Projet LI316-2013fev
Traitement d'images par l'algorithme "mean-shift"

D. Béréziat, N. Thome

Version du 25 février 2013

Avertissement : le contenu du sujet peut évoluer au cours du temps (soit pour corriger des erreurs, soit pour expliciter des imprécisions). Les étudiants sont tenus de s'informer des évolutions en allant en cours.

1 Introduction

L'algorithme "mean-shift" désigne une méthode pour calculer rapidement les maxima locaux d'une fonction de densité calculée dans un espace de représentation quelconque. En traitement d'images, cet espace sera typiquement la répartition des niveaux de gris, ou des couleurs (en choisissant un espace adéquat pour représenter ces couleurs), ou des couleurs et des positions des pixels, ..., en fonction du problème de traitement d'image à résoudre. Nous allons voir des exemples concrets. Auparavant, expliquons comment fonctionne cet algorithme.

La première étape consiste à bâtir un estimateur de la fonction de densité d'un échantillon de vecteurs $\mathbf{x}_1, \cdots, \mathbf{x}_n$ à valeurs dans $\mathbb{R}^d$. On rappelle que la fonction de densité donne la probabilité que le vecteur $\mathbf{x}$ puisse se réaliser. L'estimateur de la fonction de densité f est :

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^{n} K(\frac{\mathbf{x} - \mathbf{x}_i}{h}) \tag{1}$$

où $K : \mathbb{R}^d \to [0, 1]$ est une fonction dérivable appelée noyau. Puisque f est une densité, rappelons que $\int_{\mathbb{R}^d} f(\mathbf{x}) d\mathbf{x} = 1$. Finalement la fonction f est représentée par un ensemble de fonction K, centrée sur l'échantillon x_i. Le paramètre h délimite en quelque sorte la zone d'influence du noyau K.

Le choix du noyau K est important, en particulier si le nombre des échantillons est faible, pour bien approximer la fonction f. En effet, si le nombre des échantillons est grand (c'est le cas en image), la loi des grands nombres assure une bonne représentation de f, indépendamment du choix de K. Si le nombre des échantillons devient trop petit, il est assez intuitif que $\hat{f}$ dépendra bien davantage de K. En pratique, les images étant de grande taille, le choix d'un noyau générique tel que la fonction gaussienne ou le noyau "plat" ($K(\mathbf{x}) = 1$ si $\|x\| \leq 1$ et 0 sinon) donne de bons résultats.

La fonction K donnée, on peut calculer les maxima locaux de f en déterminant les zéros de son gradient [1]. Et il est facile de calculer son gradient. À partir de maintenant, nous supposons que le noyau K est radialement symétrique, c'est-à-dire que :

$$K(\mathbf{x}) \propto k(\|\mathbf{x}\|^2)$$

avec $k : \mathbb{R}^+ \to \mathbb{R}^+$ c'est-à-dire qu'il ne dépend que de la norme de $\mathbf{x}$ et pas de sa direction. Calculons le

1. Le gradient de f est par définition : $\nabla f = \left(\frac{\partial f}{\partial x^1}, \cdots, \frac{\partial f}{\partial x^d}\right)$ avec $\mathbf{x} = (x^1, \cdots, x^d)$

gradient de $\hat{f}$:

$$\begin{aligned}\nabla\hat{f}(\mathbf{x}) &= \frac{1}{nh^d}\sum_{i=1}^{n}\nabla K(\frac{\mathbf{x}-\mathbf{x}_i}{h}) \\ &= \frac{1}{nh^d}\sum_{i=1}^{n}\nabla k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right) \\ &= \frac{2}{nh^d}\sum_{i=1}^{n}\frac{\mathbf{x}-\mathbf{x}_i}{h^2}k'\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)\end{aligned}$$

Notons $g_h(\mathbf{x}) = -k'(\|\mathbf{x}/h\|^2)$, l'équation précédente est ré-écrite sous cette forme :

$$\begin{aligned}\nabla\hat{f}(\mathbf{x}) &= \frac{2}{nh^{d+2}}\left[\sum_{i=1}^{n}g_h(\mathbf{x}-\mathbf{x}_i)\right]\left[\frac{\sum_{i=1}^{n}\mathbf{x}_i g_h(\mathbf{x}-\mathbf{x}_i)}{\sum_{i=1}^{n}g_h(\mathbf{x}-\mathbf{x}_i)}-\mathbf{x}\right] \\ &= \frac{2}{nh^{d+2}}\left[\sum_{i=1}^{n}g_h(\mathbf{x}-\mathbf{x}_i)\right]\mathbf{m}_h(\mathbf{x})\end{aligned}$$

Définition 1 *La quantité :*

$$\mathbf{m}_h(\mathbf{x}) = \frac{\sum_{i=1}^{n}\mathbf{x}_i g_h(\mathbf{x}-\mathbf{x}_i)}{\sum_{i=1}^{n}g_h(\mathbf{x}-\mathbf{x}_i)} - \mathbf{x} \tag{2}$$

est appelé "mean-shift".

Mean-shift signifie décalage moyen : en effet, pour un point $\mathbf{x}$ dans l'espace de représentation, il représente bien un décalage par un barycentre sur les points $\mathbf{x}_i$ et pondéré par la fonction g_h.

Le théorème suivant est l'algorithme du mean-shift :

Théorème 1 *Soit un point $\mathbf{x} \in \mathbb{R}^d$ pris dans l'espace de représentation. Alors le schéma numérique suivant :*

$$\mathbf{x}^0 = \mathbf{x} \tag{3}$$

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \mathbf{m}_h(\mathbf{x}^t) \tag{4}$$

converge vers un point $\mathbf{x}^\infty$ dont la densité $\hat{f}(\mathbf{x}^\infty)$ est un maximum local.

Les principaux avantages de cet algorithme sont que, d'une part, on ne fait pas vraiment d'hypothèse sur la nature de l'estimateur de f autrement que par le choix de la fonction k, d'autre part, le nombre de maxima locaux de $\hat{f}$ n'a pas besoin d'être connu à l'avance (contrairement à d'autres méthodes).

Le rôle du noyau

L'action du noyau est très important dans l'algorithme du Mean-Shift. Il permet de lisser plus ou moins la fonction $\hat{f}$ et donc il détermine le nombre de maxima locaux de $\hat{f}$. Puisque l'algorithme Mean-Shift permet, à partir d'un point de l'espace de représentation, de converger vers le maxima le plus proche, le noyau détermine donc le nombre de classes que l'on obtiendra après application de l'algorithme.

Dans la figure 1, nous avons calculé et affiché la répartition des niveaux de gris dans une image naturelle. Cette répartition de niveau représente donc la fonction de densité. En choisissant des noyaux dont le support est de plus en plus grand (voir les courbes à gauche dans la figure 2), on remarque alors que $\hat{f}$ devient de plus en plus lisse et contient de moins en moins de maxima locaux (courbes à droite dans la figure 2). Dans ces figures, les noyaux sont $K(\mathbf{x}) = \exp(-\left\|\frac{\mathbf{x}}{h}\right\|^2)$ avec h égale respectivement à 1, 5 et 10. Il conviendra donc de paramétrer correctement h en fonction de l'application souhaitée.

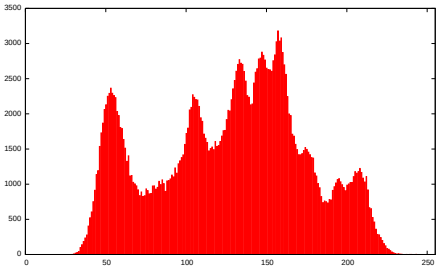
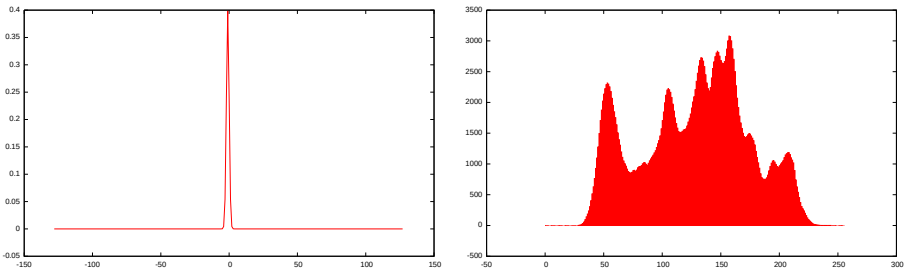
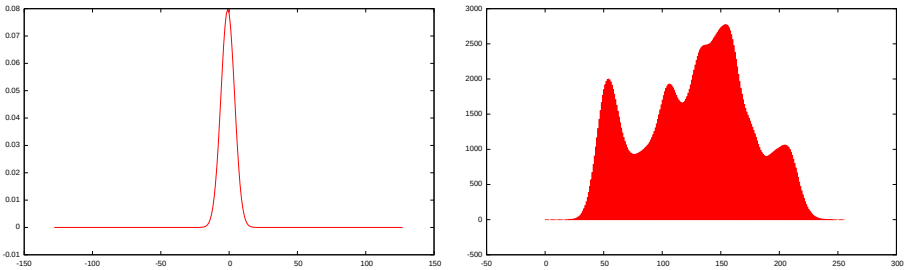


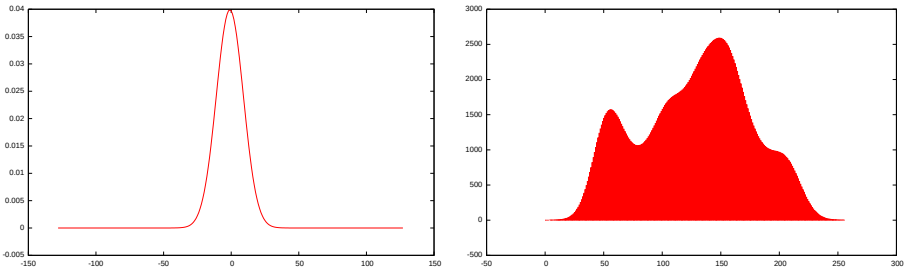
FIGURE 1 – Un histogramme calculé sur une image naturelle (Léna)



(a) $h = 1$ (b) $\hat{f}$, 8 maxima locaux



(c) $h = 1$ (d) $\hat{f}$, 4 maxima locaux



(e) $h = 1$ (f) $\hat{f}$, 2 maxima locaux

FIGURE 2 – Différentes estimations de f en fonction du noyau et de sa taille

Le choix du noyau

Les expressions de k et g_h dépendent du noyau choisi, on donne ici ces expressions dans le cas de deux noyaux particuliers.

Noyau d'Epanechikov

Dans ce cas on a : $k(x) = \begin{cases} 1-x & \text{si } x \in [0,1] \\ 0 & \text{sinon} \end{cases}$ et $K(\mathbf{x}) \propto \begin{cases} 1-||\mathbf{x}||^2 & \text{si } ||\mathbf{x}|| \leq 1 \\ 0 & \text{sinon} \end{cases}$.

On en déduit que $g_h(\mathbf{x}) = -k'(||\frac{\mathbf{x}}{h}||^2) = \begin{cases} 1 & \text{si } ||\mathbf{x}||^2 < h^2 \\ 0 & \text{sinon} \end{cases}$.

La fonction g_h définit un noyau "plat", c'est-à-dire de valeur constante constante dans l'hypersphère de rayon h (noté $S_h(\mathbf{x})$). En reportant dans l'équation 4, on a : $\mathbf{x}^{t+1} = \frac{1}{N_\mathbf{x}} \sum_{\mathbf{x}_i \in S_h(\mathbf{x}^t)} \mathbf{x}_i$ avec $N_\mathbf{x}$ le nombre d'éléments $\mathbf{x}_i$ appartenant à $S_h(\mathbf{x}^t)$. Dans ce cas, l'algorithme du mean-shift revient simplement, à partir d'un point $\mathbf{x}$, à calculer la moyenne des points $\mathbf{x}_i$ dans le voisinage de $\mathbf{x}$ (défini par $S_h(\mathbf{x})$), et à itérer le processus jusqu'à convergence.

Noyau Gaussien

Dans ce cas on a : $k(x) = \exp(-\frac{1}{2}x)$ et $K(\mathbf{x}) \propto \exp(-\frac{1}{2}||\mathbf{x}||^2)$.
On en déduit $g_h(\mathbf{x}) = -k'(||\frac{\mathbf{x}}{h}||^2) = -\frac{1}{2h}\exp(-\frac{1}{2}||\frac{\mathbf{x}}{h}||^2)$.
La fonction g_h est donc également une fonction gaussienne (au signe près). Cette fonction n'est en théorie pas à support fini (elle ne s'annule jamais), mais on peut légitimement approximer ses valeurs à 0 pour $\|\mathbf{x}\| \geq 3h$. Ce noyau ainsi tronqué devient une fonction à support fini, comme le noyau plat. Ceci permet pour le calcul de $\mathbf{m}_h(\mathbf{x})$ de ne considérer que les points qui sont dans le "voisinage" de $\mathbf{x}$, et donc d'accélérer les calculs.

2 L'algorithme du Mean-Shift

2.1 Illustration

La figure 3 illustre le principe du fonctionnement de l'algorithme Mean-Shift dans le cas $d = 1$: tous les niveaux de gris compris entre 0 et 75 sont remplacés par la valeur 55 (la classe la plus probable), et ainsi de suite pour les autres plages de valeurs.

2.2 Espace de représentation

Suivant le cadre applicatif, l'espace de représentation $\mathbb{R}^d$ associé à chaque pixel pourra varier. Dans les applications visées, on considérera des valeurs scalaires (image de niveau de gris, et donc $d = 1$ dans l'équation 1) ou vectorielles (par exemple une image couleur RGB, et dans ce cas $d = 3$).

D'autre part, il est également possible d'incorporer les informations de localisation spatial dans le vecteur de représentation, de sorte que $\mathbf{x} = (\mathbf{x}^s, \mathbf{x}^r)$, où $\mathbf{x}^s$ est la partie du vecteur correspondant aux composantes spatiales (dimension 2 pour une image), et $\mathbf{x}^r$ la partie du vecteur correspondant aux composantes d'apparence (dimension 1 ou 3 pour une image). La dimension du vecteur d'état estimé en chaque pixel est donc $d = \dim(\mathrm{x^s}) + \dim(\mathrm{x^r})$. Le noyau global $K(\mathbf{x})$ est défini comme le produit des noyaux dans les deux domaines, si bien que :

$$K(\mathbf{x}) = k\left(\left\|\frac{\mathbf{x}^s}{h_s}\right\|^2\right) k\left(\left\|\frac{\mathbf{x}^r}{h_r}\right\|^2\right)$$

Le noyau aura donc un support défini par (h_s, h_r).

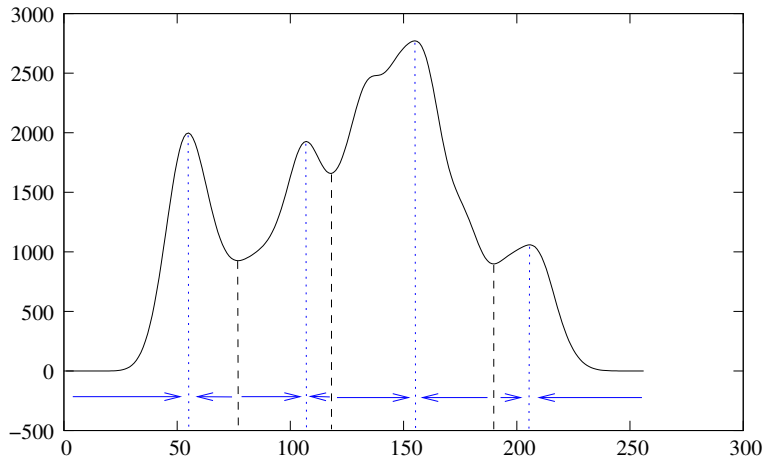


FIGURE 3 – Classification par Mean-Shift pour le cas $d = 1$: chaque valeur est remplacée par la classe la plus proche et la plus probable.

Les étudiants soucieux d'approfondir le sujet peuvent lire cet article [1] qui est disponible sur Internet via une requête sur un moteur de recherche.

2.3 Algorithme du mean-shift

Voici l'algorithme du mean-shift générique :

Algorithme 1: Procédure Mean-Shift

Data : $\mathbf{x}_1, \cdots, \mathbf{x}_n$
Input : $\mathbf{x}$
Input : h, s, $maxIter$ (paramètre)
$k := 0$;
repeat
  $\mathbf{y} := \mathbf{x}$;
  $\mathbf{x} := \frac{\sum_{i=0}^{n} \mathbf{x}_i g_h(\mathbf{y}-\mathbf{x}_i)}{\sum_{i=0}^{n} g_h(\mathbf{y}-\mathbf{x}_i)}$;
until $k > maxIter$ *ou* $\|\mathbf{y} - \mathbf{x}\| < s$;
$\mathbf{y} := \mathbf{x}$;
return $\mathbf{y}$;

La seconde ligne de la boucle Repeat est donc le cœur de l'algorithme mean-shift. La fonction $g_h(\mathbf{x})$ s'écrit donc :

$$g_h(\mathbf{x}) = -k'\left(\left\|\frac{\mathbf{x}^s}{h_s}\right\|^2\right) k'\left(\left\|\frac{\mathbf{x}^r}{h_r}\right\|^2\right)$$

avec $\mathbf{x} = (\mathbf{x}^s, \mathbf{x}^r)$ et $h = (h_s, h_r)$. La complexité dans le cas général d'un tel algorithme est en $n^2 dT$ où T est le nombre moyen d'itérations. C'est donc un algorithme très lourd !

3 Applications en traitement d'images

Cet algorithme peut avoir plusieurs applications pour le traitement des images. **Le but de ce projet est donc d'en implanter quelques unes.**

3.1 Débruitage

Comme nous l'avons vu, l'algorithme du Mean-Shift permet de calculer les maxima locaux de la fonction de densité f. Notamment, connaissant la valeur $\mathbf{x}$ d'un pixel, l'algorithme Mean-Shift permet de calculer la valeur la plus proche de $\mathbf{x}$, $\mathbf{x}'$, dont $\hat{f}(\mathbf{x}')$ est un maxima local.

Dans les applications de débruitage, on cherche à modifier la valeur des pixels bruités par une valeur plus conforme. Qu'est-ce qu'une valeur *plus conforme* dans ce contexte ? C'est une valeur statistiquement plus probable et qui reste du même ordre de grandeur. L'hypothèse sous-jacente consiste à supposer que la valeur du bruit dans l'espace de représentation ne correspondra pas à un maximum local (cette hypothèse est raisonnable pour des bruits blanc gaussien ou uniforme, par exemple), et donc que l'algorithme convergera vers une valeur où le bruit a été supprimé. Donc l'algorithme du mean-shift répond à la question : il suffit d'attribuer au pixel la classe la plus proche et la plus fréquente, celle qui correspond au maximal local le plus proche dans l'espace de représentation.

Algorithme 2: Débruitage d'une image

Input : I image
Output : J image (position et couleur)

$i := 0$;
foreach *pixel* p *dans* I **do**
  $\mathbf{x}_i := (p.x, p.y, I(p))$;
  $i++$;
foreach *pixel* p *dans* I **do**
  $\mathbf{x} := (p.x, p.y, I(p))$;
  `/* On retourne une image qui contient non seulement la couleur filtrée (composante` $\mathbf{x}^r$`) mais aussi la composante spatiale filtrée (`$\mathbf{x}^s$`). */`
  $J[p] := MeanShift(\mathbf{x}_1, \cdots, \mathbf{x}_n, \mathbf{x}, h_s, h_r, s, maxIter)$;

La figure 4 montre un exemple de débruitage d'une image couleur (espace RGB). Considèrons l'image de Léna originale (figure 4(a)) et bruitons-là avec un bruit blanc gaussien (écart-type $\sigma = 0.2 \times 255$), comme illustré à la figure 4(b)). Un résultat de débruitage avec l'algorithme du Mean-Shift est montré à la figure 4(c)).

(a) Image originale (b) Image bruitée (c) Image débruitée ($h_r = 0.05$, $h_s = 8$)

FIGURE 4 – Exemple de débruitage par l'algorithme du Mean-Shift

Pour les images couleurs, il peut être pertinent d'utiliser des espaces de représentation des couleurs autre que RGB. En particulier, un des principaux inconvénient de RGB et que la mesure de distance euclidienne dans cet espace n'est pas toujours corrélée à la mesure perceptuelle de distance entre les couleurs. Il existe

des espaces plus "perceptuels", notamment HUE ou le YUV. La commande Inrimage `cnvcol` permet de passer de RGB à HUE et vis-versa. Pour YUV, il faut se débrouiller soi-même.

3.2 Segmentation

La segmentation consiste à décomposer l'image en régions.
Pour segmenter une image on appliquera donc les opérations suivantes :

1. Pour chaque pixel, appliquer l'algorithme du mean-shift et remplacer la valeur de chaque pixel par la valeur de convergence de l'algorithme.
2. Fusionner les valeurs obtenues qui sont à une distance d'au plus h_s dans la composante spatiale et d'au plus h_r dans la composante couleur h_r en gardant la valeur de densité maximale.

Algorithme 3: Segmentation d'une image

Input : I image
Output : S image

$J := Debruit(I, h_s, h_r, s, maxIter)$;
$His :=$ Histogramme de J;
foreach *pixel* p *dans* I **do**
  $\mathbf{x} := (J[p].\mathbf{x}^s, J[p].\mathbf{x}^p)$;
  foreach *pixel* q *dans* I **do**
    $\mathbf{y} := (J[q].\mathbf{x}^s, J[q].\mathbf{x}^p)$;
    if $\|\mathbf{x}^s - \mathbf{y}^s\| < h_s$ *et* $\|\mathbf{x}^r - \mathbf{y}^r\| < h_r$ **then**
      $S[p] := His[J[p]] > His[J[q]]?J[p].\mathbf{x}^r : J[q].\mathbf{x}^r$;

(a) Image originale (b) Image filtrée ($h_r = 0.02$, $h_s = 8$) (c) Image segmentée ($h_r = 0.02$, $h_s = 8$)

FIGURE 5 – Exemple de segmentation par l'algorithme du Mean-Shift.

Travail à réaliser

Nous suggérons d'abord d'implémenter l'algorithme du mean-shift dans le cas où $\mathbf{x} = (\mathbf{x}^s, \mathbf{x}^r)$ et en supposant des noyaux plats pour les deux composantes (on a donc deux paramètres h_s et h_r). On peut se limiter au cas où $\mathbf{x}^s$ contient la coordonnée d'un pixel dans l'image et $\mathbf{x}^r$ est un triplet RGB décrivant la couleur du pixel dans l'image. Le choix du noyau plat va permettre d'une part de simplifier les calculs et surtout de limiter le temps de calcul puisque **considérer un noyau spatial plat revient simplement**

à appliquer la procédure du Mean-Shift dans le voisinage du pixel à traiter. L'algorithme du mean-shift sera utilisée pour le débruitage et la segmentation.

Dans un second temps, nous pourrons considérer des noyaux gaussiens pour $\mathbf{x}^s$ et $\mathbf{x}^r$. L'algorithme devient alors très lourd et il convient de réfléchir à des optimisations possibles. Nous suggérons par exemple de :

- limiter le calcul du mean-shift dans un voisinage (donc de considérer un noyau spatial plat) ;
- de garder mémoire des valeurs déjà calculées. Mais le nombre couleur est grand ($2^{24} \simeq 16$ millions). Une solution est d'utiliser une table de haschage. La bibliothèque `glib` offre de telles routines (voir `http://developer.gnome.org/glib/2.31/glib-Hash-Tables.html`).

Déroulement du projet

- Le projet compte comme la note de contrôle continu de l'UE.
- Le travail est à réaliser impérativement en binôme.
- Une soutenance est prévue à la séance du dernier TME.
- Les étudiants présenterons, à l'aide d'un ordinateur, leur solutions algorithmiques ainsi qu'une démonstration du code.
- Il n'y aura pas de rapport écrit.
- L'évaluation portera uniquement sur l'oral (la compréhension du sujet) et le code fourni par les étudiants (la quantité de travail fourni).
- Toutes améliorations pertinentes du sujet seront appréciées par les enseignants.

Références

[1] D. Comaniciu and P. Meer. Mean shift : A robust approach toward feature space analysis. *Transaction On Pattern Analysis and Machine Intelligence*, 24(5) :603–669, May 2002.

[2] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *Transaction On Pattern Analysis and Machine Intelligence*, 25(5) :564–577, 2003.