Université Paris Descartes
UFR de Mathématiques et Informatique
45, rue des Saints-Pères 75270 Paris cedex 06

MASTER 1

PROGRAMMATION

Projet "Débruitage d'image par ACP" - Scilab

Rédaction du rapport
Le rapport devra contenir :
- Une introduction expliquant et illustrant le problème.
- Une explication détaillée des algorithmes et codes et leur interprétation.
- Plusieurs exemples d'application.

Attention, votre rapport et votre code ne doivent comporter aucun copier-coller provenant d'internet, sous peine de sanction importante sur la note finale. Si vous voulez citer une page web ou un article, faites-le en utilisant des guillemets et en indiquant la source de la citation, mais n'en abusez pas.

Mise en route
Depuis un terminal (placez vous dans le dossier de votre choix) tapez les commandes suivantes pour télécharger les fichiers nécessaires au TP :

```
wget http://w3.mi.parisdescartes.fr/~jdelon/enseignement/MA106/TPimages.zip
unzip tdimage.zip
cd TPimages
```

Le dossier TPimages contient plusieurs images qui vous serviront pour le projet.
Pour ouvrir et lire des images sous Scilab, on utilisera la toolbox SIVP qui est installée sur les machines. Commencez par charger la toolbox dans l'interface graphique de Scilab. Une fois la toolbox chargée, les lignes suivantes vous permettent d'ouvrir et d'afficher une image dans Scilab :

```
uo = double(imread('simpson.png'));        // on ouvre l'image simpson.png
figure(1);imshow(uo/255);                  // on affiche l'image
```

1 Introduction

Ce projet porte sur le débruitage d'images par Analyse en composantes principales (ACP). Il est inspiré de l'article [1].

Dans ce qui suit, on considère des image discrètes à $n \times m$ pixels et prenant leurs valeurs dans $[0, 255]$. On rappelle qu'une image peut se représenter dans Scilab sous la forme d'une matrice $n \times m$.
Etant donnée une image u_0 inconnue, on suppose qu'on observe

$$v = u_0 + b$$

où b est une image de bruit blanc gaussien d'écart-type s. Le but de ce projet est de débruiter v, c'est-à-dire de retrouver les valeurs des pixels de u_0.

On commence par créer l'image à débruiter. On part pour cela d'une image u_0, à laquelle on ajoute une image de bruit blanc gaussien b.

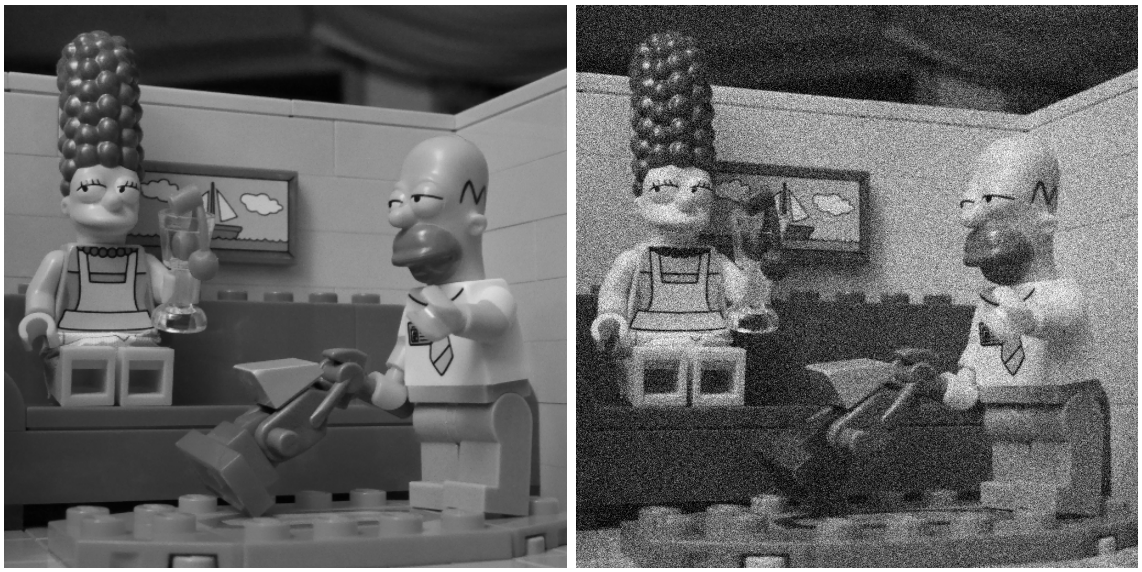

FIGURE 1 – A gauche l'image originale u. A droite l'image v qui a été construite en ajoutant un bruit blanc gaussien d'écart-type 40 à u. On veut, à partir de v, essayer de retrouver l'image u le mieux possible.

► **Question 1 :** *Ecrire une fonction* `[v]=Bruite(u0,s)` *qui prend en entrée une image* u_0 *et un écart-type* s*, lui ajoute une image* b *de même taille que* u_0 *et composée de nombres aléatoires suivant la loi normale de moyenne* 0 *et d'écart type* s*, et renvoie l'image* v *ainsi obtenue. Pour générer l'image de bruit, on pourra utiliser la fonction* `grand`*.*

2 Découpage en patchs

On oublie maintenant que l'on connaît u et on va essayer à partir de l'image v de retrouver les valeurs de u. Pour cela, on commence par découper l'image en patchs. Les patchs sont des petites images carrées de taille $(2f+1)\times(2f+1)$ (par exemple 7×7 si $f=3$) extraites de l'image de départ. En Scilab, pour une image u, le patch de coordonnées i,j et de largeur $2f+1$ est la sous-image

```
u(i-f:i+f,j-f:j+f)
```

Au lieu de débruiter v, on va extraire tous les patchs possibles de v et les débruiter ensemble. Pour cela, on commence par extraire les patchs de u et les mettre dans une matrice Y. Remarquons qu'il y a un patch centré en chaque pixel de l'image v, donc $n\times m$ patches en tout.

► **Question 2 : Etape d'extraction.** *Expliquer ce que fait la fonction* `[Y]=PatchOut(u,f)` *suivante. Quel est le lien entre la matrice* Y *et l'image* u *? Que contient exactement la* k*-ième colonne de* Y *?*

```
function Y=PatchOut(u,f)
    [nl,nc] = size(u);
    Y       = zeros((2*f+1)^2,nl*nc);
    for i = 1:2*f+1
        for j = 1:2*f+1
            v = zeros(nl,nc);
            v(1:nl-i+1,1:nc-j+1) = u(i:$,j:$);
            Y(i+((2*f+1)*(j-1)),:) = matrix(v,1,nl*nc);
        end
    end
endfunction
```

3 Débruitage par ACP

Une fois les patchs extraits, on les considère comme un ensemble de vecteurs colonnes $Y = (Y_1, \ldots, Y_{nm})$, chaque vecteur étant dans $\mathbb{R}^{(2f+1)^2}$. Pour éliminer le bruit de l'image v, on propose de seuiller ces vecteurs (éliminer leurs petits coefficients) après les avoir transformés dans une base orthonormée bien choisie. Cette base orthonormée est apprise en effectuant une analyse en composantes principales sur l'ensemble des vecteurs $(Y_1, \ldots, Y_{nm})$. On calcule pour cela le vecteur moyenne m_Y et la matrice de covariance C de cet ensemble des vecteurs :

$$m_Y = \frac{1}{nm}\sum_{k=1}^{nm} Y_k, \quad C = \frac{1}{nm}\sum_{k=1}^{nm}(Y_k - m_Y)(Y_k - m_Y)^t.$$

► **Question 3 :** *Ecrire une fonction* `[mY,C,Yc]=moyCov(Y)` *qui calcule le vecteur moyenne et la matrice de covariance d'un ensemble de vecteurs colonnes* $Y = (Y_1, \ldots, Y_{nm})$*, et qui renvoie également la matrice centrée* $Y_c = (Y_1 - m_Y, \ldots, Y_{nm} - m_Y)$*.*

L'analyse en composantes principales consiste alors à calculer les vecteurs propres et valeurs propres de la matrice C. Plus précisément, si $\lambda_1 \geq \cdots \geq \lambda_{(2f+1)^2}$ sont les valeurs propres de C, et $X_1, \ldots, X_{(2f+1)^2}$ les vecteurs propres correspondants, on appelle X_k la k-ième composante principale de l'ensemble des vecteurs $(Y_1, \ldots, Y_{nm})$.

► **Question 4 :** *Ecrire une fonction* `X=ACP(Y)` *qui calcule une base orthonormée* X *de vecteurs propres de la matrice de covariance* C*.*

Chaque vecteur Y_k peut être écrit dans cette nouvelle base orthonormée X comme

$$Y_k = m_Y + Y_k - m_Y = m_Y + \sum_{i=1}^{(2*f+1)^2} X_i^t(Y_k - m_Y)X_i.$$

Seuillage dur. Pour débruiter les vecteurs Y_k, on propose de mettre à 0 tous les coefficients $X_i^t(Y_k - m_Y)$ inférieurs en valeur absolue à un seuil η fixé. On construit donc le nouveau vecteur

$$Z_k = m_Y + \sum_{i=1}^{(2*f+1)^2} \phi\left(X_i^t(Y_k - m_Y)\right) X_i, \tag{1}$$

où ϕ est la fonction de seuillage dur

$$\phi(t) = t.\mathbf{1}_{|t|>\eta}. \tag{2}$$

Si on note X la matrice des vecteurs propres $(X_1, \ldots, X_{(2f+1)^2})$, ce seuillage peut se faire très simplement en Scilab sur tous les vecteurs Y_k en calculant

```
Yproj=X'*(Y-mY);
Yproj(abs(Yproj)<eta)=0;
Z = X*Yproj + mY*ones(1,size(Y,2));
```

► **Question 5 :** *Expliquez précisément ce que fait chacune des 3 lignes de code écrites ci-dessus.*

► **Question 6 :** *Ecrire une fonction* `[Z]=debruitePCA(Y,eta)` *qui calcule l'analyse en composantes principales de* Y*, et qui met à* 0 *ses petits coefficients dans la base* X *avant de reconstruire* Z*.*

Seuillage doux. La reconstruction précédente, qui utilise un seuillage dur, est un peu brutale. Pour seuiller de manière plus douce, on peut remplacer la fonction ϕ de l'équation (2) par la fonction

$$\psi(t) = \max(0, |t| - \eta) * sign(t) = (t - \eta) * \mathbf{1}_{t>\eta} + (t + \eta) * \mathbf{1}_{t<-\eta}. \tag{3}$$

► **Question 7 :** *Ecrire une nouvelle fonction* `[Z]=debruitePCAdoux(Y,eta)` *qui calcule l'analyse en composantes principales de* Y*, et qui met à* 0 *ses petits coefficients par seuillage doux dans la base* X *avant de reconstruire* Z*.*

4 Reconstruction

Maintenant que l'on a plusieurs méthodes pour débruiter la matrice Y, il faut reconstruire l'image débruitée u correspondante. On a pour cela plusieurs choix.

► **Question 8 :** *Expliquer pourquoi on peut simplement extraire une ligne k de Z et utiliser la commande* `u = matrix(Z(k,:),n,m);` *pour reconstruire l'image débruitée.*

► **Question 9 :** *Chaque ligne de Z donne une version débruitée de Y mais décalée en espace. Afin de débruiter encore un peu l'image, on se propose de moyenner ces différentes versions. Vous pouvez pour cela utiliser le code suivant, en prenant le soin d'expliquer en détail ce qu'il fait :*

```
function u=Reconstruct(Y,nl,nc)
  [N,M]=size(Y); f = 0.5*(sqrt(N)-1);
  t = zeros(nl,nc,N);
  nb = zeros(nl,nc);
  for x = 1:2*f+1
    for y = 1:2*f+1
      i = (2*f+1) *(y-1)+x
      w = matrix(Y(i,:),nl,nc);
      t(x:$,y:$,i) = w(1:nl-x+1,1:nc-y+1);
      nb(x:$,y:$) = nb(x:$,y:$) +1;
    end
  end
  u = sum(t,3)./nb;
endfunction
```

► **Question 10 :** *Tester la méthode complète de débruitage ainsi obtenue, pour plusieurs valeurs de s (le niveau de bruit) et plusieurs valeurs du paramètre η. Pour évaluer les résultats obtenus, vous pourrez par exemple calculer la distance L^2 entre l'image débruitée et l'image de départ u_0. Peut-on trouver un lien entre le niveau de bruit s et le paramètre η qui donne les meilleurs résultats ? Tester la méthode sur plusieurs images.*

5 Localisation

La méthode précédente utilise une seule base orthonormée pour représenter et seuiller tous les patchs de l'image. On parle donc de méthode globale. Afin de mieux s'adapter aux spécificités locales de l'image, on peut essayer de l'appliquer localement, sur des sous-régions, par exemple en découpant l'image en rectangles.

► **Question 11 :** *Ecrire une fonction qui découpe l'image en rectangles disjoints (par exemple 4×4 rectangles) et applique la méthode de débruitage précédente indépendamment à tous ces rectangles avant de reconstruire l'image. Quel défaut observez-vous sur le résultat ? Proposer une solution pour éviter ce défaut.*

Références

[1] C.-A. Deledalle and J. Salmon and A. Dalalyan, Image denoising with patch based PCA : local versus global, *proceedings of BMVC 2011*, 2011.