ThermoFisher
SCIENTIFIC

Rapport de stage
3ème année d'école d'ingénieurs

Développement d'un algorithme de recalage non rigide d'images

Tuteur en entreprise : Sébastien MARTIN
Tuteur académique : Mathieu COLIN

Table des matières

Table des figures

1 Introduction

Thermo Fisher Scientific est une entreprise multinationale américaine implantée dans le développement de biotechnologies. L'entreprise résulte de la fusion en 2006 de Thermo Electron Corporation et de Fisher Scientific. Le groupe affiche plus de 17 milliards de dollars de chiffre d'affaires en 2016 et environ 65 000 employés. Les domaines d'études de l'entreprise sont assez vastes : sciences de la vie, sciences industrielles, sciences pharmaceutiques. Depuis peu, Thermo Fisher Scientific tend à proposer des solutions à ses clients comprenant des softwares en plus du matériel de laboratoire. En 2017, Thermo Fisher Scientific a ainsi racheté FEI (Field Electron and Ion Company), entreprise spécialisée dans les microscopes électroniques. Le groupe FEI avait quant à lui racheté VSG (Visualization Science Group) en 2016. Les locaux de VSG à Paris et Grenoble ont été fermés afin de centraliser l'activité à Bordeaux. De part ses multiples rachats, la moitié des personnes de mon équipe sont en télétravail, y compris mon tuteur de stage Sébastien Martin à Angers.

Le fonctionnement de l'équipe s'inscrit dans le cadre de développement agile de solutions. L'équipe a un product owner, autrement dit c'est lui qui réalise la feuille de route produit à partir de l'analyse des besoins clients. Mon tuteur Sébastien est le scrum master, il doit garantir que le processus agile se déroule correctement. L'équipe doit remplir un certain nombre de user stories (demande clients) sur une durée de trois semaines, appelée un sprint. Tous les jours, un daily meeting est réalisé sur Skype, c'est le moment où l'équipe se synchronise et où les différents membres peuvent relever un problème.

Mon équipe travaille sur une librairie multi plateforme de C++, .NET et Java, qui développe des algorithmes de traitement d'images.

Cette librairie contient un algorithme de recalage rigide d'images. Dans une optique d'amélioration continue de cette librairie, ce stage a pour objectif de développer un nouvel algorithme de recalage non rigide en langage C++.

2 Etat de l'art du recalage d'images

2.1 Définitions

Par recalage, on entend la mise en correspondance d'images afin de les comparer. On recale une image dite mobile sur une image dite de référence.

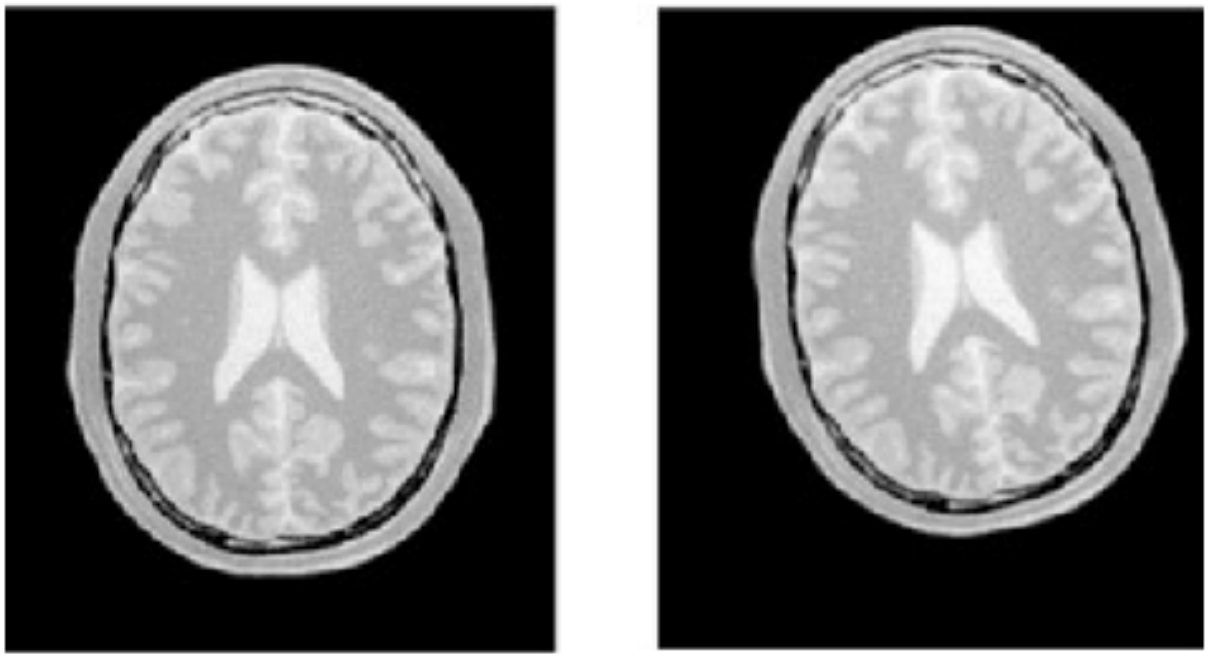

Figure1 : Image de référence (gauche) et Image mobile (droite) [B1]

En ce qui concerne la figure 1, en faisant subir une rotation à l'image mobile, il est possible de la recaler sur l'image de référence. Lorsque les transformations appliquées sont des translations, rotations ou homothéties, il est question de recalage affine. Mais lorsque les images sont différentes à une échelle plus fine, il est nécessaire que la transformation ait plus de degrés de liberté, il s'agit du recalage élastique. Il est alors recherché un champ de correspondance dense entre tous les pixels des images sur lequel des contraintes de régularité sont appliquées.
Les algorithmes de recalage sont donc caractérisés par la nature de la transformation appliquée, mais aussi par le type d'image recalée. Il existe le recalage mono-modal, c'est-à-dire avec deux clichés provenant d'une même source comme sur la figure 1 et le recalage multimodal, par exemple une image PET-scan sur une image CT-scan. Ce sujet de stage va porter seulement sur le recalage mono-modal.

2.2 Applications

Le recalage d'images trouve de nombreuses applications en imagerie médicale. Par exemple, il peut être fait appel au recalage pour s'abstraire des variations du corps humain: les battements du cœur et les gonflements du poumon qui influencent la forme des organes au moment de la prise d'un cliché, une tumeur qui se déforme considérablement pendant son évolution, les organes d'un patient qui se déforment suite à une radiothérapie ...

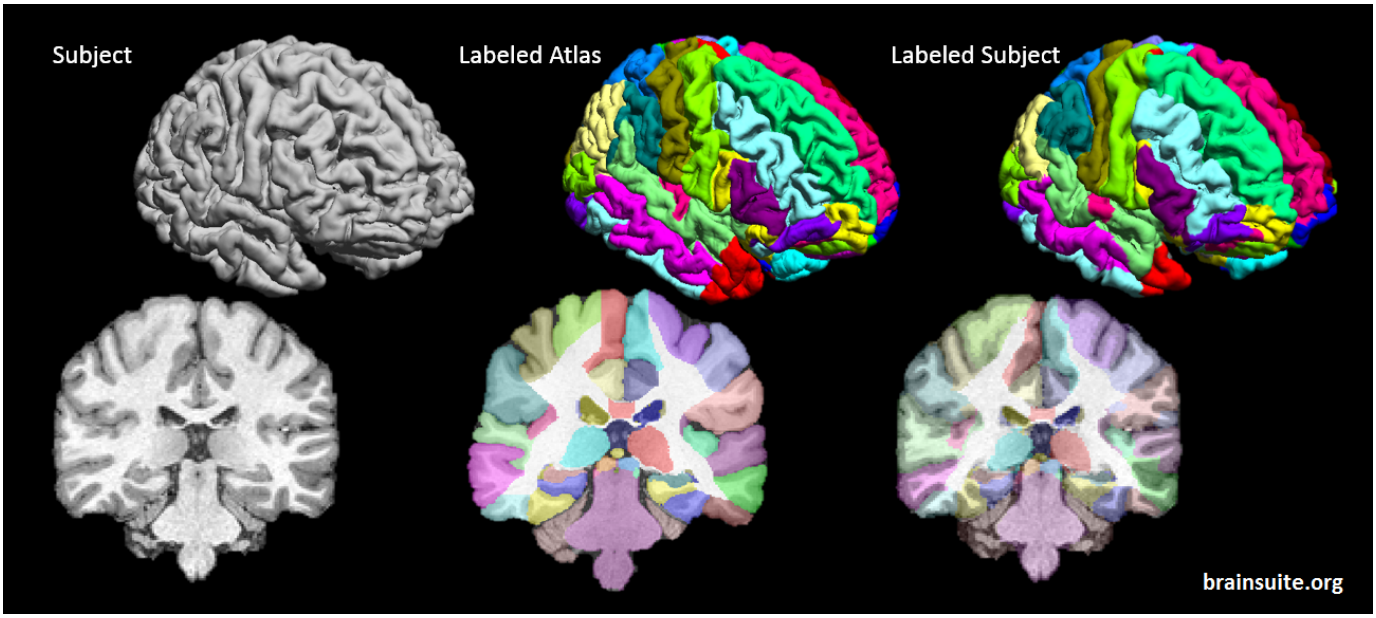

Figure2 : Segmentation par atlas [B2]

Une autre utilisation du recalage est la segmentation par atlas. La segmentation consiste au partitionnement de différentes régions de l'image sélectionnée sur un certain critère. Il existe une technique particulière de segmentation qui est la segmentation par atlas. L'atlas est une base spatiale de référence. Par exemple sur la figure 2, l'image de cerveau au milieu est prise comme référence. Une segmentation permet d'identifier différentes zones du cerveau. Le but du recalage dans cette situation est d'obtenir une segmentation du cerveau du sujet, image de gauche. Dans ce but, le cerveau de référence est recalé sur le cerveau du sujet. Le résultat est un champ de correspondance entre ces deux images. Ce champ est ensuite appliqué à la segmentation du cerveau de référence, ce qui engendre une segmentation du cerveau du sujet.

2.3 Contexte mathématique de l'étude [B3]

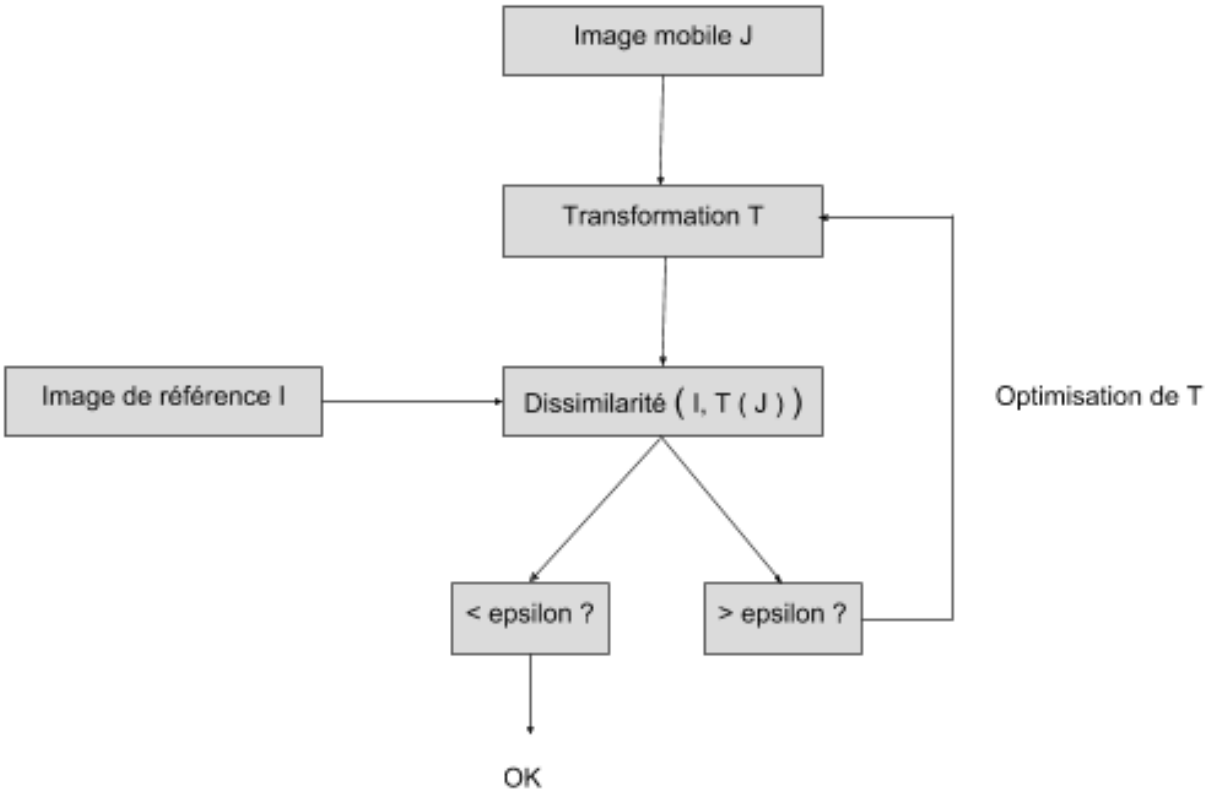

Figure3 : Principe général de l'algorithme

Soient deux images, l'une fixe que l'on appellera I et l'autre mobile que l'on appellera J. On appelle aussi I et J les fonctions qui relient un point de l'espace à son intensité respectivement dans l'image I et dans l'image J.
Soit $\Omega \subset \mathbb{R}^d$ l'espace des images avec d la dimension de l'image égale à 2 ou à 3.
Soit $u : \Omega \to \Omega$ un champ de déplacement.
La transformation T recherchée est de la forme :

$$T : \Omega \to \Omega$$
$$x \mapsto x + u(x)$$

Soit D une certaine distance.

La figure 3 met en en avant que la difficulté de l'algorithme est l'évaluation de la transformation T.

Quelle transformation T sera optimale au sens où I et J seront le plus similaires possible. Le problème peut alors se formuler de la manière suivante :

on cherche T qui minimise $D(I\ ,\ J(T))$ **.**

Malheureusement, une minimisation directe de la distance D (aussi appelée mesure de dissimilarité dans la littérature) a certains inconvénients. En effet, la résolution de ce problème va conduire à une solution qui va associer à chaque point de l'image I un déplacement pour l'apparier sur son homologue de l'image J. Cependant, le champ de déplacement obtenu ne sera généralement pas régulier. Des chevauchements de certains points peuvent apparaître ce qui a pour effet que la transformation ne soit plus inversible.

Dans la pratique, le problème que l'on cherche à résoudre est en fait le suivant :

on cherche T qui minimise $D(I\ ,\ J(T)) + \alpha\ S(u)$

avec $\alpha > 0$ et S une application qui va lisser le champ de déplacement pour le régulariser.

3 Algorithmes implémentés et premiers résultats

Il a été implémenté pendant ce stage différents algorithmes qui vont être présentés dans cette section.
Les détails techniques de l'implémentation seront traités en annexe.

3.1 Algorithme de ré-échantillonnage [A2]

Comme son nom l'indique cette classe permet de ré-échantillonner une image aux dimensions souhaitées. C'est-à-dire que l'utilisateur de la classe peut redéfinir les caractéristiques de l'image : son origine, sa taille, la dimension de ses pixels.

Cet algorithme est essentiel car il permet aussi d'appliquer un champ de déplacement à une image puis de la ré-échantillonner. Il sera utilisé notamment pour appliquer le champ de forces calculé à chaque itération à l'image mobile. Il sera aussi utilisé dans le cadre de l'approche pyramidale pour sous-échantillonner les images mais nous y reviendrons plus tard.

Pour tester exactement le fonctionnement de l'algorithme de ré-échantillonnage, j'ai créé une image 4x4 sous la forme d'un damier noir et blanc et une image de déplacement. Ensuite, j'ai fait les calculs à la main de la nouvelle valeur d'intensité de chaque pixel et les ai comparées avec les résultats de l'algorithme. J'ai ensuite fait une série de tests plus visuels pour m'assurer du comportement de l'algorithme sur des images un peu plus complexes. La figure 4 est un damier de taille 500 x 500 dont les pixels sont égaux à 0 ou à 100. Il faut bien avoir l'esprit en regardant ce damier qu'un carreau n'est pas un pixel mais en réalité 125 pixels. L'image de déplacements de la figure 5 présentée a été obtenue avec l'algorithme d'interpolation par spline. Nous reviendrons plus tard sur cet algorithme.

Un principe important dans le développement de la librairie est l'intégration continue. L'équipe a investi dans de nombreux tests automatiques qui permettent de s'assurer au jour le jour du bon fonctionnement des algorithmes implémentés. Le principe fondamental est qu'un nouveau développement ne peut être intégré dans le produit que si tous les tests automatiques se déroulent sans erreur sur toutes les plateformes. Les résultats de la figure 6 sont ainsi sauvegardés et serviront de BaseLine (image de référence) dans l'intégration continue.

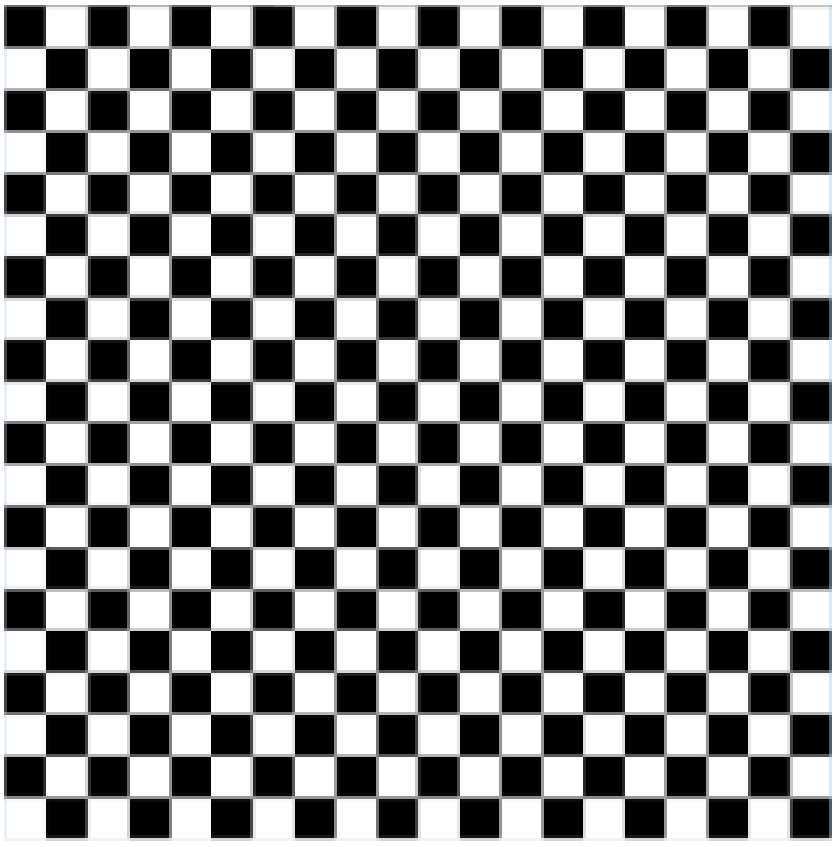

Figure4 : Damier noir et blanc

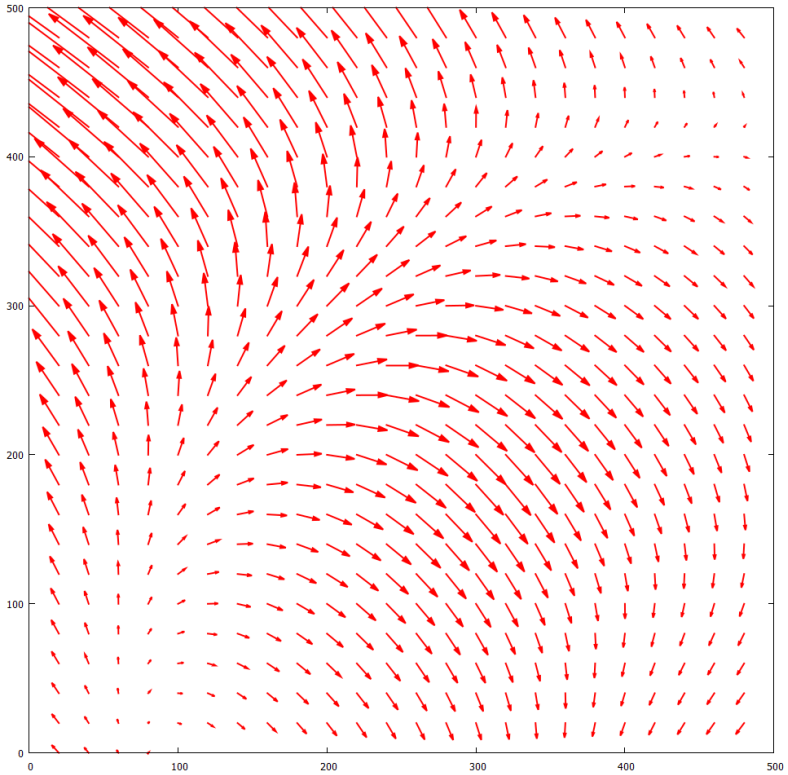

Figure5 : Champ de déplacement

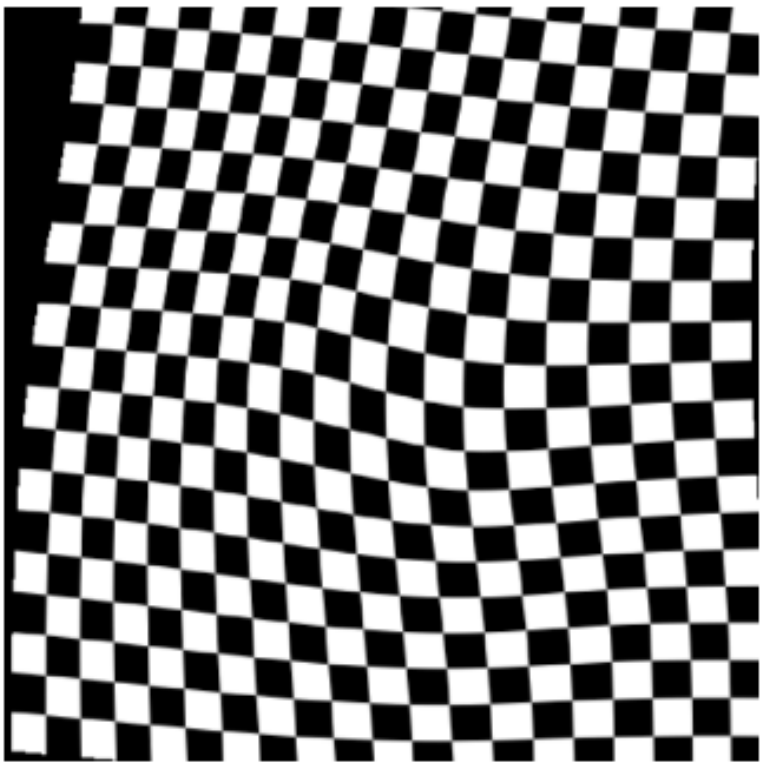

Figure6 : Damier déplacé avec l'algorithme de ré-échantillonnage

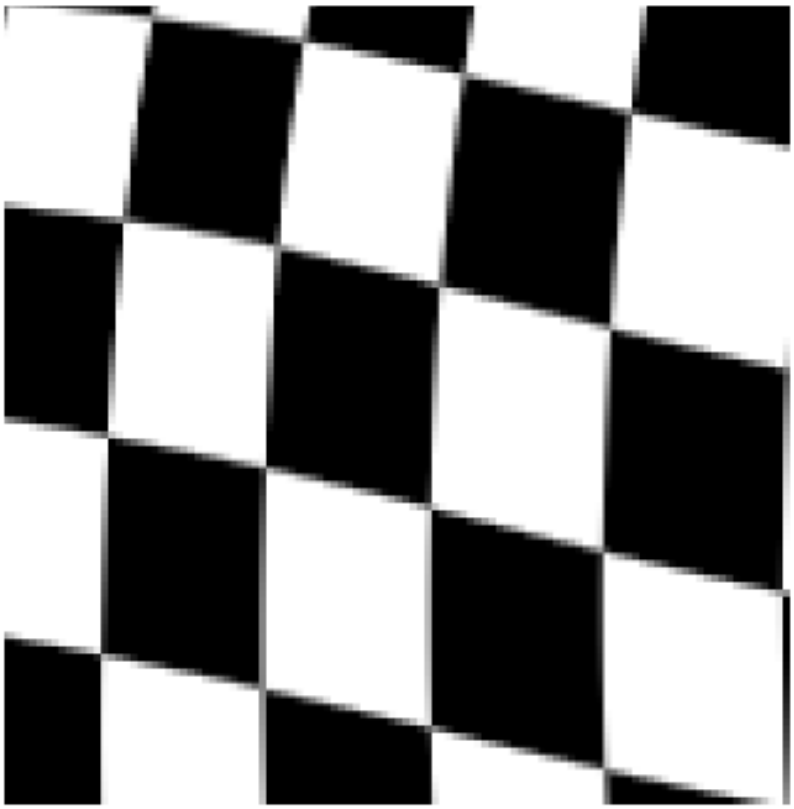

Figure7 : Zoom sur le damier déplacé

Après avoir appliqué le champ déplacement de la figure 5 au damier de la figure 4, il est obtenu le damier de la figure 6. Il peut être remarqué en haut à gauche de cette figure une zone noire. Lorsqu'un pixel est déplacé hors de la grille de l'image, sa valeur est extrapolée à une valeur fixée arbitrairement par l'utilisateur, ici zéro. Par contre lorsqu'un pixel est déplacé dans la grille de l'image sa valeur est interpolée . Ceci est mis en avant sur la figure 7, la frontière entre deux carreaux du damier est une échelle de gris .

3.2 Algorithme de recalage élastique

L'algorithme qu'il a été choisi d'être implémenté dans le cadre du stage est l'algorithme des Démons, qui reste une référence dans la littérature concernant le recalage d'images. Il consiste à calculer des « forces » (qui sont en réalité des petits déplacements) à appliquer sur l'image mobile afin de diminuer la distance D avec l'image de référence. Il s'agit en réalité d'un algorithme de descente de gradient adapté au cas du recalage. Le livre *Numerical Methods for Image Registration [B3]* explique très bien le principe de l'algorithme.

Dans le cas de la mesure SSD (Sum Squared Difference (of intensities)), l'expression des ces forces en un pixel x est $(I(x) - JoT(x))\nabla JoT(x)$. Le détail de la démonstration afin d'arriver à ce résultat est détaillé en annexe [A1]. Regardons comment fonctionne cette relation d'un point de vue physique. L'expression de ces forces contient deux termes importants : $\nabla JoT(x)$ est la direction de la force, c'est-à-dire que la force aura la même direction que la plus grande variation d'intensité dans l'image mobile, autrement dit si l'on se situe sur un point à la frontière entre deux zones colorées distinctes, la force sera orthogonale à cette frontière. Par exemple, la figure 8 illustre deux disques de tailles différentes que l'on souhaiterait recaler. Les forces qui vont s'appliquer sur les contours du disque mobile vont donc être orthogonales à son périmètre, comme représenté sur la figure 9.

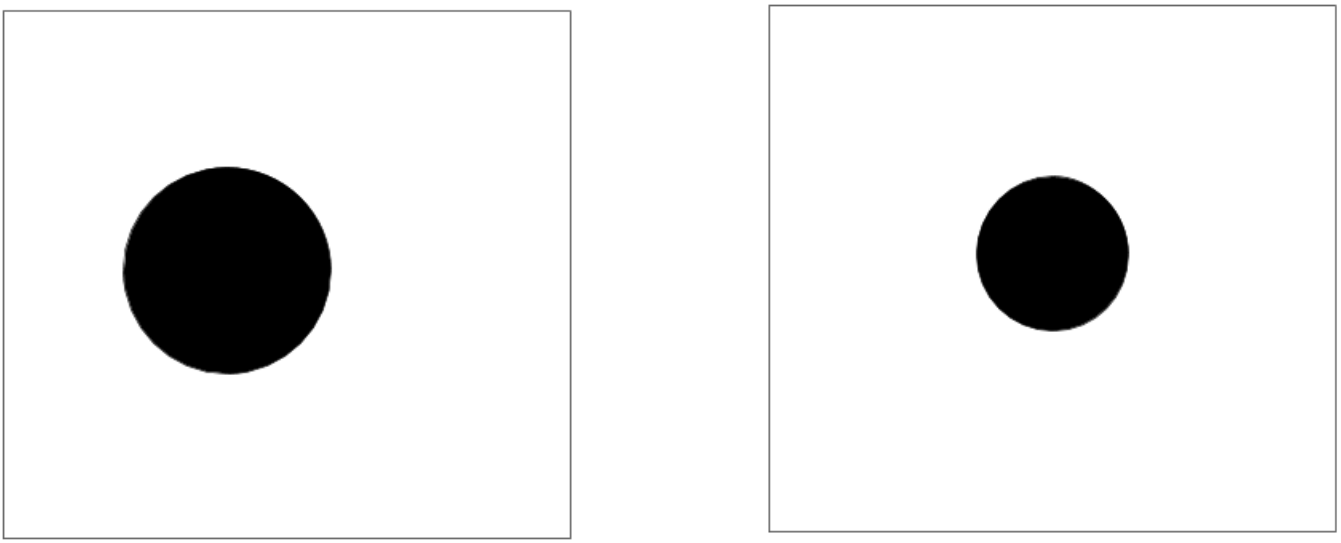

Figure8 : Image de référence (gauche) et image mobile (droite)

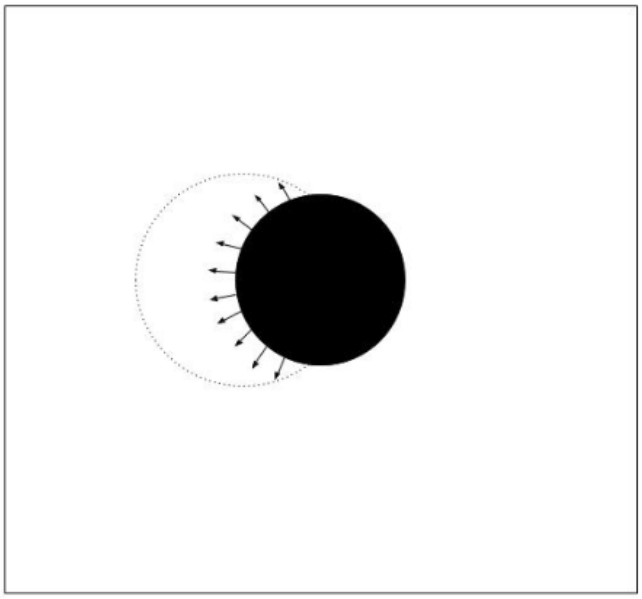

Figure9 : Représentation des forces obtenues avec l'algorithme des démons.

Ensuite, le terme $\nabla JoT(x)$ est pondéré par $(I(x) - JoT(x))$ qui rend compte de la différence d'intensité entre les deux images. Ainsi, plus cette différence est élevée, plus la norme de la force appliquée en un point sera importante.

Une fois que l'on a calculé les forces à appliquer sur l'image mobile, il est appliqué un filtre de lissage au champ de force. On applique ensuite ce champ de forces à l'image mobile. On obtient ainsi une nouvelle image sensée être plus « similaire » à l'image de référence. Puis l'opération est réitérée jusqu'à satisfaire un certain critère de similarité.
Tout en respectant une approche pyramidale, cette classe va enchaîner les étapes suivantes :

1) Définition de l'espace de travail image : consiste à sélectionner l'intersection de l'image mobile et de l'image de référence, ou à utiliser une zone de travail définie par l'utilisateur
2) Ré-échantillonnage des deux images aux dimensions de l'espace de travail
TANT QUE (dissimilarité < ε)
- 3) Calcul des forces et de la similarité
- 4) Régularisation du champ de forces
- 5) Calcul du max des forces
- 6) « Scaling » des forces : on divise le champ de forces par $2 f_{max}$
- 7) Addition du champ de forces au champ de déplacement (initialement nul)
- 8) Régularisation du champ de déplacements
- 9) Ré-échantillonnage et déplacement de l'image mobile

Dans la littérature il est souvent question de forces mais en réalité il s'agit de petits déplacements. A chaque itération, ces petits déplacements sont accumulés dans une image de déplacement, qui sera la sortie de l'algorithme. Il s'agit du champ de déplacement à appliquer à l'image mobile pour la recaler sur l'image de référence.

J'ai testé l'algorithme sur deux images de disque de taille 100 x 100 qui sont présentées sur la figure 8. Le but est de recaler le plus petit des deux disques sur le plus grand.

Le déplacement final qui est la sortie de l'algorithme est présenté sur la figure 10.
Le damier de taille 100 x 100 est utilisé sur la figure 11 pour mieux visualiser l'effet de ce champ de déplacement.

Un plan séquence de l'image mobile qui se déforme sera présenté lors de la soutenance.

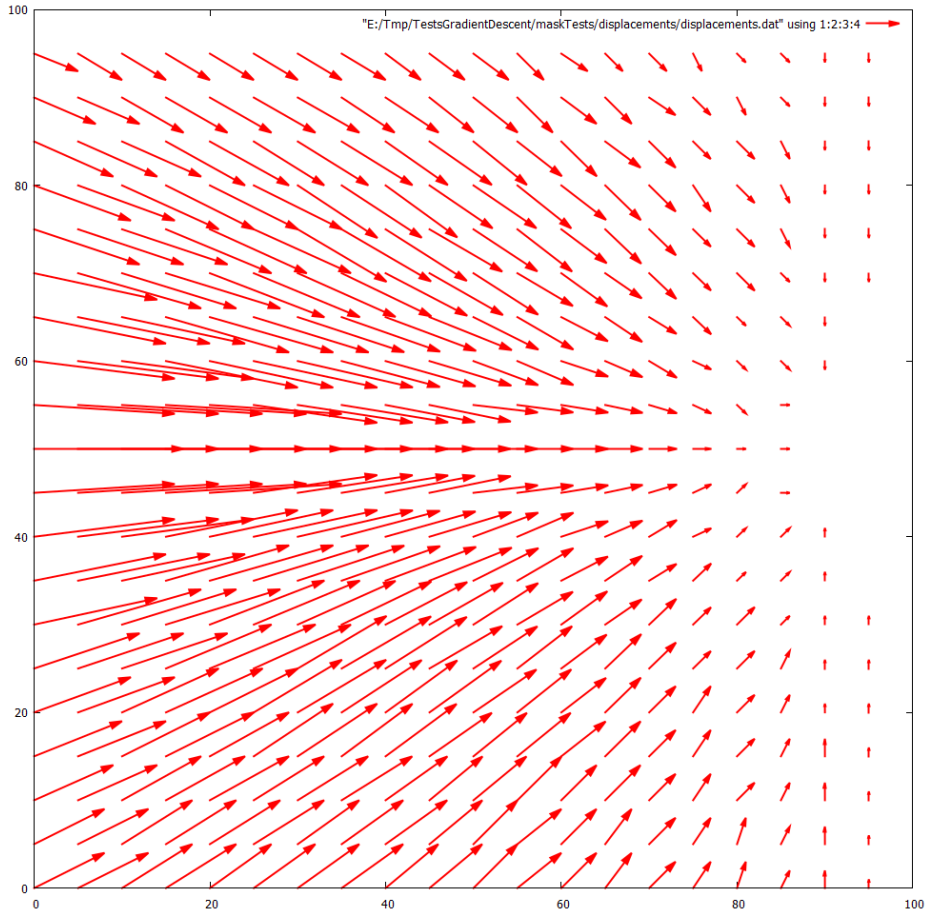

Figure10 : Champ de déplacement en sortie de l'algorithme

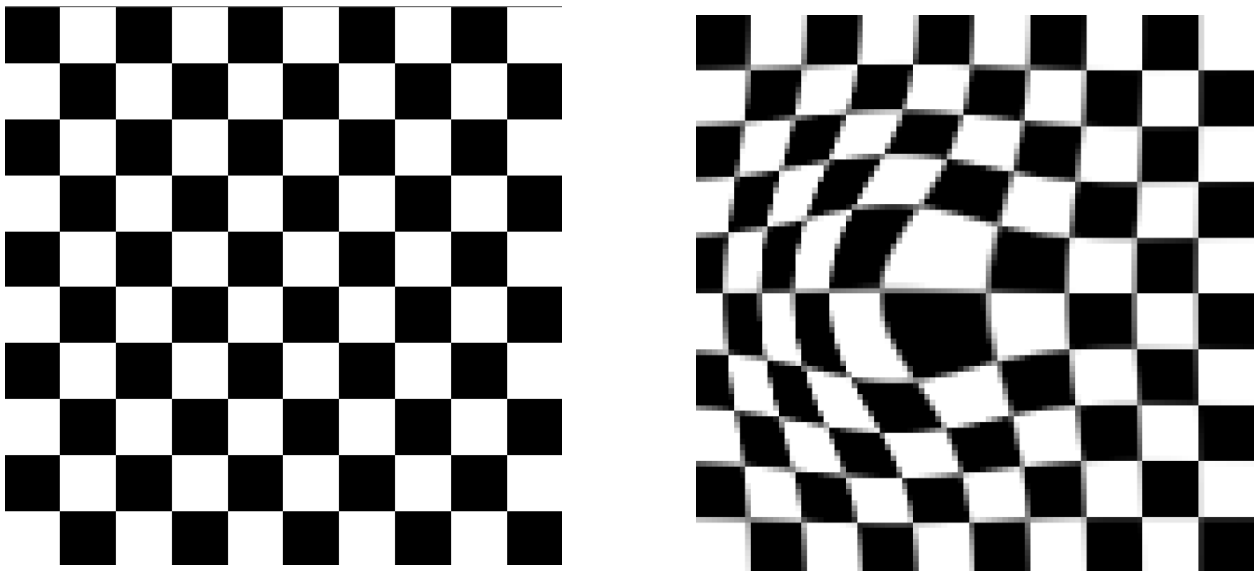

Figure11 : Utilisation d'un damier pour visualiser le champ de déplacement

3.3 Approche pyramidale

Le but de cet algorithme est d'accélérer les calculs. Il apporte aussi une certaine stabilité de la solution dans la recherche de la transformation optimale car le recalage est effectuée sur des images plus régulières contenant moins de détails. Cela permet d'éviter de tomber dans des minima locaux.

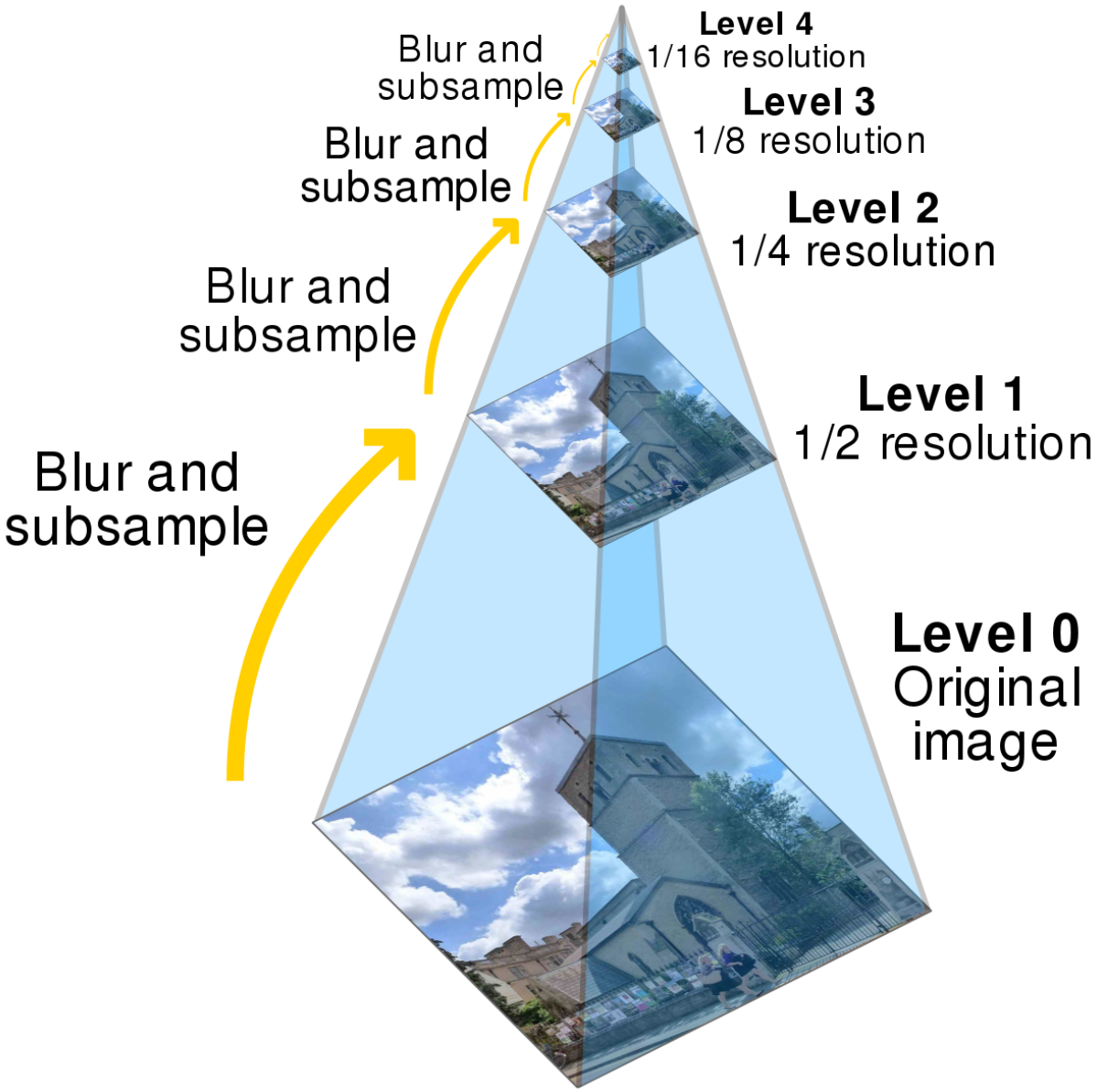

Figure12: Exemple de pyramide à 4 étages [B5]

Pour passer d'un étage à l'étage supérieur de la pyramide de la figure 12, afin de respecter le critère de Niquist, l'image est d'abord moyennée avec un filtre moyenneur, par exemple chaque pixel se voit attribuer une nouvelle valeur qui est la moyenne avec ses huit autres voisins. Une fois moyennée, l'image est sous-échantillonnée avec l'algorithme de ré-échantillonnage. Concrètement la taille des pixels est multipliée par deux et leur nombre est divisé par deux. La nouvelle image obtenue conserve ainsi le principal de l'information de l'image précédente tout en ayant éliminé les détails.
Une approche pyramidale peut être réalisée dans la situation du recalage non rigide d'images car les algorithmes utilisés sont itératifs. L'image de déplacements n'est pas calculée directement, c'est une accumulation de champ de forces. A chaque étape l'image de déplacements obtenue est un peu plus proche de l'image de déplacements exacte.

Continuons avec l'exemple d'une pyramide à quatre niveaux de la figure 12. Dans le cas du recalage, deux pyramides sont créées : une pyramide qui a comme image originale l'image de référence, et l'autre l'image mobile. L'algorithme démarre au niveau 4 et l'image de déplacements ne contient que des zéros. L'algorithme des démons permet le calcul du champ de forces. Ce champ de force est intégré à l'image de déplacements. Puis un nouveau champ de forces est calculé jusqu'à ce que le critère de dissimilarité entre les deux images soit satisfait. A la fin de cette première étape, il a été obtenu une image de déplacements calibrée aux dimensions du niveau 4. Pour cette première étape les calculs vont être rapides car l'algorithme est appliqué à deux images contenant chacune seize fois moins de pixels que les images originales. Maintenant cette image de déplacements va être sur-échantillonnée pour qu'elle soit calibrée aux dimensions du niveau 3. L'algorithme ne redémarre pas d'une image de déplacement nulle mais d'une image de déplacements qui est déjà une première approximation de l'image finale. L'algorithme des démons est ensuite ré-appliqué. Et ainsi de suite jusqu'à remonter au niveau zéro.

La pyramide à deux étages du disque mobile de la figure 8 est représentée sur la figure 13. L'image du niveau 2 est 4 fois moins résolue que l'image originale. Le disque est moins net, moins détaillé, néanmoins l'information principale est toujours présente, on reconnaît bel et bien qu'il s'agit d'un disque.

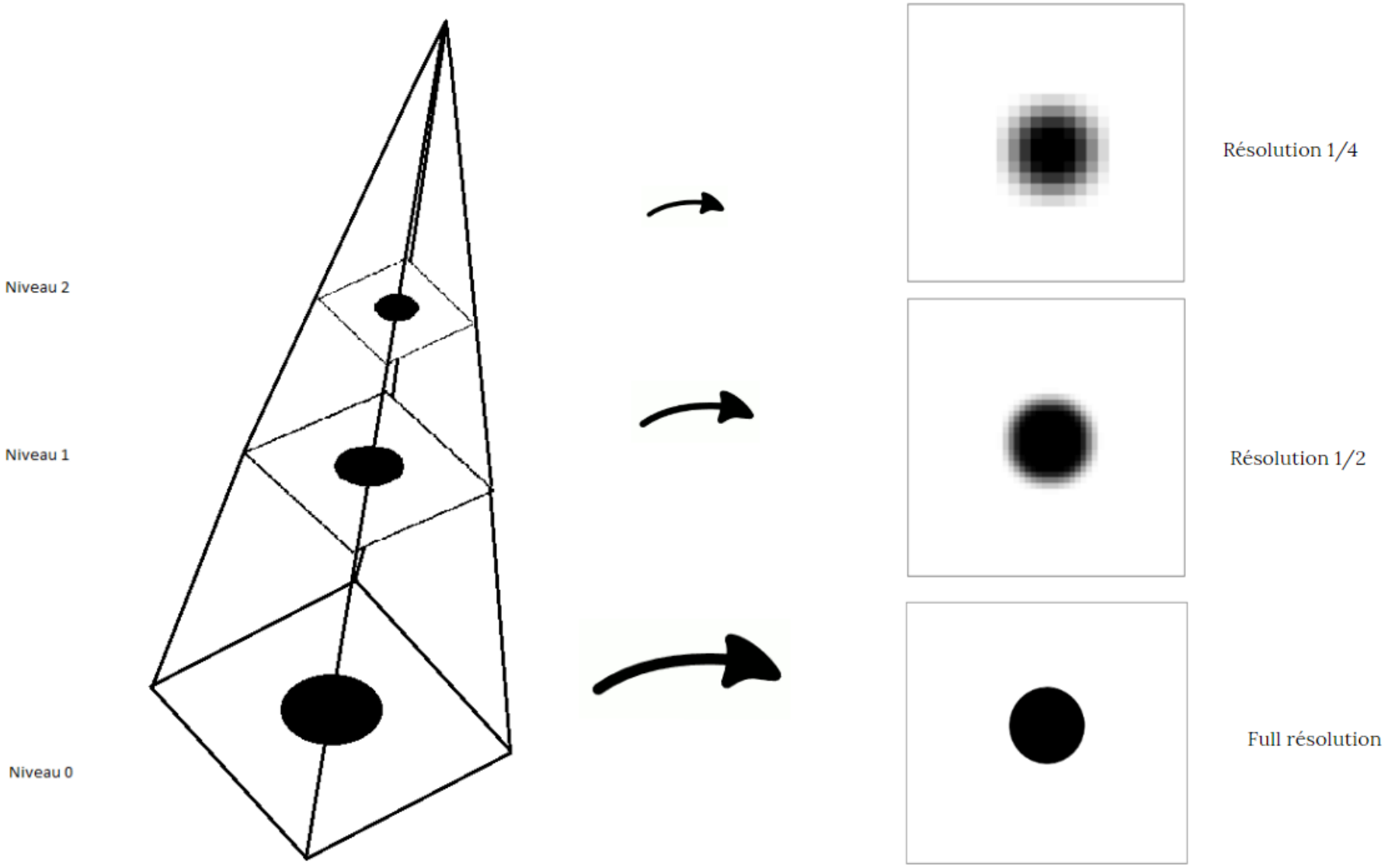

Figure13: Représentation de la pyramide du disque de la figure 8.

3.4 Algorithme d'interpolation par spline (Thin Plate Spline)

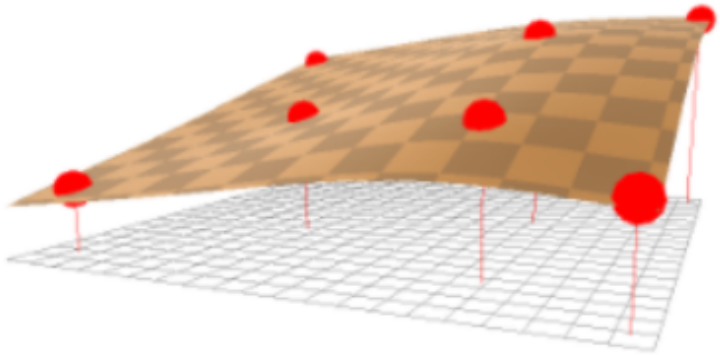

Figure14 : Recherche d'une surface avec « l'énergie de courbure » la plus faible [B 6]

Le but de cet algorithme est de générer facilement une image de déplacements dense à partir de quelques points éparses où ont été définis un déplacement. Il s'agit d'un algorithme d'interpolation. L'utilisateur rentre quelques points où il va définir un déplacement en x et en y et ensuite l'algorithme va interpoler dans tout l'espace image des déplacements. Le but étant de trouver la surface passant par tous les points définis par l'utilisateur qui "plie", qui courbe le moins comme le met en avant la figure 14. La figure 16 illustre un exemple de génération de champ de déplacement dense à partir du champ de déplacements constitué seulement de 4 points sur la figure 15.

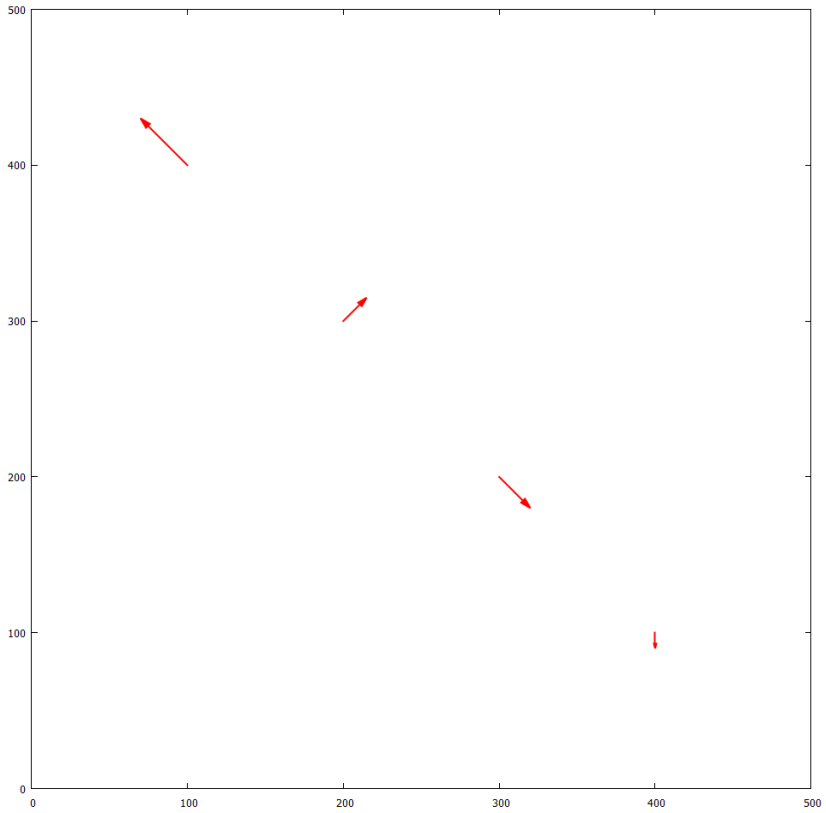

Figure15 : Champ éparse de déplacements

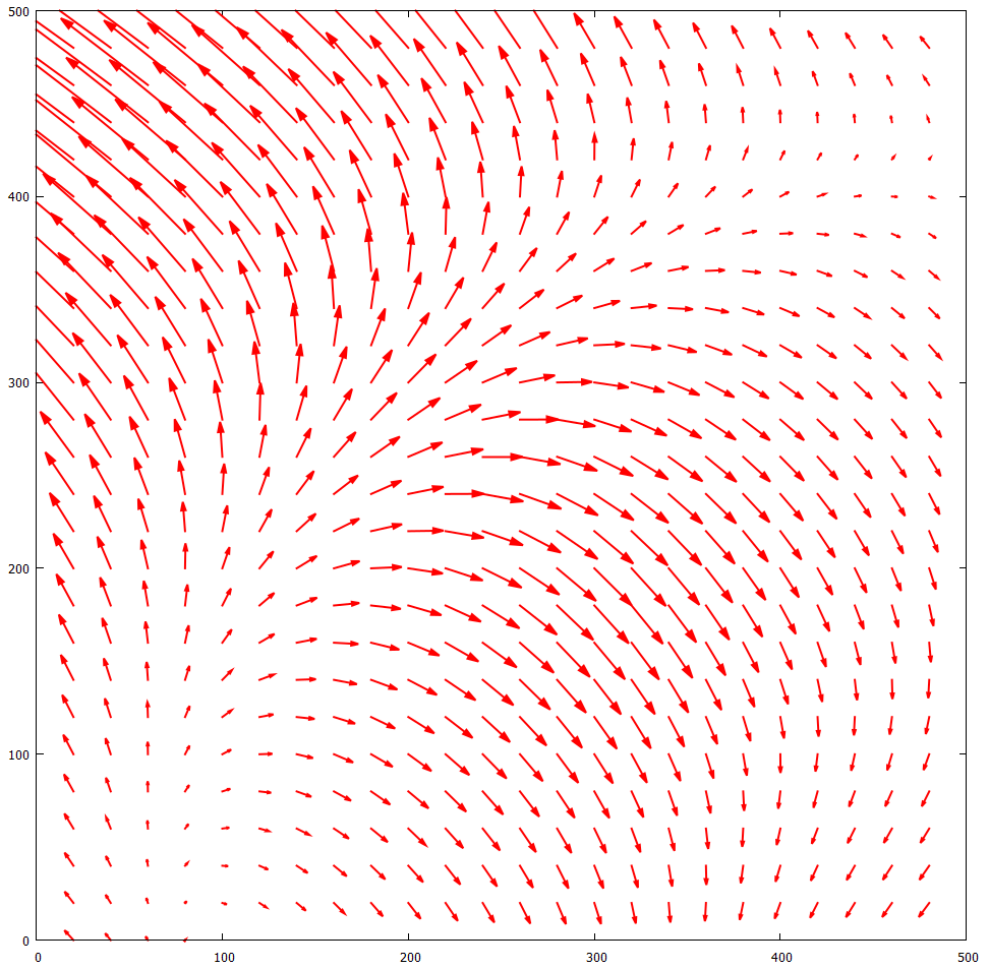

Figure16 : Champ de déplacement dense obtenu par interpolation

Cet algorithme peut être utilisé pour effectuer un recalage « manuel ». L'utilisateur définit un certain nombre de points qui sont les mêmes sur les deux images puis il va manuellement définir les déplacements à appliquer sur les points de l'images mobile pour venir les faire correspondre sur les points de l'image de référence.

Cet algorithme est aussi notamment utile pour écrire un test de l'algorithme de recalage élastique. Une image de déplacement est générée à partir de quelques points au préalable définis par l'utilisateur. L'image de test est déplacée puis ré-échantillonnée avec l'image de déplacement générée précédemment. Ensuite on applique l'algorithme de recalage élastique entre l'image de test et l'image déplacée et il doit être vérifié que le champ de déplacement obtenu en sortie est proche celui généré par interpolation splinique.

4 Comparaisons des temps de calculs

Il va être étudié dans cette section les performances de l'algorithme implémenté.

Les comparaisons des temps de calculs vont être effectuées sur les deux disques de la figure 8. Il s'agit d'images flottantes 32 bits et de taille 100 x 100.

L'arrêt de l'algorithme est demandé lorsque la variation relative de la métrique est inférieure à 1 %.

La métrique choisie est la SSD : Sum Of Squared Differences.

	Temps de calcul (sec)	Nombre d'itérations
Algorithme de base	0,53435	81
Parallélisation	0,467681	81
Approche multi-résolution (1 étage)	0.338136	91
Approche multi-résolution (2 étages)	0.308329	104

Figure17: Comparaison des temps de calculs et du nombre d'itérations des différents algorithmes

Le tableau de la figure 17 met en avant que la parallélisation et l'approche multi-résolution permettent de diviser le temps de calculs quasiment par deux sur les images testées.

Il peut être fait un petit commentaire concernant l'augmentation du nombre d'itérations avec l'approche multi-résolution. Cela semble logique dans la mesure où le recalage doit mettre à peu près le même nombre d'itérations sur les images originales que sur les images les plus sous-résolues, ce qui varie est seulement le temps de calcul car il y a moins de pixels. Ainsi, le fait de terminer le recalage dans les niveaux plus résolus en repartant du champ de déplacement obtenu auparavant ne coûte que quelques itérations. Le nombre d'itérations en approche multi-résolution est donc sensiblement le même mais légèrement supérieur que sans approche multi-résolution.

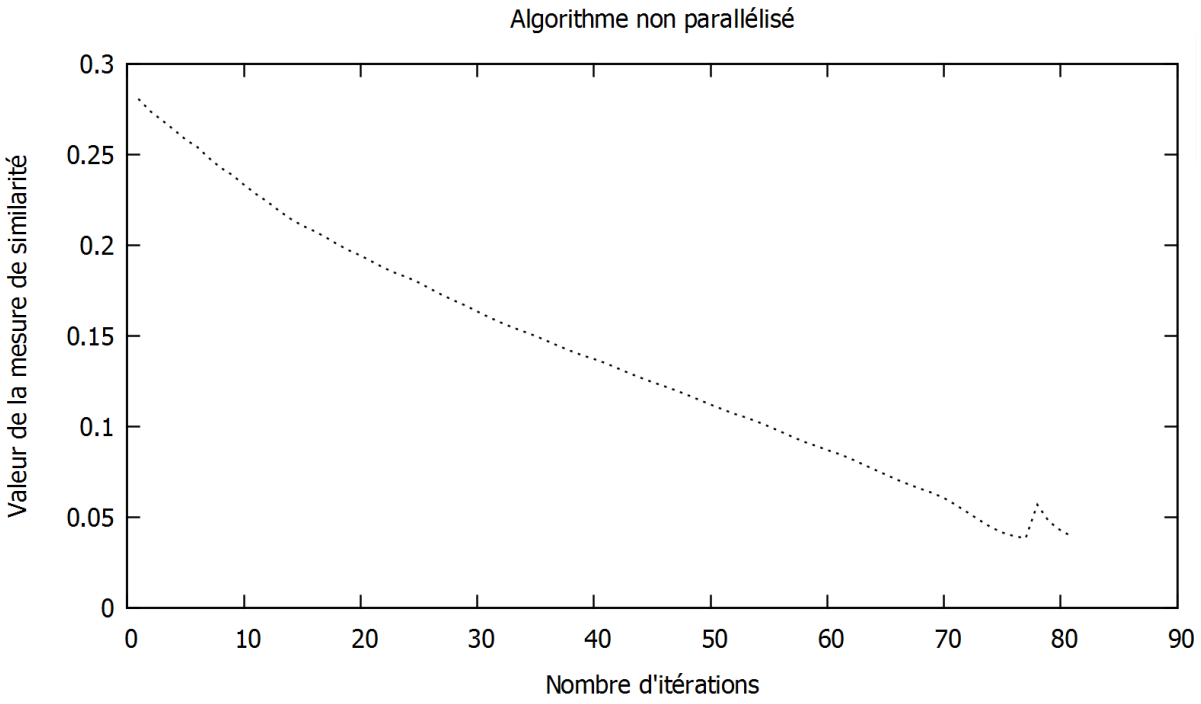

Figure18 : Algorithme sans amélioration

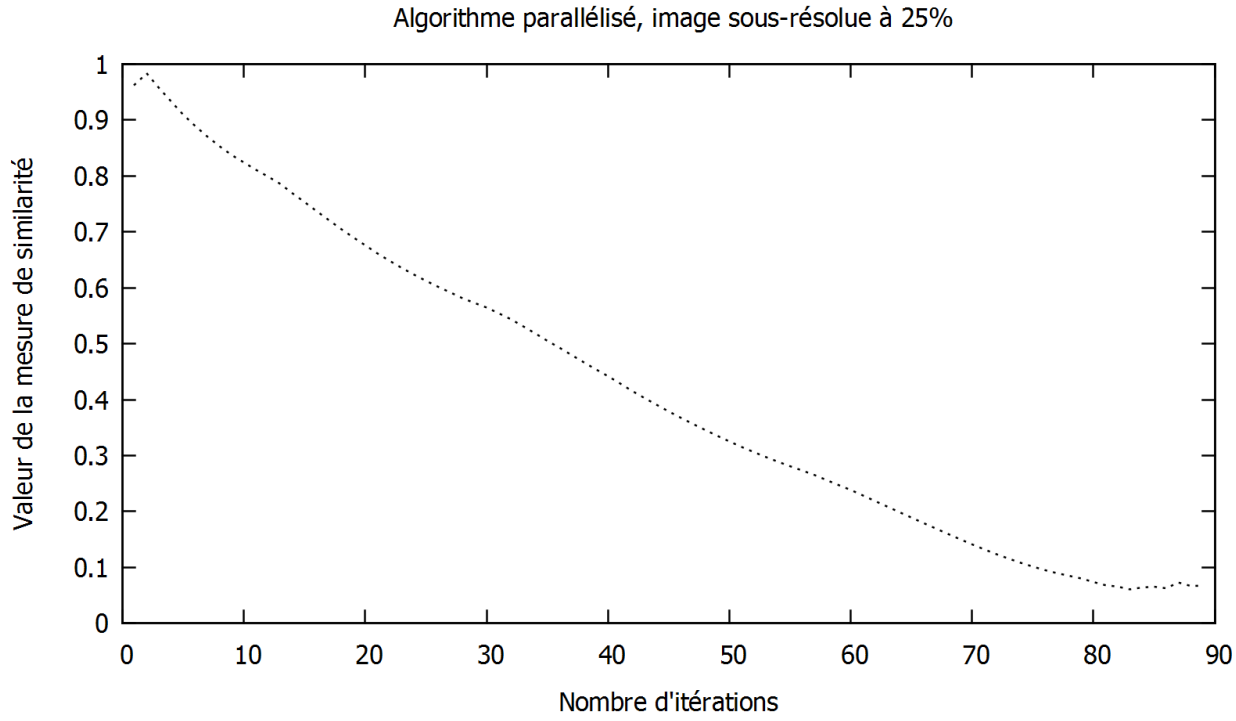

Figure19 : Approche multi-résolution parallélisée, deuxième étage de la pyramide

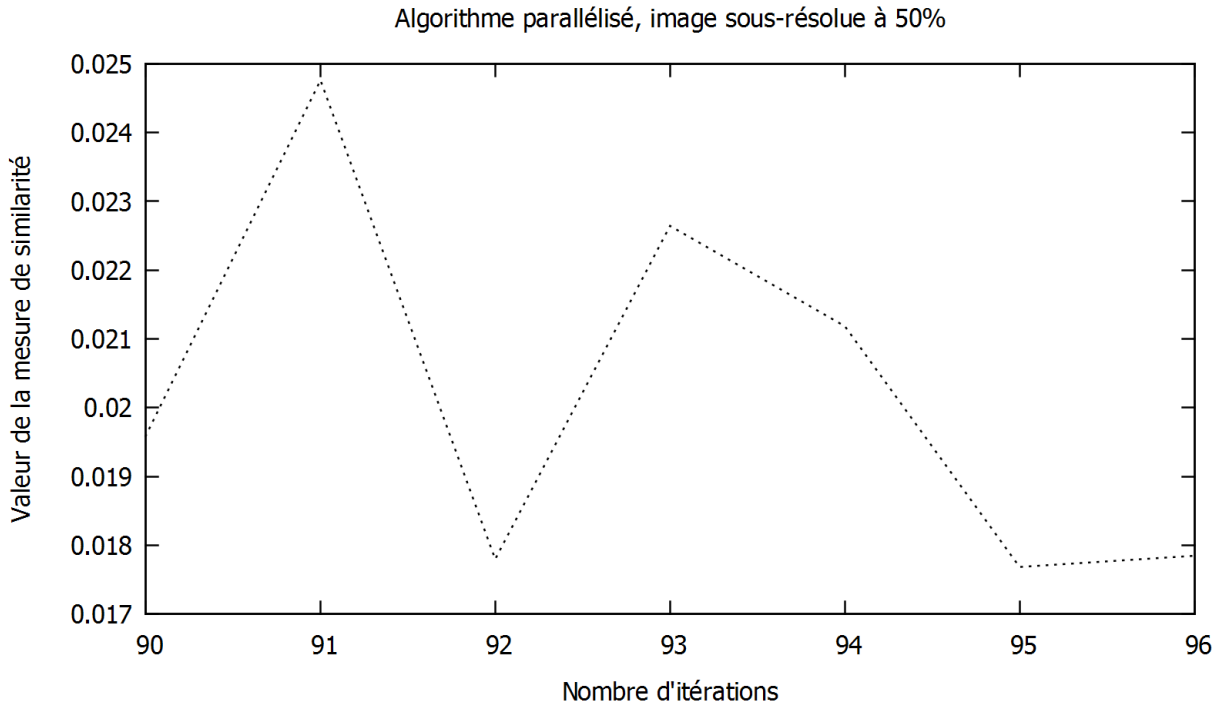

Figure20: Approche multi-résolution parallélisée, premier étage de la pyramide

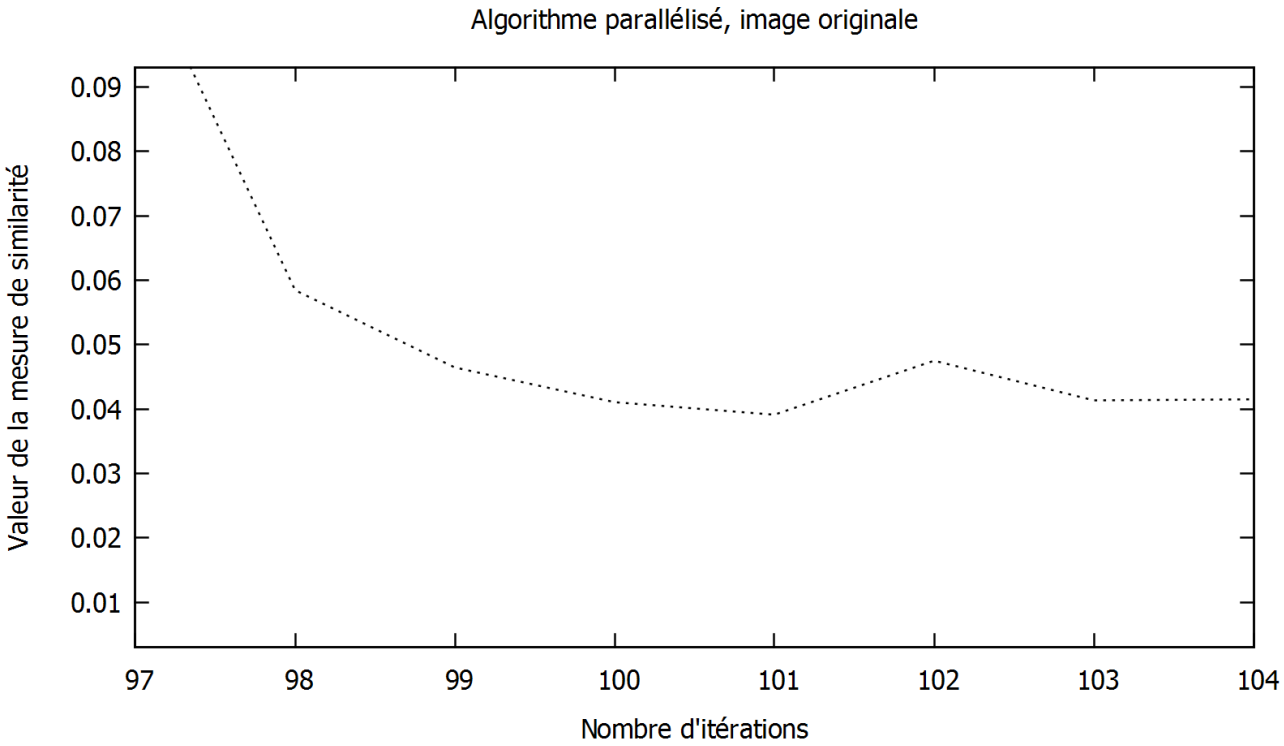

Figure21: Approche multi-résolution parallélisée, full résolution

La figure 18 témoigne de la convergence de l'algorithme sans améliorations en 81 itérations. Sur les dernières itérations la mesure de dissimilarité croît légèrement ce qui entraîne l'arrêt de l'algorithme.

L'évolution de la mesure de dissimilarité dans le cas d'une approche multi-résolution à trois niveaux est présentée sur les figures 19, 20 et 21. La plus grande partie du recalage est effectué à l'étage le plus sous-résolu présenté en figure 19. Lorsque l'algorithme remonte ensuite dans des niveaux de résolution plus élevés, le recalage s'effectue sur des différences moins marquées entre les deux images, il est moins important. Ainsi, la figure 20 illustre une mesure de dissimilarité qui oscille autour d'une valeur moyenne et la figure 21 met en avant une légère décroissance de la dissimilarité sur quelques itérations.

5 Validation de l'algorithme

Le but de cette partie est de réaliser une validation visuelle de l'algorithme de recalage. L'idée est d'utiliser l'algorithme de Thin Plate Spline afin de générer un champ de déplacement dense. A l'aide de l'algorithme de ré-échantillonnage, ce champ dense de déplacement est ensuite appliqué sur une coupe transversale de cerveau présentée sur la figure 22 [B7]. Cela produit une seconde image de cerveau, mise en avant sur la figure 23. Le but est ensuite de recaler ces deux images est de comparer le champ de déplacement résultant avec le champ de déplacement généré avec les Thin Plate Spline.

Figure22: Coupe transversale de référence [B6]

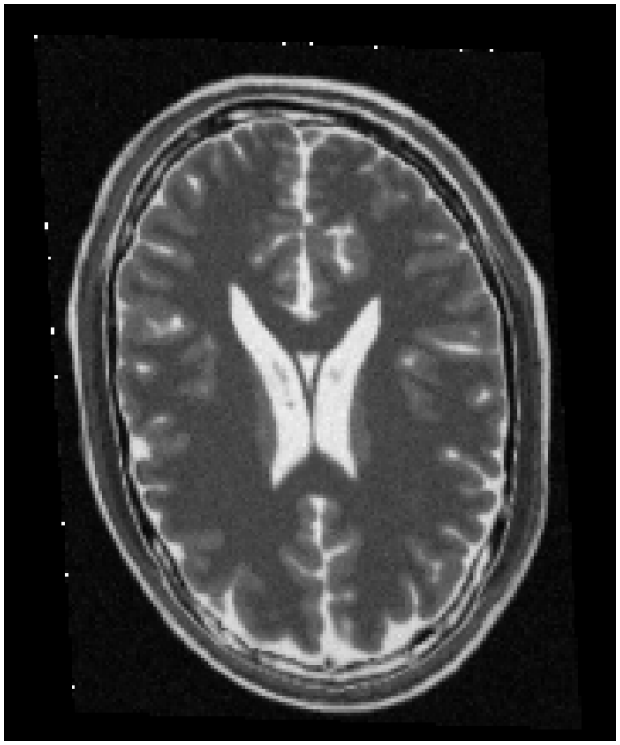

Figure23: Image de référence (fig 21) transformée par Thin Plate Spline

Figure24: Image de référence (fig21) recalée sur la transformée (fig 22)

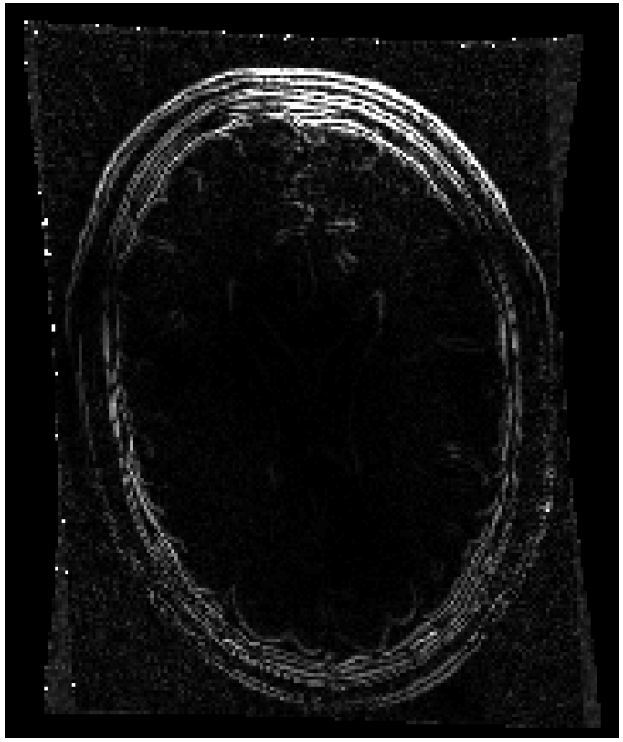

Figure25: Image de différences entre la fig 22 et la fig 23

Le recalage de la figure 23 sur la figure 22 est représenté sur la figure 24. Il est difficile de voir le résultat à l'œil nu, la figure 25 permet de voir les zones de la figure 24 où le recalage est moins précis, ces zones sont en blanc.

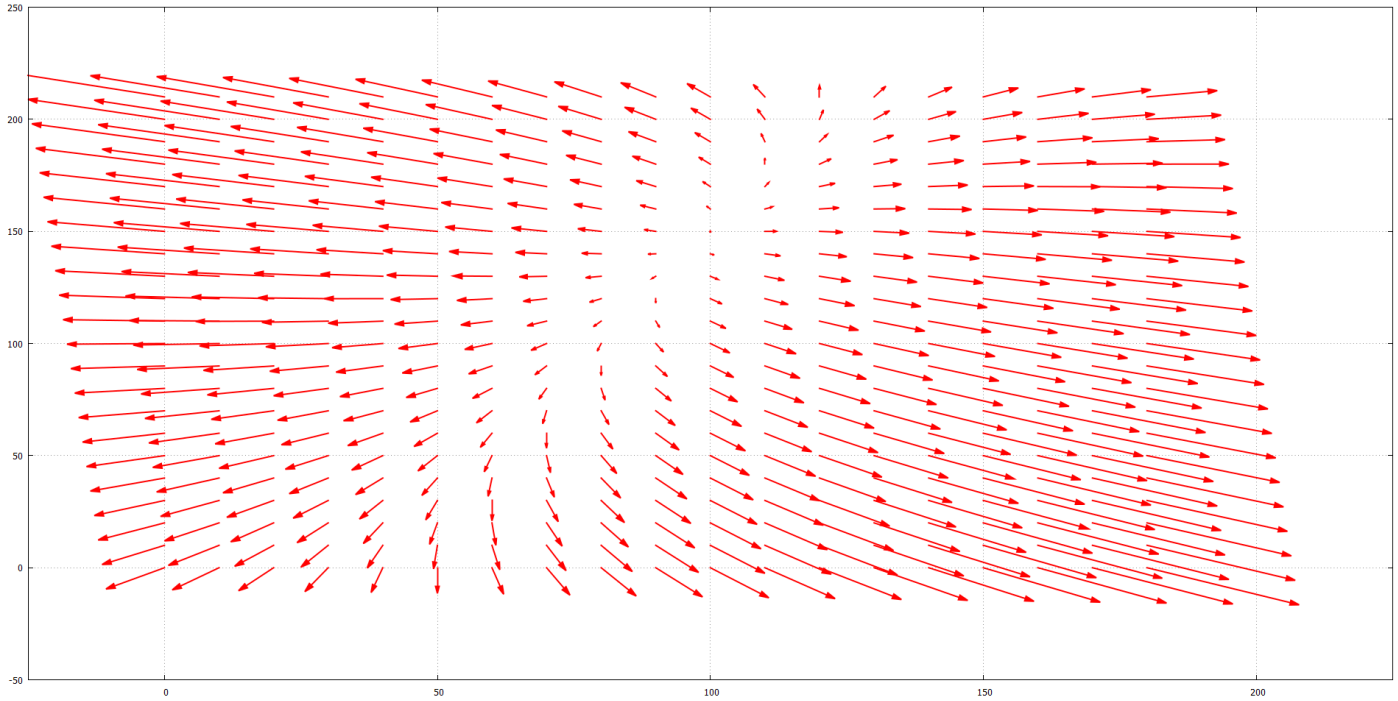

Figure26 : Champ de déplacement généré par Thin Plate Spline

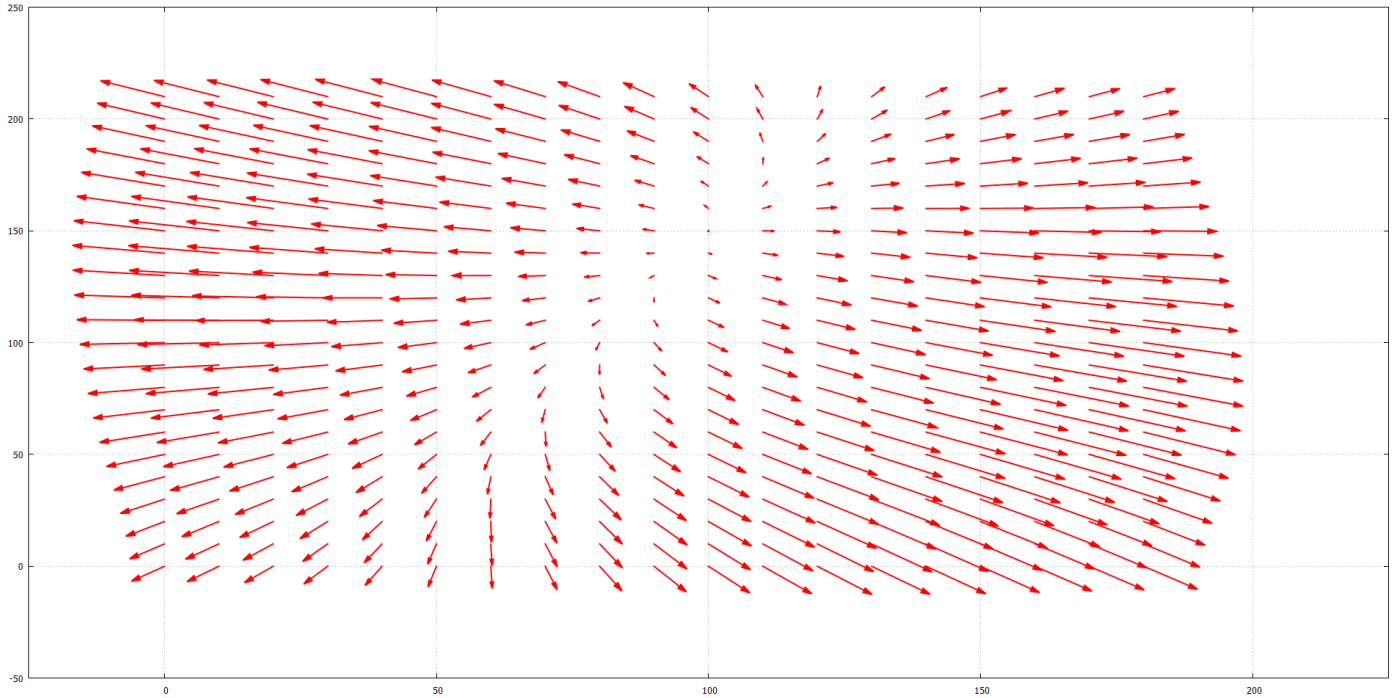

Figure27 : Champ de déplacement obtenu par le recalage des deux images de cerveau

La partie centrale des champs de déplacements des figures 26 et 27 sont sensiblement similaires. Cela correspond au recalage de la partie de l'image qui nous intéresse : la coupe transversale de cerveau. Les différences au bord sont dues aux zones noires des images de cerveau qui ne sont pas totalement noires.

6 Améliorations envisageables

Pour l'instant l'algorithme implémenté ne propose qu'une seule méthode de recalage élastique basé sur la maximalisation d'une certaine mesure de similarité : la Sum Squared Differences. Cependant, il existe d'autres mesures de similarité ou de dissimilarité . Par exemple en [A1], il est mentionné la mesure de corrélation. L'expression des forces à appliquer sur l'image mobile découle du gradient de la métrique utilisée. Ainsi, si la mesure de corrélation est utilisée, les forces seront différentes, et l'algorithme en sera totalement changé. Une amélioration envisageable est donc de proposer à l'utilisateur différentes métriques avec leur expression de forces associée.

La sortie intéressante de l'algorithme est un champ de déplacement. Cependant, pour des raisons liées à l'algorithme de ré-échantillonnage (voir en [A2]), la sortie en question est le champ de déplacement inverse de celui à appliquer sur l'image mobile pour la recaler sur l'image de référence. Il serait alors intéressant d'implémenter un algorithme permettant d'inverser le champ de déplacement en sortie de l'algorithme.

Enfin, l'algorithme doit aussi être implémenté en 3D.

7 Conclusion :

Ce sujet de stage s'est inscrit dans l'optique d'une amélioration continue de la librairie interne proposant des algorithmes de traitement d'images. Cette librairie contenait déjà un algorithme de recalage rigide d'images et le sujet du stage consistait en l'implémentation d'un algorithme de recalage élastique d'images. Le recalage élastique a l'avantage de recaler les images à une échelle plus fine que le recalage rigide. Le recalage d'images est la recherche d'un champ de correspondance dense entre deux images. Cette technique repose sur la recherche d'une transformation optimale à appliquer à l'image étudiée afin de la recaler sur l'image cible.
Dans ce but, il a été conçu deux classes en C++. Une classe pour calculer le champ de déplacements à appliquer sur l'image mobile à chaque itération. Le champ de déplacements est calculé à l'aide de l'algorithme des démons de Tirion. L'autre classe permet de ré-échantillonner une image aux dimensions souhaitées et de lui appliquer un champ de déplacement calculé précédemment. L'algorithme général suit une approche multi-échelle, le recalage est d'abord effectué sur des images sous-échantillonnées, de sorte à ce que l'algorithme soit plus stable mais aussi plus rapide. La dernière étape de ce stage a été la validation visuelle sur de vraies images de cerveau. Pour cela, il a été comparé les champs de déplacement en sortie de l'algorithme avec des champs de déplacements préalablement calculés avec un algorithme d'interpolation par Thin Plate Spline.

Ce stage a été une expérience positive. J'en aurai décousu avec l'image de l'ingénieur qui travaille tout seul dans son bureau. Être ingénieur en programmation c'est aussi prendre du temps pour s'intéresser au travail des gens autour de soi que ce soit autour d'un café ou lors de séances de peer-programming.
Le traitement d'images était un domaine complètement nouveau, cependant les mathématiques sous-jacentes auront été une très bonne surprise. L'expérience et les connaissances acquises en programmation seront probablement indispensables pour la suite de ma carrière. J'ai découvert de nouveaux outils comme Git et Cmake qui m'étaient totalement inconnus au début du stage et dont l'utilité sera indéniable pour mon avenir.

Annexe

[A1] Explications de l'algorithme des démons :

La démonstration suivante est largement inspirée du papier [B4].

On cherche à minimiser la grandeur suivante :

$$SSD(T) = \int (I - JoT)^2$$

Pour appliquer une descente de gradient, on recherche le minimum de cette distance. On peut par exemple effectuer un développement de Taylor de SSD (Sum Squared Difference) et ensuite identifier le gradient.

$$SSD(T+u) = \int (I - Jo(T+u))^2 = \int (I^2 - 2\, I\ Jo(T+u) + (Jo(T+u))^2)$$

En effectuant toujours un développement de Taylor :

$$Jo(T+u) = JoT + (\nabla_J\, o\ T)^T .\ u$$

Puis en réinjectant cette relation dans la première :

$$SSD(T+u) = SSD(T) + 2 \int (JoT - I) .\ (\nabla_J\, oT)^T .\ u + o(||u||^2)$$

Enfin par identification le gradient est :

$$\nabla_{SSD} = (JoT - I) .\ \nabla_J\, oT$$

La forme continue de la SSD a été utilisée pour cette démonstration mais évidemment c'est la forme discrète qui a été utilisée pour l'implémentation :

$$\sum_{x \in \Omega} (I(x) - J(x))^2$$

Une autre distance fréquemment utilisée en recalage d'images est la corrélation :

$$Corr(T) = \langle \frac{I - \mu(I)}{\sigma(I)} | \frac{J - \mu(J)}{\sigma(J)} \rangle_{L2}$$

avec $\mu(I) = |\Omega|^{-1} \int_\Omega I(x)\, dx$ et $\sigma(I) = \mu((I - \mu(I))^2)$

La corrélation entre deux images est le produit scalaire dans L_2 entre ces deux images normalisées. Cette distance peut donc être interprétée comme le cosinus de l'angle entre les deux images. Une condition nécessaire au bon recalage de deux images est déjà qu'elles soient alignées, c'est-à-dire que le cosinus de l'angle entre les deux images doit être égal à un ce qui va avoir pour effet de maximiser la corrélation.

[A2] Explications de l'algorithme de ré-échantillonnage (en 2D)

Le principe de cette classe est d'appliquer un champ de déplacement à l'image à recaler. Le champ de déplacement est sous la forme d'une image à deux ou trois composantes spectrales. Cette image contient en fait les déplacements $\overrightarrow{u(x)}$ associés à chaque pixel de coordonnées x . Chaque pixel va ainsi être déplacé par une transformation T, de sorte à ce qu'un pixel en x se retrouve en:

$$T: \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

$$T(x) = x + u(x)$$

Figure28 : Exemple de transformation T

La figure 28 met en avant le problème que doit gérer l'algorithme de ré-échantillonnage. La transformation T qui a été évaluée à l'aide de l'algorithme des démons ne fait pas forcément correspondre un pixel à une coordonnée entière de la grille de l'image de sortie.

Les difficultés de l'implémentation de cet algorithme résident dans le fait que l'image d'entrée, l'image de sortie, et l'image contenant le champ de déplacements sont définis potentiellement dans les des repères différents, c'est-à-dire avec une origine et un spacing (longueur d'un pixel dans une direction donnée) différents dans toutes les directions.

Après avoir créé une image de sortie au bon type et au bonnes dimensions, le code va parcourir les pixels de l'image de sortie et les différentes étapes implémentées dans le body (partie du code qui parallélise les calculs) sont les suivantes:

1) récupération des coordonnées d'un pixel de l'image de sortie dans le repère de l'image de sortie

2) conversion de ces coordonnées dans le repère physique

3) conversion des coordonnées du repère physique dans le repère de l'image contenant les déplacements

4) la nouvelle coordonnée n'ayant pas une valeur entière nécessairement, il faut l'interpoler :

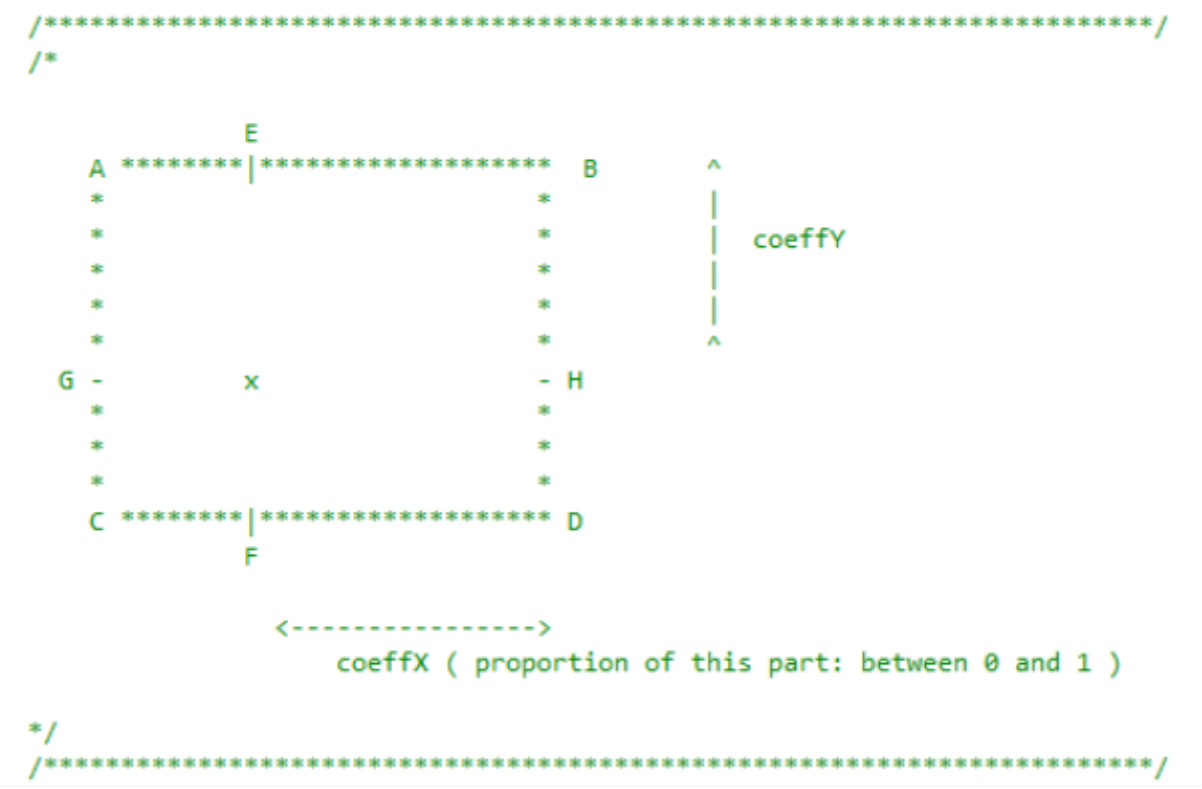

Figure29: Interpolation bilinéaire du déplacement

5) L'algorithme propose pour le moment deux méthodes d'interpolation. La première méthode est celle du plus proche voisin. Cette méthode est souvent favorisée dans le cas d'images binaires (deux niveaux de couleurs) ou label (à demander définition exacte). La deuxième est l'interpolation bilinéaire. Les déplacements sont interpolés aux points E et au point F puis ils sont interpolés au point courant entre E et F. C'est une interpolation bilinéaire.

6) Il faut ensuite convertir les coordonnées actuelles du pixel dans le repère de l'image d'entrée pour savoir où est-ce que va se trouver ce pixel après transformation.

7) Là encore les coordonnées obtenues ne sont pas entières et il est nécessaire de refaire une interpolation afin de récupérer une valeur approchée de l'intensité.

8) Il est assigné sa valeur d'intensité au pixel de l'image de sortie.

[A3] Explications Thin Plate Spline

Dans l'exemple de la figure 29, $(x\ ,\ y)$ représente les coordonnées $(x\ ,\ y)$ dans l'image de déplacement qu'on est en train de générer, et la surface qui vient interpoler les points de contrôle représente le déplacement en x par exemple. On a donc deux surfaces à trouver. La fonction d'interpolation que l'on cherche a la forme:

$$f\ (x\ ,\ y) = a_1 + a_2\ x + a_3\ y + \sum_{i\ =1}^{n} \omega_i\ \ U(\left|P_i\ - (x\ ,\ y)\right|)$$

avec $U(r) = r^2 \log\ r$

Les (P_i) sont les points de contrôle, les coefficients (a_i) rendent compte de la partie rigide de la surface, de sorte que si l'utilisateur ne définisse que trois points de contrôle la surface définie soit bien un plan, et les (ω_i) sont une famille de n coefficients témoignant de la courbure de la surface. Il est imposé à ces coefficients :

$$\sum_{i\ =1}^{n} \omega_i\ = 0,\ \sum_{i\ =1}^{n} \omega_i\ \ x_i\ = 0,\ \sum_{i\ =1}^{n} \omega_i\ \ y_i\ = 0$$

La mise en système des conditions d'interpolation aux points de contrôle $f\ (x_i\ ,\ y_i) = v_i$ ainsi que les équations citées ci-dessus mène à l'équation suivante:

$$V = L\ W$$

avec :

$$V = \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \\ 0 \\ 0 \\ 0 \end{pmatrix}\ ,\qquad W = \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_n \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}\ ,\qquad L = \begin{pmatrix} K & P \\ P^T & 0 \end{pmatrix}\ ,\qquad P = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ \vdots & & \vdots \\ 1 & x_n & y_n \end{pmatrix}$$

$$K = \begin{pmatrix} U(r_{11}) & U(r_{12}) & \cdots & U(r_{1\,n}) \\ \vdots & & & \vdots \\ U(r_{n\,1}) & U(r_{n\,2}) & \cdots & U(r_{nn}) \end{pmatrix}\qquad \text{avec}\quad U(r) = r^2 \log\,(r)\quad \text{et}\quad r_{i\,j} = \left|P_i\ - P_j\right|$$

Le système étant assez petit (taille du nombre de points définis par l'utilisateur) je ne me suis pas vraiment posé de questions concernant l'efficacité des différentes méthodes de résolution de systèmes linéaires, j'ai choisi la décomposition en valeur singulière proposée par Eigen avec une méthode de Jacobi two-sided.

Bibliographie :

[B1] site ITK, librairie open source de traitement d'images pour la médecine
https://itk.org/ITKExamples/src/Registration/Common/PerformMultiModalityRegistrationWithMutualInformation/Documentation.html

[B2] http://brainsuite.org/wp-content/uploads/2015/01/BCI-DNI_brain_atlas_subject_mapped2.png

[B3] Numerical Methods for Image Registration, Jan MODERSITZKI, Oxford Science Publications

[B4] Understanding the "Demon's Algorithm": 3D Non-rigid Registration by Gradient Descent, Xavier Pennec, Pascal Cachier, and Nicholas Ayache

[B5] https://en.wikipedia.org/wiki/Pyramid_(image_processing)

[B6] site utilisé pour comprendre l'interpolation splinique:
http://profs.etsmtl.ca/hlombaert/thinplates/

[B7] site brainweb utilisé pour récupérer des images 3D de cerveau:
http://brainweb.bic.mni.mcgill.ca/brainweb/

Remerciements :

Tout d'abord, j'aimerais remercier mon tuteur Sébastien MARTIN pour sa patience et pour son enseignement en traitement d'images et en C++.

Je remercie également Delphine DUCLAP, Yohan PICCOLI, Damien PERAL, Sébastien FAVE qui ont su me rendre ce stage plus doux par leur bonne humeur et leur pédagogie.

Je remercie Martial PAPILLON pour le temps précieux qu'il m'a accordé et pour ses apartés culturels.

Enfin, je voudrais dire un grand merci à mes parents et à Boris Rosiers qui m'ont soutenu au début de ce stage dans les moments de doute.