École polytechnique

Traitement massif de données

INF442-5***

Projet informatique : *Détection par "boosting"*

Un détecteur a pour but de localiser dans une image les régions contenant un type particulier d'objet (des visages par exemple). Les détecteurs actuels, dans leur grande majorité, fonctionnent en calculant un vecteur de caractéristiques dans des régions rectangulaires de toute taille et de toute dimension. Sur chaque région, un classifieur est alors appliqué au vecteur pour déterminer la présence ou l'absence de l'objet recherché. Vu le grand nombre de tels rectangles qui doivent être considérés dans une image quelconque, le vecteur de caractéristiques doit être extrait très rapidement. Nous allons ainsi utiliser des caractéristiques introduites par Paul Viola et Michael Jones [3] afin de créer un détecteur générique que nous appliquerons à la détection de visages.

Deux types d'erreurs seront distinguées pour évaluer le classifieur :

- un faux positif : un objet a été détecté alors qu'il n'y en avait pas.
- un faux négatif : un objet était présent mais n'a pas été détecté.

1 Caractéristiques employées

Nous avons besoin de caractéristiques extraites sur des zones rectangulaires permettant des calculs rapides. Nous considérerons ainsi les caractéristiques représentées figure 1 pour un grand ensemble de tailles et de positions dans l'image. Pour chaque figure, il s'agit de faire la somme des valeurs des pixels colorés en blanc moins la somme des valeurs des pixels colorés en noir. Quelque soit les rectangles considérées, la méthode dite de "l'image intégrale" permet un tel calcul en temps constant. En effet, en calculant une carte sur l'image correspondant à la somme de tous les pixels du coin en haut à gauche dans l'image (voir figure 2), il est possible d'en déduire toutes les caractéristiques rectangulaires mentionnées précédemment par un ensemble de soustractions et d'additions (voir figure 3).

1.1 Calcul de l'image intégrale

Soit i une image. L'image intégrale I est défnie de la manière suivante :

$$I(x,y) = \sum_{x'<x, y'<y} i(x',y').$$

Question : Ecrire une fonction prenant en argument une image et renvoyant l'image intégrale. L'algorithme doit parcourir l'image une seule fois, on utilisera pour cela les relations de récurrence suivantes :

$$s(x,y) = s(x,y-1) + i(x,y)$$

.

$$I(x,y) = I(x-1,y) + s(x,y).$$

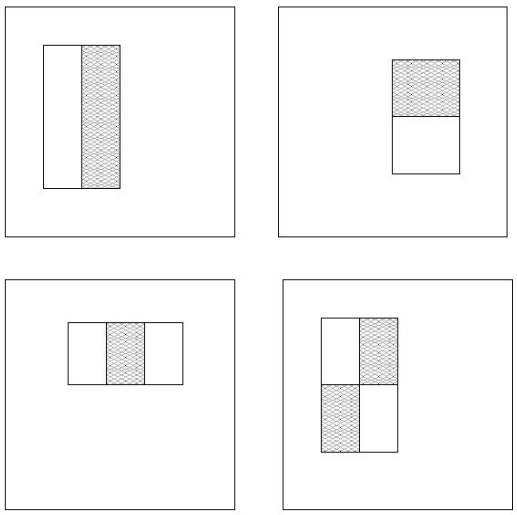

FIGURE 1 – Illustration des caractéristiques utilisées : la somme des pixels présents dans le rectangle blanc sont soustraits de ceux présents dans le rectangles noir. Ces caractéristiques sont utilisées à toutes les tailles et toutes les positions.

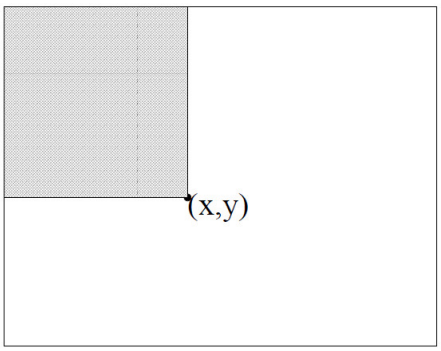

FIGURE 2 – La valeur de l'image intégrale en x, y est égale à la somme des pixels en haut à gauche de ce point.

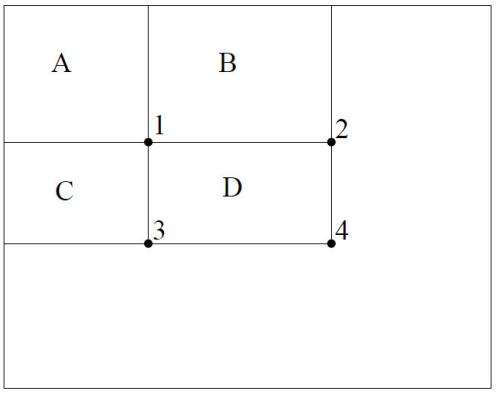

FIGURE 3 – Illustration de l'intérêt de l'image intégrale : la somme des pixels présents dans le rectangle D peut être calculée en temps constant comme la somme de l'image intégrale aux points 4 et 1 dont on soustrait la valeur de l'image intégrale aux points 2 et 3.

1.2 Calcul des caractéristiques

Question : Implémenter de manière parallèle le calcul des caractéristiques introduites précédemment sur une zone de taille 92×112 pour toute taille et toute position dans l'image en imposant au côté d'un rectangle une taille minimale de 4 pixels. On prendra un incrément de position et de taille des carrés de 4 pixels. Commenter brièvement le nombre de caractéristiques ainsi obtenu.

2 Création d'un classifieur

2.1 Création des classifieurs faibles

Chaque composante du vecteur précédemment obtenu va à présent être associée à un classifieur h_i qui est défini par deux paramètres réels w_{i1} et w_{i2}. Une image de vecteur de caractéristiques x, dont on note X_i la i-ème composante, est considérée comme étant associée à un visage par le classifieur h_i si $w_{i1}X_i + w_{i2} \geq 0$, on note alors $h_i(x) = 1$. À l'inverse ce n'est pas un visage si $w_{i1}X_i + w_{i2} < 0$, on note alors $h_i(x) = -1$.

Nous allons entrainer ces classifieurs en utilisant la méthode du perceptron [2] en utilisant la base d'apprentissage contenue dans le répertoire "Train". Ce répertoire contient deux sous-répertoires : un contenant les images correspondant à des images de visages (nommé répertoire "visage"), l'autre (nommé répertoire "nonvisage") contenant des images autres que des visages.

Pour chaque classifieur h_i, il s'agit d'apprendre ces paramètres à partir de la base d'apprentissage avec l'algorithme suivant :

- Initialiser $w_{i1} = 1$ et $w_{i2} = 0$
- Pour k variant de 1 à K :

  On choisit aléatoirement une image dans la base d'apprentissage dont la i-ème composante du vecteur de caractéristiques x_k est noté X_{ki} et dont la classe est notée $c_k \in \{-1, 1\}$. On modifie alors les paramètres de la manière suivante :
  - $w_{i1} = w_{i1} - \epsilon(h_i(x_k) - c_k)X_{ki}$
  - $w_{i2} = w_{i2} - \epsilon(h_i(x_k) - c_k)$

Question : Entrainer de manière parallèlisée les classifieurs en utilisant la méthode du perceptron décrite ci-dessus. Le pas ϵ et le nombre d'itérations K seront choisis de manière à assurer une convergence satisfaisante des paramètres.

2.2 Boosting des classifieurs faibles

Les classifieurs ainsi définis sont dits "faibles" car leurs performances individuelles sont très limitées. Nous allons utiliser l'algorithme Adaboost [1] sur la base de validation (répertoire "Validation" structuré de même façon que précédemment) pour créer un classifieur ayant de bonnes performances de détection en utilisant une combinaison linéaire de ces classifieurs faibles.

Dans l'algorithme qui suit, on note n le nombre d'images de la base de validation. On note x_j et c_j respectivement le vecteur de caractéristiques et la classe de la j-ème image

- Soient $\lambda_{k,1}, \lambda_{k,2}, \ldots, \lambda_{k,n}$ un ensemble de poids, initialement fixés à $\frac{1}{n}$
- Soit E une fonction d'erreur : $E(h(x), c) = 0$ si $h(x) = c$, 1 sinon.
- Soit F_k le classifieur résultat. Initialement, $F_k = 0$
- Pour k allant de 0 à N
  - Choisir le classifieur faible h^k minimisant l'erreur pondérée ϵ_k sur la base de validation : $\epsilon_k = \sum_{j=1}^{n} \lambda_{k,j} E(h^k(x_j), c_j)$
  - On définit $\alpha_k = \frac{1}{2} ln(\frac{1-\epsilon_k}{\epsilon_k})$
  - $F_{k+1} = F_k + \alpha_k h^k$
  - Les poids sont mis à jour :
    - $\forall j, \lambda_{k+1,j} = \lambda_{k,j} \exp(-c_j \alpha_k h^k(x_j))$
    - Renormaliser les poids $\lambda_{k+1,j}$ tels que $\sum_{j=0}^{n} \lambda_{k+1,j} = 1$

Le classifieur final F est défini de la manière suivante : $F(x) = 1$ si $F_N(x) \geq \theta . \sum_{k=1}^{N} \alpha_k$, -1 sinon. θ est un paramètre qui varie entre -1 et 1 qui permet d'ajuster la sensibilité du détecteur.

Question : Implémenter de manière parallélisée l'algorithme ci-dessus. On ajustera le paramètre N de manière à assurer une convergence de l'algorithme satis-

faisante.

3 Test du classifieur final

On cherche dans cette section à évaluer la performance du détecteur entrainé à la section précédente. On utilise pour cela la base de test contenue dans le répertoire "Test" structuré de même façon que précédemment.

1. *Question :* En faisant varier le paramètre θ de -1 à 1, calculer le taux de faux négatifs en fonction du taux de faux positifs. Représentez vos résultats sur un graphe.
2. *Question :* Ajustez votre algorithme pour tester la performance de votre descripteur sur des images quelconques. Commenter le temps de calcul de votre classifieur.

Références

[1] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. 1997.

[2] Frank Rosenblatt. The perceptron : A probabilistic model for information storage and organization in the brain. In *Psychological Review*, volume 65, pages 386–408, 1958.

[3] Paul Viola and Michael Jones. Robust real-time object detection. In *International Journal of Computer Vision*, 2001.