CY-Tech 2022 - 2023	Analyse numérique **TP noté**	**Mathématiques** ING1-GMI

Réparation d'image par SVD

1 Introduction

Le traitement d'images est une préoccupation importante dans de nombreux domaines, tant industriels que d'usage quotidien. Différentes techniques sont disponibles suivant l'usage qui doit être fait et les objectifs de l'utilisateur : minimiser les pertes, maîtriser la qualité, etc.
Nous voyons dans ce TP une méthode simple utilisant la décomposition en valeurs singulières (SVD) pour réparer une image déteriorée par un pliage par exemple.
Cette méthode est décrite par l'article [1].

2 Notions mathématiques

On rappelle le théorème vu en cours sous la référence 7.2

Théorème - Construction d'une décomposition en valeurs singulières
Soit n et p deux entiers naturels non nuls et $A \in M_{np}(\mathbb{R})$ de rang r. On note $(\sigma_i)_{1\leqslant i\leqslant r}$ les valeurs singulières de A.
Soit $(u_i)_{1\leqslant i\leqslant n}$ une base orthonormale de vecteurs singuliers à gauche telle que u_i soit un vecteur propre de AA^T associé à la valeur propre σ_i^2 si $i \leqslant r$. On note U la matrice de la base $(u_i)_{1\leqslant i\leqslant n}$.
Soit $(v_j)_{1\leqslant j\leqslant p}$ une base orthonormale de vecteurs singuliers à droite telle que v_j soit un vecteur propre de A^TA associé à la valeur propre σ_j^2 . On note V la matrice de la base $(v_j)_{1\leqslant j\leqslant p}$.
Soit $\Sigma \in M_{np}(\mathbb{R})$ la matrice définie par $\Sigma_{ij} = \begin{cases} \sigma_i & \text{si } i = j \text{ et } i \leqslant r \\ 0 & \text{sinon} \end{cases}$.
Alors $A = U\ \Sigma\ V^T$.

Corollaire
On considère la décomposition définie dans le théorème ci-dessus. Alors $A = \sum_{i=1}^{r} \sigma_i\ u_i\ v_i^T$.
Symboliquement, on peut représenter la situation sous la forme suivante dans le cas où $n > p$ par exemple.

$$A = U\ \Sigma\ V^\top = \begin{pmatrix} u_{1,1} & \dots & \dots & u_{1,n} \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ u_{n,1} & \dots & \dots & u_{n,n} \end{pmatrix} \begin{pmatrix} \sigma_1 & & 0 & \\ & \ddots & & 0_{r,p-r} \\ 0 & & \sigma_r & \\ & 0_{n-r,r} & & 0_{n-r,p-r} \end{pmatrix} \begin{pmatrix} v_{1,1} & \dots & v_{p,1} \\ \vdots & \ddots & \vdots \\ v_{1,p} & \dots & v_{p,p} \end{pmatrix}$$

Une approximation possible de A par une matrice de rang k consiste à ne conserver que les $k \leqslant r$ plus grandes valeurs singulières et à reconstituer la matrice A_k donnée par :

$$A_k = U_k\ \Sigma_k\ V_k^T$$

où U_k est la matrice de la base $(u_i)_{1\leqslant i\leqslant k}$, V_k la matrice de la base $(v_j)_{1\leqslant j\leqslant k}$, et Σ_k la matrice diagonale contenant les k premières valeurs singulières de A.
Cette approximation est parfois qualifiée d'approximation de rang faible.

Nous nous intéressons à l'utilisation de cette approximation pour réparer une image abimée.

Une image peut être modélisée comme une superposition de 3 couleurs (dites canaux) pour chaque pixel (rouge, vert, bleu), chaque pixel étant lui-même habituellement codé sur une échelle de 256 valeurs (de 0 à 255). La taille de l'image est identique dans chaque canal, et correspond aux dimensions de la matrice des pixels : nombre de lignes et nombre de colonnes.
La commande *imread* de Python donne une matrice de 3 dimensions.
Pour quantifier la qualité de l'approximation ainsi réalisée, on peut évaluer l'erreur quadratique $\|A-A_k\|_2$ ou l'erreur selon la norme de Frobenius $\|A - A_k\|_2$.

La procédure de réparation fonctionne de la manière suivante :

1. La partie abimée (pliure) est gommée et ensuite repérée dans un fichier annexe appelé masque, permettant de repérer les pixels abimés qu'il faut restaurer.
2. Grâce à ce repérage, la matrice de l'image A, peut être écrite comme la réunion : $A = A(\text{saine}) \cup A(\text{abimée})$.
3. A partir de la matrice de l'image A, on obtient l'approximation de rang faible A_k. Celle-ci peut également s'écrire : $A_k = A_k(\text{saine}) \cup A_k(\text{abimée})$.
4. Pour réparer l'image : dans la partie saine, on conserve A(saine), en revanche, dans la partie abimée, on prend A_k(abimée), ce qui nous donne une nouvelle matrice pour l'image : $\tilde{A} = A(\text{saine}) \cup A_k(\text{abimée})$.
5. La qualité de la réparation est évaluée grace à la norme $\|A - \tilde{A}\|_F$ et comparée à un seuil de précision souhaité ε.
6. Si $\|A - \tilde{A}\|_F > \varepsilon$, la matrice A est remplacée par $\tilde{A}$ et on recommence le processus précédent.

L'idée derrière cette méthode est que l'approximation de rang faible tient compte de tous les pixels. La partie A_k(abimée) est donc obtenue par une sorte de mélange avec les pixels voisins.
L'itération de ce procédé induit une sorte d'interpolation des valeurs des pixels abimés à partir des pixels sains. Ce procédé fait disparaître petit à petit la dégradation et restaure l'image initiale.

3 Travail demandé

Mathématiques

1. Justifier l'écriture matricielle $A = \sum_{i=1}^{r} \sigma_i \, u_i \, v_i^T$ traduisant la décompositttion en valeurs singulières : $A = U \, \Sigma \, V^T$.
2. Justifier que l'erreur quadratique est donnée par : $\|A - A_k\|_2 = \sigma_{k+1}$.
3. Justifier que l'erreur de Frobenius vérifie : $\|A - A_k\|_F^2 = \sigma_{k+1}^2 + \sigma_{k+2}^2 + ... + \sigma_r^2$.

Informatique
Vous devez coder des fonctions permettant de :

1. Lire une image, identifier ses caractéristiques et accéder à ses canaux.
2. Restaurer une image en noir et blanc à partir d'une image fournie et d'un masque de la ligne à éliminer.
3. Restaurer une image en couleurs à partir d'une image fournie et d'un masque de la ligne à éliminer.

3.1 Données et outils pour les images

1. Une image en noir-et-blanc à restaurer (*cheval-abime2-nb.png*), cette même image en couleur à restaurer (*cheval-abime2.png*) ainsi que le masque de la ligne à éliminer (*cheval-abime2-mask.png*) sont également fournis.
2. Le package pyplot de matplotlib est largement suffisant pour le travail demandé.
   Vous utiliserez, en particulier, les commandes :
   — imread
   — imshow

3. La décomposition SVD peut être réalisée grâce à la commande np.linalg.svd
   Il vous appartient de trouver l'usage de ces commandes dans la documentation.

4 Modalités de rendu

4.1 Date
Vous devez envoyer votre rendu par mail à votre professeur avant vendredi 14 avril à l'heure indiquée par votre professeur. Tout retardataire aura la note 0.
4.2 Consignes
— Le rapport pdf doit avoir une qualité de travail scientifique, les formules mathématiques doivent être propres et lisibles (latex conseillé).
— Les groupes seront des binômes, constitués à la convenance des élèves.
— Sur la première page vous indiquerez, les noms et prénoms des membres du groupe, le titre du TP.
— Le rapport doit comporter impérativement l'analyse de la méthode. Aucun code informatique ne doit y être inséré, hormis si vous avez à formuler une remarque sur l'implémentation d'une formule par exemple.
— Il est vivement conseillé de recourir à des graphiques ou des tanleaux pour présenter vos résultats.
— Le rendu se fera impérativement via une archive .zip nommée TPnote_Nom1_Nom2.zip comportant les codes sources python, les éventuels fichiers de données, et le pdf de votre rapport.
4.3 Fonctionnalités minimales du code :
— Restauration d'une image en noir et blanc.
— Restauration d'une image en couleurs.

5 Références

[**1**] Kong Hoong Lem, *Truncated singular value decomposition in ripped photo recovery*, ITM Web of Conferences 36, 04008 (2021)