Fast and Adaptive Bidimensional Empirical Mode Decomposition Using Order-Statistics Filter Based Envelope Estimation implementation

Antoine Colmard et Nicolas Prugne

23 janvier 2015

Table des matières

Table des figures

Chapitre 1

Présentation du sujet

Cet article aborde l'amélioration apportée par Bhuiyan et al. [1] à la Décomposition Empirique Multimodale Bidimensionnelle (*Bidimensional Empirical Mode Decomposition*). L'amélioration utilise une estimation de l'enveloppe se basant sur un filtre à statistique d'ordre afin d'améliorer les performances de l'algorithme original.

L'algorithme original développé par Huang et al. [2] était utilisé dans le cadre du traitement du signal pour analyser des données non linéaires et non stationnaires. La technique décompose le signal en ses fonctions modales intrinsèques (IMF) puis extrait les distributions temps-fréquence de chaque fonction. Cette décomposition a ensuite été étendue à deux dimensions. Cette décomposition itérative se base principalement sur la recherche des extremums de l'image et de l'interpolation de ceux-ci. Ces étapes peuvent consister en des routines compliquées et coûteuses en temps. Certaines images peuvent donc prendre plusieurs heures ou jours pour être décomposées. L'autre défaut des implémentations actuelles de BEMD repose dans le fait que les points d'interpolation ne sont pas choisis en bordure. Ceci résulte en une imprécision de l'interpolation qui peut se propager jusqu'au centre de l'image.

La décomposition possède de nombreuses applications pratiques telles que l'analyse d'imagerie médicale, de patterns, de textures, etc. Cependant, son implémentation actuelle ne permet pas le traitement efficace d'image de grande taille. Cette perte de détails empêche l'extraction précise d'informations par BEMD. Ainsi, il est important de pouvoir apporter des améliorations à BEMD afin de pouvoir étendre son champ d'application.

L'amélioration proposée dans l'article de Bhuiyan et al. révise les étapes de recherche de maximas et de minimas ainsi que l'estimation des enveloppes de BEMD afin d'en améliorer les performances. L'algorithme ainsi défini est désigné par *Fast and Adaptive Bidimensional Empirical Mode Decomposition* (FABEMD).

Dans cet article, nous aborderons l'algorithme de FABEMD, l'implémentation que nous avons fait de celui-ci et les résultats obtenus.

Chapitre 2

Méthode

Dans cette partie, nous verrons les différentes étapes de BEMD et les améliorations apportées par FABEMD. Ensuite, nous verrons plus en détail l'implémentation que nous avons fait de FABEMD.

2.1 Partie théorique

Pour décomposer une image I en ses différentes fonctions modales intrinsèques, BEMD effectue à chaque la décomposition i d'une image source S_i en son BIMF F_i. S_i est l'image résiduelle obtenue telle que $S_i = S_{i-1} - F_{i-1}$ et $S_i = I$. F_i, quant à elle, est obtenue après plusieurs itérations. Les images intermédiaires obtenues à chaque itération j seront désignées par F_{T_j}. Les étapes de l'algorithme peuvent ensuite être décrites de la façon suivante :

$i \leftarrow 1$
$S_i \leftarrow I$
répéter
  $j \leftarrow 1$
  $F_{T_j} \leftarrow S_i$
  répéter
    $P_j \leftarrow maximums(F_{T_j})$
    $U_{E_j} \leftarrow interpolation(P_j)$
    $Q_j \leftarrow minimums(F_{T_j})$
    $L_{E_j} \leftarrow interpolation(Q_j)$
    $M_{E_j} = (U_{E_j} + L_{E_j}/2)$
    $F_{T_{j+1}} = F_{T_j} - M_{E_j}$
    $D \leftarrow \frac{\sum_{x=1}^{M}\sum_{y=1}^{N}|F_{T_{j+1}}(x,y) - F_{T_j}(x,y)|^2}{\sum_{x=1}^{M}\sum_{y=1}^{N}|F_{T_j}(x,y)|^2}$
    $j \leftarrow j + 1$
  jusqu'à $D < seuil$
  $F_i \leftarrow F_{T_j}$
  $i \leftarrow i + 1$

$S_i \leftarrow S_{i-1} - F_{i-1}$
jusqu'à $nombreExtremums(S_i) < 3$
$R \leftarrow I$

Comme vu précédemment, FABEMD apporte des améliorations aux étapes de recherches d'extremums et de créations d'enveloppes.

2.1.1 Recherche d'extremums locaux

Pour la recherche d'extremums locaux, FABEMD effectue une recherche de maximum (resp. minimum) pour chaque pixel de l'image. Si la valeur de pixel est strictement supérieure (resp. inférieure) à celle de tous ses voisins, alors le pixel est un maximum local (resp. minimum local). Généralement, le choix d'une fenêtre de recherche de $3x3$ permet d'obtenir les extremums optimaux pour l'image.

2.1.2 Création des enveloppes

Pour la création des enveloppes hautes et basses, FABEMD se base sur les extremums calculés précédemment. Pour chaque point de P_j (resp. Q_j), on recherche sa distance à son voisin le plus proche dans P_j (resp. Q_j). Ces distances sont stockées dans un tableau d'adjacence de distance des maximums ($d_{adj-max}$) et des minimums ($d_{adj-min}$). Ces distances permettent de déterminer les tailles des fenêtres pour les filtres d'ordre statistique. Il existe ensuite différentes façons de choisir la taille w_{en-g} des fenêtres : celle-ci peuvent être les même pour le filtre de l'enveloppe basse et haute ou non. Cependant les tailles les plus intéressantes sont le minimum des minimums des distances de $d_{adj-max}$ et $d_{adj-min}$ (d_1) ou le maximum des maximums des distances de $d_{adj-max}$ et $d_{adj-min}$ (d_4).

Ensuite, la valeur de chaque pixel des enveloppe sera la valeur maximale (resp. minimale) dans une fenêtre de taille w_{en-g} autour du pixel. Les enveloppes hautes et basses sont ensuite lissées par un filtre moyenneur de taille w_{en-g}.

2.2 Implémentation

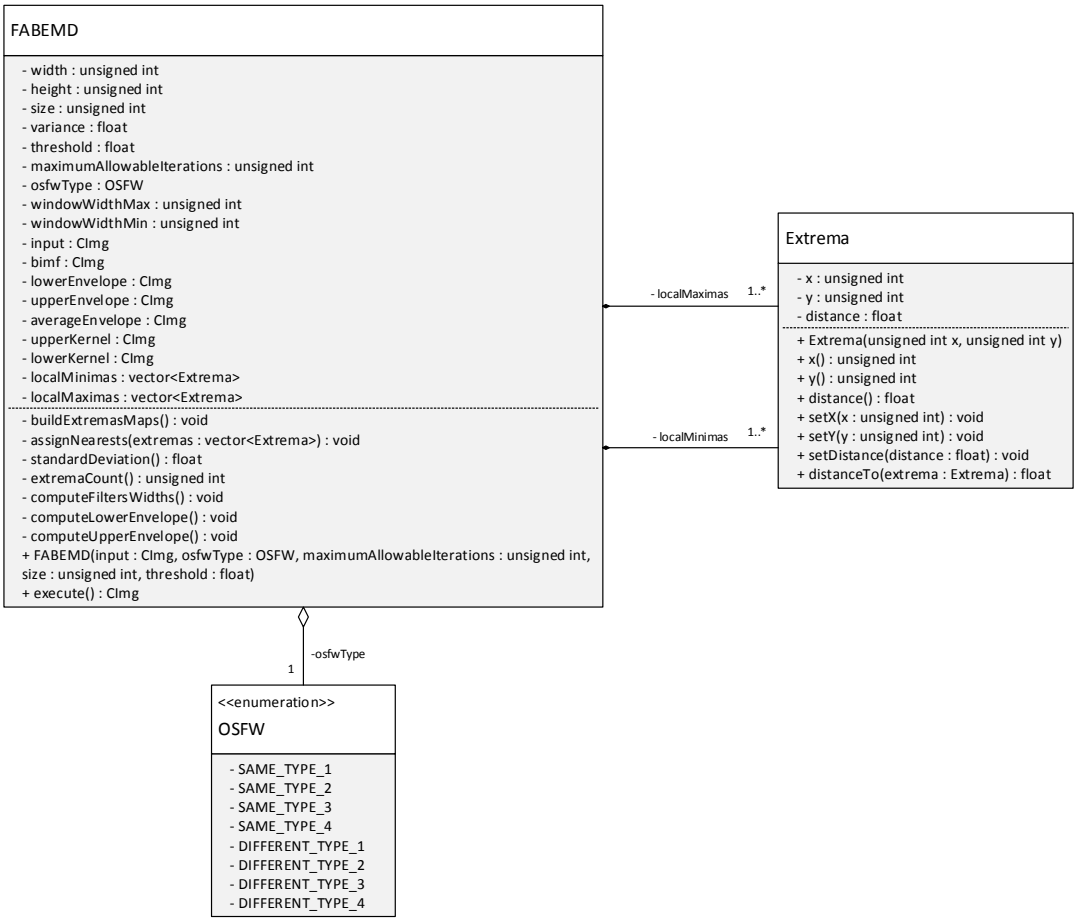
Après avoir vu la partie théorique de FABEMD, nous allons voir en détail l'implémentation que nous en avons fait avec la bibliothèque CImg.

2.2.1 Analyse UML

Pour effectuer la décomposition d'une image à l'aide de FABEMD, nous avons créé une classe `Extrema` pour stocker les extremums de P_j et Q_j ainsi que les distances de $d_{adj-max}$ et $d_{adj-min}$. De plus, une classe `FABEMD` a été créée pour prendre en charge l'exécution de l'algorithme. L'énumération `OSFW`

a également été créée pour choisir la méthode de choix de taille de filtre. La figure 2.1 décrit les interactions entre ces classes.

FIGURE 2.1 – Diagramme UML de l'application



Recherche d'extremums

Pour la recherche d'extremums, la taille de la fenêtre est paramétrable dans le constructeur de `FABEMD`. Cependant, conserver la taille de 3 est conseillée. La construction s'effectue ensuite dans la méthode `buildExtremasMaps` et affecte les extremums dans les vecteurs `localMinimas` et `localMaximas`.

Calcul des tailles de filtres

Une fois les extremums détectés, pour chaque extremum, on recherche la distance à son plus proche voisin avec la méthode `assignNearests`. L'attribut

`distance` de la classe `Extrema` stocke cette valeur. Les vecteurs de distance sont ensuite triés selon leur distance et la distance peut être choisie en fonction du `osfwType` de la classe `FABEMD` grace à la méthode `computeFiltersWidths`.

Calcul des enveloppes

Une fois les tailles de filtres définies, les enveloppes hautes et basses sont calculés avec la méthode `computeUpperEnvelope` et `computeLowerEnvelope`. Pour chaque pixel, on parcourt son voisinage (de taille `windowWidthMax` (resp. `windowWidthMin`) et recherche la valeur la plus élevée (resp. la plus faible) qui lui est ensuite affectée. L'enveloppe ainsi créée est convoluée avec le noyau moyenneur `upperKernel` (resp. `lowerKernel`) pour effectuer le lissage.

Stockage des résultats

Les résultats de l'algorithme (BIMF et résidu) ainsi que l'image originale sont retournés par l'algorithme. L'image originale est contenue à l'indice $z = 0$ et est suivie des différents BIMF et enfin du résidu. L'application principale permet de visualiser ces différents résultats en utilisant la molette de la souris sur l'affichage.

Chapitre 3

Résultats

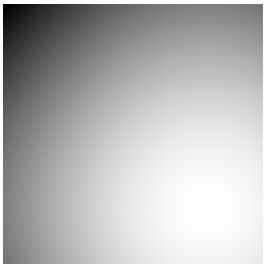
Dans cette partie, nous allons tout d'abord analyser les tests effectués sur des données de synthèse pour valider l'implémentation de FABEMD. Ensuite, nous verrons les résultats obtenus sur des données réelles.

3.1 Validation sur données de synthèse

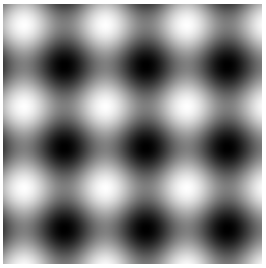
Pour valider l'implémentation faite de l'algorithme FABEMD, nous allons tout d'abord analyser les résultats produits sur des données de synthèse. Pour ce faire, nous allons générer des images composées de sinusoïdes (figure 3.1) et nous chercherons à extraire celles-ci dans chaque BIMF produit par FABEMD. Nous effectuerons nos tests en faisant varier deux des paramètres ayant une influence sur les décompositions produites : le choix de la taille des filtres et le nombre maximal d'itérations pour la génération d'un BIMF. On peut observer qu'on obtient des résultats similaires aux résultats obtenus dans l'article. Le type 4 (figures 3.3 et 3.5) de taille de filtre permet d'obtenir des résultats en un nombre d'itérations moins important. De même, comme décrit par Bhuiyan et al., les résultats obtenus par FABEMD sont optimaux lorsque l'on utilise un nombre maximal d'itérations égal à 1 (figures 3.2 et 3.3). Par la suite, on favorisera donc ce paramètre pour l'utilisation de FABEMD sur des données réelles.

3.2 Données réelles

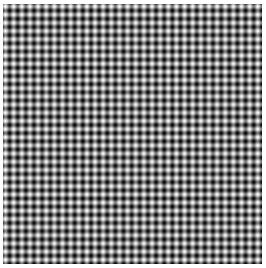
L'implémentation faite de FABEMD permet d'obtenir rapidement des décompositions correctes en fonctions modales intrinsèques. Nous pouvons désormais tester notre application sur des images réelles. Ici, nous utiliserons au maximum une itération pour la génération du BIMF et nous testerons avec les types de fenêtre de filtre 1 et 4. L'image de test utilisée sera celle utilisée dans l'article de recherche.



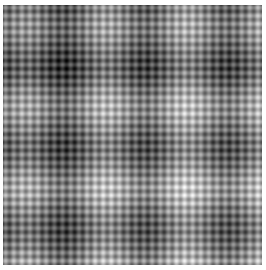
(a) Composante 1



(b) Composante 2

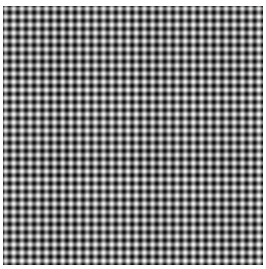


(c) Composante 3

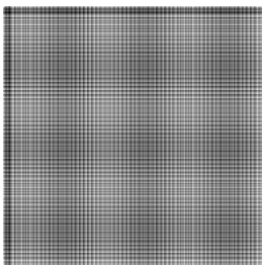


(d) Somme des composantes

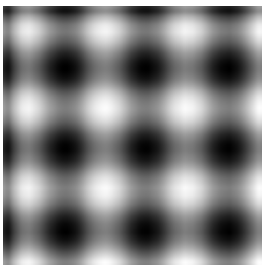
FIGURE 3.1 – Image de synthèse et composantes utilisées pour les tests



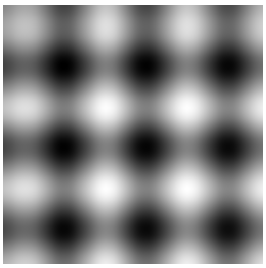
(a) BIMF 1



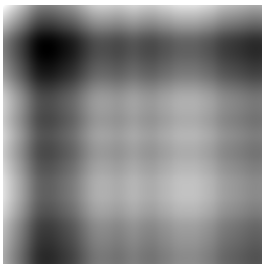
(b) BIMF 2



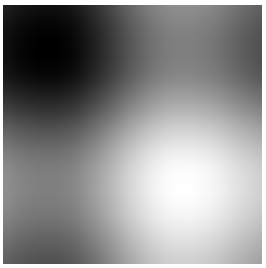
(c) BIMF 3



(d) BIMF 4

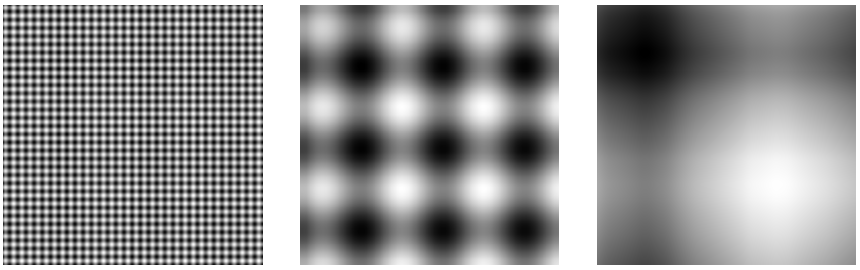


(e) BIMF 5



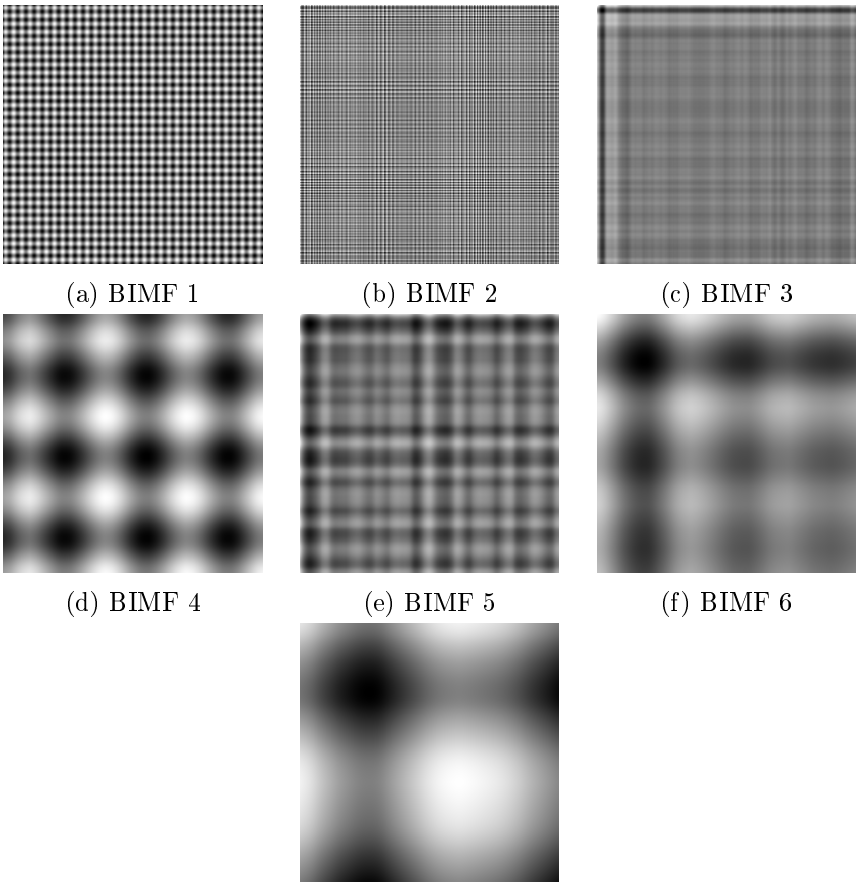
(f) BIMF 6

FIGURE 3.2 – Résultats obtenus avec FABEMD en utilisant au maximum 1 itération et une taille de filtre de type 1



(a) BIMF 1 (b) BIMF 2 (c) BIMF 3

FIGURE 3.3 – Résultats obtenus avec FABEMD en utilisant au maximum 1 itération et une taille de filtre de type 4

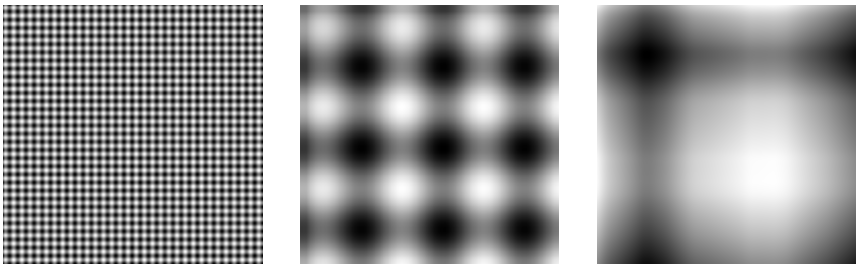


(a) BIMF 1 (b) BIMF 2 (c) BIMF 3

(d) BIMF 4 (e) BIMF 5 (f) BIMF 6

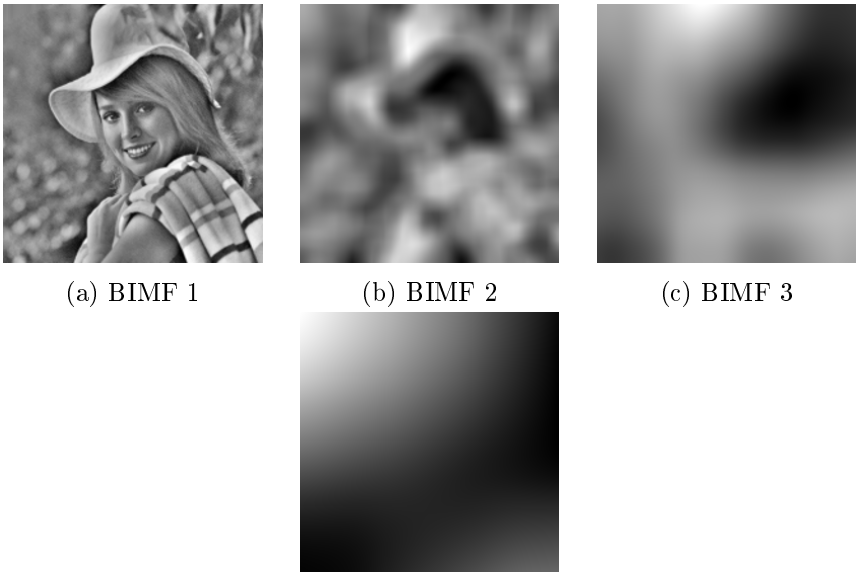
(g) BIMF 7

FIGURE 3.4 – Résultats obtenus avec FABEMD en utilisant au maximum 5 itérations et une taille de filtre de type 1



(a) BIMF 1 (b) BIMF 2 (c) BIMF 3

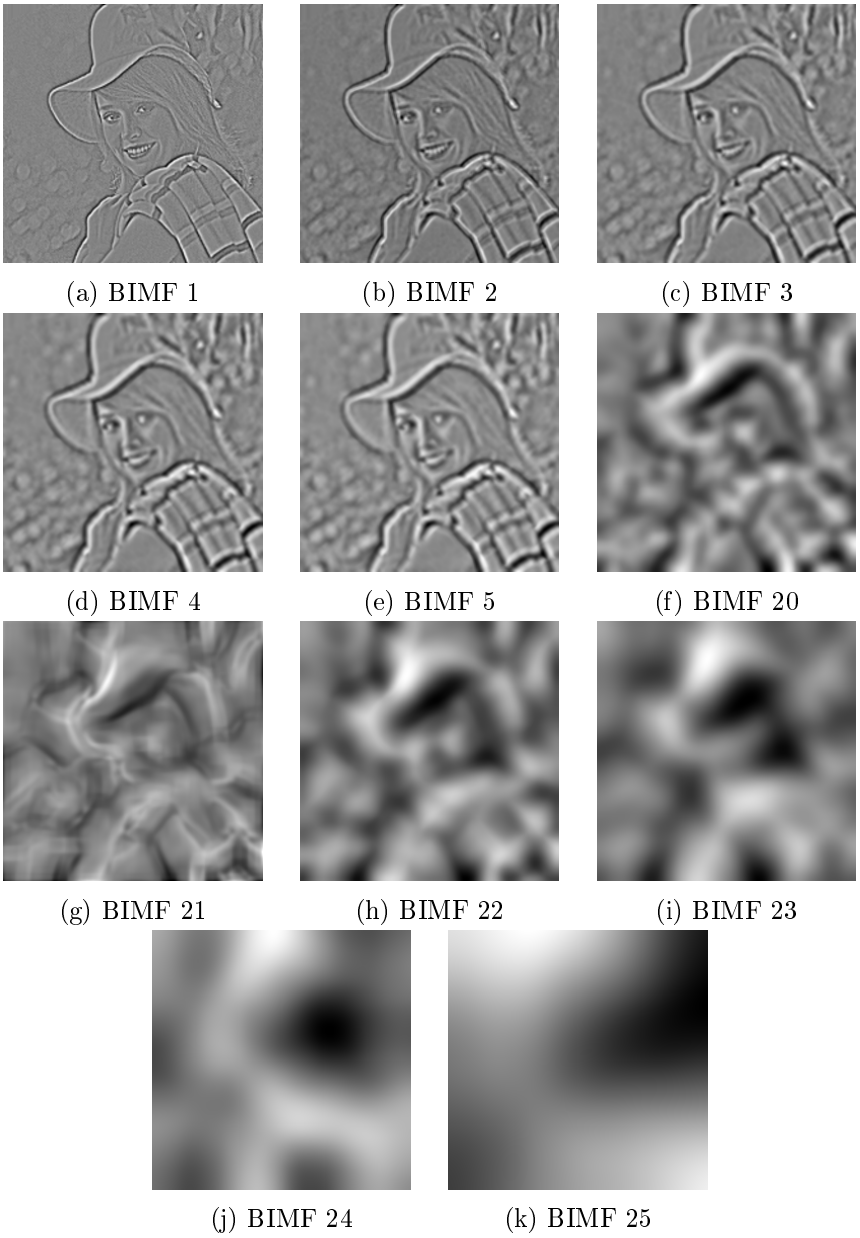
FIGURE 3.5 – Résultats obtenus avec FABEMD en utilisant au maximum 5 itérations et une taille de filtre de type 4



(a) BIMF 1 (b) BIMF 2 (c) BIMF 3

(d) BIMF 4

FIGURE 3.6 – Résultats obtenus avec FABEMD en utilisant au maximum 1 itérations et une taille de filtre de type 4 sur l'image d'Elaine



(a) BIMF 1 (b) BIMF 2 (c) BIMF 3

(d) BIMF 4 (e) BIMF 5 (f) BIMF 20

(g) BIMF 21 (h) BIMF 22 (i) BIMF 23

(j) BIMF 24 (k) BIMF 25

FIGURE 3.7 – Résultats obtenus avec FABEMD en utilisant au maximum 1 itérations et une taille de filtre de type 1 sur l'image d'Elaine

Chapitre 4

Discussion

Les résultats obtenus pour les données de synthèse sont satisfaisant et se conforme aux résultats attendus. L'implémentation de FABEMD permet d'obtenir des résultats rapides mais les performances pourraient être optimisées en utilisant des algorithmes de flou moyenneur plus performants. En effet, dans l'implémentation faite, la plupart du temps de calcul est passé dans la réalisationde ce flou.

Enfin, les résultats obtenus sur des images réelles ont une convergence assez lente[1] lorsque le type de taille de filtre est de type 1. Cependant, ceux-ci permettent d'obtenir des résultats similaires à ceux attendues une fois que des fréquences faibles de BIMF sont atteintes.

[1] L'intégralité de ces résultats sont consultables sur le dépôt de ce projet : `https://github.com/Dramloc/FABEMD`

Bibliographie

[1] Sharif M. A. Bhuiyan, Reza R. Adhami, and Jesmin F. Khan. Fast and adaptive bidimensional empirical mode decomposition using order-statistics filter based envelope estimation. *EURASIP J. Adv. Sig. Proc.*, 2008.

[2] S. R. Long et al. N. E. Huang, Z. Shen. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. *Proceedings of the Royal Society A.*