Déebruitage à l'aide d'ondelettes

SD205 - Estimation non paramétrique

03.04.2017

Le but de ce TP est de réaliser et évaluer des algorithmes de débruitage de signaux et images basés sur une décomposition en ondelettes. Le TP se déroulera en Python, à l'aide du module PyWavelet (pywt), qui permet de réaliser les transformées en ondelettes, ainsi que d'implémenter plusieurs méthodes de débruitage. Avant de commencer le TP, il faut :

- créer un répertoire de travail ;
- télécharger les fichiers supplémentaires de la page
  `https://cagnazzo.wp.imt.fr/?page_id=1329` ou
  `http://cagnazzo.wp.imt.fr` → activites → enseignement → cours → SD205 ;
  - Images monochrome de test
  - Script Python `denoising.py` : c'est le script principal du TP, avec des parties à completer
  - Modules Python `sure.py` et `wtTools.py`

1 Transformée en ondelettes d'un signal 1-D

1. Créez un signal polynomial par morceaux d'ordre ≤ 3 et de taille $N = 512$ points
2. Calculez la TOD sur 3 niveaux avec le filtre Daubechies avec 4 moments nuls : `w = pywt.Wavelet('db4')`
3. Quel type de variables restitue `pywt.wavedec` ?
4. Affichez les histogrammes des différents sous-bandes : commentez sur le nombre de coefficients nuls
5. Qu'est-ce qui se passe si vous utilisez un filtre avec moins moments nuls (`'db1'`) ?

2 Transformée en ondelettes du bruit

1. Créez du bruit Gaussien blanc (BGB) avec `noiseSamples= sigma*np.random.randn(2**N)`
2. Affichez l'histogramme du bruit (`plt.hist`) et calculez l'écart type empirique avec `noiseSamples.std()`
3. Calculez l'histogramme des la transformée en ondelettes discrete (TOD) du bruit. Avec la syntaxe `noiseWT = pywt.wavedec(noiseSamples, w, mode='per', level=nLevel)`, la variable `noiseWT` est la liste des sousbandes. Transformez cette liste en array pour pouvoir calculer l'histogramme et l'écart type empirique.

3 Débruitage d'un signal 1-D

1. Ajoutez du BGB avec écart type `sigma` au signal polynomial. Calculez le SNR avec `wtt.sbSNR`. `wtt.sbbSNR(x,y,lev)` calcule le SNR sousbande par sousbande pour `lev` niveaux de décomposition.

Si `lev=0` (ce qui est la valeur par défaut) le SNR est calculé globalement. Ouvrez wtTools.py pour comprendre l'implémentation de cette fonction.

2. La fonction `wtt.coeff1Dthresh` effectue le seuillage dur ou doux. Ouvrez `wtTools.py` pour en comprendre le fonctionnement.
3. Le seuil universel est calculé comme $\sigma\sqrt{2\log K_m}$, avec K_m le nombre de coefficients de détail, qui est $K_m = K(1 - 2^{-J})$ pour J niveaux de décomposition.
4. Comment est-il calculé le seuil minimax dans `wtt.Minimax` ? Cette fonction restitue le seuil normalisé par rapport à `sigma` en fonction de K_m
5. Effectuez le seuillage dur et doux sur le signal bruité avec le seuil Minimax et le seuil universel. Commentez les résultats
6. Répétez ces opérations pour le signal "wavy". Commentez le résultat.
7. Répétez ces opérations en changeant l'écart type du bruit (entre 0.1 et 1), le nombre de niveaux de décomposition (entre 2 et 5). Commentez les résultats.

4 Transformée en ondelettes d'images

1. Chargez une des images fournies et ajoutez un bruit blanc gaussien d'écart-type $\sigma_b = 30$. Affichez l'image originale et bruité.

   Utilisez `plt.imshow` pour l'affichage et `plt.set_cmap` pour définir la "colormap" de la figure. Pour l'image bruitée, il faut couper les valeur négatives pour supérieures à 255, et afficher des entiers sur 8 bits.
2. Effectuer la décomposition en ondelettes périodiques de l'image originale et de l'image bruitée sur 4 niveaux de résolution en utilisant les ondelettes de Daubechies avec 3 moments nuls : `wav='db3'` La décomposition est effectuée à l'aide de la commande `wavedec2(image, filterbank, mode, level)` La variable `mode` définit la gestion des bords de l'image. Avec les filtres orthogonaux il faut la périodisation.
3. La variable `coeff` est une liste qui contient le différents niveau de résolution de la DWT. `coeff[0]` contient la sous-bande d'approximation ; `coeff[n]` est une liste des 3 sousbandes de détail à niveau `n`.

   En vue de cela, expliquez le fonctionnement de la fonction `coeffs_to_array` dans le module `wtTools`
4. Affichez les valeurs absolues des coefficients d'ondelette de l'image originale avec `plt.imshow` et avec la fonction `wtView` fournie dans le module `wtTools`.

   Quelle est la différence entre ces deux modes de visualisation ? [1]
5. Où sont-ils localisés les coefficients d'ondelettes de grand amplitude ?
6. Affichez les histogrammes des sousbandes d'ondelette. Que peut-on observer ?
7. Estimez l'écart type du bruit avec la médiane des valeurs absolues de la sousbande HH.

5 Débruitage d'image

1. Affichez la transformée de l'image bruitée
2. Utiliser la fonction `sbSNR` dans le module `wtTools` pour calculer le SNR sous-bande par sous-bande et aussi globalement. Utilisez `wtt.sbSNR(arr,arrN,NLEV,1)` pour afficher les SNR sousbande par sousbande, et utilisez `wtt.sbSNR(arr,arrN,0,1)` pour afficher le SNR globale. Comparez le SNR de

1. La premier fonction fait une remise à l'échelle globale, et la deuxième une remise à l'échelle sous-bande par sous-bande ; en outre, cette dernière affiche la valeur absolue des coefficients, et utilise le blanc pour les coefficients petits et le noir pour les coefficients grands.

la sousbande d'approximation et des sousbandes de détails avec le SNR globale. Vous pouvez calculer le SNR entre `img` et `noisyImg` avec `wtt.sbSNR(img, noisyImg)` ou avec `wtt.sbSNR(arr, arrN)`. Les deux opérations donnent-elles le même résultat ? Pourquoi ?

3. Débruiter l'image à l'aide d'une méthode de seuillage dur ou doux. Dans un premier cas, vous testerez 50 valeurs du seuil comprises entre 0 et $T_U * 1.2$, ou T_U est le seuil universel. Affichez ensuite le SNR de l'image débruitée en fonction du seuil. Est-ce que cette méthode peut être utilisée en pratique pour calculer le seuil ? Pourquoi ?
4. Calculez le seuil universel et Minimax, et effectuer le seuillage dur et doux. Comparez les résultats avec le cas précédent. Commentez.
5. Répetez les opérations de débruitage en changeant d'image, de valeur de `sigma`, de nombre de niveaux de décomposition, de filtre d'ondelette (`'db3','db4', 'db5'`)
6. Ouvrez la fonction `sure.hybridDenoisig` dans `sure.py`. Qu'est-ce que cette fonction fait ?
7. Effectuez le débruitage SURE sur l'image et comparez le résultat avec les cas précédents.
8. Comparez le débruitage par ondelettes au débruitage linéaire optimale (filtre de Wiener). Quelle est la réponse en fréquence du filtre de Wiener ?
9. Mettre en œuvre une méthode de débruitage par ondelettes invariantes par translation (Undecimated DWT). Une façon simple de procéder (mais ce n'est pas le plus efficace en terme de calculs) consiste à considérer toutes les translations périodiques de l'image de facteur de translation (dx, dy), où chacun des dx, dy est compris entre 0 et $2^{J_{\max}} - 1$, $J_{\max}$ étant le nombre de niveaux de décomposition maximal. Pour chacune des images translatées, on effectue un seuillage SURE, puis on calcule la moyenne de toutes les estimées ainsi obtenues. La translation pourra s'effectuer comme montré dans `denoising.py`