Projet d'optimisation linéaire

Background removal in a video

Deadline : 5 décembre 2025

Étant donné une vidéo, il est souvent utile de pouvoir séparer l'arrière-plan statique de l'avant-plan en mouvement ; voir Figure 1 pour une illustration.

Figure 1: Séparation de l'avant-plan et de l'arrière-plan dans une séquence vidéo.

Supposons que l'on ait une vidéo en noir et blanc représentée par une matrice $X \in \mathbb{R}^{p \times t}$ où p est le nombre de pixels dans chaque image de la séquence vidéo et t le nombre d'images. Ainsi, la j-ème colonne de X, notée $X(:, j)$, est le vecteur contenant l'intensité des pixels de l'image prise au temps j. Pour ce faire, on a vectorisé chaque image de la séquence vidéo en un vecteur ; voir Figure 2 pour une illustration.

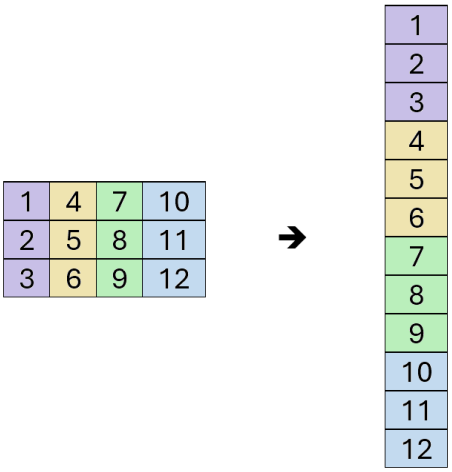

Figure 2: Vectorisation d'une matrice, d'une image vers un vecteur.

La matrice $X \in \mathbb{R}^{p \times t}$ peut être modélisée comme suit

$$X \quad \approx \quad F \quad + \quad B,$$

où F représente l'avant-plan en mouvement (foreground) et B représente l'arrière-plan statique (background) ; comme sur la Figure 1.

Dans le cas d'un arrière-plan statique, chaque colonne de B est répétée, $B(:,j) = b \in \mathbb{R}^p$ pour le même vecteur b et pour tout $j = 1, 2, \ldots, t$. Si on permet une mise à l'échelle (par exemple en raison de différentes illuminations liées au soleil et aux nuages), on préférera utiliser $B(:,j) = a_j b$ pour un scalaire a_j pour chaque j. En d'autres termes, la matrice B est une matrice de rang 1. Si on note $a \in \mathbb{R}^t$, on a $B = ba^\top$.

Remarque 1 (Rang de B)**.** *Dans le cas où l'arrière-plan est également en mouvement (par exemple des feuilles bougeant sous l'effet du vent, un escalator en mouvement, ou les marées à la plage), on va remplacer $B = ba^\top$ par une matrice de rang faible*[1]*.*

La matrice F est une matrice creuse, c'est-à-dire avec un grand nombre d'éléments égaux à zéro, puisque l'avant-plan ne concerne qu'un petit nombre de pixels. Pour minimiser le nombre d'entrées non nulles d'un vecteur, une approche connue est de minimiser la norme 1 de ce vecteur[2] ; pour rappel, la norme 1 du vecteur x est donnée par $\|x\|_1 = \sum_i |x_i|$.

Ainsi, comme $F \approx X - B$, on pourrait résoudre le problème suivant

$$\min_{a \in \mathbb{R}^t, b \in \mathbb{R}^p} \|X - ba^\top\|_1,$$

où $\|X - ba^\top\|_1 = \sum_{i=1}^n \sum_{j=1}^m |X_{i,j} - b_j a_i|$. Ceci n'est pas un problème d'optimisation linéaire.

Estimation de b Pour faciliter notre travail, on suppose d'abord que $a = [1, 1, \ldots, 1]$ (luminosité constante dans les images). La solution optimale pour b est la solution optimale du problème suivant :

$$\min_{b \in \mathbb{R}^p} \sum_{j=1}^{t} \|X(:,j) - b\|_1. \tag{1}$$

Il s'avère que ce problème peut être découplé en p problèmes indépendants : pour $i = 1, 2, \ldots, p$,

$$\min_{b_i \in \mathbb{R}} \sum_{j=1}^{t} |X(i,j) - b_i|. \tag{2}$$

On a donc p problèmes à 1 variable à résoudre.

1. Modélisez le problème (2) comme un problème d'optimisation linéaire. Expliquez votre raisonnement.
2. Écrivez ce problème linéaire sous forme standard.
3. Utilisez le langage de programmation de votre choix pour résoudre ces p problèmes indépendants et ainsi estimer b pour les données fournies (voir ci-dessous). Choix possibles : *Octave* et sa fonction `glpk`, c'est un langage extrêmement similaire à Matlab, excepté qu'Octave est un logiciel libre (=gratuit, voir par exemple `https://www.gnu.org/software/octave/`), *Matlab* avec la fonction `linprog`, ou encore *Python* avec la fonction `linprog` de numpy (voir Moodle pour un exemple). Afficher la solution b, l'arrière-plan (en faisant l'opétation inverse de la vectorisation).
4. La solution obtenue est-elle un sommet du polyèdre correspondant ? Justifiez.

Bonus. Il s'avère que la solution optimale de (1) est la médiane des $X(:,j)$. Pouvez-vous expliquer cela ?

[1] Voir Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. Journal of the ACM (JACM), 58(3), 1-37.

[2] Voir par exemple Candes, E. J., & Tao, T. (2005). Decoding by linear programming. IEEE Transactions on Information Theory, 51(12), 4203-4215, ou Baraniuk, R. G. (2007). Compressive sensing [lecture notes]. IEEE Signal Processing Magazine, 24(4), 118-121.

Estimation de a À partir du b estimé, il faut calculer a_j pour chaque j (illumination optimale) et résoudre t problèmes : pour $j = 1, 2, \ldots, t$

$$\min_{a_j \in \mathbb{R}} \|X(:, j) - a_j b\|_1 = \sum_{i=1}^{p} |X(i, j) - a_j b_i|. \tag{3}$$

Répondez aux mêmes 4 questions pour ce problème (3) que pour le problème (2) ci-dessus.

Résultat final Une fois que a (illuminations) et b (arrière-plan) sont calculés, on peut retrouver l'avant-plan en calculant $F = X - ba^\top$. Affichez quelques images de cet avant-plan (c'est-à-dire des colonnes de F dévectorisées) : avez-vous pu extraire les personnes marchant dans ce parc ? Pouvez-vous créer une vidéo qui montre cet avant-plan en mouvement ?

Données La matrice $X \in \mathbb{R}^{p \times t}$ représente des images comme à la Figure 1, avec $p = 152 \times 232 = 35264$ pixels et $t = 100$ images dans la séquence vidéo. Elle est disponible sur Moodle.

Bonus : Raffinement de la solution

Une fois que a est calculé, on peut recalculer b, et ainsi de suite, pour améliorer la solution de manière itérative. Implémentez cette stratégie : permet-elle d'améliorer la solution en diminuant la valeur de $\|X - ba^\top\|_1$? Et visuellement ?

Consignes

Le travail se réalise par groupe de 2 (*un* groupe de 3 est autorisé si nécessaire). Veuillez fournir avec le rapport les codes implémentés et résultats en annexe. Le tout est à déposer sur Moodle pour le 5 décembre au plus tard.