Rapport intermédiaire 2/2
Traitement des données audio-visuelles

SADURNI Thomas

Département Sciences du Numérique - Filière Image et Multimédia
2020-2021

Table des matières

Table des figures

1 Introduction

Bienvenue dans cette deuxième partie du rapport de l'ensemble des TP que nous avons pu faire au cours de cette année à l'ENSEEIHT en Images et Multimédia. Après avoir vu les modèles graphiques probabilistes et les principales méthodes variationnelles au cours des six premiers TPs, nous continuerons de manipuler les méthodes variationnelles dans les deux prochaines TPs en étudiant les contours actifs et la décomposition d'image. Nous verrons ensuite la notion de transformations d'images en s'intéressant à la tomographie, principe utilisé, entre autres, dans l'imagerie médicale. Dans l'UE *Traitement des données audio-visuelles*, il y a bien le mot *audio*. Nous aurons enfin l'occasion d'aborder le son dans les trois derniers TPs de l'année toujours en s'intéressant à la transformation de données. Compression audio, reconstitution simplifié de la très connue application de reconnaissance de musique *Shazam* et enfin la séparation de source.
En général, la majorité du code des TPs était fournie, nous devions implanter les fonctions principales comprises à l'intérieur. L'objectif ici n'est pas de détailler le code mais de discuter des résultats et d'observer l'influence des paramètres. Chaque TP comprend des exercices obligatoires (entre 1 et 3) et un dernier facultatif.

2 Suite et fin des méthodes variationnelles

Dans cette première partie du second rapport, nous allons continuer notre étude des méthodes variationnelles. Nous avions vu la segmentation par classification, la détection d'objets dans une image, la restauration d'image ou encore le photomontage. Ici, nous nous pencherons sur les contours actifs et la décomposition d'image.

2.1 Contours actifs

Un modèle de contour actif, aussi appelé *snake* est une structure dynamique du traitement d'images utilisé en vision artificielle principalement et introduit par l'équipe Kass, Witkin et Teropoulos. Ce modèle est formé d'une série de points placés autour d'un objet d'intérêt d'une image, et représentant une courbe fermée ou non et qui va tenter de se déplacer pour épouser la forme de l'objet en question. L'idée est de déplacer les points pour se rapprocher des zones de fort gradient en conservant la courbure du contour et la répartition des points.
Une itération de l'algorithme calcule l'énergie interne, qui dépend des points du contour (régularité de la courbure, espacement des points ...) et l'énergie externe, qui correspond à l'adéquation aux contours et pénalise les faibles gradients du niveau de gris. On souhaite dans notre cas forcer la fermeture de la courbe initiale et obtenir une courbe qui épouse les contours de l'objet. La difficulté réside dans le choix des termes de régularisation à donner pour les énergies, le poids de la pénalisation de la longueur et de la courbure.

Le premier exercice nous propose d'afficher le champ de forces externes correspondant à l'énergie externe où celle-ci est calculé en appliquant un filtre gaussien à l'image avant de calculer son gradient. Dans les deux images de la figure 1, on observe les résultats pour les deux énergies, respectivement sans et avec le filtrage sur l'image.

Rapidement, on remarque que le champ de forces est plus cohérent dans la seconde image, c'est-à-dire celle qui à subit le filtrage gaussien. En effet, dans la première image, les champs de forces sont désordonnés, et ne sont pas attirés vers les contours des pièces de l'image. L'application du filtre corrige ce problème et les champs de force situés sur et autour des pièces pointent bien vers les contours. Loin des contours de des pièces, l'énergie est très faible ce qui fait qu'un *snake* mal initialisé pourrait ne pas bouger. Le principe de contours actifs est très utilisé en imagerie médicale notamment pour l'IRM. On observe les mêmes résultats sur une image d'IRM d'un cerveau (figure 2).

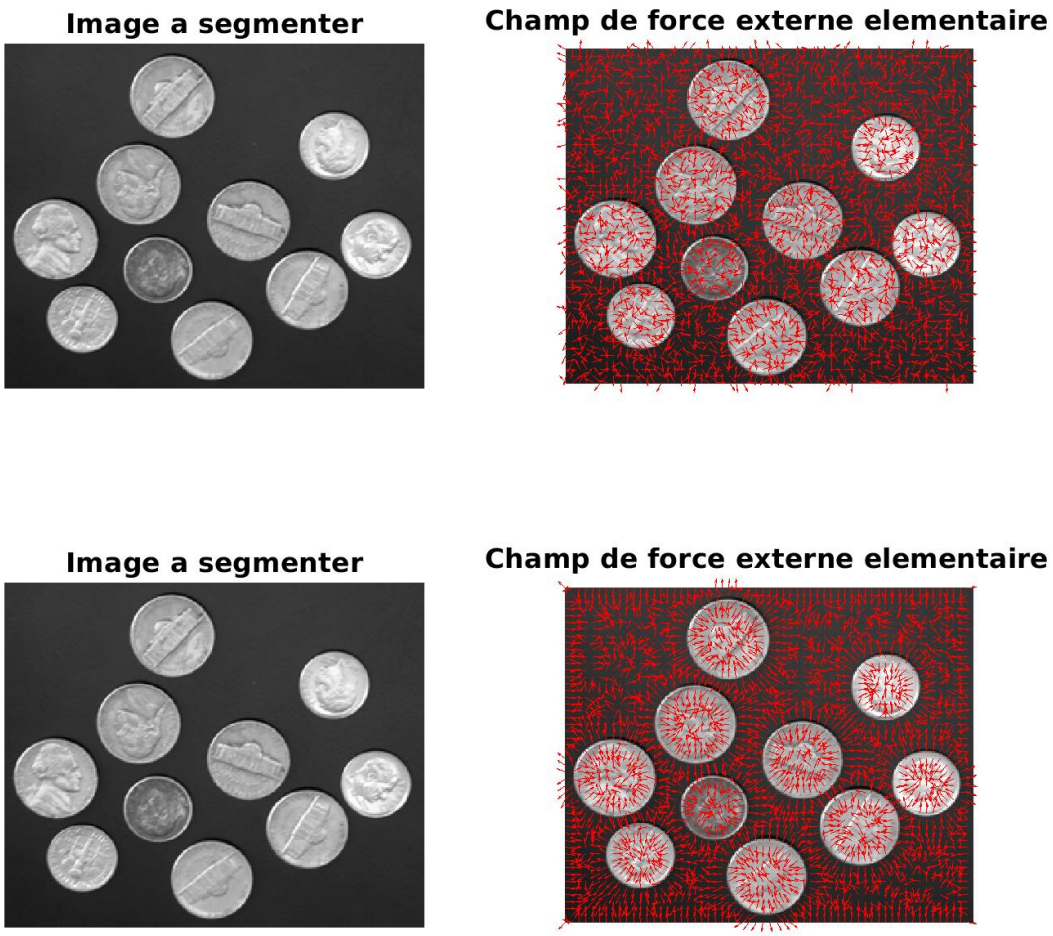

FIGURE 1 – Résultats des champs de force sans et avec l'application d'une filtre gaussien sur l'image

En ce qui concerne les bords de l'image, il pourrait être pratique d'appliquer une force aux bords qui ramènerait le snake vers l'intérieur de l'image.

Comme vous l'aurez bien compris, l'objectif est de contourer des objets ou des parties de l'image, les pièces dans l'une, la tumeur dans l'autre. Le deuxième exercice implémente l'algorithme du contour actif. Celui-ci consiste, de manière itérative, à calculer la discrétisation de la courbe à partir de l'énergie externe pour la faire converger vers les contours de la pièce ou de la tumeur.
Sans toucher les paramètres initiaux, on obtient la figure 3. Le résultat est très correct mais le nombre d'itérations paraît assez élevé pour la première image (j'ai diminué le paramètre *alpha* pour la seconde) et les résultats diffèrent un peu en fonction de la pièce en raison des contrastes entre celles-ci et le fond de l'image.

On va maintenant s'intéresser à la détection de la tumeur sur l'image de l'IRM. La série d'images qui suit montre aussi l'influence des paramètres sur les résultats. Si on choisit *alpha* trop faible, il n'y a pas d'activité, la courbe initiale ne bouge pas car le terme de régularisation est quasi nul. Au contraire, si *alpha* est trop important le résultat est inexploitable et forcément incorrecte, le *snake* se recroqueville sur lui-même (figure 4).

Revenons sur des valeurs cohérentes et observons en fonction de la largeur de la courbe initiale.

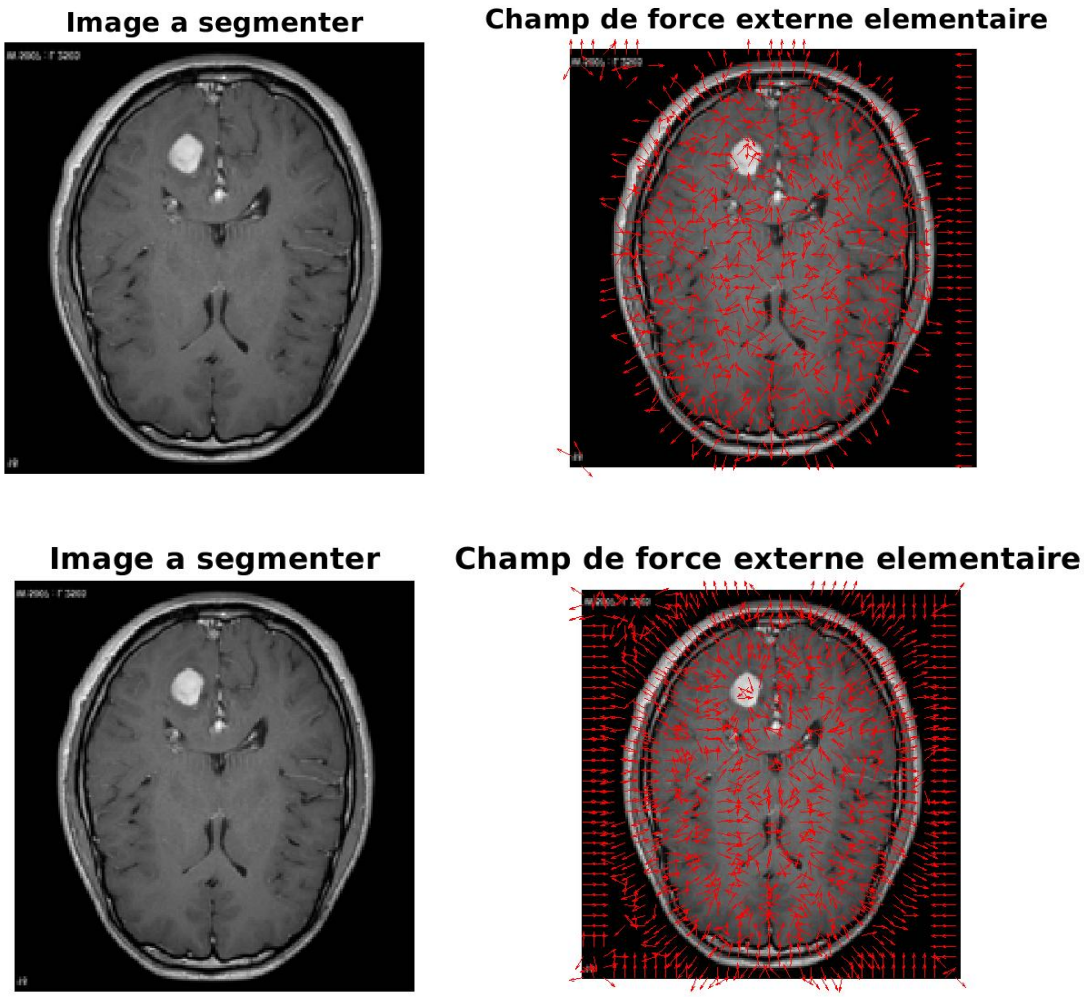

FIGURE 2 – Résultats des champs de force sans et avec l'application d'une filtre gaussien sur l'image IRM

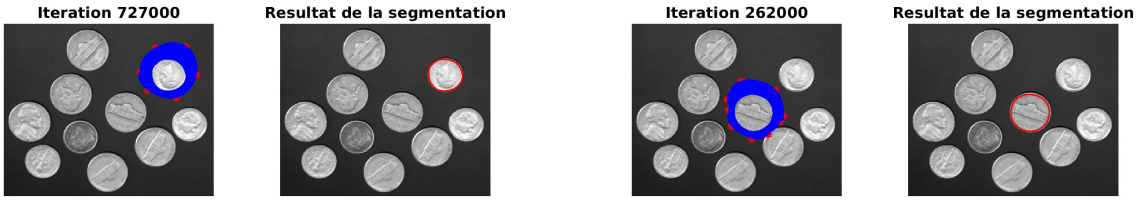

FIGURE 3 – Résultat du contour actif sur des pièces.

En trouvant des paramètres corrects, la convergence se fait bien au niveau de la tumeur. En revanche, le contour n'est pas très bien réalisé car on a l'impression qu'il traverse l'objet (la tumeur ici).

Pour pallier ce problème, une solution consiste à faire diffuser l'énergie vers les contours. C'est l'objet de l'exercice facultatif du TP. On prolonge les champs de forces là où ils sont faibles on affiche le champ de forces externe GVF, ce qui revient à résoudre des équations d'Euler-Lagrange (figure 7). On s'intéresse ici à une image non filtrée, contrairement aux exercices précédents.

Les résultats sont bien meilleurs sur l'image de l'IRM, on peut même essayer de segmenter une

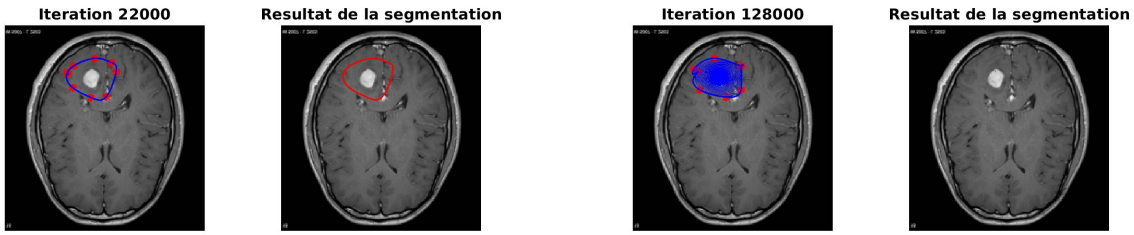

FIGURE 4 – Résultat du contour actif sur l'IRM avec des *alpha* incohérents

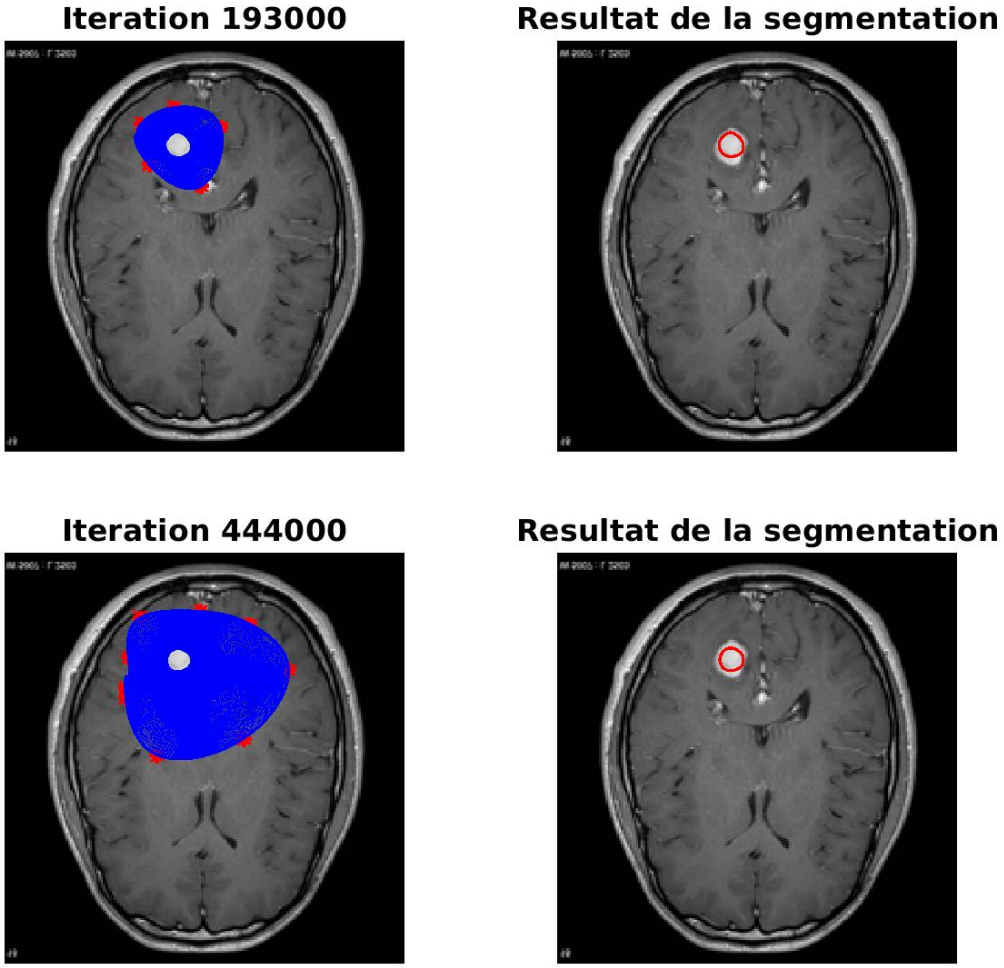

FIGURE 5 – Résultat du contour actif sur l'IRM avec des zones initiales différentes

image très complexe pour observer l'efficacité du contour actif. Les images de la figure 8 affichent la segmentation de plusieurs fruits. Le calcul est un peu long mais le résultat très satisfaisant.

Les méthodes d'obtention de l'équation d'évolution du contour actif sont variées mais nous nous intéressons essentiellement à des méthodes variationnelles, qui consistent en la minimisation d'un critère. Ainsi, l'algorithme des contours actifs peut être très efficace si les paramètres sont bien choisis, ce qui n'est pas une mince affaire. Il est aussi pertinent lorsqu'il s'agit d'isoler des formes convexes régulières et idéales, pour des images plus complexes avec des formes et des concavités *strictes*, il faudra travailler davantage sur les paramètres de régularisation de la forme. Nous avons manipulé seulement des images, il serait intéressant de voir ce qu'il est possible de faire sur des vidéos en utilisant le même principe.

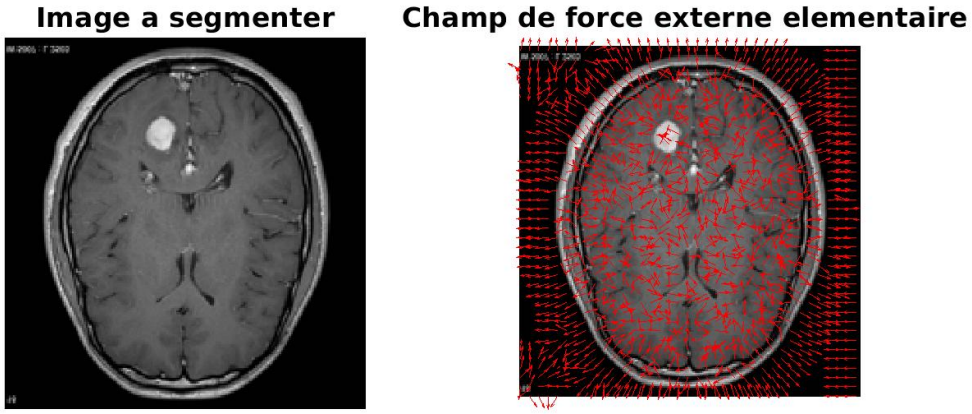

FIGURE 6 – Nouveau champ de force pour l'image IRM

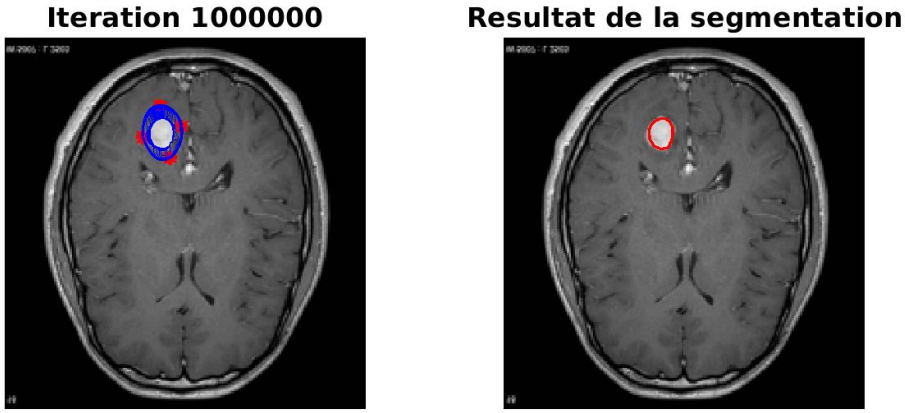

FIGURE 7 – Nouveau contour actif pour l'image IRM

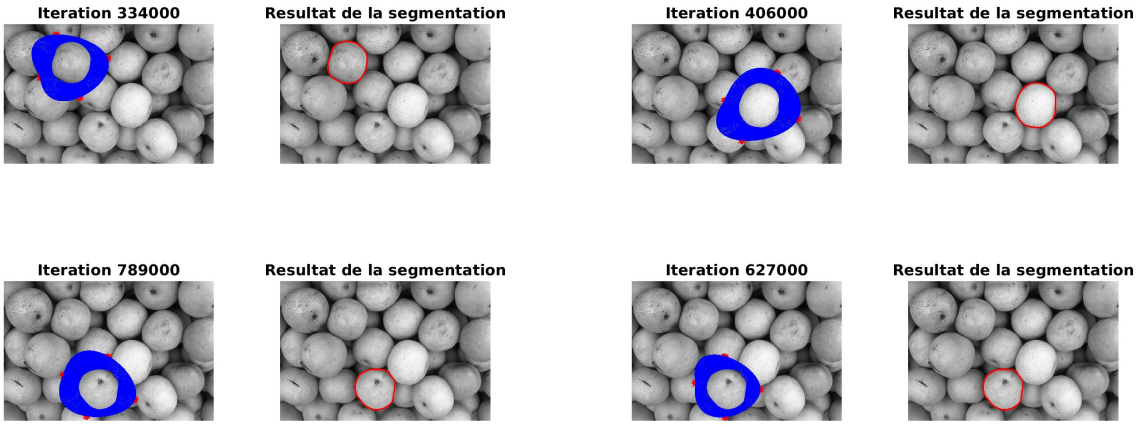

FIGURE 8 – Résultat du contour actif sur différents fruits

2.2 Décomposition d'une image

La dernière partie des méthodes variationnelles concerne la décomposition d'image. Ainsi, pour une image u donnée, on cherche une paire d'images $(\bar{u},\bar{u_c})$ de même dimension que u telles que

$u = \bar{u} + \bar{u_c}$. Pour résoudre un tel problème, on utilise un a priori pour garantir l'unicité de la solution. Nous avions déjà rencontré un problème de décomposition d'image au TP5 avec le débruitage d'une image u. Ici, nous nous intéressons à la décomposition structure + structure d'une image, qui est l'exemple le plus courant dans la décomposition d'image. La structure contient plutôt les basses fréquences alors que les hautes fréquences se retrouvent généralement dans la texture. Cette décomposition s'appelle aussi *cartoon texture* car en bande dessinée les contours des personnages et des objets représentent la texture alors que les aplats de couleurs constituent la structure.
La structure $\bar{u}$ doit être assez proche de u mais ses contours doivent être adoucis. Pour les images numériques, c'est la transformée de Fourier discrète qui est le plus adaptée. L'intérêt vient du fait que la valeur d'un pixel de l'image a une influence sur la totalité du spectre. Trois étapes surviennent donc pour la décomposition d'une image : calcul du spectre de l'image par transformée de Fourier, modification du spectre, et calcul de la transformée de Fourier inverse du spectre.

L'exercice initial nous propose de tester les trois exemples de modification du spectre d'une grille visible sur la figure 9.

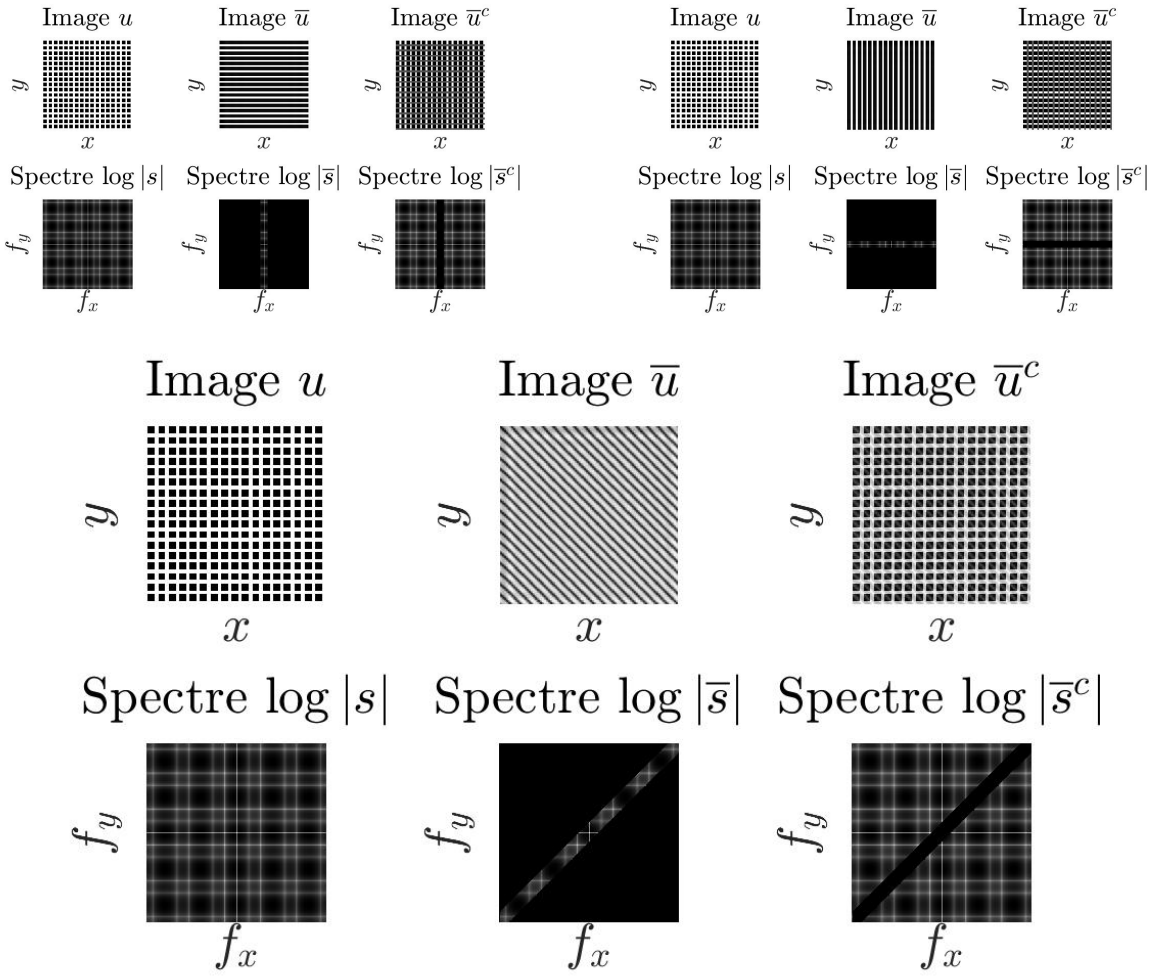

FIGURE 9 – Exemples de trois modification du spectre d'une grille

Nous allons maintenant décomposer en structure + texture de l'image *Barbara*. Il faut savoir que la fréquence des textures dépend fortement de l'image et de sa taille. Pour avoir un résultat convenable, on n'effectue pas de partition franche entre les pixels du spectre (figure 10), mais une partition douce qui affecte à chaque pixel un poids d'autant plus élevé que sa distance à l'origine est faible (figure 11).

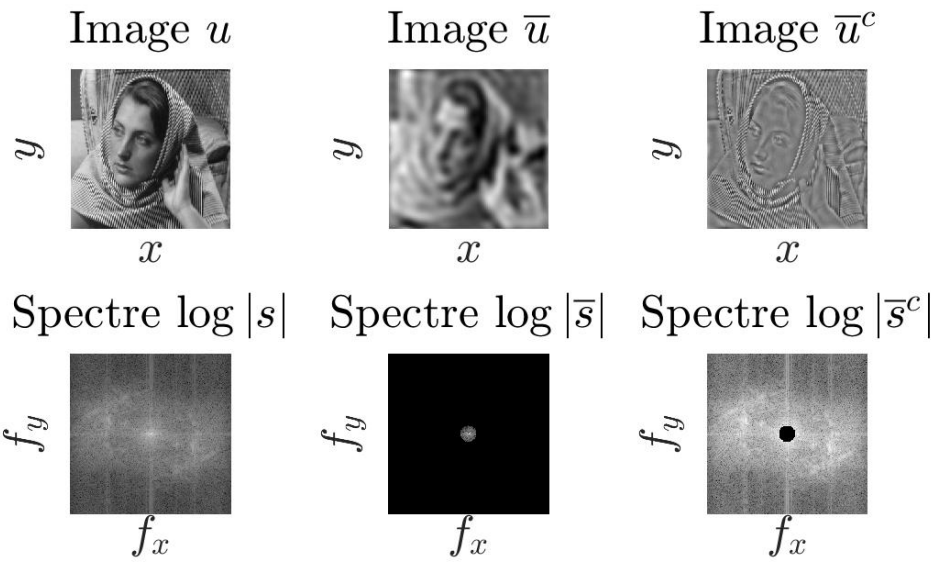

FIGURE 10 – Décomposition par partition $franche$

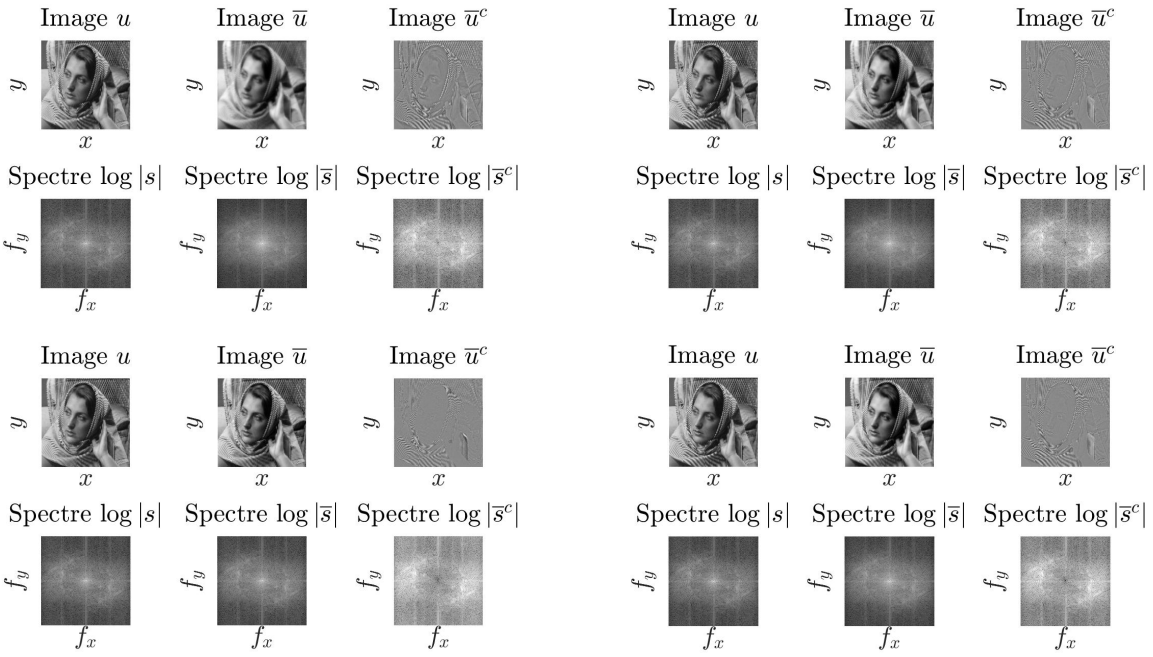

FIGURE 11 – Décompositions par partition *douce* avec *eta* de plus en plus grand

Nous pouvons remarquer que les résultats sont plus convaincants car nous avons lissé la séparation entre les fréquences de la partie *cartoon* et celles de la partie texture. Mais la décomposition d'une image peut aussi être réalisée par méthode variationnelle, sans recouvrir à la TFD. Ce modèle est appelé *ROF* et fait apparaitre la variation totale. Les résultats sont bien meilleurs et visibles sur la figure 12.

Comme dit précédemment, nous attendons que l'image décomposée soit proche de l'image initiale avec des variations de niveau de gris moins brutales. L'algorithme itératif est expliqué en détail dans l'énoncé du TP. Nous pouvons aussi nous intéresser à des images en couleur, en voici quelques-unes.

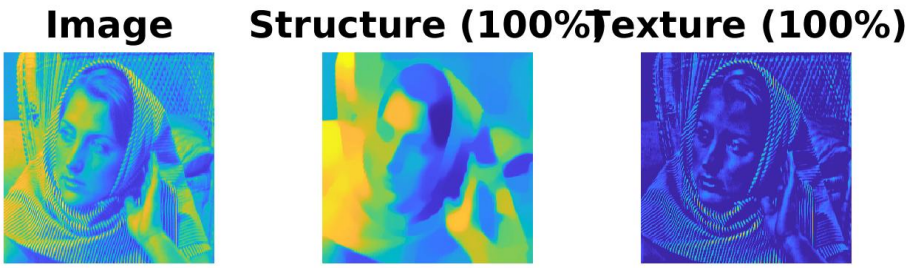

FIGURE 12 – Décomposition par partition *franche*

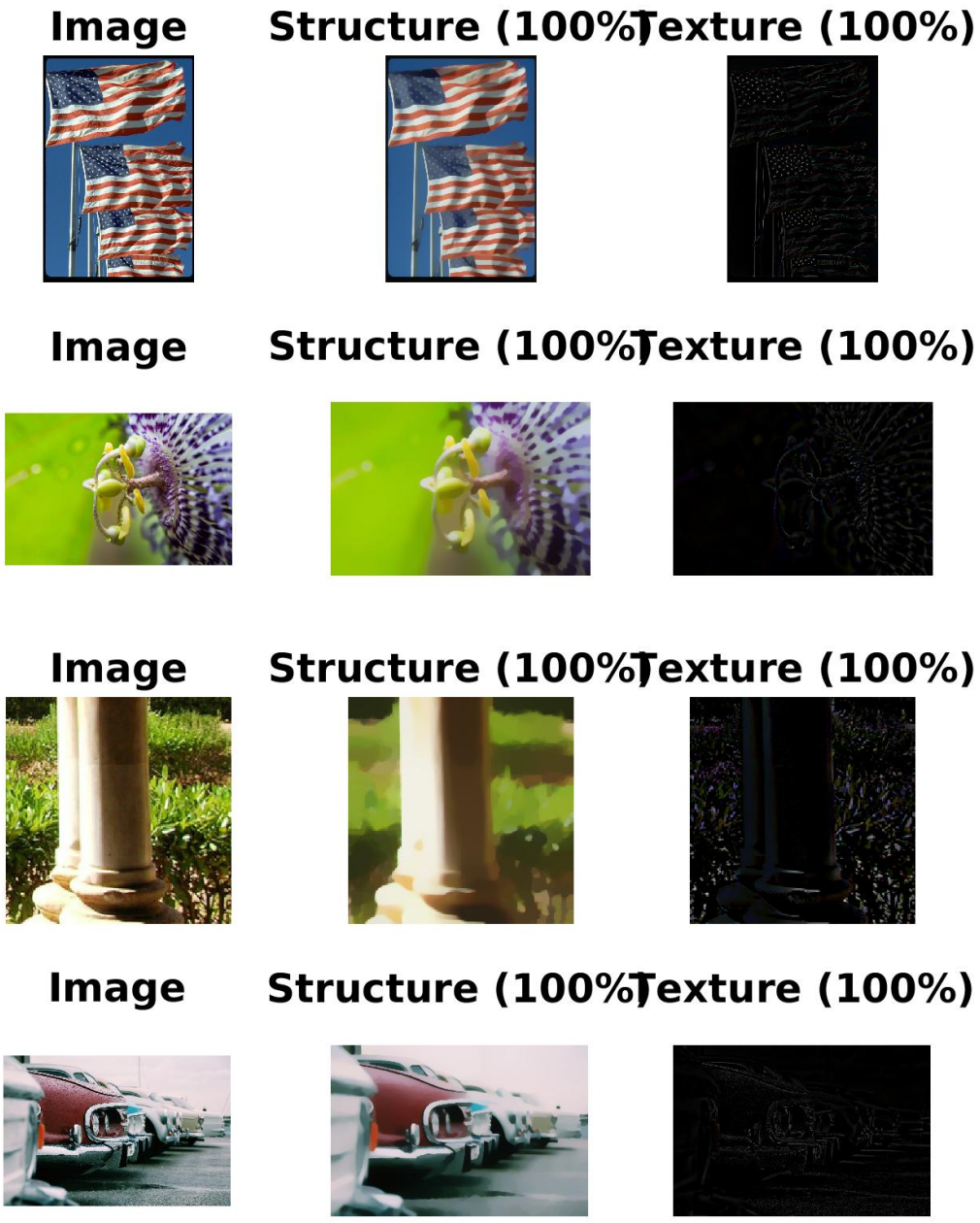

FIGURE 13 – Décompositions sur plusieurs images couleurs avec le modèle ROF

Évidemment, le temps de calcul est très important, quand les images que j'ai choisies sont assez

volumineuses, mais les résultats proposés sont propres et les différences sont claires.
Dans la dernière partie du TP, qui est facultative, on applique un modèle variationnel qui diffère du précédent car il fait intervenir la transformée de Fourier du signal à retrouver. Ce modèle *TV-Hilbert* est mixte car le terme d'attache aux données dépend des spectres de u et de $\bar{u}$. Or, comme la transformation de Fourier est non linéaire, on doit mettre en place une descente de gradient pour résoudre le problème. On obtient les images de la figure 14.

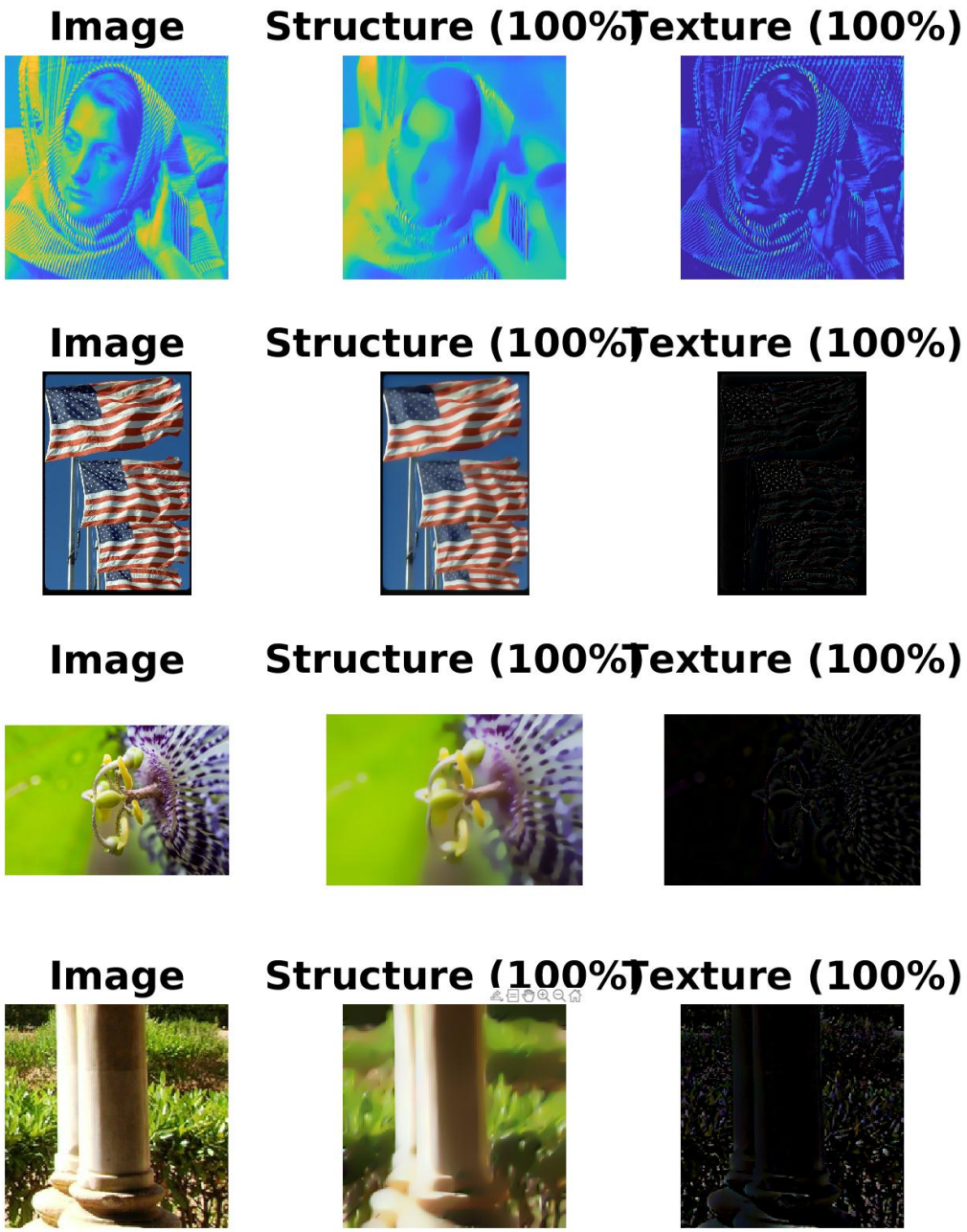

FIGURE 14 – Décompositions sur plusieurs images couleurs avec le modèle TV-Hilbert

Il est vrai que la différence est assez peu visible...
La décomposition d'image est un problème classique du traitement d'images et utile pour bien comprendre son fonctionnement. On pourrait s'intéresser à la décomposition des ombrages, qui dépend du relief et de l'éclairage, mais il me semble que nous nous y attarderons en troisième année.

3 Transformations

Nous attaquons maintenant la dernière partie de ces TP de *Traitement des données audio-visuelles* qui concerne les transformations. Nous verrons d'abord les transformations sur les images avec la tomographie, utilisée dans l'imagerie médicale, puis nous nous attarderons sur l'audio, avec la compression MP3, l'implémentation de *Shazam* simplifié et enfin la séparation de sources.

3.1 Tomographie

La tomographie est une technique d'imagerie qui permet de mesurer en trois dimensions le volume d'un objet à partir de mesures effectuées en dehors de cet objet. Elle est utilisée en imagerie médicale, en sismographie ou en mécanique des matériaux. Nous verrons ici la transformation de Radon d'une image, modèle mathématique des mécanismes tomographiques qui consistent à projeter l'image sur une droite avec un angle polaire. Le but de la tomographie est de reconstruire l'image à partir d'une série de transformées de Radon obtenue par la prise de mesure sur plusieurs angles. Les rayons X permettent cela pour le corps humain. Le but ici est de retrouver l'image à partir de son sinogramme, une image regroupant les données de l'image f discrète. Pour cela, nous implantons l'algorithme de Kaczmarz qui résout le système à partir du sinogramme et de la distance de chaque rayon à travers chaque pixel, tous deux sont fournis. Voici les résultats pour différents nombres de parcours de l'ensemble des équations noté k_{max}.

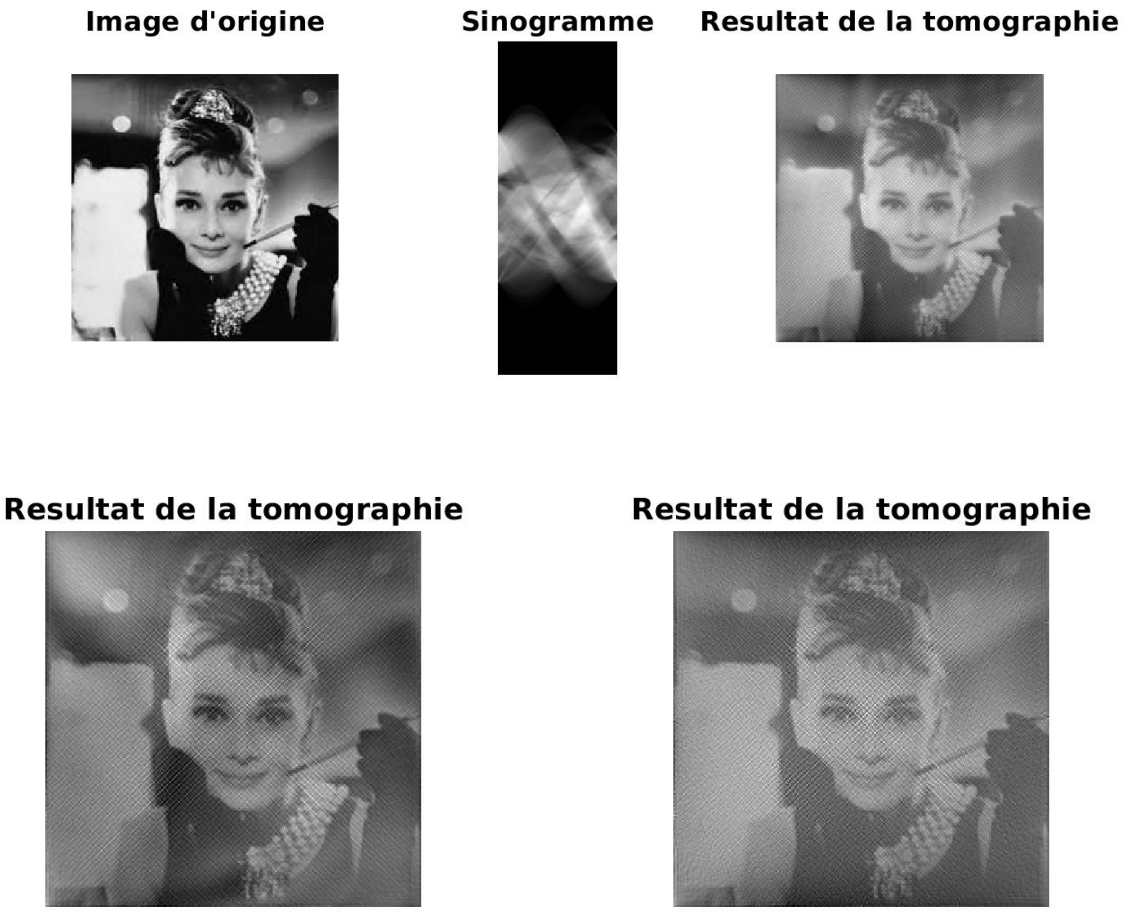

FIGURE 15 – Résultats de la tomographique pour k_{max} valant respectivement 1, 5 et 20

Sur la figure 20, l'image la plus nette est celle pour k=5, ce qui n'est pas surprenant car les calculs n'ont pas besoin d'être fait 20 fois chacun, car à force ils seront modifiés. On remarque aussi qu'une seule itération par donnée est suffisante pour obtenir une image visible. Le principal problème de cet algorithme est le temps d'exécution. En effet, il faut plusieurs dizaines de secondes sur les PC de l'ENSEEIHT pour avoir le résultat pour un nombre de parcours supérieur à 10. Nous allons donc voir dans la prochaine partie de ce sujet une autre approche pour restituer une image à partir de son sinogramme. Celle-ci consiste à dé-projeter les données.

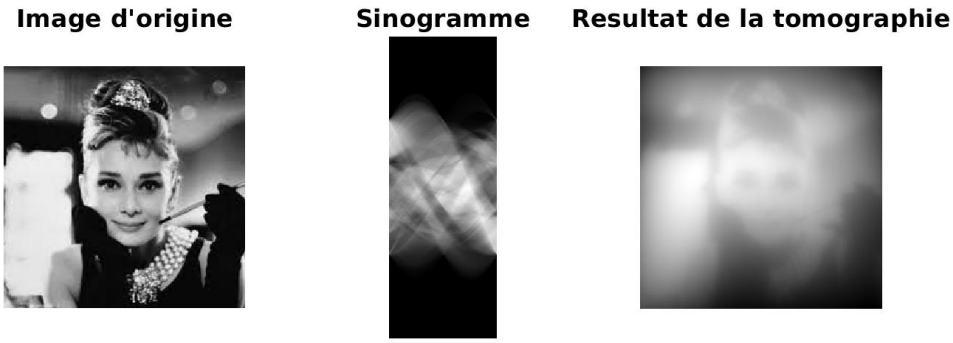

FIGURE 16 – Reconstruction de l'image à partir du sinogramme

Les résultats de la figure 16 sont visiblement flous, car les hautes fréquences sont moins représentées dans le sinogramme que les hautes fréquences. Pour éviter ce problème, l'idée est d'appliquer un filtre aux colonnes du sinogramme pour rétablir l'équilibre entre les basses et hautes fréquences. L'algorithme de reconstruction est fait par la méthode directe de Fourier. Nous calculons d'abord la transformation de Fourier 1D de chaque projection par fft dans le domaine de fréquence, on applique ensuite le filtre (dans notre cas, nous utilisons le filtre de Ram-Lak qui est simplement à la valeur absolue de la fréquence), puis on calcule la transformée inverse pour obtenir l'image reconstruite et enfin on rétroprojette les projections filtrées pour obtenir l'image $f(x, y)$.

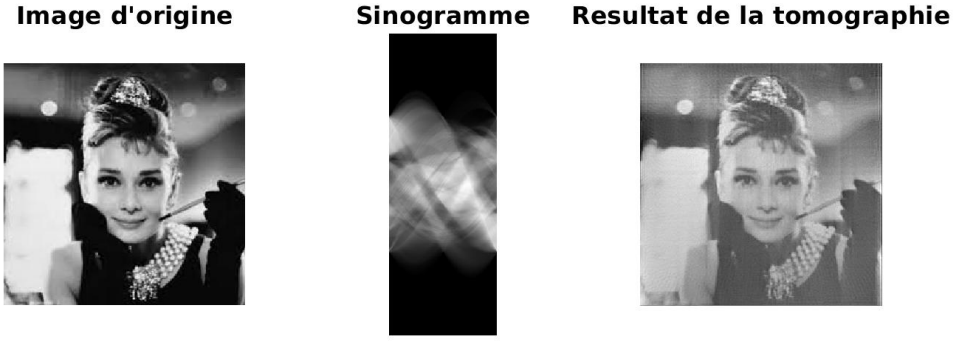

FIGURE 17 – Reconstruction filtrée de l'image à partir du sinogramme

Le résultat est ici bien meilleur. Le choix du filtre peut être une étape importante de l'algorithme de rétroprojection. La qualité de l'image est directement impactée par le choix du filtre. Nous avons choisi d'utiliser le plus simple de filtres, celui de Ram-Lak, mais il en existe d'autres : le filtre rampe,

le filtre Hann, le filtre cosinus etc.

Nous avons étudié dans ce TP le problème de reconstruction de l'image à partir de ses données de projection en géométrie parallèle. Nous avons présenté deux méthodes analytiques pour répondre à ce problème : méthode du filtrage de la rétroprojection et méthode de rétro-projection des projections filtrées, l'une plus efficace que l'autre.

3.2 Compression audio

Nous avons passé neuf TP à traiter des images, il est temps de passer aux applications audio. Au cours des trois prochains TP, nous aurons l'occasion de voir, entre autres, la compression audio. De manière générale, la compression audio se passe en deux étapes :

1. les prétraitements : décorrélation, transformations (Fourier pour le son, cosinus pour les images, ondelettes pour les films), changement d'espace colorimétrique etc
2. le codage : entropique ou algorithmique.

C'est le codage qui effectue la compression, mais les deux sont nécessaires à la réduction de la taille des données car le prétraitement permet d'accroître le taux de compression. La compression est très utile car elle permet un gain de place et de temps de transmission considérable quand il s'agit de film hollywoodien qui peuvent peser jusqu'à 8 To.

Il existe deux types de compression, avec ou sans perte. Dans ce TP, nous allons nous intéresser à la compression avec perte avec notamment le format MP3, mondialement connu dans le son. Cette compression avec perte est adaptée aux données audiovisuelles car les données peuvent être gigantesques et nos capacités auditives (et perceptuelles) sont limitées entre 20 Hz et 20 kHz, d'où l'intérêt des transformations car elles permettent de séparer les hautes et basses fréquences. Cette séparation a notamment été parcouru lors du TP sur la décomposition texture/structure. Nous allons nous intéresser à la transformation de Gabor d'un signal acoustique. Elle consiste à obtenir le spectre instantané du signal en multipliant ce dernier par une fenêtre glissante. Elle transforme la fonction réelle en une fonction complexe bidimensionnelle. Enfin, la majeure partie de l'information acoustique réside dans les basses fréquences, on représente alors le signal sonore en temps-fréquence par un sonagramme. Il s'agit de couper les hautes fréquences pour réduire en partie la taille du fichier.
Dans un premier temps, on effectue la transformée de Gabor avec une fenêtre glissante dont l'image partitionne le temps d'enregistrement. On obtient ainsi une image représentant le son passé en paramètre. Voici deux exemples dans la figure 18, le premier pour le son vocal 0380 et pour le son de Beethoven.

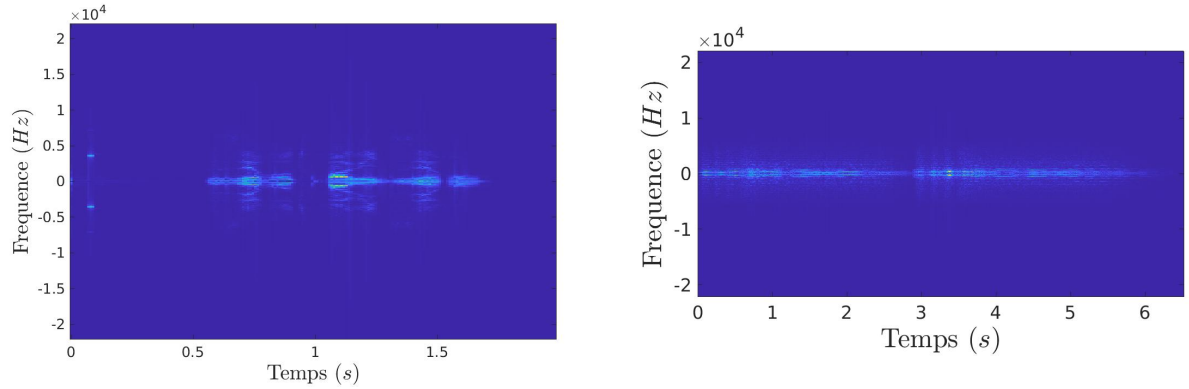

FIGURE 18 – Transformée de Gabor sur le son vocal 0380 et Beethoven

On remarque ici que le spectre est quasiment symétrique par rapport à l'axe des abscisses. Avec le script *restitution_1* donné, on peut écouter le son restitué sans distorsion et avec seulement la partie réelle. Ainsi, après suppression de la partie imaginaire du spectre le son, est certes de moins bonne qualité car un peu haché et perturbé mais il reste audible, de même lorqu'on supprime la phase, le son reste audible mais on entend un son de robot. Une autre façon de compresser sans perdre la partie imaginaire ou la phase de la transformée de Gabor est de supprimer les coefficients de Fourier des fréquences négatives et des hautes fréquences car la capacité auditive moyenne d'un humain est inférieure à 20 kHz, et il est impossible pour l'oreille de distinguer des fréquences au-dessus de 4000 Hz. Le deuxième exercice permet donc de faire cela avec une certaine proportion. Plus cette proportion est élevée, plus le son est de bonne qualité mais moins le taux de compression est important. En effet, si on dépasse une proportion de 0.5, le son perd de la qualité, et le fichier est plus volumineux que l'origine, inutile donc. Des valeurs cohérentes de la proportion, pour un son audible et une taille de fichier suffisamment réduite est autour de 0.25, pour un taux de compression autour de 4.

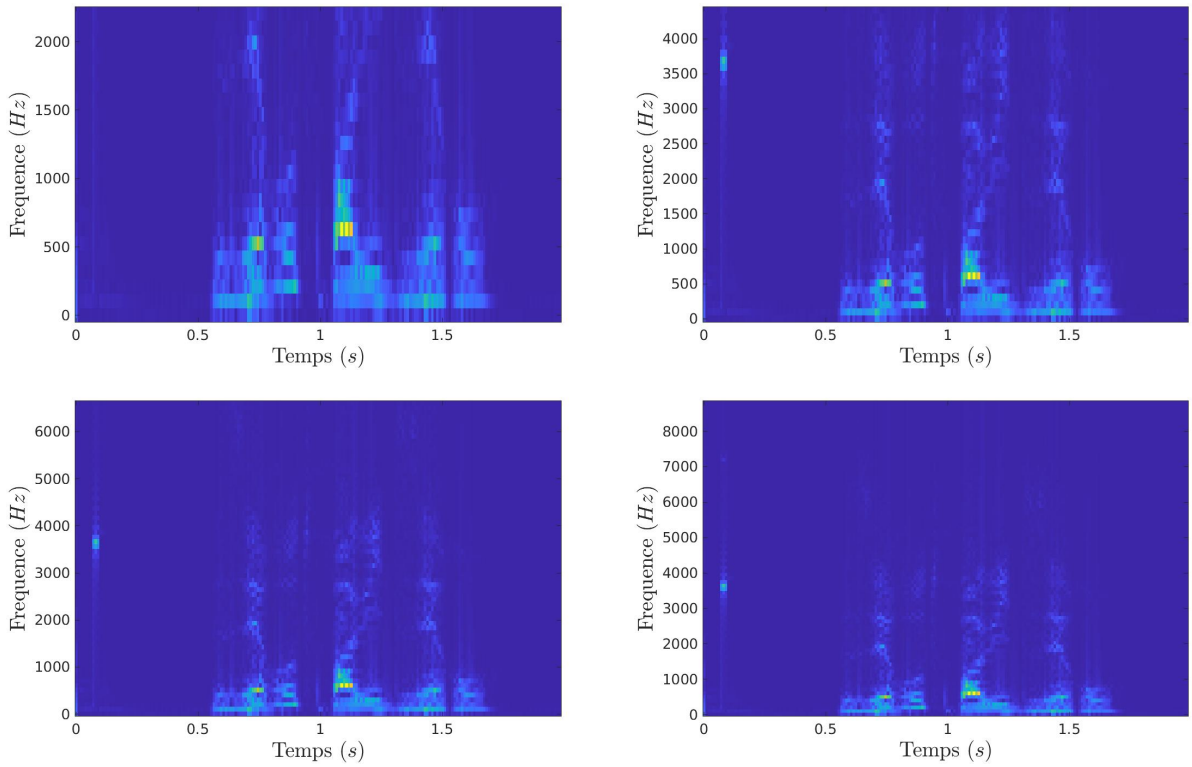

FIGURE 19 – Sonagramme du signal original avec une proportion respective de 0.1 0.2 0.3 et 0.4 pour le son 0380

Pour une proportion égale à 0.1, le taux de compression est élevée, autour de 10 fois moins que la taille originale, il va de soi que le son est de très mauvaise qualité. Le son est faible, très bruité et assez peu audible. Inversement, avec un taux de compression faible (réduction de moitié), le son est audible et clair, assez peu modifié par rapport à l'original. Il faut donc trouver un juste équilibre entre la qualité du son et le taux de compression. On va donc maintenant s'intéresser à la compression acoustique, qui diffère peu de ce que nous avons vu jusqu'à présent. Il consiste à conserver qu'une faible proportion des plus grands coefficients de Fourier. C'est l'objet du troisième exercice.

En testant différentes valeurs de proportion et de fréquence à conserver (figure **??**), on arrive à un son audible (certes robotique mais audible) et une compression audio d'un facteur 10, ce qui est surprenant ! C'est le principe de la compression MP3 que nous connaissons tous, en version simplifiée car nous n'avons pas effectué le codage.

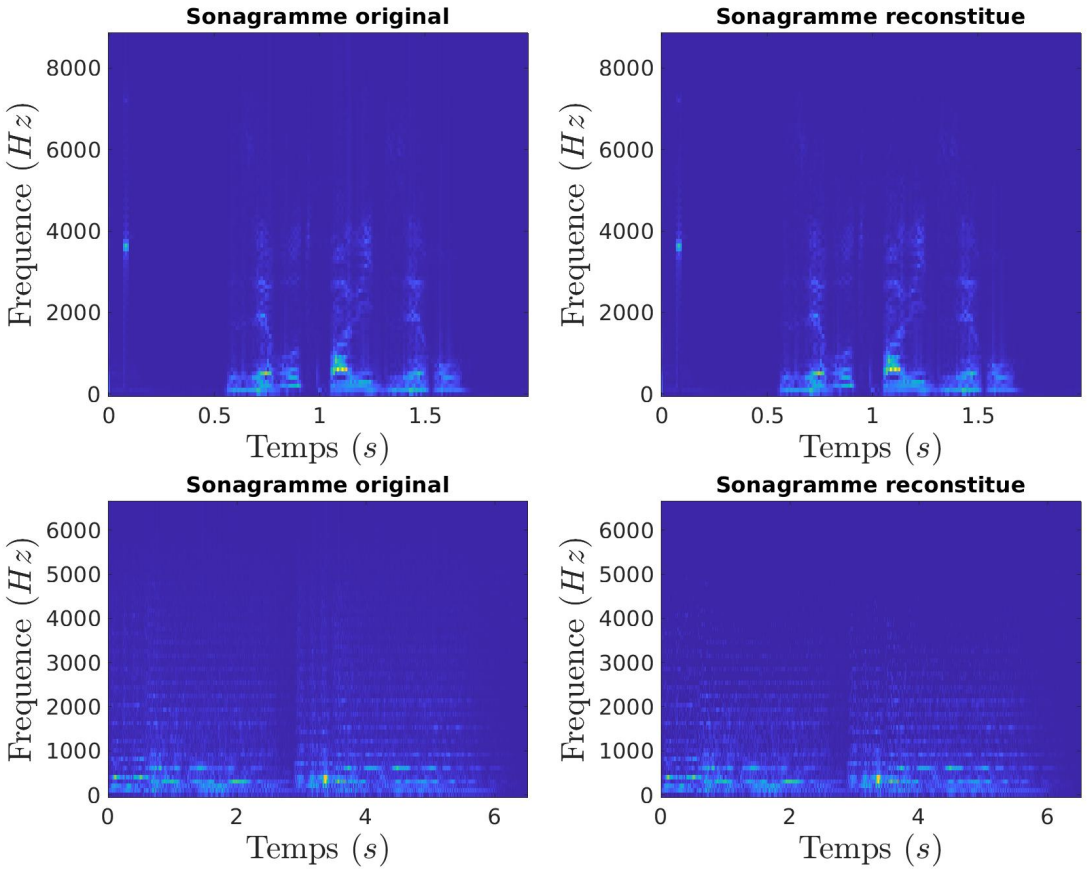

FIGURE 20 – Comparaison des sonagrammes originaux et reconstitués pour les sons *parole* et *Beethoven*

3.3 Reconnaissance musicale

La reconnaisse automatique de signaux acoustiques est utilisée le plus souvent dans la commande vocale, la dictée vocale, ou la reconnaissance musicale depuis les années 2000. Dans la continuité de l'étude précédente sur la compression audio et les empreintes sonores, nous allons nous intéresser à la reconnaissance musicale comme peut le faire l'application *Shazam* et, plus récemment, *Hum to search* de Google.
Nous avons produit un sonagramme d'un signal acoustique échantillonné à la fréquence d'échantillonnage standard et qui contient des coefficients de Fourier afin de réduire la quantité d'informations sans trop alléger le signal. En revanche, pour la reconnaissance musicale, c'est encore trop. Il faut encore réduire la dimensionnalité du signal afin d'alléger le stockage et le traitement des données. Dans le cas de *Shazam*, on extrait du sonagramme un nuage de point 2D. Dans l'exercice 1, on passe de 17600 valeurs complexes à 96 valeurs réelles, figure 21. Cette empreinte sonore est, comme son nom l'indique, empreinte d'un morceau de musique et suffit, en général, à identifier le morceau.

Nous avons ici découpé l'intervalle de fréquence en 6 sous-bandes de manière à former une partition régulière en log-fréquence. Les sous-bandes sont donc plus larges dans les hautes fréquences, ce qui est utile car il y plus d'informations dans les basses fréquences. En réalité, seuls les maxima supérieurs à un seuil (qui est calculé selon la somme de la moyenne et de l'écart-type des maxima) sont retenus. La figure 21 montre le sonagramme où seules les fréquences positives inférieures à 2000 Hz.

Pour identifier un morceau, il faut bien entendu que celui-ci soit dans une base de données. La base de données de *Shazam* comporte plusieurs millions de titres associés à leur compositeur, album,

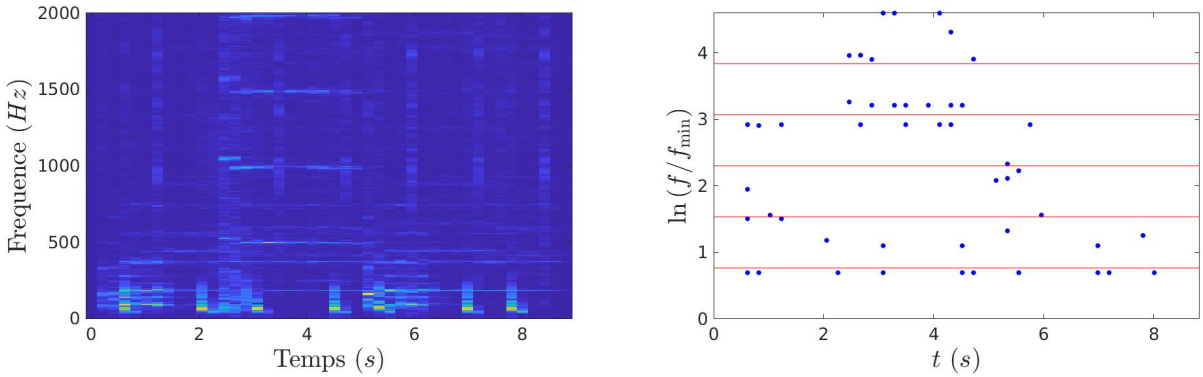

FIGURE 21 – Empreinte sonore obtenue (à droite) à partir du sonagramme (à gauche)

interprète etc. Elle contient les empreintes sonores complètes de chaque son. Ainsi, lorsqu'un son est identifié par *Shazam*, l'empreinte sonore de l'extrait est calculé et comparée avec celles présentes dans la base de données. Il faut au préalable effectuer un recalage temporel entre les empreintes sonores, c'est l'objectif du prochain exercice. Ceci consiste à parcourir sur une période temporelle les empreintes sonores et de calculer un score pour chacune d'entre elles. Le score correspond à la distance minimale entre l'empreinte écoutée et l'empreinte complète du morceau. Une fois celui-ci trouvé, on décale, l'extrait sur l'empreinte complète et on affiche le résultat sur la figure 22.

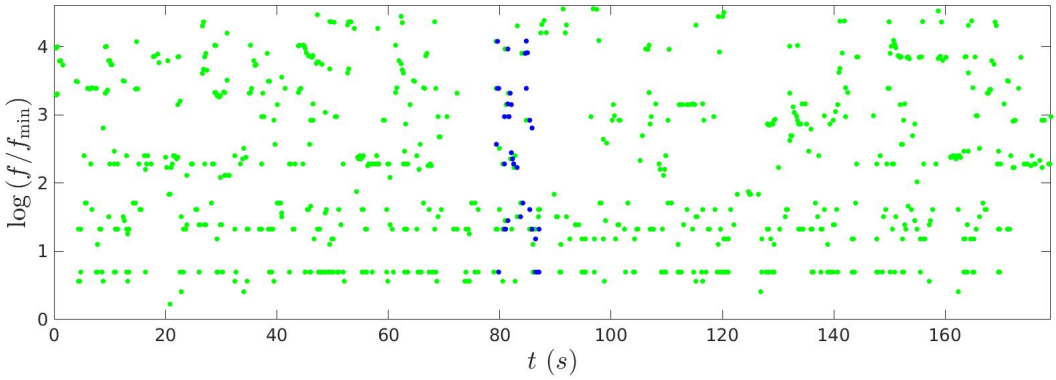

FIGURE 22 – Décalage de l'extrait écouté, en bleu, sur l'empreinte complète, en verte.

Nous travaillions jusqu'à présent sur un seul morceau, le but de *Shazam* est de reconnaître l'extrait parmi des millions. Comme dit précédemment, Shazam fonctionne en plusieurs étapes : d'abord le calcul de l'empreinte sonore, puis le décalage de celle-ci sur chaque son de la base de donné, et enfin le calcul du score pour afficher le résultat de la reconnaissance. L'objectif du dernier exercice est de compléter le script donné pour mettre en place ce fonctionnement. On obtient des résultats concluant, tous les extraits sont reconnus (y compris les Spice Girls) par notre *Shazam* simplifié car il ne comporte "que" 73 éléments dans la base de données. Il peut se trouver que le son enregistré ne soit pas dans la base de données, et que *Shazam* affiche quand même un résultat, faux donc. Dans l'exercice facultatif, on met en place une validation pour que l'application donne un message de "non reconnaissance du morceau" si le score trouvé n'est pas au-dessous d'un certain seuil.

Shazam n'a jusqu'alors pas eu de concurrence sur le marché des applications de reconnaissance musicale. Mais récemment, des applications qui permettent de reconnaître un son à partir d'un fredonnement ou d'un sifflement sont disponibles, bien plus efficaces, car elles utilisent l'apprentissage profond, ce qui n'est pas le cas de *Shazam*.

3.4 Séparation de sources

On peut se servir de la séparation de source pour séparer les partitions de deux instruments joués sur un unique morceau, pour rejouer séparément à chaque partition des deux instruments. En revanche, un tel problème est mal posé car il semble y avoir une infinité de solutions. Dans notre cas, simple, nous avons à disposition un "dictionnaire" de notes pour chaque instrument.
Si on joue un morceau de piano avec une seule note à chaque mesure, alors le *sonagramme* S est une matrice $f * m$ où m désigne le nombre de mesures. L'égalité matricielle suivante peut s'écrire avec A la matrice d'activation, qui équivaut à une partition, et D le dictionnaire : $S = DA$. Pour un morceau joué avec plusieurs instruments, dans notre cas le piano et la trompette, on peut trouver la matrice d'activation si l'on connaît les deux dictionnaires. Ainsi $S_{duo} = D_{duo}A_{duo}$, avec $D_{duo} = [D_{piano}|D_{trompette}]$. Par conséquent $S_{duo} = D_{piano}A_{piano} + D_{trompette}A_{trompette} = S_{piano} + S_{trompette}$ avec les S_i les sonagrammes de i. Pour ce système, on se contente d'une solution approchée, en moindres carrés. On utilise donc la pseudo inverse pour trouver A. On sépare la sépare ensuite pour obtenir les deux matrices pour chaque instrument.

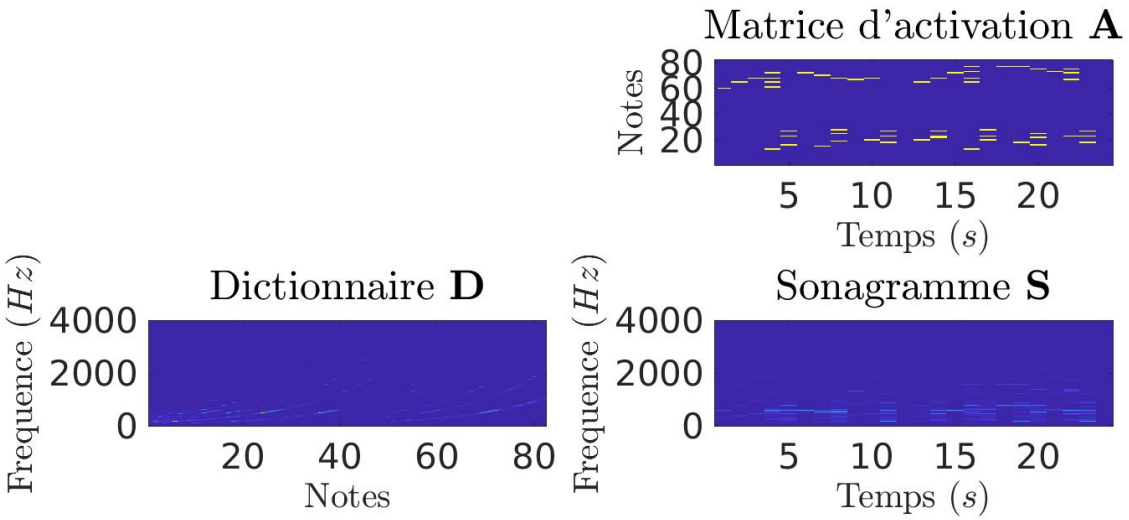

FIGURE 23 – Matrice d'activation à séparer et dictionnaires

Pour une séparation sans connaître les dictionnaires, le problème est beaucoup plus compliqué. Il y a plus d'inconnues, comme le nombre de notes, et le problème est encore mal posé, c'est-à-dire qu'il y a encore une infinité de solutions. Une solution consiste réécrire le problème sous la forme $|S| = |D|A$. C'est le problème *Non negative matrix factorization* (NMF) qui est mieux posé grâce aux contraintes de positivités des inconnues D et A. Par méthode itérative de descente de gradient, on résout le problème.

4 Conclusion

Ceci met un terme à ce (long) rapport des TPs de l'UE *Traitement des données audio-visuelles*. Au cours de ces deux rapports nous avons pu voir les différentes méthodes variationnelles et des transformations de données audio-visuelles. De l'estimation de paramètres à *Shazam* en passant par du photomontage et de l'inpainting, nous avons pu voir de façon très concrète ce qui pouvait nous attendre en tant qu'ingénieur spécialisé en Image & Multimédia (la meilleure spécialisation ?). Plus sérieusement, le côté mathématique de la retouche d'image, de l'inpainting, du photomontage est très intéressant et nous voyant directement les effets des applications mathématiques que nous utilisons, ce qui n'est pas le cas dans la plupart des matières à l'ENSEEIHT. Nous avons une vision concrète des choses que nous pouvons faire en imagerie et je trouve cela très intéressant. Le côté ludique des cours et des explications lors des TPs nous permet de mieux comprendre comment *imaginer*

et implanter les fonctions en *Matlab*. Je ne sais pas si j'ai obtenu les meilleurs résultats possibles pour les TPs présentés ci-dessus mais je pense avoir des résultats corrects et rigoureux pour pouvoir comprendre, d'un point de vue extérieur, ce que j'ai l'occasion de faire en cours de TAV. Les TPs ne sont vraiment pas simples, et il faut, à mon sens, travailler beaucoup chez soi le week-end (ou la veille du rendu dépendant du profil de l'étudiant) pour avoir un résultat correct. Pour ma part, je consacrais environ trois heures en plus des séances pour essayer de finir le TP est avoir de bons résultats, ce qui est à mon sens, un peu trop, mais au moins ce fut intéressant. Merci et à l'année prochaine (ou pas).