Fh(\phi'))\right)\overline{F}\left(H_{\phi'}u(\phi') - u_2\right)$$

Avec $\phi' = \phi + \phi_t$

2.2 ∇J_2

On rappelle que

$$J_2(\phi) = \frac{1}{2}\|\nabla_d u(\phi)\|_2^2$$

$$\begin{aligned} \nabla J_2(\phi+\epsilon) &= \frac{1}{2}\|\nabla_d u(\phi+\epsilon)\|_2^2 \\ &= \frac{1}{2}\|\nabla_d\left(u(\phi) + jac_u(\phi)\epsilon\right)\|_2^2 + o(\epsilon) \\ &= J_2(\phi) + \langle \nabla_d u(\phi), \nabla_d Jac_u(\phi)\epsilon\rangle + o(\epsilon) \\ &= J_2(\phi) + \langle Jac_u(\phi)^T\nabla_d^T\nabla_d u(\phi), \epsilon\rangle + o(\epsilon) \end{aligned}$$

d'où

$$\nabla J_2(\phi) = Jac_u(\phi)^T\nabla_d^T\nabla_d u(\phi)$$

2.3 ∇J_3

On rappelle que

$$J_3(\phi) = \frac{1}{2}\|\phi\|_2^2$$

Il est trivial que

$$\nabla J_3(\phi) = \phi$$

Programmes

I – Simulations

```
function [h ht] = convolueur(nrow,ncol,decphi,diametre_pupille,coeffs)
%% Programme qui fournit h(phi) et h(phi +phit) à partir des coeffs de
%% Zernike du diamètre de la pupille et de phit
[x,y] = meshgrid(linspace(-1,1,ncol),linspace(-1,1,nrow));
phi_calcul = calcul_phi(coeffs,[nrow ncol]);
p_calcul = func_p(x,y,diametre_pupille);

%% Calcul de h(phi)
h = infft((p_calcul).*exp(1i*(phi_calcul)));
h = h.*conj(h);
h = h/sum(h(:));

%% Calcul de h(phi+phit)
ht = infft((p_calcul).*exp(1i*(phi_calcul+decphi)));
ht = ht.*conj(ht);
ht = ht/sum(ht(:));
```

```
function New_IM = convolution(h,IM)
%% Programme qui réalise une convolution de IM par h
New_IM = real(infft(nfft(h).*nfft(IM)));
```

```
function New_IM = deconvolution(h,IM)
%% Programme qui déconvolue une image qui a été floutée par h
New_IM = real(infft(nfft(IM)./(nfft(h)+1e-15)));
```

```
function result=func_p(x,y,diametre)
%% Fonction qui représente la pupille
result = fftshift(double(x.^2+y.^2 < diametre^2/4));
```

```
function [ M ] = GetMat2(n,m)
%% fonction qui nous obtient l'image d'un cercle;
if nargin == 1
    [M1 M2] = meshgrid(linspace(-1,1,n),linspace(-1,1,n));
else
    [M1 M2] = meshgrid(linspace(-1,1,m),linspace(-1,1,n));
end
M = double(M1.*M1 + M2.*M2 > 0.7);
end
```

```
function M=calcul_phi(coeffs,taille)
%% fonction qui nous calcul phi à partir des coefficients de Zernike;
addpath('E:\Cours\4A\Projet_SPIM\Matlab\Zernick\');
l = length(coeffs);
Pz = func_Pz(l-1,taille);
M = reshape(Pz*coeffs(:),taille(1),taille(2));
```

```
%% Programme principal qui simule un floutage
close all;
diametre = 1.3;
coeffs =  5.0 *[1;1;1;1];
IM = GetMat2(80);
[h ht] = convolueur(size(IM,1),size(IM,2),0.1,diametre,coeffs);

IMfloue = convolution(h,IM);
IMflouеDec = convolution(ht,IM);
IMNet = deconvolution(h,IMfloue);

phi_theorique = calcul_phi(coeffs,size(IM));

figure(1);colormap gray(256); imagesc(IMfloue);
title('Cas n°1 : Flou');
figure(2);colormap gray(256); imagesc(IMNet);

figure(3);colormap gray(256); imagesc(IM);
title('Cas n°1 : Net');
figure(4); surf(phi_theorique);
title('phi');
save donnee IMfloue IMflouеDec IM phi_theorique diametre coeffs;
```

II- Les Polynômes de Zernike

```
function [n,m,i]=indices(k)
%% fonction qui nous donne les paramètres du i-ème élément des
coefficients de Zernike
n = 0;
u = 0;
while u < k
    n = n+1;
    u = n*(n+2);
end;
if u==k
    m=0;
    i=0;
    return;
end;
delta = k - n^2 +1;
i = mod(delta,2)==0;
if i==0
    delta = delta+1;
end;
m = n-delta/2+1;
```

```
function R = CoeffsR( n,m )
%% fonction qui calcules les coefficients radiaux du polynôme de Zernike
R = zeros(1,n-m+1);
```

```
if mod(n-m,2)==0
    mult = 1;
else
    mult = -1;
end;
for s = 1 : n-m+1
    R(s) = mult * nchoosek(n,s-1)*nchoosek(2*n-m-s+1,n);
    mult = - mult;
end;
```

```
function M = CoeffZernick( n,m,i,taille )
%% Fonction qui calcule les elements de la base de Zernike
M = zeros(taille);
R = CoeffsR(n,m);
c = mod(i,2)==0;
s = 1-c;
center = floor(taille / 2);
rhotot = norm(taille,2)/2;
for i = 1 : taille(1)
    for j = 1 : taille(2)
        vecteur = [i j]-center;
        rho = norm(vecteur,2)/rhotot;
        psy = angle(vecteur(1) + 1i*vecteur(2));
        if (rho == 0)
            if (m == 0)
                M(i,j) = R(1);
            elseif (n == m)
                M(i,j) = 0;
            else
                M(i,j) = 0;
            end;

        else
            M(i,j) = rho^(2*n-m)*polyval(R,rho^(2))*(c*cos(m*psy)+s*sin(m*psy));
        end;
    end;
end;
end
```

```
function M = func_Pz(l,taille)
%% fonction qui nous donne la matrice de passage des coeffs de Zernike
dans l'espace des phase (matrice réduit à la taille l)

M = zeros(prod(taille),l+1);
for k = 0 : l
    [n m i] = indices(k);
    Zer = CoeffZernick(n,m,i,taille);
    M(:,k+1) = Zer(:);
end;
```

III – Calculs

```
function M = nfft(X)
n = numel(X);
M = fft2(X);
M = M/sqrt(n);
```

```
function M = infft(X)
n = numel(X);
M = ifft2(X);
M = M*sqrt(n);
```

```
function h = func_h(phi,p)
h = p.*exp(1i*phi);
h = infft(h);
h = h.*conj(h);
```

```
function b = func_b(h_phi,h_phit,u1,u2)
b = infft(conj(nfft(h_phi)).*nfft(u1) + conj(nfft(h_phit)).*nfft(u2));
```

```
function d = func_d( p,phi,phi_t,lambda )
Fhx = zeros(size(phi));
Fhy = zeros(size(phi));
Fhx(1,1)= 1;
Fhx(end,1) = -1;
Fhy(1,1)= 1;
Fhy(1,end) = -1;
Fhx = nfft(Fhx);
Fhy = nfft(Fhy);
Fhphi = nfft(func_h(phi,p));
Fhphi_t = nfft(func_h(phi+phi_t,p));
Fhx = Fhx .* conj(Fhx);
Fhy = Fhy .* conj(Fhy);
Fhphi = Fhphi .* conj(Fhphi);
Fhphi_t = Fhphi_t.* conj(Fhphi_t);
d = 1./(Fhphi_t + Fhphi + lambda*(Fhx + Fhy));
```

```
function u = func_u(d,b)
u = infft(d.*nfft(b));
```

```
function Jach = Jac_h(phi,p,epsilon)
c = p.*exp(1i*phi);
Aepsilon = 1i*infft(c.*epsilon);
c = infft(c);
Jach = conj(c).*Aepsilon + conj(Aepsilon).*c;
```

```
function Jachtrans=Jac_h_trans(p,phi,epsilon)
if norm(imag(epsilon))~=0
    Jachtrans=Jac_h_trans(p,phi,real(epsilon)) + 1i *
Jac_h_trans(p,phi,imag(epsilon));
    return;
end;
c = infft(p.*exp(1i*phi));
c_epsb = c.*conj(epsilon);
A_cb_eps = p.*exp(1i*phi).*conj(nfft(-1i*c_epsb));
Jachtrans = A_cb_eps + conj(A_cb_eps);
```

```
function Jac_b = Jac_b(Jach_phi_eps,Jach_phit_eps,u1,u2)
Jac_b = infft((nfft(u1).*conj(nfft(Jach_phi_eps)) +
(nfft(u2).*conj(nfft(Jach_phit_eps)))));
```

```
function Jacbt = Jac_b_trans(u1,u2,p,phi,phit,epsilon)
if norm(imag(epsilon))~=0
    Jacbt = Jac_b_trans(u1,u2,p,phi,phit,real(epsilon)) + 1i *
Jac_b_trans(u1,u2,p,phi,phit,imag(epsilon));
    return;
end;
v1 = conj(nfft(conj(epsilon)));
v2 = nfft(u2) .* v1;
v1 = nfft(u1) .* v1;
v1 = infft(v1);
v2 = infft(v2);
v1 = Jac_h_trans(p,phi,v1);
v2 = Jac_h_trans(p,phi+phit,v2);
v1 = conj(v1);
v2 = conj(v2);

Jacbt = v1 + v2;
```

```
function Jacd = Jac_d (d, ffthphi, ffthphit, phi, phi_t, p,epsilon)
a = nfft(Jac_h(phi,p,epsilon));
a = conj(ffthphi).*a;
Jacd = a + conj(a);
a = nfft(Jac_h(phi+phi_t,p,epsilon));
a = conj(ffthphit).*a;
Jacd = Jacd + a + conj(a);
Jacd = -d.*d.*Jacd;
```

```
function Jacdtrans = Jac_d_trans(d,phi,phit, p,ffthphi, ffthphit, epsilon)
if norm(imag(epsilon))~=0
    Jacdtrans = Jac_d_trans(d,phi,phit, p,ffthphi, ffthphit,
real(epsilon)) + 1i * Jac_d_trans(d,phi,phit, p,ffthphi, ffthphit,
imag(epsilon));
    return;
end;
eps = -d.*d.*epsilon;
```

```
a = ffthphi.*eps ;
a = infft(a);
a = Jac_h_trans(p,phi,a);
Jacdtrans = a + conj(a);
a = ffthphit.*eps ;
a = infft(a);
a = Jac_h_trans(p,phi+phit,real(a))-1i*Jac_h_trans(p,phi+phit,imag(a));
Jacdtrans = Jacdtrans + a + conj(a);
```

```
function Jacu = Jac_u(b,Jacdeps,d,Jacbeps)
Jacu  = infft(nfft(b).*Jacdeps+d.* nfft(Jacbeps));
```

```
function Jacutrans = Jac_u_trans(d,p,u1,u2, phi,phi_t,fftb,ffthphi,
ffthphit, epsilon)
if norm(imag(epsilon))~=0
    Jacutrans = Jac_u_trans(d,p,u1,u2, phi,phi_t,fftb,ffthphi, ffthphit,
real(epsilon)) + 1i * Jac_u_trans(d,p,u1,u2, phi,phi_t,fftb,ffthphi,
ffthphit, imag(epsilon));
    return;
end;
ffteps = conj(nfft(conj(epsilon)));
a = fftb .* ffteps;
Jacutrans = Jac_d_trans(d,phi,phi_t,p,ffthphi,ffthphit,a);

a = d.*ffteps;
a = conj(infft(conj(a)));
a = Jac_b_trans(u1,u2,p,phi,phi_t,a);
Jacutrans = Jacutrans +a;
```

```
function gJ =
gradient_J1(p,phi,phit,ffthphi,ffthphit,d,fftb,fftuphi,ub,u1,u2)
K = infft(ffthphi.*fftuphi);
K = K - ub;
K = conj(nfft(conj(K)));
A = fftuphi.*K;
A = conj(infft(conj(A)));
A = Jac_h_trans(p,phi,A);
B = ffthphi.*K;
B = conj(infft(conj(B)));
B = Jac_u_trans(d,p,u1,u2, phi,phit,fftb,ffthphi, ffthphit, B);
gJ = A+B;
```

```
function gJ = gradient_J2(d,p,u1,u2,phi,phit,fftb,fftu,ffthphi,ffthphit)
Fhx = zeros(size(fftu));
Fhy = zeros(size(fftu));
Fhx(1,1)= 1;
Fhx(end,1) = -1;
Fhy(1,1)= 1;
Fhy(1,end) = -1;
Fhx = nfft(Fhx);
Fhy = nfft(Fhy);
```

```
gradxu = infft(Fhx.*fftu);
gradxu = infft(conj(Fhx).*nfft(conj(gradxu)));
gradxu = conj(gradxu);
gJ = Jac_u_trans(d,p,u1,u2,phi,phit,fftb,ffthphi,ffthphit,gradxu);

gradyu = infft(Fhy.*fftu);
gradyu = infft(conj(Fhy).*nfft(conj(gradyu)));
gradyu = conj(gradyu);
gJ = gJ + Jac_u_trans(d,p,u1,u2,phi,phit,fftb,ffthphi,ffthphit,gradyu);
```

```
function g = gradient_J(p,phi,phit,u1,u2,alpha,beta,gamma)
lambda = beta/alpha;
hphi = func_h(phi,p);
ffthphi = nfft(hphi);
hphit = func_h(phi+phit,p);
ffthphit = nfft(hphit);
bphi = func_b(hphi,hphit,u1,u2);
dphi = func_d(p,phi,phit,lambda);
uphi = func_u(dphi,bphi);
fftu = nfft(uphi);
fftb= nfft(bphi);

%% Calcul de gradient J2
g2 = gradient_J2(dphi,p,u1,u2,phi,phit,fftb,fftu,ffthphi,ffthphit);

%% Calcul de gradient J3
g3 = phi;

%% Cacul de gradient J1 de phi
g1phi = gradient_J1(p,phi,phit,ffthphi,ffthphit,dphi,fftb,fftu,u1,u1,u2);

%% Calcul de gradient J1 de phi+phit
hphi = func_h(phi+phit,p);
ffthphi = nfft(hphi);
hphit = func_h(phi+2*phit,p);
ffthphit = nfft(hphit);
bphi = func_b(hphi,hphit,u1,u2);
dphi = func_d(p,phi+phit,phit,lambda);
uphi = func_u(dphi,bphi);
fftu = nfft(uphi);

g1phit =
gradient_J1(p,phi+phit,phit,ffthphi,ffthphit,dphi,fftb,fftu,u2,u1,u2);

%% Calcul de Gradient_J

g =  alpha*(g1phi+g1phit)+(beta*g2)+(gamma*g3);%g1phit
g = real(g);
```

IV - Defloutage

```
%% ALgorithme de Descente de Gradient
close all;
```

```
clear all;
addpath('E:\Cours\4A\Projet_SPIM\Matlab\simulation\');
addpath('E:\Cours\4A\Projet_SPIM\Matlab\Opti\');
addpath('E:\Cours\4A\Projet_SPIM\Matlab\Zernick\');
load donnee;

%% Initialisation des paramètres
epsilon = 1e-200;
nrow = size(IMfloue,1);
ncol = size(IMfloue,2);
[x,y] = meshgrid(linspace(-1,1,nrow),linspace(-1,1,ncol));
p_calcul = func_p(x,y,diametre);
phit = 0.1;
delta = 100;
alpha = 1;
beta = 1e-11;
gamma= 1e-11;

%% Initialisation de la boucle
z = 5.0 *[1;1;1;1];%[0 0 0 0]';
Pz = func_Pz(length(z)-1,size(IM));
phiprec = zeros(size(IMfloue));

n = 0;
s = 1e10;
Jprec = 0;
Jsuiv =
func_J(p_calcul,reshape(Pz*z,size(IM,1),size(IM,2)),phit,IMfloue,IMfloueDec
,alpha,beta,gamma);

CF = [Jsuiv];%% Cost Function
while n<20
    gradphi =
gradient_J(p_calcul,reshape(Pz*z,size(IM,1),size(IM,2)),phit,IMfloue,IMflou
eDec,alpha,beta,gamma);
    g = Pz'*gradphi(:);
    zprec = z;
    zsuivtry = z - s*g;
    Jprec = Jsuiv;
    Jsuiv =
func_J(p_calcul,reshape(Pz*zsuivtry,size(IM,1),size(IM,2)),phit,IMfloue,IMf
loueDec,alpha,beta,gamma);
    %% regularisation du paramètre s
    while ((Jsuiv > Jprec))
        s = s/2;
        zsuivtry = z - s*g;
        Jsuiv =
func_J(p_calcul,reshape(Pz*zsuivtry,size(IM,1),size(IM,2)),phit,IMfloue,IMf
loueDec,alpha,beta,gamma);
    end;
    z = zsuivtry;
    delta = (Jsuiv - Jprec);
    disp(sprintf('delta:%1.5g; Jsuiv : %1.5g',delta,Jsuiv));
    CF = [CF;Jsuiv];
    if (abs(delta)<epsilon) n = n+1; else n =0; end;
end;
plot(CF);
phi = calcul_phi(z,size(IM));
disp(z)
```

```
%% plotage des données theorique
figure;
colormap(gray);
imagesc(IM);
figure;
colormap(gray);
imagesc(IMfloue);

%% Calcul de l'image défloutée
hphi = func_h(phi,p_calcul);
hphi = hphi/(sum(hphi(:)));

%ImageDefloue=func_u(d,b);
ImageDefloue=deconvolution(hphi,IMfloue);

%% plotage de l'image défloutée
figure;colormap(gray);
imagesc(ImageDefloue);

%% plotage de la phase calculée
figure; surf(phi);
```