INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN NORMANDIE

Projet TIM: La méthode de Viola et Jones et les caractéristiques de Haar

Léopold Maillard
Mathias Van Audenhove
Lucie Clair

Octobre - Décembre 2020

Table des matières

1 Introduction

La méthode de Viola et Jones est une méthode de reconnaissance de forme très efficace et révolutionnaire à sa parution. Paul Viola et Michael Jones, employés au Cambridge Research Laboratory de la société américaine Compaq, publient la méthode qui porte leur nom pour la première fois en 2001 dans le journal scientifique International "Journal of Computer Vision" [7]. Leur objectif était de résoudre deux gros problèmes que pose la reconnaissance faciale : la vitesse et la robustesse [2].

2 Les éléments de la méthode de Viola et Jones

2.1 Les caractéristiques de Haar

Une des premières optimisations mise en place par Viola et Jones est de travailler non pas directement sur les valeurs des pixels mais sur des caractéristiques [2]. Ces caractéristiques synthétisent l'information apportée par les pixels et la simplifie. Leur utilisation augmente donc significativement la rapidité de l'algorithme. Nous vous présentons ci-dessous quelques exemples de caractéristiques rectangulaires :

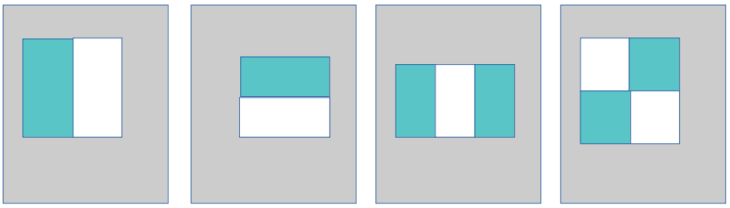

FIGURE 1 – Caractéristiques rectangulaires

Le principe est simple : on soustrait la somme des valeurs des pixels de la zone blanche à celle de la zone bleue. Notons que nous n'avons donné ici que quelques exemples de ces caractéristiques, il en existe énormément.

2.2 Pourquoi utiliser les caractéristiques de Haar, exemple de la détection de visages

Tous les visages humains partagent des propriétés communes, et celles-ci peuvent être retrouvées grâce aux caractéristiques de Haar.

On peut remarquer par exemple que, sur des photos comportant des visages, la région des yeux est usuellement plus sombre que celle des joues supérieures. De façon similaire, la région du nez semble plus lumineuse, celle des sourcils plus sombre.[1]

De simples caractéristiques rectangulaires de Haar comme celles présentées précédemment peuvent ainsi correspondre à ces similitudes :

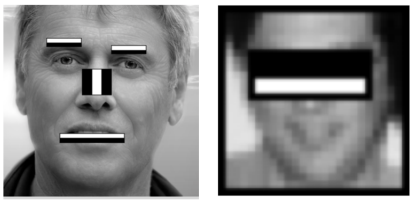

FIGURE 2 – Quelques caractéristiques de Haar correspondant à des spécificités du visage humain [4]

On devine alors qu'il s'agira de sélectionner ces caractéristiques particulièrement pertinentes pour détecter l'objet d'intérêt, mais avant cela des milliers d'entre elles doivent être calculées au sein de l'image.

2.3 Les images intégrales

L'utilisation des caractéristiques vues précédemment entraîne de nombreux calculs de sommes et soustractions. C'est pourquoi Viola et Jones font intervenir les images intégrales.
Pour une position donnée x, y, l'image intégrale équivaut à la somme des valeurs des pixels au dessus et à gauche de x, y inclus [2] :

$$i(x,y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

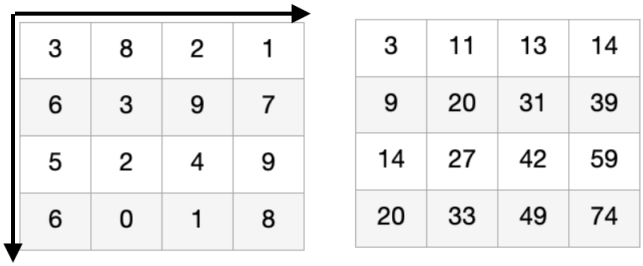

FIGURE 3 – Exemple d'une image 4x4 et son image intégrale.

Tout d'abord,l'image intégrale peut être obtenue d'un seul parcours de l'image originale. Il n'est donc pas coûteux de la calculer. De plus, une fois calculée, nous pouvons obtenir la somme des valeurs des pixels de n'importe quel rectangle de l'image en simplement quatre accès suivant le schéma suivant :

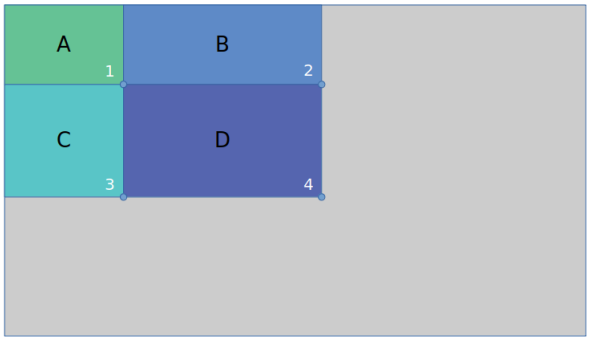

FIGURE 4 – Calcul de somme des pixels de D

En effet, la somme des pixels dans D se calcule de la manière suivante : $4+1-(2+3)$. [2] Les images intégrales simplifient le calcul des caractéristiques de Haar de manière indéniable.

2.4 Classifieur et utilisation d'AdaBoost

Bien que le calcul des caractéristiques soit grandement allégé par le modèle de l'image intégrale, nous allons être confrontés à un nombre trop important de caractéristiques de pseudo-Haar. Viola et Jones ont alors pensé à sélectionner un petit nombre de ces caractéristiques, afin de les combiner et ainsi aboutir à un classifieur efficace [2].

La méthode utilisée pour sélectionner les meilleures caractéristiques est la sélection par boosting. Celle-ci consiste à pondérer des classifieurs "faibles" en fonction de la qualité de leur classification, pour ensuite les combiner et aboutir à un classifieur "fort", plus robuste. Le principe général de ces algorithmes de boosting est présenté dans l'illustration ci-dessous :

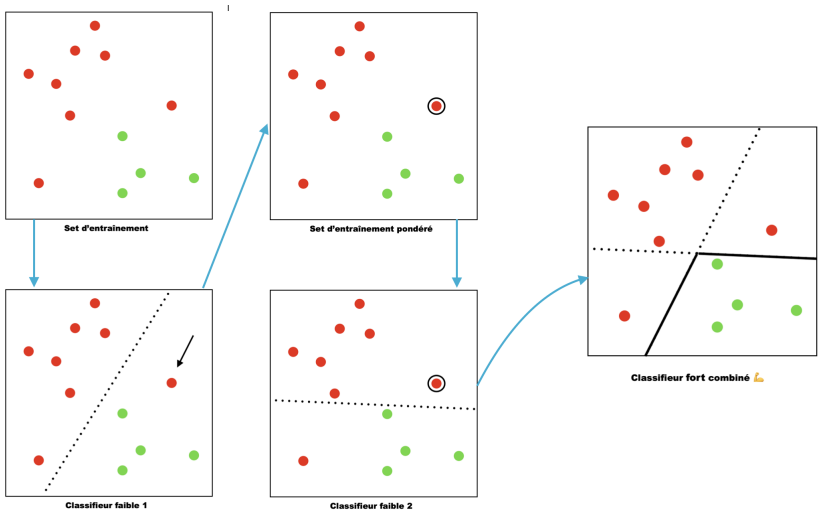

FIGURE 5 – Schéma du fonctionnement d'Adaboost

Plus particulièrement, Viola et Jones ont eu l'idée d'assimiler chaque caractéristique à un classifieur faible. Il s'agit ainsi, pour chaque caractéristique, de déterminer la fonction de seuillage permettant de classifier correctement le plus d'exemples d'apprentissage possible, c'est à dire de déterminer le seuil à partir duquel on classe les exemples comme étant positifs ou négatifs [8]. On retiendra finalement la caractéristique rectangulaire produisant les meilleurs résultats, donc celle pour laquelle le taux d'erreur de classification est le plus faible.

L'algorithme de boosting effectuera ainsi plusieurs fois cette opération, et ajoutera la caractéristique retenue à chaque itération à la liste des caractéristiques sélectionnées aux itérations précédentes, constituant ainsi le classifieur fort [7].

Viola et Jones s'appuient en particulier sur l'algorithme AdaBoost qui se veut adaptatif [6]. En effet, les caractéristiques sont choisies en tenant compte des caractéristiques sélectionnées précédemment. L'algorithme fera ainsi en sorte de sélectionner des caractéristiques permettant de classifier correctement des exemples jusqu'alors mal classés par le classifieur courant. Adaboost permet donc d'obtenir un classifieur fort par une somme pondérée des classifieurs faibles sélectionnés.

Le boosting permet ainsi de sélectionner une petite quantité de caractéristiques pseudo-Haar parmi les dizaines de milliers existantes, celles qui sont les plus significatives pour la reconnaissance visuelle. On améliore ainsi le temps d'exécution et la qualité de la prédiction, un nombre considérable de caractéristiques peu pertinentes n'étant pas utilisées.

2.5 La structure en cascade

Le processus consistant à tester la présence de l'objet recherché dans une fenêtre, à chacune des positions possibles au sein de l'image, et ce pour différentes échelles de taille, est très coûteux.

Pour remédier à ce problème, on définit une cascade de classifieurs, un ordre prédéfini de classifieurs à appliquer pour optimiser l'algorithme. Pour qu'un objet soit détecté dans une certaine fenêtre, elle doit être acceptée par chacun des classifieurs. On teste une fenêtre avec le classifieur N si et seulement si le classifieur N-1 n'a pas rejeté la fenêtre, et on répète cette action jusqu'à avoir atteint un rejet ou bien la fin de la cascade, auquel cas la fenêtre est acceptée.

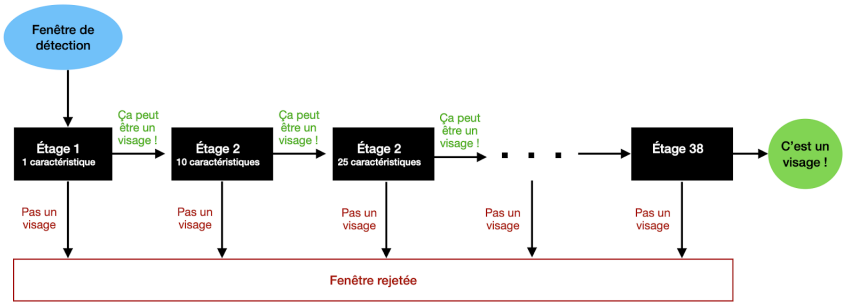

FIGURE 6 – Schéma de la structure en cascade

Le premier classifieur appliqué à une zone va dans l'immense majorité des cas rejeter la présence de l'objet, puisque la probabilité que l'objet soit contenu dans une fenêtre prise au hasard est faible. C'est pourquoi il est avantageux de choisir comme premier classifieur un classifieur simple, capable de rejeter la majorité des fenêtres pour un moindre coût. Les classifieurs les plus simples sont donc situés au début de la cascade. Si une fenêtre n'a été rejetée par aucun des premiers classifieurs, alors elle a des chances non négligeables d'effectivement contenir l'objet recherché. On lui applique alors les classifieurs les plus forts, les plus coûteux, qui sont situés en fin de cascade.

L'un des facteurs qui peut influencer le résultat est la taille de cette cascade de classifieurs. Plus une cascade est courte, plus il y aura de faux positifs, puisque ses critères de sélection sont alors trop simples. Au contraire, une cascade trop longue pourrait avoir des critères de sélection trop sévères, et donc rejeter des fenêtres qui pourtant s'avèrent contenir l'objet recherché. Pour donner un ordre d'idées, Viola et Jones utilisent dans leur méthode originale 38 étages.

L'objectif est alors de trouver la cascade de classifieurs qui convient à nos exigences, c'est-à-dire ayant un taux de détection de l'objet supérieur ou égal à celui désiré, ainsi qu'un taux de faux positifs inférieur ou égal à celui désiré.

3 Les grandes étapes de la méthode

3.1 L'étape d'apprentissage

L'objectif de l'étape d'apprentissage est d'obtenir un classifieur en cascade permettant de reconnaître efficacement un objet d'intérêt. Pour cela, on commence par calculer les caractéristiques de pseudo Haar de milliers d'exemples positifs, qui correspondent à l'objet que l'on cherche à reconnaître, et négatifs.

FIGURE 7 – Des exemples d'apprentissage positifs et négatifs pour la reconnaissance de visages [5].

Une fois les caractéristiques calculées, il faudra entraîner la cascade. Chaque étage de la cascade est constitué d'un classifieur fort obtenu par la sélection successive des meilleures caractéristiques par Boosting, jusqu'à ce que le classifieur de l'étage ait les performances souhaitées [7].

3.2 L'étape de détection

Une fois notre cascade de classifieurs obtenue, on pourra l'appliquer sur des images tests afin d'y détecter la présence éventuelle d'un objet d'intérêt. Ainsi, l'image est parcourue par une fenêtre de détection appliquée :

— À toutes les positions, car l'objet d'intérêt peut être n'importe où dans l'image.
— À toutes les échelles, car l'objet peut être de toutes tailles (il peut occuper toute l'image ou qu'une petite partie).

Pour chaque application de la fenêtre de détection :

— On calcule les caractéristiques de Haar qui sont utilisées par le classifieur du premier étage de la cascade
— On calcule la réponse du classifieur : si elle est positive, il se peut que la fenêtre comporte l'objet, on passe à l'étage suivant. Si elle est négative, on passe à la fenêtre suivante.

— Si la réponse est positive à chaque étage, cela signifie que la fenêtre comporte l'objet à détecter.

La fenêtre de détection étant appliquée partout sur l'image, il se peut qu'un même objet soit détecté plusieurs fois. Une dernière étape consiste donc à fusionner les détections qui correspondent au même objet.

4 Ce que nous avons implémenté

4.1 Le calcul de l'image intégrale

L'implémentation de l'ensemble de la méthode n'étant pas de notre niveau nous avons pris la décision de n'implémenter qu'une petite partie. Ainsi, nous avons pu nous assurer de notre bonne compréhension du sujet.

Nous avons donc commencé par coder la fonction de calcul de l'image intégrale, indispensable pour calculer efficacement les caractéristiques de Haar au sein d'une image.

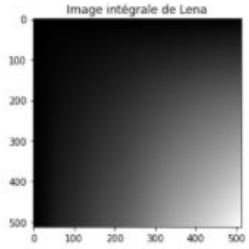

FIGURE 8 – Image intégrale obtenue après implémentation

4.2 L'extraction des caractéristiques de Haar

On a vu que lors de l'étape d'apprentissage, on calculait les caractéristiques de Haar de milliers d'exemples. On a donc implémenté l'extraction des caractéristiques rectangulaires de 5 types différents :

FIGURE 9 – Les 5 types de caractéristiques rectangulaires extraites [5]

Ces caractéristiques sont calculées à toutes les positions et pour toutes les longueurs et largeurs possibles. Pour une image de 24x24 pixels (la dimension des exemples utilisés par Viola et Jones dans leur méthode [2]), on obtient ainsi 43200 caractéristiques de type (a), 27600 de type (b), 43200 de type (c), 27600 de type (d) et 20736 de type (e). Cela fait au total pas moins de 162336 caractéristiques différentes.

4.3 L'application d'une caractéristique

Nous avons implémenté l'application d'une caractéristique particulière à tous les points d'une image comportant un visage. L'idée était de faire correspondre la zone des yeux à une caractéristique rectangulaire.

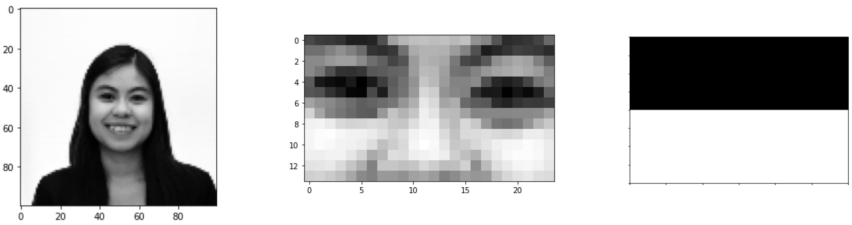

FIGURE 10 – Image utilisée pour l'application, un zoom sur la zone des yeux et la caractéristique utilisée pour faire la correspondance.

Après application de la caractéristique à tous les points de l'image, le résultat obtenu présente des zones plus claires que d'autres : elles correspondent à la soustraction de la somme des pixels d'une zone plus sombre à celle d'une zone plus claire de l'image. On observe bien cela aux coordonnées des yeux (avec un décalage haut-gauche).

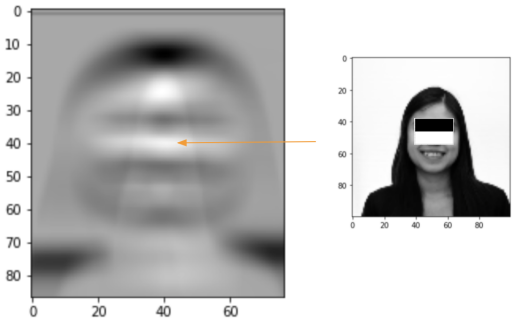

FIGURE 11 – Résultat de l'application de la caractéristique et correspondance avec la zone des yeux.

Cette application est également très intéressante car elle permet de bien comprendre l'intérêt de la cascade. On voit en appliquant une seule caractéristique qu'il ne sera pas utile de s'attarder sur certaines zones de l'image (le fond par exemple) qui ne renvoie rien de particulier.

5 Application : Détection de visages

Les applications de cette méthode sont nombreuses de part sa robustesse et son efficacité à reconnaître des objets. Nous avons fait le choix de nous intéresser plus particulièrement à la reconnaissance de visages.

N'ayant pas implémenté l'ensemble de la méthode, nous avons utilisé les modèles déjà entraînés proposés par OpenCV [3]. Cela nous a permis d'observer l'efficacité ainsi que les limites de la méthode.

5.1 Détection sur une image figée

Dans un premier temps, nous avons testé le modèle pré-entrainé sur une image figée.

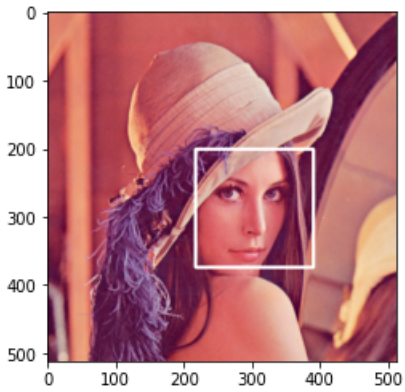

FIGURE 12 – Détection du visage de Lena

Le visage est bien détecté.

5.2 Détection de visage sur une vidéo

Un des objectifs de Viola et Jones était de proposer une solution en temps réel, nous avons donc testé le modèle sur une vidéo. En effet, la reconnaissance de visages est assez rapide pour que nous ne nous apercevions pas du temps de calcul. La vidéo reste fluide. Cependant, on observe que les visages ne sont

pas toujours bien reconnus et que parfois des visages sont détectés alors qu'il n'y en a pas. De nombreux facteurs peuvent expliquer cela et ne remettent pas forcément en cause la structure de la méthode de Viola et Jones. Nous avons utilisé le modèle pré-entrainé d'OpenCV [3]. Certains choix ont été faits durant son implémentation comme le nombre d'étage de cascade par exemple. Peut être que ces choix ne sont pas adaptés à l'utilisation que nous faisons du modèle.

FIGURE 13 – Détection de visage en temps réel

5.3 Détection de visage en temps réel et réalité augmentée

Pour aller plus loin, nous nous sommes essayés à la réalité augmentée. Notre objectif était de pouvoir ajouter un filtre type "Snapchat" (oreilles de chien) en temps réel sur la vidéo prise par la webcam de l'ordinateur. N'ayant aucune connaissance, notre collage du filtre sur l'image obtenue par la webcam est restée assez expérimentale. En effet, nous ajoutons chaque pixel des oreilles un par un au bon endroit au dessus du visage. Cela fonctionne dans le sens où les oreilles sont bien placées et suivent le visage mais cela prend beaucoup de temps. La vidéo n'est pas du tout fluide. Notre méthode n'est bien entendu pas optimale.

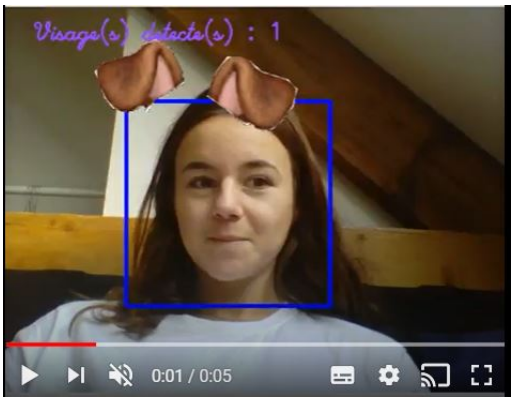

FIGURE 14 – Tentative de réalité augmentée

Références

[1] Computerphile. *Detecting Faces (Viola Jones Algorithm)*. URL : https://www.youtube.com/watch?v=uEJ71VlUmMQt=36s. (accessed: 12.10.2020).

[2] Paul Viola et Michael Jones. "Rapid Object Detection using a Boosted Cascade of Simple Features". In : *Journal of Computer Vision* (2001). DOI : http://bit.ly/CRapidObjectDetectPa-per.

[3] Documentation OpenCV. *Cascade Classifier*. URL : https://docs.opencv.org/3.4/db/d28/tutorialcascade%20classifier.html. (accessed: 03.10.2020).

[4] Global Software Support. *Computer Vision - Haar-Features*. URL : https://www.youtube.com/watch?v=F5rysk51txQ. (accessed: 10.11.2020).

[5] Yi-Qing Wang. "An Analysis of the Viola-Jones Face Detection Algorithm". In : (2013). DOI : http://www.ipol.im/pub/art/2014/104/article.pdf.

[6] Contributeurs de Wikipedia. *AdaBoost*. URL : https://fr.wikipedia.org/wiki/AdaBoost. (accessed: 28.10.2020).

[7] Contributeurs de Wikipedia. *Méthode de Viola et Jones*. URL : https://fr.wikipedia.org/wiki/%E1%B8%BEethodede%20Viola%20et%20Jones. (accessed: 12.10.2020).

[8] Contributeurs de Wikipedia. *Viola–Jones object detection framework*. URL : https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework. (accessed: 28.10.2020).