R5.A.12

Segmentation et classification

19 octobre 2023 – B. COLOMBEL

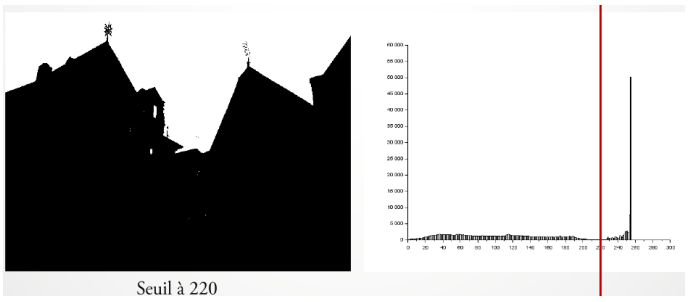
Sommaire

1 Seuillage automatique — La méthode d'Otsu

1.1 Motivations

Réaliser un seuil (**`threshold`** en anglais) d'une image consiste à placer tous les pixels au dessus d'un certain niveau de gris T à la valeur blanc (255) et tous les autres à la valeur 0.

En examinant l'histogramme d'une image, on peut parfois voir un seuil « évident » apparaitre :



Seuil à 220

Mais « évident » ne veut pas toujours dire « interressant » :

Seuil à 205

Une méthode permet de trouver un seuil « sympathique » automatiquement ; elle a été proposé par Obuyuki OTSU en 1979.

1.2 Méthode d'Otsu

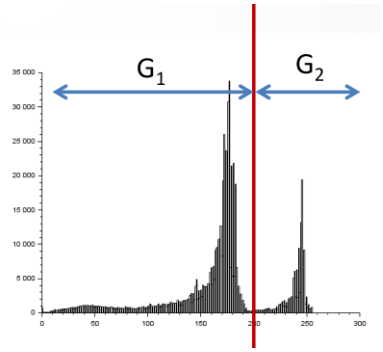
La binarisation au moyen du seuillage combiné prend en paramètre un niveau de seuillage t.

Il existe différentes stratégie afin de déterminer un niveau de seuil automatiquement et ainsi avoir une fonction de seuillage sans paramètre.

La méthode d'OTSU fait partie des méthodes les plus connues pour cela. Elle est basée sur une approche classification : on considère que seuiller l'image à un niveau t revient à classifier les pixels de l'image en 2 classes blanc et noir. On peut alors mesurer la qualité d'un niveau de seuillage par la qualité de la classification qu'il génère. Seuiller l'image de manière automatique revient alors à trouver le niveau de seuil qui génère la meilleure classification des pixels.

Plus précisément. Lorsque l'on choisit un seuil T, on divise l'histogramme H de l'image en deux groupes G_1 et G_2.
On peut voir le problème de deux façons différentes :

1. chercher deux groupes où les pixels « se ressemblent » au sein d'un même groupe, c'est-à-dire **minimiser l'inertie intra-classe** ;
2. chercher deux groupes les plus dissemblables possibles (dont les moyennes sont les plus éloignées), c'est-à-dire **maximiser l'inertie inter-classe**.



Nous avons vu dans la partie *classification*, que si l'on note
— μ_1 la moyenne et V_1 la variance de G_1 ;
— μ_2 la moyenne et V_2 la variance de G_2 ;
— N le nombre de pixels ;
alors

$$I_{\text{intra}} = \frac{|G_1|}{N} V_1 + \frac{|G_2|}{N} V_2 \quad \text{et} \quad I_{\text{inter}} = \frac{|G_1||G_2|}{N^2} (mu_1 - \mu_2)^2$$

De plus, on a

$$V = I_{\text{intra}} + I_{\text{inter}} \quad \text{(valeur fixe)}$$

Ainsi, le problème revient soit à minimiser l'inertie intra-classe ou à maximiser la variance inter-classe.

Il est un peu plus simple de calculer l'inertie inter-classe que l'inertie intra-classe car la première n'est basée que sur les moyennes.

Pour deux classes, on implémentera cette méthode en testant tous les seuils possibles et en retenant celui qui maximise l'inertie inter-classe.

2 Segmentation couleur avec l'algorithmes des k-means

L'algorithme des moyennes mobiles (k-means) peut être utilisé pour regrouper les pixels d'une image couleur, et diminuer au final le nombre de couleurs présentes dans l'image.

Chaque pixel de l'image sera considéré comme un point (à trois coordonnées).
À la fin de l'algorithme, on remplace la couleur de chaque pixel par la couleur, c'est-à-dire ses coordonnées, du centre de gravité de la classe auquel il appartient.

Plus on diminue le nombre de classes, plus on perd de l'information par rapport à l'image d'origine.

3 Travail demandé

Partie 1. Implémenter la méthode de OTSU en testant tous les seuils possibles et en retenant celui qui maximise la variance inter-classe.

La fonction principale prendra en entrée une image en niveau de gris codée sur 8 bits et retournera en sortie le seuil « optimal » selon cette méthode.

Partie 2. L'algorithme des moyennes mobiles (k-means) peut être utilisé pour regrouper les pixels d'une image couleur, et diminuer au final le nombre de couleurs présentes dans l'image.

Implémenter cet algorithme afin de réaliser cette tâche.

La fonction principale prendra en entréer une image couleur ou en niveau de gris et le nombre k de classe recherchée et retournera en sortie l'image composée, après traitement, de k couleurs.

Partie 3. Tester votre algorithme du k-means sur des images noir et blanc. Que donne votre algorithme lorsque vous tentez de trouver deux clusters seulement (est-ce mieux ou moins bien qu'Otsu) ?

Partie 4. Un problème avec l'algorithme du k-means est qu'il faut connaître le nombre de clusters que l'on souhaite obtenir ...

Pour tenter d'évaluer si le nombre de clusters est bon ou pas, on pourrait répéter plusieurs fois l'algorithme et voir si les attributions de groupes sont stables d'une itération à une autre. Si elles ne le sont pas, il y a de fortes chances que le nombre de clusters soit trop bas/élevé.

Tester cette hypothèse sur les images `Mimi1` (à peu près 10-12 couleurs), `Mimi2` (entre 10 et 12 couleurs), `Mimi3` (entre 8 et 11 couleurs) et `Caillou` (entre 8 et 10 couleurs).
Les résultats sont-ils stables lorsque le nombre de clusters est très bas/très grand/proche de la réalité ?
Pouvez-vous proposer une mesure de stabilité permettant de quantifier si un nombre de cluster choisi est pertinent ?

À rendre.

- **Partie 1.** Rendre un programme écrit en `python` qui pourra prendre la forme d'un notebook `Jupyter` ou d'un fichier `python` (extension `.py`).
- **Partie 2.** Rendre un programme écrit en `python` qui pourra prendre la forme d'un notebook `Jupyter` ou d'un fichier `python` (extension `.py`).
- Rendre un fichier `.pdf` présentant votre travail, les résultats obtenus, et toute remarque qui vous paraitrait pertinente.
  En particulier, il devra s'y trouver vos tests / remarques / réponses aux parties 3 et 4 (mais pas que ...).

Votre pdf sera illustré avec les images founies avec le sujet.

Avertissement. Vous avez le droit de discuter et même de regarder le code de vos collègues, mais vous n'avez pas le droit de copier le code. **Le plagiat est une fraude grave**.
L'utilisation [...] d'outils d'intelligence artificielle (comme ChatGPT ou autre) lors de la production de travaux personnels ou de groupe de toute nature, susceptible de faire l'objet d'une évaluation, est considérée comme une fraude passible de poursuites disciplinaires