TP1 – Analyse en Composantes Principales

Corrélation entre les trois canaux d'une image RVB

Lancez le script `exercice_0`, qui lit l'image `autumn.tif` codée en RVB (rouge, vert, bleu) et la stocke dans une matrice tridimensionnelle `I` :

- Les valeurs entières `I(i,j,1)`, `I(i,j,2)` et `I(i,j,3)`, comprises entre 0 et 255 (8 bits), sont les *niveaux de couleur* du pixel situé sur la ligne `i` et la colonne `j`, dans les canaux R, V et B.
- La matrice `I` est scindée en trois matrices bidimensionnelles `R`, `V` et `B` correspondant aux trois canaux.
- Les matrices `I`, `R`, `V`, `B` sont affichées sous forme d'images.

Vous remarquez tout d'abord que l'œil humain est plus sensible à la lumière verte qu'aux lumières rouge et bleue. Vous observez également une forte corrélation, c'est-à-dire une forte « ressemblance », entre les trois canaux, à part pour les régions très colorées (l'arbre situé au centre de l'image « disparaît » dans le canal bleu).

Exercice 1 : coefficients de corrélation et proportions de contraste

En considérant un pixel comme un point de l'espace $\mathbb{R}^3$, le script `exercice_1` affiche l'ensemble de ces points 3D dans un repère dont les axes correspondent aux trois niveaux de couleur. Le nuage ainsi créé présente une forme très allongée parallèle au vecteur $[1, 1, 1]$, ce qui confirme l'observation précédente, à savoir que les trois canaux sont fortement corrélés. Écrivez la fonction `correlation_contraste`, appelée par le script `exercice_1`, dont le rôle est de :

- Calculer la matrice `Sigma` de variance/covariance des variables aléatoires correspondant aux trois canaux (matrice de taille 3×3).
  Attention : n'oubliez pas de centrer la matrice `X` des données, et n'utilisez pas les fonctions `var` et `covar`, qui appliquent des prétraitements aux données.

- Calculer le *coefficient de corrélation linéaire* $r_{i,j} \in [-1, 1]$ de chaque paire de canaux qui s'écrit, pour $(i, j) \in \{\mathrm{R}, \mathrm{V}, \mathrm{B}\}^2$:
  $$r_{i,j} = \frac{\sigma_{i,j}}{\sigma_i \sigma_j} \tag{1}$$
  où σ_i désigne l'écart-type du canal i, σ_j l'écart-type du canal j, et $\sigma_{i,j}$ la covariance des canaux i et j.
  Attention : ne confondez pas écart-type et variance !

- Calculer la *proportion de contraste* de chaque canal qui s'écrit, pour $i \in \{\mathrm{R}, \mathrm{V}, \mathrm{B}\}$:
  $$c_i = \frac{\sigma_i^2}{\sigma_\mathrm{R}^2 + \sigma_\mathrm{V}^2 + \sigma_\mathrm{B}^2} \tag{2}$$
  où σ_i^2 désigne la variance du canal i.
  Remarque : la somme des proportions de contraste est égale à 1.

Transmission d'une image couleur par un seul canal

Le choix d'un espace de représentation des couleurs s'est posé lorsque les chaînes de télévision sont passées à la couleur, dans les années 1960. En effet, il était inutile de transmettre trois canaux R, V et B aux utilisateurs (encore nombreux) possédant des téléviseurs noir et blanc, qui ne pouvaient afficher qu'un seul canal. Le principal critère était de maximiser la proportion de contraste de ce canal unique. La conversion d'une image couleur en une image en niveaux de gris consiste alors en une réduction de dimension. En choisissant de conserver la première composante principale, l'analyse en composantes principales (ACP) peut alors être considérée comme technique très générale de réduction de dimension.

Exercice 2 : analyse en composantes principales

Écrivez la fonction `ACP`, appelée par le script `exercice_2`, qui effectue l'ACP des données de la matrice `X` et retourne `C` qui contient les 3 composantes principales (laissez la seconde sortie de la fonction à 0 pour le moment) :

- La matrice `Sigma` de variance/covariance, étant symétrique réelle, admet une base orthonormée de vecteurs propres. Calculez ses valeurs et vecteurs propres à l'aide de l'appel : `[W,D] = eig(Sigma)`.
- Les valeurs propres de `Sigma` sont stockées sur la diagonale de la matrice `D`. Triez ces valeurs par ordre décroissant, à l'aide des fonctions `diag` et `sort` (avec l'option `'descend'`).
- Les vecteurs propres de `Sigma`, appelés *vecteurs principaux* dans le cas de l'ACP, sont stockés sur les trois colonnes de la matrice de changement de base `W`. La matrice `W` est donc orthogonale, c'est-à-dire que son inverse est égale à sa transposée. Elle constitue la matrice de passage entre le repère RVB et le nouveau repère ayant pour axes les *axes principaux*. Calculez la matrice `C` des coordonnées des pixels dans ce nouveau repère, appelées *composantes principales*.
  Attention : n'oubliez pas d'appliquer aux colonnes de `W` la permutation qui a permis de trier les valeurs propres de `Sigma` par ordre décroissant.

Les bornes de l'image `I` (et donc de la matrice `X`) étaient auparavant 0 et 255 car il s'agit des limites prises par les différents pixels sur chaque canal. Afin de visualiser les trois composantes suivant la même échelle, il est nécessaire de calculer les valeurs minimales et maximales de la matrice `C` des composantes principales et les mettre dans la variable de sortie `bornes_C` de la fonction `ACP`.

La conversion de l'image RVB en niveaux de gris correspond donc à la première composante principale obtenue à partir des 3 coefficients de la première colonne de la matrice `W` de sorte que :

$$C_1(i,j) = W(1,1) * R(i,j) + W(2,1) * V(i,j) + W(3,1) * B(i,j) \quad (3)$$

Complétez la troisième sortie de la fonction `ACP` pour qu'elle renvoie le vecteur **normalisé** des coefficients de la première colonne de `W` pouvoir comparer ces coefficients avec ceux du standard. En pratique, l'utilisation de l'ACP est trop coûteuse. C'est pourquoi il a été décidé de créer un standard de conversion entre couleurs et niveaux de gris, indépendant de l'image, où les coefficients pour la conversion valent respectivement [0,299 0,587 0,114] pour le canaux rouge, vert et bleu, coefficients utilisés dans la fonction `grb2gray` de Matlab.

Le script `exercice_2` affiche les trois colonnes de la matrice `C` sous forme d'images, à l'aide de la fonction `reshape`, et calcule les coefficients de corrélation linéaire et les proportions de contraste des composantes principales grâce à la fonction `correlation_contraste` de l'exercice 1. Observez la proportion de contraste de la première composante principale.

Remarque : La fonction `eig` ne garantit rien sur le signe des vecteurs propres. Comme ce signe est aléatoire (mais constant, pour une même session Matlab), il y a une chance sur deux pour que le contraste de la première composante principale soit inversé, ce qui se traduit par un ciel noir !

Détection du daltonisme par le test de Hishihara

Relancez le script `exercice_1` en remplaçant l'image `autumn.tif` par `image.png`. Le chiffre 2 apparaît nettement dans l'image RVB et dans le canal R, mais ce chiffre est illisible dans les canaux V et B. Cette image fait partie d'un ensemble d'images, appelées *planches de Hishihara*, dont le but est de déceler les anomalies de la vision affectant la perception des couleurs, que l'on désigne globalement sous le terme de *daltonisme*. Statistiquement, environ 8% des garçons sont daltoniens, contre seulement 1% des filles. Or, si vous êtes daltonien, vous devriez percevoir le chiffre 2 dans le canal rouge, mais pas dans l'image RVB.

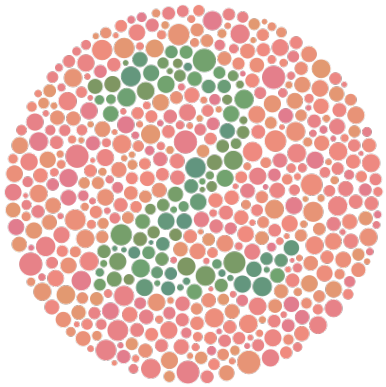

FIGURE 1 – Planche de Hishihara représentant le chiffre 2.

L'explication de ce phénomène est liée à la structure très particulière du nuage de points 3D, en forme de flèche. En relançant le script `exercice_2`, vous observez que le chiffre 2 est très lisible dans la deuxième composante principale, alors qu'il est illisible dans les deux autres composantes principales. Et pourtant, c'est bien la proportion de contraste de la première composante principale qui est la plus élevée (environ 89%). Le contraste d'une image peut donc être élevé sans que cette image soit lisible ! Les planches de Hishihara ont été conçues de telle sorte que le chiffre soit illisible dans la première composante principale, alors que pour un daltonien, tout se passe comme si seule cette composante principale était perceptible. Alors, êtes-vous daltonien ?

Exercice 3 : mise en échec du test de Hishihara

D'où vient que, dans l'exemple précédent, la première composante principale soit la plus contrastée, alors que le chiffre 2 y est illisible ? L'astuce de Hishihara a consisté à afficher des taches de couleur sur fond blanc. C'est donc le fond blanc qui augmente artificiellement le contraste de la première composante principale, alors que ce fond ne contribue aucunement à la lecture du chiffre 2.

Écrivez la fonction `noircir_pixels_blancs`, appelée par le script `exercice_3`, qui détecte les pixels blancs de l'image `image.png`, puis modifie la couleur de ces pixels, de sorte que le chiffre 2 devienne lisible dans la première composante principale. Vous serez alors parvenu à mettre en échec le test de Hishihara, sans avoir pour autant modifié aucun des pixels colorés de `image.png` !