PIDR :
Tomographie et rétroprojection filtrée

Abel Quérou
Elève ingénieur en 2ème année
à Télécom Nancy

Pierre-Arnaud Blanc
Elève ingénieur en 2ème année
à Télécom Nancy

Jean-Baptiste Bellet
chercheur-encadrant PIDR
Institut Elie Cartan de Lorraine

***Résumé*—La tomographie est un moyen efficace de sonder la matière sans l'endommager, ce qui est très utile dans beaucoup de domaines. L'objet à étudier est déplacé (rotation, translation verticale) à l'intérieur d'un faisceau de rayonnement pénétrant. Une partie de l'énergie sera absorbée et chaque traversée du faisceau donne un point, formant ainsi l'image d'une superposition de sinusoïdes, appelée "sinogramme", inexploitable directement. Une transformation mathématique, dite de Radon, est nécessaire pour revenir à la forme de l'objet initial. Dans notre étude, nous avons appliqué une $\ll$ rétroprojection filtrée $\gg$ d'abord sur des exemples théoriques (disque) et l'avons ensuite implémentée en Matlab, puis en langage C. Enfin, notre code fut appliqué sur les fantômes de Shepp-Logann et nous avons obtenu un optimum pour le PSNR autour de 12 à 13 qui se trouve au voisinage d'un pas de 0 à 2 et d'une taille de 100 à 200.**

Mots-clés : tomographie, transformée de Radon, rétroprojection filtrée, Matlab

I. Introduction

Contexte général : La tomographie, qui veut dire étymologiquement "dessiner une coupe" en grec, est une technique permettant d'obtenir des images de coupes d'objets ou d'êtres vivants à partir d'un rayonnement, habituellement des rayons X. Son histoire est intiment liée à celle de l'imagerie médicale, qui va se développer à partir de 1896 avec l'utilisation des rayons X [1]. Mais les simples radiographies sont limitées, car elles ne prennent pas en compte la structure tridimensionnelle du corps humain. Pour pallier ce problème, le scanner est créé en 1971 par Godfrey Hounsfield et ses collègues à l'EMI Group de Londres (les producteurs des Beatles). Cette avancée majeure de l'imagerie médicale [2] n'a pu aboutir que par la remise au goût du jour des travaux du mathématicien austro-hongrois Johann Radon en 1917 sur la théorie de la transformée qui porte maintenant son nom. Cette théorie trouvera donc son application 60 ans plus tard , notamment lors de l'invention du scanner par Hounsfield en Angleterre associée aux travaux théoriques sur la tomodensitométrie par l'américain d'origine sud-africaine Allan MacLeod Cormack, qui leur vaudra à tous deux le prix Nobel de médecine en 1979. La tomographie est un moyen très efficace de sonder la matière sans l'endommager. C'est pour cela qu'elle est utilisée aussi bien en médecine en imagerie médicale, en archéologie et en lutherie pour l'étude d'artéfacts et d'objets fragiles, en industrie pétrochimique pour l'étude du sous-sol ou bien dans le domaine nucléaire pour l'étude d'éléments radioactifs non manipulables ou confinés [3]. Les applications de la tomographie sont donc plurielles et importantes, ce qui a entrainé des études multiples, avec de nombreuses retombées.

Problématique : La tomographie nécessite donc une partie logicielle destinée à procéder à une reconstruction tridimensionnelle de l'objet étudié. En effet, l'image obtenue lors d'une tomographie n'est pas directement exploitable : elle est formée d'une superposition de sinusoïdes, appelée sinogramme. En théorie, le sinogramme est obtenu après une transformée de Radon de l'objet d'étude. Pour que l'image soit lisible et exploitable par un être humain, il est nécessaire de faire une reconstruction. Il existe plusieurs méthodes différentes de reconstruction, mais nous nous intéresserons ici uniquement à la rétro-projection filtrée.

Approche proposée : Le but de ce Projet Interdisciplinaire ou de Découverte de la Recherche (PIDR) est, après avoir effectué une recherche bibliographique sur la tomographie et les méthodes de reconstructions mathématiques existantes, d'établir une reconstruction d'une image en 2D, en commençant par l'image des fantômes de Shepp-Logan. C'est une image d'un modèle humain créée par L. Shepp et B.F. Logan en 1974, qui sert de test pour les algorithmes de reconstructions.

Notre méthode de rétro-projection filtrée, qui sera codée en langage Matlab, se déroule en quatre étapes : (i) on applique une transformée de Fourier à l'image de départ, (ii) on filtre le résultat, (iii) on applique une transformée de Fourier inverse et (iv) finalement on réalise la rétroprojection. En plus des résultats de notre code, nous présenterons une application sur un disque. Dans un second temps, nous avons converti ce programme en C, un langage compilé, pour gagner du temps calcul, car la complexité temporelle du code est assez importante, parce que le nombre de données devient très élevé dès que l'image devient importante.

Bibliographie sommaire : Le cours très complet sur la tomographie de Télécom Paris Tech de la référence [4] présente diverses méthodes de reconstruction de la fonction f de l'atténuation en (x, y) du volume traversé. Il distingue d'une part les méthodes dites analytiques, où l'on discrétise les formules d'inversion, et d'autre part les méthodes algébriques où l'on discrétise l'équation de projection, et on obtient un système d'équations linéaires. Il cite plusieurs méthodes analytiques : la Transformée de Fourier Discrète avec rétroprojection discrète, la rétroprojection discrète, l'in-

version directe discrète, la déconvolution 2D discrète et la rétroprojection filtrée discrète. Il indique que la rétroprojection filtrée est la méthode la plus répandue car elle présente de nombreux avantages. Plusieurs filtres sont possibles, comme les filtres numériques RAM-LAK et de Shepp et Logan, le filtre cosinus, de Hamming ou celui de Wiener. Dans notre cas, nous avons appliqué la rétroprojection filtrée avec le filtre de Ram-Lak.

La référence [5] donne aussi une approche théorique de la rétroprojection filtrée. Elle discute des mérites respectifs des filtres rampe et des filtres lissants, comme celui de HANN ou HAMMING ou de leur multiplication. Dans notre cas, nous nous sommes limités au filtre rampe.

La référence [6] donne également une approche théorique de la rétroprojection filtrée, et insiste sur les différents filtres possibles.

La référence [7] présente les avantages et inconvénients de 2 méthodes de reconstruction : la rétroprojection filtrée et la reconstruction itérative. On retrouve les deux méthodes (l'une analytique et l'autre algébrique) expliquées dans le cours d'Isabelle Bloch.

La référence [8] est un mémoire de fin d'études vietnamien qui donne les éléments d'un code en Matlab pour modéliser la transformée de Radon dans des applications médicales.

La référence [9] présente une collection complète de routines en langage C pour calculer la Transformée de Fourier Discrète avec certains cas particuliers, le plus rapidement possible.

II. Explication Théorique

A. Principe physique pour acquérir l'image tomographique

Lorsqu'on envoie des rayons X à travers un objet, ce dernier atténue une partie de l'énergie reçue. Cette atténuation est linéaire, nous permettant d'avoir une formule simple de l'intensité reçue (I) par rapport à l'intensité émise (I_0) en intégrant l'atténuation linéaire sur une droite D :

$$I = I_0 \cdot \exp(-\int_D f(x,y)dv)$$

L'intensité reçue donne une mesure intégrale de la « densité » des différentes couches traversées par le faisceau de rayonnement lors de sa traversée de l'objet. S'il s'agit d'un corps humain, par exemple, cette densité va varier selon que le faisceau traverse des masses osseuses, du tissu mou ou des cavités. On cherche donc à reconstruire $f(x,y)$ en tout point de l'espace pour obtenir la coupe de l'objet étudié.

Comme chaque mesure correspond à une droite D, le dispositif d'acquisition doit faire un balayage angulaire de l'objet à étudier pour obtenir l'ensemble des mesures permettant de reconstituer l'objet d'étude (Fig.1).

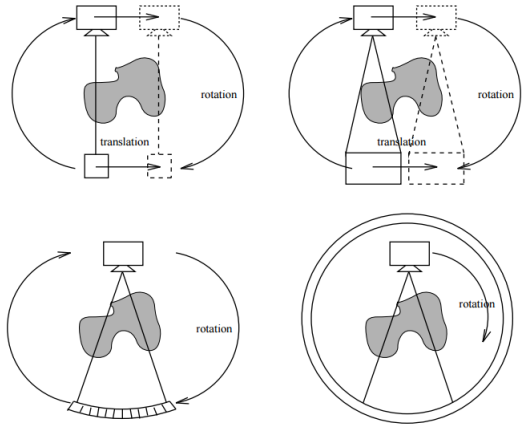


FIGURE 1. Différents balayages [4]

B. Transformée de Radon

La transformée de Radon, qui correspond à l'opérateur des projections de f, donc de l'image que l'on obtient après une radiographie, forme une image appelée sinogramme (Fig.2).

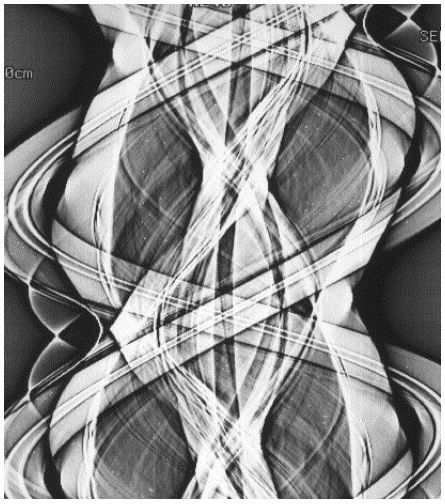


FIGURE 2. Exemple de sinogramme[4]

La transformée de Radon s'exprime par la relation suivante : soit $f(x,y)$ une fonction continue. La transformée de Radon le long d'un droite L est définie par :

$$R_f = \int_L f(x,y)dl$$

Comme droite, nous allons utiliser la droite des projetés, donc L est défini par $p = x\cos\theta + y\sin\theta$ où θ est l'angle de l'axe des rayonnements par rapport à l'origine. La perpendiculaire à L est défini par $q = -x\sin\theta + y\cos\theta$ comme on peut le voir en Fig.3.

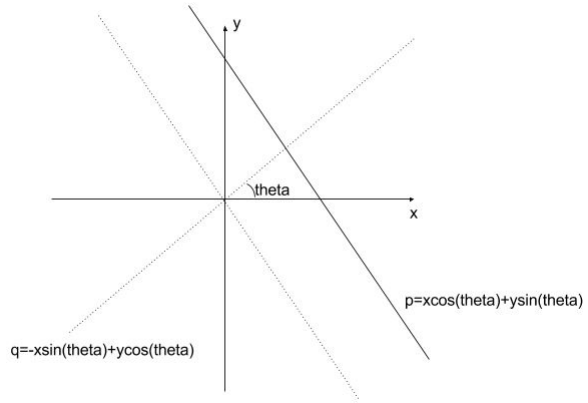


FIGURE 3. illustration pour la transformée de Radon

Ainsi à partir des deux équations de droites, nous avons obtenu $x = p\cos\theta - q\sin\theta$ et $y = p\sin\theta + q\cos\theta$, et en remplaçant dans $f(x, y)$ on obtient ainsi :

$$\int_{-\infty}^{+\infty} f(p\cos\theta - q\sin\theta, p\sin\theta + q\cos\theta)dq$$

Calcul de la transformée de Radon d'un disque : Soit S la surface du disque unité, définie par la relation :

$$S = \{(x, y) \in \Re^2 | x^2 + y^2 < 1\}$$

$$\text{soit } f : \begin{cases} \Re^2 \to \Re \\ (x, y) \mapsto \begin{cases} 1 & si \quad x^2 + y^2 < 1 \\ 0 & sinon \end{cases} \end{cases}$$

Nous avons donc : $R(\theta, p) = \int_{-\infty}^{+\infty} f(x, y)\delta q$ la transformée de Radon de ce disque. On trouve facilement que :

$$p^2 + q^2 = 1 \to q = \pm\sqrt{1 - p^2}$$

et par linéarité de l'intégrale dans la formule de la transformée de Radon :

$$R(\theta, p) = \int_{-\infty}^{-\sqrt{1-p^2}} f(x, y)dq +$$

$$\int_{-\sqrt{1-p^2}}^{+\sqrt{1-p^2}} f(x, y)dq + \int_{+\sqrt{1-p^2}}^{+\infty} f(x, y)dq$$

On simplifie quand $f(x, y) = 0$, c'est à dire quand $|p| > 1$:

$$R(\theta, p) = \int_{-\sqrt{1-p^2}}^{+\sqrt{1-p^2}} f(x, y)dq$$

ce qui donne finalement :

$$R(\theta, p) = \begin{cases} 2\sqrt{1 - p^2} & si|p| \leq 1 \\ 0 & sinon \end{cases}$$

La Fig.4 représente le disque et sa projection.

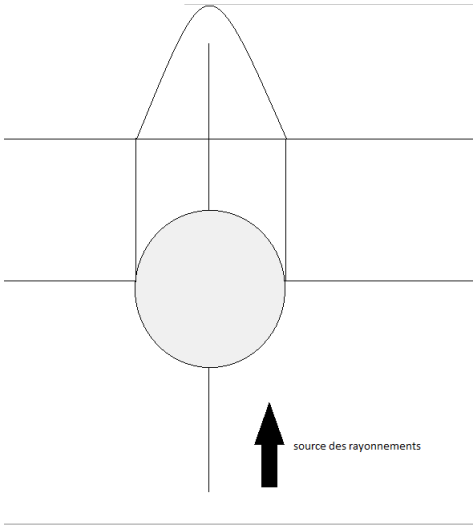


FIGURE 4. transformée de Radon d'un disque et sa projection

C. La rétroprojection filtrée

Le principe de la rétroprojection filtrée est de sommer l'ensemble des valeurs de projections obtenues par les acquisitions effectuées autour de l'image. La formule est la suivante :

Soit p_θ l'ensemble des mesures de projections, on a :

$$Retroprojection(p) = \int_0^\pi p_\theta(x\cos\theta + y\sin\theta)\delta\theta$$

Trois choses sont à noter : (i) la rétroprojection se calcule analogiquement, (ii) la rétroprojection n'est pas l'inverse de la transformée de Radon, (iii) l'image retournée est l'image d'origine floutée.

En effet, lorsqu'on établit la relation entre la rétroprojection et l'image de départ, on remarque que cela correspond à la réponse impulsionnelle d'un filtre. Ainsi, pour atténuer le flou de la rétroprojection, un filtrage dans le domaine de Fourier est effectué [4].

III. RÉSULTATS

A. résultats de notre code en langage Matlab

1) la génération de données : Nous avons utilisé dans un premier temps comme objets d'étude les images d'un modèle humain appelées les fantômes de Shepp-Logan. Créées par ces derniers en 1974, ils servent de test pour les algorithmes de reconstructions (Fig.5).

FIGURE 5. Fantômes de Shepp-Logan tels que générés par Matlab

Pour modéliser la phase rayonnement pour obtenir le sinogramme sur lequel nous allons travailler pour la rétroprojection filtrée, nous avons appliqué la fonction Matlab de transformée de Radon. Cette fonction permet de choisir l'angle du rayonnement simulé ainsi que le nombre de mesures en faisant varier le pas de discrétisation. En sortie de la transformée de Radon, nous obtenons une matrice R qui regroupe l'ensemble des mesures récoltées par angle de rayonnement simulé et un vecteur xp contenant les coordonnées radiales correspondant à chaque rangée de R. Le sinogramme ainsi obtenu est le suivant (Fig.6) :

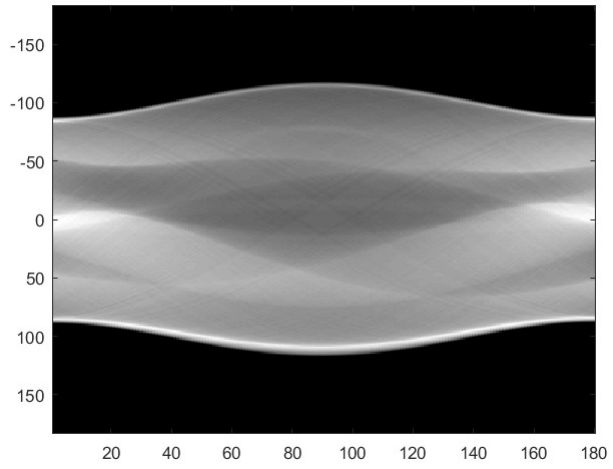


FIGURE 6. Sinogramme des fantômes de Shepp-Logan

2) rétroprojection discrète : **Filtre de l'image :** La première étape consiste à appliquer une transformée de Fourier à la matrice R résultante de la transformée de Radon de Matlab. Ensuite nous avons ensuite créé et utilisé un filtre de Ram-Lak (Fig.7). Ce filtre a trois avantages majeurs : (i) il est simple à implémenter (deux rampes), (ii) il s'effectue rapidement, donc on gagne en temps de calcul pour la reconstruction, et (iii) il atténue efficacement le bruit sans trop lisser l'image.

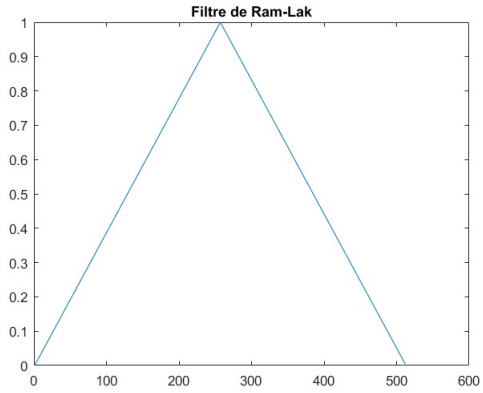


FIGURE 7. filtre de Ram-Lak

On applique le filtre à la matrice R par multiplication et appliquons au résultat une transformée de Fourier inverse.

rétroprojection : On crée une matrice B de la taille de R qui recueillera l'image résultat. En bouclant sur les colonnes de R, donc chaque angle de mesure, on détermine les projections u avec la formule suivante : $u = (((x-R_x)/2)*\cos(rad)-((y-R_y)/2)*\sin(rad))+xp_{offset}$ avec (x, y) les points de B, R_x et R_y la taille de R, rad l'angle en radian de l'axe de projection et xp_{offset} le décalage. Pour être sûr que les projections u soient entières, nous appliquons une interpolation de Taylor. Nous utilisons finalement une LUT pour restreindre les valeurs résultats dans l'ensemble [0 ;255]. On rappelle qu'une Look-Up Table (LUT) est utilisée en traitement d'image pour établir une correspondance entre les valeurs des points et les couleurs affichées.

L'image ci-après (Fig.8) montre que nous avons bien obtenu un résultat du fantôme de départ. Il nous faut maintenant évaluer sa qualité.

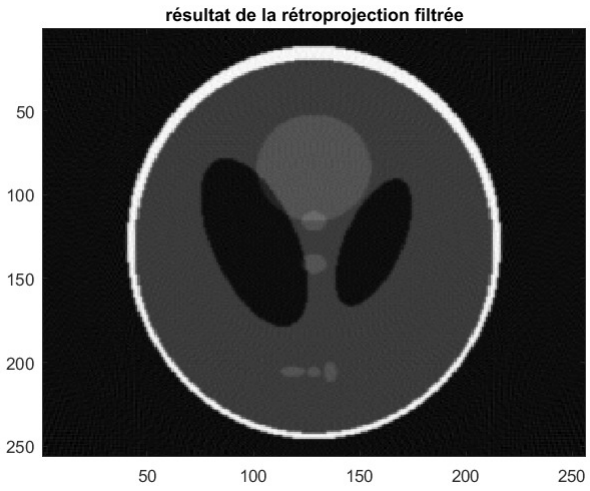


FIGURE 8. Résultat de la rétroprojection filtrée obtenus suite à notre code Matlab

3) quantification et évaluation des résultats: Pour évaluer les images obtenues, nous avons utilisées le ratio PSNR (Peak Signal to Noise Ratio), qui se calcule de la manière suivante : $PSNR = 10 \cdot \log_{10}(\frac{d^2}{EQM})$ où EQM est l'erreur quadratique moyenne qui s'obtient par la formule suivante : $EQM = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}(I_0(i,j) - I_f(i,j))^2$ où I_0 est l'image originale et I_f l'image finale. Ce ratio est la manière classique d'évaluer en traitement de l'image les distorsions par rapport à l'image d'origine. Plus ce ratio augmente, plus l'image est de bonne qualité.

Ainsi nous pouvons faire varier la qualité de l'image résultante de la rétroprojection en modifiant la taille de l'image en pixels ou le pas angulaire des rayonnements simulés.

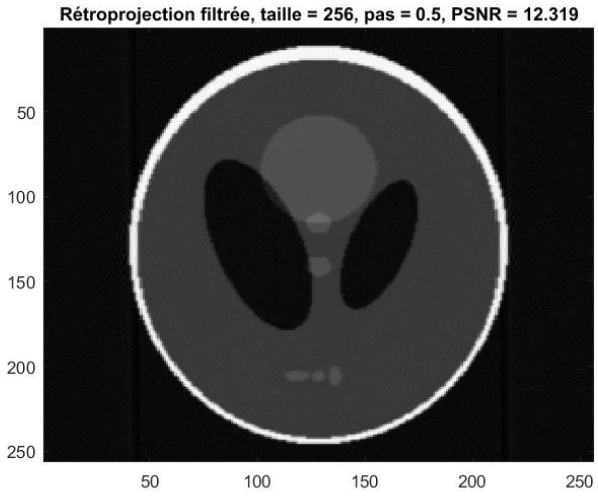


FIGURE 9. Résultat taille=256 pas=0.5

Pour la Fig.9, avec un pas angulaire de 0.5 et une image à 256 pixels, on observe une image assez proche de l'original, avec un $PSNR = 12,32$. Cette image est tout à fait interpretable par un humain, à la différence du sinogramme.

3.1 : influence du pas : Observons maintenant l'influence du pas angulaire dans les figures suivantes :

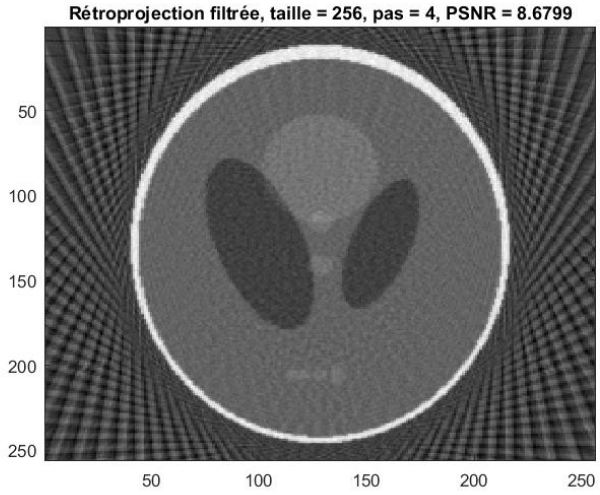


FIGURE 10. Résultat taille=256 pas=4

Ici l'image de la Fig.10 a un pas angulaire de 4. Des lignes parasites dues à un trop grand pas angulaire apparaissent. Le PSNR diminue à 8,68.

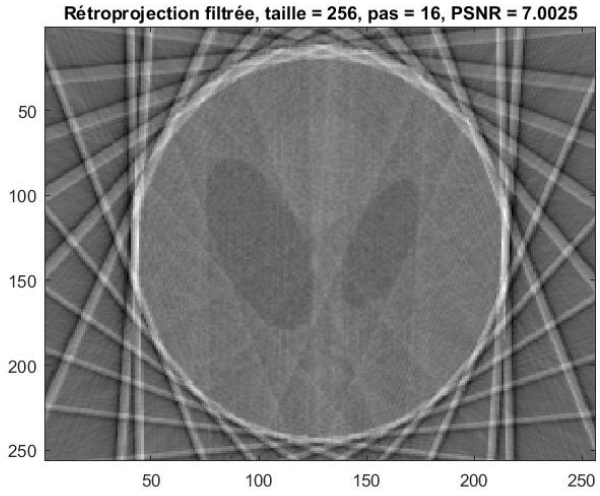


FIGURE 11. Résultat taille=256 pas=16

En augmentant encore le pas angulaire jusqu'à 16, on observe que l'image de la Fig.11 devient quasiment inexploitable, qu'elle perd nettement en contraste et que les lignes parasites sont encore plus nettes. Le PSNR chute à 7.

A des fluctuations près, pour un pas de 1 degré, le PSNR forme une cloche, avec un maximum de 13 autour d'une image de taille 90. C'est donc la taille qu'il faut opter pour un pas de 1.

3.2 influence de la taille : Observons maintenant l'influence de la taille sur la qualité de l'image.

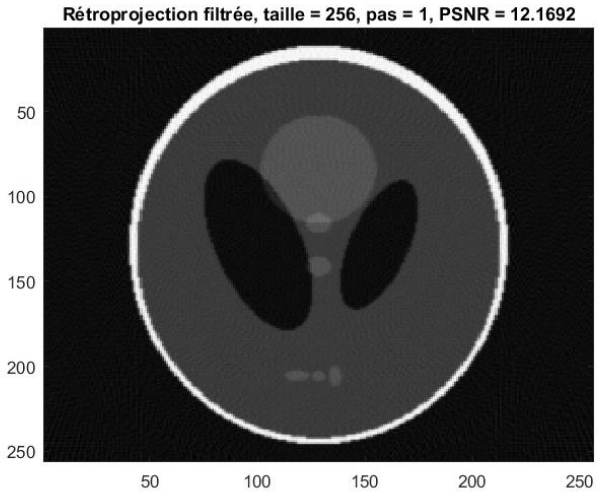


FIGURE 12. Résultat taille=256 pas=1

On va utliser la Fig.12 comme étalon pour observer l'influence de la taille. La Fig.12 a une taille de 256 pour un pas de 1 et elle obtient un PSNR de $12,17$.

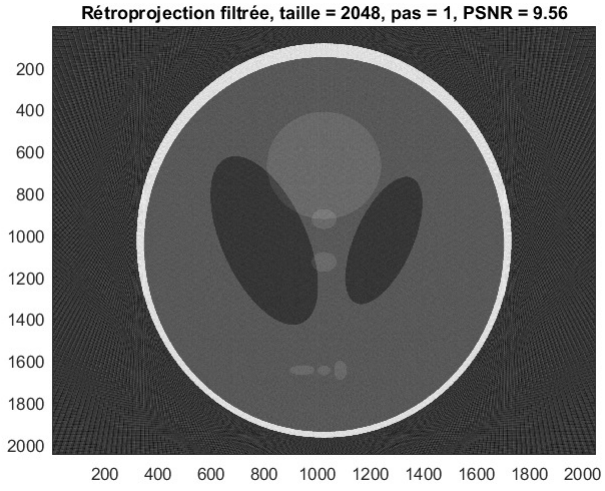


FIGURE 13. Résultat taille=2048 pas=1

Cette image (Fig.13) a un pas angulaire de 1 comme la Fig.12, mais à une taille plus grande de 2048 pixels. Le PSNR de 9,56 est plus bas que celui de la Fig.12, mais cela s'explique par le fait qu'on a augmenté la taille de l'image, sans pour autant diminuer le pas angulaire, aboutissant pour un même pas angulaire à un résultat moins efficace.

3.3 optimisation en fonction de la taille et du pas : Pour voir l'influence de ces paramètres, nous avons calculé le PSNR sur 1000 images, de taille allant de 1 pixels à 1000, avec un pas variant lui aussi.

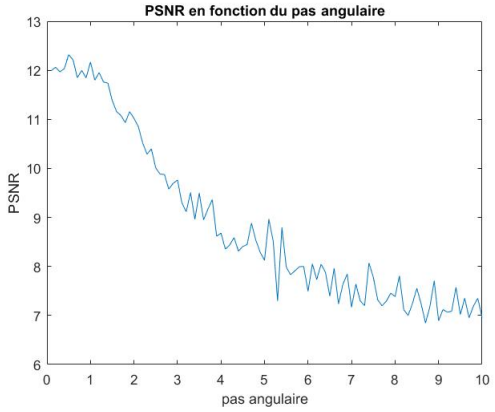


FIGURE 14. PSNR en fonction du pas

La Fig.14 montre la variation du PSNR d'une image de 256 pixels en fonction du pas angulaire. On remarque qu'il diminue drastiquement jusqu'à 10 rad, puis qu'il se stabilise.

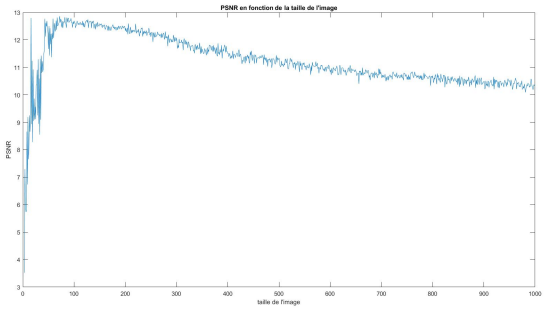


FIGURE 15. PSNR en fonction de la taille

La Fig.15 permet d'observer que le PSNR diminue légèrement avec l'augmentation de la taille de l'image pour un pas fixe de 1. Pour les images les plus petites, le PSNR n'a pas de réel signification, car les images sont tellement petites que la variation angulaire n'a plus de sens.

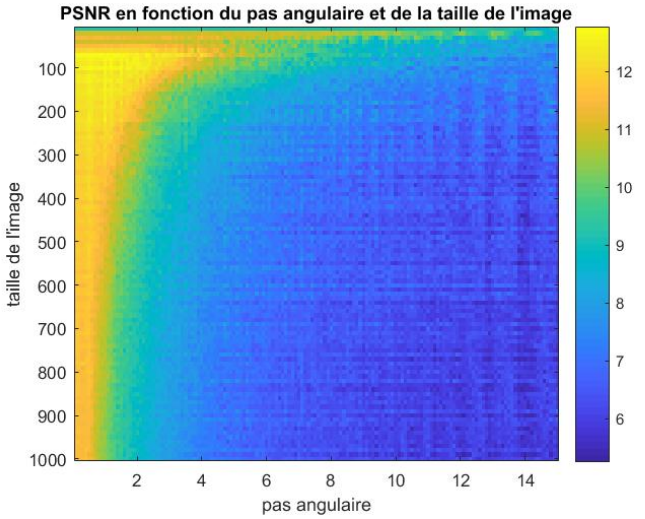


FIGURE 16. PSNR en fonction de la taille et du pas

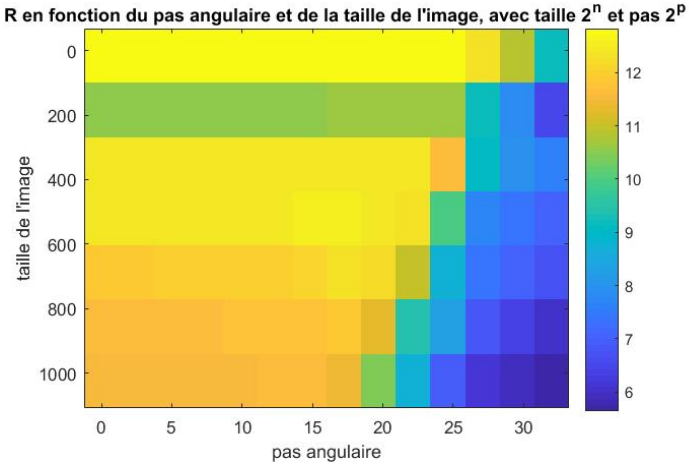


FIGURE 17. PSNR en fonction de la taille et du pas

Enfin nous avons fait un graphe (Fig.16 et Fig.17) qui étudie la variation des deux paramètres à la fois. Pour plus de lisibilité, cette figure (Fig.16) se trouve également en annexe. Les meilleurs résultats se situent autour de la taille 100 à 200 et d'un pas de 0 à 2.

Le pas angulaire semble donc être le paramètre le plus sensible sur la qualité du rendu. Dès qu'on prend un pas angulaire supérieur à 1 radian, l'image se dégrade très rapidement.

L'optimum se trouve au voisinage d'un pas de X et d'une taille de Y, où le PSNR atteint X.

B. Adaptation de notre code en langage C

Le code que nous avons écrit en langage C est équivalent algorithmiquement au code Matlab. Notre plus grande difficulté fut de remplacer des fonctions pré-établies dans Matlab par des librairies équivalentes.

1) chargement des données : Pour la génération de l'image des fantômes de Shepp-Logan, nous avons récupéré celle de Matlab, et nous avons enregistré sa transformée de Radon dans deux documents textes, l'un contenant les valeurs de la matrice R, l'autre contenant des informations sur la transformée (taille de l'image, offset).

Nous récupérons l'ensemble dans notre code, le tableau local grâce à un parser (analyse syntaxique de notre fichier texte). L'ensemble est copié dans un tableau alloué dynamiquement pour pouvoir finalement le renvoyer en valeur de retour.

2) filtre : L'enjeu essentiel pour la réalisation des filtres est l'utilisation de la transformée de Fourier. Nous avons utilisé la librairie fftw. Notre principal problème lors de l'utilisation de fftw fut la différence de format de données. Dans notre code, on utilise un tableau à deux dimensions (hauteur, largeur) alors que fftw utilise un tableau à une dimension. Donc la première étape a été la mise en forme des données pour qu'elles soient compatibles avec fftw. C'est une étape particulièrement lente car on parcourt le tableau en entier. Cette étape n'a pas été nécessaire en Matlab. Ensuite nous avons pu enfin appliquer la fonction fftw_plan_dft_2d et fftwexecute et ainsi après création de notre filtre de Ram-Lak, l'appliquer aux résultantes des fonctions de fftw par multiplication complexe. Finalement, on applique la transformée inverse de fftw et on re-adapte le format pour la suite du code.

3) Rétroprojection : Notre fonction rétroprojection prend deux paramètres, le tableau précédent sous format double** et le document texte qui contient les informations sur la transformée. L'algorithme est similaire à celui du C, mais les index des tableaux commencent à 0 et non à 1 comme en Matlab. Certaines sécurités ont aussi été mises en place pour éviter les erreurs de type "core dump" comme les sorties de tableaux.

4) Ecriture des données : Ensuite, les résultats sont écrits dans un fichier texte, puis ré-interprétés par Matlab pour être affiché. On applique au préalable une LUT d'expansion dynamique pour occuper toutes les plages de gris.

5) Fonction annexe : Nous utilisons deux fonctions annexes, une pour afficher la matrice placée en paramètre dans le terminal, essentiellement dans un but de contrôle du bon déroulement de l'algorithme, l'autre fonction permet la libération mémoire des tableaux alloués dynamiquement.

6) Comparaison avec Matlab et résultat : **Résultats en C :** Nos résultats en C ne sont pas identiques à ceux en Matlab pour une image de taille=256 et de pas=1, on obtient en Matlab un PSNR de 4,57 et en C un PSNR de 4,93. Cela est dû entre autres à des problèmes avec le filtre, notamment la transmission de donnée entre fftw avec le reste du code, de problème d'échange entre le C et Matlab et de décalage d'indice.

Comparaison avec Matlab : Malgré des résultats peu significatifs, on a un temps d'exécution relativement similaire. C'est dû notamment au fait que Matlab est conçu pour le calcul matriciel, alors que le C ne l'est pas. En C, notre code est trop proche de celui en Matlab, donc multiplication de parcours de tableau. Il en résulte un ralentissement du programme.

IV. Conclusion

Comme expliqué dans cet article, nous avons appliqué une « rétroprojection filtrée » d'abord sur des exemples théoriques (disque) et l'avons ensuite implémentée en Matlab, puis en langage C. Enfin, notre code fut appliqué sur les fantômes de Shepp-Logan en faisant varier le pas angulaire et la dimension de l'image. Les PSNR varient pour un pas de 1 radian entre 12.41 pour une image de 64 pixels et 9.56 une image de 2048 pixels et varient pour une image à 512 pixels entre 11.28 pour un pas de 1 et 6.1 pour un pas de 64 radian. On remarque aussi que le pas angulaire joue beaucoup plus sur l'interprétation humaine de l'image résultante que la dimension de l'image. Les temps de calculs sont prohibitifs dès que les dimensions des images calculées deviennent très importantes. En effet, pour un test sur 1000 images de 1 à 1000 pixels avec un pas variant de 1 à 90, il nous a fallu près de 8h pour effectuer l'ensemble des calculs (rétroprojection+PSNR). Notre code doit être optimisé si l'on veut l'appliquer à des images conséquentes, on devrait implémenter correctement les filtres, le code C devrait pouvoir afficher par lui même les images résultats, et il reste à le tester sur des sinogrammes d'acquisition d'objets d'études réels.
L'étude de la tomographie et de la rétroprojection filtrée fut instructive pour nous lors de la réalisation de ce PIDR.

Remerciements

Nous tenons à remercier notre encadrant PIDR, M. Bellet, qui a été présent durant ces 5 mois d'étude pour répondre à nos questions, nous conseiller sur certains choix à prendre. Son expertise dans le domaine de la tomographie a su nous orienter lors de la quinzaine de réunions téléphoniques que nous avons eu les mercredis de cette année. Le sujet qu'il nous a proposé fut riche d'enseignement et instructif pour nous deux.

Nous tenons à remercier également M. Bombardier, professeur de Traitement Numérique de l'Image à Télécom Nancy en deuxième année, car son cours fut particulièrement utile par comprendre les traitements que nous avons dû effectuer sur nos images et analyser nos images résultantes.

RÉFÉRENCES

[1] Grasland-Mongrain Pol, *Le principe de la tomographie* France : 1 Mai 2014,(http://polgm.free.fr/petitescuriesdunet/index.php?post/2014/04/Le-principe-de-tomographie)

[2] Desbat L., *Le scanner médical, la tomographie* TIMC-IMAG, UJF : 2004

[3] Blanc JY, Clément B, Von der Hardt P, *Fuel bundle examination techniques for the Phebus Fission Product Test*, IAEA TCM on behaviour of LWR Core Materials under Accident Conditions. Dimitrovgrad, Russia : 9-13 Oct, 1995 (http://www.iaea.org/inis/collection/NCLCollectionStore/_Public/28/023/28023810.pdf)

[4] Bloch Isabelle, *Reconstruction d'images de tomographie*, Télécom ParisTech France (Isabelle.Bloch@enst.fr)

[5] Dubois F., *Reconstruction des images tomographiques par rétroprojection filtrée*, CHU Saint Etienne, Revue de l'ACOMEN, vol 4, numéro 2 Saint Etienne, France : 1998

[6] Buvat Irène , *Reconstruction tomographique* France : Mars 2011, (http://www.guillemet.org/irene/coursem/APRAMENrecon.pdf)

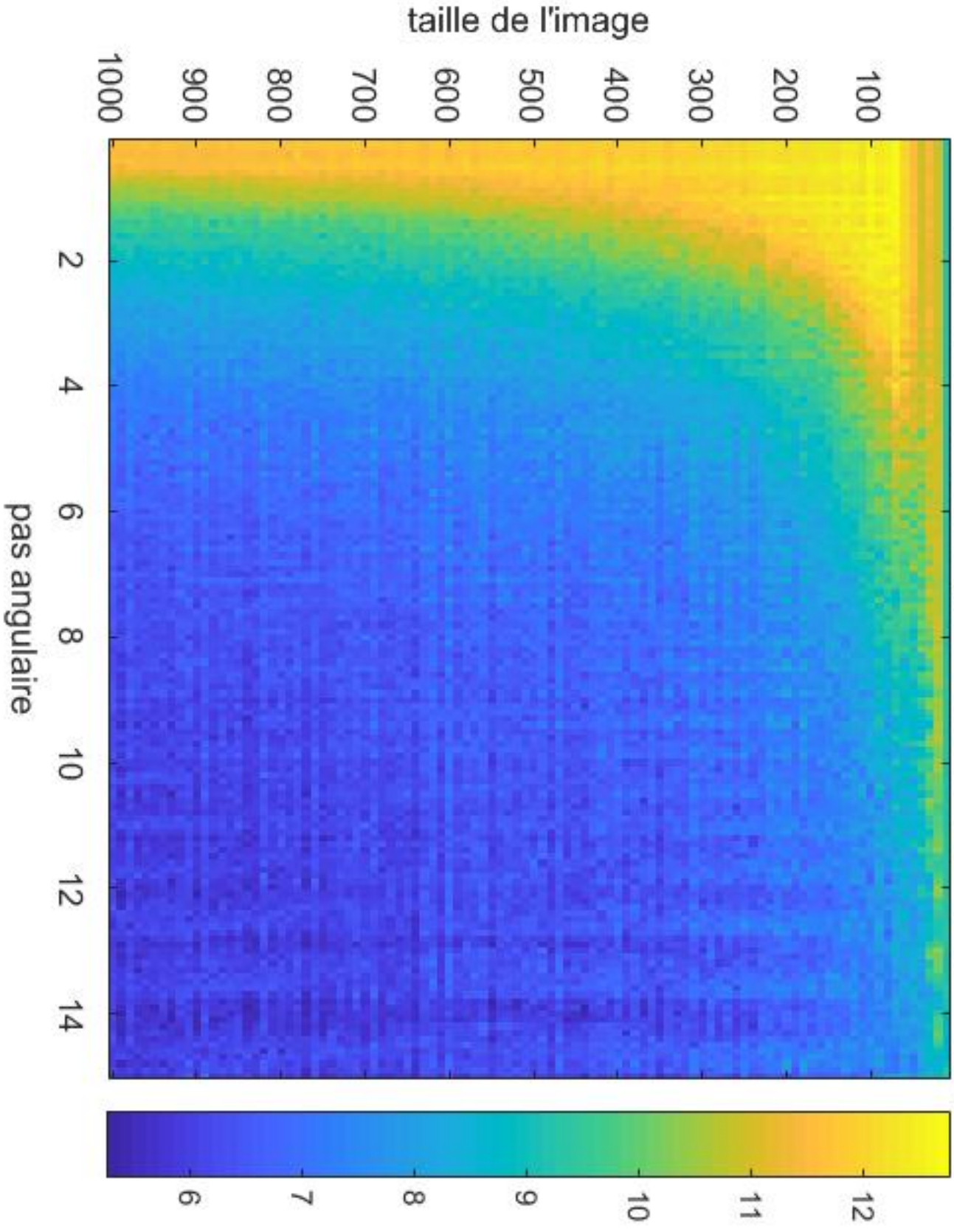
[7] Buvat Irène, *Rétroprojection filtrée et reconstruction itérative*, U494 INSERM Paris, France : 8 Janv. 2002 (http://www.guillemet.org/irene/coursem/APRAMENrecon.pdf)

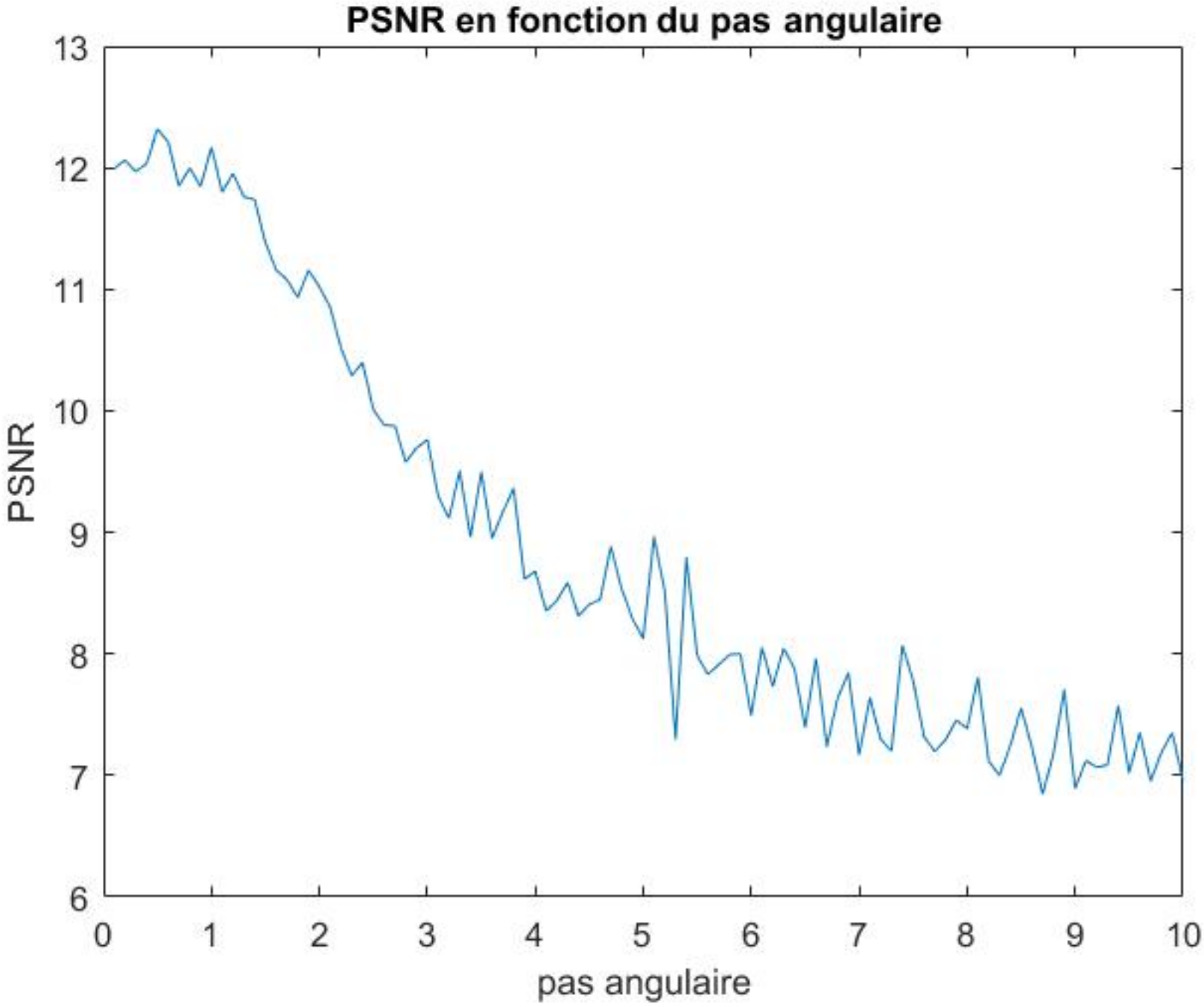
[8] Trong Tôn Pham, tuteur : Dr.Nguyen Dinh Thuc, *La transformation de Radon et son application en traitement de la radiographie pulmonaire* Ho Chi Minh ville, Vietman : Juillet 2004

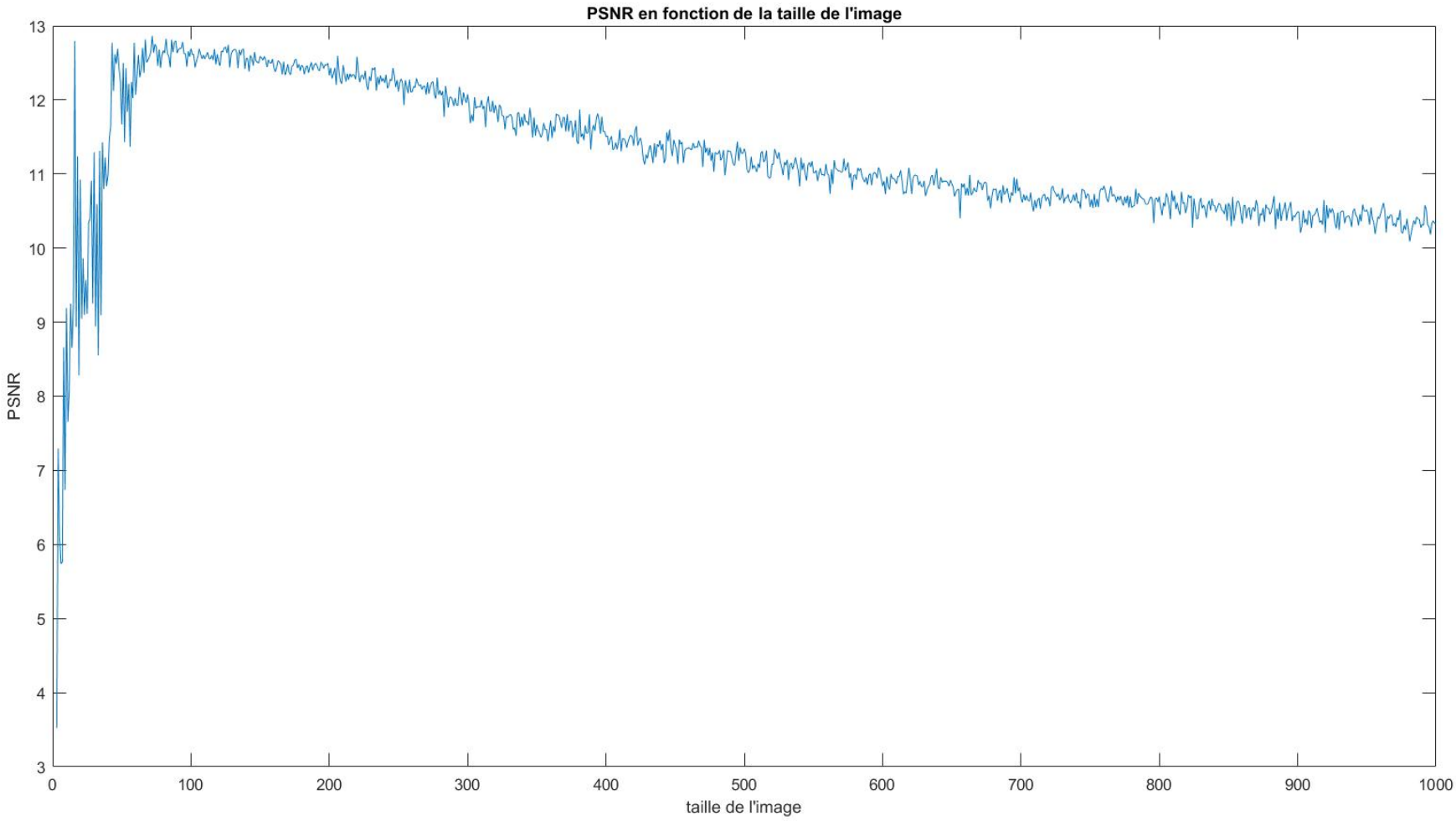
[9] Frigo Matteo, Johnson Steven G., *FFTW*, version 3.3.6-pl1 15 Janv. 2017 (http://www.fftw.org/fftw3.pdf)

V. Annexes

A. images PSNR :




PSNR en fonction du pas angulaire
PSNR
pas angulaire
13
12
11
10
9
8
7
6
0
1
2
3
4
5
6
7
8
9
10


PSNR en fonction de la taille de l'image
PSNR
taille de l'image
13
12
11
10
9
8
7
6
5
4
3
0
100
200
300
400
500
600
700
800
900
1000

B. Code Matlab

1) Launcher :


```
taille = 256;
pas = 1;
[I, R, xp] = radon_gen(taille,180,pas);
img = retroprojection_discrete(R,xp,taille,180,pas);
P = PSNR(I, img);
figure
colormap('gray');
imagesc(img);
title("Rétroprojection filtrée, taille = "+taille+", pas = "+pas+", PSNR = "+P)
m = 1000; %taille maximun de l'image
p_max = 15; %pas maximun

psnr_vect = zeros(1,m);
psnr_vect_bis = zeros(1,m);
for n = 1:m
    [I, R, xp] = radon_gen(n,180,1);%le p a 1
    img = retroprojection_discrete(R,xp,n,1);%le p a 1
    P = PSNR(I, img);
    P_bis = psnr(I, img);

    psnr_vect(n)=P;
    psnr_vect_bis(n)=P_bis;
end
figure;
plot(1:m,psnr_vect);
title("PSNR en fonction de la taille de l'image");
xlabel("taille de l'image");
ylabel("PSNR");
figure;
plot(1:m,psnr_vect_bis);
title("PSNR en fonction de la taille de l'image avec fonction native");
xlabel("taille de l'image");
ylabel("PSNR");



p_vect = [0.1:0.1:p_max];
psnr_vect2 = zeros(1,length(p_vect));
for p = 1:length(p_vect)
    truc  = p_vect(p)

    [I2, R2, xp2] = radon_gen(256,180,truc);%n a 256
    img2 = retroprojection_discrete(R2,xp2,256,180,truc);


    P2 = PSNR(I2, img2);

    psnr_vect2(p)=P2;

end
figure;
plot(p_vect,psnr_vect2);
```

```
title("PSNR en fonction du pas angulaire");
xlabel("pas angulaire");
ylabel("PSNR");


psnr_vect3 = zeros(m,p_max);

p_vect = [0.1:0.1:p_max];
m_vect = [10:10:m];
psnr_vect3 = zeros(length(m_vect),length(p_vect));

for n = 1:length(m_vect)
    m_vect(n)
   for p = 1 :length(p_vect)
       pas = p_vect(p);
       taille = m_vect(n);
       [I3, R3, xp3] = radon_gen(taille,180,pas);
       img3 = retroprojection_discrete(R3,xp3,taille,180,pas);
       P3 = PSNR(I3, img3);
       psnr_vect3(n,p)=P3;
   end
end


figure;
surf(p_vect,m_vect,psnr_vect3);
title("PSNR en fonction du pas angulaire et de la taille de l'image");
xlabel("pas angulaire");
ylabel("taille de l'image");
zlabel("PSNR");
figure;
imagesc(p_vect,m_vect,psnr_vect3);
colorbar;
title("PSNR en fonction du pas angulaire et de la taille de l'image");
xlabel("pas angulaire");
ylabel("taille de l'image");
zlabel("PSNR");


m_vect2 = [4:10];
for i = 1:length(m_vect2)
    m_vect2(i)=2^m_vect2(i);
end
p_vect2 = [-8:5];
for i = 1:length(p_vect2)
    p_vect2(i)=2^p_vect2(i);
end
psnr_vect4 = zeros(length(m_vect2),length(p_vect2));
for n = 1:length(m_vect2)
    m_vect2(n)
   for p = 1 :length(p_vect2)
       pas = p_vect2(p);
       taille = m_vect2(n);
```

```
        [I3, R3, xp3] = radon_gen(taille,180,pas);
        img3 = retroprojection_discrete(R3,xp3,taille,180,pas);
        P3 = PSNR(I3, img3);
        psnr_vect4(n,p)=P3;
    end
end


figure;
surf(p_vect2,m_vect2,psnr_vect4);
title("PSNR en fonction du pas angulaire et de la taille de l'image, avec taille ↙
2^n et pas 2^p")
xlabel("pas angulaire");
ylabel("taille de l'image");
zlabel("PSNR");
figure;
imagesc(p_vect2,m_vect2,psnr_vect4);
title("PSNR en fonction du pas angulaire et de la taille de l'image, avec taille ↙
2^n et pas 2^p")
colorbar;
xlabel("pas angulaire");
ylabel("taille de l'image");
zlabel("PSNR");
```

2) PSNR :


```
function [ P ] = PSNR( I,img )
%n = size(I)
%M = n;
%N = n;
[N, M] = size(I);
D = 0;
for (m = 1:M)
    for (n = 1:N)
        D = D+(I(m,n)-img(m,n))^2;
    end
end
D = D/(N*M);

%PSNR

P = 10*log10((255^2)/D);

end
```

3) radon_gen :


```
function [ I,R, xp ] = radon_gen( n, theta_max,p )
%transformer de radon
%n la taille de l'imge, theta_max l'angle maximun et p le pas angulaire
I = phantom('Modified Shepp-Logan',n);
%subplot(3,1,1);
% figure
% imshow(I);

theta = 1:p:theta_max; %nombre d'angle lors de la transformer de radon
[R,xp]=radon(I,theta);
%subplot(3,1,2);
% figure
% colormap('gray');
% imagesc(theta,xp,R);

end
```

4) retroprojection_discrete :


```
function [ img  ] = retroprojection_discrete( R,xp,n,theta_max,p )
%avec R la matrice de la transformer de radon, xp , n la taille de  l'image
%et p le pas angulaire
[Rx,theta_max_2]=size(R);
width = 2^nextpow2(Rx);
Rx =n;
Ry = n;
theta_vect = 1:p:theta_max;
%theta_vect = linspace(1,theta_max,p);

xp_offset = abs(min(xp))+1;

B = zeros(Rx,Ry);

proj_fft = fft(R, width);
%filter = ones(width,1);
filter = 2*[0:( width/2-1),width/2:-1:1]'/width;

% figure;
% plot(1:width,filter);
% title("Filtre de Ram-Lak");

for i = 1:length(theta_vect)%i = 1:theta_max

    filtered(:,i) = proj_fft(:,i).*filter;
end

proj = real(ifft(filtered));

for k = 1:length(theta_vect)
    Q = proj(:,k);
    rad = theta_vect(k)*pi/180;
    %rad = k*pi/180;

    for (x = 1:Rx)
        for (y = 1:Ry)
            u = ((x-Rx/2)*cos(rad)-(y-Ry/2)*sin(rad))+xp_offset;
            u1 = ceil(u);
            u2 = floor(u);

            if u1 == u2
                %B(y,x) = B(y,x)+Q(u1);
                B(y,x) = B(y,x)+proj(u1,k);
            else
                %q = Q(u2)+(u-u2)*((Q(u1)-Q(u2))/(u1-u2));%formule de toyalor young↙
pour interpoler la valeur de Q(u)
                q =proj(u2,k)+(u-u2)*((proj(u1,k)-proj(u2,k))/(u1-u2));
                B(y,x) = B(y,x)+q;

        end
    end
    end
img = B*(pi/theta_max);
```

```
img_min = min(min(img));
img_max = max(max(img));

for (i = 1:Rx)
   for (j = 1:Ry)
        img(i,j) = round((255/(img_max-img_min))*(img(i,j)-img_min));
    end
end
%figure
%colormap('gray');
%imagesc(img);
img;

end
```

C. Code C

D. main


```c
#include <stdio.h>
#include <stdlib.h>
#include "retroproj.h"
#include <math.h>
//#include "filtre.h"
#include<time.h>


int main(int argc, char const *argv[]) {


  double** R;
  double** B;


  R = chargement("./truc2.txt","info.txt");
  //afficherMatice(R,256,180);

  B = retroprojectionDiscrete(R,"info.txt");

  ecritMatrice(B,256,"./resultat.txt");

  return 0;
}
```

```
#include <stdio.h>
#include <stdlib.h>


double** chargement(char * , char *);
void afficherMatice(double**, int, int);
void liberationMem(double ** , int );
double** retroprojectionDiscrete(double** , char * );
void ecritMatrice(double **,int, char * );
```

```c
#include <stdio.h>
#include <stdlib.h>
#include "retroproj.h"
#include <math.h>

#define PI 3.14159265359

void afficherMatice(double** matrice,int Rx, int theta_max){
  for(int i =0;i<Rx;i++){
    for (int j =0; j<theta_max;j++){
      printf("%f\t",matrice[i][j]);
    }
    printf("\n");
  }
  printf("fin affichage\n");
}

double** chargement(char * radon, char * info){
  /*fonction permettant le chargement en memoire sous la forme d'un tableau de la
  transformer de Radon*/
  /*ne pas oublier de liberer la mémoire après les traitements*/
  FILE * fichier_info;
  fichier_info = fopen(info,"r");
  int Rx;
  int theta_max;
  int xp_offset;

  fscanf(fichier_info,"Rx = %d, theta_max = %d, xp_offset = %d\n",&Rx,&theta_max,&xp_offset);
  fclose(fichier_info);

  FILE * fichier_radon;
  fichier_radon = fopen(radon,"r");

  /* chargement de la matrice de la transformer de Radon dans un tableau local*/


  double R[Rx][theta_max];
  for(int i =0;i<Rx;i++){
    for (int j =0; j<theta_max;j++){
      fscanf(fichier_radon, "%lf\t",&R[i][j]);
      //printf("%lf\n", R[i][j]);
    }
    fscanf(fichier_radon,"\n");
  }

  /* creation d'un tableau alloer dynamiquement en meémoir pour pouvoir retourner la
  transformer de Radon*/

  double ** R_tab;
  R_tab = malloc((Rx)*sizeof(double));

  for(int u =0;u<Rx;u++){
    R_tab[u] = malloc((theta_max)*sizeof(double));
    for (int v =0; v<theta_max;v++){
      R_tab[u][v] = R[u][v];
    }
  }
  printf("chargement ok\n");
  return R_tab;
}

int chargementXp_offset(char* file){
  FILE * fichier_info;
  fichier_info = fopen(file,"r");
  int Rx;
  int theta_max;
  int xp_offset;

  fscanf(fichier_info,"Rx = %d, theta_max = %d, xp_offset = %d\n",&Rx,&theta_max,&xp_offset);
  fclose(fichier_info);
  return xp_offset;
```

```
}

void liberationMem(double ** R, int Rx){ /*erreur dans les free (revoir l'allocation de
mémoire, erreur dans l'allocation d'un pointeur ?)*/
  for(int i =0;i<Rx;i++){
    free(R[i]);
  }
  //free(R);
}

double** retroprojectionDiscrete(double** proj, char * info){

//chargement des differentes données
  FILE * fichier_info;
  fichier_info = fopen(info,"r");
  int Rx;
  int theta_max;
  int xp_offset;

  fscanf(fichier_info,"Rx = %d, theta_max = %d, xp_offset = %d\n",&Rx,&theta_max,&xp_offset);
  fclose(fichier_info);

  Rx = 256;
  int k, x, y, u1, u2;
  double u, q;
  double B[Rx][Rx];
  double rad;

  for(k=1;k<theta_max+1;k++){
    rad = (k*PI)/100;
    for(x=0;x<256;x++){
      for(y=0;y<256;y++){
        u=((x-Rx/2)*cos(rad)-(y-Rx/2)*sin(rad)+xp_offset);
        u1 = (int) ceil(u);
        u2 = (int) floor(u);

        if(u1 == u2){
          B[x][y] = B[x][y]+(proj[u1][k])*(PI/theta_max);
        }
        else{
          q = (proj[u2][k]+(u-u2)*((proj[u1][k]-proj[u2][k])/(u1-u2)))*(PI/theta_max);
          B[x][y] = B[x][y] +q;
        }
      }
    }
  }

  printf("fin retroproj\n" );

  double** B_tab;
  B_tab = malloc(Rx*sizeof(double));
  for(int u=0;u<Rx;u++){
    B_tab[u] = malloc(Rx*sizeof(double));
    for(int v=0;v<Rx;v++){
      B_tab[u][v] = B[u][v];
    }
  }
  return B_tab;
}


void ecritMatrice(double ** matrice,int Rx, char * file){

  double m_min = 300.00;
  double m_max =0.00;
  double m;
  int i,j;
  for( i =0;i<Rx;i++){
    for (j =0; j<Rx;j++){
      m = matrice[i][j];
```

```
      if (m_min>m && m>1){
        m_min = m;
      }
      if(m_max < m && m<255){
        m_max = m;
      }
    }
  }
  printf("m_min = %f, m_max = %f\n",m_min,m_max );
  for(i=0;i<Rx;i++){
    for(j=0;j<Rx;j++){
      matrice[i][j] = round((255/(m_max-m_min))*(matrice[i][j]-m_min));
    }
  }


  FILE * fichier_matrice;
  fichier_matrice = fopen(file,"w");

  for( i =0;i<Rx;i++){
    for (j =0; j<Rx;j++){
      fprintf(fichier_matrice, "%lf\t",matrice[i][j]);
    }
    fprintf(fichier_matrice,"\n");
  }
}
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <fftw3.h>

void transformer(double * , int , int , double *, double * );
double* inverse(double * , double * , int , int);
void filtre(double* ,double*, double *,double * , int, int);
void fitreRamLak(double *, double *, int, int);
double * miseForme(double ** , int , int );
double ** miseFormeInv(double * , int, int);
```

```c
#include <stdio.h>
#include <stdlib.h>
//#include<complex.h>
#include <math.h>
#include <fftw3.h>


void transformer(double * in, int Rx, int Ry, double *res_re, double * res_im){
  fftw_complex * spatial_repr;
  fftw_complex * freq_repr;
  fftw_plan plan;

  /*double * res_re;
  double * res_im;*/

  int i,j,x,y;

  spatial_repr = malloc(sizeof(fftw_complex)*Rx*Ry);
  freq_repr = malloc(sizeof(fftw_complex)*Rx*Ry);
  //printf("spatial_repr\n" );
  for(i=0;i<Rx*Ry;i++){
    spatial_repr[i][0] = in[i];
    spatial_repr[i][1] = 0;
    //printf("%f+i*(%f)\n",spatial_repr[i][0],spatial_repr[i][1] );
  }

  plan = fftw_plan_dft_2d(Rx,Ry,spatial_repr,freq_repr,FFTW_FORWARD,FFTW_ESTIMATE);
  fftw_execute(plan);

  /*res_re = malloc(sizeof(double)*Rx*Ry);
  res_im = malloc(sizeof(double)*Rx*Ry);*/

  //printf("freq_repr\n" );
  for(i=0;i<Rx*Ry;i++){
    res_re[i] = freq_repr[i][0];
    res_im[i] = freq_repr[i][1];
    //printf("%f+i*(%f)\n",freq_repr[i][0],freq_repr[i][1] );
  }

  for(j=0;j<Ry;j++){
    for(i=0; i<Rx;i++){
      x=i;
      y=j;
      if(i<Rx/2 && j<Ry/2){
        x=i+Rx/2;
        y=j+Ry/2;
      }
      if(i>=Rx/2 && j<Ry/2){
        x=i-Rx/2;
        y=j+Ry/2;
      }
      if(i<Rx/2 && j>=Ry/2){
        x=i+Rx/2;
        y=j-Ry/2;
      }
      if(i>=Rx/2 && j>=Ry/2){
        x=i+Rx/2;
        y=j+Ry/2;
      }
      printf("y*Rx+x = %d, j*Rx+i = %d\n",y*Rx+x,j*Rx+i );
      res_re[y*Rx+x]=freq_repr[j*Rx+i][0];
      res_im[y*Rx+x]=freq_repr[j*Rx+i][1];
      //printf("%f+i*(%f)\n",res_re[i],res_im[i]);
    }
  }

  fftw_destroy_plan(plan);
  fftw_free(spatial_repr);
  fftw_free(freq_repr);

}
```

```c
double* inverse(double * reIn, double * imIn, int Rx, int Ry){
  fftw_complex * spatial_repr;
  fftw_complex * freq_repr;
  fftw_plan plan;
  double * out;

  int i,j,x,y;

  spatial_repr = malloc(sizeof(fftw_complex)*Rx*Ry);
  freq_repr = malloc(sizeof(fftw_complex)*Rx*Ry);

  for(j=0;j<Ry;j++){
    for(i=0; i<Rx;i++){
      x=i;
      y=j;
      if(i<Rx/2 && j<Ry/2){
        x=i+Rx/2;
        y=j+Ry/2;
      }
      if(i>=Rx/2 && j<Ry/2){
        x=i-Rx/2;
        y=j+Ry/2;
      }
      if(i<Rx/2 && j>=Ry/2){
        x=i+Rx/2;
        y=j-Ry/2;
      }
      if(i>=Rx/2 && j>=Ry/2){
        x=i+Rx/2;
        y=j+Ry/2;
      }
      freq_repr[j*Rx+i][0]=reIn[y*Rx+x];
      freq_repr[j*Rx+i][1]=imIn[y*Rx+x];
    }
  }
  plan = fftw_plan_dft_2d(Rx,Ry,freq_repr,spatial_repr,FFTW_BACKWARD,FFTW_ESTIMATE);
  fftw_execute(plan);

  out = malloc(sizeof(double)*Rx*Ry);
  for(i=0;i<Rx*Ry;i++){
    out[i]=spatial_repr[i][0]/(Rx*Ry);
    //printf("%f\n",out[i] );
  }
  return out;
  fftw_destroy_plan(plan);
  fftw_free(spatial_repr);
  fftw_free(freq_repr);
}

void filtre(double* reImg,double*imImg, double *reFiltre,double * imFiltre, int Rx, int Ry){
  double a, b, c, d;
  int i;

  for(i=0; i<Rx*Ry;i++){
    a = reImg[i];
    b = imImg[i];
    c = reFiltre[i];
    d = imImg[i];

    reImg[i] = a*c-b*d;
    imImg[i] = b*c + a*d;
  }
}

void fitreRamLak(double * reFiltre, double * imFiltre, int Rx, int Ry){
  double * f;
  int i;
  f = malloc(sizeof(double)*Rx*Ry);
  for (i=0;i<Rx*Ry;i++){
    if(i<(Rx*Ry)/2){
```

```
      f[i] = i;
    }
    else{
      f[i] = (Rx*Ry)-i;
    }
    fftw_complex * filtre_av;
    fftw_complex * filtre_ap;
    fftw_plan plan;

    filtre_av = fftw_malloc(sizeof(fftw_complex)*Rx*Ry);
    filtre_ap = fftw_malloc(sizeof(fftw_complex)*Rx*Ry);

    plan = fftw_plan_dft_1d(Rx*Ry,filtre_av,filtre_ap,FFTW_FORWARD,FFTW_ESTIMATE);
    fftw_execute(plan);

    for(i=0;i<Rx*Ry;i++){
      reFiltre[i]=filtre_ap[i][0];
      imFiltre[i]=filtre_ap[i][1];
    }

  }
}

double * miseForme(double ** matrice, int Rx, int Ry){
  double * out;
  int i,j;
  out = malloc(sizeof(double)*Rx*Ry);
  for(j=0;j<Ry;j++){
    for(i=0;i<Rx;i++){
      out[i+j*Rx]=matrice[i][j];
    }
  }
  return out;
}

double ** miseFormeInv(double * matrice, int Rx, int Ry){
  double ** out;
  int i,j;
  out = malloc(sizeof(double)*Ry);

  for(j=0;j<Ry;j++){
    out[j]=malloc(sizeof(double)*Rx);
    for(i=0;i<Rx;i++){
      printf("(%d,%d)\n",j,i);
      out[i][j]=matrice[i+j*Rx];
    }
  }
  return out;
}
```